Journal of Statistical Software

January 2023, Volume 105, Issue 7. doi: 10.18637/jss.v105.i07

Expanding Tidy Data Principles to Facilitate Missing Data Exploration, Visualization and Assessment of Imputations

Nicholas Tierney
Monash University,
Telethon Kids Institute

Dianne Cook
Monash University

Abstract

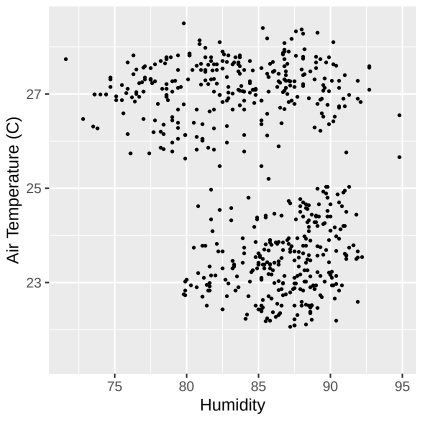
Despite the large body of research on missing value distributions and imputation, there is comparatively little literature with a focus on how to make it easy to handle, explore, and impute missing values in data. This paper addresses this gap. The new methodology builds upon tidy data principles, with the goal of integrating missing value handling as a key part of data analysis workflows. We define a new data structure, and a suite of new operations. Together, these provide a connected framework for handling, exploring, and imputing missing values. These methods are available in the R package **naniar**.

Keywords: statistical computing, statistical graphics, data science, data visualization, **tidyverse**, data pipeline, R.

1. Introduction

As data science has become a more solid field, theories and principles have developed to describe best practices. One such idea is "tidy data", which defines a clean, analysis-ready format that informs workflows converting raw data through a data analysis pipeline (Wickham 2014). Another idea is the grammar of graphics, describing how to map data values into a visualization (Wilkinson 2012). These principles have been widely adopted, but do not address the problem of missing data. In particular, there is little guidance on how to handle missing values in a data analysis workflow. Most analysis and visualization software simply drop missing values when making a plot, although some (**ggplot2**) provide a warning (Figure 1).

The imputation literature focuses on ensuring valid statistical inference is made from incomplete data. This is approached chiefly through probabilistic modeling, assuming the mecha-



```
Warning message:
removed 171 rows containing
missing values (geom_point).
```

Figure 1: How **ggplot2** behaves when displaying missing values. A warning message is displayed, but missing values are not shown in the plot.

nism of missing data is known to the analyst. It does not address how to explore, understand, and handle missing data structures and mechanisms.

However, something must be known about the missing value structure to produce a complete dataset for analysis, whether by case- or variable-deletion, or with imputation. Decision tree models or latent group analysis can reveal structures or patterns of missingness (Tierney, Harden, Harden, and Mengersen 2015; Barnett, McElwee, Nathan, Burton, and Turrell 2017), but are not definitive. While there are times where the missing data mechanism is obvious, determining this is typically not straightforward. A straightforward case is where missing values are encoded in the data in a non standard format that still indicates missingness, such as "N.A.", "N/A", or "missing". However, identifying these amongst all data can be problematic as it is hard to know what you are looking for, like finding a needle in a haystack. A more challenging case is where missing values arise due to the increasing or decreasing values of other variables (including themselves). For example, in weather data, temperature values might go missing due to their own high temperature. Or similarly, other measurements go missing due to high temperatures. Identifying this requires exploring the data, and the analysis path to take is obvious, as missing values are often omitted from plots and other statistical summaries. To understand their structure and possible mechanisms, the analyst must explore the data with visualizations, summaries, and modeling, in an iterative fashion.

While there have been many software tools for exploring missing data, they do not work together. The graphics literature provides several solutions for exploring missings visually by incorporating them into the plot in some way. For example, imputing values to be 10% below the minimum to include all observations into a scatter plot (Cook and Swayne 2007), or treating missing values as an equal category of data (Unwin, Hawkins, Hofmann, and Siegl 1996). The ideas from the graphics literature need to be translated into tidy data tools to integrate missing data handling in a data analysis pipeline. We have implemented these tools for exploring missing data into the R package **naniar** (Tierney, Cook, McBain, and Fay 2023), available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=naniar`.

This paper is organized in the following way. Section 2 provides the background to tidy data principles and tools (Section 2.1) and missing data representations (Section 2.2). Section 3 summarizes existing software for handling missing data. The new work extending the tidy tools to facilitate exploring, visualizing, and imputing missing data is discussed in Section 4. Graphics (Section 5) and numerical summaries (Section 6) of missing values are then discussed. An application illustrating the use of the new methods is shown in Section 7. Section 8 discusses strengths, limitations, and future directions.

2. Background

2.1. Tidy data concepts and methods

Features of tidy data were formally described in Wickham (2014), and were discussed in terms of their importance for data science by Donoho (2017), and tools for data analysis. Tidy data is defined by Wickham and Grolemund (2017) as:

1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell.

Tidy data is easier to work with and analyze because the variables are in the same format as they would be put into modeling software. This helps the analyst works swiftly and clearly, closing up opportunities for errors. Tidy data principles are general, but are comprehensively implemented in the R programming language, so this paper focuses on R (R Core Team 2022).

Tidy tools require the same tidy data input and output. This consistency means multiple tools can be composed together into a sequence, allowing for rapid, elegant, and complex operations. Contrasting tidy tools are messy tools. These have tidy input but messy output. Messy tools slow down analysis by shifting the focus from analysis to transforming output so it is the right shape for the next step in the analysis. This makes the work at each step harder to predict, and more complex and difficult to maintain. This disrupts workflow, and invites errors. Tidy tools fall into three broad categories: data manipulation, visualization, and modeling.

Data manipulation

Data manipulation is made input- and output-tidy with R packages **dplyr** and **tidyr** (Wickham, François, Henry, and Müller 2022a; Wickham 2022b). These provide the five "verbs" of data manipulation: data reshaping, sorting, filtering, transforming, and aggregating. Data reshaping goes from long to wide formats; sorting arranges rows in a specific order; filtering removes rows based on a condition; transforming, changes existing variables or adds new ones; aggregating creates a single value from many values, say, for example, in computing the minimum, maximum, and mean.

Visualizations

Visualization tools only have tidy data as their input, as the output is a graphic. The popular domain specific language **ggplot2** maps variables in a dataset to features (referred to as aesthetics) of a graphic (Wickham 2016). For example, a scatterplot can be created by mapping two variables to the x and y axes, and specifying a point geometry.

Modeling

Modeling tools work well with tidy data, as they have a clear mapping from variables in the data to the formula for a model. For example in R, y regressed on x and z is: `lm(y ~ x + z)`. Modeling tools are input tidy, but their output is always messy – it is not in the right format for subsequent steps in analysis. For example, estimated coefficients, predictions, and residuals from one model cannot be easily combined with the output of another model. Messy models have been partially addressed with the **broom** package (Robinson, Hayes, and Couch 2022), which tidies up model outputs into a tidy data format for data analysis, and the developing **recipes** package, which helps make modeling input- and output-tidy (Kuhn and Wickham 2022).

Tidy data and tidy tools

Defining tidy data and tidy tools has resulted in a growing set of packages known collectively as the **tidyverse** (Wickham *et al.* 2019). These are constructed to share similar principles in their design and behavior, and cover the breadth of an analysis - from importing, tidying, transforming, visualizing, modeling, to communicating (Wickham and Grolemund 2017; Wickham *et al.* 2019). This has led to more tools for specific parts of analysis – from reading in data with **readr**, **readxl**, and **haven**, to handling character strings with **stringr**, dates with **lubridate**, and performing functional programming with **purrr** (Wickham, Hester, and François 2022b; Wickham and Bryan 2022; Wickham, Miller, and Smith 2022c; Wickham 2022a; Grolemund and Wickham 2011; Henry and Wickham 2020). It has also led to a burgeoning of new packages for other fields following similar design principles, creating fluid workflows for new domains. For example, the **tidytext** (Silge and Robinson 2016) package for text analysis, the **tsibble** (Wang, Cook, and Hyndman 2020) package for time series data, and **tidycensus** (Walker 2022) for working with US census and boundary data.

Tidy formats for missing data

Current tools for missing data are messy. Missing data tools can be used to perform imputations, missing data diagnostics, and data visualizations. However, these tools suffer the same problems as modeling: they use tidy input, but produce messy output – their output is challenging to integrate with other steps of data analysis. The complex, often multivariate nature of imputation methods also makes them difficult to represent. Visualization methods for missing data do not map data features to the aesthetics of a graphic, as in **ggplot2**, limiting expressive exploration.

Taking existing methods from the missing data graphics literature, and translating and expanding them into tidy data and tidy tools would create more effective data visualizations. Defining these concepts allows the focus to be more general than just software, but rather, an extensible framework for tidy tools to explore missing data.

2.2. Missing data representation and dependence

The convention for representing missingness is a binary matrix, B, for data y with i rows and j columns:

$$b_{ij} = \begin{cases} 1, & \text{if } y_{ij} \text{ is missing} \\ 0, & \text{if } y_{ij} \text{ is observed} \end{cases}$$

There are many ways each value can be missing, we adopt the notation used in (van Buuren 2018). The information in B can be used to arrive at three categories of missing values: missing completely at random (MCAR), missing at random (MAR), and missing not at random (MNAR). The distribution of missing values in b_{ij} can depend on the entire dataset, represented as $Y = (Y_{\text{obs}}, Y_{\text{miss}})$. This relationship can be defined by the *missing data model* $\mathsf{P}(b_{ij} \mid Y_{\text{obs}}, Y_{\text{miss}}, \psi)$, the probability of missingness is conditional on data observed, data missing, and some probability parameter of missingness, ψ. This helps to precisely define categories of missing values.

MCAR is where values being missing have no association with observed or unobserved data, that is, $\mathsf{P}(B = 1 \mid Y_{\text{obs}}, Y_{\text{miss}}) = \mathsf{P}(B = 1 \mid \psi)$. Essentially, the probability of an observation being missing is unrelated to anything else, only the parameter ψ, the overall probability of missingness. Although a convenient scenario, it is not actually possible to confirm, or clearly distinguish from MAR, as it relies on statements on data unobserved. In MAR, missingness only depends on data observed, not data missing, that is, $\mathsf{P}(B = 1 \mid Y_{\text{obs}}, Y_{\text{miss}}, \psi) = \mathsf{P}(B = 1 \mid Y_{\text{obs}}, \psi)$. Some structure or dependence between missing and observed values is allowed, provided it can be explained by data observed, and some overall probability of missingness. In MNAR, missingness is related to values observed, and unobserved: $\mathsf{P}(B = 1 \mid Y_{\text{obs}}, Y_{\text{miss}}, \psi)$. This assumes conditioning on all observations: data goes missing due to some phenomena unobserved, including the structure of the missing data itself. This presents a challenge in analysis, as it is difficult to verify, and implies bias in analysis due to the unobserved phenomena. Visualizations can help assess whether data may be MCAR, MAR or MNAR.

3. Existing Software

Methods for exploring, understanding, and imputing missing data are more accessible now than they have ever been. Values can be imputed with one value (single imputation), or multiple values (multiple imputation), creating m datasets. This section discusses existing software for single and multiple imputation, and missing data exploration.

3.1. Imputation

VIM (Kowarik and Templ 2016) implements well-used imputation methods: K nearest neighbors, regression, hot-deck, and iterative robust model-based imputation. These diverse approaches allow for imputing with semi-continuous, continuous, count, and categorical data. **VIM** identifies imputed cases by adding an indicator variable with a suffix `_imp`. So, `Var1` has a sibling column, `Var1_imp`, with values `TRUE` or `FALSE` indicating imputation. **VIM** also has a variety of visualization methods, discussed in Section 3.2. **simputation** provides an interface to imputation methods from **VIM**, in addition providing hotdeck imputation, and the expectation-maximization (EM) algorithm (van der Loo 2022; Dempster, Laird, and Rubin 1977). **simputation** provides a consistent formula interface for all imputation methods, and always returns a dataframe with the updated imputations. **Hmisc** (Harrell Jr 2022), provides predictive mean matching, **imputeTS** (Moritz and Bartz-Beielstein 2017), provides time series imputation methods, and **missMDA** (Josse and Husson 2016), imputes data using principal component analysis.

Multiple imputation is often regarded as best practice for imputing values (Schafer and Graham 2002), as long as appropriate caution is taken (Sterne *et al.* 2009). Popular and robust methods for multiple imputation include the **mice**, **Amelia**, and **mi** packages (van Buuren and Groothuis-Oudshoorn 2011; Honaker, King, and Blackwell 2011; Su, Gelman, Hill, and Yajima 2011). **mice** implements the method of chained equations, using a variable-wise algorithm to calculate the posterior distribution of parameters to generate imputed values. The workflow in **mice** revolves around imputing data, returning completed data, and fitting a model and pooling the results.

Amelia (Honaker *et al.* 2011) assumes data are multivariate normal, and samples from the posterior, and allows for incorporation of information on the values in a prior. It uses the computationally efficient (and parallelizable) expectation-maximization bootstrap (EMB) algorithm (Honaker and King 2010). **norm** (Novo and Schafer 2022), provides multiple imputation using EM for multivariate normal data, drawing from methods in the NORM software (Schafer 1999). **norm** does not provide a framework for tracking missing values, instead providing tools for making inference from multiple imputation.

mi (Su *et al.* 2011) also uses Bayesian models for imputation, providing better handling of semi-continuous values, and data with structural or perfect correlation. A collection of analysis models are also provided in **mi**, to work with data it has imputed. These include linear models, generalized linear models, and their corresponding Bayesian components. This approach promotes fluid workflow, with a similar penalty to tidying up model output, which is still messy.

Summary

Each imputation method provides practical methods for different use cases, but most have different output structures, and do not have consistent interfaces in their implementation. This makes them inherently messy and challenging to integrate into an analysis pipeline. For example, combining different imputation methods from different pieces of software is not currently straightforward. **simputation** resolves some of these complications with a simple approach of a unified syntax for all imputation, and always returns a dataframe of imputed values. This reduces the friction of working with other tools, but comes at the cost of identifying imputed values. An ideal approach would use consistent, simple data structures that work with other analysis tools, and help track missing values. This would make imputation outputs tidy, streamlining subsequent analysis.

3.2. Exploration

Missing data packages are focused primarily on making inferences, and exploring imputed values, not on exploring relationships in missing values, and identifying possible patterns. Texts covering the exploration phase of missing data have the same problem as with modeling: the input is tidy, but the output does not work with other tools (van Buuren 2018); this is inefficient. Methods for exploring missing values are primarily covered in literature on interactive graphics (Swayne and Buja 1998; Unwin *et al.* 1996; Cook and Swayne 2007), and are picked up again in a discussion of a graphical user interface (Cheng, Cook, and Hofmann 2015).

The missingness matrix B can be used to assess missing data dependence. It has been used in interactive graphics, dubbed a “shadow matrix”, to link missing and imputed values to the data, facilitating their display (Swayne and Buja 1998), focusing heavily on multivariate numeric data. This is an idea upon which this new work builds.

The **MANET** (missings are now equally treated) software (Unwin *et al.* 1996) focused on multivariate categorical data, with missingness explicitly added as a category. **MANET** also provided univariate visualizations of missing data using linked brushing between a reference plot of the missingness for each variable, and a plot of the data as a histogram or barplot. The **MANET** software is no longer maintained and cannot be installed. The approach of (Swayne and Buja 1998) in the software XGobi, further developed in **ggobi** (Cook and Swayne 2007), focused on multivariate quantitative data. Missingness is incorporated into plots in **ggobi** by setting them to be 10% below the minimum value.

MissingDataGUI (Cheng, Cook, and Hofmann 2011) provides a graphical user interface (GUI) for exploring missing data structure, both numerically and visually. Using a GUI to explore missing data facilitates rapid insight into missingness structures. However, this comes as a trade off, as insights are not captured or recorded with a GUI, making it challenging to incorporate into reproducible analyses. This distracts and breaks analysis workflow, inviting mistakes.

VIM (visualizing and imputing missing data) provides visualization methods to identify and explore observed, imputed, and missing values. These include spinograms, spinoplots, missingness matrices, plotting missingness in the margins of other plots, and other summaries. However, these visualizations do not map variables to graphical aesthetics, creating friction when moving through analysis workflows, making them difficult to extend to new circumstances. Additionally, data used to create the visualizations cannot be accessed, posing a barrier to further exploration.

The **inspectdf** package provides tools for summarizing, comparing, and visualizing dataframes (Rushworth 2022). In particular the `inspect_na()` function provides a summary of missing values for variables. `inspect_cat()` summarizes the levels of categories in each column in a dataset. The `show_plot()` function shows a higher lever overview of the contents of a dataframe and its categories, as well as missing values. All of these functions allow for comparing missing values across two dataframes. For example, comparing the missing values in two dataframes with the same column names.

ggplot2 removes missing values with a warning (Figure 1), and only incorporates missingness into visualizations when mapping a discrete variable with missings to a graph aesthetic. This has some limitations, shown in Figure 2, a boxplot visualization of school grade and test scores. If there are missings in a continuous variable like test score, **ggplot2** omits the missings and prints a warning message. However, if a discrete variable like school year has missing values, an NA category is created for school year, where scores are placed.

3.3. Approaches in other languages

Missing data can be explored and visualized in closed source software such as Stata (StataCorp 2021), SAS (SAS Institute Inc. 2020), and SPSS (IBM Corporation 2022). These provide basic table frequencies of missing data, and basic visualizations on missingness patterns. However, none of these software provide functions to specifically visualize missing values in bivariate or multivariate settings, and it is only possible to visualize missing values after performing data analysis, counting and summarizing missingness. Imputation methods from mean to regression, EM, and multiple imputation are all available, with out of the box diagnostics provided to assist comparing imputations. Python also provides exploratory data analysis for missing data within **missingno** (Bilogur 2018), and the **scikit-learn** package provides imputation methods (Pedregosa *et al.* 2011).

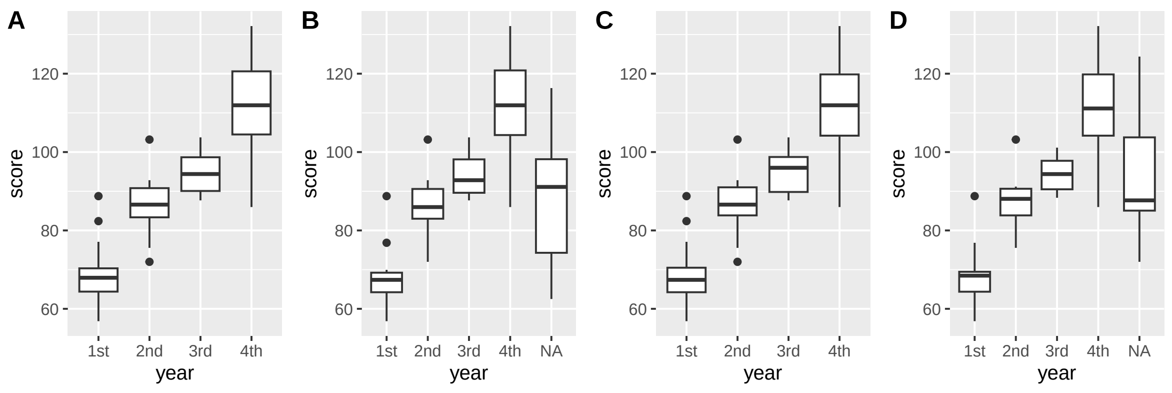


Figure 2: **ggplot2** provides different visualizations depending on what type of data has missing values for data of student test scores in school year. (A) Data is complete; (B) Missings are only in year – an additional "NA" boxplot is created; (C) Missings only in scores, no additional missingness information is shown; (D) Missings in both scores and year, additional missing information is shown. The missingness category is only shown when there are missings in categorical variables such as year (B and D). In (C), no missingness information is given on the graphic, despite there being missings in score, and a warning message is displayed about the number of missing values omitted.

4. Tidy framework for missings

Applying **tidyverse** principles has the potential to clarify missing data exploration, visualization, and imputation. This section discusses how these principles are applied for data structures (Section 4.1), common operations (verbs, Section 4.3), graphics (Section 5), and data summaries (Section 6). Care has been taken to make the names and design these features intuitive for their purpose, and is discussed throughout.

4.1. Data structure

A data structure facilitating exploration of missing data needs to be simple to reason with and to transport with a data analysis, otherwise it will not be used. A useful template common in missing data literature, is the B matrix, where 0 and 1 indicate not missing and missing, respectively. This matrix was used to explore missing values in the interactive graphics library XGobi, called a "missing value shadow", or "shadow matrix", defined as a copy of the original data with indicator values of missingness. The shadow matrix could be interactively linked to the data. However, there are some limitations to the shadow matrix. Namely, the values 0 and 1 can be confusing representations of missing values, since it is not clear if 0 indicates an absence of observation, or the presence of a missing value.

We propose a new form for tidy representation of missing data based on these ideas from past research. Four features are added to the shadow matrix to facilitate analysis of missing data, illustrated in Figure 3. These are briefly described below, and discussed in further detail in Section 4.2.

1. *Missing value labels*: Simple labels for missing and not missing to clearly identify missing values for analysis and plotting. We propose "NA" and "!NA" (Figure 3). This improves

the 0 and 1 values in B, which do not clearly identify missingness. These values follow a principle: "clarity of labelling" – the matrix's meaning is transparent, and anybody looking at these values could understand what they mean, which is not the case of binary values. Equally, these values could instead be "missing" or "present".

2. *Special missing values*: Building on *missing value labels*, the values in the shadow matrix can be "special" missing values, indicated by "NA_<suffix>". For example: "NA_instrument" uses a short label, "instrument", indicating instrument error resulting in missing values. These could be also used to indicate imputations (Figure 3).

3. *Coordinated names*: Variable names in the shadow matrix gain a consistent short suffix, "_NA", keeping names coordinated throughout analysis (Figure 3). It makes a clear distinction with "var_NA" being a random variable of the missingness of a variable, "var". This suffix is short and easy to remember during data analysis, and shifts the focus from the value of a variable, to its missingness state. Although this might break naming conventions for lower case "snake_case" variables (e.g., Wickham 2014), this distinguishes these columns from other columns.

4. *Connectedness*: Binding the shadow matrix column-wise to the original data as a factor creates a single, connected, nabular data format, in sync with the data, at the cost of a larger file. It is useful for visualization, summaries, and tracking imputed values, discussed in more detail in Section 4.2.

4.2. Nabular data

Nabular data binds the shadow matrix column-wise to the original data. It is a portmanteau of "NA" and "tabular". *Nabular* data explicitly links missing values to data, keeping corresponding observations together and removing the possibility of mismatching records (Figure 3). *Nabular* data facilitates visualization and summaries by allowing the user to reference the missingness of a variable in a coordinated way: the missingness of "var", as "var_NA" during analysis. *Nabular* data is a snapshot of the missingness of the data. This means when *nabular* data are imputed, those imputed values can easily be identified in analysis ("var" vs "var_NA"). Imputing values on *nabular* data automatically tracks these imputations. Together these features make it possible to do tasks that are not possible by simply adding a binary missing matrix, B during data analysis.

Nabular data is similar to classical data formats with quality or flag variables associated with each measured variable, e.g., Scripps CO2 data (Keeling *et al.* 2005), GHCN data (Durre, Menne, and Vose 2008). Using additional columns to represent missingness information follows best practices for data organization, described in (Ellis and Leek 2018) and (Broman and Woo 2018): (1) keep one thing in a cell and (2) describe additional features of variables in a second column. Here they suggest to indicate censored data with an extra variable called "VariableNameCensored", which would be `TRUE` if censored, otherwise `FALSE`. This information can now be represented in the shadow columns as special missing values. Encoding special missing values is achieved by defining logical conditions and suffixes. This is implemented with the `recode_shadow()` function in **naniar** (discussed in Section 4.3).

Special missing values are not a new idea, and have been implemented in other statistical programming languages, SPSS, SAS, and Stata. These typically represent missing values as a

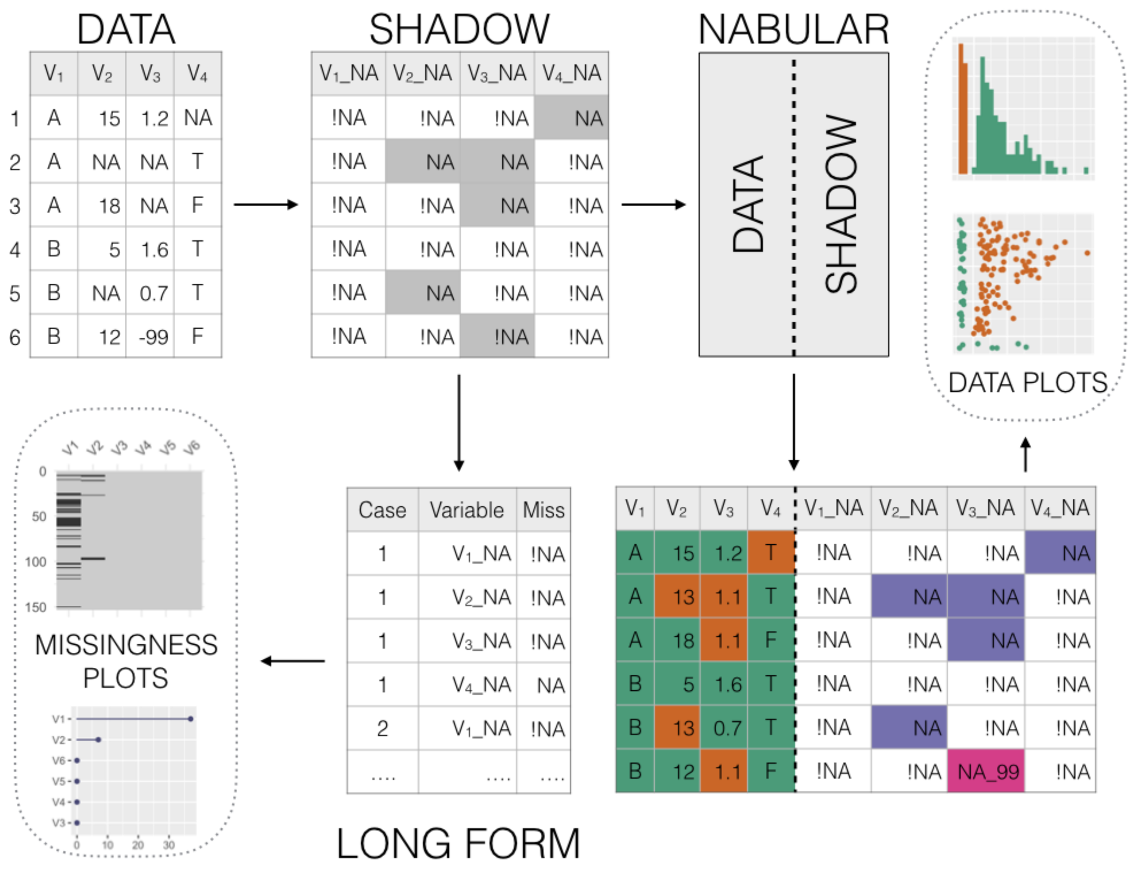


Figure 3: The process of creating nabular data. Data transformed to shadow matrix, where values are either not missing or missing: "!NA" or "NA". The shadow matrix can be converted to long form to create missingness summary plots. Nabular data is created by binding the columns of the data and shadow matrix. Special missing values (such as "−99") are identified as special missings, and values imputed and tracked. Nabular data can be used to identify imputations and explore data values alongside missings, providing a useful format for missing data exploration and analysis.

full-stop, ".", and record special missing values as ".a" – ".z". These special values from these languages break the tidy principle of one column having one type of value, as they record both the value, and the multivariate missingness state.

4.3. Missing data operations

Common missing data operations can be considered verbs, in the tidyverse sense. For missing data, these include: "*scan*", "*replace*", "*add*", "*shadow*", "*impute*", "*track*", and "*flag*". Data can be "*scanned*" to find possible missings not coded as NA. These values can then be "*replaced*" with NA. To facilitate exploration, summaries of missingness can be "*added*" as a summary column to the original data. The data can be augmented with the "*shadow matrix*" values, helping explore missing data, as well as facilitating the process of "*imputing*", and "*tracking*". Finally, unusual or specially coded missing values can be "*flagged*".

Function	Adds column which
`add_n_miss(data)`	contains the number missing values in a row
`add_any_miss(data)`	contains whether there are any missing values in a row
`add_prop_miss(data)`	contains the proportion of missing values in a row
`add_miss_cluster(data)`	contains the missing value cluster

Table 1: Overview of the `add` functions in **naniar**

.

"scan": Searching for common missing value labels

This operation is used to search the data for specific conventional representations of missings, such as, "N/A", "MISSING", "−99". This is implemented in the function `miss_scan_count()`, which returns a table of occurrences of that value for each variable. A list of common NA values for numbers and characters, can be provided to help check for typical representations of missings. These are implemented in **naniar** as objects '`common_na_numbers`' and '`common_na_strings`'.

"replace": Replacing values with missing values

Once possible missing values have been identified, these values can be replaced. For example, a dataset could have the values "−99" meaning a missing value. **naniar** implements replacement with the function, `replace_with_na()`. Values "−99" could be replaced in the "x" column with: `replace_with_na(dat_ms, replace = list(x = -99))`. For operating on multiple variables, there are scoped variants for `replace_with_na()`: `_all`, `_if`, and `_at`. This means `replace_with_na_all()` operates on **all** columns, `replace_with_na_at()` operates **at** specific columns, and `replace_with_na_if()` makes a conditional change on columns **if** they meet some condition (such as `is.numeric()` or `is.character()`).

"add": Adding missingness summary variables

Understanding the missingness structure can be improved by adding summary information alongside the data. For example, in (Tierney *et al.* 2015), the proportion of missings in a row is used as the outcome in a model to identify variables important in predicting missingness structures. **naniar** implements a series of functions to add these missingness summaries to the data, starting with `add_`. These are inspired by the `add_count()` function in **dplyr**, which adds count information for specified groups or conditions. **naniar** provides operations to add the number or proportion of missingness, the missingness cluster, or the presence of any missings, with: `add_n_miss()`, `add_prop_miss()`, `add_miss_cluster()`, and `add_any_miss()`, respectively (Table 1). There are also functions for adding information about shadow values, and readable labels for any missing values with `add_label_shadow()` and `add_label_missings()`.

"shadow": Creating nabular data

Nabular data has the shadow matrix column-bound to existing data. This facilitates visualization and summaries, and allows for imputed values to be tracked. *Nabular* data can be created with `nabular()`:

```
R> nabular(dat_ms)

# A tibble: 5 x 6
      x y         z x_NA  y_NA  z_NA
  <dbl> <chr> <dbl> <fct> <fct> <fct>
1     1 A      -100 !NA   !NA   !NA
2     3 N/A     -99 !NA   !NA   !NA
3    NA <NA>    -98 NA    NA    !NA
4   -99 E      -101 !NA   !NA   !NA
5   -98 F        -1 !NA   !NA   !NA
```

"flag": Describing different types of missing values

Unusual or spurious data values are often identified and "flagged". For example, there might be special codes to mark an individual dropping out of a study, known instrument failure in weather instruments, or for values censored in analysis. **naniar** provides tools to encode these special types of missingness in the shadow matrix of *nabular* data, using `recode_shadow()`. This requires specifying the variable to contain the flagged value, the condition for flagging, and a suffix. This is then recoded as a new factor level in the shadow matrix, so every column is aware of all possible new values of missingness. For example, "-99" could be recoded to indicate a broken machine sensor for the variable "x" with the following:

```
R> nabular(dat_ms) %>%
+    recode_shadow(x = .where(x == -99 ~ "broken_sensor"))

# A tibble: 5 x 6
      x y         z x_NA             y_NA  z_NA
* <dbl> <chr> <dbl> <fct>            <fct> <fct>
1     1 A      -100 !NA              !NA   !NA
2     3 N/A     -99 !NA              !NA   !NA
3    NA <NA>    -98 NA               NA    !NA
4   -99 E      -101 NA_broken_sensor !NA   !NA
5   -98 F        -1 !NA              !NA   !NA
```

"impute": Imputing values

naniar does not reinvent the wheel by creating the need for customized imputation procedures. Instead, the `nabular` format means existing methods using imputation replacement in place now automatically track missing values. **naniar** provides a few imputation methods to facilitate exploration and visualization: `impute_below()`, `impute_mean()`, and `impute_median()`. While useful to explore structure in missingness, they are not recommended for use in analysis. `impute_below()` imputes values below the minimum value, with some controllable jitter (random noise) to reduce overplotting.

Similar to **simputation**, each `impute_` function returns the data with values imputed. However, **naniar** does not use a formula syntax, instead each function has "scoped variants" `_all`, `_at` and `_if`. `impute_` functions with no scoped variant, (`impute_mean()`), will work on

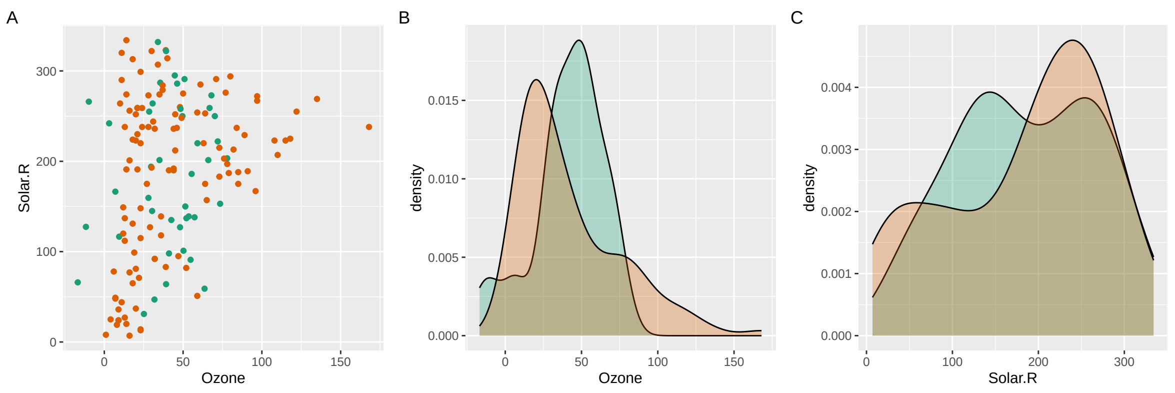


Figure 4: Scatterplot (A) and density plots (B and C) of ozone and solar radiation from the airquality dataset containing imputed values from a linear model. Imputed values are colored green, and data values orange. Imputed values are similar, but slightly trended to the mean.

a single vector, but not a data.frame. One challenge with this approach is imputed value locations are not tracked. This issue is resolved with *nabular* data covered in the following section.

"track": Shadow and impute missing values

To evaluate imputations they need to be tracked. This is achieved by first using *nabular* data, then imputing, and imputed values can then be referred to by their shadow variable, "_NA" (Figure 4). The code below shows the track pattern, first using `nabular()`, then imputing with `impute_lm()`. `label_shadow()` then adds a label to facilitate identifying missings:

```
R> aq_imputed <- nabular(airquality) %>%
+   as.data.frame() %>%
+   simputation::impute_lm(Ozone ~ Temp + Wind) %>%
+   simputation::impute_lm(Solar.R ~ Temp + Wind) %>%
+   add_label_shadow()
R> head(aq_imputed)
```

```
       Ozone  Solar.R Wind Temp Month Day Ozone_NA Solar.R_NA Wind_NA Temp_NA
1   41.00000 190.0000  7.4   67     5   1      !NA        !NA     !NA     !NA
2   36.00000 118.0000  8.0   72     5   2      !NA        !NA     !NA     !NA
3   12.00000 149.0000 12.6   74     5   3      !NA        !NA     !NA     !NA
4   18.00000 313.0000 11.5   62     5   4      !NA        !NA     !NA     !NA
5  -11.67673 127.4317 14.3   56     5   5       NA         NA     !NA     !NA
6   28.00000 159.5042 14.9   66     5   6      !NA         NA     !NA     !NA
  Month_NA Day_NA any_missing
1      !NA    !NA Not Missing
2      !NA    !NA Not Missing
3      !NA    !NA Not Missing
4      !NA    !NA Not Missing
5      !NA    !NA     Missing
6      !NA    !NA     Missing
```

`any_missing`	`min`	`mean`	`median`	`max`
Missing	−16.86418	41.22494	45.4734	78
Not Missing	1.00000	42.09910	31.0000	168

Table 2: Output of **dplyr** summary statistics of imputed vs non imputed values for the variable "Ozone". The `any_missing` column denotes imputed values ("Missing", since they were previously missing), and non-imputed values ("Not Missing"). The mean and median values are similar, but the minimum and maximum values are very different.

Multiple missing or imputed values can be mapped to a graphical element in **ggplot2** by setting the `color` or `fill` aesthetic in ggplot to `any_missing`, a result of the `add_label_shadow()` function. (Figure 4). Imputed values can also be compared to complete case data, grouping by `any_missing`, and then summarizing, similar to other **dplyr** summary workflows shown below in Table 2, showing similarities and differences in imputation methods.

```
R> aq_imputed %>%
+   group_by(any_missing) %>%
+   summarise_at(.vars = vars(Ozone), .funs = lst(min, mean, median, max))
```

5. Graphics

Missing values are often ignored when plotting data - which is why the data visualization software, **MANET**, was named and is an acronym corresponding to "missings are now equally treated" (Unwin *et al.* 1996). However, plots can help to identify the type of missing value patterns, even those of MCAR, MAR or MNAR. Here we summarize how to systematically explore missing patterns visually, and define useful plots to make, relative to the *nabular* data structure.

5.1. Overviews

The first step is to get an overview of the extent of the missingness. Overview visualizations for variables and cases are provided with `gg_miss_var()` and `gg_miss_case()` (Figure 5A), drawing attention to the amount of missings, and ordering by missingness. The "airquality" dataset (from base R), is shown, and contains daily air quality measurements in New York, from May to September, 1973. We learn from Figure 5A–B, that two variables contain missings, approximately one third of observations have one missing value, and a tiny number of observations are missing across two variables. These overview plots are created from the shadow matrix in long form (Figure 3). Numerical statistics can also be reported (Section 6).

The shadow matrix can be put into long form, allowing both the variables and cases to be displayed using a heatmap style visualization, with `vis_miss()` from **visdat** (Tierney 2017) (Figure 3). This also provides numerical summaries of missingness in the legend, and for each column (Figure 5C–D). Clustering can be applied to the rows, and columns arranged in order of missingness (Figure 5D). Similar visualizations are available in other packages such as **VIM**, **mi**, **Amelia**, and **MissingDataGUI**. A key improvement is `vis_miss()` orients the visualization analogous to a regular data structure: variables form columns and are named

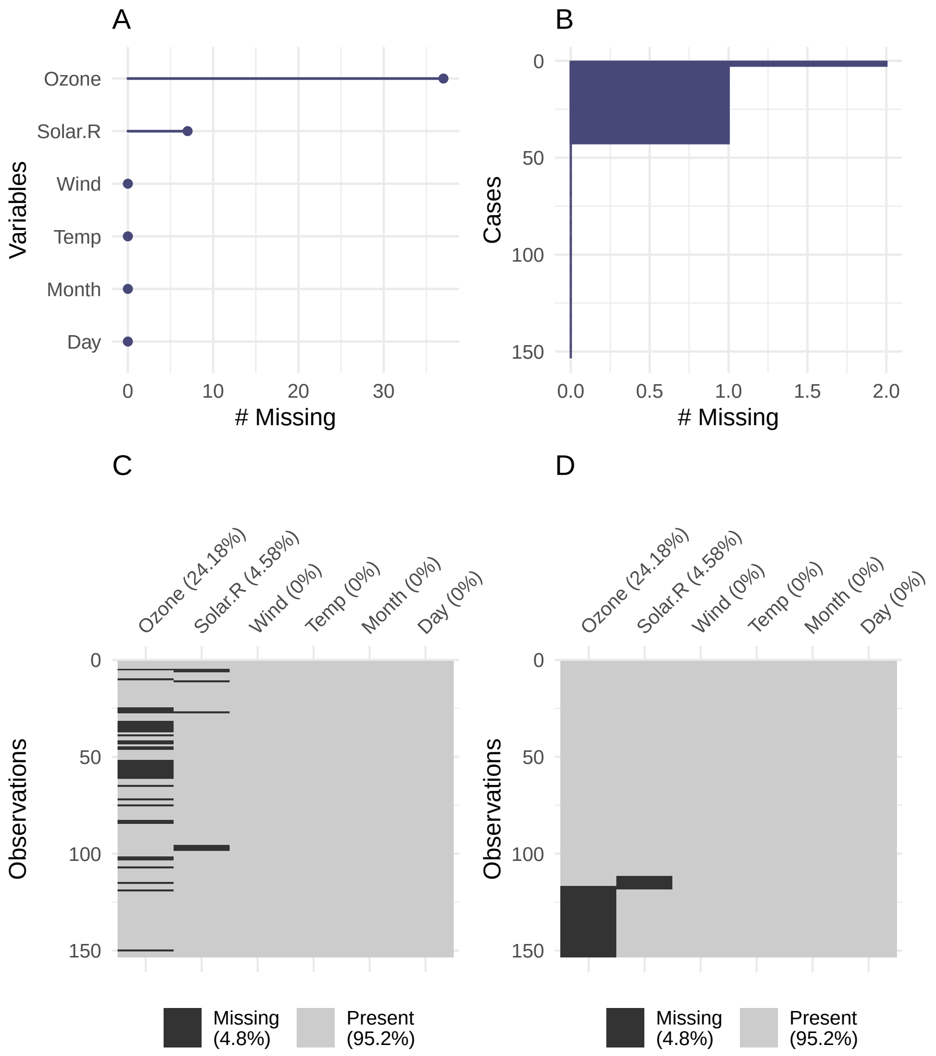


Figure 5: Graphical summaries of missingness in the airquality data. Missings in variables (A) and cases (B), and for an overview of all missingness as a heatmap in (C), and with clustering applied (D). There are missing values in "Ozone" and "Solar.R", with "Ozone" having more missings. Not many cases have two missings. Most missingness is from cases with one missing value. The default output (C) and ordered by clustering on rows and columns (D). These overviews are made possible using the shadow matrix in long form. There are only missings in ozone and solar radiation, and there appears to be some structure to their missingness.

at the top, and each row is an observation. Using **ggplot2**, as the foundation, makes the plot easily customizable.

The number of times observations are missing together can be visualized using an "upset plot" (Conway, Lex, and Gehlenborg 2017). An alternative to a Venn diagram, an upset plot shows the size and features of overlapping sets, and scales well with more variables. An upset plot can be constructed from the shadow matrix, as shown in Figure 6 which displays the overlapping counts of missings in the airquality data. The bottom right shows the combinations of missingness, the top panel shows the size of these combinations, and

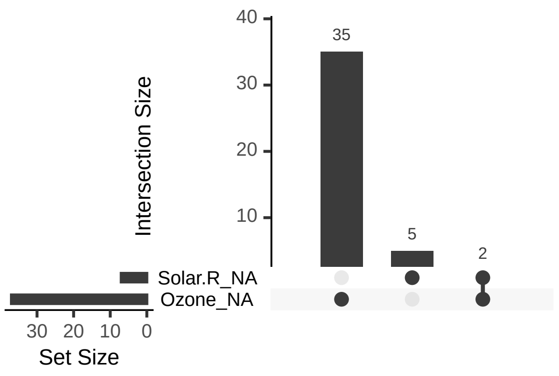


Figure 6: The pattern of missingness in the airquality dataset shown in an upset plot. Only "Ozone" and "Solar.R" have missing values, and "Ozone" has the most missing values. There are 2 cases where both "Solar.R" and "Ozone" have missing values.

the bottom left shows missingness in each variable. This provides similar information to Figure 5A–D, but more clearly illustrating overlapping missingness, where 2 cases are missing together in variables "Solar.R" and "Ozone".

5.2. Univariate

Missing values are typically not shown for univariate visualizations such as histograms or densities. Two ways to use *nabular* data to present univariate data with missings are discussed. The first imputes values below the range to facilitate visualizations. The second displays two plots of the same variable according to the missingness of a chosen variable.

Imputing values below the range

To visualize the amount of missings in each variable, the data is transformed into *nabular* form, then values are imputed below the range of data using `impute_below_all()`. Figure 7A shows a histogram of "Ozone" values on the right in green, and the histogram of missing "Ozone" values on the left, in orange. The missings in "Ozone" are imputed and "Ozone_NA" is mapped to the fill aesthetic. *Nabular* data facilitates adding counts of missingness to a histogram, allowing examination of a variable's distribution of values, and also the magnitude of missings.

Univariate split by missingness

The distribution of a variable can be shown according to the missingness of another variable. The shadow matrix part of *nabular* is used to handle the faceting, and color mapping. Figure 7 shows the values of temperature when ozone is present, and missing, using a faceted histogram (B), and an overlaid density (C). This shows how values of temperature are affected by the missingness of "Ozone", and reveals a cluster of low temperature observations with missing "Ozone" values. This type of plot can facilitate exploring missing data distributions. For example, we would expect if data were MCAR, for values to be roughly uniformly missing throughout the histogram or density.

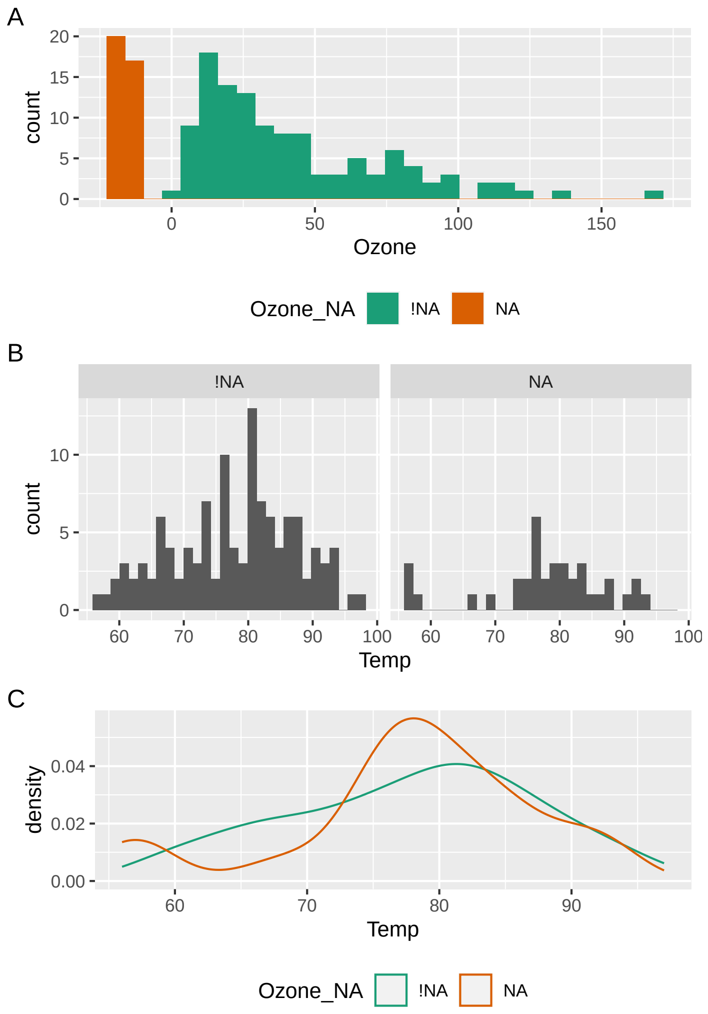


Figure 7: Univariate summaries of missingness. (A) A histogram using nabular data to show the values and missings in “Ozone”. Values are imputed below the range to show the number of missings in “Ozone”and colored according to missingness of “Ozone” (“Ozone_NA”). There are about 35 missings in “Ozone”. Panel C shows temperature according to missingness in “Ozone” from in the airquality dataset. A histogram of temperature faceted by the missingness of “Ozone” (B), or a density of temperature colored by missingness in “Ozone” (C). These show a cluster of low temperature observations with missing “Ozone” values, but temperature is otherwise similar.

5.3. Bivariate

To visualize missing values in two dimensions the missing values can be placed in plot margins, by imputing values below the range of the data. Using *nabular* data identifies imputed values, and color makes missingness pre-attentive (Treisman 1985). The steps of imputing and coloring are combined into `geom_miss_point()`. Figure 8 shows a mostly uniform spread of missing values for “Solar.R” and “Ozone”. As `geom_miss_point()` is a defined **ggplot2** geometry, it works with features such as faceting and mapping other variables to graphical aesthetics.

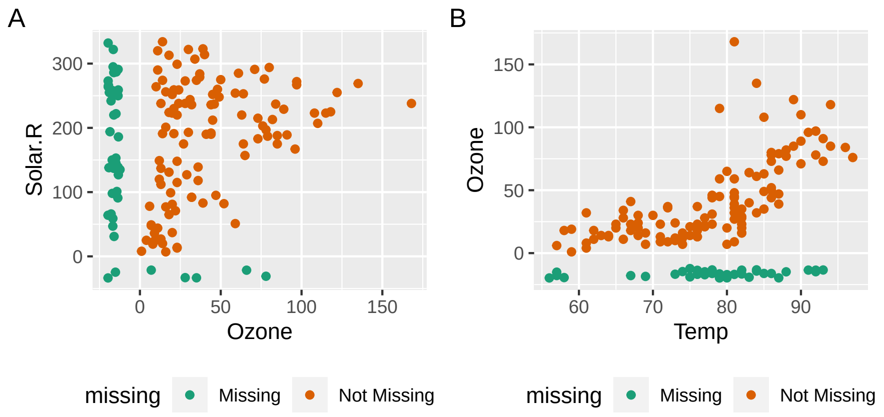


Figure 8: Scatterplots with missings displayed at 10% below for the airquality dataset. Scatterplots of ozone and solar radiation (A), and ozone and temperature (B). There are missings in ozone and solar radiation, but not temperature.

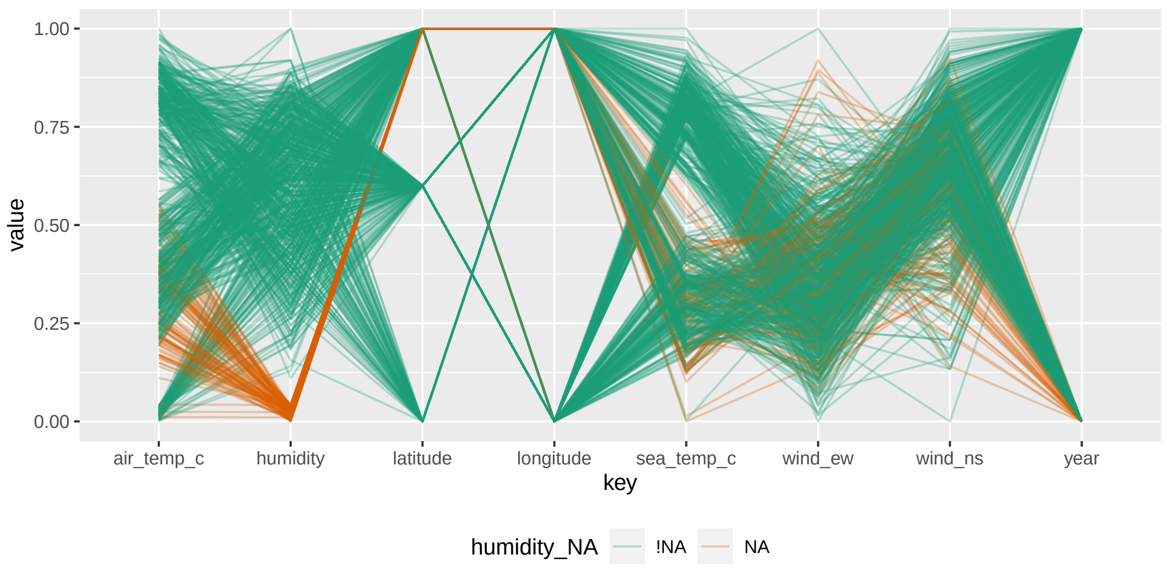


Figure 9: Parallel coordinate plot shows missing values imputed 10% below range for the oceanbuoys dataset. Values are colored by missingness of humidity. Humidity is missing for low air and sea temperatures, and is missing for one year and one location.

5.4. Multivariate

Parallel coordinate plots can help to visualize missingness beyond two dimensions. They transform variables to the same scale, ranging between 0 and 1. The `oceanbuoys` dataset from **naniar** is used for this visualization, containing measurements of sea and air temperature, humidity, and east west and north south wind speeds. Data was collected in 1993 and 1997, to understand and predict El Niño and El Niña. Figure 9 is a parallel coordinate plot of `oceanbuoys`, with missing values imputed to be 10% below the range, and values colored according to whether humidity was missing ("humidity_NA"). Figure 9 shows humidity is missing at low air and sea temperatures, and humidity is missing in one year, and one location.

6. Numerical summaries

This section describes approaches to summarizing missingness, and an implementation in **naniar**. Numerical summaries should be easy to remember with consistent names and output, returning either a single number (Section 6.1), or a dataframe (Section 6.2), so they integrate well with plotting and modeling tools. How these work with other tools in an analysis pipeline is shown in (Section 6.3).

6.1. Single number summaries

The overall number, proportion, or percent of missing values in a dataset should be simple to calculate. **naniar** provides the functions `n_miss()`, `prop_miss()` and `pct_miss()`, as well as their complements. Summaries for variables and cases are made by appending `_case` or `_var` to these summaries. An overview is shown in Table 3.

6.2. Summaries and tabulations of missing data

Presenting the number and percent of missing values for each variable, or case, provides a summary usable in data handling, or in models to inform imputation. For example, potentially dropping variables, or deciding to include others in an imputation model. Another useful approach is to tabulate the frequency of missing values for each variable or case; that is, the number of times there are zero missings, one missing, two, and so on. These summaries and tabulations are shown for variables in Tables 4 and 5, and implemented with `miss_var_summary()` and `miss_var_table()`. Case-wise (row-wise) summaries and tabulations are implemented with `miss_case_summary()` and `miss_case_table()`.

These summaries order rows by the number of missings (`n_miss()`), to show the most missings at the top. The number of missings across a repeating span, or finding "runs" or "streaks"

Missing function	Missing value	Complete function	Complete value
`n_miss`	44.00	`n_complete`	874.00
`prop_miss`	0.05	`prop_complete`	0.95
`pct_miss`	4.79	`pct_complete`	95.21
`pct_miss_case`	27.45	`prop_complete_case`	72.55
`pct_miss_var`	33.33	`pct_complete_var`	66.67

Table 3: Single number summaries of missingness and completeness of the airquality dataset. The functions follow consistent naming, making them easy to remember, and their use clear.

`Variable`	`n_miss`	`pct_miss`
Ozone	37	24.2
Solar.R	7	4.6
Wind	0	0.0
Temp	0	0.0
Month	0	0.0
Day	0	0.0

Table 4: `miss_var_summary(airquality)` provides the number and percent of missings in each variable in airquality. Only ozone and solar radiation have missing values.

`n_miss_in_var`	`n_vars`	`pct_vars`
0	4	66.7
7	1	16.7
37	1	16.7

Table 5: The output of `miss_var_table(airquality)`, tabulating the amount of missing data in each variable in airquality. This shows the number of variables with 0, 7, and 37 missings, and the percentage of variables with those amounts of missingness. There are few missingness patterns.

`Month`	`Variable`	`n_miss`	`pct_miss`
5	Ozone	5	16.1
5	Solar.R	4	12.9
5	Wind	0	0.0
5	Temp	0	0.0
5	Day	0	0.0
6	Ozone	21	70.0
6	Solar.R	0	0.0
6	Wind	0	0.0
6	Temp	0	0.0
6	Day	0	0.0

Table 6: Output of `airquality %>% group_by(Month) %>% miss_var_summary()` provides a grouped summary of the missingness in each variable, for each month of the airquality dataset. Only the first 10 rows are shown. There are more ozone missings in June than May.

of missings in given variables can also be useful for identifying missingness patterns, and are implemented with `miss_var_span()`, and `miss_var_run()`.

6.3. Combining numerical summaries with grouping operations

It is useful to explore summaries and tabulations within groups of a dataset. **naniar** works with **dplyr**'s `group_by()` operator to produce grouped summaries, which work well with the "pipe" operator. Table 6 shows an example of missing data summaries for airquality, grouped by month.

7. Application

This section shows how the methods described so far are used together in a data analysis workflow. We analyse a case study of housing data for the city of Melbourne from January 28, 2016 to March 17, 2018. The data was compiled by scraping weekly property clearance data (Pino and Tierney 2019). There are $27,247$ properties, and 21 variables in the dataset. The variables include real estate type (town house, unit, house), suburb, selling method, number of rooms, price, real estate agent, sale date, and distance from the central business district.

The goal in analyzing this data is to accurately predict Melbourne housing prices. The data contains many missing values. As a precursor to building a predictive model, this analysis focuses on understanding the patterns of missingness.

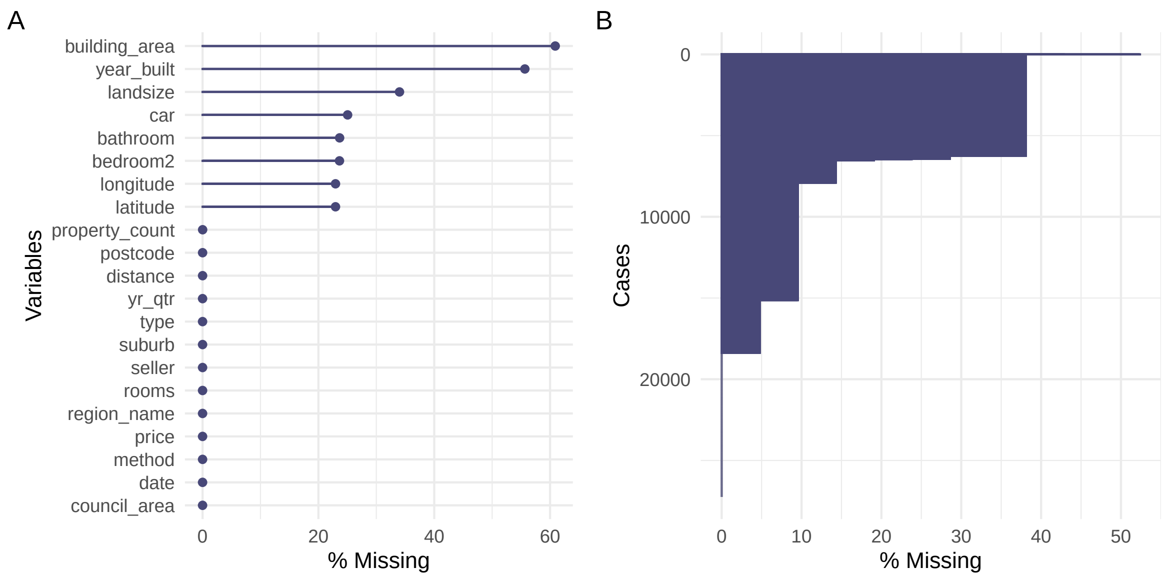


Figure 10: The amount of missings in variables (A) and cases (B) for Melbourne housing data. (A) Build area and year built have more than 50% missing, and car, bathroom, bedroom2 and longitude and latitude have about 25% missings. (B) Cases are missing $5 - 50\%$ of values. The majority of missingness is in selected cases and variables.

7.1. Exploring patterns of missingness

Figure 10A shows 9 variables with missing values. The most missings are in "building area", followed by "year built", and "land size", with similar amounts of missingness in "Car", "bathroom", "bedroom2", "longitude", and "latitude". Figure 10B reveals there are up to 50% missing values in cases, and the majority of cases have more than 5% values missing. The variables "building area" and "year built" have more than 50% missing data, and so could perhaps be omitted from subsequent analysis, as imputed values are likely to be spurious. Three missingness clusters are revealed by visualizing missingness in the whole dataset, clustering and arranging the rows and columns of the data (Figure 11).

Figure 12 shows missingness patterns with an upset plot (Conway *et al.* 2017), displaying 8 intersecting sets of missing variables. Two patterns stand out: two, and five variables missing, providing further evidence of the missingness patterns seen in Figures 10 and 12.

Tabulating the number of missings in variables in Table 7 (left) shows three groups of missingness. Tabulating missings in cases (Table 7, right) shows six patterns of missingness. These overview plots lead to the removal of two variables from with more than 50% missingness from analysis: "building area" and "year built".

7.2. Exploring missingness patterns for imputation

Using information from Section 7.1, the following variables are explored for features predicting missingness: "latitude", "longitude", "bedroom2", "bathroom", "car", and "land size".

Missingness structure is explored by clustering the missing values into groups. Then, a classification and regression trees (CART) model is applied to predict these missingness clusters using the remaining variables (Tierney *et al.* 2015; Barnett *et al.* 2017). Two clusters of miss-

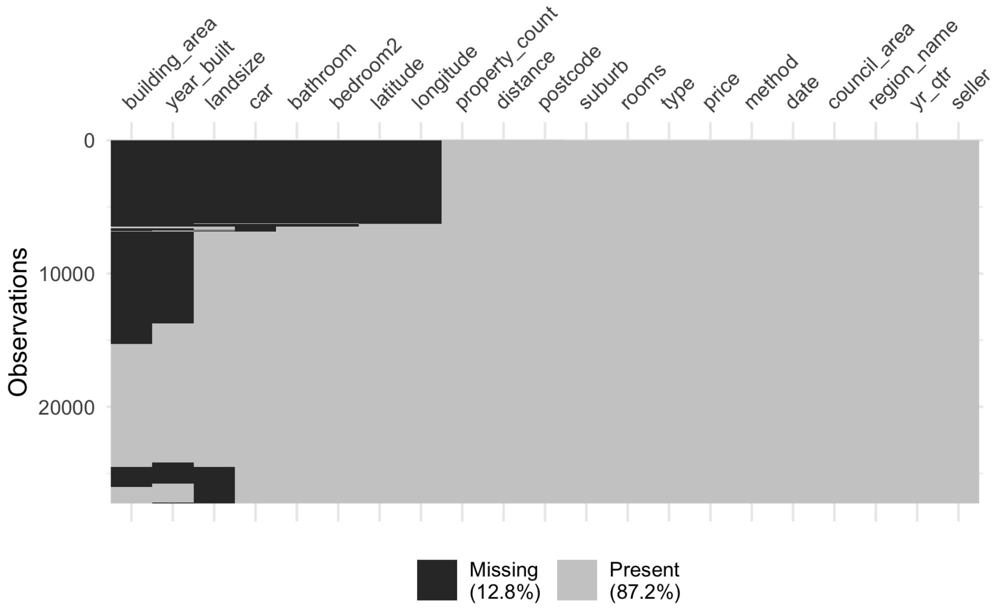


Figure 11: Heatmap of clustered missingness for housing data reveals structured missingness. Three groups of missingness are apparent. At the top: building area to longitude; the middle: building area and year built; the end: building area, year built, and landsize.

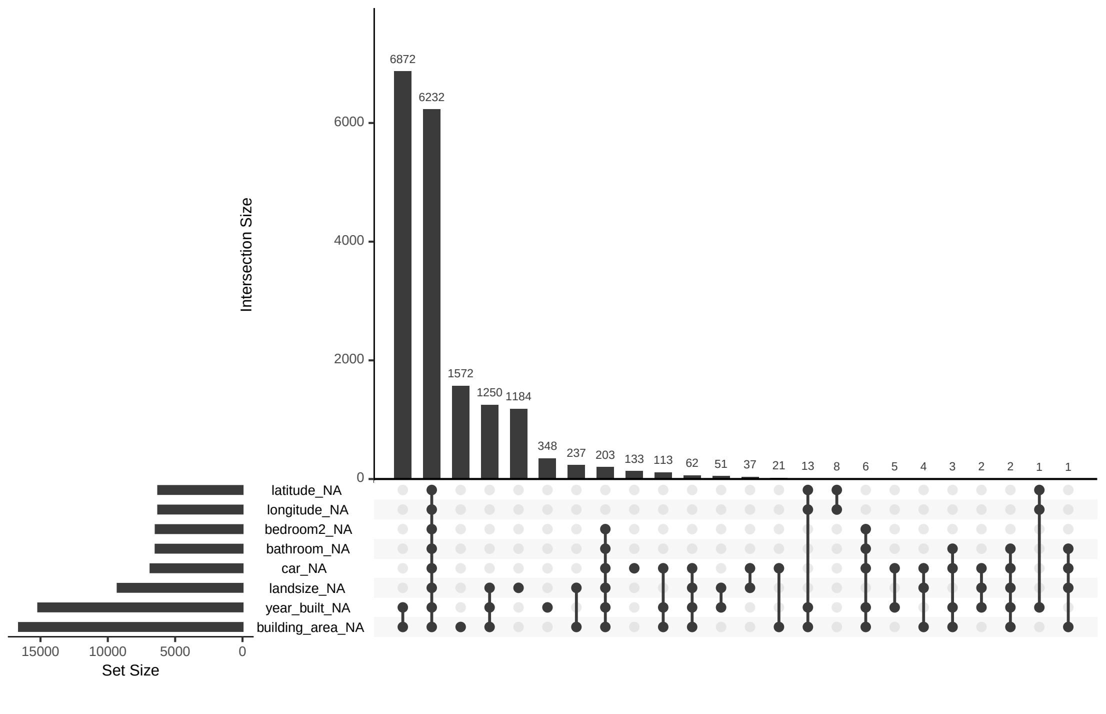


Figure 12: An upset plot of 8 sets of missingness in the housing data. Missingness for each variable is shown on the bottom left. Connected dots show co-occurrences of missings in variables. Two missingness patterns are clear, year built and building area, and latitude through to building area.

n_miss_in_var	n_vars	pct_vars
0	10	47.6
1	2	9.5
3	1	4.8
6254	2	9.5
6441	1	4.8
6447	1	4.8
6824	1	4.8
9265	1	4.8
15163	1	4.8
16591	1	4.8

n_miss_in_case	n_cases	pct_cases
0	8887	32.6
1	3237	11.9
2	7231	26.5
3	1370	5.0
4	79	0.3
5	8	0.0
6	203	0.7
8	6229	22.9
9	2	0.0
11	1	0.0

Table 7: Summary tables to help understand missingness patters. Output of `miss_var_table(housing)` (left), tabulating missingness for variables, and output of `miss_case_table(housing)`. There are 13 variables with $0-3$ missings, 6 variables have $6,000-10,000$ missings, 2 variables have $15,000-17,000$ missings. About 30% of cases have no missings, 45% of cases have $1-6$ missings, and about 23% of cases have 8 or more missings. There are different patterns of missingness in variables and cases, but they can be broken down into smaller groups.

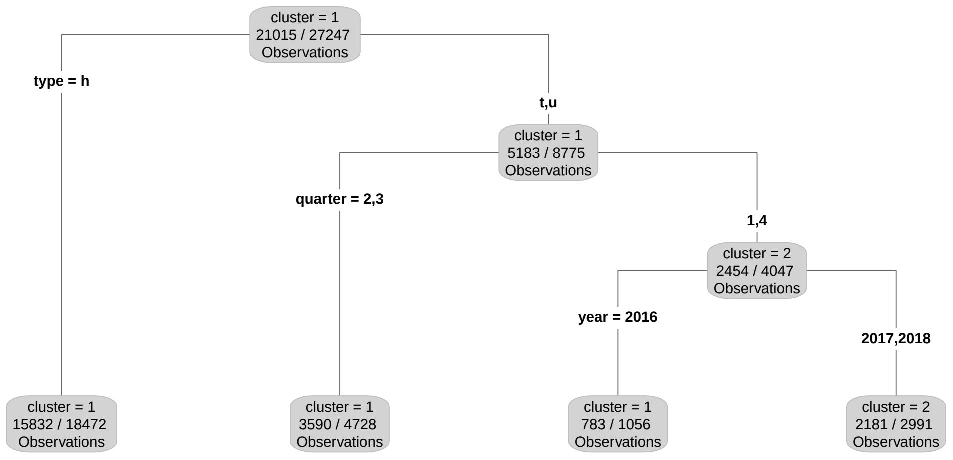


Figure 13: Decision tree output predicting missingness clusters. Type of house, year quarter, and year were important for predicting missingness cluster. The cluster with the most missingness was for quarters 1 and 4, for 2017 and 2018. Type of house, year, and year quarter are important features related to missingness structure.

ingness are identified and predicted using all variables in the dataset with the CART package **rpart** (Therneau and Atkinson 2022), and plotted using the **rpart.plot** package (Milborrow 2022). Importance scores reveal the following variables as most important in predicting missingness: "rooms", "price", "suburb", "council area", "distance", and "region name". These variables are important for predicting missingness, so are included in the imputation model.

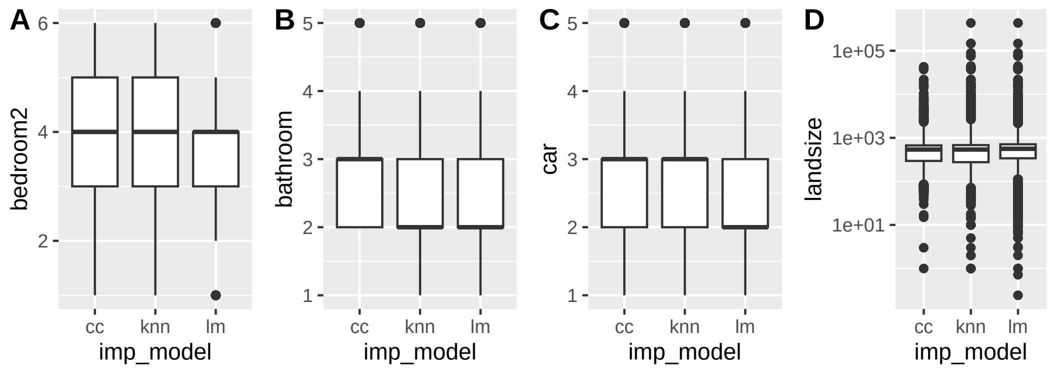


Figure 14: Boxplots of complete case data, and data imputed with KNN or linear model for different variables. (A) number of bedrooms, (B) number of bathrooms, (C) number of carspots, and (D) landsize (on a log10 scale). KNN had similar results to complete case, and linear model had a lower median for cars and fewer extreme values for bedrooms.

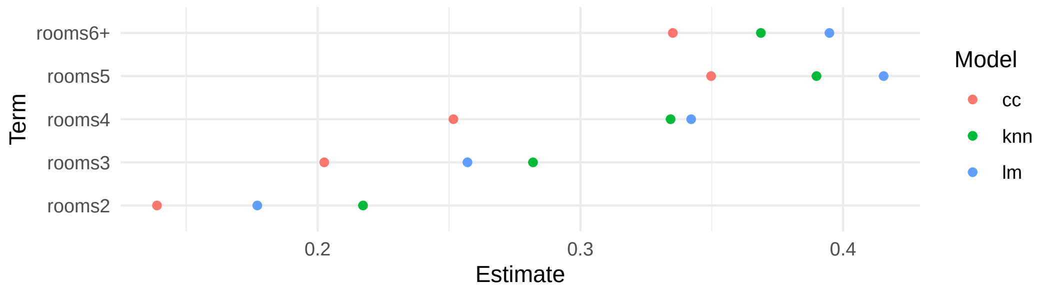


Figure 15: The coefficient estimate for the number of rooms varies according to the imputed dataset. Complete case dataset produced lower coefficients, compared to imputed datasets.

7.3. Imputation and diagnostics

simputation is used to implement two imputations: simple linear regression and K nearest neighbors. Values are imputed stepwise in ascending order of missingness. The track missings pattern is applied (described in Section 4.3), to assess imputed values. Imputed datasets are compared on their performance in a model predicting log house price for 4 variables (Figure 14). Compared to k-nearest neighbors (KNN) imputed values, the linear model imputed values closer to the mean.

7.4. Assessing model predictions

Coefficients of the linear model of log price vary for room for different imputed datasets (Figure 15). Notably, complete cases underestimate the impact of room on log price. A partial residual plot (Figure 16) shows there is not much variation amongst the models from the differently imputed datasets.

7.5. Summary

The **naniar** and **visdat** packages implement the methods discussed in the paper, building on existing tidy tools and strike a compromise between automation and control, making

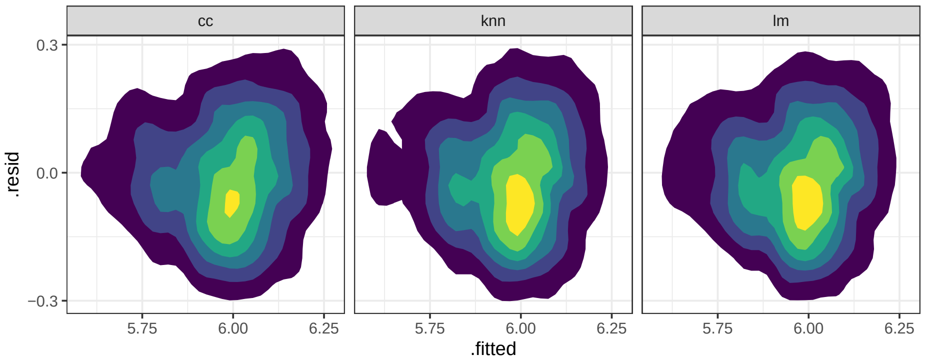


Figure 16: Partial residual plot for each data set, complete cases (cc), and imputed with KNN (knn) or linear model (lm). These are plotted as two dimensional filled density plots with 7 bins estimated. The bins are colored according to concentration of points in that area – brighter colors indicated higher concentration. Compared to complete cases, imputed data has more points clustered closer to 0 for residuals, and and around 6 fitted values.

analysis efficient, readable, but not overly complex. Each tool has clear intent and effects – summarizing, plotting or generating data or augmenting data in some way. This not only reduces repetition and typing in an analysis, but most importantly, allows for clear expression of intent, making exploration of missing values fluent.

8. Discussion

This paper has described new methods for exploring, visualizing, and imputing missing data. The work was motivated by recent developments of tidy data, and extends them for better missing value handling. The methods have standard outputs, function arguments, and behavior. This provides consistent workflows centered around data analysis that integrate well with existing imputation methodology, visualization, and modeling.

The *nabular* data structures discussed in the paper are simple by design, to promote flexibility. They could be used to create different visualizations than were shown in the paper. The analyst can use the data structures to decide on appropriate visualization for their problem. The data structures could also be used to support interactive graphics, in the manner of **MANET** and **GGobi**. Linking the plots (via linked brushing) could facilitate exploration of missingness, and could be implemented with **plotly** (Sievert 2020) for added interactivity. Animating between different sets of imputed values could also be possible with packages like **gganimate** (Pedersen and Robinson 2020).

Other data structures such as spatial data, time series, networks, and longitudinal data would be supported by the inherently tabular, *nabular* data, if they are first structured as wide tidy format. Large data may need special handling, and additional features like efficient storage of purely imputed values and lazy evaluation. Special missing value codes could be improved by creating special classes, or expanding low level representation of NA at the source code level. Similarly, list columns, present a challenge for analysis as their expansion into vectors in a dataframe is not always possible as they have different lengths, but should still be flagged for further analysis.

The methodology described in this paper can be used in conjunction with other approaches to understand multivariate missingness dependencies (e.g., decision trees (Tierney *et al.* 2015), latent group analysis (Barnett *et al.* 2017), and PCA (Lê, Josse, and Husson 2008)). Evaluating imputed values using a testing framework like (van Buuren 2018) is also supported.

The approach meshes with the dynamic nature of data analysis, allowing the analyst to go from raw data to model data in a fluid workflow.

Acknowledgments

The authors would like to thank Miles McBain, for his key contributions and discussions on the **naniar** package, in particular for helping implement `geom_miss_point()`, and for his feedback on ideas, implementations, and names. We also thank Colin Fay for his contributions to the **naniar** package, in particular for his assistance with the `replace_with_na()` functions. We also thank Earo Wang and Mitchell O'Hara-Wild for the many useful discussions on missing data and package development, and for their assistance with creating elegant approaches that take advantage of the tidy syntax. We would also like to thank those who contributed pull requests and discussions on the **naniar** package, in particular Jim Hester and Romain François for improving the speed of key functions, Ross Gayler for discussion on special missing values, and Luke Smith for helping **naniar** be more compliant with **ggplot2**. We would also like to thank Amelia McNamara for discussions on the paper.

References

Barnett AG, McElwee P, Nathan A, Burton NW, Turrell G (2017). "Identifying Patterns of Item Missing Survey Data Using Latent Groups: An Observational Study." *BMJ Open*, **7**(10), e017284. doi:10.1136/bmjopen-2017-017284.

Bilogur A (2018). "**Missingno**: A Missing Data Visualization Suite." *Journal of Open Source Software*, **3**(22), 547. doi:10.21105/joss.00547.

Broman KW, Woo KH (2018). "Data Organization in Spreadsheets." *The American Statistician*, **72**(1), 2–10. doi:10.1080/00031305.2017.1375989.

Cheng X, Cook D, Hofmann H (2011). ***MissingDataGUI**: A GUI for Missing Data Exploration.* R package version 0.1-1, URL https://CRAN.R-project.org/src/contrib/Archive/MissingDataGUI/.

Cheng X, Cook D, Hofmann H (2015). "Visually Exploring Missing Values in Multivariable Data Using a Graphical User Interface." *Journal of Statistical Software*, **68**(1), 1–23. doi:10.18637/jss.v068.i06.

Conway JR, Lex A, Gehlenborg N (2017). "**UpSetR**: An R package for the Visualization of Intersecting Sets and Their Properties." *Bioinformatics*, **33**(18), 2938–2940. doi:10.1093/bioinformatics/btx364.

Cook D, Swayne DF (2007). *Interactive and Dynamic Graphics for Data Analysis: With R and **GGobi***. Springer-Verlag, New York. doi:10.1007/978-0-387-71762-3.

Dempster AP, Laird NM, Rubin DB (1977). "Maximum Likelihood from Incomplete Data Via the EM Algorithm." *Journal of the Royal Statistical Society B*, **39**(1), 1–22. doi:10.1111/j.2517-6161.1977.tb01600.x.

Donoho D (2017). "50 Years of Data Science." *Journal of Computational and Graphical Statistics*, **26**(4), 745–766. doi:10.1080/10618600.2017.1384734.

Durre I, Menne MJ, Vose RS (2008). "Strategies for Evaluating Quality Assurance Procedures." *Journal of Applied Meteorology and Climatology*, **47**(6), 1785–1791. doi:10.1175/2007jamc1706.1.

Ellis SE, Leek JT (2018). "How to Share Data for Collaboration." *The American Statistician*, **72**(1), 53–57. doi:10.1080/00031305.2017.1375987.

Grolemund G, Wickham H (2011). "Dates and Times Made Easy with **lubridate**." *Journal of Statistical Software*, **40**(1), 1–25. doi:10.18637/jss.v040.i03.

Harrell Jr FE (2022). ***Hmisc**: Harrell Miscellaneous.* R package version 4.7-1, URL https://CRAN.R-project.org/package=Hmisc.

Henry L, Wickham H (2020). ***purrr**: Functional Programming Tools.* R package version 0.3.4, URL https://CRAN.R-project.org/package=purrr.

Honaker J, King G (2010). "What to Do about Missing Values in Time-Series Cross-Section Data." *American Journal of Political Science*, **54**(2), 561–581. doi:10.1111/j.1540-5907.2010.00447.x.

Honaker J, King G, Blackwell M (2011). "**Amelia II**: A Program for Missing Data." *Journal of Statistical Software*, **45**(1), 1–47. doi:10.18637/jss.v045.i07.

IBM Corporation (2022). *IBM SPSS Statistics 29.* IBM Corporation, Armonk. URL https://www.ibm.com/spss/.

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

Keeling CD, Piper SC, Bacastow RB, Wahlen M, Whorf TP, Heimann M, Meijer HA (2005). "Atmospheric CO2 and 13CO2 Exchange with the Terrestrial Biosphere and Oceans from 1978 to 2000: Observations and Carbon Cycle Implications." In IT Baldwin, MM Caldwell, G Heldmaier, RB Jackson, OL Lange, HA Mooney, ED Schulze, U Sommer, JR Ehleringer, M Denise Dearing, TE Cerling (eds.), *A History of Atmospheric CO2 and Its Effects on Plants, Animals, and Ecosystems*, Ecological Studies, pp. 83–113. Springer-Verlag, New York, NY. doi:10.1007/0-387-27048-5_5.

Kowarik A, Templ M (2016). "Imputation with the R Package **VIM**." *Journal of Statistical Software*, **74**(1), 1–16. doi:10.18637/jss.v074.i07.

Kuhn M, Wickham H (2022). ***recipes**: Preprocessing Tools to Create Design Matrices.* R package version 1.0.1, URL https://CRAN.R-project.org/package=recipes.

Lê S, Josse J, Husson F (2008). "**FactoMineR**: An R Package for Multivariate Analysis." *Journal of Statistical Software*, **25**(1), 1–18. doi:10.18637/jss.v025.i01.

Milborrow S (2022). ***rpart.plot**: Plot **rpart** Models: An Enhanced Version of* `plot.rpart`. R package version 3.1.1, URL `https://CRAN.R-project.org/package=rpart.plot`.

Moritz S, Bartz-Beielstein T (2017). "**imputeTS**: Time Series Missing Value Imputation in R." *The R Journal*, **9**(1), 207. `doi:10.32614/rj-2017-009`.

Novo AA, Schafer JL (2022). ***norm**: Analysis of Multivariate Normal Datasets with Missing Values.* R package version 1.0-10.0; Ported to R by Alvaro A. Novo. Original by Joseph L. Schafer, URL `https://CRAN.R-project.org/package=norm`.

Pedersen TL, Robinson D (2020). ***gganimate**: A Grammar of Animated Graphics.* R package version 1.0.7, URL `https://CRAN.R-project.org/package=gganimate`.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D (2011). "**Scikit-Learn**: Machine Learning in Python." *The Journal of Machine Learning Research*, **12**, 2825–2830.

Pino T, Tierney N (2019). *njtierney/Melb-Housing-Data: Melbourne Housing Data Releae.* `doi:10.5281/zenodo.2575484`.

R Core Team (2022). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

Robinson D, Hayes A, Couch S (2022). ***broom**: Convert Statistical Objects into Tidy Tibbles.* R package version 1.0.1, URL `https://CRAN.R-project.org/package=broom`.

Rushworth A (2022). ***inspectdf**: Inspection, Comparison and Visualisation of Data Frames.* R package version 0.0.12, URL `https://CRAN.R-project.org/package=inspectdf`.

SAS Institute Inc (2020). *The SAS System, Version 15.2.* SAS Institute Inc., Cary. URL `https://www.sas.com/`.

Schafer JL (1999). "NORM: Multiple Imputation of Incomplete Multivariate Data under a Normal Model, Version 2." *Software for Windows*, **95**, 98.

Schafer JL, Graham JW (2002). "Missing Data: Our View of the State of the Art." *Psychological Methods*, **7**(2), 147–177. `doi:10.1037/1082-989x.7.2.147`.

Sievert C (2020). *Interactive Web-Based Data Visualization with R, **plotly**, and **shiny**.* 1st edition. Chapman and Hall/CRC, Boca Raton, FL. `doi:10.1201/9780429447273`.

Silge J, Robinson D (2016). "**tidytext**: Text Mining and Analysis Using Tidy Data Principles in R." *The Journal of Open Source Software*, **1**(3), 37. `doi:10.21105/joss.00037`.

StataCorp (2021). *Stata Statistical Software: Release 17.* StataCorp LLC, College Station. URL `https://www.stata.com/`.

Sterne JAC, White IR, Carlin JB, Spratt M, Royston P, Kenward MG, Wood AM, Carpenter JR (2009). "Multiple Imputation for Missing Data in Epidemiological and Clinical Research: Potential and Pitfalls." *BMJ*, **338**. `doi:10.1136/bmj.b2393`.

Su YS, Gelman A, Hill J, Yajima M (2011). "Multiple Imputation with Diagnostics (**mi**) in R: Opening Windows into the Black Box." *Journal of Statistical Software*, **45**(1), 1–31. doi:10.18637/jss.v045.i02.

Swayne DF, Buja A (1998). "Missing Data in Interactive High-Dimensional Data Visualization." *Computational Statistics*, **13**(1), 15–26. doi:10.2307/1390754.

Therneau T, Atkinson B (2022). ***rpart**: Recursive Partitioning and Regression Trees.* R package version 4.1.16, URL https://CRAN.R-project.org/package=rpart.

Tierney N (2017). "**visdat**: Visualising Whole Data Frames." *The Journal of Open Source Software*, **2**(16), 355. doi:10.21105/joss.00355.

Tierney N, Cook D, McBain M, Fay C (2023). ***naniar**: Data Structures, Summaries, and Visualisations for Missing Data.* R package version 1.0.0, URL https://CRAN.R-project.org/package=naniar.

Tierney NJ, Harden FA, Harden MJ, Mengersen KL (2015). "Using Decision Trees to Understand Structure in Missing Data." *BMJ Open*, **5**(6), e007450. doi:10.1136/bmjopen-2014-007450.

Treisman A (1985). "Preattentive Processing in Vision." *Computer Vision, Graphics, and Image Processing*, **31**(2), 156–177. doi:10.1016/s0734-189x(85)80004-9.

Unwin A, Hawkins G, Hofmann H, Siegl B (1996). "Interactive Graphics for Data Sets with Missing Values – **MANET**." *Journal of Computational and Graphical Statistics*, **5**(2), 113–122. doi:10.1080/10618600.1996.10474700.

van Buuren S (2018). *Flexible Imputation of Missing Data.* CRC press. doi:10.1201/9780429492259.

van Buuren S, Groothuis-Oudshoorn K (2011). "**mice**: Multivariate Imputation by Chained Equations in R." *Journal of Statistical Software*, **45**(1), 1–67. doi:10.18637/jss.v045.i03.

van der Loo M (2022). ***simputation**: Simple Imputation.* R package version 0.2.8, URL https://CRAN.R-project.org/package=simputation.

Walker K (2022). ***tidycensus**: Load US Census Boundary and Attribute Data as **tidyverse** and **sf**-Ready Data Frames.* R package version 1.2.2, URL https://CRAN.R-project.org/package=tidycensus.

Wang E, Cook D, Hyndman RJ (2020). "A New Tidy Data Structure to Support Exploration and Modeling of Temporal Data." *Journal of Computational and Graphical Statistics*, **29**(3), 466–478. doi:10.1080/10618600.2019.1695624.

Wickham H (2014). "Tidy Data." *Journal of Statistical Software*, **59**(1), 1–23. doi:10.18637/jss.v059.i10.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis.* 2 edition. Springer-Verlag. doi:10.1007/978-3-319-24277-4.

Wickham H (2022a). ***stringr**: Simple, Consistent Wrappers for Common String Operations.* R package version 1.4.1, URL https://CRAN.R-project.org/package=stringr.

Wickham H (2022b). ***tidyr**: Tidy Messy Data.* R package version 1.2.0, URL https://CRAN.R-project.org/package=tidyr.

Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019). "Welcome to the Tidyverse." *Journal of Open Source Software*, **4**(43), 1686. doi:10.21105/joss.01686.

Wickham H, Bryan J (2022). ***readxl**: Read Excel Files.* R package version 1.4.1, URL https://CRAN.R-project.org/package=readxl.

Wickham H, François R, Henry L, Müller K (2022a). ***dplyr**: A Grammar of Data Manipulation.* R package version 1.0.10, URL https://CRAN.R-project.org/package=dplyr.

Wickham H, Grolemund G (2017). *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data.* 1st edition. O'Reilly Media, Sebastopol, CA.

Wickham H, Hester J, François R (2022b). ***readr**: Read Rectangular Text Data.* R package version 2.1.2, URL https://CRAN.R-project.org/package=readr.

Wickham H, Miller E, Smith D (2022c). ***haven**: Import and Export SPSS, Stata and SAS Files.* R package version 2.5.1, URL https://CRAN.R-project.org/package=haven.

Wilkinson L (2012). "The Grammar of Graphics." In JE Gentle, WK Härdle, Y Mori (eds.), *Handbook of Computational Statistics: Concepts and Methods*, Springer Handbooks of Computational Statistics, pp. 375–414. Springer-Verlag, Berlin, Heidelberg. doi:10.1007/978-3-642-21551-3_13.

Affiliation:

Nicholas Tierney
Monash University
Melbourne, Victoria, Australia
and
Telethon Kids Institute
Perth, Western Australia, Australia
E-mail: nicholas.tierney@gmail.com

Dianne Cook
Monash University
Melbourne, Victoria, Australia

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
January 2023, Volume 105, Issue 7 *Submitted:* 2020-05-14
doi:10.18637/jss.v105.i07 *Accepted:* 2022-04-01