Expanding tidy data principles to facilitate missing data exploration, visualization and assessment of imputations

Nicholas Tierney, Department of Econometrics and Business Statistics, Monash University,

Corresponding author (nicholas.tierney@gmail.com)

Dianne Cook, Department of Econometrics and Business Statistics, Monash University

September 6, 2018

Abstract:

Despite the large body of research on missing value distributions and imputation, there is comparatively little literature 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 a goal to integrating missing value handling as an integral part of data analysis workflows. New data structures are defined along with new functions (verbs) to perform common operations. Together these provide a cohesive framework for handling, exploring, and imputing missing values. These methods have been made available in the R package `naniar`.

Keywords: workflow, statistical computing, data science, data visualization, tidyverse, data pipeline

1 Introduction

Tidy data (Wickham 2014) is a relatively new principle and suite of tools that facilitate the process of converting raw data into a clean, analysis-ready format that works efficiently in a data analysis pipeline. In tidy data, missing values can be handled implicitly or explicitly, at the analyst's discretion. However, when making a plot, missing values are simply dropped, albeit with a warning (see Figure 1). Most other analysis software will simply drop cases with missings without warning.

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

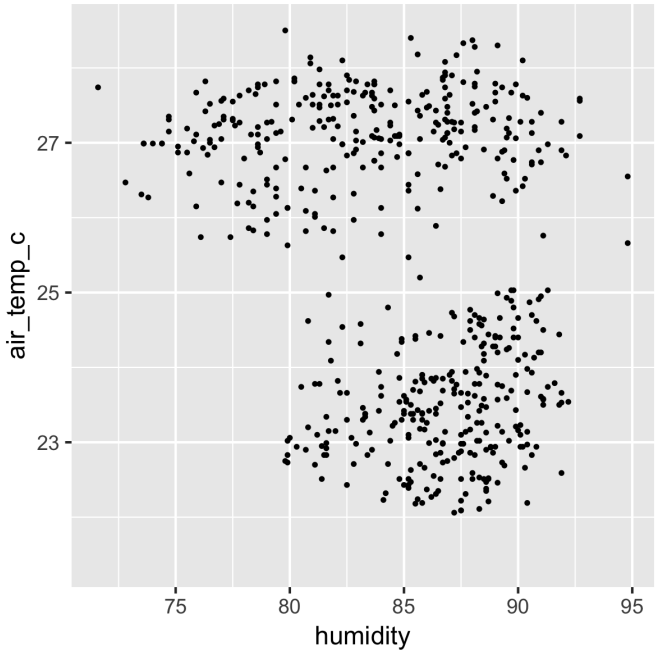

Figure 1: *How ggplot behaves when displaying missing values: Warning message (Left) is thrown prior to plot (Right); ggplot2 does not display missing values in the plot, but at least throws a warning message to the user, which is better than many statistical models.*

The imputation literature does not address how to handle missing data. Producing a complete dataset for analysis, whether by case or variable deletion, or with imputation, requires knowing something about the structure of the missing values. To understand their structure, one needs to generate visualizations, calculate summaries, and perform exploratory modelling. The imputation methods literature (e.g. Little (1988), Rubin (1976), Simon and Simonoff (1986), Schafer and Graham (2002), Buuren (2012)) focuses on ensuring valid statistical inference is made from incomplete data. They approach this chiefly through probabilistic modelling, and assume the mechanism of missing data is known to the analyst.

Exploring missing data to find the missingness mechanism is difficult, because it is an iterative process with many dead ends and often no clear answer. There is no silver bullet for missing data that will reveal it's mechanism to the analyst. Despite it being crucial to explore and understand missing data, there is very little research on how to make this process more efficient so that it integrates into analysis pipelines. Although

decision tree models or latent group analysis can reveal structures or patterns of missingness (Barnett et al. 2017; Tierney et al. 2015), these are not definitive and usually require discussion with data curators to validate. The analyst must explore missing data using visualizations, summaries, and exploratory analyses, but there is a lack of tools to efficiently explore missing data.

The graphics literature (Swayne and Buja 1998; Unwin et al. 1996) provides several solutions for exploring missings visually. These approaches incorporate missings into the plot in some way. For example, imputing values 10% below the range to display all values in a scatter plot. These methods require the data to be linked to the missing or imputed values; this link is typically hidden from the user. The approaches from the graphics literature have not been translated and integrated into tidy data. This limits the analysts capacity to efficiently explore missing values. Extending tidy data and tidy tools to account for missing values would create more efficient and robust workflows and analysis.

The paper is organized in the following way to discuss the problem of missing data. The next section (2.1) is an introduction to tidy data principles and tools. Missing value imputation (2.2 and 2.3) and exploration literature (2.4) are then discussed. Section 3 discusses extensions to the tidy methodology that facilitates operations for exploring, visualizing, and imputing missing data. The application of these new methods is illustrated using a case study in section 4. Finally, section 5 discusses strengths, limitations, and future directions.

2 Background

2.1 Tidy data concepts and methods

Features of tidy data were formally discussed in 2014 (Wickham 2014), subsequently generating much discussion (see Donoho (2017) and commentary), and tools for data analysis. Tidy data is defined as:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

Tidy data is easier to work with and analyze because the variables are in the same format as they would be put into modelling software. This helps the analyst works swiftly and clearly, closing up opportunities for errors.

With tidy data come tidy tools, which have the same input and output: tidy data. This consistency means multiple tools can be composed together in sequence, allowing for rapid, elegant 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 incites errors.

Tidy tools fall into three broad categories: data manipulation, visualization, and modelling. These are now discussed in turn.

2.1.1 Data manipulation

Data manipulation is made input and output tidy with R packages `dplyr` and `tidyr` (Wickham et al. 2017a; Wickham and Henry 2018). 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 adding new ones; **aggregating** creates a single value from many values, say for example by computing the minimum, maximum, and mean.

2.1.2 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 2009). For example, a scatterplot can be created by mapping two variables to the x and y axes, and specifying a point geometry. The graph can then be changed to map a third variable to color of the points, and a fourth variable to size.

2.1.3 Modelling

Modelling 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)`. Modelling tools are input tidy, but their output is always messy. For example, estimated coefficients, predictions, and residuals from one model cannot be easily combined with the output of another model. The `recipes` package is a tidy tool under development to help make modelling input and output tidy (Kuhn and Wickham 2018).

2.1.4 The tidyverse

Defining tidy data and tidy tools has resulted in the growth of a set of packages known collectively as the "tidyverse" (Ross et al. 2017). These are designed to share similar principles in their design and behavior, and cover the breadth of an analysis - going from importing, tidying, transforming, visualizing, modelling, to communicating (Ross et al. 2017; Wickham 2017; Wickham and Grolemund 2016). This has led to a growth of 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` (Grolemund and Wickham 2011; Henry and Wickham 2017; Wickham 2018; Wickham and Bryan 2017; Wickham et al. 2017b; Wickham and Miller 2018). It has also led to the growth of new packages for other fields that follow similar design principles and create fluid workflows for new domains. For example, the `tidytext` package for text analysis, the `tsibble` package for time series data, and `tidycensus` for working with US census and boundary data (Silge and Robinson 2016; Walker 2018; Wang et al. 2018).

2.1.5 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 are primarily *messy*, and suffer the same problems as modelling for imputation: They use tidy input, but produce messy output. The complex, often multivariate nature of imputation methods also makes 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.

Past graphics research on missing data has not yet been framed in a tidy way. The graphics literature has explored methods for missing value exploration and imputation. Translating and expanding these to fit within tidy data and tidy tools would create more efficient workflows.

2.2 Missing data representation and dependence

The information on missing values in a dataset can be represented as a missingness matrix, R, where 0 is observed data and 1 is missing data, where for data y with i rows and j columns, the value r_{ij} is given by:

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

There are many ways each value can go missing. The information in R can be used to arrive at three categories of missing values: Missing completely at random (MCAR), Missing at random, and missing not at random. **MCAR** is where values being missing have no association with observed or unobserved data, that is $Pr(missing|observed, unobserved) = Pr(missing)$. Essentially, the probability of an observation being missing is unrelated to anything else. Although this is a convenient scenario, it is not actually possible to confirm, as it relies on statements on data unobserved. **MAR** refers to cases where the missingness only depends on the data observed, and not data unobserved, that is,$Pr(missing|observed, unobserved) = Pr(missing|observed)$. Some structure or dependence between the missing values and observed values is allowed, provided that this can be explained by the data observed. Finally, **MNAR** refers to the missingness

being related to values unobserved: $Pr(missing|observed, unobserved)$. This assumes that conditioning on all available observed data, the data goes missing due to some phenomena unobserved. This presents a challenge in analysis, as it is difficult to verify this state, and also implies bias in analysis due to the unobserved phenomena.

Visualizations can help assess whether data is MCAR, MAR or MNAR. Imputation is recommended in most cases of missingness dependence, where the values are plausible (e.g., imputing a pulse for someone deceased), as this can help re-assess the classification of missingness.

2.3 Imputation

Imputation methods are more accessible than they have ever been, and the number of methods and implementations in software continues to grow. Values can be imputed with one value (single imputation), or multiple values (multiple imputation), creating m datasets. Methods and software for single and multiple imputation are now discussed.

2.3.1 Single imputation

`VIM` (Kowarik and Templ 2016) efficiently implements well-used imputation methods k-nearest neighbor, regression, hot-deck, and iterative robust model-based imputation. This diversity of approaches allow the package to deal semi-continuous, continuous, count, and categorical data. VIM identifies imputed cases by adding an indicator variable with a suffix `_imp`. So `Var1` would have a sibling indicator column, `Var1_imp`, with TRUE or FALSE values to indicate imputed value. `VIM` also has a variety of visualization methods, discussed in section 2.4. The `simputation` package includes imputation methods for linear regression, decision trees, K nearest neighbors, hotdeck imputation, and the EM algorithm (Dempster et al. 1977; van der Loo 2017). A key feature of its design is that it returns a regular dataframe. This approach contrasts from other imputation software, which may provide custom classes of data or additional objects, which require additional work to prepare for subsequent analysis. Providing a dataframe with imputed values included in the data reduces the friction of working with other tools, but comes at the cost identifying imputed values. Other software for imputation includes `Hmisc` (Harrell Jr et al. 2018), which provide predictive mean matching, `imputeTS` (Moritz and Bartz-Beielstein 2017), which provide time series imputation methods, and `missMDA` (Josse and Husson 2016), which imputes data using principal components analysis.

2.3.2 Multiple imputation

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 (Gelman and Hill 2011; Honaker et al. 2011; van Buuren and Groothuis-Oudshoorn 2011). The `mice` package implements the method of chained equations, using a variable-wise algorithm to calculate the posterior distribution of parameters, from which uses to generate imputed values. The suggested workflow in `mice` revolves around imputing data, returning completed data, and fitting a model and pooling the results. The `Amelia` package (Honaker et al. 2011) assumes data are multivariate normal, and samples from the posterior using the computationally efficient (and parallelizable) Expectation-Maximization Bootstrap (EMB) algorithm (Honaker and King 2010) and allows for incorporation of information on the values in a prior. The `mi` package (Gelman and Hill 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`, which work with the specially multiply imputed data, including 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. Another software for performing multiple imputation is `norm` (R by Alvaro A. Novo. Original by Joseph L. Schafer <jls@stat.psu.edu>. 2013; Schafer 1999), which uses methods from the NORM software from Schafer (1999), such as multiple imputations using EM for multivariate normal data. The `norm` package does not provide a framework for tracking missing values, but instead provides tools for making inference from multiple imputation. Each of these multiple imputation methods provide practical methods, but do not have consistent interfaces across their methods or implementation, which makes then inherently messy, as their output is always different. A consistent data structure that tracks missing values by default would make the data outputs tidy, streamlining subsequent analysis.

2.4 Exploration

The primary focus of most missing data packages is exploring imputed values, not on exploring relationships in missing values, and identifying possible patterns. Texts that do cover exploration of missing data have the same problem as with modelling: the input is tidy, but the output is messy and does not work with other tools (Buuren 2012); this is inefficient. Methods for exploring missing values are primarily covered in literature on interactive graphics (Cook and Swayne 2007; Swayne and Buja 1998; Unwin et al. 1996), and are picked up again in a discussion of a graphical user interface (Cheng et al. 2015).

Missing values in a dataset can be represented as a binary matrix where 0 is present and 1 is missing (Rubin 1976; Schafer and Graham 2002). This is often referred to as a matrix R, which can be used to assess missing data dependence. This idea of a matrix of missing values has also been used in interactive graphics, dubbed a "shadow matrix", to link missing and imputed values to the data, facilitating their display in graphics (Swayne and Buja 1998). This work also focuses heavily on multivariate numeric data. This is an idea upon which this new work builds.

The approach of Unwin et al. (1996) focussed on multivariate categorical data, where missingness is explicitly added as an additional 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 approach of Swayne and Buja (1998) in the software XGobi, further developed in ggobi (Cook and Swayne 2007), focussed more on multivariate quantitative data. Missingness is incorporated into plots in ggobi by setting them to be 10% below the minimum value.

MissingDataGUI provides a user interface for exploring missing data structure both numerically and visually. Using a GUI to explore missing data may make it easier to glean valuable insight into important structures for missingness. However, these insights are typically not captured or recorded, so it is difficult to incorporate them into reproducible analyses. This distracts and breaks analysis workflow, inviting mistakes.

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

`ggplot2` incorporates missingness into visualizations only when mapping a discrete variable to a graph aesthetic. This has some limitations. For example, for data with school grades and test scores, a boxplot visualization could have school grade mapped to the x axis, and test score to the y axis (See Figure 2). If there are missings in a continuous variable like test score, `ggplot2` omits the missing values and prints a warning message. However, if a discrete variable like school year has missing values, an NA category is created for school year, and the scores are placed into this.

Figure 2, shows four scenarios of missingness for test scores across school year - the first contains no missingness, the second contains missingness for school year, the third contains missingness in scores, and the fourth contains missingness in both.

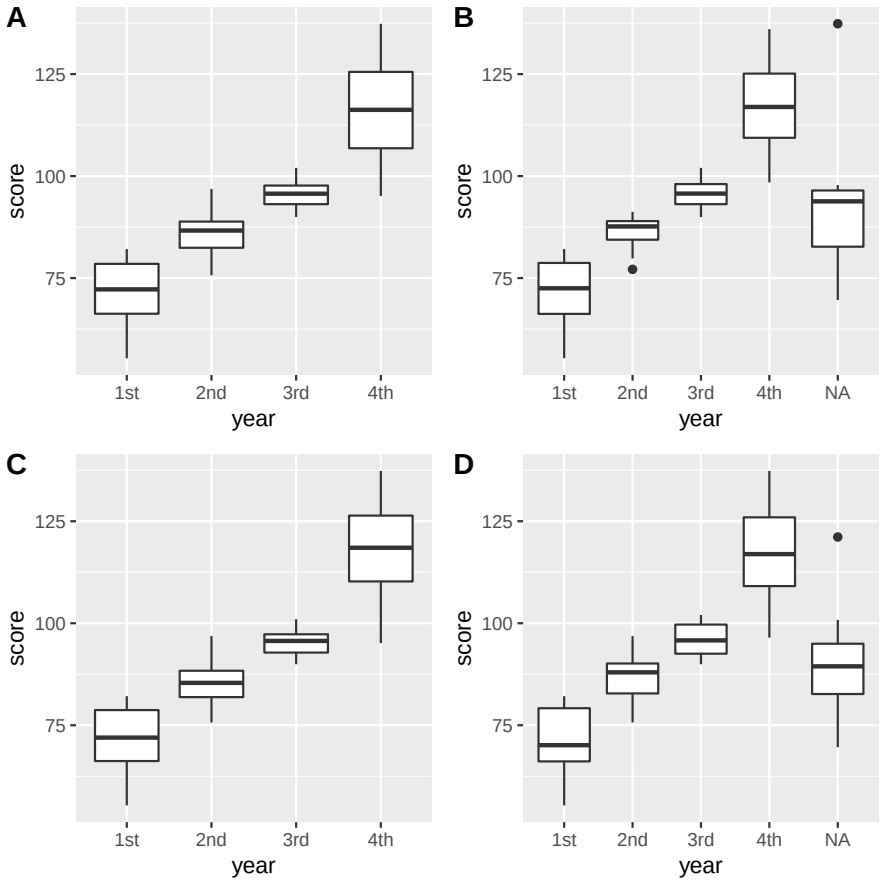

Figure 2: *ggplot2 provides different visualizations depending on what type of data has missing values. (A) Data is complete and graphic is presented; (B) Missings are only in category variable year - an additional 'NA' boxplot is created; (C) Missings only in scores, no additional missingness information is shown; (D) Missings in both scores and school year, additional missing information is shown. The missingness category is only shown when there are missings in categorical variables such as year (plots (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.*

3 Extensions to tidyverse

Applying tidyverse principles to the domain of missing data clarifies and facilitates missing data exploration, visualization, and imputation. This section discusses how these principles are applied, and proceeds in three parts. Section 3.1 discusses *nabular* data, a tidy data structure for missing values that tracks missings throughout the analysis and allows for special missing values. Section 3.2 covers visualizations to understand missingness. Section 3.3 discusses data summaries, and Section 3.4 covers the common "verbs" used to solve problems that arise when working with missing data. The design and naming of these features is an important component that helps the analyst compose them efficiently in data pipelines, and is discussed throughout.

3.1 Data structure

To explore missing data there needs to be a good representation of missing values that integrate into analysis pipelines. A common representation of missing values is a binary matrix the same dimension as the data, where 0 and 1 indicate not missing and missing. This is typically referred to as a matrix R, used to describe missingness dependence patterns, (see for example Rubin (1976) and Schafer and Graham (2002)). This matrix is used to explore missing values in the interactive graphics library XGobi, where it was called a "shadow matrix", defined as a copy of the original data with indicator values of missingness: "NA" and "!NA" for missing and not missing, represented as 0 and 1 (see Figure 3 parts 2 and 3 below).

age	name	year
NA	Phil	1988
10	John	NA
21	NA	1999

(1)

age	name	year
1	0	0
0	0	1
0	1	0

(2)

age	name	year
NA	!NA	!NA
!NA	!NA	NA
!NA	NA	!NA

(3)

age_NA	name_NA	year_NA
NA	!NA	!NA
!NA	!NA	NA
!NA	NA	!NA

(4)

Figure 3: *Progression of creating shadow matrix data. (1-2) Data to a binary representation of missings, (2-3) Binary format converted to a shadow matrix (3-4) New and improved shadow matrix with changed variable names. This clearly links a variable to its state of missingness.*

Three features are added to the shadow matrix to facilitate analysis:

1. Coordinated names: Variables in the shadow matrix gain the same name as in the data, with the suffix "_NA".
2. Special missing values: Values in the shadow matrix can be "special" missing values, indicated as `NA_suffix`, where "suffix" is a very short message of the type of missings.
3. Cohesiveness: Binding the shadow matrix column-wise to the original data creates a cohesive *nabular* data form, useful for visualization and summaries.

These additions are now discussed in turn.

3.1.1 Coordinated names

A consistent short suffix "`_NA`" for each variable keeps names coordinated throughout analysis (See Figure 3 part 4). The suffix is short and easy to remember in analysis or visualization. It also makes a clear distinction that `var_NA` is a random variable of the missingness of a variable, `var`. This subtle change is important as it changes the focus from the value of a variable to it's missingness state, making intent clear when performing analysis.

3.1.2 Special missings

Values in the shadow matrix are "factor" type values, with text values "NA" and "!NA" and underlying number values 1 and 0, for missing and not missing. Extending these to include "special" missing values labelled as "NA_<suffix>" allows for representing additional features such as instrument failure and drop-out as "NA_instr_fail", and "NA_drop_out". The underlying number representation changes from binary to integer. For example, instrument failure and drop out would be represented as 2 and 3, see Table 1 for an example. Encoding these special missing values is achieved by defining logical conditions and suffixes with the `recode_shadow` function in `naniar`, shown in Section 3.4.5.

Table 1: *Example data of temperature, and its observed value, shadow representation, and underlying value. The temperature value of -99 is represented in the shadow column as 'NA_instr', which in turn has the underlying numeric value of 2. This captures additional information about the data that is otherwise difficult to record.*

temp	temp_NA	NA_value
-99	NA_instr	2
NA	NA	1
-1	NA_dropout	3
106	!NA	0

3.1.3 Nabular data

Nabular data binds the shadow matrix column-wise to the original data, and is so named as a portmanteau of `NA` and `tabular`. *Nabular* data keeps corresponding rows together, removing the possibility of mismatching records, and explicitly links missing values to the data. *Nabular* data facilitates visualization and summaries by allowing the user to reference the missingness of a variable, `var`, as `var_NA`. *Nabular* data is a snapshot of the missingness of the data. This means when *nabular* data is imputed, those imputed values can easily be identified in analysis by referring to the appropriate, coordinated names. *Nabular* data is created using the command `bind_shadow()` or `nabular()` (see Figure 4). *Nabular* data is not unlike classical data formats that have quality or flag columns associated with each measured variable, e.g. Scripps CO2 data (Keeling et al. 2005), GHCN data (Durre et al. 2008).

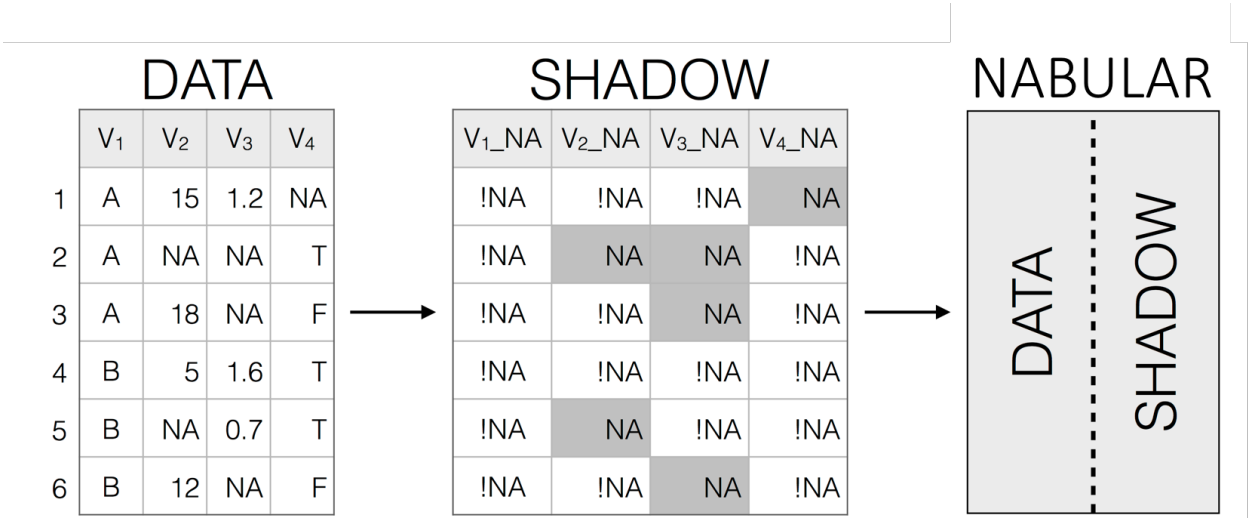

Figure 4: *The process of creating nabular data. Data transformed to shadow matrix, and nabular data contains the shadow matrix, column bound to the data. Nabular data can be created using 'bind_shadow' or 'nabular()' functions. Nabular data provides a useful format for missing data exploration and analysis.*

A data structure for missing data provides key benefits over on-the-fly creating a logical matrix where TRUE means missing, and FALSE means present. Firstly, it can be confusing to users whether TRUE or FALSE means missing or present; `NA` and `!NA` are clearer. Secondly, special missing values cannot be added fluently during analysis with a logical matrix. Finally, the logical matrix cannot capture which values are imputed, if imputation has already taken place. Imputing values on *nabular* data automatically tracks these value changes.

Using additional columns to represent missingness information follows recently published best practices for data organization, described in Ellis and Leek (2017) and Broman and Woo (2017): (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 is now represented in the shadow columns as additional integers, along with the other missing information.

Other statistical programs represent missingness values as a full-stop, `.`, and allow for recording special missing values as `.a` through to `.z`. The special values from these languages break the rule of "keeping one thing in a cell", as they record both the value and the multivariate missingness state.

3.2 Visual summaries

Visualizing missingness in a dataset is essential to understand its structure. This section discusses visualizations that are easy to remember and compose with other functions in a pipeline. Four different areas of visualization

of missing data are discussed: overviews (Section 3.2.1); univariate (Section 3.2.2), bivariate (Section 3.2.3), and multivariate (Section 3.2.4). All plots are created using `ggplot2`, giving users clearer control over the plot appearance, and customizability.

3.2.1 Overview plots

The number of missings in each variable and case are visualized using `gg_miss_var` and `gg_miss_case` (see Figure 5 below). These are shown using the "airquality" dataset, included in base R, which contains daily air quality measurements in New York, from May to September, 1973. Rather than highlighting the amount of complete data, these visualizations draw attention to the amount of missings, ordering by missingness.

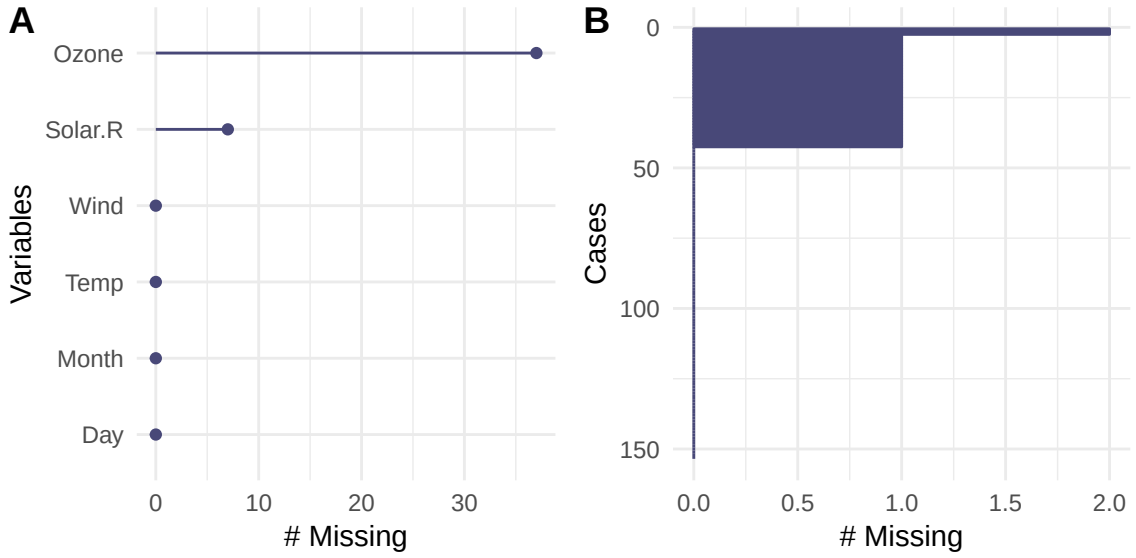

Figure 5: *Graphical summaries of missingness in variables and cases for the airquality data. (A) Missings in each variable and (B) in each case. There are missing values in Ozone and Solar.R, with Ozone having more missings, and not many cases have two missing values, with most missingness being from cases with one missing value.*

All missings in a dataset can be displayed using a heatmap style visualization. This is achieved using `vis_miss()` from the `visdat` package (Tierney 2017), which also provides summaries of missingness overall in the legend, and for each column (see Figure 6). The user can also apply clustering to the rows, and arrange columns by missingness (Figure 6B). Similar visualizations are available in other missing data packages such as `VIM`, `mi`, `Amelia`, and `MissingDataGUI`. A key improvement is that `vis_miss()` orients the visualization analogous to a regular data structure: variables form columns in the visualization and are named at the top, and each row is an observation. Using `ggplot2` means the plot can be customized and combined with other ggplot graphics.

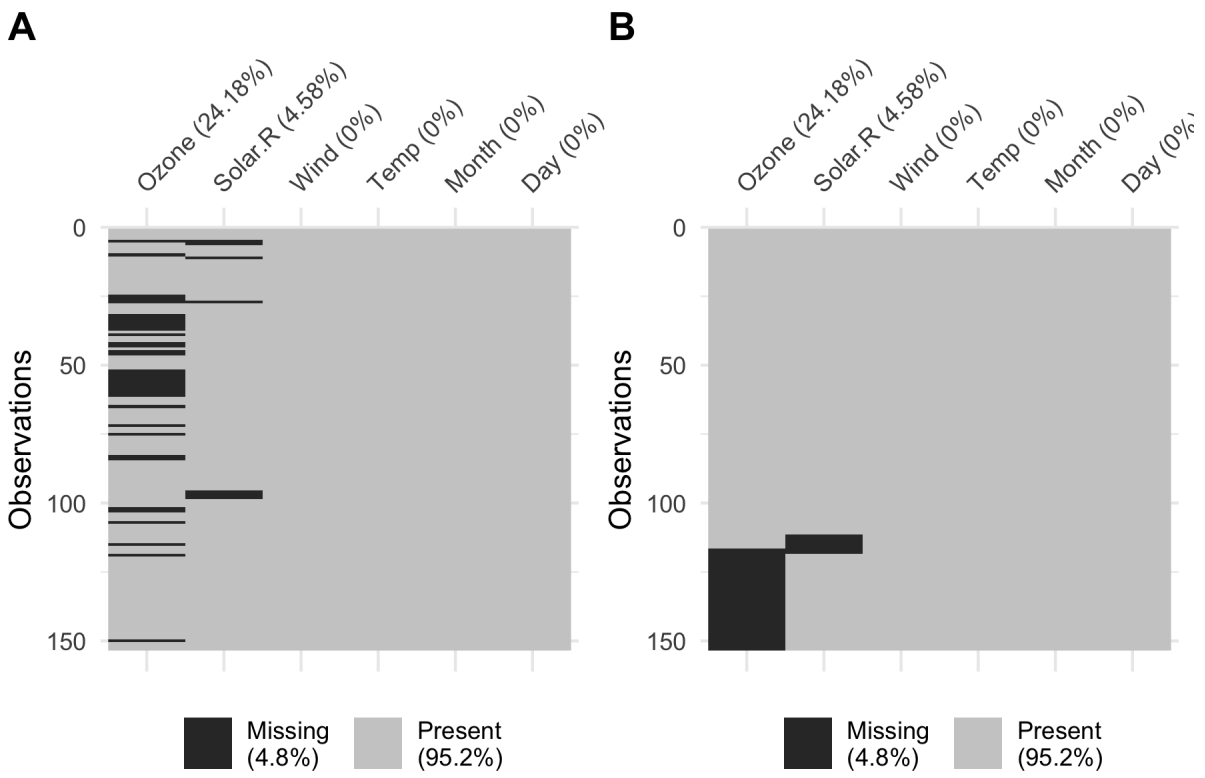

Figure 6: *Heatmap visualizations of missing data for the airquality dataset. (A) The default output and (B) ordered by clustering on rows and columns. There are only missings in ozone and solar radiation, and there appears to be some structure to their missingness.*

The number of times certain variables go missing together can be visualized using an "upset plot" (Conway et al. 2017), a visualization technique for showing the size and features of sets in data that is similar to a venn diagram, but scales better with more variables (Figure 7).

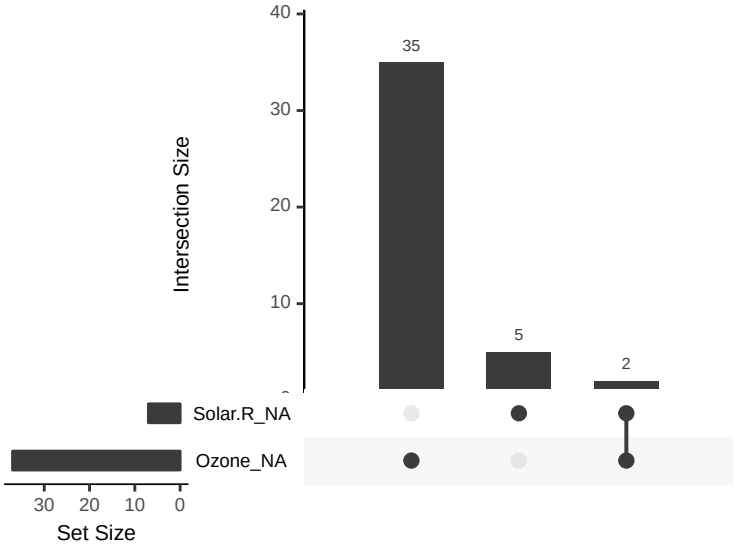

Figure 7: *The pattern of missingness in the airquality dataset shown in an upset plot. Only Ozone and Solar.R have missing values, and Ozone has the most missing values. There are 2 cases where both Solar.R and Ozone have missing values.*

Figure 7 shows that only Ozone and Solar.R have missing values. The dots in the bottom right show the combinations of missingness, with the bottom left showing the missingness in each variable, and the top showing the missingness for the combinations. The bottom left shows that Ozone has the most missing values, and the top shows 2 cases where both Solar.R and Ozone have missing values together.

3.2.2 Univariate plots

Missing values are by default 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.

3.2.2.1 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 values below the range of data using `impute_below` (by default imputing 10% below the range). Visualizing this as a histogram shows missing values on its left. Ozone is shown in a different color by referring to the variable `Ozone` as `Ozone_NA` in a fill aesthetic in ggplot, shown in Figure 8.

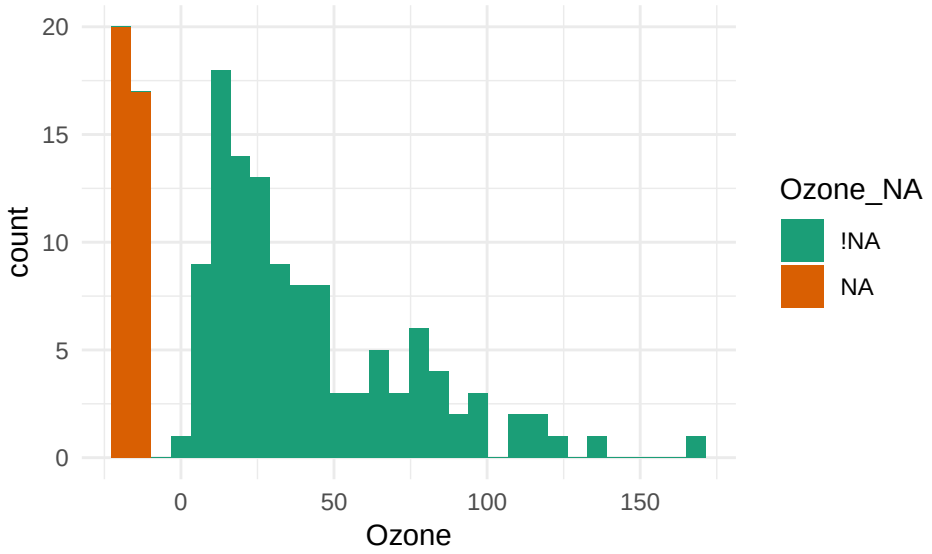

Figure 8: *A histogram using nabular data to show the values and missings in ozone. Values are imputed below the range to show the number of missings in Ozone and colored according to missingness of ozone ('Ozone_NA'). There are about 35 missings in Ozone.*

3.2.2.2 Univariate split by missingness

Missingness of one variable can also be used to display different distributions in another. Figure 9 shows the values of temperature when ozone is present, and missing. Figure 9A is a faceted histogram, and Figure 9B is an overlaid density. 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.

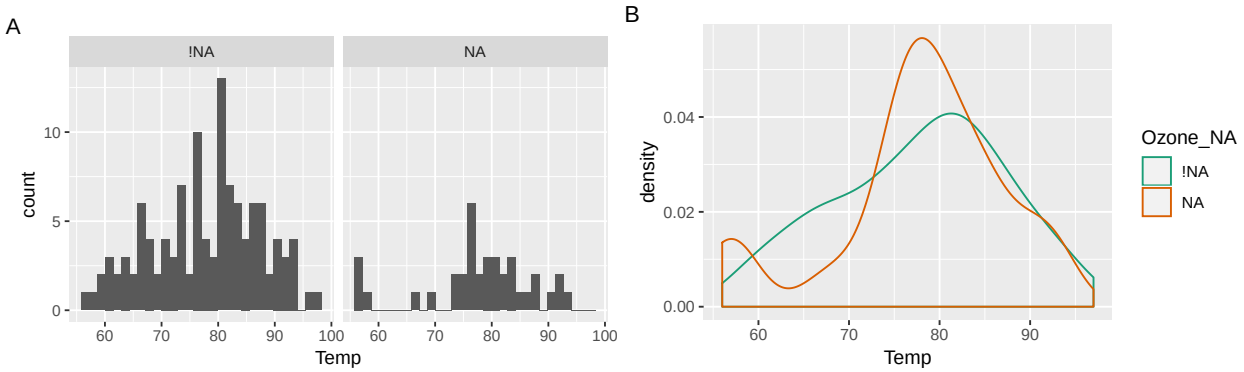

Figure 9: *A visualization of temperature according to missingness in ozone from in the airquality dataset. A histogram of temperature facetted by the missingness of ozone (A), or a density of temperature colored by missingness in ozone (B). The distribution shows a cluster of low temperature observations with missing ozone values, but the temperature values are otherwise similar.*

3.2.3 Bivariate plots

To visualize missing values in two dimensions the missing values can be placed in the plot margins. This is achieved by imputing values below the range of the data. As in Section 3.2.2.1, using *nabular* data identifies imputed values, and color makes missingness pre-attentive (Treisman 1985). The steps of imputing and coloring have been been combined into `geom_miss_point()` from the `naniar` package (Figure 10), where there is a mostly uniform spread of missing values for Solar.R and Ozone.

Figure 10: *Scatterplots with missings displayed at 10 percent 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.*

As `geom_miss_point` is a defined geometry for `ggplot2`, it provides features such as faceting and mapping other variables to aesthetics such as color, shape, or size. The shadow format by itself can also be used to visually explore some patterns of missingness.

3.2.4 Multivariate plots

Parallel coordinate plots can visualize missingness beyond two dimensions. They transform variables to the same scale, and typically make the data values range between 0 and 1. To showcase this visualization, the dataset `oceanbuoys` from `naniar` is used, which contains measurements of moored ocean buoys to understand and predict El Niño and El Niña. The data were collected in 1993 and 1997, and contains information on the sea and air temperature, humidity, and east west and north south wind directions.

Figure 11 shows a parallel coordinate plot of the `oceanbuoys` data, with missing values imputed to be 10% below the range, and values colored according to whether humidity contained missing values. Although a 10% below imputation is not the most ideal way to display missing values in a parallel coordinate plot, it

shows that humidity is missing at low air and sea temperatures, and that humidity is missing in one year, and one location.

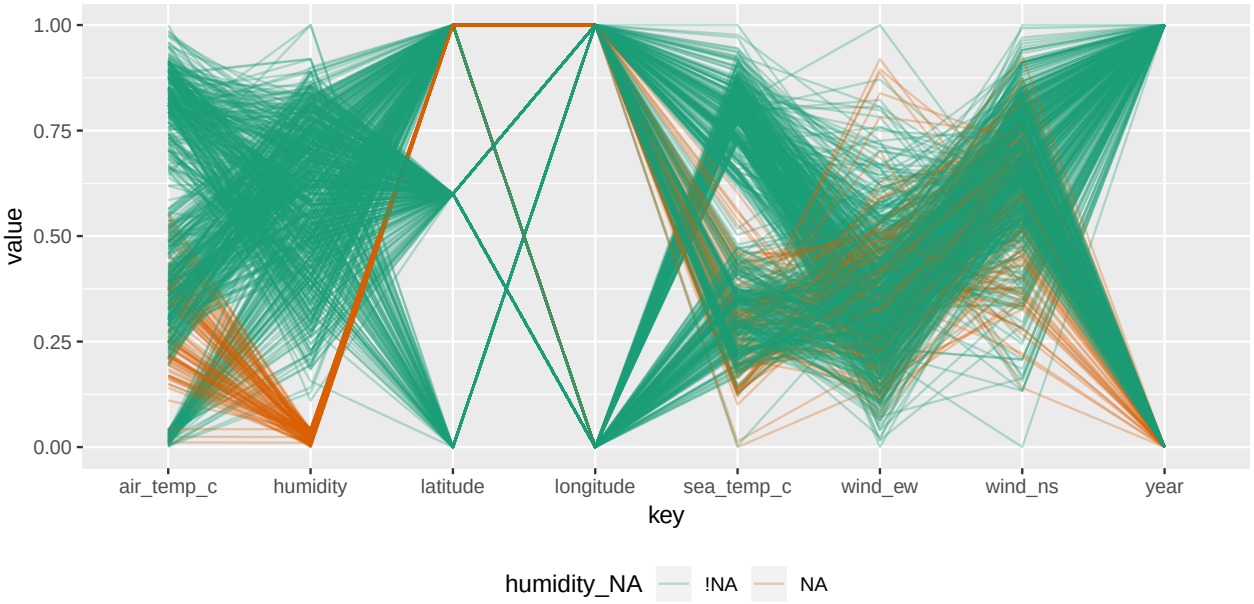

Figure 11: *Parallel coordinate plot shows missing values imputed 10% below range for the oceanbuoys dataset. Values are colored by missingness of humidity. Humidity is missing for low air and sea temperatures, and is missing for one year and one location.*

3.3 Numerical summaries

There are different ways to summarize the amount of missing and complete values in a dataset. This section describes approaches to summarizing missingness, implemented in the `naniar` package. The functions are easy to remember as they follow consistent naming, and their output is consistent, and returns a single number or dataframe. This means they integrate well with plotting and modelling tools. Section 3.3.1 discusses single number summaries for all data and for cases and variables ; Section 3.3.2 discusses variable- and case-wise summaries; and Section 3.3.3 shows how the design of these tools works with other tools in an analysis pipeline.

3.3.1 Single number summaries

The overall number of missing values in a dataset is shown with `n_miss`, and the proportion or percent missings, `prop_miss` and `pct_miss`. Complementing these are functions to reveal the number of complete values: `n_complete`, `prop_complete`, and `pct_complete`. These be extended to summarize the number/amount of variables and cases that contain a missing values by appending `_case` or `_var` to any of these summaries. Table 2 shows the names and output of these functions on the dataset airquality.

Table 2: *Single number summaries of missing and complete data, applied to the airquality dataset. These functions follow consistent naming, to make them easy to remember, and to make their use clear.*

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_var	27.45	prop_complete_var	0.73
pct_miss_case	33.33	pct_complete_case	66.67

3.3.2 Summaries and tabulations of missing data

Presenting the number and percent of missing values for each variable, or case, provides a summary that can be used by models to inform imputations or data handling. 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, one, two, and so on, missing values. These summaries and tabulations are shown for variable in Tables 3 and 4, using the `miss_var_summary` and `miss_var_table` functions. Functions are also provided for case-wise summaries and tabulations with `miss_case_summary` and `miss_case_table`, which order by `n_miss`, so the most missings are always shown at the top. Other summaries provided in `naniar` include summaries of missingness across a provided repeating span, `miss_var_span`, and finding the streaks or runs of missingness in a given variable in `miss_var_run`.

Table 3: *Summary of the number and percent of missings in each variable. Only ozone and solar radiation have missing values.*

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: *Tabulation of the amount of missing data in each variable. This shows the number of variables with no missings, 7 missings, and 37 missings, and the percentage of variables with those amounts of missingness. There are not many patterns of missingness.*

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

3.3.3 Combining numerical summaries with grouping operations

These summaries and tabulations can be composed with the "pipe" operator to perform grouped summaries using the `group_by` operator from `dplyr`. These summaries return dataframes, which may be used by other plotting devices, modelling tools, or other summaries. Table 5 shows an example of the missing data summaries for the airquality dataset in each variable, for each month of the year. These datasets can be summarized in visualizations, or values input into a model.

Table 5: *A summary of the missingness in each variable, for each Month of the airquality dataset, with only the first 10 rows of output shown. There are more ozone missings in June than May.*

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

3.4 Verbs

Common operations on missing data can be considered to be verbs: scan, replace, add, shadow, impute, track, and flag. Missing values can be **scanned** to find possible missings not coded as `NA`. These values can then be **replaced** with an `NA` value. To facilitate exploration, summaries of missingness can be **added** as a summary column to the original data. Data values can be augmented with **shadow** values, which helps explore missing data, as well as facilitating the process of **imputing**, and **tracking**. Finally, unusual or specially coded missing values can be **flagged**. These verbs are described in this section.

3.4.1 scan: Searching for common missing value labels

There are many common values that mean "missing". For example, "N/A" "Not Available", "MISSING", or white space `" "`. Before replacing these values with missing values (`NA`), it is useful to first *scan* the data to get a sense of the magnitude of the problem. The `miss_scan_count()` function finds where specified values occur in data. For example, for the example data, `dat_ms`:

All occurrences of -99 can be searched for using the code:

Table 6: *An example dataset with variables x, y and z containing some common unusual missing values.*

x	y	z
1	A	-100
3	N/A	-99
NA	NA	-98
-99	E	-101
-98	F	-1

```
miss_scan_count(data = dat_ms, search = -99)
```

This returns a table of the occurrences of that value in each variable (Table 7). A list of common NA values is provided in `naniar` in the `common_na_strings` and `common_na_numbers` for characters and numbers. These contain values like "n/a", "na", and ".", and -9, -99.

Table 7: *Table of the occurences of the search '-99' in the data, 'dat_ms'. There is one occurence if -99 in variables x and z.*

Variable	n
x	1
y	0
z	1

3.4.2 replace: Replacing missing value label with another

Once the magnitude of the unusual missing values has been assessed, these values need to be replaced with appropriate missing values, using `replace_with_na`. This is useful when there is certainty of which values mean missing. For example "N/A", "N A", "Not Available", and "missing". Using the previous toy dataset, `dat_ms`, the value -99, or both -99 and 98 can be replaced with NA as follows:

```
replace_with_na(dat_ms, replace = list(x = -99))
```

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

```
replace_with_na(dat_ms, replace = list(x = c(-99, -98)))
```

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

For flexibility, there are scoped variants for `replace_with_na`: `_all`, `_if`, and `_at`. This means `replace_with_na` works for **all** variables with `replace_with_na_all`, **at** selected variables with `replace_with_na_at`, and **if** variables that meet some condition with `replace_with_na_if`. The syntax for these is slightly different to `replace_with_na`, but powerful:

```
replace_with_na_at(data = dat_ms,
                   .vars = "x",
                   condition = ~.x == -99)
```

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

```
replace_with_na_if(data = dat_ms,
                   .predicate = is.character,
                   condition = ~.x == "N/A")
```

```
## # A tibble: 5 x 3
##       x y         z
##   <dbl> <chr> <dbl>
## 1     1 A      -100
## 2     3 <NA>    -99
## 3    NA <NA>    -98
```

```
## 4   -99 E      -101
## 5   -98 F        -1
```

```
replace_with_na_all(data = dat_ms,
                    condition = ~.x == -99)
```

```
## # A tibble: 5 x 3
##       x y         z
##   <dbl> <chr> <dbl>
## 1     1 A      -100
## 2     3 N/A      NA
## 3    NA <NA>    -98
## 4    NA E      -101
## 5   -98 F        -1
```

3.4.3 add: Adding missingness summary variables

Summary information such as the number of missings in a given case can be useful in understanding missingness structure, and even more so when this information is kept alongside the data. Functions starting with `add_` exist in `dplyr` to add count information for particular groups or conditions to the data. Similarly, `naniar` includes `add_` functions to add missingness summary information back to the data, such as the number or proportion of missingness, the missingness cluster, or if there are any missings, using: `add_n_miss()` `add_prop_miss()`, `add_miss_cluster`, and `add_any_miss()`, respectively.

An example use of these features is in Tierney et al. (2015), where the proportion, number, or cluster of missings is used as the outcome in a model. The variables in the dataset can then be used to predict the outcome, identifying variables and values important in predicting missingness structures. There are also functions for adding information about shadow values or nice labels for if there are any missing values with `add_label_shadow()` and `add_label_missings()`:

```
add_n_miss(dat_ms)
```

```
## # A tibble: 5 x 4
##       x y         z n_miss_all
##   <dbl> <chr> <dbl>      <int>
## 1     1 A      -100          0
## 2     3 N/A     -99          0
## 3    NA <NA>    -98          2
## 4   -99 E      -101          0
```

```
## 5   -98 F        -1          0
```

```
add_prop_miss(dat_ms)
```

```
## # A tibble: 5 x 4
##       x y         z prop_miss_all
##   <dbl> <chr> <dbl>         <dbl>
## 1     1 A      -100         0
## 2     3 N/A     -99         0
## 3    NA <NA>    -98         0.667
## 4   -99 E      -101         0
## 5   -98 F        -1         0
```

```
add_miss_cluster(dat_ms)
```

```
## # A tibble: 5 x 4
##       x y         z miss_cluster
##   <dbl> <chr> <dbl>        <int>
## 1     1 A      -100            1
## 2     3 N/A     -99            1
## 3    NA <NA>    -98            2
## 4   -99 E      -101            1
## 5   -98 F        -1            1
```

3.4.4 shadow: Creating `nabular` data

Nabular data has the shadow matrix column-bound to the existing data. This facilitates visualization and summaries, and allows for imputed values to be appropriately tracked. *Nabular* data can be created with `bind_shadow()` or `nabular()`. The `bind_shadow()` function is so named to borrow from existing functions named `bind` - such as `bind_rows()` and `bind_cols()` from `dplyr`, which bind rows or columns into a dataframe. An example use of `bind_shadow()` on the example data, `dat_ms` is shown below:

```
bind_shadow(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
```

3.4.5 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. These special types of missingness can be encoded in the shadow matrix of *nabular* data, using the `recode_shadow()` function. This provides a fluent interface to recode, or flag, the shadow information in the shadow matrix as a special type of missing value. Using `recode_shadow()` requires specifying the variable you want to contain the flagged value, the condition you want this to occur at, and a suffix for the new type of missing value. This is then recoded as a new factor level in the shadow matrix, so that every column is aware of all possible new values of missingness. For example, the values -99 and -98 could be recoded to mean a broken machine sensor for the variable x:

```
dat_ms %>%
  bind_shadow() %>%
  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
```

3.4.6 impute: Imputing values

`naniar` does not reinvent the wheel for imputation, as there are many robust R packages for imputation, such as `mice`, `mi`, `VIM`, and `simputation`. However, `naniar` does provide a few imputation methods to facilitate exploration and visualizations, which were not otherwise available: `impute_below` and `impute_mean` and `impute_median`. These imputation methods are useful to explore structure in missingness, but are not recommended for use in analysis.

The `impute_below` function imputes values below the minimum of the data, with some jitter to reduce overplotting, but has options to change the default amount it is shifted below, and the amount of jitter.

```
##       Ozone Solar.R
## 1  41.00000     190
## 2  36.00000     118
## 3  12.00000     149
## 4  18.00000     313
## 5 -19.72321      NA
## 6  28.00000      NA
```

Similar to `simputation`, each `impute_` function returns the data with values imputed. However, `naniar` does not use a formula syntax, and instead each function implements "scoped variants" `_all`, `_at` and `_if`. Here, `_all` operates on **all** columns, `_at` operates **at** specific columns, and `_if` operates on columns **if** they meet some condition (such as `is.numeric` or `is.character`). The `impute_` functions with no scoped variant, e.g., `impute_mean`, will work on a single vector, but not a data.frame. An example usage is shown below:

```
impute_mean(airquality$Ozone) %>% head()
```

```
## [1] 41.00000 36.00000 12.00000 18.00000 42.12931 28.00000
```

```
impute_mean_at(airquality, .vars = vars(Ozone)) %>% head()
```

```
##      Ozone Solar.R Wind Temp Month Day
## 1 41.00000     190  7.4   67     5   1
## 2 36.00000     118  8.0   72     5   2
## 3 12.00000     149 12.6   74     5   3
## 4 18.00000     313 11.5   62     5   4
## 5 42.12931      NA 14.3   56     5   5
## 6 28.00000      NA 14.9   66     5   6
```

```
impute_mean_if(airquality, .predicate = is.integer) %>% head()
```

```
##      Ozone  Solar.R Wind Temp Month Day
## 1 41.00000 190.0000  7.4   67     5   1
## 2 36.00000 118.0000  8.0   72     5   2
## 3 12.00000 149.0000 12.6   74     5   3
## 4 18.00000 313.0000 11.5   62     5   4
## 5 42.12931 185.9315 14.3   56     5   5
## 6 28.00000 185.9315 14.9   66     5   6
```

```
impute_mean_all(airquality) %>% head()
```

```
##      Ozone  Solar.R Wind Temp Month Day
```

```
## 1 41.00000 190.0000  7.4   67     5   1
## 2 36.00000 118.0000  8.0   72     5   2
## 3 12.00000 149.0000 12.6   74     5   3
## 4 18.00000 313.0000 11.5   62     5   4
## 5 42.12931 185.9315 14.3   56     5   5
## 6 28.00000 185.9315 14.9   66     5   6
```

One drawback to imputation functions in this form is that the location of imputed values is not tracked. This is covered in Section 3.4.7.

3.4.7 track: Shadow and impute missing values

To evaluate imputations they need to be tracked, which is achieved by combining the verbs `bind_shadow`, `impute_`, and `add_label_shadow`. The missing values can then be referred to by their shadow variable, `_NA`. The missingness of any observation can be referred to with `any_missing` (Figure 12). The code chunk below shows the track pattern, first using `bind_shadow`, then imputing with `impute_knn`, and adding a `label_shadow`:

```
aq_imputed <- airquality %>%
  bind_shadow() %>%
  as.data.frame() %>%
  simputation::impute_knn(Ozone ~ .) %>%
  simputation::impute_knn(Solar.R ~ .) %>%
  add_label_shadow() %>%
  as_tibble()
```

Missing values can then be shown in a scatterplot by setting the `color` aesthetic in ggplot to `any_missing` (Figure 12A), or in a density plot looking at one variable, using the `fill = any_missing`, (Figures 12B) and 12C).

```
ggplot(aq_imputed,
       aes(x = Ozone,
           y = Solar.R,
           color = any_missing)) +
  geom_point() +
  scale_color_brewer(palette = "Dark2") +
  theme(legend.position = "bottom")


ggplot(aq_imputed,
```

```
       aes(x = Ozone,
           fill = any_missing)) +
  geom_density(alpha = 0.3) +
  scale_fill_brewer(palette = "Dark2") +
  theme(legend.position = "none")

ggplot(aq_imputed,
       aes(x = Solar.R,
           fill = any_missing)) +
  geom_density(alpha = 0.3) +
  scale_fill_brewer(palette = "Dark2")
```

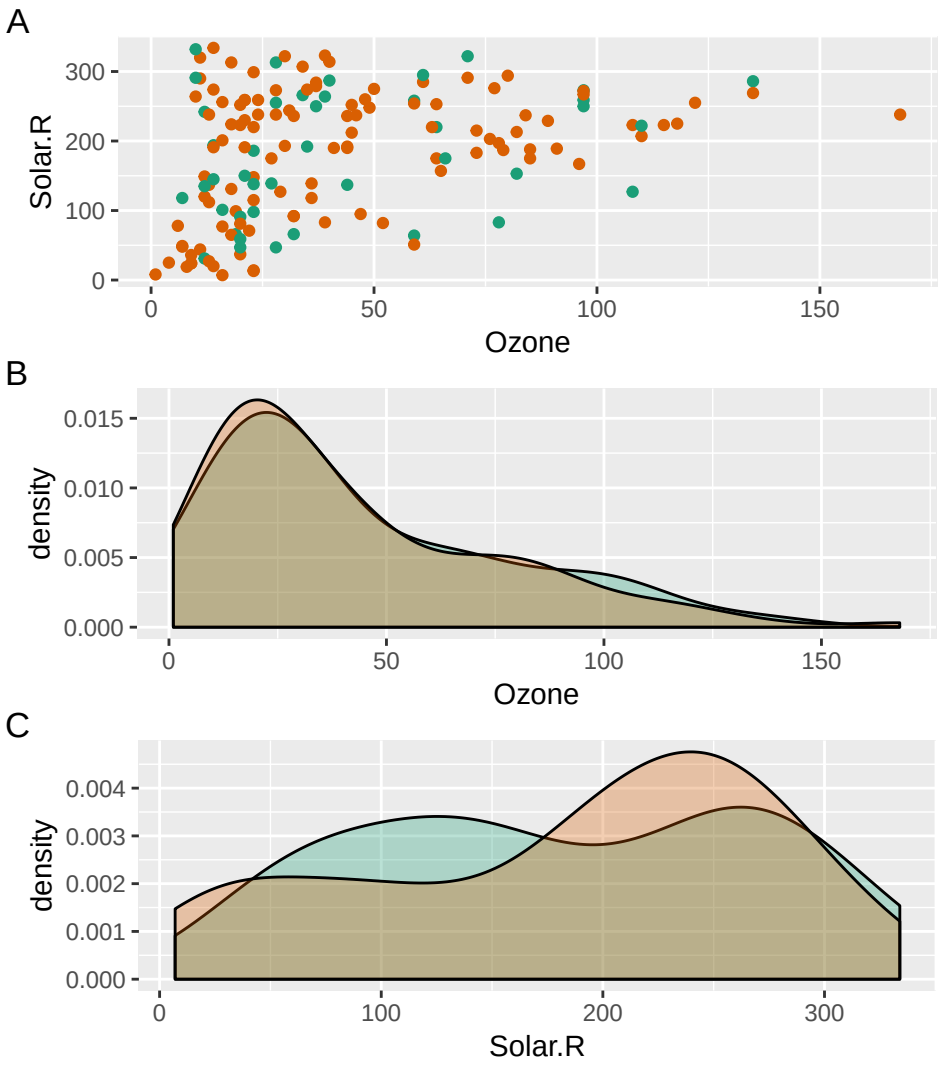

Figure 12: *Scatterplot (A) and density plots (B and C) of ozone and solar radiation from the airquality dataset containing imputed values imputed using simputations 'impute_knn' function, with imputed values colored green and data values orange. Imputed values are similar, but slightly different to existing data.*

Imputed values can also be compared to complete case data by grouping by `any_missing`, and then summarizing (Table 8).

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

Table 8: *Summary statistics of the previously imputed and complete cases. The mean and median values are similar, but the minimum and maximum values are different. The comparison of imputed values is similar to other dplyr summary workflows.*

any_missing	min	mean	median	max
Missing	7	43.6	30	135
Not Missing	1	42.1	31	168

4 Application

To illustrate the methods, data on housing for the city of Melbourne from January 28, 2016 to March 17, 2018 is used. The data was compiled by scraping weekly property clearance data (Pino 2018). There are 27,247 properties, and 21 variables in the dataset. The variables include the type of real estate (town house, unit, house), suburb, method of selling, number of rooms, price, real estate agent, date of sale, and the distance from the Central Business District (CBD).

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 focusses on understanding the patterns of missingness. This section shows how the methods from previous sections are used together in a data analysis workflow.

4.1 Exploring patterns of missingness

Figure 13A 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 13B reveals there are up to 50% missing values in cases, and that 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.

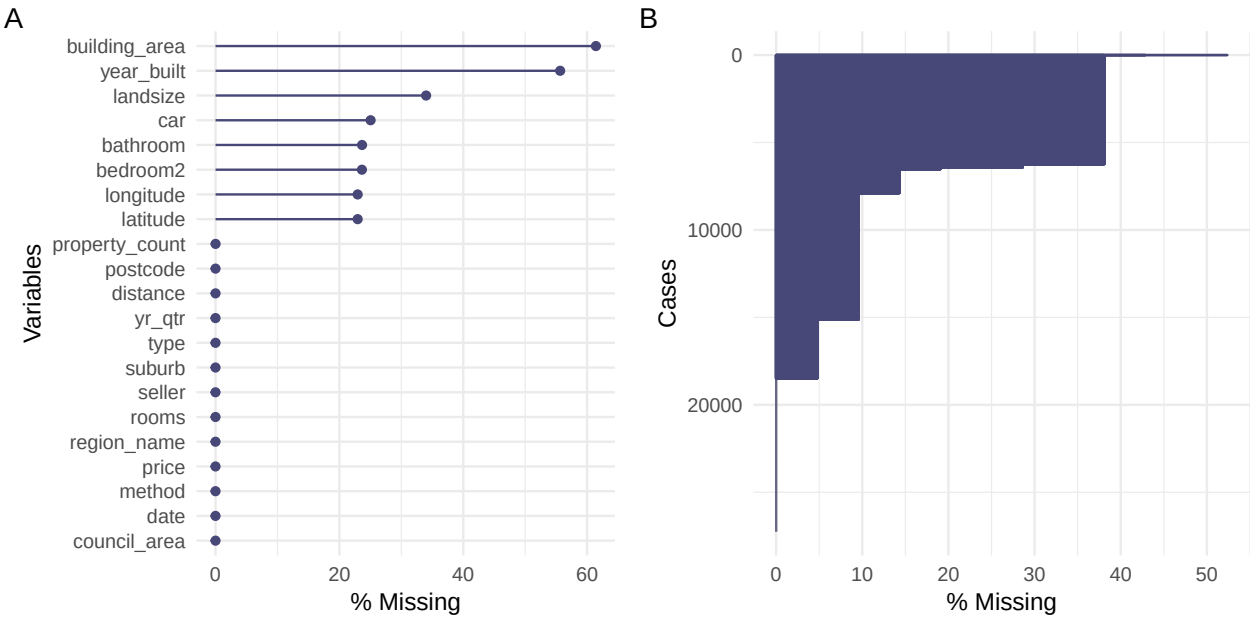

Figure 13: *The amount of missings in each variable (A) and in each case (B) for Melbourne housing data. (A) Build area and year built have more than 50% missing values, and car, bathroom, bedroom2 and longitude and latitude have about 25% missings. (B) There are between 5 and 50% missing values in cases. There are many missing values in the data, with the majority of missingness being in selected cases and variables.*

Possible missingness structures are revealed by visualizing missingness in the whole dataset, clustering and arranging the rows and columns of the data (Figure 14). There are three main clusters of missing data: at the top there are many variables missing together; in the middle, building price and year built missing together, at the bottom, building area, year built, and land size are missing.

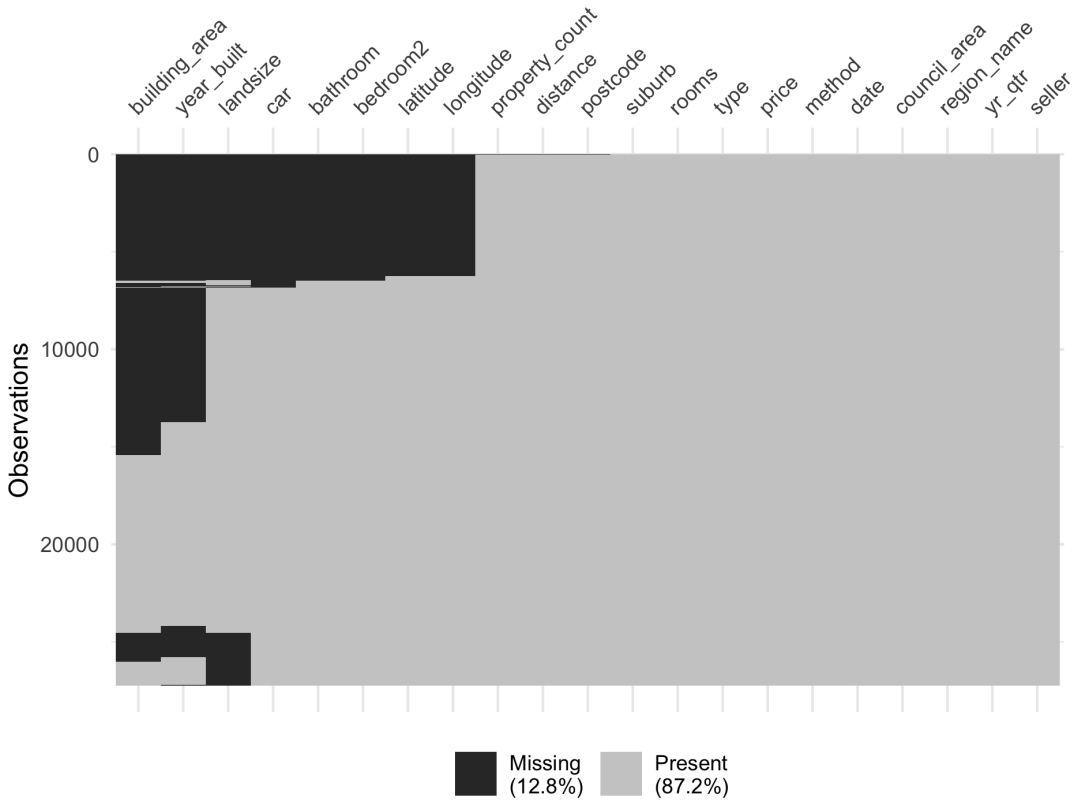

Figure 14: *Heatmap of clustered missingness for the housing data. Three groups of missingness are apparent, at the top for building area to longitude, the middle for building area and year built, at the end for building area, year built, and landsize. There is some structure in the missings.*

Missingness patterns can also be shown with an `upset` plot (Conway et al. 2017), to display the 8 of intersecting sets of missing variables (see Figure 14. Two patterns stand out: two, and five variables missing. This provides further evidence of the patterns of missingness seen in Figures 13 and 15.

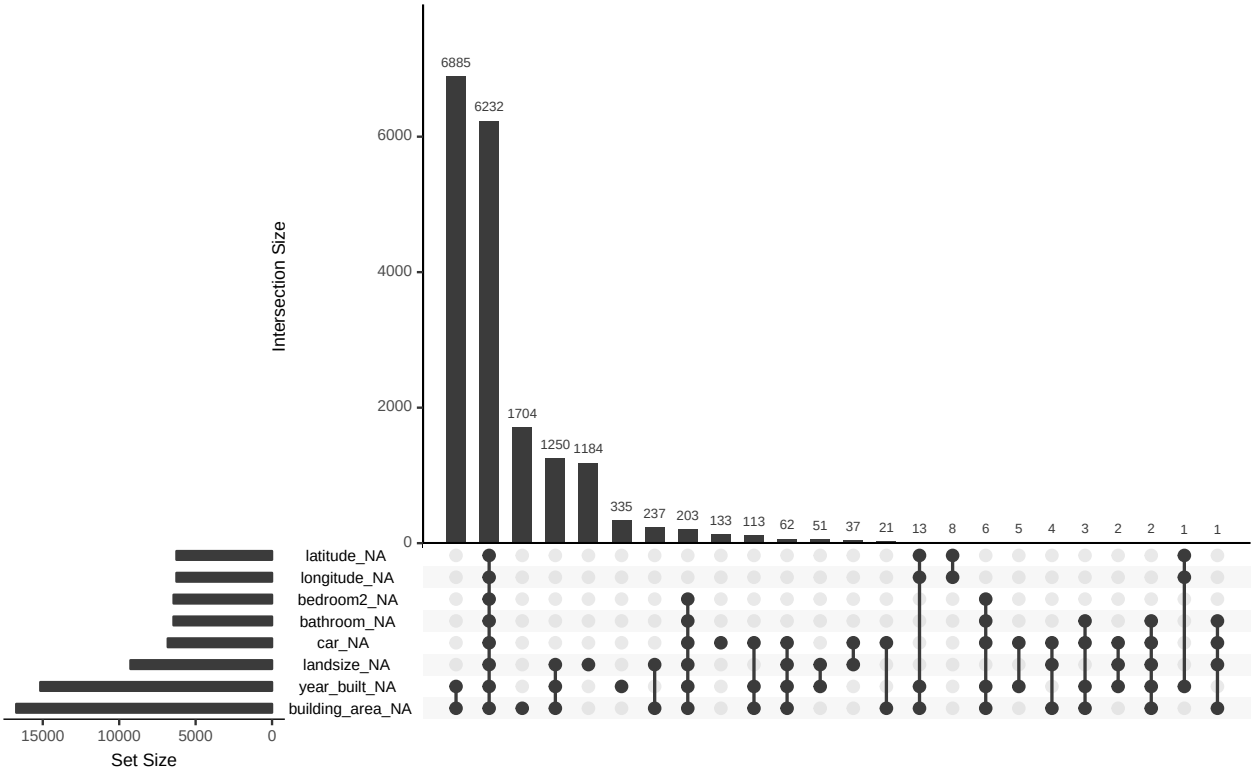

Figure 15: *An upset plot of eight sets of missingness in the housing data. Two missingness patterns are clear, year built and building area, and lattitude through to building area.*

Tabulating the number of missings in variables (see Table 9 (left)) shows three groupings: 6254 - 6824 missings cases in variables, 9265 in another variable, and between 15163 and 16736 missing. Tabulating missings in cases (Table 9 (right)) shows 32% of cases have complete data, 26% have 2 missing, 22% have 8 missing, 12% have 1 missing, and 5% have 3 missing, the remaining cases with missings are less than 1%. Two variables with more than 50% missingness are omitted from analysis: building area and year built.

4.2 Exploring missingness patterns for imputation

Using the information from the overview visualizations and tables, the following variables are explored for features predicting missingness: Land size, latitude, longitude, bedroom2, bathroom, car, and land size.

Missingness structure can be better understood and evaluated by using the data to predict where the data goes missing. This can be achieved by clustering the missing values into groups, and then applying classification and regression trees (CART) to find those variables and values that predict missingness clusters (Barnett et al. 2017; Tierney et al. 2015). Two clusters are identified, one for each missingness cluster. The missingness clusters are predicted using all variables in the dataset with the CART package `rpart` (Therneau and Atkinson 2018), and plotted using the `rpart.plot` package (Milborrow 2018). Importance scores from the CART model describe those variables important for predicting missingness: rooms, price, suburb, council

Table 9: *Tabulating missingness for variables (left) and cases (right) to understand missingness patterns. 14 variables have between 0 and 3 missings, then 6 variables have 6000 - 9000 missings, and 2 variables have 15 - 16,000 missing values. About 30% of cases have no missing values, then 45% of cases have between 1 and 6 missing values, and then 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.*

n_miss_in_var	n_vars	pct_vars	n_miss_in_case	n_cases	pct_cases
0	10	47.6	0	8755	32.1
1	2	9.5	1	3356	12.3
3	1	4.8	2	7244	26.6
6254	2	9.5	3	1370	5.0
6441	1	4.8	4	79	0.3
6447	1	4.8	5	8	0.0
6824	1	4.8	6	203	0.7
9265	1	4.8	8	6229	22.9
15163	1	4.8	9	2	0.0
16736	1	4.8	11	1	0.0

area, distance, and region name. These variables are important for predicting missingness, so are important to include in the imputation model.

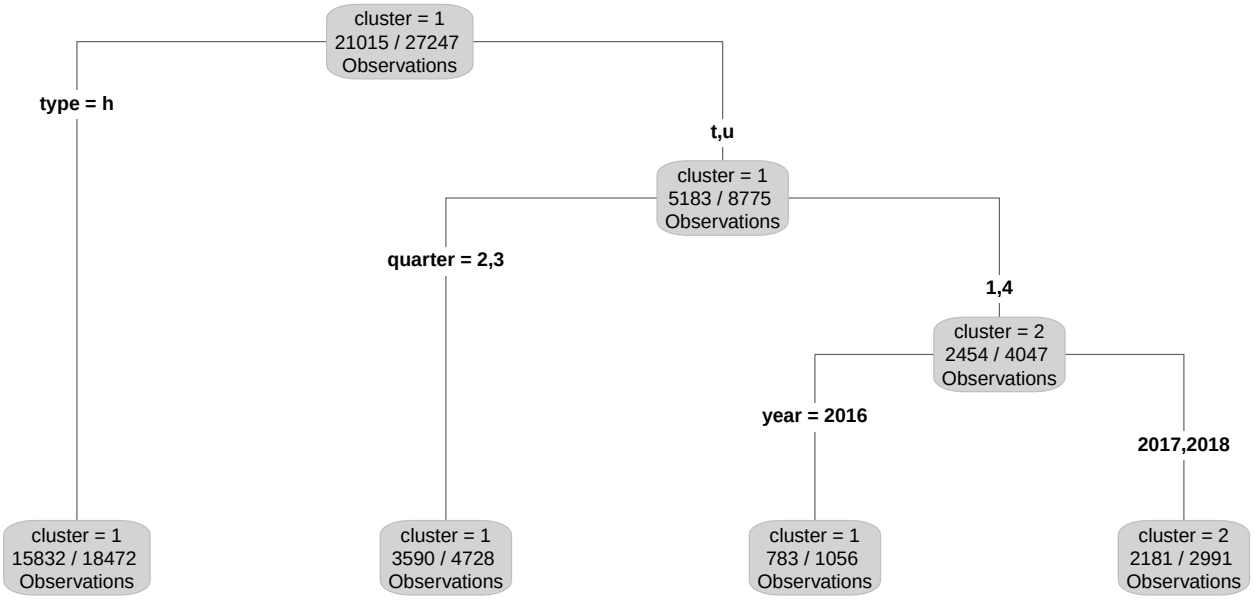

Figure 16: *Decision tree output predicting the clusters of missingness. Type of house, the year quarter, and year were important for predicting missingness cluster. The cluster with the most missingness was for quarters 1 and 4, for 2017 and 2018. Type of house, year, and year quarter are important features related to the missingness structure.*

4.3 Imputation and diagnostics

Two different imputation techniques are used: simple linear regression and K nearest neighbors. Values are imputed stepwise in ascending order of missingness. The `simputation` R package (van der Loo 2017) is used to impute the values, applying the track missings pattern described in Section 3.4.7, to assess imputed values.

Imputed data are combined by row binding imputation models. The data are reshaped into long format to compare imputed values for each model for 4 variables (Figure 17). Compared to KNN imputed values, the linear model imputed values closer to the mean value.

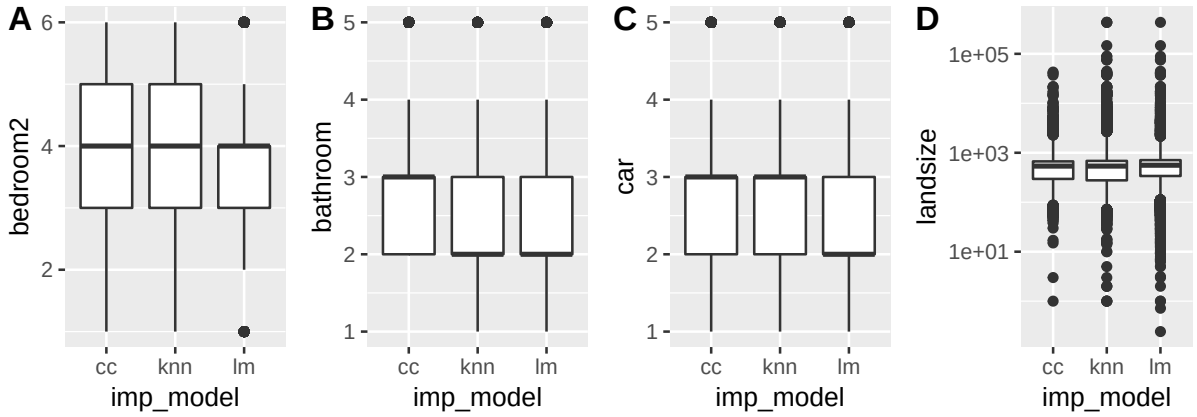

Figure 17: *Boxplots of complete case data, and data imputed with KNN or linear model for different variables. (A) number of bedrooms, (B) number of bathrooms, (C) number of carspots, and (D) landsize (on a log10 scale). KNN had similar results to complete case, and linear model had a lower median for cars and fewer extreme values for bedrooms.*

4.4 Assess model predictions

The coefficients of the linear model of log price vary for room (Figure 18), for different imputed datasets. Notably, complete cases resulted in underestimating the impact of room on log price. A partial residual plot (Figure 19) shows there is not much variation amongst the models from the different datasets.

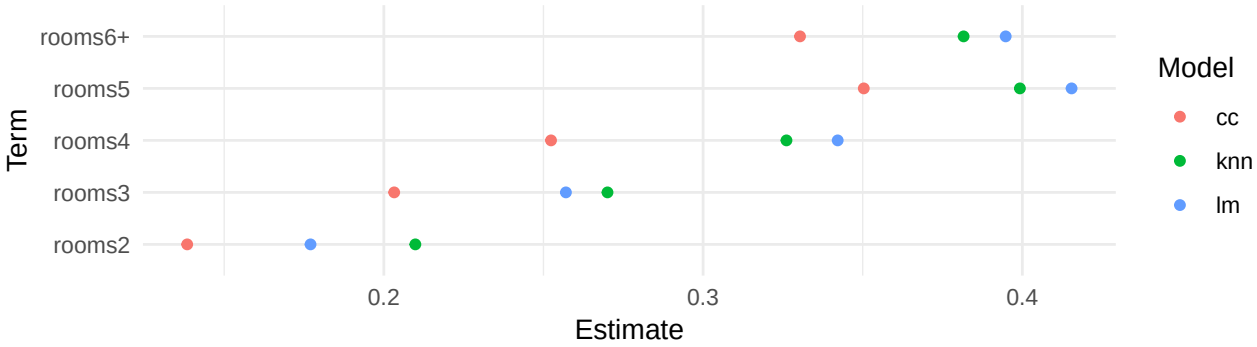

Figure 18: *Visualization of the variation in coefficients for linear model of log price for each of the different datasets for the number of rooms. In red is complete case (cc), in green is the knn imputed dataset, and in blue the imputed by linear model. Using the complete case dataset produced smaller coefficients compared to the imputed models.*

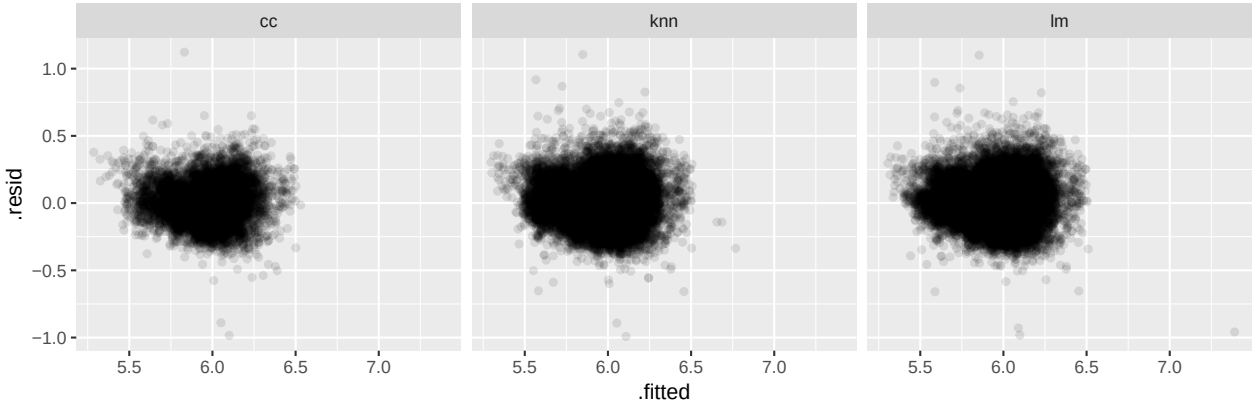

Figure 19: *Partial residual plot for each data set, complete cases (cc), imputed with KNN (knn), and imputed with a linear model (lm). There is not much variation amongst the different datasets from the different imputation methods.*

4.5 Summary

The `naniar` and `visdat` packages build on existing tidy tools and strike a compromise between automation and control that makes analysis efficient, readable, but not overly complex. Each tool has clear intent and effects - plotting or generating data or augmenting data in some way. This reduces repetition and typing for the user, making exploration of missing values easier as they follow consistent rules with a declarative interface.

5 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 modelling.

The new data structures discussed in the paper 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 (linked brushing) to explore missingness or animating between different sets of imputed values. New packages like `plotly` (Sievert 2018) facilitate interactive graphics, and packages like `gganimate` facilitate animations (Pedersen and Robinson 2017).

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.

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 et al. 2008)) . Evaluating imputed values using a testing framework like Buuren (2012) 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.

6 Acknowledgements

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`.

References

Barnett, A. G., McElwee, P., Nathan, A., Burton, N. W., and Turrell, G. (2017), "Identifying patterns of item missing survey data using latent groups: An observational study," *BMJ open*, bmjopen.bmj.com, 7, e017284.

Broman, K. W., and Woo, K. H. (2017), *Data organization in spreadsheets*, PeerJ Preprints; PeerJ Inc.

Buuren, S. van (2012), *Flexible imputation of missing data*, CRC Press.

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

Conway, J. R., Lex, A., and Gehlenborg, N. (2017), "UpSetR: An R package for the visualization of intersecting sets and their properties," *Bioinformatics*, 33, 2938–2940.

Cook, D., and Swayne, D. F. (2007), *Interactive and dynamic graphics for data analysis with R and GGobi*, Springer Publishing Company, Incorporated.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977), "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B, Statistical methodology*, [Royal Statistical Society, Wiley], 39, 1–38.

Donoho, D. (2017), "50 years of data science," *Journal of computational and graphical statistics: a joint publication of American Statistical Association, Institute of Mathematical Statistics, Interface Foundation of North America*, Taylor & Francis, 26, 745–766.

Durre, I., Menne, M. J., and Vose, R. S. (2008), "Strategies for evaluating quality assurance procedures," *Journal of Applied Meteorology and Climatology*, American Meteorological Society, 47, 1785–1791.

Ellis, S. E., and Leek, J. T. (2017), *How to share data for collaboration*, PeerJ Preprints; PeerJ Inc.

Gelman, A., and Hill, J. (2011), "Opening windows to the black box," *Journal of Statistical Software*, 40.

Grolemund, G., and Wickham, H. (2011), "Dates and times made easy with lubridate," *Journal of Statistical Software*, 40, 1–25.

Harrell Jr, F. E., Charles Dupont, and others. (2018), *Hmisc: Harrell miscellaneous*.

Henry, L., and Wickham, H. (2017), *Purrr: Functional programming tools*.

Honaker, J., and King, G. (2010), "What to do about missing values in Time-Series Cross-Section data," *American journal of political science*, 54, 561–581.

Honaker, J., King, G., and Blackwell, M. (2011), "Amelia II: A program for missing data," *Journal of Statistical Software*, 45, 1–47.

Josse, J., and Husson, F. (2016), "MissMDA: A package for handling missing values in multivariate data analysis," *Journal of Statistical Software, Articles*, 70, 1–31. https://doi.org/10.18637/jss.v070.i01.

Keeling, C. D., Piper, S. C., Bacastow, R. B., Wahlen, M., Whorf, T. P., Heimann, M., and Meijer, H. A. (2005), "Atmospheric CO2 and 13CO2 exchange with the terrestrial biosphere and oceans from 1978 to 2000: Observations and carbon cycle implications," in *A history of atmospheric CO2 and its effects on plants, animals, and ecosystems*, eds. I. T. Baldwin, M. M. Caldwell, G. Heldmaier, R. B. Jackson, O. L. Lange, H. A. Mooney, E.-D. Schulze, U. Sommer, J. R. Ehleringer, M. Denise Dearing, and T. E. Cerling, New York, NY: Springer New York, pp. 83–113.

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

Kuhn, M., and Wickham, H. (2018), *Recipes: Preprocessing tools to create design matrices*.

Lê, S., Josse, J., and Husson, F. (2008), "FactoMineR: A package for multivariate analysis," *Journal of Statistical Software*, 25, 1–18. https://doi.org/10.18637/jss.v025.i01.

Little, R. J. A. (1988), "A test of missing completely at random for multivariate data with missing values," *Journal of the American Statistical Association*, Taylor & Francis, 83, 1198–1202.

Milborrow, S. (2018), *Rpart.plot: Plot 'rpart' models: An enhanced version of 'plot.rpart'*.

Moritz, S., and Bartz-Beielstein, T. (2017), "imputeTS: Time Series Missing Value Imputation in R," *The R Journal*, 9, 207–218.

Pedersen, T. L., and Robinson, D. (2017), *Gganimate: A grammar of animated graphics*.

Pino, T. (2018), "Melbourne housing market," https://www.kaggle.com/anthonypino/melbourne-housing-market/version/21.

R by Alvaro A. Novo. Original by Joseph L. Schafer <jls@stat.psu.edu>., P. to (2013), *Norm: Analysis of multivariate normal datasets with missing values*.

Ross, Z., Wickham, H., and Robinson, D. (2017), *Declutter your R workflow with tidy tools*, PeerJ Preprints; PeerJ Inc.

Rubin, D. B. (1976), "Inference and missing data," *Biometrika*, Oxford University Press, 63, 581–592.

Schafer, J. (1999), *NORM: Multiple imputation of incomplete multivariate data under a normal model*, University Park: Pennsylvania State University, Department of Statistics.

Schafer, J. L., and Graham, J. W. (2002), "Missing data: Our view of the state of the art," *Psychological methods*, 7, 147–177.

Sievert, C. (2018), *Plotly for r*.

Silge, J., and Robinson, D. (2016), "Tidytext: Text mining and analysis using tidy data principles in r," *JOSS*, The Open Journal, 1. https://doi.org/10.21105/joss.00037.

Simon, G. A., and Simonoff, J. S. (1986), "Diagnostic plots for missing data in least squares regression," *Journal of the American Statistical Association*, Taylor & Francis, 81, 501–509.

Sterne, J. a C., White, I. R., Carlin, J. B., Spratt, M., Royston, P., Kenward, M. G., Wood, A. M., and Carpenter, J. R. (2009), "Multiple imputation for missing data in epidemiological and clinical research: Potential and pitfalls," *BMJ*, 338, b2393.

Swayne, D. F., and Buja, A. (1998), "Missing data in interactive high-dimensional data visualization," *Computational statistics*, researchgate.net, 1–8.

Therneau, T., and Atkinson, B. (2018), *Rpart: Recursive partitioning and regression trees*.

Tierney, N. (2017), "Visdat: Visualising whole data frames," *JOSS*, Journal of Open Source Software, 2, 355. https://doi.org/10.21105/joss.00355.

Tierney, N. J., Harden, F. A., Harden, M. J., and Mengersen, K. L. (2015), "Using decision trees to understand structure in missing data," *BMJ open*, bmjopen.bmj.com, 5, e007450.

Treisman, A. (1985), "Preattentive processing in vision," *Computer vision, graphics, and image processing*, 31, 156–177.

Unwin, A., Hawkins, G., Hofmann, H., and Siegl, B. (1996), "Interactive graphics for data sets with missing values: MANET," *Journal of computational and graphical statistics: a joint publication of American Statistical Association, Institute of Mathematical Statistics, Interface Foundation of North America*, [American Statistical Association, Taylor & Francis, Ltd., Institute of Mathematical Statistics, Interface Foundation of America], 5, 113–122.

van Buuren, S., and Groothuis-Oudshoorn, K. (2011), "mice: Multivariate imputation by chained equations in r," *Journal of Statistical Software*, 45, 1–67.

van der Loo, M. (2017), *Simputation: Simple imputation.*

Walker, K. (2018), *Tidycensus: Load us census boundary and attribute data as 'tidyverse' and 'sf'-ready data frames.*

Wang, E., Cook, D., and Hyndman, R. (2018), *Tsibble: Tidy temporal data frames and tools.*

Wickham, H. (2009), *Ggplot2: Elegant graphics for data analysis*, Springer-Verlag New York.

Wickham, H. (2014), "Tidy data," *Journal of statistical software*, 59, 1–23.

Wickham, H. (2017), "The tidy tools manifesto," https://cran.r-project.org/web/packages/tidyverse/vignettes/manifesto.html.

Wickham, H. (2018), *Stringr: Simple, consistent wrappers for common string operations.*

Wickham, H., and Bryan, J. (2017), *Readxl: Read excel files.*

Wickham, H., Francois, R., Henry, L., and Müller, K. (2017a), *Dplyr: A grammar of data manipulation.*

Wickham, H., and Grolemund, G. (2016), *R for data science: Import, tidy, transform, visualize, and model data*, "O'Reilly Media, Inc.".

Wickham, H., and Henry, L. (2018), *Tidyr: Easily tidy data with 'spread()' and 'gather()' functions.*

Wickham, H., Hester, J., and Francois, R. (2017b), *Readr: Read rectangular text data.*

Wickham, H., and Miller, E. (2018), *Haven: Import and export 'spss', 'stata' and 'sas' files.*