

Missing Data

Data Wrangling in R

Dealing with Missing Data

Missing data types

One of the most important aspects of data cleaning is missing values.

Types of “missing” data:

- **NA** - general missing data
- **NaN** - stands for “**N**ot **a** **N**umber”, happens when you do $0/0$.
- **Inf** and **-Inf** - Infinity, happens when you take a positive number (or negative number) by 0.

Missing Data with Logicals

Logical operations return **NA** for **NA** values. Think about it, the data could be > 2 or not we don't know, so R says there is no **TRUE** or **FALSE**, so that is missing:

```
x <- c(0, NA, 2, 3, 4)
x > 2
```

```
[1] FALSE    NA FALSE  TRUE  TRUE
```

Missing Data Issues

Mathematical operations with **NA** result in **NAs**.

```
y <- c(1,2,3,NA)  
sum(y)
```

```
[1] NA
```

```
mean(y)
```

```
[1] NA
```

Missing Data Issues

Logicals: TRUE is evaluated as 1 and FALSE is evaluated as 0.

```
x <- c(TRUE, TRUE, TRUE, TRUE, FALSE, NA)  
sum(x)
```

```
[1] NA
```

```
sum(x, na.rm = TRUE)
```

```
[1] 4
```

Finding Missing data

- `is.na` - looks for `NAN` and `NA`
- `is.nan` - looks for `NAN`

```
test <- c(0, NA, -1, NaN)  
is.na(test)
```

```
[1] FALSE TRUE FALSE TRUE
```

```
is.nan(test)
```

```
[1] FALSE FALSE FALSE TRUE
```

Useful checking functions

Do we have any **NAs**? (**any** can help)

```
A <- c(1, 2, 3, NA)
B <- c(1, 2, 3, 4)
```

```
any(is.na(A)) # are there any NAs – YES/TRUE
```

```
[1] TRUE
```

```
any(is.na(B)) # are there any NAs– NO/FALSE
```

```
[1] FALSE
```


Finding **NA** values with **count()**

Check the values for your variables, are they what you expect?

`count()` is a great option because it gives you:

1. The unique values
2. The amount of these values

Check if rare values make sense.

Finding **NA** values with `count()`

```
library(readr)
bike <- read_csv("https://sisbid.github.io/Data-Wrangling/labs/Bike_Lanes.csv")
count(bike, subType)
```

```
# A tibble: 4 × 2
  subType      n
  <chr>    <int>
1 STCLN         1
2 STRALY         3
3 STRPRD      1623
4 <NA>           4
```

naniar

Sometimes you need to look at lots of data... the [naniar package](#) is a good option.

```
#install.packages("naniar")  
library(naniar)
```

Air quality data

The `airquality` dataset comes with R about air quality in New York in 1973.

```
?airquality # use this to find out more about the data
```

naniar: pct_complete()

This can tell you if there are missing values in the dataset.

```
pct_complete(airquality)
```

```
[1] 95.20697
```

Or for a particular variable:

```
airquality %>% select(Ozone) %>%  
pct_complete()
```

```
[1] 75.81699
```

`naniar:miss_var_summary()`

To get the percent missing (and counts) for each variable as a table, use this function.

```
miss_var_summary(airquality)
```

```
# A tibble: 6 × 3
  variable n_miss pct_miss
  <chr>      <int>    <num>
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
```

miss_case_summary which rows (IDs) have missing data

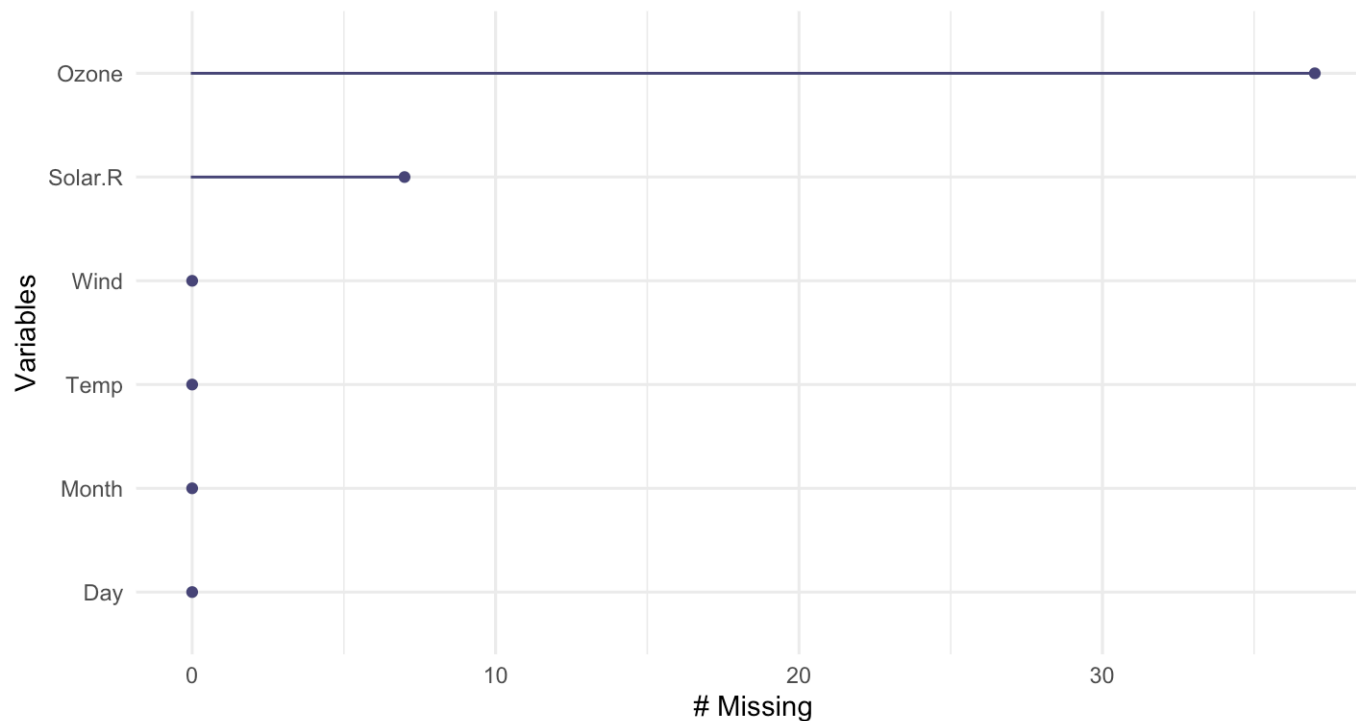
```
miss_case_summary(airquality)
```

```
# 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
```

nan~~i~~ar plots

The `gg_miss_var()` function creates a nice plot about the number of missing values for each variable, (need a data frame).

```
gg_miss_var(airquality)
```



`filter()` and missing data

Be **careful** with missing data using subsetting!

`filter()` removes missing values by default. Because R can't tell for sure if an NA value meets the condition. To keep them need to add `is.na()` conditional.

Think about if this is OK or not - it depends on your data!

filter() and missing data

What if NA values represent values that are so low it is undetectable?

Filter will drop them from the data.

```
airquality %>% filter(Ozone < 5)
```

	Ozone	Solar.R	Wind	Temp	Month	Day
1	1	8	9.7	59	5	21
2	4	25	9.7	61	5	23

filter() and missing data

`is.na()` can help us keep them.

```
airquality %>% filter(Ozone < 5 | is.na(Ozone))
```

	Ozone	Solar.R	Wind	Temp	Month	Day
1	NA	NA	14.3	56	5	5
2	NA	194	8.6	69	5	10
3	1	8	9.7	59	5	21
4	4	25	9.7	61	5	23
5	NA	66	16.6	57	5	25
6	NA	266	14.9	58	5	26
7	NA	NA	8.0	57	5	27
8	NA	286	8.6	78	6	1
9	NA	287	9.7	74	6	2
10	NA	242	16.1	67	6	3
11	NA	186	9.2	84	6	4
12	NA	220	8.6	85	6	5
13	NA	264	14.3	79	6	6
14	NA	273	6.9	87	6	8
15	NA	259	10.9	93	6	11
16	NA	250	9.2	92	6	12
17	NA	332	13.8	80	6	14
18	NA	322	11.5	79	6	15
19	NA	150	6.3	77	6	21
20	NA	59	1.7	76	6	22
21	NA	91	4.6	76	6	23
22	NA	250	6.3	76	6	24
23	NA	135	8.0	75	6	25

To remove rows with NA values for a variable use `drop_na()`

A function from the `tidyr` package. (Need a data frame to start!)

Disclaimer: Don't do this unless you have thought about if dropping NA values makes sense based on knowing what these values mean in your data.

airquality

	Ozone	Solar.R	Wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8.0	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	NA	NA	14.3	56	5	5
6	28	NA	14.9	66	5	6
7	23	299	8.6	65	5	7
8	19	99	13.8	59	5	8
9	8	19	20.1	61	5	9
10	NA	194	8.6	69	5	10
11	7	NA	6.9	74	5	11
12	16	256	9.7	69	5	12
13	11	290	9.2	66	5	13
14	14	274	10.9	68	5	14
15	18	65	13.2	58	5	15
16	14	334	11.5	64	5	16
17	34	307	12.0	66	5	17
18	6	78	18.4	57	5	18
19	30	322	11.5	68	5	19
20	11	44	9.7	62	5	20

To remove rows with NA values for a variable use `drop_na()`

```
airquality %>% drop_na(Ozone)
```

	Ozone	Solar.R	Wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8.0	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	28	NA	14.9	66	5	6
6	23	299	8.6	65	5	7
7	19	99	13.8	59	5	8
8	8	19	20.1	61	5	9
9	7	NA	6.9	74	5	11
10	16	256	9.7	69	5	12
11	11	290	9.2	66	5	13
12	14	274	10.9	68	5	14
13	18	65	13.2	58	5	15
14	14	334	11.5	64	5	16
15	34	307	12.0	66	5	17
16	6	78	18.4	57	5	18
17	30	322	11.5	68	5	19
18	11	44	9.7	62	5	20
19	1	8	9.7	59	5	21
20	11	320	16.6	73	5	22
21	4	25	9.7	61	5	23
22	32	92	12.0	61	5	24
23	23	13	12.0	67	5	28
24	45	252	14.9	81	5	29
25	115	223	5.7	79	5	30
26	27	270	7.4	76	5	31

Drop all NAs with `drop_na()`

Drops rows with **any** missing data in **any** column.

```
airquality %>% drop_na()
```

	Ozone	Solar.R	Wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8.0	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	23	299	8.6	65	5	7
6	19	99	13.8	59	5	8
7	8	19	20.1	61	5	9
8	16	256	9.7	69	5	12
9	11	290	9.2	66	5	13
10	14	274	10.9	68	5	14
11	18	65	13.2	58	5	15
12	14	334	11.5	64	5	16
13	34	307	12.0	66	5	17
14	6	78	18.4	57	5	18
15	30	322	11.5	68	5	19
16	11	44	9.7	62	5	20
17	1	8	9.7	59	5	21
18	11	320	16.6	73	5	22
19	4	25	9.7	61	5	23
20	32	92	12.0	61	5	24
21	23	13	12.0	67	5	28
22	45	252	14.9	81	5	29
23	115	223	5.7	79	5	30

Drop columns with any missing values

Use the `miss_var_which()` function from `naniar`

```
miss_var_which(airquality) # which columns have missing values
```

```
[1] "Ozone"    "Solar.R"
```

Drop columns with any missing values

`miss_var_which` and function from `naniar` (need a data frame)

```
airquality %>% select(!miss_var_which(airquality))
```

	Wind	Temp	Month	Day
1	7.4	67	5	1
2	8.0	72	5	2
3	12.6	74	5	3
4	11.5	62	5	4
5	14.3	56	5	5
6	14.9	66	5	6
7	8.6	65	5	7
8	13.8	59	5	8
9	20.1	61	5	9
10	8.6	69	5	10
11	6.9	74	5	11
12	9.7	69	5	12
13	9.2	66	5	13
14	10.9	68	5	14
15	13.2	58	5	15
16	11.5	64	5	16
17	12.0	66	5	17
18	18.4	57	5	18
19	11.5	68	5	19
20	9.7	62	5	20
21	9.7	59	5	21
22	16.6	73	5	22
23	9.7	61	5	23

Change a value to be NA

Let's say we think that all 0 values should be NA.

```
library(readr)
bike <-read_csv("https://sisbid.github.io/Data-Wrangling/labs/Bike_Lanes.csv")
bike <-bike %>% select(type, dateInstalled)
bike
```

```
# A tibble: 1,631 × 2
```

	type	dateInstalled
	<chr>	<dbl>
1	BIKE BOULEVARD	0
2	SIDEPATH	2010
3	SIGNED ROUTE	2010
4	SIDEPATH	0
5	BIKE LANE	2011
6	SIGNED ROUTE	2007
7	SIGNED ROUTE	2007
8	SIGNED ROUTE	2007
9	BIKE LANE	2009
10	SHARROW	2007

```
# i 1,621 more rows
```

Change a value to be NA

The `na_if()` function of `dplyr` can be helpful for changing all 0 values to NA.

More on `mutate()` soon!

```
bike %>%  
  mutate(dateInstalled = na_if(dateInstalled, 0))
```

```
# A tibble: 1,631 × 2  
  type          dateInstalled  
  <chr>          <dbl>  
1 BIKE BOULEVARD      NA  
2 SIDEPATH           2010  
3 SIGNED ROUTE        2010  
4 SIDEPATH           NA  
5 BIKE LANE           2011  
6 SIGNED ROUTE        2007  
7 SIGNED ROUTE        2007  
8 SIGNED ROUTE        2007  
9 BIKE LANE           2009  
10 SHARROW            2007  
# i 1,621 more rows
```

Change NA to be a value

The `replace_na()` function (part of the `tidyr` package), can do the opposite of `na_if()`.

Replacement must match the class of the other values in the column (e.g., character, numeric).

```
bike %>%  
  mutate(dateInstalled = replace_na(dateInstalled, 2005))
```

```
# A tibble: 1,631 × 2  
  type          dateInstalled  
  <chr>          <dbl>  
1 BIKE BOULEVARD      0  
2 SIDEPATH          2010  
3 SIGNED ROUTE       2010  
4 SIDEPATH           0  
5 BIKE LANE          2011  
6 SIGNED ROUTE       2007  
7 SIGNED ROUTE       2007  
8 SIGNED ROUTE       2007  
9 BIKE LANE          2009  
10 SHARROW           2007  
# i 1,621 more rows
```

Think about **NA**

THINK ABOUT YOUR DATA FIRST!

- ⚠ Sometimes removing **NA** values leads to distorted math - be careful!
- ⚠ Think about what your **NA** means for your data (are you sure ?).
 - Is an **NA** for values so low they could not be reported?
 - Or is it if it was too low and also if there was a different issue (like no one reported)?

Think about **NA**

If it is something more like a zero then you might want it included in your data like a zero instead of an **NA**.

Example: - survey reports **NA** if student has never tried cigarettes - survey reports 0 if student has tried cigarettes but did not smoke that week

⚠ You might want to keep the **NA** values so that you know the original sample size.

Word of caution

⚠ Calculating percentages will give you a different result depending on your choice to include NA values.!

This is because the denominator changes.

Word of caution - Percentages with NA

```
count(bike, dateInstalled) %>% mutate(percent = (n/(sum(n)) *100))
```

```
# A tibble: 9 × 3
  dateInstalled    n percent
  <dbl> <int>   <dbl>
1         0    126    7.73
2      2006     2    0.123
3      2007   368   22.6
4      2008   206   12.6
5      2009    86    5.27
6      2010   625   38.3
7      2011   101    6.19
8      2012   107    6.56
9      2013    10    0.613
```

Word of caution - Percentages with NA

```
bike %>% drop_na(dateInstalled) %>%  
  count(dateInstalled) %>% mutate(percent = (n/(sum(n)) *100))
```

```
# A tibble: 9 × 3  
  dateInstalled     n percent  
    <dbl> <int>   <dbl>  
1         0    126    7.73  
2      2006     2    0.123  
3      2007    368   22.6  
4      2008    206   12.6  
5      2009     86    5.27  
6      2010    625   38.3  
7      2011    101    6.19  
8      2012    107    6.56  
9      2013     10    0.613
```

Should you be dividing by the total count with **NA** values included?

It depends on your data and what **NA** might mean.

Pay attention to your data and your **NA** values!

Summary

- `is.na()`, `any(is.na())`, `count()`, and functions from `naniar` like `gg_miss_var()` can help determine if we have **NA** values
- `filter()` automatically removes **NA** values - can't confirm or deny if condition is met (need `| is.na()` to keep them)
- `drop_na()` can help you remove **NA** values from a variable or an entire data frame
- **NA** values can change your calculation results
- think about what **NA** values represent - don't drop them if you shouldn't

<https://sisbid.github.io/Data-Wrangling/labs/missing-data-lab.Rmd>