Some ideas for exploring missing data

Nicholas Tierney

Infectious Disease Ecology and Modelling group

Telethon Kids Institute, Perth, WA, Australia

## Introduction

If you’ve done data analysis, then chances are you’ve encountered missing data. I’ve encountered my fair share of missing data and I felt so frustrated by how hard it was to handle and wrangle with them that I have written two R packages (visdat, naniar), and several papers on the topic.

The goal of this article is to share some condensed ideas on exploring missing data, using the software I’ve written, naniar, and visdat. To that end, we will focus on four questions, how do we:

1. Start looking at missing data?
2. Explore missingness in variables?
3. Explore missingness relationships?
4. Explore imputed values

# Start looking at missing data?

options(tidyverse.quiet = TRUE)  
library(tidyverse)  
library(visdat)  
library(naniar)

We can use the visdat package (Tierney 2017) to get an overview of the missingness of an entire data set. It was heavily inspired by [csv-fingerprint](https://setosa.io/blog/2014/08/03/csv-fingerprints/), and functions like missmap, from Amelia (Honaker, King, and Blackwell 2011).

We can use visdat’s vis\_miss() function to visualise the missingness of a whole dataframe. [Figure 1](#fig-vis-miss-cluster-vanilla) gives an example where it displays the data as missing, or not missing, and provides the amount of missings in each columns.

vis\_miss(airquality)

|  |
| --- |
| Figure 1: An overview of missing data in the airquality dataset. The x axis shows the variables of the data, along with the amount of missingness in that variable, and the y axis shows the rows. Each cell represents the missingness of a datum. The overall missingness is given in a percentage below in the legend. We learn that there is nearly 5% missing data overall, the missing data occurs in Ozone and Solar.R, and mostly in Ozone. |

We learn that there is nearly 5% missing data overall, the missing data occurs in Ozone and Solar.R, and mostly in Ozone. The other variables do not have any missing data.

### Facetting in visdat

You can also split up the vis\_miss plots into several facetted plots via the facet argument. For example, in [Figure 2](#fig-vis-miss-facet) we facet by the Month variable.

vis\_miss(airquality, facet = Month)

|  |
| --- |
| Figure 2: An further overview of missing data in the airquality dataset. Similar to the previous graphic, we now present a facetted series of sub plots, one for each Month. We learn from this that most of the Ozone missingness happens in Month 6, and there aren’t missing values for months 6, 7, and 9. |

We learn from this that most of the Ozone missingness happens in Month 6, and there aren’t missing values for months 6, 7, and 9.

There are other functions in the visdat package that focus on other types of data, for example, vis\_value(), vis\_binary(), and vis\_compare(). To read more about the functions available in visdat see the vignette [“Using visdat”](https://CRAN.R-project.org/package=visdat/vignettes/using_visdat.html).

## Explore missingness in variables

Another approach to visualising the missingness in a dataset is to use the gg\_miss\_var plot, as seen in [Figure 3](#fig-gg-miss-var).

gg\_miss\_var(airquality)

|  |
| --- |
| Figure 3: Number of missing values for each variable. The x axis shows the number of missings, and the y axis shows each variable. We learn Ozone and Solar.R have the most missing data, and Ozone has the most missing data. |

This displays the number of missing values in each variable. We learn that there are pretty much only missing values for Ozone and Solar, with more for Ozone. Just like with vis\_miss(), we can add in facets in these plots, via the facet argument, see [Figure 4](#fig-gg-miss-var-facet).

gg\_miss\_var(airquality, facet = Month)

|  |
| --- |
| Figure 4: Similar to the above plot but one subplot for each Month. We learn that most of the missing values occur in month 6 for Ozone, and that months 6, 7, and 9 don’t have and Solar.R missing values. |

Where we learn that month 9 doesn’t have much missing data, and a lot of the missing data seems to occur in month 6.

There are more visualisations available in naniar (each starting with gg\_miss\_) - you can see these in the [“Gallery of Missing Data Visualisations” vignette](https://cran.r-project.org/package=naniar/vignettes/naniar-visualisation.html). The plots created with the gg\_miss family all have a basic theme, but you can customise them by adding components like a standard ggplot object:

gg\_miss\_var(airquality) +   
 theme\_bw() +   
 labs(y = "Number of missing observations")

It is also worth noting that for every visualisation of missing data in naniar, there is an accompanying function to extract the data used in the plot. This is important as the plot should not return a dataframe - but we want to make the data available for use by the user so it isn’t locked into a plot.

You can find these summary plots below, with miss\_var\_summary() providing the dataframe that gg\_miss\_var() is based on.

miss\_var\_summary(airquality)

# A tibble: 6 × 3  
 variable n\_miss pct\_miss  
 <chr> <int> <dbl>  
1 Ozone 37 24.2   
2 Solar.R 7 4.58  
3 Wind 0 0   
4 Temp 0 0   
5 Month 0 0   
6 Day 0 0

Which also works with group\_by():

airquality %>%   
 group\_by(Month) %>%   
 miss\_var\_summary()

Similarly, there is a data\_vis\_miss() function in the visdat package, which returns the data in the format that this visualisation requires.

data\_vis\_miss(airquality)

The aim of these is to provide the data required to make these visualisations, so if people want to create their own more customised versions of vis\_miss() or gg\_miss\_var() then they can do that.

## Exploring missingness relationships

We can identify key variables that are missing using vis\_miss() and gg\_miss\_var(), but for further exploration, we need to explore the relationship amongst the variables in this data:

* Ozone,
* Solar.R
* Wind
* Temp
* Month
* Day

Typically, when exploring this data, we might want to explore the variables Solar.R and Ozone. [Figure 5](#fig-example-geom-point) shows a scatter plot of solar radiation and ozone.

library(ggplot2)  
ggplot(airquality,   
 aes(x = Solar.R,   
 y = Ozone)) +   
 geom\_point()

|  |
| --- |
| Figure 5: Plot of Solar.R against Ozone. Solar.R is on the X axis and Ozone is on the Y axis. We learn that there is a slight positive correlation of Ozone with Solar.R |

The problem with this is ggplot does not handle missings be default, and removes the missing values. This makes them hard to explore. It also presents the strange question of “how do you visualise something that is not there?”. One approach to visualising missing data comes from [ggobi](https://en.wikipedia.org/wiki/GGobi) and [manet](http://www.rosuda.org/MANET/), where we impute “NA” values with values 10% lower than the minimum value in that variable, which puts these values in a margin area on the graphic.

This imputation is wrapped up in the geom\_miss\_point() ggplot2 geom. [Figure 6](#fig-geom-miss-point) illustrates this by exploring the relationship between Ozone and Solar radiation from the airquality dataset.

ggplot(airquality,   
 aes(x = Solar.R,   
 y = Ozone)) +   
 geom\_miss\_point() +   
 scale\_colour\_brewer(palette = "Dark2")

|  |
| --- |
| Figure 6: Improved plot of Ozone against Solar.R, we can now see the missing values are imputed 10% below the minimum value. The green dots on the x axis represent the Solar.R values that have missing Ozone. The green dots on the Y axis represent Ozone values that have missing Solar.R. The two dots in the bottom left corner are missing for both Ozone and Solar.R |

Being a proper ggplot geom, it supports all of the standard features of ggplot2, such as **facets** and **themes**. See [Figure 7](#fig-ggmissing-facet) for an example with faceting by month and a custom theme.

ggplot(airquality,   
 aes(x = Solar.R,   
 y = Ozone)) +   
 geom\_miss\_point() +   
 facet\_wrap(~Month) +   
 theme\_dark()

|  |
| --- |
| Figure 7: A faceted version of the improved Ozone against Solar.R plot where each month is split out into its own subplot. |

## Numerical summaries of missing values

naniar provide numerical summaries for missing data. Two convenient counters of complete values and missings are n\_miss() and n\_complete(). These work on both dataframes and vectors, similar to dplyr::n\_distinct()

dplyr::n\_distinct(airquality)

[1] 153

dplyr::n\_distinct(airquality$Ozone)

[1] 68

n\_miss(airquality)

[1] 44

n\_miss(airquality$Ozone)

[1] 37

n\_complete(airquality)

[1] 874

n\_complete(airquality$Ozone)

[1] 116

The syntax for the other numerical sumamries in naniar are miss\_, and then case, or var to refer to cases or variables. There are then summary, table, run, span, and cumsum options to explore missing data.

prop\_miss\_case and pct\_miss\_case return numeric value describing the proportion or percent of missing values in the dataframe.

prop\_miss\_case(airquality)

[1] 0.2745098

pct\_miss\_case(airquality)

[1] 27.45098

miss\_case\_summary() returns a numeric value that describes the number of missings in a given case (aka row), the percent of missings in that row.

miss\_case\_summary(airquality)

# A tibble: 153 × 3  
 case n\_miss pct\_miss  
 <int> <int> <dbl>  
 1 5 2 33.3  
 2 27 2 33.3  
 3 6 1 16.7  
 4 10 1 16.7  
 5 11 1 16.7  
 6 25 1 16.7  
 7 26 1 16.7  
 8 32 1 16.7  
 9 33 1 16.7  
10 34 1 16.7  
# ℹ 143 more rows

miss\_case\_table() tabulates the number of missing values in a case / row. Below, this shows the number of missings in a case:

miss\_case\_table(airquality)

# A tibble: 3 × 3  
 n\_miss\_in\_case n\_cases pct\_cases  
 <int> <int> <dbl>  
1 0 111 72.5   
2 1 40 26.1   
3 2 2 1.31

We can interpret this output as follows:

* There are 111 cases with 0 missings, which comprises about 72% of the data.
* There are then 40 cases with 1 missing, these make up 26% of the data.
* There are then 2 cases with 2 missing - these make up 1% of the data.

Similar to pct\_miss\_case(), prop\_miss\_case(), pct\_miss\_var() and prop\_miss\_var() returns the percent and proportion of variables that contain a missing value.

prop\_miss\_var(airquality)

[1] 0.3333333

pct\_miss\_var(airquality)

[1] 33.33333

miss\_var\_summary() then returns the number of missing values in a variable, and the percent missing in that variable.

miss\_var\_summary(airquality)

# A tibble: 6 × 3  
 variable n\_miss pct\_miss  
 <chr> <int> <dbl>  
1 Ozone 37 24.2   
2 Solar.R 7 4.58  
3 Wind 0 0   
4 Temp 0 0   
5 Month 0 0   
6 Day 0 0

Finally, miss\_var\_table(). This describes the number of missings in a variable.

* There are 4 variables with 0 missings, comprising 66.67% of variables in the dataset.
* There is 1 variable with 7 missings
* There is 1 variable with 37 missings

miss\_var\_table(airquality)

# A tibble: 3 × 3  
 n\_miss\_in\_var n\_vars pct\_vars  
 <int> <int> <dbl>  
1 0 4 66.7  
2 7 1 16.7  
3 37 1 16.7

There are also summary functions for exploring missings that occur over a particular span or period of the dataset, or the number of missings in a single run:

* miss\_var\_run(), and
* miss\_var\_span()

miss\_var\_run() can be particularly useful in time series data, as it allows you to provide summaries for the number of missings or complete values in a single run. The function miss\_var\_run() provides a data frame of the run length of missings and complete values. To explore this function we will use the built-in dataset, pedestrian, which contains hourly counts of pedestrians from four locations around Melbourne, Australia, from 2016.

To use miss\_var\_run(), you specify the variable that you want to explore the runs of missingness for, in this case, hourly\_counts:

miss\_var\_run(pedestrian,  
 hourly\_counts)

# A tibble: 35 × 2  
 run\_length is\_na   
 <int> <chr>   
 1 6628 complete  
 2 1 missing   
 3 5250 complete  
 4 624 missing   
 5 3652 complete  
 6 1 missing   
 7 1290 complete  
 8 744 missing   
 9 7420 complete  
10 1 missing   
# ℹ 25 more rows

The miss\_var\_span() function is used to determine the number of missings over a specified repeating span of rows in variable of a dataframe. Similar to miss\_var\_run(), you specify the variable that you wish to explore, you then also specify the size of the span with the span\_every argument.

miss\_var\_span(pedestrian,  
 hourly\_counts,  
 span\_every = 100)

# A tibble: 377 × 6  
 span\_counter n\_miss n\_complete prop\_miss prop\_complete n\_in\_span  
 <int> <int> <int> <dbl> <dbl> <int>  
 1 1 0 100 0 1 100  
 2 2 0 100 0 1 100  
 3 3 0 100 0 1 100  
 4 4 0 100 0 1 100  
 5 5 0 100 0 1 100  
 6 6 0 100 0 1 100  
 7 7 0 100 0 1 100  
 8 8 0 100 0 1 100  
 9 9 0 100 0 1 100  
10 10 0 100 0 1 100  
# ℹ 367 more rows

# Explore imputed values

Using the [simputation](https://cran.r-project.org/package=simputation) package, we impute values for Ozone using the impute\_lm() function, then visualise the data, as seen in [Figure 8](#fig-simpute-invisible).

library(simputation)  
library(dplyr)  
  
airquality %>%  
 impute\_lm(Ozone ~ Temp + Wind) %>%  
 ggplot(aes(x = Temp,  
 y = Ozone)) +   
 geom\_point()

|  |
| --- |
| Figure 8: Imputed values are not visible. A plot of Ozone by Temperature. The Imputed Ozone values are not visible because we have no way to identify them in the data. |

Note that we no longer get any warnings regarding missing observations - because they are all imputed! But this comes at a cost: we also no longer have information about where the imputations are - they are now sort of invisible.

We can track a copy of the missing data locations by using the function nabular(), which binds another dataset to the current one which notes the locations of the missing data. This is a really important idea with naniar, but to keep it brief, for each column with missing values, a new column is created to help identify misingness. For example, a new column called Solar.R\_NA is created:

nabular(airquality) |>   
 select(starts\_with("Solar.R")) |>   
 head()

# A tibble: 6 × 2  
 Solar.R Solar.R\_NA  
 <int> <fct>   
1 190 !NA   
2 118 !NA   
3 149 !NA   
4 313 !NA   
5 NA NA   
6 NA NA

The key takeaway here is that there is now a copy of the data bound to it, with each column ending in \_NA, and the values either being “NA” for missing, or “!NA” for not missing. For more details on the theory underlying this, and the benefits of this, we recommend reading our paper, “Expanding Tidy Data Principles to Facilitate Missing Data Exploration, Visualization and Assessment of Imputations” (Tierney and Cook 2023).

Using the shadow matrix to keep track of where the missings are, you can actually keep track of the imputations, colouring by what was previously missing in Ozone. For example, let’s create the nabular data, then impute the data and plot it in [Figure 9](#fig-simpute-visible-lm).

aq\_lm\_imputed <- airquality %>%  
 nabular() %>%   
 as.data.frame() %>%   
 impute\_lm(Ozone ~ Temp + Wind)  
ggplot(aq\_lm\_imputed,  
 aes(x = Temp,  
 y = Ozone,  
 colour = Ozone\_NA)) +   
 geom\_point() +   
 scale\_colour\_brewer(palette = "Dark2")

|  |
| --- |
| Figure 9: Regression imputed values of Ozone in a scatterplot of Ozone vs Temperature. Temperature is on the X axis and Ozone is on the Y axis, and the points are coloured by whether they are imputed - ‘NA’ indicates a previously missing value that has been imputed. We learn that the regression imputation imputes the values approximately amongst the rest of the data, at the surface level, this appears to be a good imputation. |

Let’s contrast this to how poor imputing just the mean value is, using impute\_mean\_all() in [Figure 10](#fig-simpute-visible-mean).

aq\_mean\_imputed <- airquality %>%  
 nabular() %>%   
 as.data.frame() %>%   
 impute\_mean\_all()  
  
head(aq\_mean\_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 42.12931 185.9315 14.3 56 5 5 NA NA !NA !NA  
6 28.00000 185.9315 14.9 66 5 6 !NA NA !NA !NA  
 Month\_NA Day\_NA  
1 !NA !NA  
2 !NA !NA  
3 !NA !NA  
4 !NA !NA  
5 !NA !NA  
6 !NA !NA

ggplot(aq\_mean\_imputed,  
 aes(x = Temp,  
 y = Ozone,  
 colour = Ozone\_NA)) +   
 geom\_point() +   
 scale\_colour\_brewer(palette = "Dark2")

|  |
| --- |
| Figure 10: The same plot as above, but using mean imputation. We learn that mean imputation places the imputed values all at a single point, which does not represent the variation we see in the data. |

# Conclusion

In this software corner we have demonstrated the use of the visdat and naniar R packages for exploring and understanding missing data.

# References

Honaker, James, Gary King, and Matthew Blackwell. 2011. “: A Program for Missing Data.” *Journal of Statistical Software* 45 (1): 1–47. <https://doi.org/10.18637/jss.v045.i07>.

Tierney, Nicholas. 2017. “: Visualising Whole Data Frames.” *The Journal of Open Source Software* 2 (16): 355. <https://doi.org/10.21105/joss.00355>.

Tierney, Nicholas, and Dianne Cook. 2023. “Expanding Tidy Data Principles to Facilitate Missing Data Exploration, Visualization and Assessment of Imputations.” *Journal of Statistical Software* 105 (7): 1–31. <https://doi.org/10.18637/jss.v105.i07>.