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 **NA**s.

```
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 `NA`s? (`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)
library(janitor)
ufo <- read_csv(
  "https://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data_complete.csv")
```

```
Warning: One or more parsing issues, call `problems()` on your data frame for
  dat <- vroom(...)
  problems(dat)
```

```
ufo <- clean_names(ufo)

count(ufo, country)
```

```
# A tibble: 6 × 2
  country     n
  <chr>   <int>
1 au        593
2 ca       3266
3 de        112
4 gb       2050
5 us      70293
6 <NA>    12561
```

`naniar`

Sometimes you need to look at lots of data… the `naniar` package is a good option.

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

naniar: `pct_complete()`

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

```
pct_complete(ufo)
```

```
[1] 97.27646
```

Or for a particular variable:

```
ufo |> select(shape) |>
pct_complete()
```

```
[1] 96.4917
```

naniar::miss_var_summary()

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

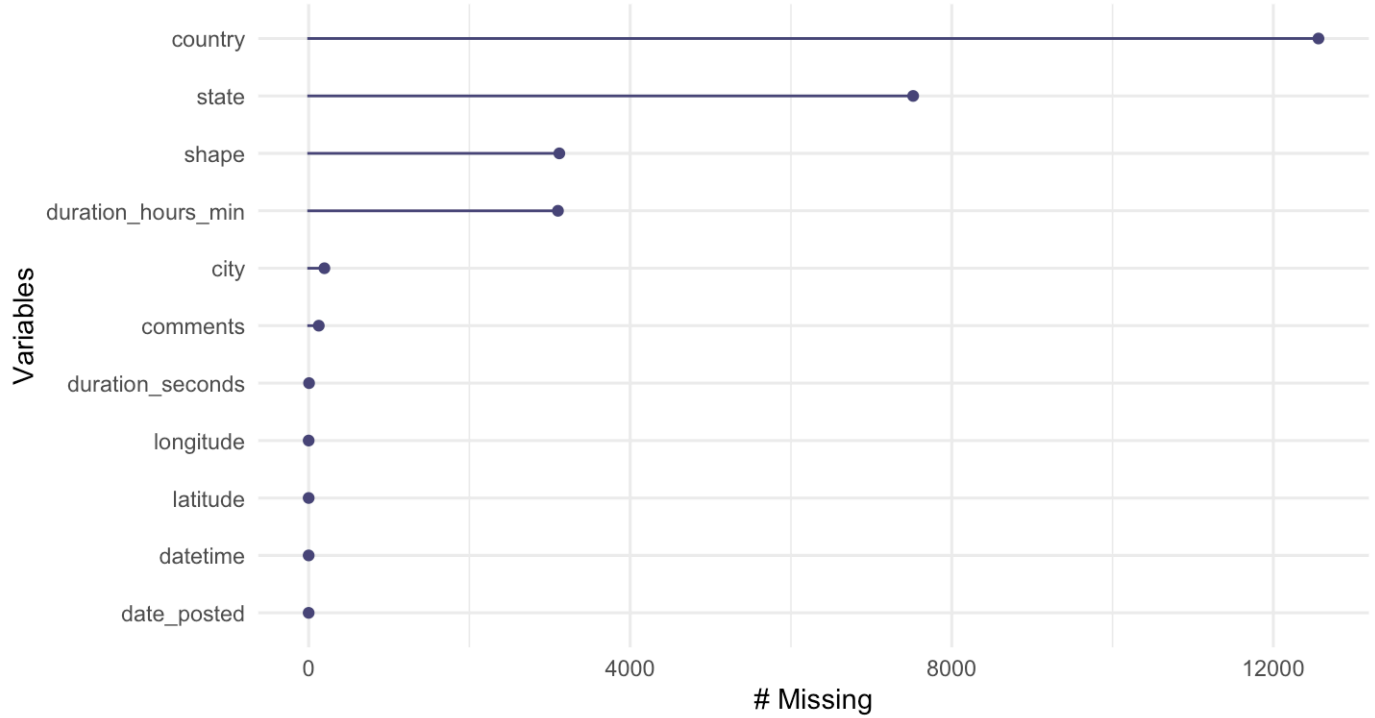
```
miss_var_summary(ufo)
```

```
# A tibble: 11 × 3
   variable           n_miss pct_miss
   <chr>               <int>    <num>
 1 country             12561 14.1
 2 state                7519  8.46
 3 shape                3118  3.51
 4 duration_hours_min   3101  3.49
 5 city                  196  0.221
 6 comments              126  0.142
 7 duration_seconds        5  0.00563
 8 datetime                0  0
 9 date_posted             0  0
10 latitude                0  0
11 longitude               0  0
```

`naniar` plots of number missing

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

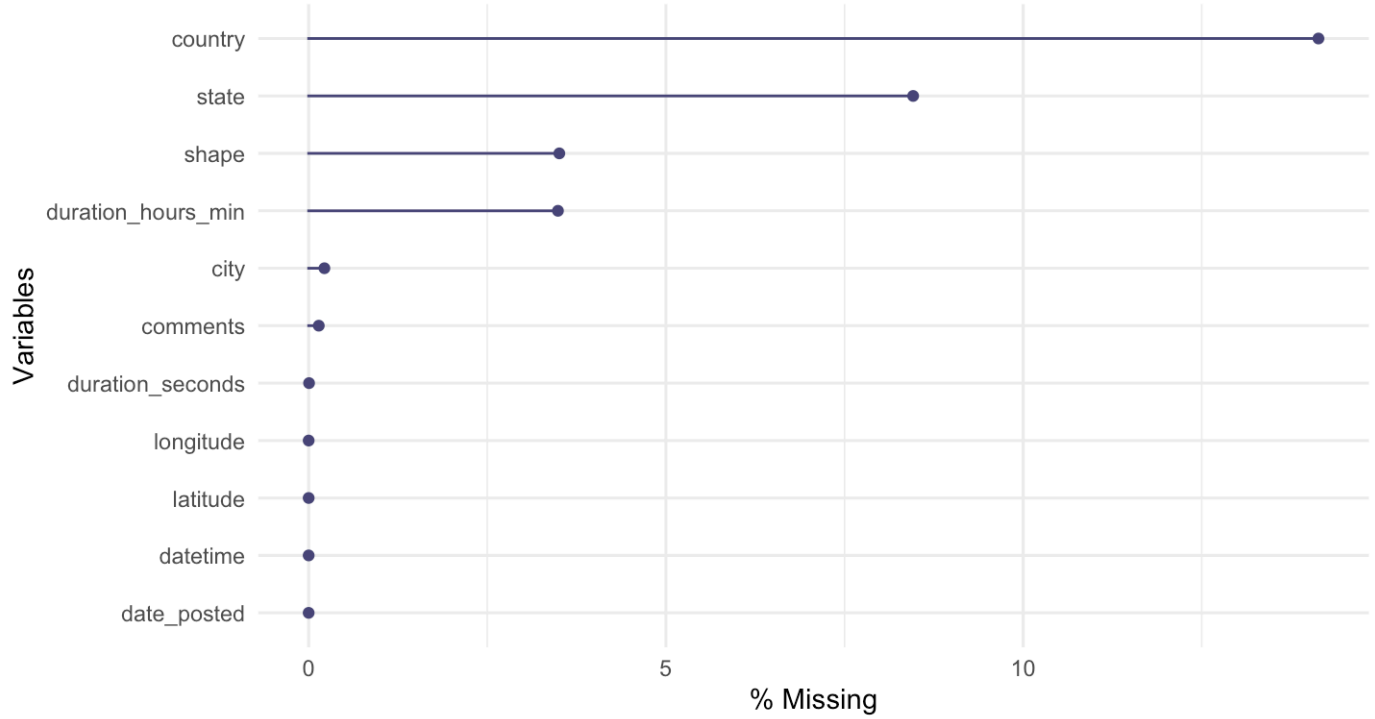
```
gg_miss_var(ufo)
```



`naniar` plots of percent missing

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(ufo, show_pct = TRUE)
```



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.

```
count(ufo, country)
```

```
# A tibble: 6 × 2
  country     n
  <chr>   <int>
1 au        593
2 ca       3266
3 de        112
4 gb       2050
5 us      70293
6 <NA>    12561
```

```
ufo |> filter(country == "de") |> dim()
```

```
[1] 112  11
```

filter() and missing data

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

```
ufo |> filter(country == "de" | is.na(country)) |> dim()
```

```
[1] 12673    11
```

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.

```
ufo
```

```
# A tibble: 88,875 × 11
   datetime         city  state country shape duration_seconds duration_hours_r
   <chr>            <chr> <chr> <chr>   <chr>             <dbl> <chr>
 1 10/10/1949 20:… san … tx    us      cyli…              2700 45 minutes
 2 10/10/1949 21:… lack… tx    <NA>    light              7200 1-2 hrs
 3 10/10/1955 17:… ches… <NA>  gb      circ…                20 20 seconds
 4 10/10/1956 21:… edna  tx    us      circ…                20 1/2 hour
 5 10/10/1960 20:… kane… hi    us      light               900 15 minutes
 6 10/10/1961 19:… bris… tn    us      sphe…               300 5 minutes
 7 10/10/1965 21:… pena… <NA>  gb      circ…               180 about 3 mins
 8 10/10/1965 23:… norw… ct    us      disk               1200 20 minutes
 9 10/10/1966 20:… pell… al    us      disk                180 3  minutes
10 10/10/1966 21:… live… fl    us      disk                120 several minutes
# ℹ 88,865 more rows
# ℹ 4 more variables: comments <chr>, date_posted <chr>, latitude <chr>,
#   longitude <chr>
```

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

```
ufo |> drop_na(state)
```

```
# A tibble: 81,356 × 11
   datetime        city  state country shape duration_seconds duration_hours_n
   <chr>           <chr> <chr> <chr>   <chr>             <dbl> <chr>
 1 10/10/1949 20:… san … tx    us      cyli…              2700 45 minutes
 2 10/10/1949 21:… lack… tx    <NA>    light              7200 1-2 hrs
 3 10/10/1956 21:… edna  tx    us      circ…                20 1/2 hour
 4 10/10/1960 20:… kane… hi    us      light               900 15 minutes
 5 10/10/1961 19:… bris… tn    us      sphe…               300 5 minutes
 6 10/10/1965 23:… norw… ct    us      disk               1200 20 minutes
 7 10/10/1966 20:… pell… al    us      disk                180 3  minutes
 8 10/10/1966 21:… live… fl    us      disk                120 several minutes
 9 10/10/1968 13:… hawt… ca    us      circ…               300 5 min.
10 10/10/1968 19:… brev… nc    us      fire…               180 3 minutes
# ℹ 81,346 more rows
# ℹ 4 more variables: comments <chr>, date_posted <chr>, latitude <chr>,
#   longitude <chr>
```

Drop all NAs with `drop_na()`

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

```
ufo |> drop_na() |> dim()
```

```
[1] 69528    11
```

```
ufo |> drop_na()
```

```
# A tibble: 69,528 × 11
   datetime         city  state country shape duration_seconds duration_hours_r
   <chr>            <chr> <chr> <chr>   <chr>             <dbl> <chr>
 1 10/10/1949 20:… san … tx    us      cyli…              2700 45 minutes
 2 10/10/1956 21:… edna  tx    us      circ…                20 1/2 hour
 3 10/10/1960 20:… kane… hi    us      light               900 15 minutes
 4 10/10/1961 19:… bris… tn    us      sphe…               300 5 minutes
 5 10/10/1965 23:… norw… ct    us      disk               1200 20 minutes
 6 10/10/1966 20:… pell… al    us      disk                180 3  minutes
 7 10/10/1966 21:… live… fl    us      disk                120 several minutes
 8 10/10/1968 13:… hawt… ca    us      circ…               300 5 min.
 9 10/10/1968 19:… brev… nc    us      fire…               180 3 minutes
10 10/10/1970 16:… bell… ny    us      disk               1800 30 min.
# ℹ 69,518 more rows
# ℹ 4 more variables: comments <chr>, date_posted <chr>, latitude <chr>,
#   longitude <chr>
```

Find columns with any missing values

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

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

```
[1] "city"                "state"               "country"
[4] "shape"               "duration_seconds"    "duration_hours_min"
[7] "comments"
```

Drop **columns** with any missing values

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

```
ufo |> select(!miss_var_which(ufo))
```

```
# A tibble: 88,875 × 4
   datetime         date_posted latitude   longitude
   <chr>            <chr>       <chr>      <chr>
 1 10/10/1949 20:30 4/27/2004   29.8830556 -97.9411111
 2 10/10/1949 21:00 12/16/2005  29.38421   -98.581082
 3 10/10/1955 17:00 1/21/2008   53.2       -2.916667
 4 10/10/1956 21:00 1/17/2004   28.9783333 -96.6458333
 5 10/10/1960 20:00 1/22/2004   21.4180556 -157.8036111
 6 10/10/1961 19:00 4/27/2007   36.5950000 -82.1888889
 7 10/10/1965 21:00 2/14/2006   51.434722  -3.18
 8 10/10/1965 23:45 10/2/1999   41.1175000 -73.4083333
 9 10/10/1966 20:00 3/19/2009   33.5861111 -86.2861111
10 10/10/1966 21:00 5/11/2005   30.2947222 -82.9841667
# ℹ 88,865 more rows
```

Change a value to be `NA`

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

The `na_if()` function of `dplyr` can be helpful for changing all 0 values to `NA`. More on `mutate()` soon! (Here we save the change)

```
count(ufo, duration_seconds) |> tail()
```

```
# A tibble: 6 × 2
  duration_seconds     n
             <dbl> <int>
1         25248000     1
2         52623200     3
3         66276000     1
4         82800000     1
5         97836000     1
6               NA     5
```

Change a value to be NA

```
ufo <- ufo |>
  mutate(duration_seconds = na_if(duration_seconds, 0))

count(ufo, duration_seconds) |> tail()
```

```
# A tibble: 6 × 2
  duration_seconds     n
             <dbl> <int>
1         25248000     1
2         52623200     3
3         66276000     1
4         82800000     1
5         97836000     1
6               NA  7228
```

Change `NA` to be a value

The `replace_na()` function (part of the `tidyr` package), can do the opposite of `na_if()`. (Here we just print the change)

```
count(ufo, shape) |> tail()
```

```
# A tibble: 6 × 2
  shape        n
  <chr>    <int>
1 round        2
2 sphere    5755
3 teardrop   817
4 triangle  8489
5 unknown   6319
6 <NA>      3118
```

Change `NA` to be a value

The `replace_na()` function (part of the `tidyr` package), can do the opposite of `na_if()`. (Here we just print the change)

```
ufo |>
  mutate(shape = replace_na(shape, "unknown")) |>
  count(shape) |> tail()
```

```
# A tibble: 6 × 2
  shape         n
  <chr>     <int>
1 rectangle  1418
2 round         2
3 sphere     5755
4 teardrop    817
5 triangle   8489
6 unknown    9437
```

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(ufo, country) |> mutate(percent = (n/(sum(n)) *100)) |>
  arrange(desc(percent))
```

```
# A tibble: 6 × 3
  country     n percent
  <chr>   <int>   <dbl>
1 us      70293  79.1
2 <NA>    12561  14.1
3 ca       3266   3.67
4 gb       2050   2.31
5 au        593   0.667
6 de        112   0.126
```

Word of caution - Percentages with NA

```
ufo |> drop_na(country) |>
count(country) |> mutate(percent = (n/(sum(n)) *100)) |>
  arrange(desc(percent))
```

```
# A tibble: 5 × 3
  country     n percent
  <chr>   <int>   <dbl>
1 us      70293  92.1
2 ca       3266   4.28
3 gb       2050   2.69
4 au        593   0.777
5 de        112   0.147
```

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

Link to Lab