

Data I/O + Structure

Data Wrangling in R

What did I just read in?

- `nrow()` displays the number of rows of a data frame
- `ncol()` displays the number of columns
- `dim()` displays a vector of length 2: # rows, # columns

```
nrow(ufo)
```

```
[1] 88875
```

```
ncol(ufo)
```

```
[1] 11
```

```
dim(ufo)
```

```
[1] 88875    11
```

All Column Names

- `colnames()` displays the column names

```
colnames(ufo)
```

```
[1] "datetime"      "city"           "state"  
[4] "country"       "shape"          "duration (seconds)"  
[7] "duration (hours/min)" "comments"       "date posted"  
[10] "latitude"      "longitude"
```

Data Input

- Sometimes you get weird messages when reading in data.
- The `problems()` function shows you any issues with the data read-in.

```
head(problems(ufo))
```

```
# A tibble: 6 x 5
  row col expected actual file
<int> <chr> <chr>      <chr> <chr>
1   877 <NA>  11 columns 12 columns '../data/ufo/ufo_data_complete.csv'
2  1712 <NA>  11 columns 12 columns '../data/ufo/ufo_data_complete.csv'
3  1814 <NA>  11 columns 12 columns '../data/ufo/ufo_data_complete.csv'
4  2857 <NA>  11 columns 12 columns '../data/ufo/ufo_data_complete.csv'
5  3733 <NA>  11 columns 12 columns '../data/ufo/ufo_data_complete.csv'
6  4755 <NA>  11 columns 12 columns '../data/ufo/ufo_data_complete.csv'
```

```
dim(problems(ufo))
```

```
[1] 199    5
```

Data Input

- The `spec()` functions show you the specification of how the data was read in.

```
# Use cols_condense(spec(ufo)) for a summary  
spec(ufo)
```

```
cols(  
  datetime = col_character(),  
  city = col_character(),  
  state = col_character(),  
  country = col_character(),  
  shape = col_character(),  
  `duration (seconds)` = col_double(),  
  `duration (hours/min)` = col_character(),  
  comments = col_character(),  
  `date posted` = col_character(),  
  latitude = col_character(),  
  longitude = col_double()  
)
```

Data Input

Specification can be passed to `readr` functions:

```
ufo_char = read_csv("../data/ufo/ufo_data_complete.csv", col_types = cols(
  .default = col_character()
))
```

Warning: 196 parsing failures.

row	col	expected	actual	file
877	--	11 columns	12 columns	'../data/ufo/ufo_data_complete.csv'
1712	--	11 columns	12 columns	'../data/ufo/ufo_data_complete.csv'
1814	--	11 columns	12 columns	'../data/ufo/ufo_data_complete.csv'
2857	--	11 columns	12 columns	'../data/ufo/ufo_data_complete.csv'
3733	--	11 columns	12 columns	'../data/ufo/ufo_data_complete.csv'

.....
See problems(...) for more details.

```
dim(problems(ufo_char))
```

```
[1] 196    5
```

Data Input: Checking for problems

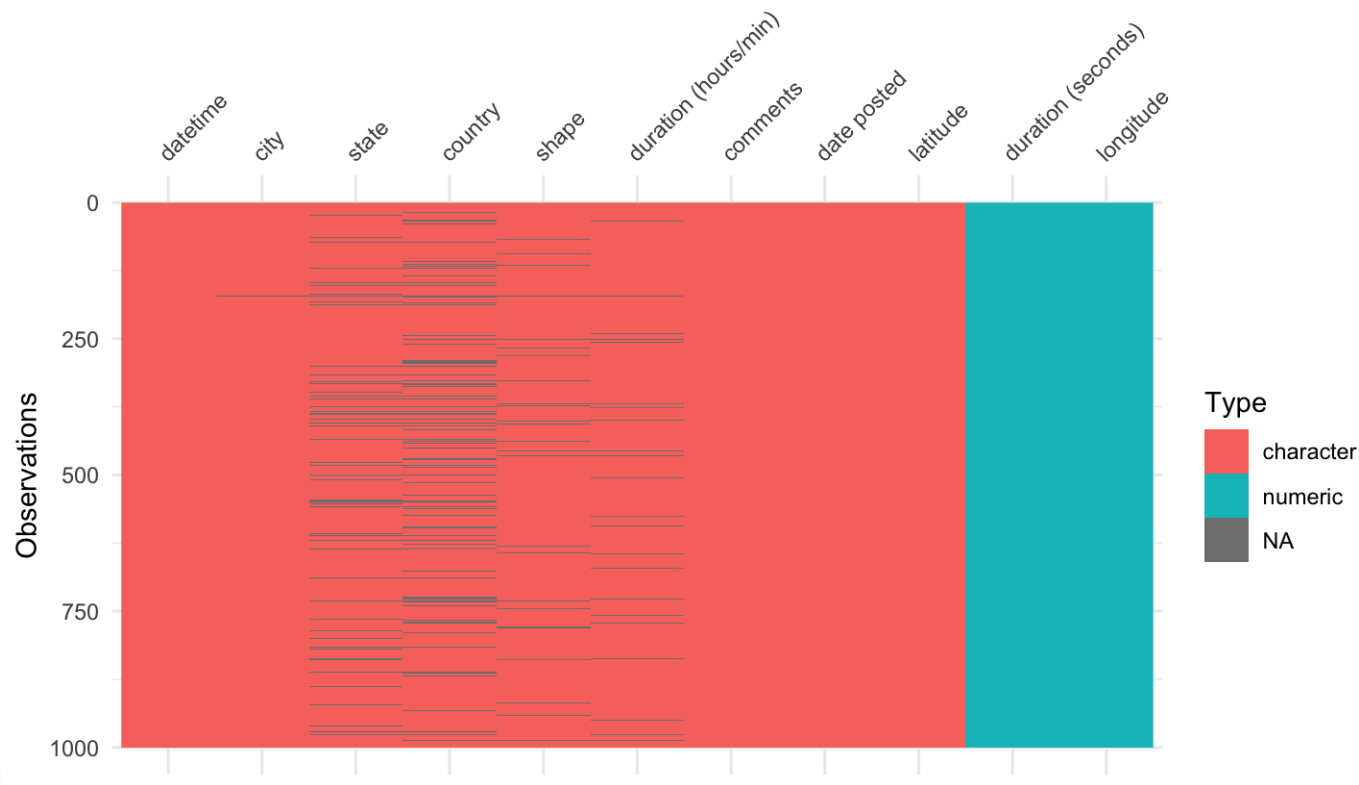
- The `stop_for_problems()` function will stop if your data had an error when reading in. If this occurs, you can use `col_types` (from `spec()`) for the problematic columns:

```
stop_for_problems(ufo)
```

Missing data with the **vizdat**/**naniar** packages

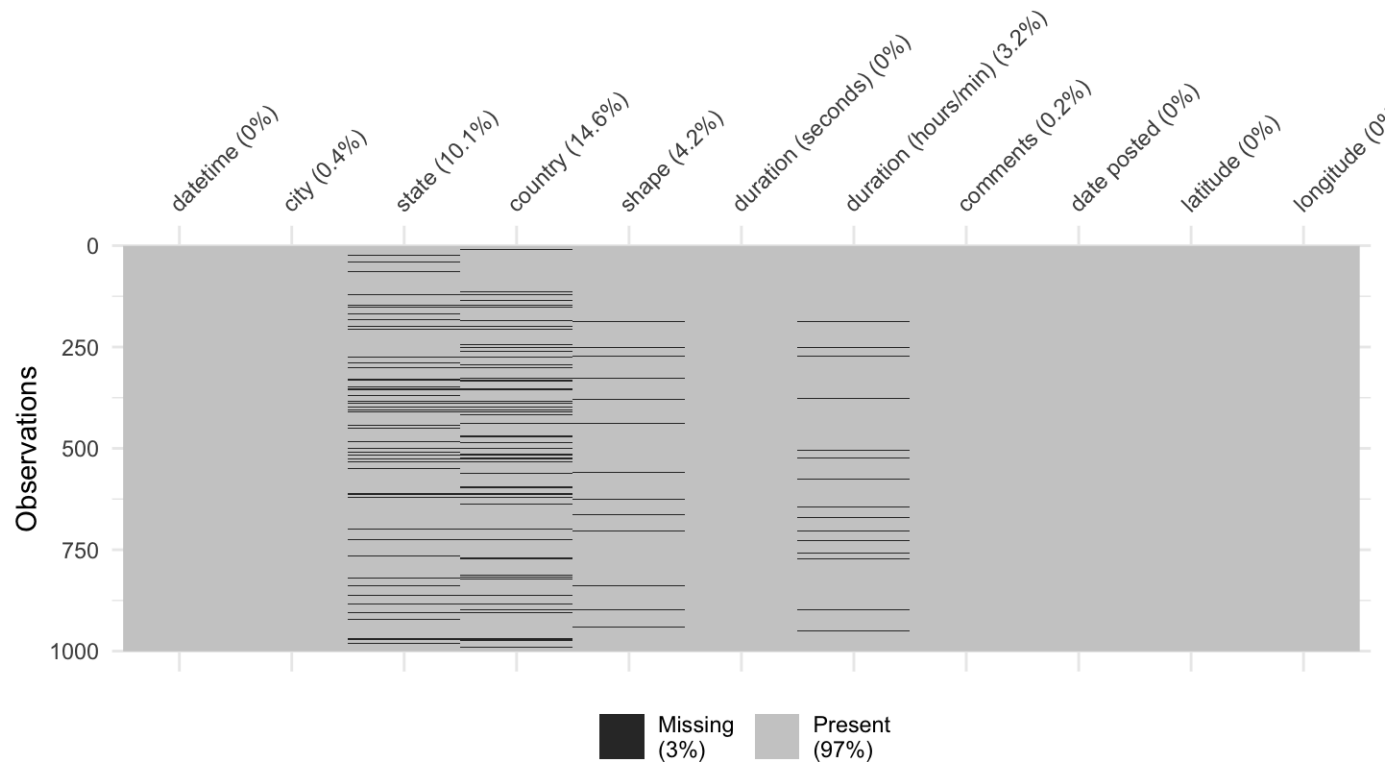
The `vis_dat` function can give you an overview

```
library(vizdat)
ufo_samp = ufo %>% sample_n(size = 1000) # Subset for big data
vis_dat(ufo_samp)
```



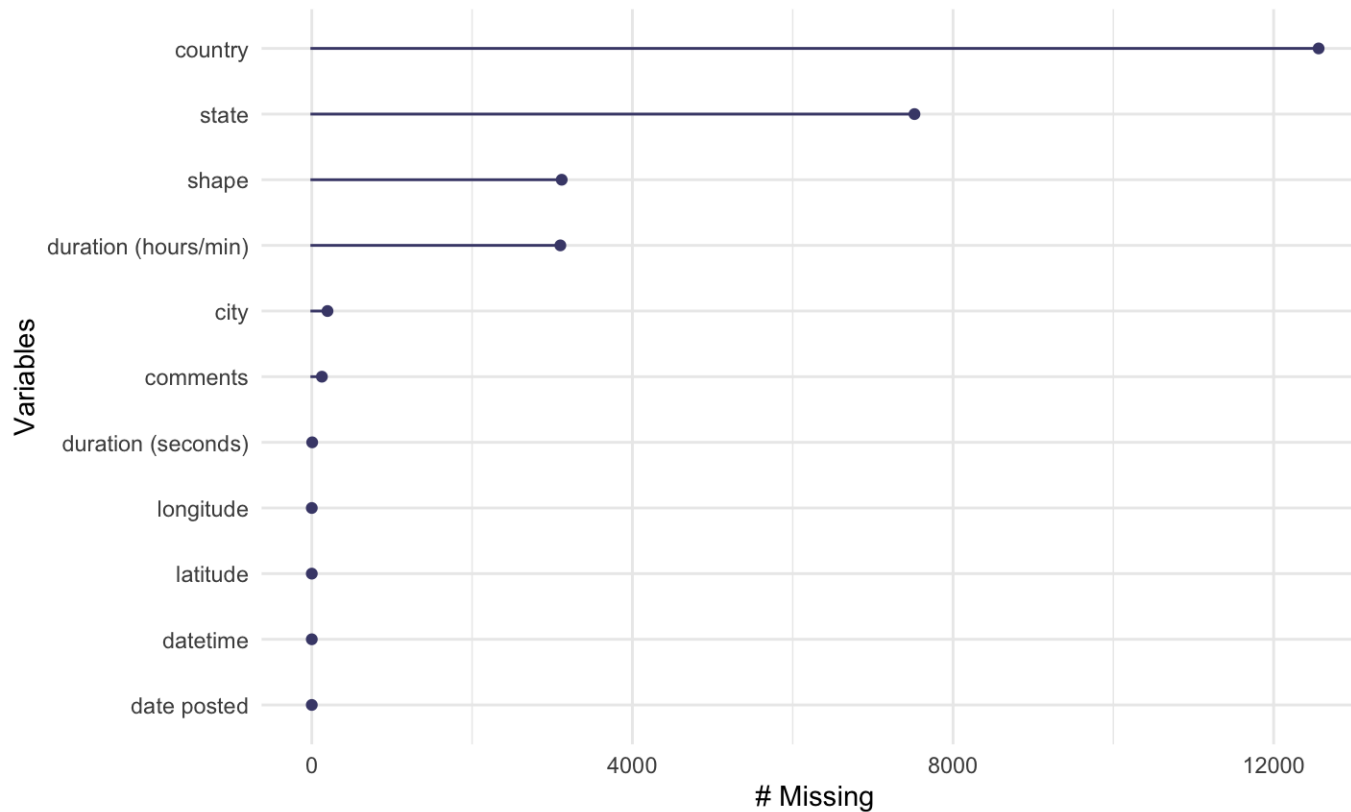
Missing data with the **vizdat**/**naniar** packages

```
vis_miss(ufo_samp)
```



Missing data with the `vizdat`/`naniar` packages

```
library(naniar)  
gg_miss_var(ufo)
```



Missing data with the **vizdat**/**naniar** packages

`miss_case_summary` which rows have missing data in order

```
miss_case_summary(ufo)
```

```
# A tibble: 88,875 x 3
  case n_miss pct_miss
  <int> <int>    <dbl>
1    877      6    54.5
2   1712      6    54.5
3   5613      6    54.5
4   7515      6    54.5
5   7625      6    54.5
6  10156      6    54.5
7  16634      6    54.5
8  18240      6    54.5
9  19813      6    54.5
10 19908      6    54.5
# ... with 88,865 more rows
```

Missing data with the **vizdat**/**naniar** packages

`miss_var_summary` which variables have missing data

```
miss_var_summary(ufo)
```

```
# A tibble: 11 x 3
  variable          n_miss pct_miss
  <chr>          <int>    <dbl>
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
```

After hours of cleaning...

More ways to save: write_rds

If you want to save **one** object, you can use `readr::write_rds` to save to a compressed `rds` file:

```
write_rds(ufo, file = "ufo_dataset.rds", compress = "xz")
```

More ways to save: `read_rds`

To read this back in to R, you need to use `read_rds`, but **need to assign it**:

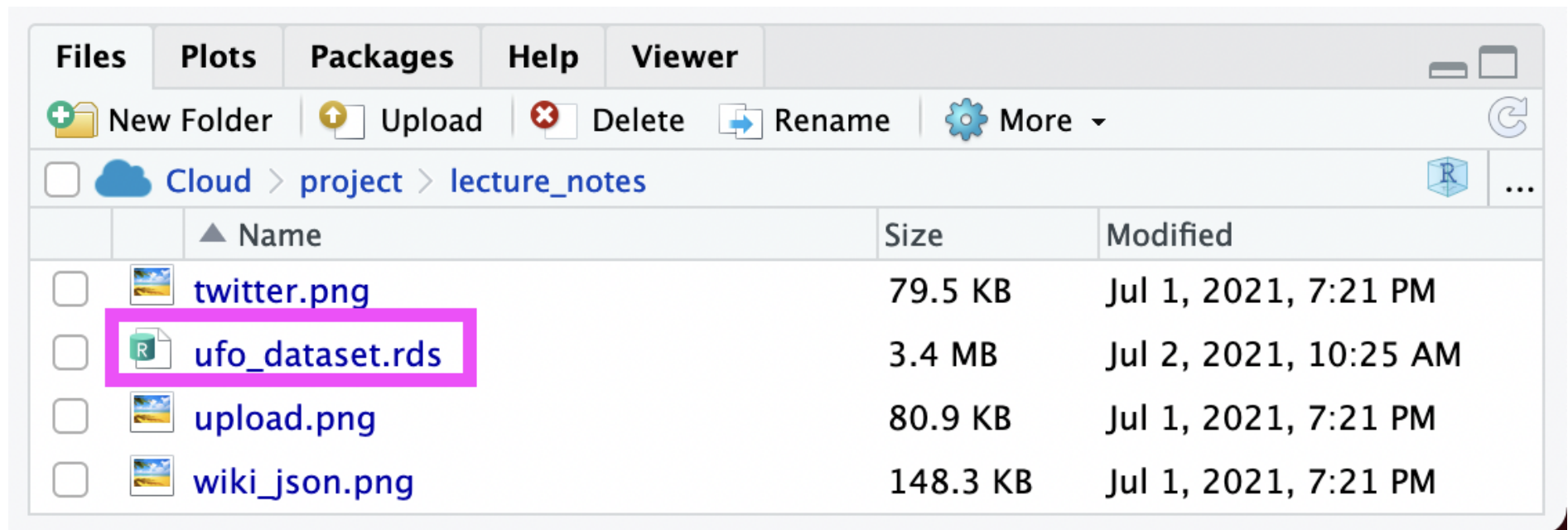
```
ufo3 = read_rds(file = "ufo_dataset.rds")  
identical(ufo, ufo3) # test if they are the same
```

```
[1] TRUE
```

More ways to save: **save**

The `save` command can save a set of R objects into an “R data file”, with the extension `.rda` or `.RData`.

```
x = 5  
save(ufo, x, file = "ufo_data.rda")
```



More ways to save: load

The opposite of `save` is `load`. The `ls()` command lists the items in the workspace/environment and `rm()` removes them

```
load(file = "ufo_data.rda")
```

Data Output

While its nice to be able to read in a variety of data formats, it's equally important to be able to output data somewhere.

`write_delim()`: Write a data frame to a delimited file "This is about twice as fast as `write.csv()`, and never writes row names."

```
args(readr::write_delim)
```

```
function (x, file, delim = " ", na = "NA", append = FALSE, col_names = !append,  
  quote_escape = "double", eol = "\n", path = deprecated())  
NULL
```

Data Output

`x`: A data frame to write to disk

`file`: the file name where you want to R object written. It can be an absolute path, or a filename (which writes the file to your working directory)

`delim`: what character separates the columns?

- `","` = .csv - Note there is also a `write_csv()` function
- `"\t"` = tab delimited

Data Output

For example, we can write back out just the first 100 lines of the `ufo` dataset:

```
write_delim(ufo[1:100,], file = "ufo_first100.csv", delim = ",")  
# same as write_csv(ufo[1:100,], file = "ufo_first100.csv")
```