Dealing with Missing Data

Koji Mizumura

2019-4-1 - 2019-04-17

Table of Contents

# Why care about missing data (NA)

library(tidyverse)  
library(magrittr)

## Introduction

* working with real-world data = working with missing data
* Missing data can have unexpected effects on the analysis
* Bad imputation can lead to poor estimates and decisions

We will learn;

* What missing values are
* How to find missing data
* How to wrangle and tidy missing data
* Explore why is data missing

The definition is

missing values are values that should have been recored but were not.

x <- c(1,NA,3,NA,NA,5)  
  
any\_na(x)  
are\_na(x)  
n\_miss(x)  
prop\_miss(x)

## Using and finding missing values

When working with missing data, there are a couple of commands that you should be familiar with - firstly, you should be able to identify if there are any missing values, and where these are.

Using the any\_na() and are\_na() tools, identify which values are missing.

# Create x, a vector, with values NA, NaN, Inf, ".", and "missing"  
x <- c(NA, NaN, Inf, ".", "missing")  
  
library(rlang)  
library(Kmisc)  
  
  
# Use any\_na() and are\_na() on to explore the missings  
any\_na(x)  
are\_na(x)

## How many missing values are there?

One of the first things that you will want to check with a new dataset is if there are any missing missing values, and how many there are.

You could use are\_na() to and count up the missing values, but the most efficient way to count missings is to use the n\_miss() function. This will tell you the total number of missing values in the data.

You can then find the percent of missing values in the data with the pct\_miss function. This will tell you the percentage of missing values in the data.

# Use n\_miss() to count the total number of missing values in dat\_hw  
  
head(dat\_hw)  
n\_miss(dat\_hw)  
  
# Use n\_miss() on dat\_hw$weight to count the total number of missing values  
n\_miss(dat\_hw$weight)  
  
# Use n\_complete() on dat\_hw to count the total number of complete values  
n\_complete(dat\_hw)  
  
# Use n\_complete() on dat\_hw$weight to count the total number of complete values  
n\_complete(dat\_hw$weight)  
  
# Use prop\_miss() and prop\_complete() on dat\_hw to count the total number of missing values in each of the variables  
prop\_miss(dat\_hw)  
prop\_complete(dat\_hw)

## Working with missing values

R stores missing values as NA, which have some special behavior. Now that you can define missing data and understand how R stores missing values, can you predict what will happen when we operate with some missing values?

What is the output of the following four commands in R? Try them out in the code console to test them before you submit your answer.

## Why care about missing values?

Two summaries 1. Basic summaries of missingness: n\_miss, n\_complete 2. Dataframe summaries of missingness: miss\_var\_summary, miss\_case\_summary

These functions work with group\_by, dplyr functions.

## Summarizing missingness

Now that you understand the behavior of missing values in R, and how to count them, let’s scale up our summaries for cases (rows) and variables, using miss\_var\_summary() and miss\_case\_summary(), and also explore how they can be applied for groups in a dataframe, using the group\_by function from dplyr.

# Summarise missingness in each variable of the `airquality` dataset  
library(naniar)  
miss\_var\_summary(airquality)  
## # A tibble: 6 x 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  
  
# Summarise missingness in each case of the `airquality` dataset  
miss\_case\_summary(airquality)  
## # A tibble: 153 x 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  
## # ... with 143 more rows  
  
# Return the summary of missingness in each variable, grouped by Month, in the `airquality` dataset  
airquality %>% dplyr::group\_by(Month) %>% miss\_var\_summary()  
## # A tibble: 25 x 4  
## Month variable n\_miss pct\_miss  
## <int> <chr> <int> <dbl>  
## 1 5 Ozone 5 16.1  
## 2 5 Solar.R 4 12.9  
## 3 5 Wind 0 0   
## 4 5 Temp 0 0   
## 5 5 Day 0 0   
## 6 6 Ozone 21 70   
## 7 6 Solar.R 0 0   
## 8 6 Wind 0 0   
## 9 6 Temp 0 0   
## 10 6 Day 0 0   
## # ... with 15 more rows  
  
# Return the summary of missingness in each case, grouped by Month, in the `airquality` dataset  
airquality %>% dplyr::group\_by(Month) %>% miss\_case\_summary()  
## # A tibble: 153 x 4  
## Month case n\_miss pct\_miss  
## <int> <int> <int> <dbl>  
## 1 5 5 2 40  
## 2 5 27 2 40  
## 3 5 6 1 20  
## 4 5 10 1 20  
## 5 5 11 1 20  
## 6 5 25 1 20  
## 7 5 26 1 20  
## 8 5 1 0 0  
## 9 5 2 0 0  
## 10 5 3 0 0  
## # ... with 143 more rows

## Tabulating Missingness

The summaries of missingness we just calculated give us the number and percentage of missing observations for the cases and variables.

Another way to summarise missingness is by tabulating the number of times that there are 0, 1, 2, 3, missings in a variable, or in a case.

In this exercise we are going to tabulate the number of missings in each case and variable using miss\_var\_table() and miss\_case\_table(), and also combine these summaries with the the group\_by operator from dplyr. to explore the summaries over a grouping variable in the dataset.

# Tabulate missingness in each variable and case of the `airquality` dataset  
miss\_var\_table(airquality)  
## # A tibble: 3 x 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  
miss\_case\_table(airquality)  
## # A tibble: 3 x 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  
  
# Tabulate the missingness in each variable, grouped by Month, in the `airquality` dataset  
library(dplyr)  
airquality %>% group\_by(Month) %>% miss\_var\_table()  
## # A tibble: 12 x 4  
## Month n\_miss\_in\_var n\_vars pct\_vars  
## <int> <int> <int> <dbl>  
## 1 5 0 3 60  
## 2 5 4 1 20  
## 3 5 5 1 20  
## 4 6 0 4 80  
## 5 6 21 1 20  
## 6 7 0 4 80  
## 7 7 5 1 20  
## 8 8 0 3 60  
## 9 8 3 1 20  
## 10 8 5 1 20  
## 11 9 0 4 80  
## 12 9 1 1 20  
  
# Tabulate of missingness in each case, grouped by Month, in the `airquality` dataset  
airquality %>% group\_by(Month) %>% miss\_case\_table()  
## # A tibble: 11 x 4  
## Month n\_miss\_in\_case n\_cases pct\_cases  
## <int> <int> <int> <dbl>  
## 1 5 0 24 77.4   
## 2 5 1 5 16.1   
## 3 5 2 2 6.45  
## 4 6 0 9 30   
## 5 6 1 21 70   
## 6 7 0 26 83.9   
## 7 7 1 5 16.1   
## 8 8 0 23 74.2   
## 9 8 1 8 25.8   
## 10 9 0 29 96.7   
## 11 9 1 1 3.33

## Other summaries of missingness

Some summaries of missingness are particularly useful for different types of data. For example, miss\_var\_span() and miss\_var\_run().

* miss\_var\_span() calculates the number of missing values in a specified variable for a repeating span. This is really useful in time series data, to look for weekly (7 day) patterns of missingness
* miss\_var\_run() calculates the number of “runs” or “streaks” of missingness. This is useful to find unusual patterns of missingness, for example, you might find a repeating pattern of 5 complete and 5 missings.

Both miss\_var\_span() and miss\_var\_run() work with the group\_by operator from dplyr.

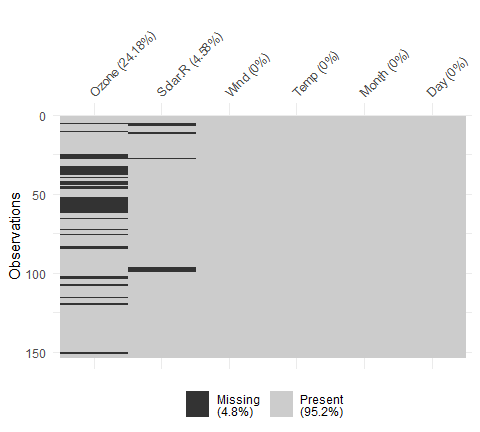
# Calculate the summaries for each run of missingness for the variable, hourly\_counts  
pedestrian %>% colnames()  
## [1] "hourly\_counts" "date\_time" "year" "month"   
## [5] "month\_day" "week\_day" "hour" "sensor\_id"   
## [9] "sensor\_name"  
miss\_var\_run(pedestrian, var = hourly\_counts)  
## # A tibble: 35 x 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   
## # ... with 25 more rows  
  
# Calculate the summaries for each span of missingness, for a span of 4000, for the variable hourly\_counts  
miss\_var\_span(pedestrian, var = hourly\_counts, span\_every = 4000)  
## # A tibble: 10 x 5  
## span\_counter n\_miss n\_complete prop\_miss prop\_complete  
## <int> <int> <dbl> <dbl> <dbl>  
## 1 1 0 4000 0 1   
## 2 2 1 3999 0.00025 1.000  
## 3 3 121 3879 0.0302 0.970  
## 4 4 503 3497 0.126 0.874  
## 5 5 745 3255 0.186 0.814  
## 6 6 0 4000 0 1   
## 7 7 1 3999 0.00025 1.000  
## 8 8 0 4000 0 1   
## 9 9 745 3255 0.186 0.814  
## 10 10 432 3568 0.108 0.892  
  
# For each `month` variable, calculate the run of missingness for hourly\_counts  
pedestrian %>% group\_by(month) %>% miss\_var\_run(var = hourly\_counts)  
## # A tibble: 51 x 3  
## month run\_length is\_na   
## <ord> <int> <chr>   
## 1 January 2976 complete  
## 2 February 2784 complete  
## 3 March 2976 complete  
## 4 April 888 complete  
## 5 April 552 missing   
## 6 April 1440 complete  
## 7 May 744 complete  
## 8 May 72 missing   
## 9 May 2160 complete  
## 10 June 2880 complete  
## # ... with 41 more rows  
  
# For each `month` variable, calculate the span of missingness of a span of 2000, for the variable hourly\_counts  
pedestrian %>% group\_by(month) %>% miss\_var\_span(var = hourly\_counts, span\_every = 2000)  
## # A tibble: 25 x 6  
## month span\_counter n\_miss n\_complete prop\_miss prop\_complete  
## <ord> <int> <int> <dbl> <dbl> <dbl>  
## 1 January 1 0 2000 0 1   
## 2 January 2 0 2000 0 1   
## 3 February 1 0 2000 0 1   
## 4 February 2 0 2000 0 1   
## 5 March 1 0 2000 0 1   
## 6 March 2 0 2000 0 1   
## 7 April 1 552 1448 0.276 0.724  
## 8 April 2 0 2000 0 1   
## 9 May 1 72 1928 0.036 0.964  
## 10 May 2 0 2000 0 1   
## # ... with 15 more rows

## how do we visualize missing values?

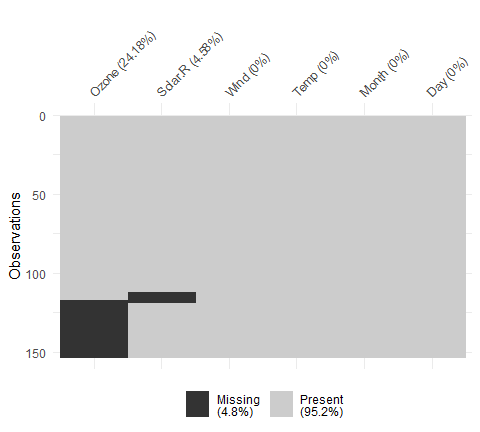
Built-in visualization in naniar package.

We will cover: - How to get a bird’s eye view of the data - How to look at missings in the variable and cases - How to generate visualizations for missing spans and across groups in thd data

naniar::vis\_miss(airquality)

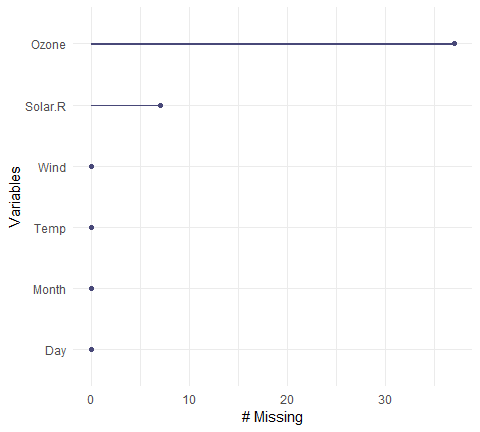


# actiate clustering  
naniar::vis\_miss(airquality, cluster = T)

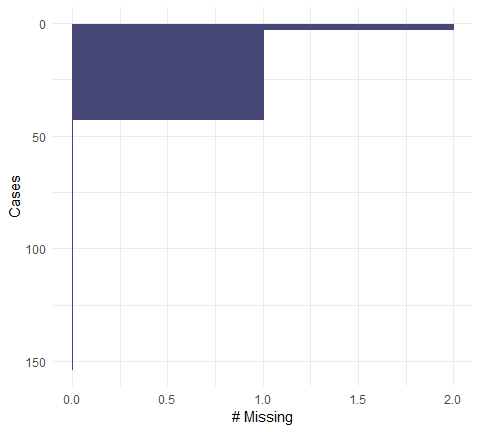


When we want to look at missings in variables and cases.

library(naniar)  
gg\_miss\_var(airquality)

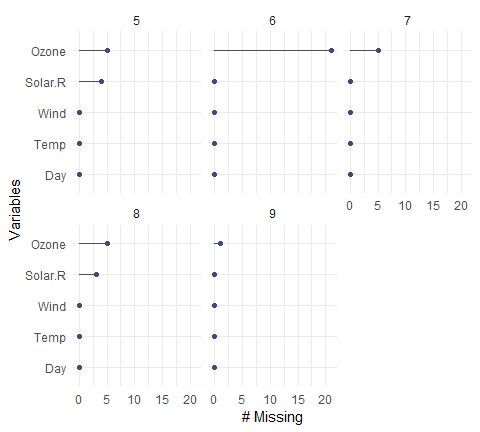


gg\_miss\_case(airquality)



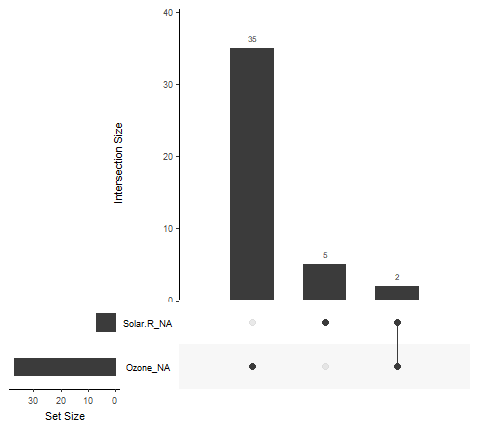
gg\_miss\_var and gg\_miss\_case can be used for faceting together.

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



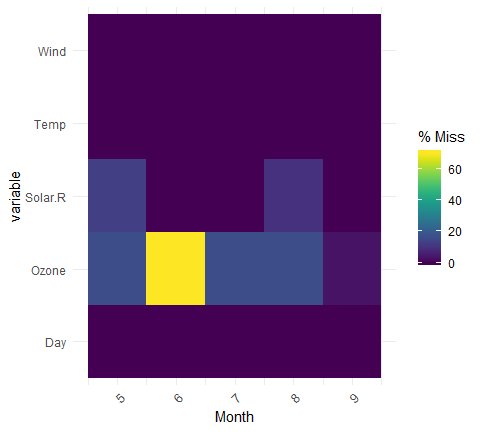
With combination, we can use gg\_miss\_upset

library(UpSetR)  
  
airquality %>%   
 as\_shadow\_upset() %>%   
 upset()

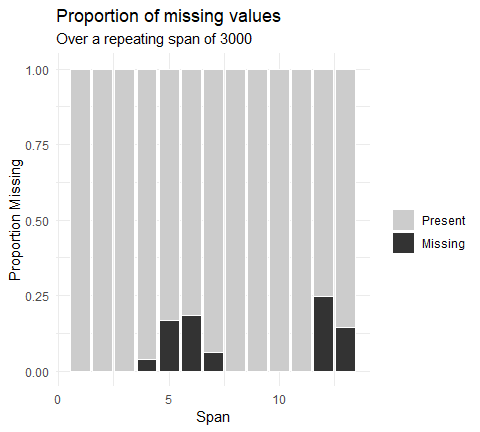


# gg\_miss\_upset(airquality)

gg\_miss\_fct(x = airquality, fct = Month)



gg\_miss\_span(pedestrian, hourly\_counts,  
 span\_every = 3000)

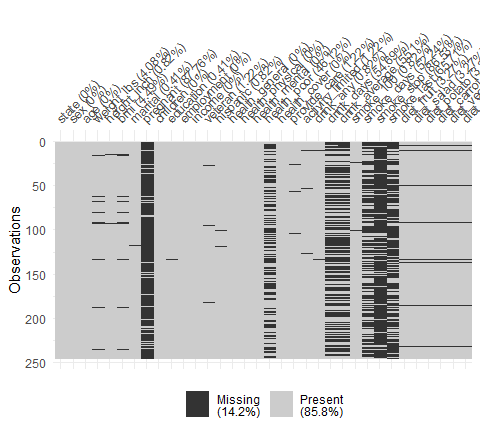


## Your first missing data visualizations

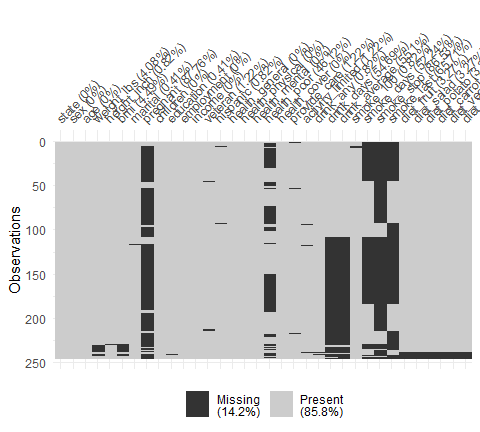
It can be difficult to get a handle on where the missing values are in your data, and here is where visualization can really help.

The function vis\_miss() creates an overview visualization of the missingness in the data. It also has options to cluster rows based on missingness, using cluster = TRUE; as well as options for sorting the columns, from most missing to least missing (sort\_miss = TRUE).

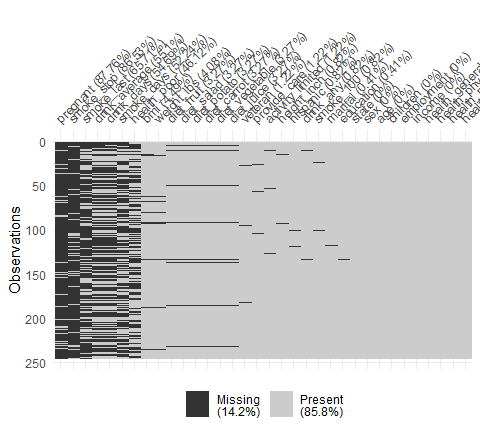
# Visualize all of the missingness in the `riskfactors` dataset  
vis\_miss(riskfactors)



# Visualize and cluster all of the missingness in the `riskfactors` dataset  
vis\_miss(riskfactors, cluster = TRUE)



# visualise and sort the columns by missingness in the `riskfactors` dataset  
vis\_miss(riskfactors, sort\_miss = TRUE)

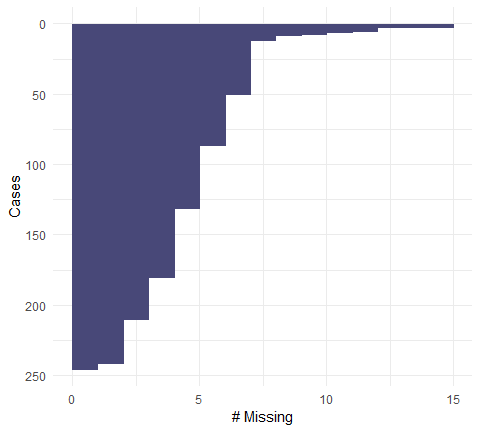


## Visualizing missing cases and variables

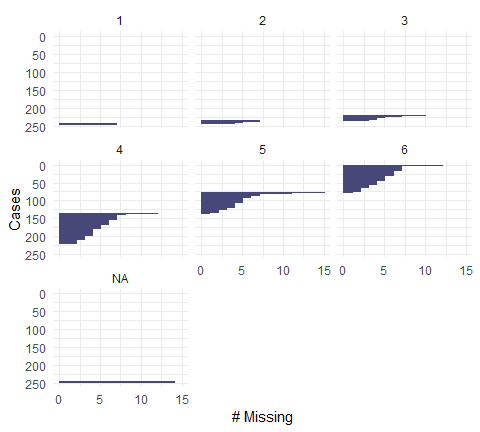
To get a clear picture of the missingness across variables and cases, use gg\_miss\_var() and gg\_miss\_case(). These are the visual counterpart to miss\_var\_summary() and miss\_case\_summary().

These can be split up into multiple plots with one for each category by choosing a variable to facet by.

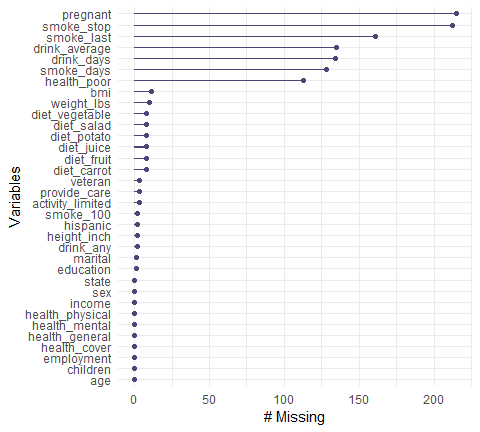
# Visualize the number of missings in cases using `gg\_miss\_case()`  
gg\_miss\_case(riskfactors)



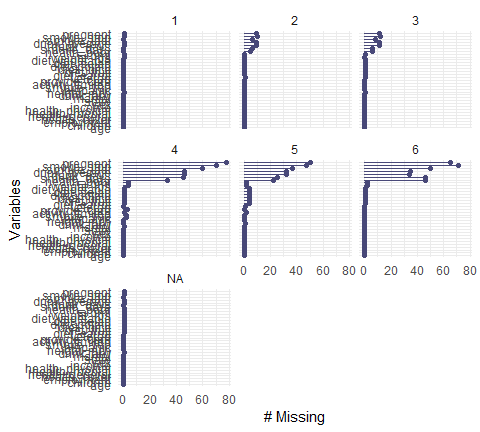
# Explore the number of missings in cases using `gg\_miss\_case()` and facet by the variable `education`  
gg\_miss\_case(riskfactors, facet = education)



# Visualize the number of missings in variables using `gg\_miss\_var()`  
gg\_miss\_var(riskfactors)



# Explore the number of missings in variables using `gg\_miss\_var()` and facet by the variable `education`  
gg\_miss\_var(riskfactors, facet = education)

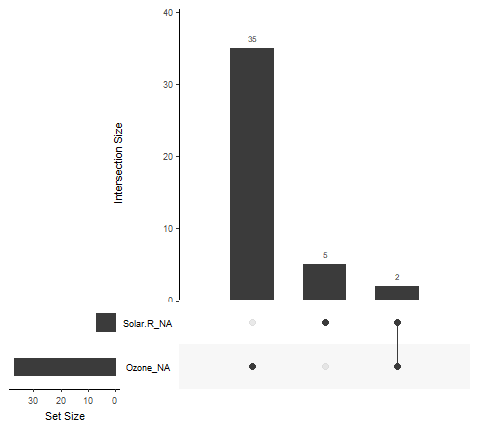


## Visualizing missingess patterns

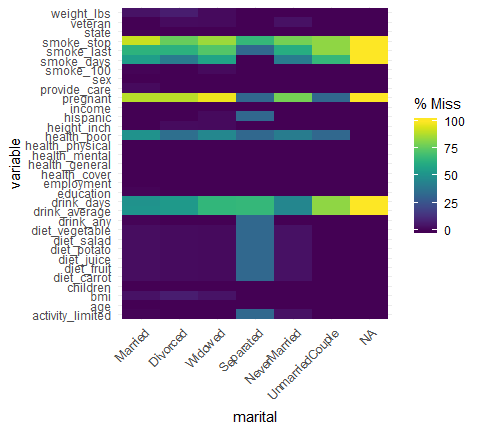
Let’s practice a few different ways to visualize patterns of missingness using:

* gg\_miss\_upset() to give an overall pattern of missingness.
* gg\_miss\_fct() for a dataset that has a factor of interest: marriage.
* and gg\_miss\_span() to explore the missingness in a time series dataset.

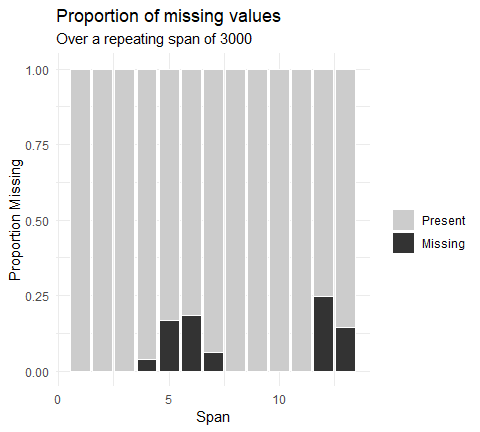
# Using the airquality dataset, explore the missingness pattern using gg\_miss\_upset()  
library(naniar)  
gg\_miss\_upset(airquality)



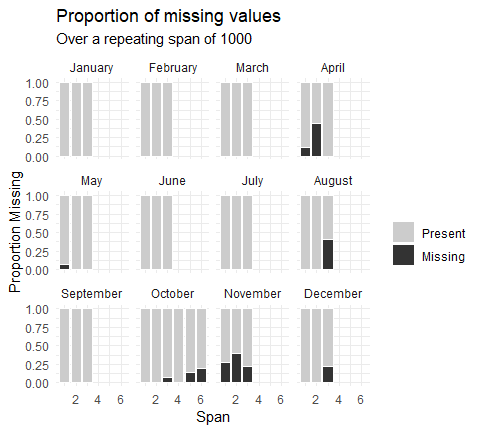
# With the riskfactors dataset, explore how the missingness changes across the marital variable using gg\_miss\_fct()  
gg\_miss\_fct(x = riskfactors, fct = marital)



# Using the pedestrian dataset, explore how the missingness of hourly\_counts changes over a span of 3000   
gg\_miss\_span(pedestrian, var = hourly\_counts, span\_every = 3000)



# Using the pedestrian dataset, explore the impact of month by facetting by month  
# and explore how missingness changes for a span of 1000  
gg\_miss\_span(pedestrian, var = hourly\_counts , span\_every = 1000, facet = month)



# Wrangling and tidying up missing values

## Searching for and replacing missing values

* How to look for hidden missing values
* Replacing missing value labels with NA
* Checking your assumptions on missingness

To search for and replace missing values

The idea is NA, however, it can be coded incorrectly (e.g., “missing” etc.).

chaos <- tibble::tribble(  
 ~ score, ~ grade, ~ place,  
 3, "N/A", -99,  
 -99, "E", 97,  
 4, "missing", 95,  
 -99, "na", 92,  
 7, "n/a", -98,  
 1, " ", "missing",  
 12, ".", 88,  
 16, " ", ".",  
 9, "N/a", 86  
)  
  
chaos  
## # A tibble: 9 x 3  
## score grade place   
## <dbl> <chr> <chr>   
## 1 3 N/A -99   
## 2 -99 E 97   
## 3 4 missing 95   
## 4 -99 na 92   
## 5 7 n/a -98   
## 6 1 " " missing  
## 7 12 . 88   
## 8 16 " " .   
## 9 9 N/a 86

chaos %>%   
 miss\_scan\_count(search = list("N/A", "N/a"))  
## # A tibble: 3 x 2  
## Variable n  
## <chr> <int>  
## 1 score 0  
## 2 grade 2  
## 3 place 0

chaos %>%   
 replace\_with\_na(  
 replace = list(grade = c("N/A", "N/a"))  
 )  
## # A tibble: 9 x 3  
## score grade place   
## <dbl> <chr> <chr>   
## 1 3 <NA> -99   
## 2 -99 E 97   
## 3 4 missing 95   
## 4 -99 na 92   
## 5 7 n/a -98   
## 6 1 " " missing  
## 7 12 . 88   
## 8 16 " " .   
## 9 9 <NA> 86

### Scoped variances of replace\_with\_na

replace\_with\_na can be repetitive:

* Use it across many different variables and values
* Complex cases, replacing values less than -1, only affect character columns.

Thus, there are other functions 1. replace\_with\_na\_all(): All variables 2. replace\_with\_na\_at(): A subset of selected viriables. 3. replace\_with\_na\_if(): A subset of variables that fullfill some conditions (numeric, character)

chaos %>%   
 replace\_with\_na\_all(condition = ~.x==-99)  
## # A tibble: 9 x 3  
## score grade place   
## <dbl> <chr> <chr>   
## 1 3 N/A <NA>   
## 2 NA E 97   
## 3 4 missing 95   
## 4 NA na 92   
## 5 7 n/a -98   
## 6 1 " " missing  
## 7 12 . 88   
## 8 16 " " .   
## 9 9 N/a 86  
  
chaos %>%   
 replace\_with\_na\_all(condition = ~ .x %in% c("N/A", "missing", "na"))  
## # A tibble: 9 x 3  
## score grade place  
## <dbl> <chr> <chr>  
## 1 3 <NA> -99   
## 2 -99 E 97   
## 3 4 <NA> 95   
## 4 -99 <NA> 92   
## 5 7 n/a -98   
## 6 1 " " <NA>   
## 7 12 . 88   
## 8 16 " " .   
## 9 9 N/a 86

## Using miss\_scan\_count()

You have a dataset with missing values coded as "N/A", "missing", and "na". But before we go ahead and start replacing these with NA, we should get an idea of how big the problem is.

Use miss\_scan\_count to count the possible missings in the dataset, pacman, a dataset of pacman scores, containing three columns:

* year: the year that person made that score.
* initial: the initials of the person.
* score: the scores of that person.

library(pacman)  
  
# Explore the strange missing values "N/A"  
miss\_scan\_count(data = pacman, search = list("N/A"))  
  
# Explore the strange missing values "missing"  
miss\_scan\_count(data = pacman, search = list("missing"))  
  
# Explore the strange missing values "na"  
miss\_scan\_count(data = pacman, search = list("na"))  
  
# Explore the strange missing values " " (a single space)  
miss\_scan\_count(data = pacman, search = list(" "))  
  
# Explore all of the strange missing values, "N/A", "missing", "na", " "  
miss\_scan\_count(data = pacman, search = list("N/A", "missing", "na", " "))

## Using replace\_with\_na

Following on from the previous dataset, we now know that we have a few strange missing values.

Now, we are going to do something about it, and replace these values with missings (e.g. NA) using the function replace\_with\_na().

# Print the top of the pacman data using `head()`  
head(pacman)  
  
# Replace the strange missing values "N/A", "na", and "missing" with `NA` for the variables, year, and score  
pacman\_clean <- replace\_with\_na(pacman, replace = list(year = c("N/A", "na", "missing"),  
 score = c("N/A", "na", "missing")))  
   
# Test if `pacman\_clean` still has these values in it?  
miss\_scan\_count(pacman\_clean, search = list("N/A", "na", "missing"))

## Using replace\_with\_na scoped variants

To reduce code repetition when replacing values with NA, use the “scoped vatriants” of replace\_with\_na():

* replace\_with\_na\_at()
* replace\_with\_na\_if()
* replace\_with\_na\_all()

The syntax of replacement looks like this:

~.x == "N/A"

This replaces all cases that are equal to “N/A”.

~.x %in% c("N/A", "missing", "na", " ")

Replaces all cases that have “N/A”, “missing”, “na”, or " ".

# Use `replace\_with\_na\_at()` to replace with NA  
replace\_with\_na\_at(pacman,  
 .vars = c("year", "month", "day"),   
 ~.x %in% c("N/A", "missing", "na", " "))  
  
# Use `replace\_with\_na\_if()` to replace with NA the character values using `is.character`  
replace\_with\_na\_if(pacman,  
 .predicate = is.character,   
 ~.x %in% c("N/A", "missing", "na", " "))  
  
# Use `replace\_with\_na\_all()` to replace with NA  
replace\_with\_na\_all(pacman, ~.x %in% c("N/A", "missing", "na", " "))

## Filling down missing values

How to handle implicit missing values.

Perspective on missing data,

* Explicitly: They are missing with NA
* Implicitly: Not shown in the data, but implied

We will use tidyr::complete for checking implicit missingness.

tetris %>%   
 tidyr::complete(name, time)

Sometimes, the tidyr::fill is useful for filling data. But, the warning is that it only solves only a few missing data problems.

## Fix implicit missings using complete()

We are going to explore a new dataset, frogger.

This dataset contains 4 scores per player recorded at different times:

* morning
* afternoon
* evening
* late\_night

Every player should have played 4 games, one at each of these times, but it looks like not every player completed all of these games.

Use the complete() function to make these implicit missing values explicit

# Print the frogger data to have a look at it  
frogger  
  
# Use `complete()` on the `time` variable to make implicit missing values explicit  
frogger\_tidy <- frogger %>% complete(name, time)

## Fix explicit missings using fill()

One type of missing value that can be obvious to deal with is where the first entry of a group is where the first entry of a group is given, but subsequent entries are marked NA.

These missing values often result from empty values in spreadsheets to avoid entering multiple names multiple times; as well as for “human readability”.

This type of problem can be solved by using the fill() function. from the tidyr package.

# Print the frogger data to have a look at it  
frogger  
  
# Use `fill()` to fill down the name variable in the frogger dataset  
frogger %>% fill(name)

## Using complete() and fill() together

Use complete() and fill() together to fix explicit and implicitly missing values in the frogger dataset.

# Print the frogger data to have a look at it  
frogger  
  
# Correctly fill() and complete() missing values so that our dataset becomes sensible  
  
frogger %>%   
 fill(name) %>%  
 complete(name, time)

## Missing data dependence

* **MCAR**: missing completely at random
* **MAR**: missing at random
* **MNAR**: Missing not at random

MCAR, missingness has no association with any data you have observed, or not observed.

For **MCAR**, the imputation is advisable. Deleting observation may reduce sample size, limiting inference, but will not bias.

MAR, missing depends on data observed, but not data observed.

For **MAR**, the imputation is recommended, delteting observaion is not ideal, as it may lead to bias.

MNAR, missingness of the reponse is related to an unobserved value relevant to the asessment of interest.

For **MNAR**, data will be biased from deletion and imoutation, inference can be limited, proceed with caution.

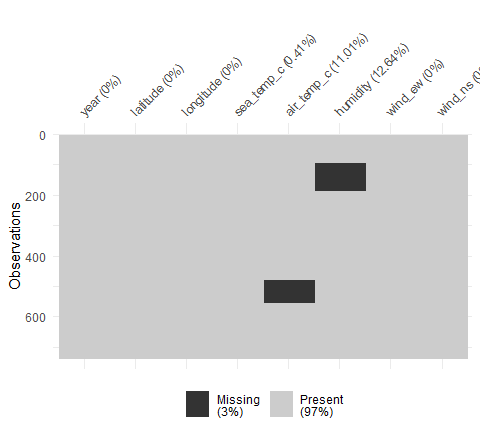
## Exploring missingness dependence

To learn about the structure of the missingness in data, you can explore how sorting changes how missingness is presented.

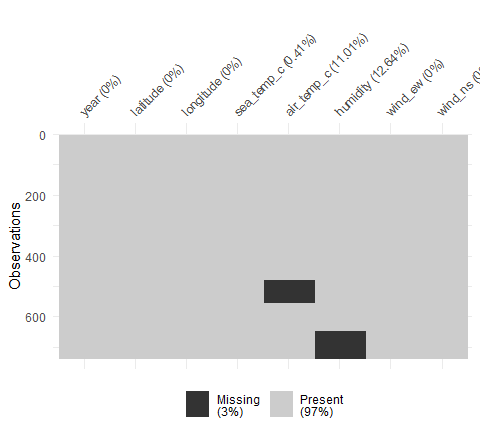
For the oceanbuoys dataset, explore the missingness with vis\_miss(), and then arrange by a few different variables

This is not a definitive process, but it will get you started to ask the right questions of your data. We explore more powerful techniques in the next chapter.

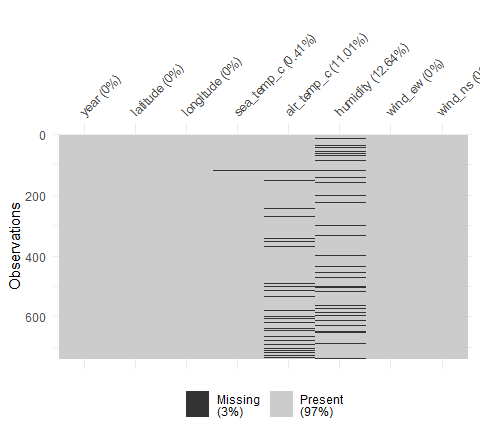
# naniar::vis\_miss()  
# Arrange by year  
oceanbuoys %>% arrange(year) %>% vis\_miss()



# Arrange by latitude  
oceanbuoys %>% arrange(latitude) %>% vis\_miss()



# Arrange by wind\_ew (wind east west)  
oceanbuoys %>% arrange(wind\_ew) %>% vis\_miss()

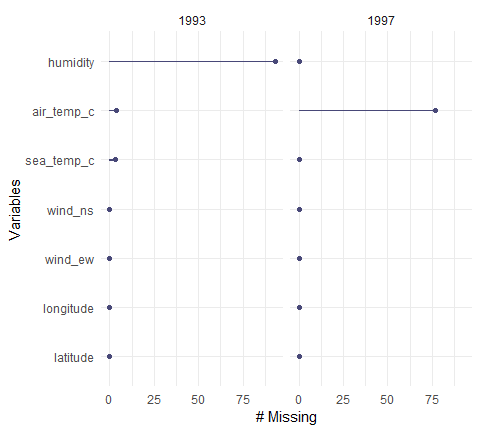


## Further exploring missingeness dependence

Using the information from earlier on the oceanbuoys dataset, which of these statements makes the most appropriate statement on the missingness type?

Try using gg\_miss\_var(), and gg\_miss\_case(), facetting by year to get more information. For example:

library(naniar)  
gg\_miss\_var(oceanbuoys, facet = year)



# Testing missing relationships

## Missing data workflows: the shadow matrix and nabular data

Census data containing, income and education

Shadow matrix is a clear representation of binary form of data(0, 1 or !NA or NA). The two main features are

1. Coordinated names
2. Clear values

Nabular data can be used instead of shadow matrix (The nabular data include both NA and original data input). The nabular data can be created by bind\_shadow.

## Creating shadow matrix data

Missing data can be tricky to think about, as they don’t usually proclaim themselves for you, and instead hide amongst the weeds of the data.

One way to help expose missing values is to change the way we think about the data - by thinking about every single data value being missing or not missing.

The as\_shadow() function in R transforms a dataframe into a shadow matrix, a special data format where the values are either missing (NA), or Not Missing (!NA).

# Create shadow matrix data with `as\_shadow()`  
as\_shadow(oceanbuoys)  
## # A tibble: 736 x 8  
## year\_NA latitude\_NA longitude\_NA sea\_temp\_c\_NA air\_temp\_c\_NA humidity\_NA  
## <fct> <fct> <fct> <fct> <fct> <fct>   
## 1 !NA !NA !NA !NA !NA !NA   
## 2 !NA !NA !NA !NA !NA !NA   
## 3 !NA !NA !NA !NA !NA !NA   
## 4 !NA !NA !NA !NA !NA !NA   
## 5 !NA !NA !NA !NA !NA !NA   
## 6 !NA !NA !NA !NA !NA !NA   
## 7 !NA !NA !NA !NA !NA !NA   
## 8 !NA !NA !NA !NA !NA !NA   
## 9 !NA !NA !NA !NA !NA !NA   
## 10 !NA !NA !NA !NA !NA !NA   
## # ... with 726 more rows, and 2 more variables: wind\_ew\_NA <fct>,  
## # wind\_ns\_NA <fct>  
  
# Create nabular data by binding the shadow to the data with `bind\_shadow()`  
bind\_shadow(oceanbuoys)  
## # A tibble: 736 x 16  
## year latitude longitude sea\_temp\_c air\_temp\_c humidity wind\_ew wind\_ns  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1997 0 -110 27.6 27.1 79.6 -6.40 5.40  
## 2 1997 0 -110 27.5 27.0 75.8 -5.30 5.30  
## 3 1997 0 -110 27.6 27 76.5 -5.10 4.5   
## 4 1997 0 -110 27.6 26.9 76.2 -4.90 2.5   
## 5 1997 0 -110 27.6 26.8 76.4 -3.5 4.10  
## 6 1997 0 -110 27.8 26.9 76.7 -4.40 1.60  
## 7 1997 0 -110 28.0 27.0 76.5 -2 3.5   
## 8 1997 0 -110 28.0 27.1 78.3 -3.70 4.5   
## 9 1997 0 -110 28.0 27.2 78.6 -4.20 5   
## 10 1997 0 -110 28.0 27.2 76.9 -3.60 3.5   
## # ... with 726 more rows, and 8 more variables: year\_NA <fct>,  
## # latitude\_NA <fct>, longitude\_NA <fct>, sea\_temp\_c\_NA <fct>,  
## # air\_temp\_c\_NA <fct>, humidity\_NA <fct>, wind\_ew\_NA <fct>,  
## # wind\_ns\_NA <fct>  
  
# Bind only the variables with missing values by using bind\_shadow(only\_miss = TRUE)  
bind\_shadow(oceanbuoys, only\_miss = TRUE)  
## # A tibble: 736 x 11  
## year latitude longitude sea\_temp\_c air\_temp\_c humidity wind\_ew wind\_ns  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1997 0 -110 27.6 27.1 79.6 -6.40 5.40  
## 2 1997 0 -110 27.5 27.0 75.8 -5.30 5.30  
## 3 1997 0 -110 27.6 27 76.5 -5.10 4.5   
## 4 1997 0 -110 27.6 26.9 76.2 -4.90 2.5   
## 5 1997 0 -110 27.6 26.8 76.4 -3.5 4.10  
## 6 1997 0 -110 27.8 26.9 76.7 -4.40 1.60  
## 7 1997 0 -110 28.0 27.0 76.5 -2 3.5   
## 8 1997 0 -110 28.0 27.1 78.3 -3.70 4.5   
## 9 1997 0 -110 28.0 27.2 78.6 -4.20 5   
## 10 1997 0 -110 28.0 27.2 76.9 -3.60 3.5   
## # ... with 726 more rows, and 3 more variables: sea\_temp\_c\_NA <fct>,  
## # air\_temp\_c\_NA <fct>, humidity\_NA <fct>

## Performing grouped summaries of missingness

Now that you can create nabular data, let’s use it to explore the data. Let’s calculate summary statistics based on the missingness of another variable.

To do this we are going to use the following steps:

* First, bind\_shadow() turns the data into nabular data.
* Next, perform some summaries on the data using group\_by() and summarise() to calculate the mean and standard deviation, using the mean() and sd() functions.

# `bind\_shadow()` and `group\_by()` humidity missingness (`humidity\_NA`)  
oceanbuoys %>%  
 bind\_shadow() %>%  
 group\_by(humidity\_NA) %>%   
 summarise(wind\_ew\_mean = mean(wind\_ew), # calculate mean of wind\_ew  
 wind\_ew\_sd = sd(wind\_ew)) # calculate standard deviation of wind\_ew  
## # A tibble: 2 x 3  
## humidity\_NA wind\_ew\_mean wind\_ew\_sd  
## <fct> <dbl> <dbl>  
## 1 !NA -3.78 1.90  
## 2 NA -3.30 2.31  
   
# Repeat this, but calculating summaries for wind north south (`wind\_ns`).  
oceanbuoys %>%  
 bind\_shadow() %>%  
 group\_by(humidity\_NA) %>%  
 summarise(wind\_ns\_mean = mean(wind\_ns),  
 wind\_ns\_sd = sd(wind\_ns))  
## # A tibble: 2 x 3  
## humidity\_NA wind\_ns\_mean wind\_ns\_sd  
## <fct> <dbl> <dbl>  
## 1 !NA 2.78 2.06  
## 2 NA 1.66 2.23

## Further exploring more combinations of missingness

It can be useful to get a bit of extra information about the number of cases in each missing condition.

In this exercise, we are going to add information about the number of observed cases using n() inside the summarise() function.

We will then add an additional level of grouping by looking at the combination of humidity being missing (humidity\_NA) and sea temperature being missing (sea\_temp\_c\_NA).

# Summarise wind\_ew by the missingness of `air\_temp\_c\_NA`  
oceanbuoys %>%   
 bind\_shadow() %>%  
 group\_by(air\_temp\_c\_NA) %>%  
 summarise(wind\_ew\_mean = mean(wind\_ew),  
 wind\_ew\_sd = sd(wind\_ew),  
 n\_obs = n())  
## # A tibble: 2 x 4  
## air\_temp\_c\_NA wind\_ew\_mean wind\_ew\_sd n\_obs  
## <fct> <dbl> <dbl> <int>  
## 1 !NA -3.91 1.85 655  
## 2 NA -2.17 2.14 81  
  
# Summarise wind\_ew by missingness of `air\_temp\_c\_NA` and `humidity\_NA`  
oceanbuoys %>%   
 bind\_shadow() %>%  
 group\_by(air\_temp\_c\_NA, humidity\_NA) %>%  
 summarise(wind\_ew\_mean = mean(wind\_ew),  
 wind\_ew\_sd = sd(wind\_ew),  
 n\_obs = n())  
## # A tibble: 4 x 5  
## # Groups: air\_temp\_c\_NA [?]  
## air\_temp\_c\_NA humidity\_NA wind\_ew\_mean wind\_ew\_sd n\_obs  
## <fct> <fct> <dbl> <dbl> <int>  
## 1 !NA !NA -4.01 1.74 565  
## 2 !NA NA -3.24 2.31 90  
## 3 NA !NA -2.06 2.08 78  
## 4 NA NA -4.97 1.74 3

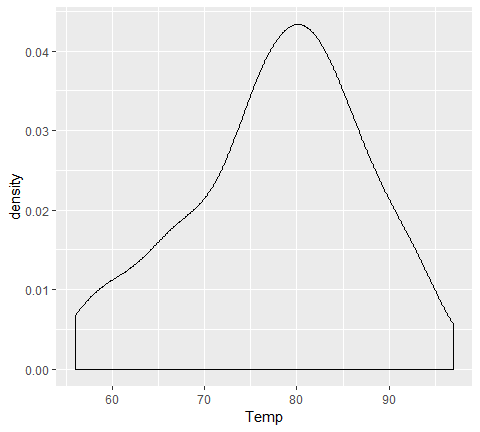
## Visualizing missingness across one variable

We explore conditional missings with ggplot

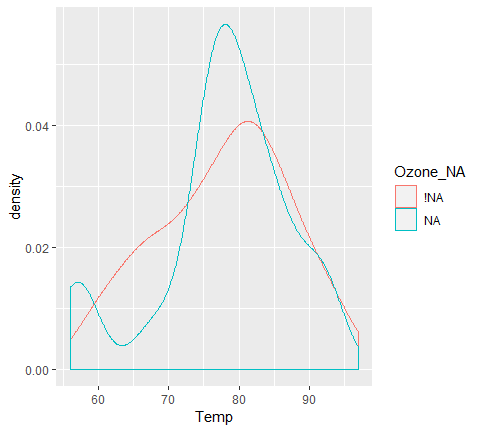
* How to use nabular data to explore how values change according to the other values going missing
* Explore visualizations:
  + densities
  + boxplots
  + dofferent methods of splitting the visualization

When we want to visualize missings using densities

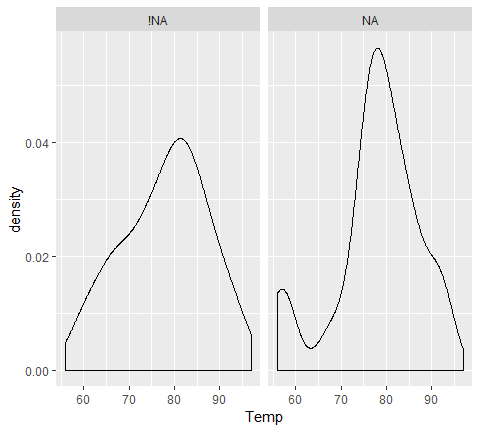
ggplot(airquality,  
 aes(x=Temp))+  
 geom\_density()



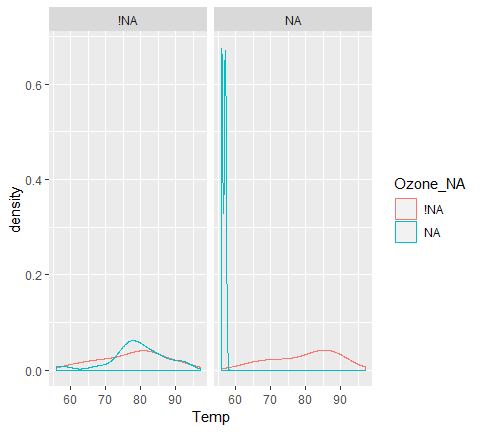
airquality %>%   
 bind\_shadow() %>%   
 ggplot(aes(x=Temp, col=Ozone\_NA))+  
 geom\_density()



airquality %>%   
 bind\_shadow() %>%   
 ggplot(aes(x=Temp))+  
 geom\_density()+  
 facet\_wrap(~Ozone\_NA)



airquality %>%   
 bind\_shadow() %>%   
 ggplot(aes(x = Temp,  
 color = Ozone\_NA))+  
 geom\_density()+  
 facet\_wrap(~Solar.R\_NA)



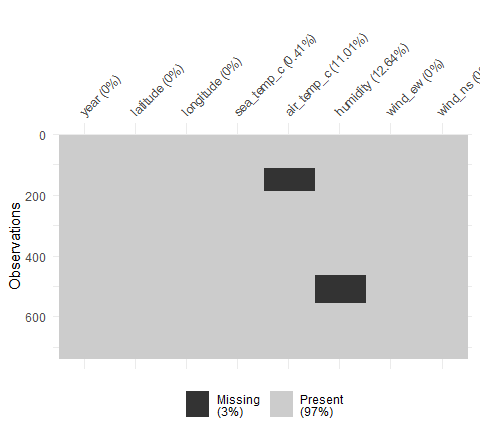
## Nabular data and filling by missingness

Summary statistics are useful to calculate, but as they say, a picture tells you a thousand words.

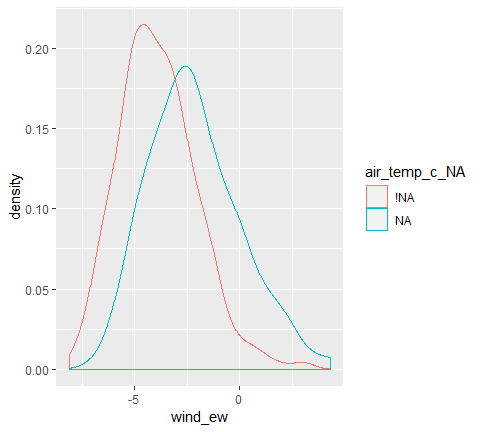
In this exercise, we are going to explore how you can use nabular data to explore the variation in a variable by the missingness of another.

We are going to use the oceanbuoys dataset from naniar.

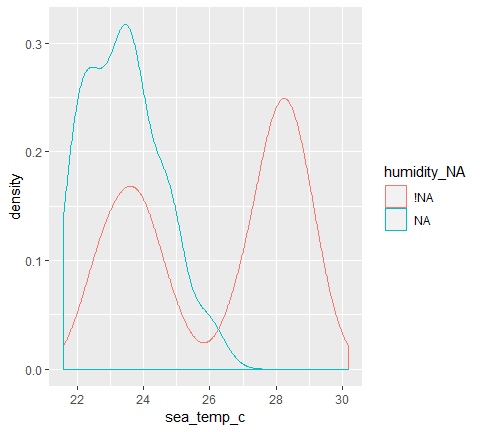
# First explore the missingness structure of `oceanbuoys` using `vis\_miss()`  
vis\_miss(oceanbuoys)



# Explore the distribution of `wind\_ew` for the missingness of `air\_temp\_c\_NA` using `geom\_density()`  
bind\_shadow(oceanbuoys) %>%  
 ggplot(aes(x = wind\_ew,   
 color = air\_temp\_c\_NA)) +   
 geom\_density()



# Explore the distribution of sea temperature for the missingness of humidity (humidity\_NA) using `geom\_density()`  
 bind\_shadow(oceanbuoys) %>%  
 ggplot(aes(x = sea\_temp\_c,  
 color = humidity\_NA)) +   
 geom\_density()

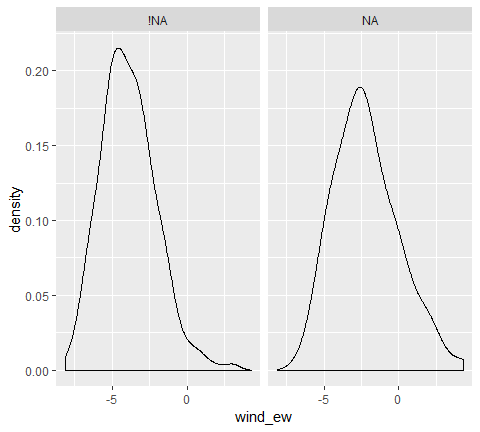


## Nabular data and summarising by missingness

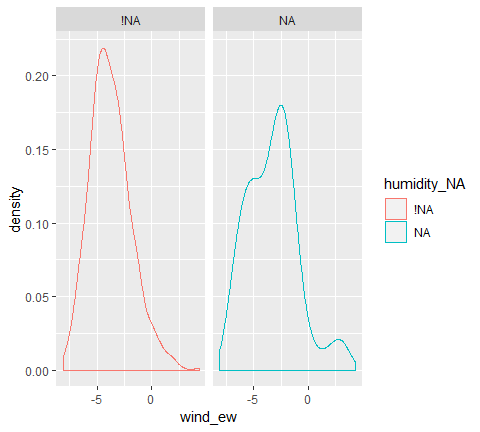
In this exercise, we are going to explore how to use nabular data to explore the variation in a variable by the missingness of another.

We are going to use the oceanbuoys dataset from naniar

# Explore the distribution of wind east west (`wind\_ew`) for the missingness of air temperature using `geom\_density()` and facetting by the missingness of air temperature (`air\_temp\_c\_NA`).  
oceanbuoys %>%  
 bind\_shadow() %>%  
 ggplot(aes(x = wind\_ew)) +   
 geom\_density() +   
 facet\_wrap(~air\_temp\_c\_NA)



# Build upon this visualisation by filling by the missingness of humidity (`humidity\_NA`).  
oceanbuoys %>%  
 bind\_shadow() %>%  
 ggplot(aes(x = wind\_ew,  
 color = humidity\_NA)) +   
 geom\_density() +   
 facet\_wrap(~humidity\_NA)

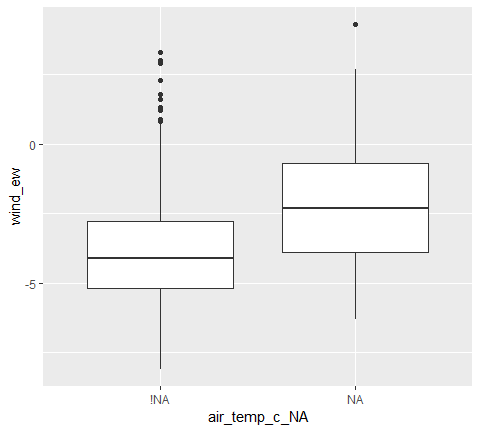


## Explore variation by missingness:boxplots

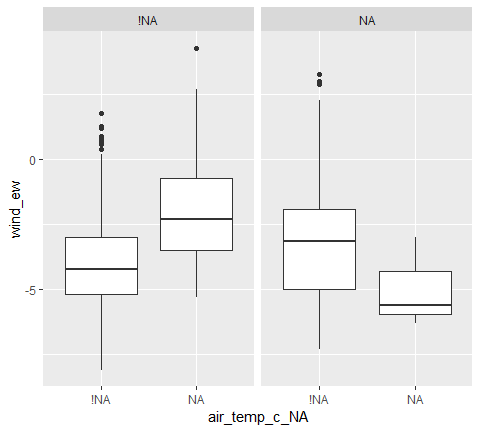
Previous exercises use nabular data along with density plots to explore the variation in a variable by the missingness of another.

We are going to use the oceanbuoys dataset from naniar, using boxplots instead of facets or others to explore different layers of missingness.

# Explore the distribution of wind east west (`wind\_ew`) for the missingness of air temperature using `geom\_boxplot()`  
oceanbuoys %>%  
 bind\_shadow() %>%  
 ggplot(aes(x = air\_temp\_c\_NA,  
 y = wind\_ew)) +   
 geom\_boxplot()



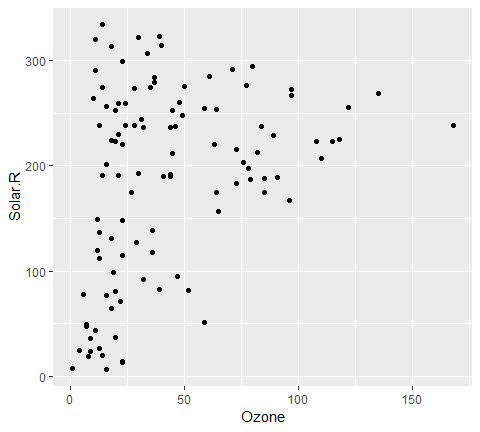
# Build upon this visualisation by facetting by the missingness of humidity (`humidity\_NA`).  
oceanbuoys %>%  
 bind\_shadow() %>%  
 ggplot(aes(x = air\_temp\_c\_NA,  
 y = wind\_ew)) +   
 geom\_boxplot() +   
 facet\_wrap(~humidity\_NA)



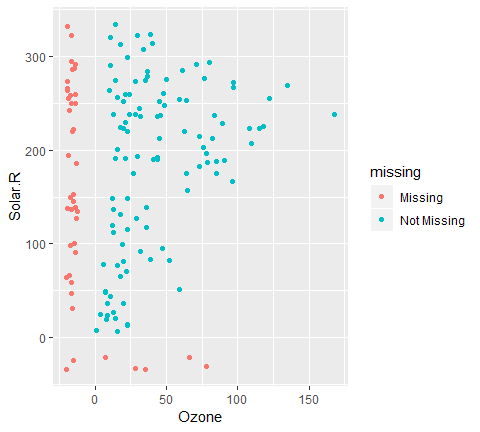
## Visualizing missingenss across two variables

Missing values are typically ignored in the scatterplot. geom\_miss\_point transforms and impute the missing values in the dataset. This is utilized with facet syntax as well.

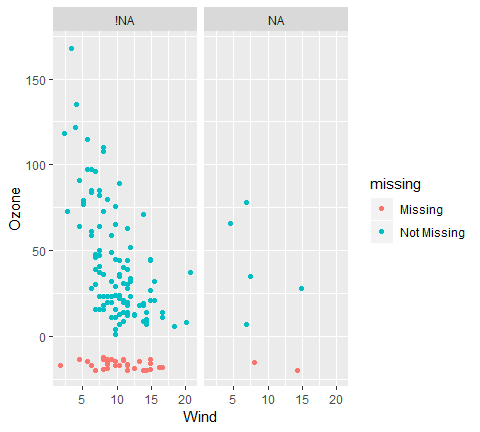
ggplot(airquality,  
 aes(x=Ozone, y = Solar.R))+  
 geom\_point()



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



airquality %>%   
 bind\_shadow() %>%   
 ggplot(aes(x = Wind,  
 y = Ozone))+  
 geom\_miss\_point()+  
 facet\_wrap(~Solar.R\_NA)



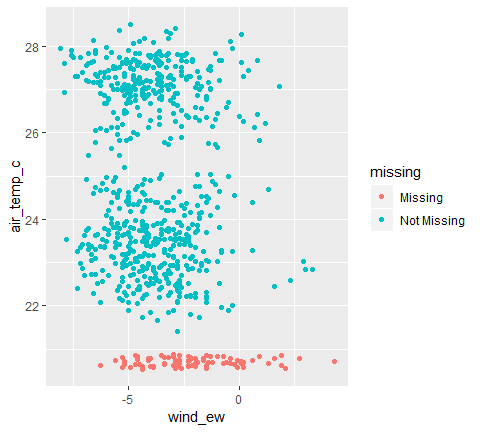
## Exploring missing data with scatterplots

Missing values in a scatterplot in ggplot2 are removed by default, with a warning.

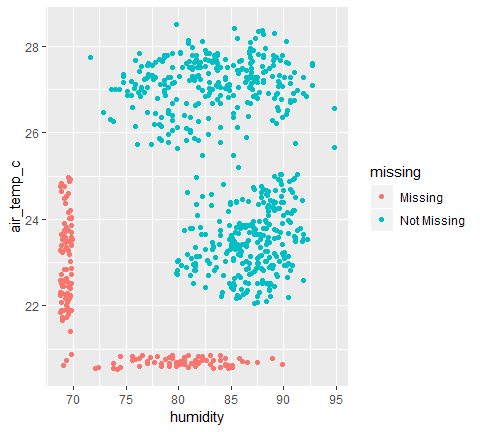
We can display missing values in a scatterplot, using geom\_miss\_point() - a special ggplot2 geom that shifts the missing values into the plot, displaying them 10% below the minimum of the variable.

Let’s practice using this visualisation with the oceanbuoys dataset.

# Explore the missingness in wind and air temperature, and display the missingness using `geom\_miss\_point()`  
ggplot(oceanbuoys,  
 aes(x = wind\_ew,  
 y = air\_temp\_c)) +   
 geom\_miss\_point()



# Explore the missingness in humidity and air temperature, and display the missingness using `geom\_miss\_point()`  
ggplot(oceanbuoys,  
 aes(x = humidity,  
 y = air\_temp\_c)) +   
 geom\_miss\_point()

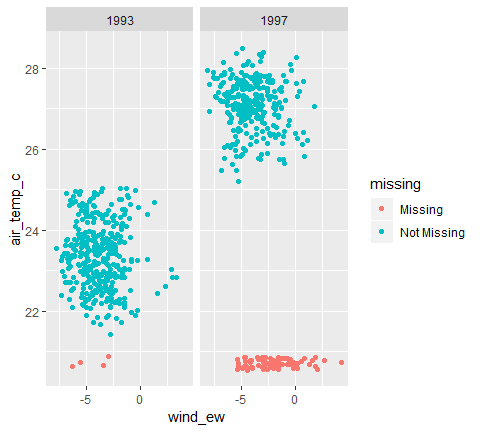


## Using facets to explore missingness

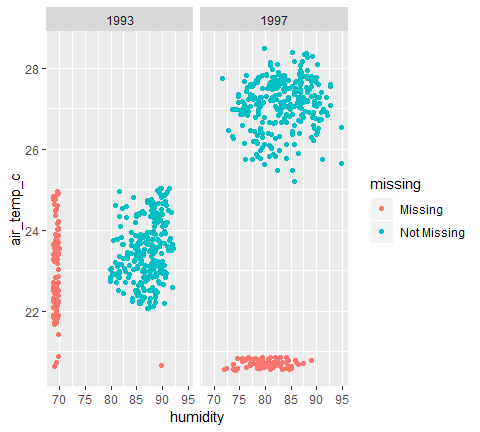
Because geom\_miss\_point() is a ggplot geom, you can use it with ggplot2 features like facetting.

This means we can rapidly explore the missingness and stay within the familar bounds of ggplot2.

# Explore the missingness in wind and air temperature, and display the missingness using `geom\_miss\_point()`. Facet by year to explore this further.  
ggplot(oceanbuoys,  
 aes(x = wind\_ew,  
 y = air\_temp\_c)) +   
 geom\_miss\_point() +   
 facet\_wrap(~year)



# Explore the missingness in humidity and air temperature, and display the missingness using `geom\_miss\_point()` Facet by year to explore this further.  
ggplot(oceanbuoys,  
 aes(x = humidity,  
 y = air\_temp\_c)) +   
 geom\_miss\_point() +   
 facet\_wrap(~year)



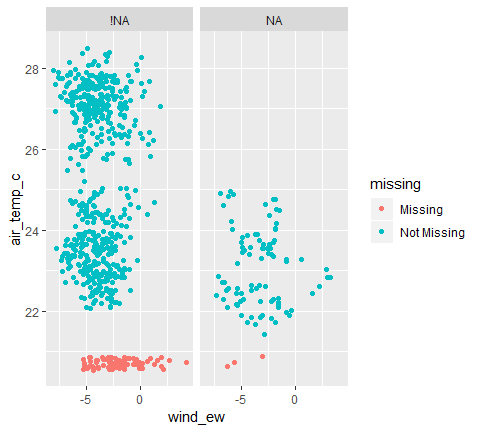
## Faceting to explore missingness (multiple plots)

Another useful technique with geommisspoint() is to explore the missingness by creating multiple plots.

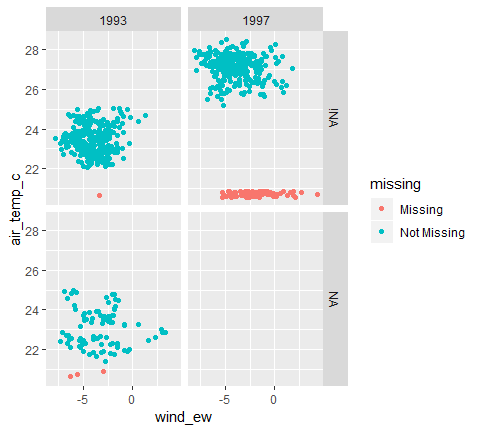
Just as we have done in the previous exercises, we can use the nabular data to help us create additional facetted plots.

We can even create multiple facetted plots according to values in the data, such as year, and features of the data, such as missingness.

# Use geom\_miss\_point() and facet\_wrap to explore how the missingness in wind\_ew and air\_temp\_c is different for missingness of humidity  
bind\_shadow(oceanbuoys) %>%  
 ggplot(aes(x = wind\_ew,  
 y = air\_temp\_c)) +   
 geom\_miss\_point() +   
 facet\_wrap(~humidity\_NA)



# Use geom\_miss\_point() and facet\_grid to explore how the missingness in wind\_ew and air\_temp\_c is different for missingness of humidity AND by year - by using `facet\_grid(humidity\_NA ~ year)`  
bind\_shadow(oceanbuoys) %>%  
 ggplot(aes(x = wind\_ew,  
 y = air\_temp\_c)) +   
 geom\_miss\_point() +   
 facet\_grid(humidity\_NA~year)



# Connecting the dots (Imputation)

## Filling in the blanks

Exloring the missing dat help us to understand the dataset. - **Using imputation to understand data structure** - **Visualizing + Exploring imputed values**

1. Imputing data to explore missingness
2. Track missing values
3. Visualize imputed values against data

impute\_below(c(5,6,7,NA,9,10))  
## [1] 5.00000 6.00000 7.00000 4.40271 9.00000 10.00000

* impute\_below\_if():

impute\_below\_if(data, is.numeric)

* impute\_below\_at():

impute\_below\_at(data, vars(var1, var2))

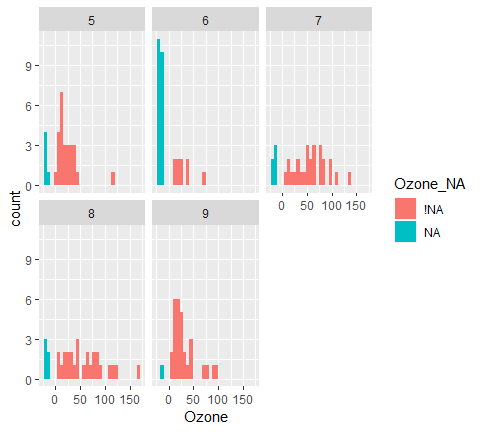
* impute\_below\_all

impute\_below\_all(data)

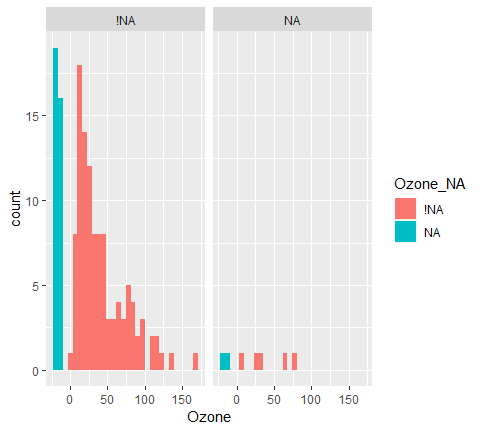
df <- tibble(  
 var1 = c(5,6,7,NA,9,10)  
)  
  
df %>%   
 impute\_below\_all()  
## # A tibble: 6 x 1  
## var1  
## <dbl>  
## 1 5   
## 2 6   
## 3 7   
## 4 4.40  
## 5 9   
## 6 10  
  
df %>%   
 bind\_shadow()  
## # A tibble: 6 x 2  
## var1 var1\_NA  
## <dbl> <fct>   
## 1 5 !NA   
## 2 6 !NA   
## 3 7 !NA   
## 4 NA NA   
## 5 9 !NA   
## 6 10 !NA  
  
df %>%   
 bind\_shadow() %>%   
 impute\_below\_all()  
## # A tibble: 6 x 2  
## var1 var1\_NA  
## <dbl> <fct>   
## 1 5 !NA   
## 2 6 !NA   
## 3 7 !NA   
## 4 4.40 NA   
## 5 9 !NA   
## 6 10 !NA

We can explore a number of missing for single variable.

aq\_imp <- airquality %>%   
 bind\_shadow() %>%   
 impute\_below\_all()  
  
ggplot(aq\_imp,  
 aes( x= Ozone,  
 fill = Ozone\_NA))+  
 geom\_histogram()+  
 facet\_wrap(~Month)

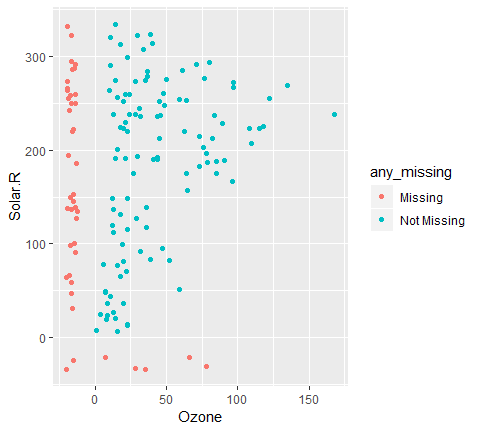


ggplot(aq\_imp,  
 aes( x= Ozone,  
 fill = Ozone\_NA))+  
 geom\_histogram()+  
 facet\_wrap(~Solar.R\_NA)



To visualize Missing values for two variables, the add\_label\_missings will do this for us.

aq\_imp <- airquality %>%   
 bind\_shadow() %>%   
 add\_label\_missings() %>%   
 impute\_below\_all()  
  
ggplot(aq\_imp,  
 aes(x = Ozone,  
 y = Solar.R,  
 color = any\_missing))+  
 geom\_point()



## Impute data below range with nabular data

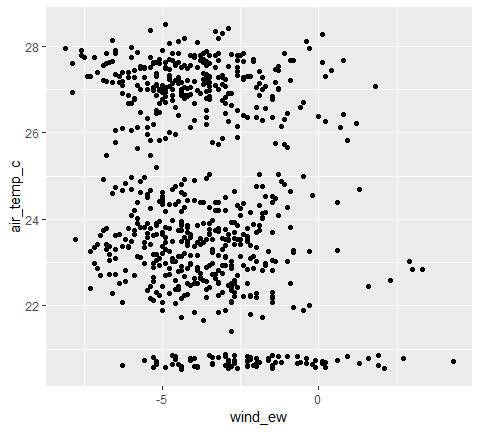
We want to keep track of values we imputed. If we don’t, it is very difficult to assess how good the imputed values are.

We are going to practice imputing data and recreate visualizations in the previous set of exercises by imputing values below the range of the data.

This is a very useful way to help further explore missingness, and also provides the framework for imputing missing values.

First, we are going to impute the data below the range using impute\_below\_all(), and then visualize the data. We notice that although we can see where the missing values are in this instance, we need some way to track them. The track missing data programming pattern can help with this.

# Impute the oceanbuoys data below the range using `impute\_below`.  
ocean\_imp <- impute\_below\_all(oceanbuoys)  
  
# Visualise the new missing values  
ggplot(ocean\_imp,   
 aes(x = wind\_ew, y = air\_temp\_c)) +   
 geom\_point()



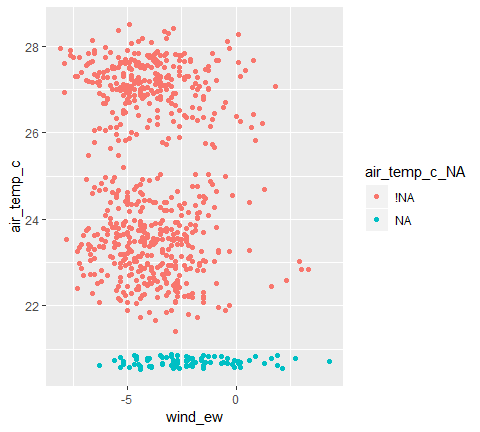
# Impute and track data with `bind\_shadow`, `impute\_below\_all`, and `add\_label\_shadow`  
ocean\_imp\_track <- bind\_shadow(oceanbuoys) %>%   
 impute\_below\_all() %>%   
 add\_label\_shadow()  
  
# Look at the imputed values  
ocean\_imp\_track  
## # A tibble: 736 x 17  
## year latitude longitude sea\_temp\_c air\_temp\_c humidity wind\_ew wind\_ns  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1997 0 -110 27.6 27.1 79.6 -6.40 5.40  
## 2 1997 0 -110 27.5 27.0 75.8 -5.30 5.30  
## 3 1997 0 -110 27.6 27 76.5 -5.10 4.5   
## 4 1997 0 -110 27.6 26.9 76.2 -4.90 2.5   
## 5 1997 0 -110 27.6 26.8 76.4 -3.5 4.10  
## 6 1997 0 -110 27.8 26.9 76.7 -4.40 1.60  
## 7 1997 0 -110 28.0 27.0 76.5 -2 3.5   
## 8 1997 0 -110 28.0 27.1 78.3 -3.70 4.5   
## 9 1997 0 -110 28.0 27.2 78.6 -4.20 5   
## 10 1997 0 -110 28.0 27.2 76.9 -3.60 3.5   
## # ... with 726 more rows, and 9 more variables: year\_NA <fct>,  
## # latitude\_NA <fct>, longitude\_NA <fct>, sea\_temp\_c\_NA <fct>,  
## # air\_temp\_c\_NA <fct>, humidity\_NA <fct>, wind\_ew\_NA <fct>,  
## # wind\_ns\_NA <fct>, any\_missing <chr>  
  
bind\_shadow(oceanbuoys) %>%   
 impute\_below\_all() %>%   
 add\_label\_shadow()  
## # A tibble: 736 x 17  
## year latitude longitude sea\_temp\_c air\_temp\_c humidity wind\_ew wind\_ns  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 1997 0 -110 27.6 27.1 79.6 -6.40 5.40  
## 2 1997 0 -110 27.5 27.0 75.8 -5.30 5.30  
## 3 1997 0 -110 27.6 27 76.5 -5.10 4.5   
## 4 1997 0 -110 27.6 26.9 76.2 -4.90 2.5   
## 5 1997 0 -110 27.6 26.8 76.4 -3.5 4.10  
## 6 1997 0 -110 27.8 26.9 76.7 -4.40 1.60  
## 7 1997 0 -110 28.0 27.0 76.5 -2 3.5   
## 8 1997 0 -110 28.0 27.1 78.3 -3.70 4.5   
## 9 1997 0 -110 28.0 27.2 78.6 -4.20 5   
## 10 1997 0 -110 28.0 27.2 76.9 -3.60 3.5   
## # ... with 726 more rows, and 9 more variables: year\_NA <fct>,  
## # latitude\_NA <fct>, longitude\_NA <fct>, sea\_temp\_c\_NA <fct>,  
## # air\_temp\_c\_NA <fct>, humidity\_NA <fct>, wind\_ew\_NA <fct>,  
## # wind\_ns\_NA <fct>, any\_missing <chr>

## Visualize imputed values in a scatterplot

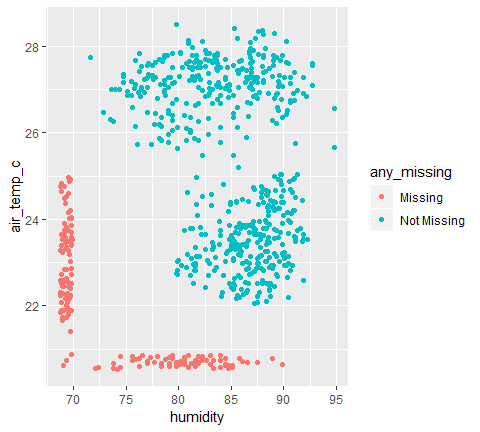
Now, let’s recreate one of the previous plots we saw in chapter three that used geom\_miss\_point().

To do this, we need to impute the data below the range of the data. This is a special kind of imputation to explore the data. This imputation will illustrate what we need to practice: how to track missing values. To impute the data below the range of the data, we use the function impute\_below\_all().

# Impute and track the missing values  
ocean\_imp\_track <- bind\_shadow(oceanbuoys) %>%   
 impute\_below\_all() %>%   
 add\_label\_shadow()  
  
# Visualise the missingness in wind and air temperature, coloring missing air temp values with air\_temp\_c\_NA  
ggplot(ocean\_imp\_track,   
 aes(x = wind\_ew, y = air\_temp\_c, color = air\_temp\_c\_NA)) +   
 geom\_point()



# Visualise humidity and air temp, coloring any missing cases using the variable any\_missing  
ggplot(ocean\_imp\_track,   
 aes(x = humidity, y = air\_temp\_c, color = any\_missing)) +   
 geom\_point()

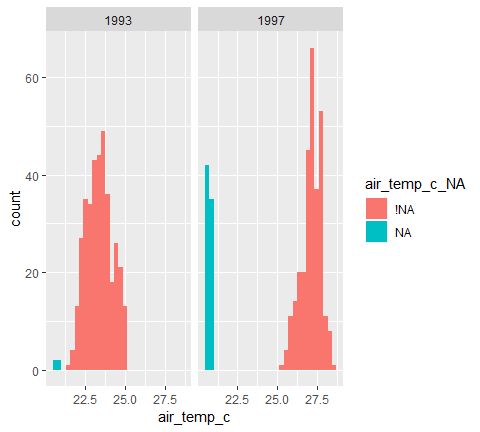


## Create histogram of imputed data

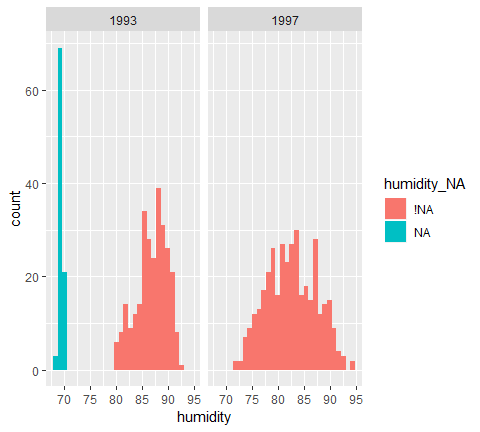
Now that we can recreate the first visualization of geom\_miss\_point(), let’s explore how we can apply this to other exploratory tasks.

One useful task is to evaluate the number of missings in a given variable using a histogram. We can do this using the ocean\_imp\_track dataset we created in the last exercise, which is loaded into this session.

# Explore the values of air\_temp\_c, visualising the amount of missings with `air\_temp\_c\_NA`.  
p <- ggplot(ocean\_imp\_track, aes(x = air\_temp\_c, fill = air\_temp\_c\_NA)) + geom\_histogram()  
  
# Expore the missings in humidity using humidity\_NA  
p2 <- ggplot(ocean\_imp\_track, aes(x = humidity, fill = humidity\_NA)) + geom\_histogram()  
  
# Explore the missings in air\_temp\_c according to year, using `facet\_wrap(~year)`.  
p + facet\_wrap(~year)



# Explore the missings in humidity according to year, using `facet\_wrap(~year)`.  
p2 + facet\_wrap(~year)



## What makes a good imputation

* Understand good/bad imputations
* Evaluate missing values(e.g., mean, scale, spread)
* Using visualization
  + boxplots
  + scatterplots
  + histogram
  + many variables

Bad example is mean imputation. To examine ad imputation, we have

* impute\_mean(data$variable)
* impute\_mean\_if(data, is.numeric
* impute\_mean\_at(data, vars(var1, var2))
* impute\_mean\_all(data

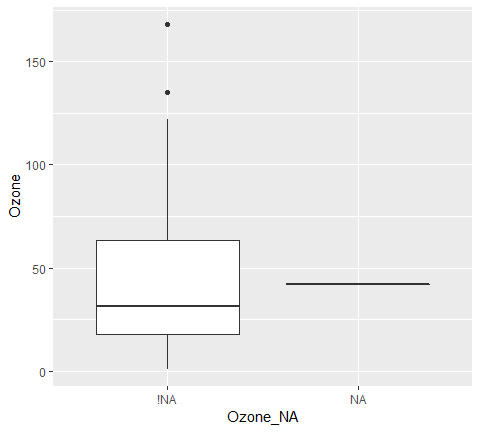
Similar to impute below, we can work on vectors, or some conditions.

aq\_impute\_mean <- airquality %>%   
 bind\_shadow(only\_miss = T) %>%   
 impute\_mean\_all() %>%   
 add\_label\_shadow()  
  
aq\_impute\_mean  
## # A tibble: 153 x 9  
## Ozone Solar.R Wind Temp Month Day Ozone\_NA Solar.R\_NA any\_missing  
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <fct> <fct> <chr>   
## 1 41 190 7.4 67 5 1 !NA !NA Not Missing  
## 2 36 118 8 72 5 2 !NA !NA Not Missing  
## 3 12 149 12.6 74 5 3 !NA !NA Not Missing  
## 4 18 313 11.5 62 5 4 !NA !NA Not Missing  
## 5 42.1 186. 14.3 56 5 5 NA NA Missing   
## 6 28 186. 14.9 66 5 6 !NA NA Missing   
## 7 23 299 8.6 65 5 7 !NA !NA Not Missing  
## 8 19 99 13.8 59 5 8 !NA !NA Not Missing  
## 9 8 19 20.1 61 5 9 !NA !NA Not Missing  
## 10 42.1 194 8.6 69 5 10 NA !NA Missing   
## # ... with 143 more rows

When evaluating imputation, explore changes/similarities in

* the mean/median(boxplot)
* the spread
* the scale

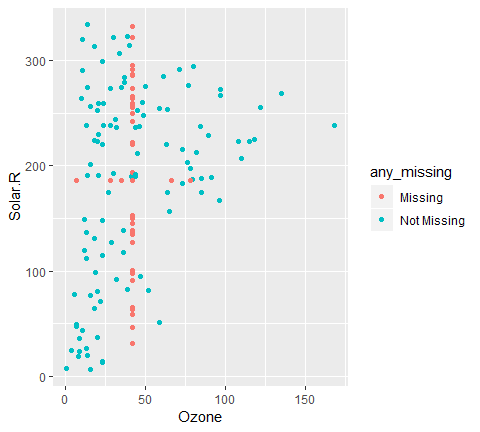
ggplot(aq\_impute\_mean,  
 aes(x = Ozone\_NA,  
 y = Ozone))+  
 geom\_boxplot()



The spread imputation can be explored by scatter plot. When evaluating imputation,s explore changes/similarities in

* **the spread( scatterplot)**

ggplot(aq\_impute\_mean,  
 aes( x = Ozone,  
 y = Solar.R,  
 col = any\_missing))+  
 geom\_point()

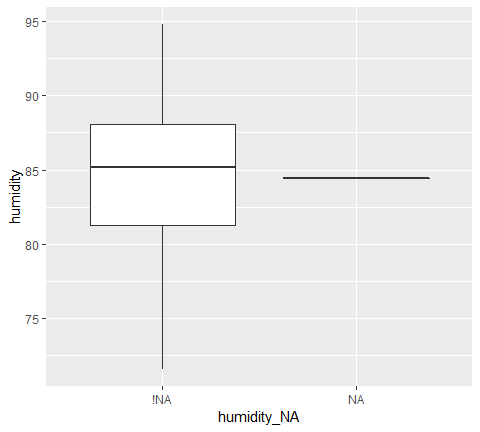


## Evaluating bad imputations

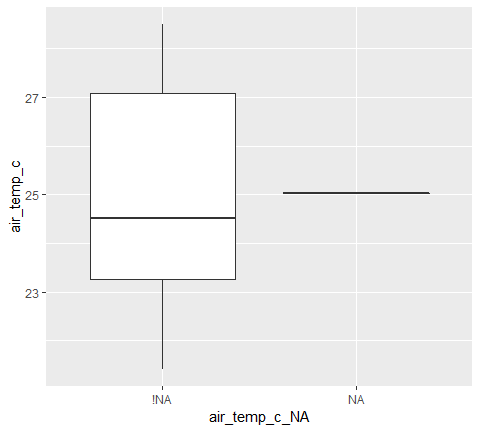
In order to evaluate imputations, it helps to know what something bad looks like. To explore this, let’s look at a typically bad imputation method: imputing using the mean value.

In this exercise we are going to explore how the mean imputation method works using a boxplot, using the oceanbuoys dataset.

# Impute the mean value and track the imputations   
ocean\_imp\_mean <- bind\_shadow(oceanbuoys) %>%   
 impute\_mean\_all() %>%   
 add\_label\_shadow()  
  
# Explore the mean values in humidity in the imputed dataset  
ggplot(ocean\_imp\_mean,   
 aes(x = humidity\_NA, y = humidity)) +   
 geom\_boxplot()



# Explore the values in air temperature in the imputed dataset  
ggplot(ocean\_imp\_mean,   
 aes(x = air\_temp\_c\_NA, y = air\_temp\_c)) +   
 geom\_boxplot()

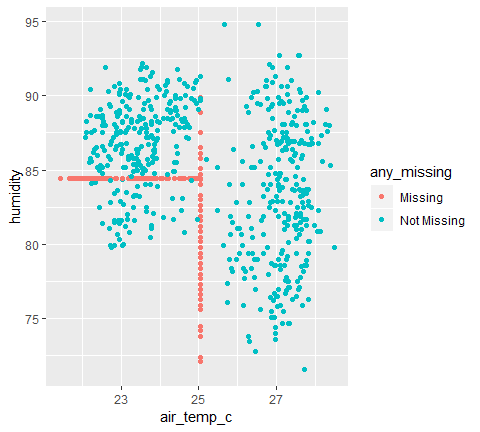


## Evaluating imputations: The scale

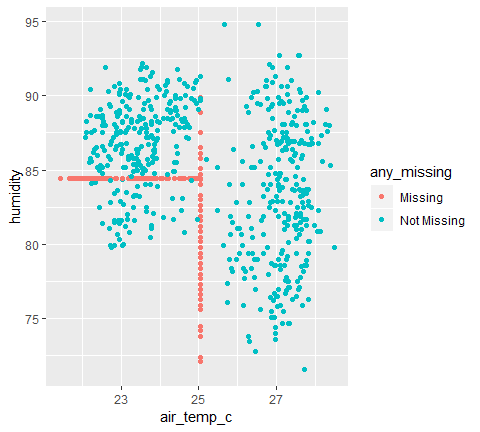
While the mean imputation might not look so bad when we compare it using a boxplot, it is important to get a sense of the variation in the data. This is why it is important to explore how the scale and spread of imputed values changes compared to the data.

One way to evaluate the appropriateness of the scale of the imputations is to use a scatterplot to explore whether or not the values are appropriate.

# Explore imputations in air temperature and humidity, coloring by the variable, any\_missing  
ggplot(ocean\_imp\_mean,   
 aes(x = air\_temp\_c, y = humidity, color = any\_missing)) +   
 geom\_point()



# Explore imputations in air temperature and humidity, coloring by the variable, any\_missing, and faceting by year  
ggplot(ocean\_imp\_mean,   
 aes(x = air\_temp\_c, y = humidity, color = any\_missing)) +   
 geom\_point()

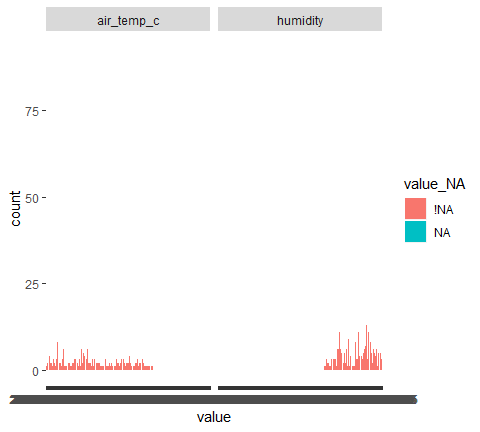


facet\_wrap(~year)  
## <ggproto object: Class FacetWrap, Facet, gg>  
## compute\_layout: function  
## draw\_back: function  
## draw\_front: function  
## draw\_labels: function  
## draw\_panels: function  
## finish\_data: function  
## init\_scales: function  
## map\_data: function  
## params: list  
## setup\_data: function  
## setup\_params: function  
## shrink: TRUE  
## train\_scales: function  
## vars: function  
## super: <ggproto object: Class FacetWrap, Facet, gg>

## Evaluating imputations: Across many variables

So far, we have covered ways to look at individual variables or pairs of variables and their imputed values. However, sometimes you want to look at imputations for many variables. To do this, you need to perform some data munging and re-arranging. This lesson covers how to perform this data wrangling, which can get a little bit hairy when considering its usage in nabular data. The function, shadow\_long() gets the data into the right shape for these kinds of visualizations.

# Gather the imputed data   
ocean\_imp\_mean\_gather <- shadow\_long(ocean\_imp\_mean,  
 humidity,  
 air\_temp\_c)  
# Inspect the data  
ocean\_imp\_mean\_gather  
## # A tibble: 1,472 x 4  
## variable value variable\_NA value\_NA  
## <chr> <chr> <chr> <chr>   
## 1 air\_temp\_c 27.14999962 air\_temp\_c\_NA !NA   
## 2 air\_temp\_c 27.02000046 air\_temp\_c\_NA !NA   
## 3 air\_temp\_c 27 air\_temp\_c\_NA !NA   
## 4 air\_temp\_c 26.93000031 air\_temp\_c\_NA !NA   
## 5 air\_temp\_c 26.84000015 air\_temp\_c\_NA !NA   
## 6 air\_temp\_c 26.94000053 air\_temp\_c\_NA !NA   
## 7 air\_temp\_c 27.04000092 air\_temp\_c\_NA !NA   
## 8 air\_temp\_c 27.11000061 air\_temp\_c\_NA !NA   
## 9 air\_temp\_c 27.20999908 air\_temp\_c\_NA !NA   
## 10 air\_temp\_c 27.25 air\_temp\_c\_NA !NA   
## # ... with 1,462 more rows  
  
# Explore the imputations in a histogram   
ggplot(ocean\_imp\_mean\_gather,   
 aes(x = value, fill = value\_NA)) +   
 geom\_histogram(stat="count") +   
 facet\_wrap(~variable)



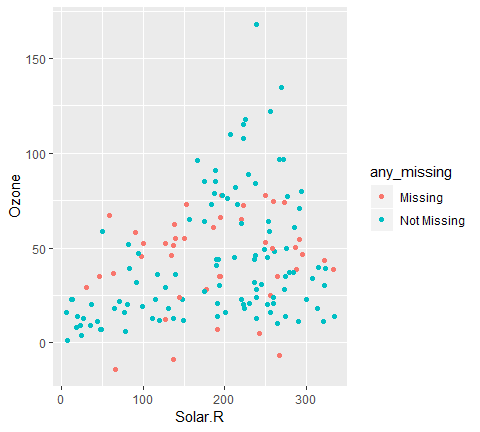
## Performing imputations

* simputation package for imputation.
* We will focus on using linear model to impute values with impute\_lm
* Assess new imputation
* Build many imputation models
* Compare imputations across different models and variables.

df <- tibble::tribble(  
 ~y, ~var1, ~var2,  
 2.67, 2.43, 3.27,  
 3.87, 3.55, 1.45,  
 NA, 2.90, 1.49,  
 5.21, 2.72, 1.84,  
 NA, 4.29, 1.15  
)  
  
library(simputation)  
df %>%   
 bind\_shadow(only\_miss = TRUE) %>%   
 add\_label\_shadow() %>%   
 impute\_lm(y~var1+var2)  
## # A tibble: 5 x 5  
## y var1 var2 y\_NA any\_missing  
## \* <dbl> <dbl> <dbl> <fct> <chr>   
## 1 2.67 2.43 3.27 !NA Not Missing  
## 2 3.87 3.55 1.45 !NA Not Missing  
## 3 5.54 2.9 1.49 NA Missing   
## 4 5.21 2.72 1.84 !NA Not Missing  
## 5 2.56 4.29 1.15 NA Missing

We will use impute\_lm for airquality data.

aq\_imp\_lm <- airquality %>%   
 bind\_shadow() %>%   
 add\_label\_shadow() %>%   
 impute\_lm(Solar.R~Wind+Temp+Month) %>%   
 impute\_lm(Ozone~Wind+Temp+Month)  
  
aq\_imp\_lm  
## # A tibble: 153 x 13  
## Ozone Solar.R Wind Temp Month Day Ozone\_NA Solar.R\_NA Wind\_NA  
## \* <dbl> <dbl> <dbl> <int> <int> <int> <fct> <fct> <fct>   
## 1 41 190 7.4 67 5 1 !NA !NA !NA   
## 2 36 118 8 72 5 2 !NA !NA !NA   
## 3 12 149 12.6 74 5 3 !NA !NA !NA   
## 4 18 313 11.5 62 5 4 !NA !NA !NA   
## 5 -9.04 138. 14.3 56 5 5 NA NA !NA   
## 6 28 178. 14.9 66 5 6 !NA NA !NA   
## 7 23 299 8.6 65 5 7 !NA !NA !NA   
## 8 19 99 13.8 59 5 8 !NA !NA !NA   
## 9 8 19 20.1 61 5 9 !NA !NA !NA   
## 10 35.2 194 8.6 69 5 10 NA !NA !NA   
## # ... with 143 more rows, and 4 more variables: Temp\_NA <fct>,  
## # Month\_NA <fct>, Day\_NA <fct>, any\_missing <chr>  
  
aq\_imp\_lm %>%   
 ggplot(  
 aes(x = Solar.R,  
 y = Ozone,  
 col = any\_missing))+  
 geom\_point()



The important is to insert bind\_shadow() and add\_label\_shadow() to detect missingness.

Another useful feature is conducts variants of lm application.

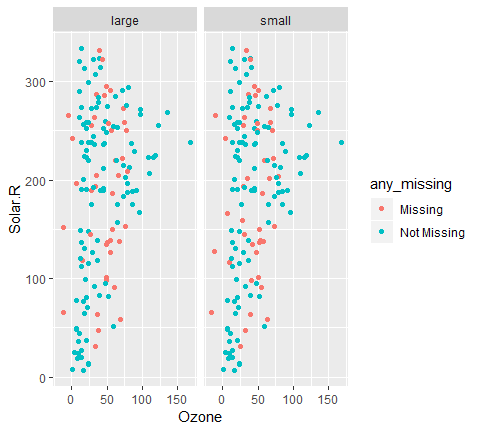
aq\_imp\_small <- airquality %>%  
 bind\_shadow() %>%  
 impute\_lm(Ozone ~ Wind + Temp) %>%  
 impute\_lm(Solar.R ~ Wind + Temp) %>%  
 add\_label\_shadow()  
  
aq\_imp\_large <- airquality %>%  
 bind\_shadow() %>%  
 impute\_lm(Ozone ~ Wind + Temp + Month + Day) %>%  
 impute\_lm(Solar.R ~ Wind + Temp + Month + Day) %>%  
 add\_label\_shadow()

To compare models, we bind the above two variants.

bound\_models <-   
 bind\_rows(small = aq\_imp\_small,  
 large = aq\_imp\_large,  
 .id = "imp\_model")  
bound\_models  
## # A tibble: 306 x 14  
## imp\_model Ozone Solar.R Wind Temp Month Day Ozone\_NA Solar.R\_NA  
## <chr> <dbl> <dbl> <dbl> <int> <int> <int> <fct> <fct>   
## 1 small 41 190 7.4 67 5 1 !NA !NA   
## 2 small 36 118 8 72 5 2 !NA !NA   
## 3 small 12 149 12.6 74 5 3 !NA !NA   
## 4 small 18 313 11.5 62 5 4 !NA !NA   
## 5 small -11.7 127. 14.3 56 5 5 NA NA   
## 6 small 28 160. 14.9 66 5 6 !NA NA   
## 7 small 23 299 8.6 65 5 7 !NA !NA   
## 8 small 19 99 13.8 59 5 8 !NA !NA   
## 9 small 8 19 20.1 61 5 9 !NA !NA   
## 10 small 29.7 194 8.6 69 5 10 NA !NA   
## # ... with 296 more rows, and 5 more variables: Wind\_NA <fct>,  
## # Temp\_NA <fct>, Month\_NA <fct>, Day\_NA <fct>, any\_missing <chr>

We can then look at the values.

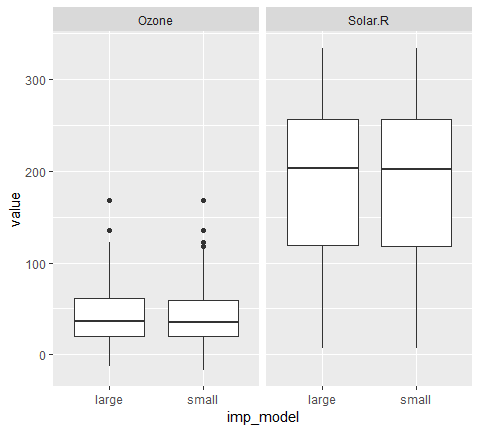
ggplot(bound\_models,  
 aes(x = Ozone,  
 y = Solar.R,  
 col = any\_missing))+  
 geom\_point()+  
 facet\_wrap(~imp\_model)



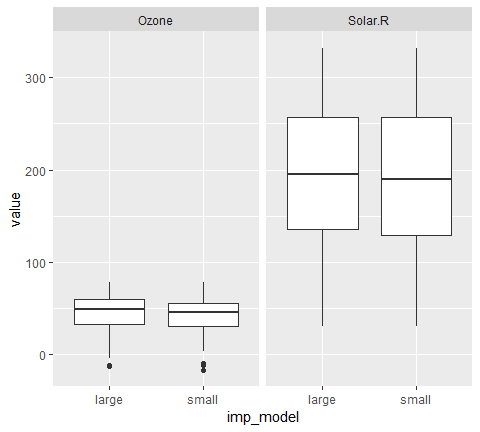
To explore imputations across multiple variables and models, we gather selected variables.

bound\_models\_gather <- bound\_models %>%  
 select(Ozone, Solar.R,  
 any\_missing, imp\_model) %>%  
 gather(key = "variable", value = "value",  
 -any\_missing, -imp\_model)  
  
bound\_models\_gather  
## # A tibble: 612 x 4  
## any\_missing imp\_model variable value  
## <chr> <chr> <chr> <dbl>  
## 1 Not Missing small Ozone 41   
## 2 Not Missing small Ozone 36   
## 3 Not Missing small Ozone 12   
## 4 Not Missing small Ozone 18   
## 5 Missing small Ozone -11.7  
## 6 Missing small Ozone 28   
## 7 Not Missing small Ozone 23   
## 8 Not Missing small Ozone 19   
## 9 Not Missing small Ozone 8   
## 10 Missing small Ozone 29.7  
## # ... with 602 more rows

ggplot(bound\_models\_gather,  
 aes(x = imp\_model,  
 y = value)) +  
 geom\_boxplot() +   
 facet\_wrap(~variable)



bound\_models\_gather %>%  
 filter(any\_missing == "Missing") %>%  
 ggplot(aes(x = imp\_model,  
 y = value)) +  
 geom\_boxplot() +   
 facet\_wrap(~variable)



## Using simputation to impute data

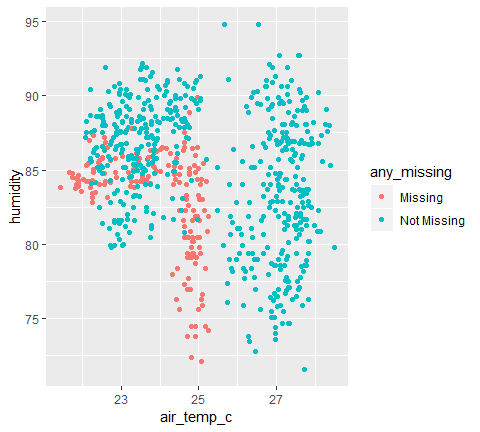
There are many imputation packages in R. We are going to focus on using the simputation package, which provides a simple, powerful interface into performing imputations.

Building a good imputation model is super important, but it is a complex topic - there is as much to building a good imputation model as there is for building a good statistical model. In this course, we are going to focus on how to evaluate imputations.

First, we are going to look at using impute\_lm() function, which imputes values according to a specified linear model.

In this exercise, we are going to apply the previous assessment techniques to data with impute\_lm(), and then build upon this imputation method in subsequent lessons.

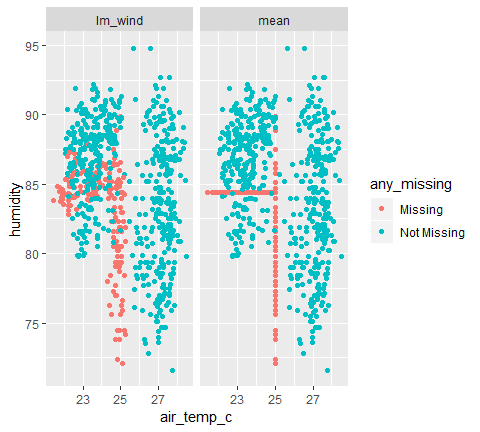
# Impute humidity and air temperature using wind\_ew and wind\_ns, and track missing values  
ocean\_imp\_lm\_wind <- oceanbuoys%>%   
 bind\_shadow() %>%  
 impute\_lm(air\_temp\_c ~ wind\_ew + wind\_ns) %>%   
 impute\_lm(humidity ~ wind\_ew + wind\_ns) %>%  
 add\_label\_shadow()  
   
# Plot the imputed values for air\_temp\_c and humidity, colored by missingness  
ggplot(ocean\_imp\_lm\_wind,   
 aes(x = air\_temp\_c, y = humidity, color = any\_missing)) +   
 geom\_point()



## Evaluating and comparing imputations

When you build up an imputation model, it’s a good idea to compare it to another method. In this lesson, we are going to compare the previously imputed dataset created using impute\_lm() to the mean imputed dataset. Both of these datasets are included in this exercise as ocean\_imp\_lm\_wind and ocean\_imp\_mean respectively.

# Bind the models together   
bound\_models <- bind\_rows(mean = ocean\_imp\_mean,  
 lm\_wind = ocean\_imp\_lm\_wind,  
 .id = "imp\_model")  
  
# Inspect the values of air\_temp and humidity as a scatterplot  
ggplot(bound\_models,   
 aes(x = air\_temp\_c,   
 y = humidity,   
 color = any\_missing)) +  
 geom\_point() +   
 facet\_wrap(~imp\_model)

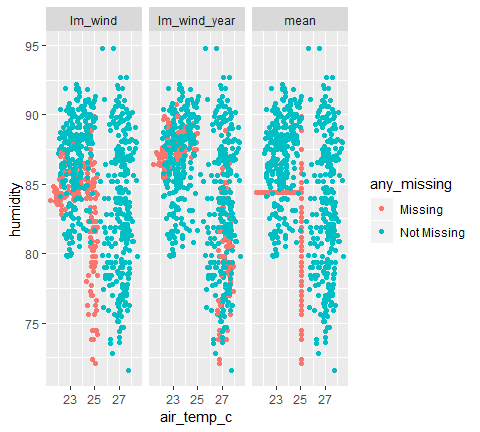


## Evaluating imputations (many models & variables)

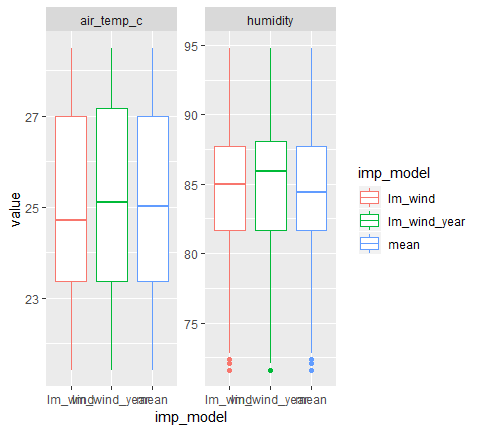
When you build up an imputation model, it’s a good idea to compare it to another method.

In this lesson, we are going to get you to add a final imputation model that contains an extra useful piece of information that helps explain some of the variation in the data. You are then going to compare the values, as previously done in the last lesson.

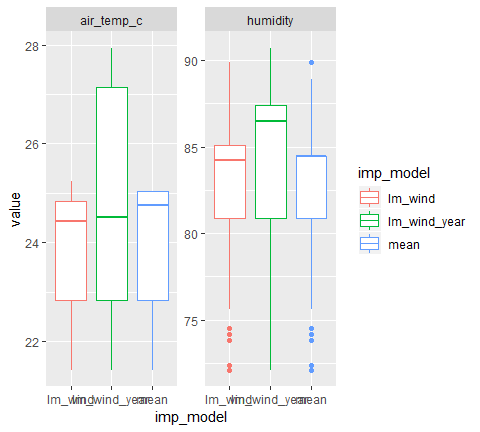
# Build a model adding year to the outcome  
ocean\_imp\_lm\_wind\_year <- bind\_shadow(oceanbuoys) %>%  
 impute\_lm(air\_temp\_c ~ wind\_ew + wind\_ns + year) %>%  
 impute\_lm(humidity ~ wind\_ew + wind\_ns + year) %>%  
 add\_label\_shadow()  
  
# Bind the mean, lm\_wind, and lm\_wind\_year models together  
bound\_models <- bind\_rows(mean = ocean\_imp\_mean,  
 lm\_wind = ocean\_imp\_lm\_wind,  
 lm\_wind\_year = ocean\_imp\_lm\_wind\_year,  
 .id = "imp\_model")  
  
# Explore air\_temp and humidity, coloring by any missings, and faceting by imputation model  
ggplot(bound\_models, aes(x = air\_temp\_c, y = humidity, color = any\_missing)) +   
 geom\_point() + facet\_wrap(~imp\_model)



# Gather the data and inspect the distributions of the values  
bound\_models\_gather <- bound\_models %>%  
 select(air\_temp\_c, humidity, any\_missing, imp\_model) %>%  
 gather(key = "key", value = "value", -any\_missing, -imp\_model)  
  
# Inspect the distribution for each variable, for each model  
ggplot(bound\_models\_gather,   
 aes(x = imp\_model, y = value, color = imp\_model)) +  
 geom\_boxplot() + facet\_wrap(~key, scales = "free\_y")



# Inspect the imputed values  
bound\_models\_gather %>%  
 filter(any\_missing == "Missing") %>%  
 ggplot(aes(x = imp\_model, y = value, color = imp\_model)) +  
 geom\_boxplot() + facet\_wrap(~key, scales = "free\_y")



## Evaluating imputations and models

Our goal is to perform an analysis after imputing data.

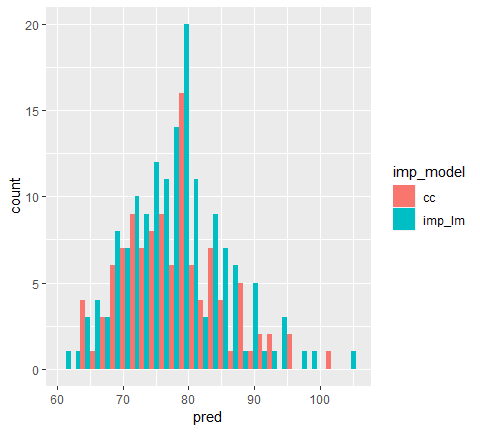
1. Complete case analysis
2. Imputation using the imputed data from the last lesson.

#1. complete case scinario  
aq\_cc <- airquality %>%   
 na.omit() %>%   
 bind\_shadow() %>%   
 add\_label\_shadow()  
  
#2. Imputation using the imputed data from the last lesson  
aq\_imp\_lm <- bind\_shadow(airquality) %>%  
 add\_label\_shadow() %>%  
 impute\_lm(Ozone ~ Temp + Wind + Month + Day) %>%  
 impute\_lm(Solar.R ~ Temp + Wind + Month + Day)  
  
 # 3. Bind the models together  
bound\_models <- bind\_rows(cc = aq\_cc,  
 imp\_lm = aq\_imp\_lm,  
 .id = "imp\_model")

After preparing the models, we fit a linear model separately.

model\_summary <- bound\_models %>%   
 group\_by(imp\_model) %>%  
 nest() %>%  
 mutate(mod = map(data,   
 ~lm(Temp ~ Ozone + Solar.R + Wind + Temp + Day + Month, data = .)),  
 res = map(mod, residuals),  
 pred = map(mod, predict),  
 tidy = map(mod, broom::tidy))  
  
model\_summary  
## # A tibble: 2 x 6  
## imp\_model data mod res pred tidy   
## <chr> <list> <list> <list> <list> <list>   
## 1 cc <tibble [111 x 1~ <S3: lm> <dbl [111~ <dbl [11~ <tibble [6 x ~  
## 2 imp\_lm <tibble [153 x 1~ <S3: lm> <dbl [153~ <dbl [15~ <tibble [6 x ~

model\_summary %>%   
 select(imp\_model,  
 pred) %>%  
 unnest() %>%  
 ggplot(aes(x = pred,  
 fill = imp\_model)) +  
 geom\_histogram(position = "dodge")



## Combining and comparing many imputation models

To evaluate the different imputation methods, we need to put them into a single dataframe. Next, you will compare three different approaches to handling missing data using the dataset, oceanbuoys.

The first method is using only the completed cases and is loaded as ocean\_cc. The second method is imputing values using a linear model with predictions made using wind and is loaded as ocean\_imp\_lm\_wind. You will create the third imputed dataset, ocean\_imp\_lm\_all, using a linear model and impute the variables sea\_temp\_c, air\_temp\_c, and humidity using the variables wind\_ew, wind\_ns, year, latitude, longitude.

You will then bind all of the datasets together (ocean\_cc, ocean\_imp\_lm\_wind, and ocean\_imp\_lm\_all), calling it bound\_models.

# Create an imputed dataset using a linear models  
ocean\_imp\_lm\_all <- bind\_shadow(oceanbuoys) %>%  
 add\_label\_shadow() %>%  
 impute\_lm(sea\_temp\_c ~ wind\_ew + wind\_ns + year + latitude + longitude) %>%  
 impute\_lm(air\_temp\_c ~ wind\_ew + wind\_ns + year + latitude + longitude) %>%  
 impute\_lm(humidity ~ wind\_ew + wind\_ns + year + latitude + longitude)  
  
# Bind the datasets  
bound\_models <- bind\_rows(mean = ocean\_imp\_mean,  
 imp\_lm\_wind = ocean\_imp\_lm\_wind,  
 imp\_lm\_all = ocean\_imp\_lm\_all,  
 .id = "imp\_model")  
# Look at the models  
bound\_models  
## # A tibble: 2,208 x 18  
## imp\_model year latitude longitude sea\_temp\_c air\_temp\_c humidity  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 mean 1997 0 -110 27.6 27.1 79.6  
## 2 mean 1997 0 -110 27.5 27.0 75.8  
## 3 mean 1997 0 -110 27.6 27 76.5  
## 4 mean 1997 0 -110 27.6 26.9 76.2  
## 5 mean 1997 0 -110 27.6 26.8 76.4  
## 6 mean 1997 0 -110 27.8 26.9 76.7  
## 7 mean 1997 0 -110 28.0 27.0 76.5  
## 8 mean 1997 0 -110 28.0 27.1 78.3  
## 9 mean 1997 0 -110 28.0 27.2 78.6  
## 10 mean 1997 0 -110 28.0 27.2 76.9  
## # ... with 2,198 more rows, and 11 more variables: wind\_ew <dbl>,  
## # wind\_ns <dbl>, year\_NA <fct>, latitude\_NA <fct>, longitude\_NA <fct>,  
## # sea\_temp\_c\_NA <fct>, air\_temp\_c\_NA <fct>, humidity\_NA <fct>,  
## # wind\_ew\_NA <fct>, wind\_ns\_NA <fct>, any\_missing <chr>

## Evaluating the different parameters in the model

We are imputing our data for a reason - we want to analyze the data!

In this example, we are interested in predicting sea temperature, so we will build a linear model predicting sea temperature.

We will fit this model to each of the datasets we created and then explore the coefficients in the data.

The objects from the previous lesson (ocean\_cc, ocean\_imp\_lm\_wind, ocean\_imp\_lm\_all, and bound\_models) are loaded into the workspace.

# Create the model summary for each dataset  
model\_summary <- bound\_models %>%   
 group\_by(imp\_model) %>%  
 nest() %>%  
 mutate(mod = map(data, ~lm(sea\_temp\_c ~ air\_temp\_c + humidity + year, data = .)),  
 res = map(mod, residuals),  
 pred = map(mod, predict),  
 tidy = map(mod, broom::tidy))  
  
# Explore the coefficients in the model  
model\_summary %>%   
 select(imp\_model,tidy) %>%  
 unnest()  
## # A tibble: 12 x 6  
## imp\_model term estimate std.error statistic p.value  
## <chr> <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 mean (Intercept) -1589. 56.7 -28.0 6.84e-118  
## 2 mean air\_temp\_c 0.441 0.0285 15.5 5.33e- 47  
## 3 mean humidity 0.0161 0.00675 2.39 1.71e- 2  
## 4 mean year 0.803 0.0286 28.1 3.51e-118  
## 5 imp\_lm\_wind (Intercept) -1742. 56.1 -31.0 1.83e-135  
## 6 imp\_lm\_wind air\_temp\_c 0.365 0.0279 13.1 2.73e- 35  
## 7 imp\_lm\_wind humidity 0.0225 0.00690 3.26 1.17e- 3  
## 8 imp\_lm\_wind year 0.880 0.0283 31.1 6.79e-136  
## 9 imp\_lm\_all (Intercept) -697. 51.8 -13.5 5.04e- 37  
## 10 imp\_lm\_all air\_temp\_c 0.890 0.0255 35.0 2.90e-158  
## 11 imp\_lm\_all humidity 0.0127 0.00463 2.75 6.03e- 3  
## 12 imp\_lm\_all year 0.351 0.0262 13.4 1.12e- 36  
best\_model <- "imp\_lm\_all"