R-miss-tastic: a unified platform for missing values methods and workflows

by Imke Mayer, Aude Sportisse, Julie Josse, Nicholas Tierney and Nathalie Vialaneix

Abstract Missing values are unavoidable when working with data. Their occurrence is exacerbated as more data from different sources become available. However, most statistical models and visualization methods require complete data, and improper handling of missing data results in information loss or biased analyses. Since the seminal work of Rubin (1976), a burgeoning literature on missing values has arisen, with heterogeneous aims and motivations. This led to the development of various methods, formalizations, and tools. For practitioners, however, it remains a challenge to decide which method is most appropriate for their problem, in part because this topic is not systematically covered in statistics or data science curricula. To help address this challenge, we have launched the 'R-miss-tastic' platform, which aims to provide an overview of standard missing values problems, methods, and relevant implementations of methodologies. Beyond gathering and organizing a large majority of the material on missing data (bibliography, courses, tutorials, implementations), 'R-miss-tastic' covers the development of standardized analysis workflows. Indeed, we have developed several pipelines in R and Python to allow for hands-on illustration of and recommendations on missing values handling in various statistical tasks such as matrix completion, estimation, and prediction, while ensuring reproducibility of the analyses. Finally, the platform is dedicated to users who analyze incomplete data, researchers who want to compare their methods and search for an up-to-date bibliography, and teachers who are looking for didactic materials (notebooks, recordings, lecture notes).

1 Context and motivation

Missing data are unavoidable as soon as collecting or acquiring data is involved. They occur for many reasons including: individuals choosing not to answer survey questions, measurement devices failing, or data having simply not been recorded. Their presence becomes even more important as data are now obtained at increasing velocity and volume, and from heterogeneous sources not originally designed to be analyzed together. As pointed out by Zhu et al. (2019), "one of the ironies of working with Big Data is that missing data play an ever more significant role, and often present serious difficulties for analysis". Despite this, the approach most commonly implemented by default in software is to toss out cases with missing values. At best, this is inefficient because it wastes information from the partially observed cases. At worst, it results in biased estimates, particularly when the distributions of the missing values are systematically different from those of the observed values (e.g., Enders, 2010, Chap. 2).

However, handling missing data in a more efficient and relevant way (than limiting the analysis to solely the complete cases) has attracted a lot of attention in the literature in the last two decades. In particular, a number of reference books have been published (Schafer and Graham, 2002; van Buuren, 2018; Carpenter and Kenward, 2012; Little and Rubin, 2019) and the topic is an active field of research (Josse and Reiter, 2018). The diversity of the missing data problems means there is great variety in the proposed and studied methods. They include model-based approaches, integrating likelihoods or posterior distributions over missing values, filling in missing values in a realistic way with single, or multiple imputations, or weighting of observations, appealing to ideas from the design-based literature in survey sampling. The multiplicity of the available solutions makes sense because there is no single solution or tool to manage missing data: the appropriate methodology to handle them depends on many features, such as the objective of the analysis, the type of data, the type of missing data and their pattern. Some of these methods are available in various software solutions. As R is one of the main pieces of software for statisticians and data scientists, with its development starting almost three decades ago (Ihaka, 1998), R offers the largest number of implemented approaches. This is also due to its ease in incorporating new methods and its modular packaging system. Currently, there are over 270 R packages on CRAN that mention missing data or imputation in their DESCRIPTION files. These packages serve many different applications, data types or types of analysis. More precisely, exploratory and visualization tools for missing data are available in packages like **naniar**, **VIM**, and **MissingDataGUI** (Tierney et al., 2021; Tierney and Cook, 2018; Kowarik and Templ, 2016; Cheng et al., 2015). Imputation methods are included in packages like **mice**, **Amelia**, and **mi** (van Buuren and Groothuis-Oudshoorn, 2011; Honaker et al., 2011; Gelman and Hill, 2011). Other packages focus on dealing with complex, heterogeneous (categorical, quantitative, ordinal variables) data or with large dimensional multi-level data, such as **missMDA**, and **MixedDataImpute** (Josse et al., 2016; Murray and Reiter, 2015). Besides R, other languages such as Python (Van Rossum and Drake, 2009), which currently only have few publicly available implementations of methods that handle missing values,

offer increasingly more solutions. Two prominent examples are: 1) the **scikit-learn** library (Pedregosa et al., 2011) which has recently added a module for missing values imputation; and 2) the **DataWig** library (Biessmann et al., 2018) which provides a framework to learn to impute incomplete data tables.

Despite the large range of options, missing data are often not handled appropriately by practitioners. This may be for several reasons. First, the plethora of options can be a double-edged sword; while it is great to have many options, finding the most appropriate method is challenging as there are so many. Second, the topic of missing data is often missing itself from many statistics and data science syllabuses, despite its omnipresence in data. So, when faced with missing data, practitioners are left powerless; quite possibly never having been taught about missing data, they do not know how to approach the problem, the dangers of mismanagement, how to navigate the methods, software, or how to choose the most appropriate method or workflow.

To help promote better management and understanding of missing data, we have released 'R-miss-tastic', an open platform for missing values. The platform takes the form of a reference website[1], which collects, organizes and produces material on missing data. It has been conceived by an infrastructure steering committee working group (ISC; its members are authors of this article), which first provided a CRAN Task View[2] on missing data[3] that lists and organizes existing R packages on the topic. The 'R-miss-tastic' platform extends and builds on the CRAN Task View by collecting, creating and organizing articles, tutorials, documentation, and workflows for analyses with missing data.

This platform is easily extendable and well documented, allowing it to seamlessly incorporate future works and research in missing values. The intent of the platform is to foster a welcoming community, within and beyond the R community. 'R-miss-tastic' has been designed to be accessible for a wide audience with different levels of prior knowledge, needs, and questions. This includes students, teachers, statisticians, and researchers. Students can use its content as complementary course material. Teachers can use it as a reference website for their own classes. Statisticians and researchers can find example analysis workflows, or even contribute information for specific areas and find collaborators.

The platform provides new tutorials, examples and pipelines of analyses that we have developed with missing data spanning the entirety of an analysis. These have been developed in R and in Python, implementing standard methods for generating missing values, and for analyzing them under different perspectives. In addition, we reference publicly available datasets that are commonly used as benchmarks for new missing values methodologies. The developed pipelines cover the entirety of a data analysis: exploratory analyses, establishing statistical and machine learning models, analysis diagnostics, and finally interpreting results obtained from incomplete data. We hope these pipelines also serve as a guide when choosing a method to handle missing values.

The remainder of the article is organized as follows: In the section entitled "Structure and content of the platform" we describe the different components of the platform, the structure that has been chosen, and the target audience. The section is organized as the platform itself, starting by describing materials for less advanced users then materials for researchers and finally resources for practical implementation. We then detail the implementation and use-cases of the provided R and Python workflows in the following section entitled "Details of the missing values workflows". Finally, in the conclusion, we outline an overview of planed future developments for the platform and interesting areas in missing values research that we would like to bring to a wider audience.

2 Structure and content of the platform

The 'R-miss-tastic' platform is released at https://rmisstastic.netlify.com/. It has been developed using the R package **blogdown** (Xie et al., 2017) which generates static websites using Hugo[4]. Live examples have been included using the tool https://rdrr.io/snippets/ provided by the website 'R Package Documentation'. The source code and materials of the platform have been made publicly available on GitHub at https://github.com/R-miss-tastic, which provides a transparent record of the platform's development, and facilitates contributions from the community.

We now discuss the structure of the 'R-miss-tastic' platform, the aim and content of each subsection, and highlight key features of the platform.

[1] https://rmisstastic.netlify.com/
[2] https://CRAN.R-project.org/package=ctv
[3] https://cran.r-project.org/web/views/MissingData.html
[4] https://gohugo.io/

Missing values workflows

An important contribution and novelty of this work is the proposal of several workflows that allow for a hands-on illustration of classical analyses with missing values, both on simulated data and on publicly available real-world data. These workflows are provided both in R and in Python code and cover the following topics:

- *How to generate missing values?* Generate missing values under different mechanisms, on complete or incomplete datasets. This is useful when performing simulations to compare methods that impute or handle missing data.
- *How to do statistical inference with missing values?* In particular, we focus on different solutions for estimating linear and logistic regression parameters with missing covariate values (maximum likelihood or multiple imputation).
- *How to impute missing values?* We compare different single imputation/matrix completion methods (e.g., using conditional models, low-rank models, etc.).
- *How to predict with missing values?* We consider building predictive models, e.g., using random forests (Breiman, 2001), on data with incomplete predictors. The workflows present different strategies to deal with missing values in the covariates both in the training set and in the test set.

The aim of these workflows is threefold: 1) they provide a practical implementation of concepts and methods discussed in the lectures and bibliography sections of the platform; 2) they are implemented in a generic way, allowing for re-use on other datasets, for integration of other estimation or imputation methods; 3) the distinction between inference, imputation, and prediction lets the user keep in mind the solutions are not the same.

Furthermore, the workflows allow for a transparent and open discussion about the proposed implementations, which can be followed on the project GitHub repository, referencing proposals and discussions about practicable extensions of the workflows.

Additionally, a workflow on *How to do causal inference with incomplete covariates/attributes in R?* demonstrates simple weighting and doubly robust estimators for treatment effect estimation using R. This workflow is based on the R implementation of the methodology proposed by Mayer et al. (2020).

We provide a more detailed view on the proposed workflows in a later section, with examples of tabular or graphical outputs that can be obtained as well as recommendations on how to interpret and leverage these outputs.

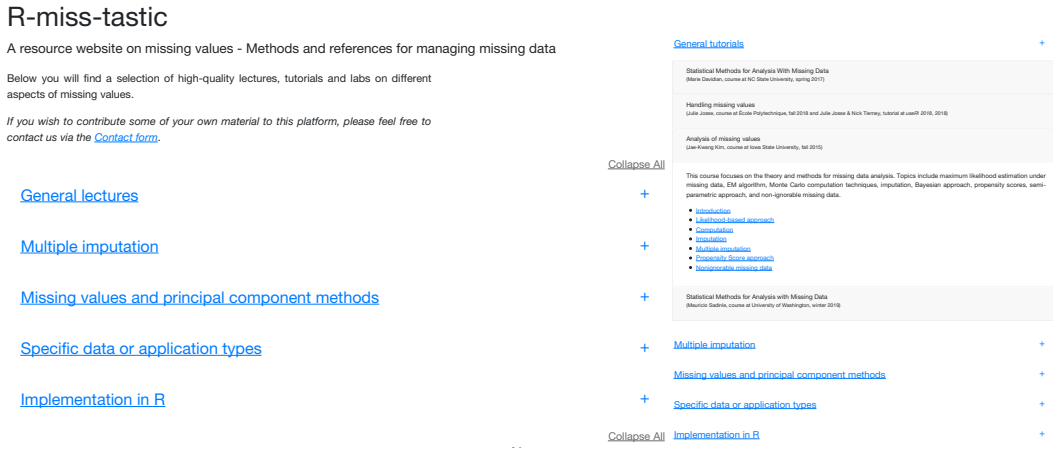
Missing values lectures

For someone unfamiliar with missing data, it is a challenge to know where to begin the journey of understanding them, and the methods to handle them. This challenge is addressed with '`R-miss-tastic`', which makes the material to get started easily accessible.

Teaching and workshop material takes many forms – slides, course notes, lab workshops, video tutorials and in-depth seminars. The material is of high quality, and has been generously contributed by numerous renowned researchers who investigate the problems of missing values, many of whom are professors having designed introductory and advanced classes for statistical analyses with missing data. This makes the material on the '`R-miss-tastic`' platform well suited for both beginners and more experienced users.

These teaching and workshop materials are described as "lectures", and are organized into five sections:

1. General lectures: Introduction to statistical analyses with missing values; the role of visualization and exploratory data analysis for understanding missingness and guiding its handling; theory and concepts are covered, such as missing values mechanisms, likelihood methods, and imputation.
2. Multiple imputation: Introduction to popular methods of multiple imputation (joint modeling and fully conditional), how to correctly perform multiple imputation and limits of imputation methods.
3. Principal component methods: Introduction to methods exploiting low-rank type structures in the data for visualization, imputation and estimation.
4. Specific data or applications types: Lectures covering in details various sub-problems such as missing values in *time series*, in *surveys*, or in treatment effect estimation (*causal inference*). Indeed, certain data types require adaptations of standard missing values methods (e.g., handling time dependence in time series (Moritz and Bartz-Beielstein, 2017)) or additional assumptions about



(a) Collapsed view **(b)** Extract

Figure 1: Overview of selected lectures available on the platform (shared by various researchers working on missing values). The lectures are grouped by topic. Each lecture contains slides or lecture notes and in certain cases also comes with corresponding R tutorials and lecture recordings.

the impact of missing values (such as the impact on confounded treatment effects in causal inference (Mayer et al., 2020)). More in-depth material, e.g., video recordings from a virtual workshop on *Missing Data Challenges in Computation, Statistics and Applications*[5] held in 2020, is also available.

5. Implementations: A non-exhaustive list of detailed vignettes describing functionalities of R packages and of Python modules that implement some of the statistical analysis methods covered in the other lectures. For example, the functionalities and possible applications of the **missMDA** R package are presented in a brief summary, allowing the reader to compare the main differences between this package and the **mice** package which is also summarized using the same summary format.

Figure 1 illustrates two views of the lectures page: Figure 1(a) shows a collapsed view presenting the different topics, Figure 1(b) shows an example of the expanded view of one topic (General tutorials), with a detailed description of one of the lectures (obtained by clicking on its title), '`Analysis of missing values`' by Jae-Kwang Kim. Each lecture can contain several documents (as is the case for this one) and is briefly described by a header presenting its purpose.

Lectures that we found very complete and thus highly recommend are:

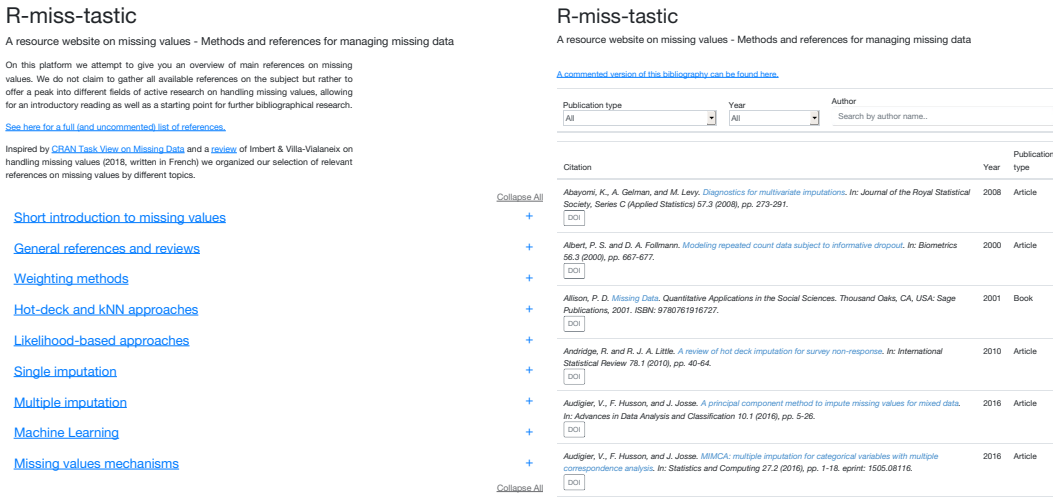
- *Statistical Methods for Analysis with Missing Data* by Mauricio Sadinle (in '`General tutorials`');
- *Missing Values in Clinical Research – Multiple Imputation* by Nicole Erler (in '`Multiple imputation`');
- *Handling missing values in PCA and MCA* by François Husson. (in '`Missing values and principal component methods`');
- *Modern use of Shared Parameter Models for Dropout (in longitudinal and time-to-event data)* by Dimitris Rizopoulos (in '`Specific data or application types`').

The purpose of these lectures is to provide either an introduction or a deeper understanding of the statistical problems and proposed solutions in terms of their (mathematical) derivation and theoretical scope. So there is less of a focus on practical demonstrations with real data, or a systematic comparison of all methods for the same problems. This is covered in the section presenting in detail the developed workflows.

References on missing values

Complementary to the *Lectures* section, this part of the platform serves as a broad overview on the scientific literature discussing missing values taxonomies and mechanisms and statistical, machine learning methods to handle them. This overview covers both classical references to books, articles, etc. such as Schafer and Graham (2002); van Buuren (2018); Carpenter and Kenward (2012); Little and Rubin (2019) and more recent developments such as Josse et al. (2019); Gondara and Wang (2018),

[5] `https://www.ias.edu/math/mdccsa`



(a) Curated version **(b)** Alphabetical version

Figure 2: Overview of curated (a) and alphabetical (b) bibliographies on missing values problems and methods.

which study the consistency of supervised learning with missing values. The entire (non-exhaustive) bibliography can be browsed in two ways: 1) a complete list, filtered by publication type and year, with a search option for the authors or, 2) a curated version. For 2), we classified the references into several domains of research or application, briefly discussing important aspects of each domain. This dual representation is shown in Figure 2 and allows for an extensive search in the existing literature, while providing some guidance for those focused on a specific topic. All references are also collected in a unique BibTex file made available in the GitHub repository[6]. This shared file allows external users to easily propose additions to the bibliography, which are then reviewed by the platform committee, composed of researchers with different focuses on missing values.

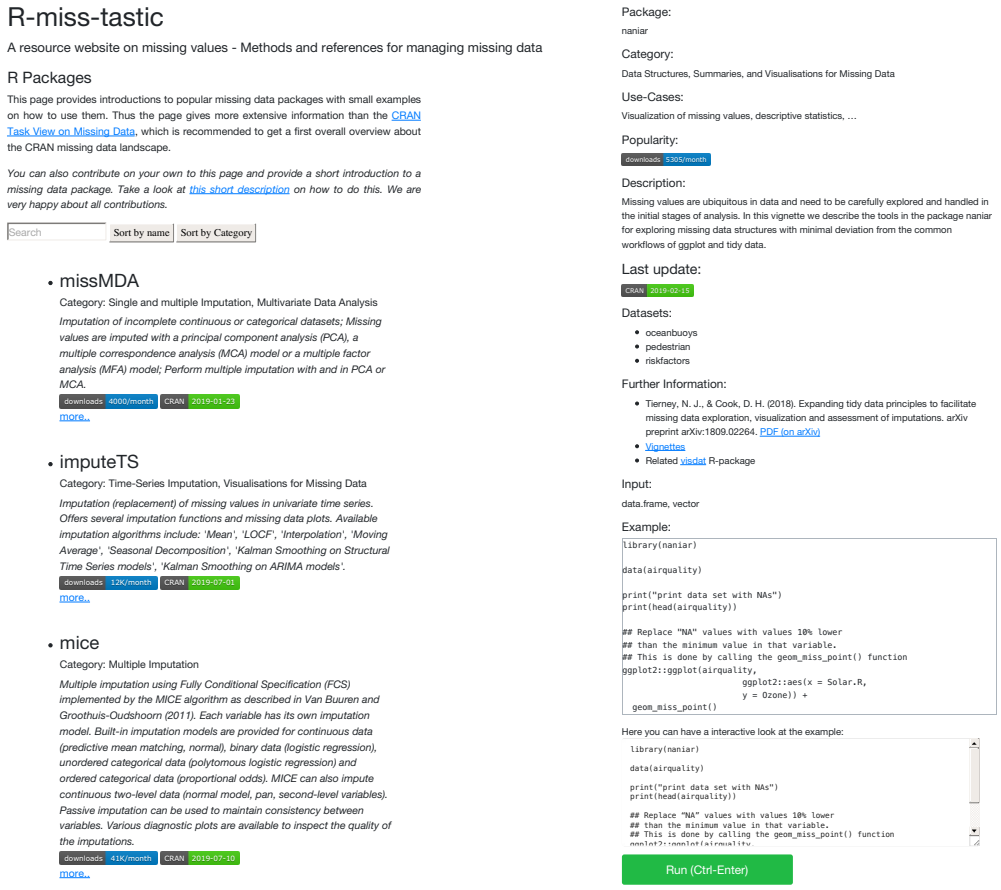
Missing values implementations

R packages As mentioned in the introduction, the platform development is based on the release of the *MissingData* CRAN Task View, which currently lists approximately 150 R packages. The CRAN Task View is continuously updated, adding new, and removing obsolete R packages. Packages are organized by topic: *exploration of missing data*, *likelihood based approaches*, *single imputation*, *multiple imputation*, *weighting methods*, *specific types of data*, and *specific application fields*. We selected only sufficiently mature and stable packages already published on CRAN or Bioconductor. This ensures all listed packages can easily be installed and used by practitioners.

Even though the Task View classifies packages into different sub-domains, it may still be a challenge for practitioners and researchers inexperienced with missing values to choose the most relevant package for their application. To address this challenge, we provide a partial and slightly more detailed overview of existing R packages, selecting the most popular and versatile ones. This overview is a blend of the Task View, and of the individual package description pages and vignettes as provided on CRAN or Bioconductor. For each selected package (7 at the date of writing of this article: **imputeTS**, **mice**, **missForest**, **missMDA**, **naniar**, **simputation** and **VIM**), we provide a category (in the style of the categorization in the Task View), a short description of use-cases, its description (as on CRAN), the usual CRAN statistics (number of monthly downloads, last update), the handled data formats (e.g., `data.frame`, `matrix`, `vector`), a list of implemented algorithms (e.g., k-means, PCA, decision tree), the list of available datasets, some references (such as articles and books), and a small chunk of code, ready for a direct execution on the platform via the *R package Documentation*[7]. Figure 3 shows the condensed view of the package page and the expanded description sheet of a given package (here **naniar**).

We believe shortlisting R packages is highly useful for practitioners new to the field, as it demonstrates data analysis that handles missing values in the data. We are aware that this selection is subjective, and we welcome external suggestions for other packages to add to this shortlist.

[6] in `resources/rmisstastic_biblio.bib`
[7] https://rdrr.io/snippets/



(a) Extract of global view

(b) Description sheet

Figure 3: Overview of selected R packages, described by scope, related data and publications, as well as a small code example.

Python modules To the best of our knowledge, very few methods are already implemented for handling missing data in Python. However, one of the major libraries for machine learning and data analysis, `scikit-learn` (Pedregosa et al., 2011) has recently proposed a module for simple and multiple imputations, `sklearn.impute`. Also, as an alternative, the `statsmodels`[8] library also has an implementation module for multiple imputation in Python now. Additionally, the `missingno` toolset (Bilogur, 2018) facilitates visualizing missing values missing values for exploratory data analyses. We regularly survey new Python implementations for handling missing values and, if pertinent, from a theoretical and practical point of view, reference them on our platform. We expect this to promote their use but also additional assessment by practitioners and researchers from the missing values (statistics/machine learning) community.

Datasets

Especially in methodology research, an important aspect is the comparison of different methods to assess their respective strengths and weaknesses. Several datasets are recurrent in the missing values literature but have not been referenced together yet. We gathered publicly available datasets that have recurrently been used for comparison or illustration purposes in publications, R packages and tutorials. Most of these datasets are already included in R packages but some are available in other data collections. Figure 4 shows how the datasets are presented, with a detailed description shown for one of the dataset ('`Ozone`', obtained by clicking on its name). The description follows the UCI Machine Learning Repository presentation (Dua and Graff, 2019), including a short description of the dataset, how to obtain it, external references describing the dataset in more details, and links to tutorials/lectures on our websites or to vignettes in R packages that use the dataset.

In addition, the *Datasets* section also references existing methods for generating missing data, given assumptions on their generation mechanisms (as in the R package **mice**).

Note, however, that the list of datasets gathered here is short compared to benchmark datasets for full data methods such as the UCI Machine Learning Repository. Therefore, our proposed list also serves as an invitation to tackle this lack of a wider variety of common benchmark datasets in the missing data community.

Additional content

This unified platform collects and edits the contributions of numerous individuals who have investigated missing values problems, and developed methods to handle them. To provide an overview of some of the main actors in this field, the list of all contributors who agreed to appear on this platform is given with links to their personal or to their research lab website.

We also provide links to other interesting websites or working groups, not necessarily working with R and Python (Van Rossum and Drake, 2009) but with other programming languages such as SAS/STAT® and STATA (StataCorp., 2019).

Two other features are finally provided to engage the community:

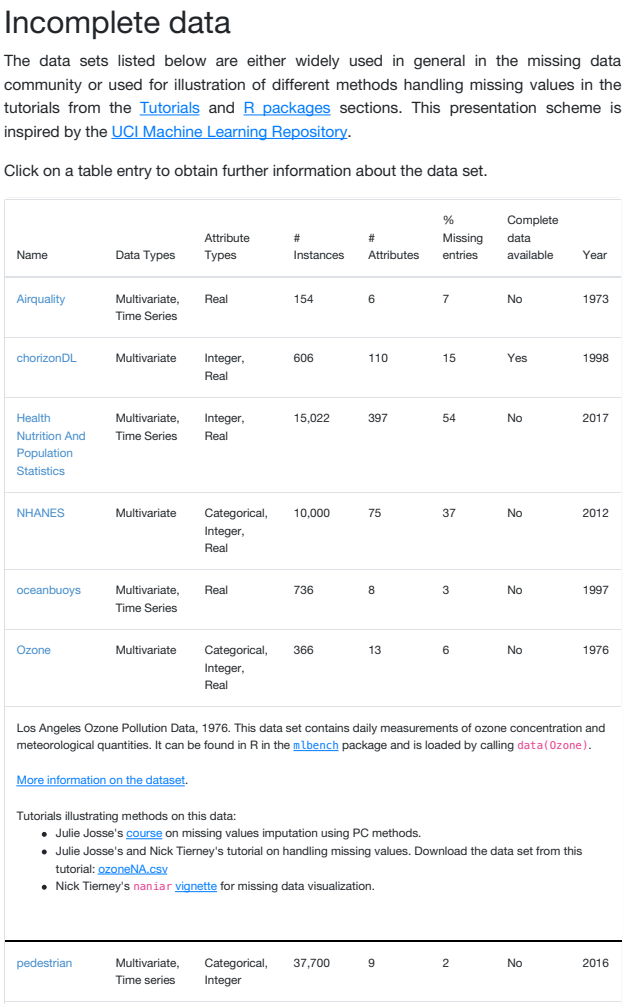
1. A regularly updated list of events such as conferences or summer schools with special focus on missing values problems, and
2. A list of recurring questions together with short answers and links for more details for every question (FAQ).

3 Details of missing values workflows

After this general introduction to the '`R-miss-tastic`' project and platform and the overview of its structure, we now turn to a more detailed presentation of the various workflows that we have developed and proposed on this platform.

To allow for both hands-on tutorials illustrating current practice and state-of-the-art and ready-to-use pipelines, we propose the workflows under different formats such as HTML, PDF, R Markdown (for R code) and IPython Notebook (for Python code). We encourage practitioners and researchers to use and adapt these workflows and propose modifications and improvements, in order to increase reproducibility and comparability of their work. Of course, we are aware that these workflows do not cover the entire spectrum of existing methods and data problems. The goal of the proposed workflows is rather to initiate a joint effort to create a larger spectrum of open-source workflows, and to encourage the use of standardized procedures to handle missing values. With an incomplete dataset

[8] https://www.statsmodels.org/stable/about.html



Incomplete data

The data sets listed below are either widely used in general in the missing data community or used for illustration of different methods handling missing values in the tutorials from the Tutorials and R packages sections. This presentation scheme is inspired by the UCI Machine Learning Repository.

Click on a table entry to obtain further information about the data set.

Name	Data Types	Attribute Types	# Instances	# Attributes	% Missing entries	Complete data available	Year
Airquality	Multivariate, Time Series	Real	154	6	7	No	1973
chorizonDL	Multivariate	Integer, Real	606	110	15	Yes	1998
Health Nutrition And Population Statistics	Multivariate, Time Series	Integer, Real	15,022	397	54	No	2017
NHANES	Multivariate	Categorical, Integer, Real	10,000	75	37	No	2012
oceanbuoys	Multivariate, Time Series	Real	736	8	3	No	1997
Ozone	Multivariate	Categorical, Integer, Real	366	13	6	No	1976

Los Angeles Ozone Pollution Data, 1976. This data set contains daily measurements of ozone concentration and meteorological quantities. It can be found in R in the `mlbench` package and is loaded by calling `data(Ozone)`.

More information on the dataset.

Tutorials illustrating methods on this data:

- Julie Josse's course on missing values imputation using PC methods.
- Julie Josse's and Nick Tierney's tutorial on handling missing values. Download the data set from this tutorial: ozoneNA.csv
- Nick Tierney's `naniar` vignette for missing data visualization.

pedestrian	Multivariate, Time series	Categorical, Integer	37,700	9	2	No	2016

Figure 4: Overview of included datasets, described by key attributes and data availability.

at hand, prior to embarking on an in-depth statistical analysis, two preliminary steps are essential: (i) a descriptive analysis leveraging visualization packages such as **VIM** (Kowarik and Templ, 2016) or **naniar** (Tierney et al., 2021); (ii) a specific aim has to be defined in order to choose a specific method to use.

An example of a method whose success crucially depends on the analyst's goal is *mean imputation*: this approach is strongly counter-indicated if the aim is to estimate parameters, but it can be consistent if the aim is to predict as well as possible (Josse et al., 2019). Following this observation, our workflows are divided into different parts, defined by the objective of the statistical analysis. We aim to present and compare the main implementations available both in R and Python for each objective. Currently there are seven workflows available on the platform and we briefly present their scope and use below. For details on the implementations we encourage the reader to open the corresponding workflows, all available on the 'R-miss-tastic' platform.

How to generate missing values?

The goal of these workflows is to propose functions to generate missing values under different mechanisms. This code aims to unify classical solutions to do this. Indeed, a usual strategy to compare imputation or estimation strategies is to introduce (additional) missing values in the dataset, and use the ground truth for these missing values to evaluate the strategies (see the following section).

Rubin (1976) classifies the cause for a lack of data into three missing data mechanisms. The missing data mechanism is said to be: (i) missing completely at random (MCAR) if the lack of data is totally independent of the data values, (ii) missing at random (MAR) if the process that causes the missing data only depends on the observed values and (iii) missing not at random (MNAR) if the unavailability of the data depends on the missing variables. See Sportisse (2021) for a recent overview on the topic.

In R In the R workflow[9], we have implemented the main function `produce_NA`[10] which facilitates generating missing values under the three missing data mechanisms outlined above. This function internally calls the `ampute` function from the **mice** package (van Buuren and Groothuis-Oudshoorn, 2011) but we chose to simplify its use while adding some additional options to specify the missing values generation. In addition, the original `ampute` function generates missing values only for a complete dataset with quantitative variables[11]. In the main function of our workflow, the user can easily introduce (additional) missing values in a complete or incomplete dataset composed of quantitative, categorical, or mixed variables, by choosing the mechanism and the percentage of missing values to be introduced. The function then returns the data matrix containing the new dataset with missing values, that also includes the missing values already present in the input data, and the indicator matrix (a binary matrix where an entry is equal to 1 if a new missing value has been generated at the same location in the data matrix and 0 otherwise).

The three main arguments are the initial dataset (`data`) in which the missing values are introduced using a given missing data mechanism (`mechanism`) and a given percentage of missing values (`perc.missing`). For example, to introduce 20% of MCAR values in the dataset X, the code is detailed below.

```
X.miss.mcar <- produce_NA(data = X, mechanism = "MCAR", perc.missing = 0.2)
X.mcar <- X.miss.mcar['data.incomp']
R.mcar <- X.miss.mcar['idx_newNA']
```

For example, if X contains three variables (fully observed) denoted as X_1, X_2, X_3, two options are available to generate MAR values:

1. The first option consists in generating missing values in X_1 by using a logistic model depending on (X_2, X_3), which are fully observed, i.e.,

$$\mathbb{P}(R_1 = 0|X;\phi) = 1/(1+\exp(-(\phi_2 X_2 + \phi_3 X_3)), \tag{1}$$

   where $\phi = (\phi_2, \phi_3)$ is the parameter of the missing data mechanism. In our function, ϕ is chosen such that the given percentage of missing values is achieved. This allows us to obtain missing values in the first variable X_1^{NA}. Then, the same strategy is performed to introduce missing values in X_2 and X_3, by using a logistic model depending on (X_1, X_3) (fully observed) and

[9] https://rmisstastic.netlify.app/how-to/generate/misssimul

[10] https://rmisstastic.netlify.app/how-to/generate/amputation.R

[11] If qualitative variables have previously been transformed by one-hot-encoding, they can also be handled by the `ampute` function of **mice**. The `produce_NA` function internally handles the transformation of qualitative variables prior to amputation.

(X_1, X_2) (fully observed) respectively. To get the final matrix containing missing values, we concatenate X_1^{NA}, X_2^{NA} and X_3^{NA} by handling the rows containing only missing values.

2. The second option consists in generating the missing values *by pattern*, i.e., by rows. In this case, the combinations of which variables are observed and missing are specified in a pattern matrix. For the MAR mechanism, in each pattern, at least one variable must be observed. An example (the choice by default) of such a pattern matrix is

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix},$$

where 0 indicates that the variable should have missing values whereas 1 means that it should be observed. For example, the first pattern means that the process which causes the missingness of the first variable X_1 depends on the values of X_2 and X_3 which are observed.

We also propose several ways to generate missing values, under the MNAR mechanism. It includes the most general one when the missingness depends on both the missing variables and the observed variables. It also includes the self-masked mechanism, where the unavailability of the data only depends on their values themselves. For example, it is possible to introduce self-masked missing values using a quantile censorship for which the form is specified by the argument `self.mask`, e.g., if set to `'lower'`, then the values are amputed based on a quantile from the lower tail of the empirical distribution such that the target proportion of missing values is achieved.

In Python To our knowledge, there is no specific module in Python to generate missing values. Consequently, we implemented such functions, in a Python workflow, which similarly to its R counterpart workflow[12] allows us to generate missing values under by different mechanisms and different percentage of missing values.[13] The key difference with the R workflow is that the dataset must be complete and can currently only contain quantitative variables. For MAR and MNAR mechanisms, only the option *not by pattern* has been implemented. In this case, for a dataset X with three variables, a variable is chosen to be fully observed (say X_3), and the process which causes the missingness of two other variables (X_1 and X_2) depends on the values of the fully observed variable, for example with the logistic model given in (1).

How to impute missing values?

The aim of these workflows (in R and Python) is to compare the most classical imputation methods and to propose a reference pipeline for comparison on simulated and real datasets, which can be easily extended with other imputation methods. Here, the imputation methods are considered as such, i.e., the objective is not to estimate a parameter or to perform a statistical analysis on a completed dataset but to impute missing values to get a complete dataset in the best possible way. Therefore, we evaluate the methods in terms of imputation quality, by using the mean squared error (MSE). More precisely, the procedure is the following one: (i) We have access to a complete dataset X, (ii) missing values are introduced in X and we get an incomplete dataset X^{NA}, (iii) this incomplete dataset is imputed and we obtain an imputed dataset X^{imp}, (iv) the MSE, which measures the error committed by the imputation of the missing values, is computed: It is the ℓ_2-norm of the difference of the imputed dataset and the complete one). Note that this procedure can also be performed on an incomplete dataset by introducing additional missing values. However, for now, both R and Python workflows only consider complete datasets.

Different types of imputation methods are included in this workflow:

1. imputation by the mean, which serves as a naive baseline.
2. conditional models, if the imputation relies on the conditional expectation or a draw from the conditional distribution of each variable given the others.
   - in R:
     - **mice** (van Buuren and Groothuis-Oudshoorn, 2011): a multiple imputations method by chained equations. Even if it is a returns several imputed datasets, they can be aggregated using the mean of the imputations to get a single imputation.
     - **missForest** (Stekhoven and Bühlmann, 2012): imputes iteratively by training random forests.

[12] https://rmisstastic.netlify.app/how-to/python/generate_html/how%20to%20generate%20missing%20values

[13] The code has been partially developed in collaboration with Boris Muzellec (Inria Paris).

- in Python:
  - `IterativeImputer` of scikit-learn library (Pedregosa et al., 2011): this function is inspired by mice, but it uses (iterative) regularized regression, imputing by the conditional expectation, and providing a simple imputation. We also use the `ExtraTreesRegressor` estimator of `IterativeImputer`, which trains iterative random forests (it is similar to **missForest** in R).

3. low-rank based models, the data matrix to impute is assumed to be generated as a low rank structure plus a noise term.
   - in R:
     - **softImpute** (Hastie et al., 2015): minimizes the re-weighted least squares error penalized by the nuclear norm.
     - **missMDA** (Josse et al., 2016): minimizes the re-weighted least squares error penalized by a mix between the ℓ_2-norm and ℓ_0-norm.
   - in Python: **softImpute** (coded for the purpose of this notebook and available here[14]), which minimizes the re-weighted least squares error penalized by the nuclear norm and with an internal cross-validation step to choose the regularization parameter.
4. machine learning methods (for the Python workflow only) using optimal transport or variational autoencoders.
   - in Python:
     - `MIWAE` (Mattei and Frellsen, 2019): imputes missing values with a deep latent variable model based on importance weighted variational inference.
     - `Sinkhorn` (Muzellec et al., 2020): randomly extracts several batches and minimizes optimal transport distances between batches to impute missing values.

For the sake of clarity, we show a comparison table (Table 1) in the Appendix, showing the difference of scope between R and Python packages used in the R-miss-tastic workflows.

In R This workflow[15] provides two main functions which compares the imputation methods: (i) on a simulated dataset for different mechanisms and percentage of missing values (`how_to_impute`) or (ii) on a list of real datasets and a given mechanism and percentage of missing values (`how_to_impute_real`).

The function `how_to_impute` takes as input a complete dataset (`X`), a list of percentages of missing values (`perc.list`) and a list of missing data mechanisms (`mecha.list`). The code to use this function is given below.

```
perc.list <- c(0.1, 0.3, 0.5)
mecha.list <- c("MCAR", "MAR", "MNAR")
res <- how_to_impute(X = X, perc.list = perc.list, mecha.list = mecha.list, nbsim = 10)
```

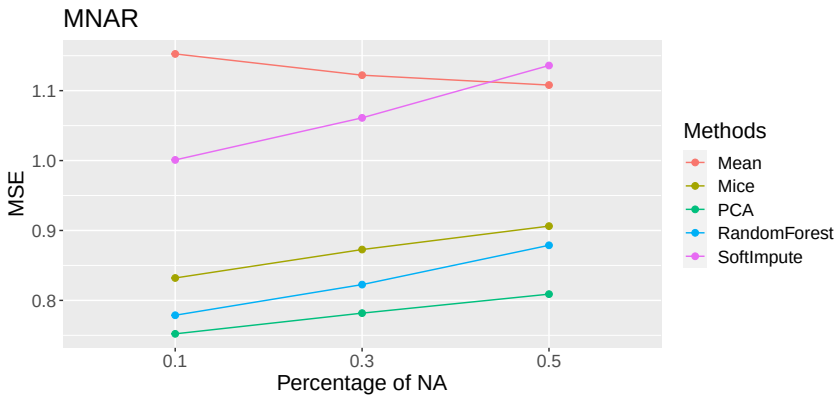
The output of the first function `how_to_impute` is the mean of the methods' MSEs for the different missing values settings by taking the average over several repetitions (the number of repetitions can be specified through the argument `nbsim`). Figure 5 shows the output of this function and its associated plot, when the simulated dataset is Gaussian with $n = 1000$ observations, $d = 10$ variables, a mean vector such that $\mu_i = 1$, $\forall i \in \{1, \ldots, d\}$ and a covariance matrix such that $\Sigma_{ij} = 0.5$ if $i \neq j \in \{1, \ldots, d\}$, and $\Sigma_{ij} = 1$ if $i = j$. First, the mean of the methods' MSEs for the different missing values settings are reported in Figure 5a. We can note that for the MCAR mechanism, the methods perform well, while for the MNAR mechanism, the results are generally closer to those of the naive imputation by the mean. As expected, most methods give worse results for high percentages of missing values. Besides, Figure 5b shows one of the associated plot for MCAR data (there is also a plot for MAR data and a plot for MNAR data). In the first part of the Appendix, this function is illustrated for a particular dataset and the code to obtain Figure 5 is given.

The second function `how_to_impute_real` takes as input a list of datasets (`datasets_list`), a list of missing data mechanisms (`mech`) and a given percentage of missing values (`perc`). It returns a table containing the mean of the MSEs for the simulations performed and a table for the summary plot shown in Figure 6. This can be particularly useful for practitioners who would like to have an indication of which method might be the most suited for a given or for several specific datasets. Here, the real datasets are taken from the UCI repository (Dua and Graff, 2019). An example of how to use this function in practice is detailed below.

[14] https://github.com/R-miss-tastic/website/blob/master/static/how-to/python/softimpute.py
[15] https://rmisstastic.netlify.app/how-to/impute/missimp

	0.1 MCAR	0.3 MCAR	0.5 MCAR	0.1 MAR	0.3 MAR	0.5 MAR	0.1 MNAR	0.3 MNAR	0.5 MNAR
X.pca	0.74	0.76	0.78	0.75	0.78	0.81	0.76	0.78	0.81
X.forest	0.77	0.8	0.86	0.78	0.81	0.87	0.78	0.81	0.88
X.mice	0.82	0.83	0.86	0.83	0.86	0.9	0.84	0.87	0.9
X.soft	0.93	0.86	0.87	0.97	1	1.1	1	1.1	1.1
X.mean	1	0.99	1	1.1	1.1	1.1	1.2	1.1	1.1

(a) Output of the function `how_to_impute` in R. The results for the MSE are truncated to two digits. Note that the line `X.pca` is the result for **missMDA**. For all methods, the default parameter choices are used.



(b) Example of plot for the MNAR mechanism.

Figure 5: Tabular and graphical outputs of the R function `how_to_impute`. The methods **mice**, **missForest**, **softImpute** and **missMDA** are compared with the naive imputation by the mean for several percentages of missing values (10%, 30%, 50%). The mean of the MSEs computed for several generations of missing values are given. In the tabular, the results are shown for several mechanisms (MCAR, MAR, MNAR) and the plot corresponds to the MNAR mechanism.

	winequality–white	winequality–red	slump	movement	decathlon
X.pca	0.92	0.9	0.9	0.47	0.98
X.forest	0.71	0.63	0.72	0.19	1
X.mice	0.86	0.76	0.66	0.095	1
X.soft	0.84	0.78	0.68	0.2	0.94
X.mean	1	1	1	1	1

Figure 6: Output of the R function `how_to_impute_real`. The results for the MSE are truncated to two digits. The methods **mice**, **missForest**, **softImpute** and **missMDA** for several real datasets in which 20% MCAR missing values have been introduced.

```
datasets_list <- list(wine_white = wine_white, wine_red = wine_red, slump = slump,
                      movement = movement, decathlon = decathlon)
names_dataset <- c("winequality-white", "winequality-red", "slump", "movement", "decathlon")
perc <- 0.2
mecha <- "MCAR"
res <-  how_to_impute_real(datasets_list = datasets_list ,perc = perc, mech = mecha,
                                  nbsim = 10, names_dataset = names_dataset)
```

An additional workflow[16] is available and compares other deep-learning imputation strategies to most classical ones on datasets simulated either with linear relationships and nonlinear relationships. The conclusions point to better behavior of the low-rank based imputation methods even when deep-learning methods are tuned.

In Python The Python workflow is very similar to its R counterpart. The two same functions, `how_to_impute` and `how_to_impute_real`, have been implemented.

[16]This workflow has been implemented by an external contributor, François Husson (Professor in Statistics, France).

How to estimate parameters with missing values in R?

This R workflow[17] is dedicated to a specific inferential framework when the aim is to estimate linear and logistic regression parameters for multivariate normal data. It is currently only available in R, as there are no analogous implementations available in Python to our knowledge.

There are two main methods to estimate parameters with missing values: maximum likelihood estimation adapted to missing values, using, e.g., EM-based algorithms or using multiple imputation. In this workflow, we compare two instances of these main methods, using available R implementations: the EM algorithm for logistic and linear regressions with the package **misaem** (Jiang et al., 2020) which uses the Stochastic Approximation of EM algorithm (SAEM Delyon et al., 1999) and multiple imputation with the package **mice**. Both strategies are valid under the MAR missing data mechanism. The workflow performs the estimation on a simulated dataset, but the dataset can be replaced with any custom dataset that the user believes satisfies the assumptions about the missing data mechanism and distribution of covariates.

The **misaem** package facilitates estimation of parameters of linear and logistic regression models from incomplete data, and also provides valid estimates of these parameters' variances. The functions `miss.lm`, `miss.glm` resemble the standard `lm` and `glm` functions both in terms of their signature and output.

The rationale behind the popular multiple imputation approach is to create $M > 1$ complete datasets by imputing the missing values with plausible values, and then to estimate a parameter of interest θ on each of the imputed datasets. The multiple estimations of θ and their variability reflect the uncertainty due to the unknown missing values. The parameter estimation is performed by applying the analytic method used, had the data been complete. This provides an estimate of the parameter θ and an estimate of the corresponding variance, for each imputed dataset. These quantities are finally 'pooled' by using specific rules named "Rubin's rules" (Rubin, 2004), leading to a final point estimate, with a corresponding estimation of its variance that takes into account the uncertainty due to missing values.

In the corresponding workflow, we compare this method to the previous EM algorithm and provide the basic lines of code required to estimate parameters of linear or logistic regression models with incomplete covariate data.

For an additional example of how to estimate regression parameters, we refer to the tutorial[18] on handling missing values in R by Julie Josse: it walks through a complete analysis, covering visualization of missing data patterns, data visualization, dimensionality reduction of incomplete data, and regression, in the presence of missing data.

How to predict in the presence of missing values?

As mentioned in the introduction, methods to deal with missing values are not the same when the aim is to estimate parameters or to predict a target variable. Josse et al. (2019) study the problem of supervised learning with missing values, i.e., when the aim is to predict an outcome y, from incomplete covariates in X. Note that contrary to the estimation setting, supervised learning involves training and test sets and both may have missing values. Josse et al. (2019) recommend to impute the training set and the test set with a same constant, such as the mean, and then to apply a universally consistent learner, i.e., a very powerful learner, such as gradient boosting, able to learn or fit any function. When forest-based methods are used to do prediction, another method is available, the Missing Incorporated in Attributes (MIA) method (Twala et al., 2008). Note that constant imputation or MIA are recommended asymptotically but when having limited data in the prediction setting, other imputation methods can outperform these asymptotically consistent methods (Josse et al., 2019). This is explored in the following workflows. The different methods are compared in terms of quality of the prediction of the outcome (AUC for a binary outcome and MSE for a continuous outcome).

In R The R workflow[19] assesses a popular strategy (two-step strategy) which involves independently imputing the training and test sets using the same imputation method. These datasets are then treated as being complete data, and regular learning algorithms are applied to predict some target variable.[20] Several imputation methods are compared, such as **mice**, **missForest**, **softImpute**, and mean imputation. Note that, until recently, using the popular **mice** package for learning predictive

[17] https://rmisstastic.netlify.app/how-to/estimate/missestim

[18] https://rmisstastic.netlify.app/tutorials/josse_bookdown_dataanalysismissingr_2020

[19] https://rmisstastic.netlify.app/how-to/external/how_to_predict_in_r

[20] This workflow has been written by an external contributor of the website, Katarzyna Woźnica (PhD student at the Warsaw University of Technology, Poland).

models on incomplete data in R was hindered by the fact that it did not allow using the same imputation model for the training and test set. This has, however, been addressed with the argument `ignore` of the R function `mice`, the details of this recent extension can be found on GitHub.[21]

In Python The Python workflow[22] compares two strategies, where the aim is to predict a target variable and the covariates may contain missing values:

1. The *two-step* strategy consists of imputing the missing values both in the training and in the test set with a method like mean imputation or `IterativeImputer` of the `scikit-learn` library, and to apply usual learning algorithms (random forests, gradient boosting, linear regression) on the imputed dataset. This learning algorithm can be applied to the imputed dataset $\tilde{X}$ but also to a new variable made of the combination of the imputed covariates $\tilde{X}$ with the response pattern R: $[\tilde{X}, R]$.
2. The *one-step* strategy performs prediction using learning methods adapted to the missing data without necessarily imputing them, such as the MIA method (Twala et al., 2008), which is in our notebook.

We propose a function, `score_pred`, which compares these strategies in terms of prediction performances by introducing missing values in complete covariates (`x_comp`) under a specific missing data mechanism (`mecha` and a given percentage of missing values (`p`). The code for calling this function is given below, when the learning algorithm is the gradient boosting and 20% of MCAR values are introduced.

```
learner = HistGradientBoostingRegressor()
p = 0.2
res = score_pred(x_comp=X, y = y, learner=learner , p=p, nbsim=10, mecha="MCAR")
```

The dataset is then split into a training set and a test set (75% in the training set, 25% in the test set) and the methods presented below are applied by considering a specific learning algorithm. The function then returns the prediction error on the test set, by comparing the ground truth (y) and the predicted outcome values on the test set for each simulation (i.e., each run for the generation of missing values). Figure 7 shows the graphical output of this function called for different learning algorithms (linear regression, random forests and gradient boosting) and for different missing data mechanisms (20% MCAR and MNAR, see the section on how to generate missing values). When the learner is linear regression, the two-step methods with added mask, both for the MCAR mechanism and the MNAR mechanism, perform well. Since the simulated dataset is generated using a linear regression model, the linear regression is expected to give better results than the other learners. In addition, for the MNAR mechanism, the one-step strategy *MIA* (especially when the gradient boosting is performed) appears to be a good choice.

Another function is specifically designed to handle datasets which already contain missing values. The second part of the Appendix shows a concrete example of this notebook on a real dataset.

This concludes the overview of the workflows developed in this project. We invite other practitioners and researchers to use and extend these methods. Overall, we hope that by creating and sharing implemented methods, new methods can be more easily developed and easily compared and evaluated.

4 Perspectives and future extensions

By providing a platform and community to discuss missing data, software, approaches and workflows, the sharing of expertise on missing data can hopefully be improved and extended more easily.

Towards uniformization and reproducibility

One way to promote and encourage practitioners and researchers in their work with missing values is to provide benchmarks and workflows around missing data. As has been shown in data competitions, community involvement produces many creative solutions and discussions that move the field forward, and challenge existing strategies. We will continue to work on our workflows and related source code. In doing so, we hope to encourage users to continue to test new methods and present results in a clear and reproducible manner. In addition, we plan to propose two types of data

[21] https://github.com/amices/mice/issues/32
[22] https://rmisstastic.netlify.app/how-to/python/predict_html/how%20to%20predict

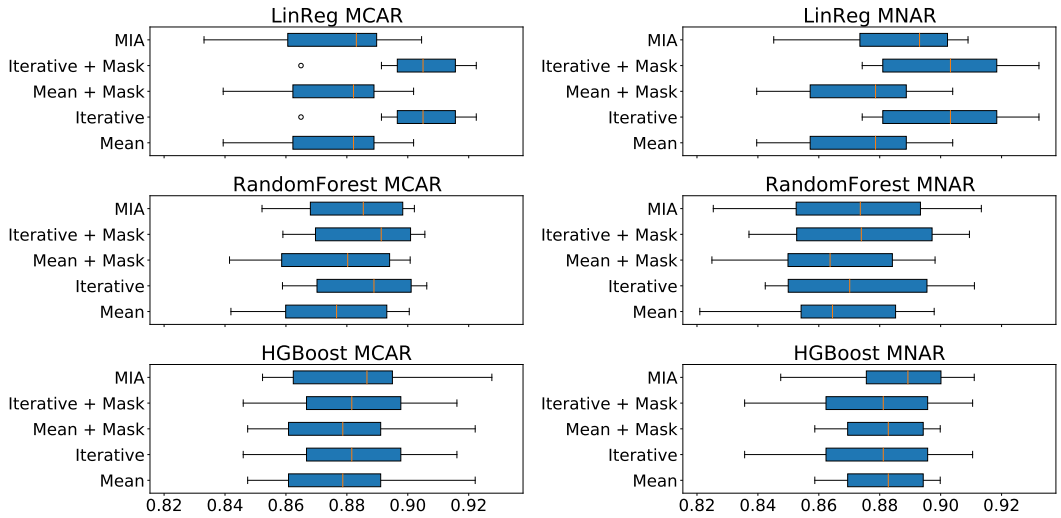


Figure 7: Plot of the function `score_pred` to compare different strategies when the aim is to predict in Python. 20% of missing values are introduced in a simulated dataset using the MCAR mechanism or the MNAR mechanism. The covariates $X \in \mathbb{R}^{1000\times 3}$ are generated under a multivariate Gaussian distribution, the parameter of the regression $\beta \in \mathbb{R}^3$ follows a random uniform distribution. The outcome y is generated according to a linear model such that $y = X\beta + \epsilon$, with ϵ representing Gaussian noise. The two-step strategies (`IterativeImputer` and the mean imputation) with or without adding a mask and the one-step strategy *MIA* are compared in terms of prediction error, and several learners are performed (linear regression, random forests, gradient boosting). The closer the result is to 1, the more accurate the prediction is (1 corresponds to perfect prediction, 0 to the worst prediction).

challenges: 1) imputation and estimation, and 2) analysis workflows. For the first part of the challenge, the objective is to find the best imputation or estimation strategy. The community will be given a dataset with missing values, for which there is actually a hidden copy of the real values. The community will then get the task of creating imputed values, which are assessed against the original dataset with complete values, to determine which imputation is best. This is similar in spirit to the Netflix prize (Bennett et al., 2007) and the M4 challenge in the time series domain (Makridakis et al., 2018). This benchmarking could be extended to other areas, such as parameter estimation, and predictive modeling with missing data. Analysis workflows could form another community challenge, assessed in a similar way to existing 'datathon' events where entries are assessed by an expert panel. Here the challenge could be to develop workflows and data visualizations from complex data. The data could have challenging features, and be combined from various data sources with complex structure, such as data with several types of missingness, images, text data, longitudinal data, and time series.

Future extensions

Possible enhancements that could be added in future releases of the platform, for which we welcome suggestions and contributions, are the following: A workflow with a focus on MNAR data and different solutions that can handle such data (as diversity of existing solutions is large, such a unified workflow will be a consequential contribution); for more applied users, a comparison of computation times of different methods, benchmarked on various types of data. Another problem that is becoming more common is missing values in data integration. Indeed, questions such as *what do I do when I have clinical data from multiple centers with different mechanisms of missing values or with systematically missing values in certain data?* or *what do I do when I have time series and missing values in one of the groups of variables?* would be also worth addressing in additional workflows.

Participation and interaction

This platform is aimed to offer a venue for the community, in the sense that we welcome every comment and question, encourage submissions of new works, theoretical or practical, either through the provided contact form or directly via the GitHub project repository. We have already received useful feedback and several external contributions, organized several remote calls and working sessions at statistics conferences. We are planning on regularly relaunching calls for new material for

the platform, for example through the R consortium blog[23], R-bloggers[24] and social media platforms. We also intend to use these channels to communicate more generally about the platform and the topic of missing values.

In order for the platform to be a reference to the community, it must provide regularly updated, user-friendly content. To achieve this goal, it is important to propose sustainable and accessible solutions for the maintenance of the 'R-miss-tastic' platform. We hope that the well documented source code of the platform facilitates external contributions and community feedback on this project.

In conclusion, the aim of this platform is to go beyond mere community participation, namely to seed meaningful community interactions, and to offer a hub of communication among groups that rarely exchange, both within, and between academia and industry.

5 Acknowledgements

This work has partially been funded by the R Consortium, Inc. We would like to thank Steffen MORITZ and François HUSSON for their active support and feedback, all contributors who have generously made their course and tutorial materials available, as well as the contributors to the workflows in R and Python code.

Bibliography

J. Bennett, S. Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. Citeseer, 2007. [p260]

F. Biessmann, D. Salinas, S. Schelter, P. Schmidt, and D. Lange. "deep" learning for missing value imputationin tables with non-numerical data. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 2017–2025, 2018. ISBN 978-1-4503-6014-2. doi: 10.1145/3269206.3272005. URL http://doi.acm.org/10.1145/3269206.3272005. [p247]

A. Bilogur. Missingno: a missing data visualization suite. *Journal of Open Source Software*, 3(22):547, 2018. doi: 10.21105/joss.00547. URL https://doi.org/10.21105/joss.00547. [p252]

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. [p248]

J. Carpenter and M. Kenward. *Multiple Imputation and its Application*. John Wiley & Sons, Dec. 2012. [p246, 249]

X. Cheng, D. Cook, and H. Hofmann. Visually exploring missing values in multivariable data using a graphical user interface. *Journal of statistical software*, 68(1):1–23, 2015. [p246]

B. Delyon, M. Lavielle, E. Moulines, et al. Convergence of a stochastic approximation version of the em algorithm. *The Annals of Statistics*, 27(1):94–128, 1999. [p258]

D. Dua and C. Graff. UCI machine learning repository, 2019. URL http://archive.ics.uci.edu/ml. [p252, 256]

C. K. Enders. *Applied missing data analysis*. Guilford press, 2010. [p246]

A. Gelman and J. Hill. Opening windows to the black box. *Journal of Statistical Software*, 40, 2011. [p246]

L. Gondara and K. Wang. Mida: Multiple imputation using denoising autoencoders. In D. Phung, V. Tseng, G. Webb, B. Ho, M. Ganji, and L. Rashidi, editors, *Proceedings of the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2018)*, Lecture Notes in Computer Science, pages 260–272. Springer International Publishing, 2018. ISBN 3319930404. doi: 10.1007/978-3-319-93040-4_21. URL https://arxiv.org/abs/1705.02737. [p249]

T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh. Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research*, 16(1):3367–3402, 2015. [p256]

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/. [p246]

[23] https://www.r-consortium.org/news/blog
[24] https://www.r-bloggers.com/

R. Ihaka. R: Past and future history. *Computing Science and Statistics*, 392396, 1998. [p246]

W. Jiang, J. Josse, M. Lavielle, and T. Group. Logistic regression with missing covariates—parameter estimation, model selection and prediction within a joint-modeling framework. *Computational Statistics & Data Analysis*, 145:106907, 2020. [p258]

J. Josse and J. P. Reiter. Introduction to the special section on missing data. *Statistical Science*, 33(2): 139–141, 2018. [p246]

J. Josse, F. Husson, et al. missmda: a package for handling missing values in multivariate data analysis. *Journal of Statistical Software*, 70(1):1–31, 2016. [p246, 256]

J. Josse, N. Prost, E. Scornet, and G. Varoquaux. On the consistency of supervised learning with missing values. *arXiv preprint*, 2019. URL https://arxiv.org/abs/1902.06931. [p249, 254, 258]

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. [p246, 254]

S. Lê, J. Josse, and F. Husson. Factominer: an r package for multivariate analysis. *Journal of statistical software*, 25:1–18, 2008. [p264]

R. J. Little and D. B. Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019. [p246, 249]

S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018. [p260]

P.-A. Mattei and J. Frellsen. Miwae: Deep generative modelling and imputation of incomplete data sets. In *International Conference on Machine Learning*, pages 4413–4423. PMLR, 2019. [p256]

I. Mayer, E. Sverdrup, T. Gauss, J.-D. Moyer, S. Wager, and J. Josse. Doubly robust treatment effect estimation with missing attributes. *Ann. Appl. Statist.*, 14(3):1409–1431, 2020. ISSN 1932-6157. doi: 10.1214/20-AOAS1356. [p248, 249]

S. Moritz and T. Bartz-Beielstein. imputeTS: Time Series Missing Value Imputation in R. *The R Journal*, 9(1):207–218, 2017. doi: 10.32614/RJ-2017-009. [p248]

J. S. Murray and J. P. Reiter. Multiple imputation of missing categorical and continuous values via bayesian mixture models with local dependence, 2015. URL http://arxiv.org/abs/1410.0438. [p246]

B. Muzellec, J. Josse, C. Boyer, and M. Cuturi. Missing data imputation using optimal transport. In *International Conference on Machine Learning*, pages 7130–7140. PMLR, 2020. [p256]

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011. [p247, 252, 256]

D. B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976. [p254]

D. B. Rubin. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004. [p258]

J. L. Schafer and J. W. Graham. Missing data: our view of the state of the art. *Psychological methods*, 7 (2):147–177, June 2002. [p246, 249]

A. Sportisse. Handling heterogeneous and mnar missing data in statistical learning frameworks: imputation based on low-rank models, online linear regression with sgd, and model-based clustering. 2021. [p254]

StataCorp. *Stata Statistical Software: Release 16*. StataCorp LLC., College Station, TX, 2019. [p252]

D. J. Stekhoven and P. Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012. [p255]

N. Tierney, D. Cook, M. McBain, and C. Fay. *naniar: Data Structures, Summaries, and Visualisations for Missing Data*, 2021. URL https://CRAN.R-project.org/package=naniar. R package version 0.6.1. [p246, 254]

N. J. Tierney and D. H. Cook. Expanding tidy data principles to facilitate missing data exploration, visualization and assessment of imputations. *arXiv preprint arXiv:1809.02264*, 2018. [p246]

B. Twala, M. Jones, and D. J. Hand. Good methods for coping with missing data in decision trees. *Pattern Recognition Letters*, 29(7):950–956, 2008. [p258, 259]

S. van Buuren. *Flexible Imputation of Missing Data, Second Edition*. CRC Press, 2018. [p246, 249]

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/. [p246, 254, 255]

G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697. [p246, 252]

Y. Xie, A. Presmanes Hill, and A. Thomas. *blogdown: Creating Websites with R Markdow*. The R Series. Chapman and Hall/CRC, 2017. ISBN 978-0815363729. [p247]

Z. Zhu, T. Wang, and R. J. Samworth. High-dimensional principal component analysis with heterogeneous missingness. *arXiv preprint arXiv:1906.12125*, 2019. [p246]

1 Appendix

Tutorial for imputing missing values in R

The goal of this tutorial is to give practical details on the R-workflow entitled *How to impute missing values?*[25]. In this workflow, users can compare the most popular methods to impute missing values in R on simulated or real datasets.

We illustrate this workflow by considering a small dataset called *decathlon*, that contains athletes' performance during two sporting events (41 rows, 13 columns[26]). It is available in the R package **FactoMineR** (Lê et al., 2008).

```
library(FactoMineR)
data(decathlon)
head(decathlon[,1:4]) # four first columns of the  dataset
         100m Long.jump Shot.put High.jump
SEBRLE  11.04      7.58    14.83      2.07
CLAY    10.76      7.40    14.26      1.86
KARPOV  11.02      7.30    14.77      2.04
BERNARD 11.02      7.23    14.25      1.92
YURKOV  11.34      7.09    15.19      2.10
WARNERS 11.11      7.60    14.31      1.98
```

If we have collected similar data, e.g., described by the same variables but for new athletes, that contain missing values, practitioners may want to know how to impute such a dataset. To address this question, we can introduce missing values under different mechanisms (MCAR, MAR or MNAR) and with different percentages of missing values (here we compare 20% and 50%) in the complete dataset and compare some imputation methods in terms of mean squared error (MSE), i.e., the error committed by the imputation of the missing values. Missing values are introduced in all covariates. The function `how_to_impute` can be used to compare the imputation methods described in the section on how to impute missing values (**missMDA**, **mice**, **missForest**, **softImpute** and the imputation by the mean) using different percentages and types of missing values given in two lists by the users. More particularly, the arguments are the following ones: the complete dataset where the missing values will be introduced (`X`), a list containing the different percentage of missing values (`perc.list`), a list containing the different missing-data mechanisms (`mecha.list`) and the number of simulations performed (`nbsim`). Note that for **missMDA**, the number of components in the PCA used to predict the missing entries is estimated using a cross-validation with the function `estim_ncpPCA`. For **softImpute**, we use a cross-validation to choose the regularization parameter (coded for the purpose of the notebook). This function returns a table with the mean of the MSEs over the simulations for the different methods and for the different missing data settings (20% MCAR values, 50% MCAR values, 20% MAR values, 50% MAR values, 20% MNAR values, 50% MNAR values).

```
perc.list <- c(0.2,0.5)
mecha.list <- c("MCAR", "MAR", "MNAR")
res <- how_to_impute(X=decat_sc, perc.list=perc.list,mecha.list=mecha.list,nbsim=10)
res
```

```
           0.2 MCAR  0.5 MCAR   0.2 MAR   0.5 MAR  0.2 MNAR  0.5 MNAR
X.pca     0.8822782 1.0537611 0.9394561 1.0873315 0.9876867 1.1026891
X.forest 0.8820789 0.9577351 0.9403659 1.0526915 0.9940827 1.0809478
X.mice    0.8610320 1.0372518 0.9559042 1.0948981 0.9887239 1.1271581
X.soft    0.7935545 0.8865989 0.8721907 0.9556373 0.8951239 0.9692859
X.mean    1.0177192 1.0306089 1.0972080 1.0770715 1.1342289 1.1077467
```

With this result in hand, we can easily visualize some of the results. Figure 8 shows the associated graphics, for each of the missing-data mechanism. The code to obtain these graphics is given below.

```
plotdf <- do.call(c, res)
plotdf <- as.data.frame(plotdf)
names(plotdf) <- 'mse'
n_perc.list <- length(perc.list)
n_mecha.list <- length(mecha.list)
```

[25]This tutorial is only an example of use but more practical details are given in the original workflow.

[26]We do not consider the last variable in this part, which is categorical. Some imputation methods do not handle mixed data.

```
methods.list <- c("PCA", "RandomForest",  "Mice", "SoftImpute", "Mean")
meth <- rep(methods.list, n_perc.list * n_mecha.list)
plotdf <- cbind(plotdf, meth)
perc <- rep(rep(as.character(perc.list), each = 5),length(mecha.list))
plotdf <- cbind(plotdf, perc)
mecha <- rep(mecha.list, each = 5 * length(perc.list))
plotdf <- cbind(plotdf, mecha)

# For MCAR data
ggplot(plotdf[plotdf['mecha'] == "MCAR",])
+ geom_point(aes(x = perc, y = mse, color = meth), size = 2)
+ ylab("MSE") + xlab("Percentage of NA")
+ geom_path(aes(x = perc, y = mse, color = meth, group = meth))
+ ggtitle("MNAR") + labs(color = "Methods") + theme(text = element_text(size = 20))

# For MAR data
ggplot(plotdf[plotdf['mecha'] == "MCAR",])
+ geom_point(aes(x = perc, y = mse, color = meth), size = 2)
+ ylab("MSE") + xlab("Percentage of NA")
+ geom_path(aes(x = perc, y = mse, color = meth, group = meth))
+ ggtitle("MNAR") + labs(color = "Methods") + theme(text = element_text(size = 20))


# For MNAR data
ggplot(plotdf[plotdf['mecha'] == "MCAR",])
+ geom_point(aes(x = perc, y = mse, color = meth), size = 2)
+ ylab("MSE") + xlab("Percentage of NA")
+ geom_path(aes(x = perc, y = mse, color = meth, group = meth))
+ ggtitle("MNAR") + labs(color = "Methods") + theme(text = element_text(size = 20))
```

For this dataset and these missing data settings, **softImpute** appears to be the best imputation method.

Tutorial for predicting in presence of missing values in Python

The goal of this tutorial is to give practical details on the Python-workflow entitled *How to predict with missing values?*[27]. In this workflow, users can compare methods in Python to predict a target variable when the covariates contain missing values.

We consider the dataset called *california_housing* (20640 rows, 9 columns). The target variable is the median house value for California districts and the covariates provide information (latitude, longitude, number of people in the district...) on the different districts.

If we know that new observations will contain missing values, an interesting question is how to predict the target variable in presence of covariates with missing values. To answer this, we can impute missing values in the covariates and compare methods which handle them and predict the target variable.

First, we generate missing values in the covariates of the dataset *california_housing*, using the function `produce_NA`. The three main arguments are the initial dataset (`X`) in which missing values are introduced using a given missing data mechanism (`mecha`) and a given percentage of missing values (`p_miss`). In the following example, we introduce 20% MCAR values.

```
XproduceNA_MCAR = produce_NA(X = x_comp, p_miss = 0.2, mecha = "MCAR")
x_MCAR = XproduceNA_MCAR['X_incomp'].numpy()
```

To predict, we consider two strategies presented in the section on how to predict in the presence of missing values: (i) The *two-step* strategy which consists of imputing missing values and applying classical methods on the completed datasets to predict, and (ii) the *one-step* strategy which predicts using methods adapted to the missing values without necessarily imputing them. The code below allows comparison of different prediction strategies nested in a two-step or a one-step strategy. To do so, we use the function `plot_score_realdatasets`, which handles datasets already containing missing values. Figure 9 shows the graphical output of this function called for different learning algorithms. When the learner is the linear regression, the two-step methods with added mask perform

[27]This tutorial is only an example of use but more practical details are given in the original workflow.

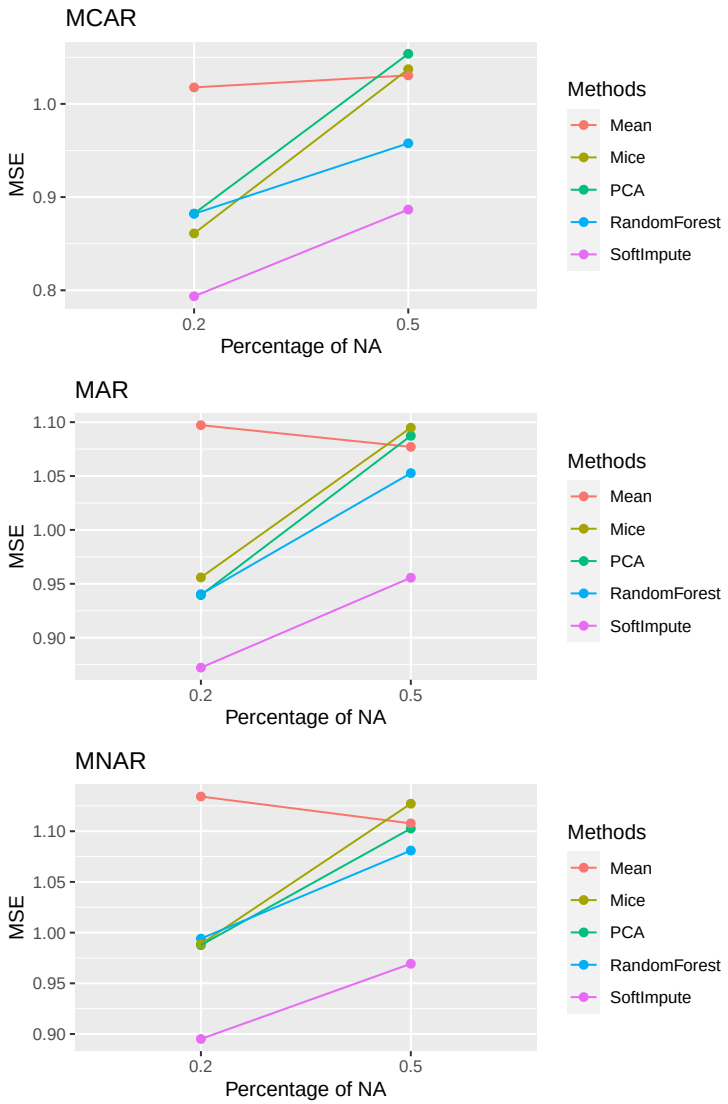


Figure 8: Graphical outputs of the R function `how_to_impute`. The methods **mice**, **missForest**, **softImpute** and **missMDA** are compared with the naive imputation by the mean for several percentages of missing values (20%, 50%). The mean of the MSEs computed for several generations of missing values are given. The results are shown for different mechanisms (MCAR, MAR, MNAR).

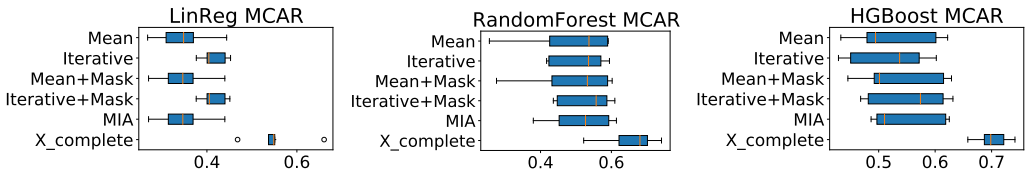


Figure 9: Graphical outputs of the Python function `plot_score_realdatasets`. The x-axis indicates the MSE. The two-steps methods considering the imputation by the mean (Mean), IterativeImputer (Iterative) with or without adding the mask and the one-step method MIA are compared with the case without missing values (X_complete). Note that the mask is the binary matrix which indicates where are the missing values. When we add the mask, we consider then an augmented matrix, with the initial matrix and the mask. The learner are the linear regression (left panel), the random forests (middle panel) and gradient boosting (right panel).

well. Indeed, since the simulated dataset is generated considering a linear regression, the linear regression is expected to give better results than the other learners.

The code for the function `plot_score_realdatasets` is given below. The main arguments are the dataset (`X`), the outcome variable (`y`) and the learning algorithm to use (`learner`).

```
learners = {'LinReg': LinearRegression()
            'RandomForest': RandomForestRegressor(),
            'HGBoost': HistGradientBoostingRegressor()}
for learner_name, learner in learners.items():
    plt.figure(figsize=(10,10))
    for ii, (X, X_name) in enumerate(zip([x_MCAR], ['MCAR'])):
        plt.subplot(3, 2, ii+1)
        plot_score_realdatasets(X, y, learner, learner_name + ' ' +X_name,x_comp)
```

Table 1: Comparison of the scope and functionality of different R and Python packages used for the same tasks in the R-miss-tastic R and Python workflows. If existing, differences between approximately equivalent packages or functions are summarized in the last column.

R implementation	Scope	Python counterpart	Differences (if any)
imputeMean (implemented in R-miss-tastic workflow)	Impute missing values of each variable by their means.	module sklearn.impute, `SimpleImputer` with `strategy='mean'`	
softImpute	Impute missing values using a low-rank completion with nuclear norm penalities	function softImpute (implemented in R-miss-tastic workflow)	The R package is better optimized than our Python version. The R implementation also has different optimization algorithms implemented (itervative SVD, iterative ALS).
mice	Give multivariate imputations by chained equations	module sklearn.impute `IterativeImputer` with `BayesianRidge`	The Python module uses iterative chained equations. However, it differs from the mice package, because it uses a ridge iterate and it returns by default a single imputation. Note that the argument `sample_posterior=True` allows to get stochastic imputations, and not multiple imputations, as the R function mice does.
missForest	Impute missing values using random forests	module sklearn.impute, `IterativeImputer` with `ExtraTreesRegressor`	The Python implementation does not exactly use random forests with CART trees, but forests with trees which choose a random split (instead of the best split per feature).
missMDA	Impute missing values using a low-rank matrix completion with penality		

Imke Mayer
Institute of Public Health, Charité – Universitätsmedizin Berlin
imke.mayer@charite.de

Aude Sportisse
Maasai, Inria Sophia Antipolis
aude.sportisse@inria.fr

Julie Josse
PreMeDICaL, Inria Sophia Antipolis
julie.josse@inria.fr

Nicholas Tierney
Department of Econometrics and Business Statistics, Monash University
nicholas.tierney@gmail.com

Nathalie Vialaneix
MIAT, Université de Toulouse, INRA
nathalie.vialaneix@inrae.fr