

Manejo de Na y otros datos

Armando Ocampo

Not available

Los Na (not available) son objetos que carecen de información dentro de nuestro data set. Esta falta de información puede ser un problema al utilizar algunas funciones. De la misma forma, existen herramientas de trabajo que nos permiten remover estos elementos o trabajar a pesar de ellos. Estos datos se pueden representar de la siguiente manera Na, NA, NAs. Algunos data sets los colocan de forma automática o pueden ser colocados por default en nuestro data set al encontrar una celda vacía.

Vamos a comenzar con algunos ejemplos con vectores completos y vectores con NAs

```
vector_cito <- c(1:10, 20:30,51)
vector_cito

not_a <- c(1:21, NA, 22:30)
not_a

vect_na <- c(1:5, NA, 14:20, NA, 21:25)
vect_na
```

Antes de comenzar a trabajar, te recomiendo hacer un resumen de tu data set

```
summary(vector_cito)
str(vector_cito)

summary(not_a)
str(not_a)

summary(vect_na)
str(vect_na)
```

Al utilizar funciones de estadística como *mean()*, *sum()*, *median()*, etc. Podemos encontrarnos con el problema que generan los Na

```
# sum
sum(not_a)

# mean
mean(not_a)

# median
median(not_a)
```

El tener Na en un conjunto de datos es tan frecuente que algunas funciones tienen la opción de ignorarlos y realizar la función pertinente

```
?sum  
  
sum(not_a, na.rm = TRUE)  
  
mean(not_a, na.rm = TRUE)  
  
median(not_a, na.rm = TRUE)
```

El argumento *na.rm=*, omite los Na para realizar la función. No los elimina del conjunto de datos original

Removiendo Na

No obstante, en algunas ocasiones es preferible determinar el número de NAs que se presentan en un data set y eliminarlos. Para conocer si nuestros datos presentan NA se utiliza la función *is.na()*

```
is.na(not_a)  
  
is.na(vect_na)
```

En conjunto con *which()* podemos conocer la posición en la que se encuentra cada Na

```
which(is.na(not_a))  
  
which(is.na(vect_na))
```

Eliminarlos es fácil, esto se puede realizar a partir de la función *na.omit()*, el cual elimina todos los Na de nuestro conjunto de datos

```
na.omit(not_a)  
  
# este código elimina los NA, para guardar el resultado lo podemos asignar a un  
# objeto nuevo o renombrar el objeto not_a  
  
obj1_sin_na <- na.omit(not_a)  
obj1_sin_na  
  
sum(obj1_sin_na)  
mean(obj1_sin_na)  
var(obj1_sin_na)  
sd(obj1_sin_na)
```

Nota: la función *na.exclude()* realiza un proceso similar para eliminar los Na del conjunto de datos. De hecho, se utiliza de forma similar. ej. *na.exclude(not_a)*

Recode

En ocasiones no es óptimo utilizar *na.omit()* ya que podemos llegar a perder información. Para ello, se recomienda darle un valor promedio a nuestros Na y mantenerlos en el conjunto de datos, este proceso se denomina *recode* o *recodificar*.

```

x <- vect_na

# utilizamos la funcion mean para obtener el promedio de los valores del vector
# sin los Na, y asignamos este valor a los sitios donde existan Na en los
# vectores

x[is.na(x)] <- mean(x, na.rm = TRUE)
# seleccionamos el numero de decimales que queremos observar
x <- round(x, 1)

# utilizamos nuestras funciones
sum(x)

```

Este proceso suele utilizarse cuando estamos trabajando con data frames, matrices, listas o tibble. Ya que `na.omit()` elimina las filas que contienen Na. Por lo que el proceso no es selectivo. En el siguiente ejemplo realizaremos un data frame con valores Na para observar como se genera el proceso de perdida de información.

```

# haciendo un df con informacion de 10 personas
personas <- LETTERS[1:10]
ejercicio_semana <- c(1,1,5,2,NA,4,2,7,NA,1)
salidas_cine <- c(1,2,2,NA,2,2,1,1,1,2)
numeros_mascotas <- c(0,NA,NA,1,0,0,2,2,1,NA)
cafe_dia <- c(1,2,1,3,NA,0,NA,1,2,1)

actividades <- data.frame(personas, ejercicio_semana, salidas_cine, numeros_mascotas, cafe_dia)

actividades

# al utilizar na.omit() podemos observar la pérdida de información que ocurre
# en el data set

na.omit(actividades)

```

Este ejemplo muestra la reducción de la información, para no perder estos datos utilizaremos recode en cada uno de las variables. Recuerda, cada variable funciona como un vector independiente.

```

# limpiando ejercicio a la semana
actividades$ejercicio_semana[is.na(actividades$ejercicio_semana)] <- mean(actividades$ejercicio_semana,
actividades$ejercicio_semana <- round(actividades$ejercicio_semana, 1)

# limpiando salidas al cine
actividades$salidas_cine[is.na(actividades$salidas_cine)] <- mean(actividades$salidas_cine, na.rm = TRUE)
actividades$salidas_cine <- round(actividades$salidas_cine, 1)

# limpiando numero de mascotas
actividades$numeros_mascotas[is.na(actividades$numeros_mascotas)] <- mean(actividades$numeros_mascotas,
actividades$numeros_mascotas <- round(actividades$numeros_mascotas, 1)

# limpiando numero de cafés al día
actividades$cafe_dia[is.na(actividades$cafe_dia)] <- mean(actividades$cafe_dia, na.rm = TRUE)
actividades$cafe_dia <- round(actividades$cafe_dia, 1)

actividades

```

Con esto se limita la información perdida y podemos realizar el resto de nuestro análisis.