Missing data in R

This lesson covers how to work with no data values in `R`.

Learning Objectives

At the end of this activity, you will be able to:

- Understand why it is important to make note of missing data values.
- Be able to define what a `NA` value is in `R` and how it is used in a vector.

What you need

You need R and RStudio to complete this tutorial. Also we recommend have you have an `earth-analytics` directory setup on your computer with a `/data` directory with it.

- How to Setup R / R Studio
- Setup your working directory

Missing data - No Data Values

Sometimes, our data are missing values. Imagine a spreadsheet in Microsoft excel with cells that are blank. If the cells are blank, we don't know for sure whether those data weren't collected, or something someone forgot to fill in. To account for data that are missing (not by mistake) we can put a value in those cells that represents `no data`.

The `R` programming language uses the value `NA` to represent missing data values.

```
planets <- c("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus",
             "Neptune", NA)
```

The default setting for most base functions that read data into `R` is to interpret `NA` as a missing value.

Customizing no data values

Sometimes, you'll find a dataset that uses another value for missing data. In some disciplines, for example -999 is sometimes used. If there are multiple types of missing values in your dataset, you can extend what `R` considers a missing value when it reads the file in using "`na.strings`" argument. For instance, if you wanted to read in a `.CSV` file named `temperature_example.csv` that had missing values represented as an empty cell, a single blank space, and the value -999, you would use:

```
# download file
download.file("https://ndownloader.figshare.com/files/7275959",
              "data/week2/temperature_example.csv")

# import data but don't specify no data values - what happens?
temp_df <- read.csv(file = "data/week2/temperature_example.csv")

# import data but specify no data values - what happens?
temp_df2 <- read.csv(file = "data/week2/temperature_example.csv", na.strings = c("NA", " ", "-999"))
```

In the example below, note how a mean value is calculated differently depending upon on how NA values are treated when the data are imported.

```
mean(temp_df$avg_temp)
## [1] NA
mean(temp_df2$avg_temp)
## [1] NA

mean(temp_df2$avg_temp, na.rm = TRUE)
## [1] 56.25
```

Notice a difference between `temp_df` and `temp_df2` ?

Challenge

- **Question**: Why, in the the example above did mean(temp_df$avg_temp) return a value of NA?

When performing mathematical operations on numbers in `R`, most functions will return the value `NA` if the data you are working with include missing values. This allows you to see that you have missing data in your dataset. You can add the argument `na.rm=TRUE` to calculate the result while ignoring the missing values.

```
heights <- c(2, 4, 4, NA, 6)
mean(heights)
## [1] NA
max(heights)
## [1] NA
mean(heights, na.rm = TRUE)
## [1] 4
max(heights, na.rm = TRUE)
## [1] 6
```

If your data include missing values, you may want to become familiar with the functions `is.na()`, `na.omit()`, and `complete.cases()`. See below for examples.

```
# Extract those elements which are not missing values.
heights[!is.na(heights)]
## [1] 2 4 4 6

# Returns the object with incomplete cases removed. The returned object is atomic.
na.omit(heights)
## [1] 2 4 4 6
## attr(,"na.action")
## [1] 4
## attr(,"class")
## [1] "omit"

# Extract those elements which are complete cases.
heights[complete.cases(heights)]
## [1] 2 4 4 6
```

Challenge

- **Question**: Why does the following piece of code return a warning?

```r
sample <- c(2, 4, 4, "NA", 6)
mean(sample, na.rm = TRUE)
## Warning in mean.default(sample, na.rm = TRUE): argument is not numeric or
## logical: returning NA
## [1] NA
```

- **Question**: Why does the warning message say the argument is not numeric?