Sequence of Numbers

Raphael Carvalho

09/05/2019

Missing Values

Any operation involving NA generally yields NA as the result. To illustrate, let's create a vector c(44, NA, 5, NA) and assign it to a variable x.

```
x <- c(44, NA, 5, NA)
```

Now, let's multiply x by 3.

```
x * 3
```

```
## [1] 132  NA  15  NA
```

To make things a little more interesting, lets create a vector containing 1000 draws from a standard normal distribution with y <- rnorm(1000).

```
y <- rnorm(1000)
```

Next, let's create a vector containing 1000 NAs with z <- rep(NA, 1000).

```
z <- rep(NA, 1000)
```

Finally, let's select 100 elements at random from these 2000 values (combining y and z) such that we don't know how many NAs we'll wind up with or what positions they'll occupy in our final vector – my_data <- sample(c(y, z), 100).

```
my_data <- sample(c(y, z), 100)
```

Let's first ask the question of where our NAs are located in our data. The is.na() function tells us whether each element of a vector is NA. Call is.na() on my_data and assign the result to my_na.

```
my_na <- is.na(my_data)
```

In our previous discussion of logical operators, we introduced the == operator as a method of testing for equality between two objects. So, you might think the expression my_data == NA yields the same results as is.na(). Give it a try.

```
my_data == NA
```

```
##   [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [24] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [47] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [70] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
##  [93] NA NA NA NA NA NA NA NA
```

Let's give that a try here. Call the sum() function on my_na to count the total number of TRUEs in my_na, and thus the total number of NAs in my_data. Don't assign the result to a new variable.

```
sum(my_na)
```

```
## [1] 42
```

Now that we've got NAs down pat, let's look at a second type of missing value – NaN, which stands for 'not a number'. To generate NaN, try dividing (using a forward slash) 0 by 0 now.

```
0/0
```

```
## [1] NaN
```

Let's do one more, just for fun. In R, Inf stands for infinity. What happens if you subtract Inf from Inf?

```
Inf/Inf
```

```
## [1] NaN
```