Tarea22

Yimmy Eman

2022-07-16

Pregunta 1

1. ¿Por qué TRUE no es un parámetro en nuestra función rescale_0_1()?
2. ¿Qué ocurriría si x contuviera algún NA y nuestro parámetro na.rm fuera FALSE?

```
rescale01 <- function(x) {
  rng <- range(x, na.rm = TRUE, finite = TRUE)
  (x - rng[1]) / (rng[2] - rng[1])
}

rescale01_alt <- function(x, na.rm = FALSE) {
  rng <- range(x, na.rm = na.rm, finite = TRUE)
  (x - rng[1]) / (rng[2] - rng[1])
}
rescale01_alt(c(NA, 1:5), na.rm = FALSE)
```

```
## [1]   NA 0.00 0.25 0.50 0.75 1.00
```

```
rescale01_alt(c(NA, 1:5), na.rm = TRUE)
```

```
## [1]   NA 0.00 0.25 0.50 0.75 1.00
```

3. En una de las variantes de rescale_0_1(), los valores infinitos quedan invariantes. Reescribe la función rescale_0_1() de modo que -Inf se reescale a 0, e Inf se reescale a 1.

Pregunta 2

Practica convirtiendo los siguientes fragmentos en funciones. Piensa qué hace cada uno y el mejor nombre que podemos darle. Piensa también qué argumentos serán necesarios y cómo puedes escribir el código para que sea lo más expresivo posible y lo menos duplicativo.

- mean(is.na(x))
- x / sum(x, na.rm = TRUE)
- sd(x, na.rm = TRUE) / mean(x, na.rm = TRUE)

```r
prop_na <- function(x) {
  mean(is.na(x))
}
prop_na(c(0, 1, 2, NA, 4, NA))
```

```
## [1] 0.3333333
```

Pregunta 3

Lee el código fuente de las siguientes funciones, investiga qué hacen y en base a ello da un mejor nombre a cada una de ellas.

```r
f1 <- function(string, prefix)
{substr(string, 1, nchar(prefix)) == prefix}

f2 <- function(x) {
  if (length(x) <= 1) return(NULL)
  x[-length(x)] }

f3 <- function(x, y){
  rep(y, length.out = length(x))}
```

```r
f1(c("abc", "abcde", "ad"), "ab")
```

```
## [1]  TRUE  TRUE FALSE
```

```r
f2(1:3)
```

```
## [1] 1 2
```

```r
f2(1:2)
```

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

```r
f2(1)
```

```
## NULL
```

```r
f3(1:3, 4)
```

```
## [1] 4 4 4
```

Pregunta 4

1. Compara y contrasta las funciones rnorm() y MASS::mvrnorm(). ¿Cómo podrías hacer que fueran más consistentes?

2. Investiga y da razones por las cuales las funciones norm_r(), norm_d(). . . . son mucho mejores que las actuales rnorm(), dnorm().

rnorm() toma muestras de la distribución normal univariante, mientras que MASS::mvrnorm toma muestras de la distribución normal multivariante. Los argumentos principales en rnorm() son n, mean, sd. Los principales argumentos es MASS::mvrnorm son n, mu, Sigma. Para ser coherentes, deben tener los mismos nombres. Sin embargo, esto es difícil. En general, es mejor ser coherente con las funciones más utilizadas, p. rmvnorm() debe seguir las convenciones de rnorm(). Sin embargo, mientras que la media es correcta en el caso multivariante, sd no tiene sentido en el caso multivariante. Sin embargo, ambas funciones son internamente consistentes. No sería una buena práctica tener mu y sd como argumentos o mean y Sigma como argumentos.

Pregunta 5

1. Busca en internet el coeficiente de skewness y de curtosis. Implementa una función que lo calcule directamente a partir de un vector de datos o la columna de un dataset.

```
skewness <- function(x, na.rm = FALSE) {
  n <- length(x)
  m <- mean(x, na.rm = na.rm)
  v <- var(x, na.rm = na.rm)
  (sum((x - m) ^ 3) / (n - 2)) / v ^ (3 / 2)
}
```

2. Escribe una función llamada both_na(), que tome dos vectores de la misma longitud y devuelva el número de posiciones donde ambos vectores tienen un NA común en la misma posición.

```
both_na <- function(x, y) {
  sum(is.na(x) & is.na(y))
}
both_na(
  c(NA, NA, 1, 2),
  c(NA, 1, NA, 2)
)
```

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

```
both_na(
  c(NA, NA, 1, 2, NA, NA, 1),
  c(NA, 1, NA, 2, NA, NA, 1)
)
```

```
## [1] 3
```

3. Las dos siguientes funciones son muy útiles. Investiga que crees que hace cada una de ellas:

- is_directory <- function(x) file.info(x)$isdir
- is_readable <- function(x) file.access(x, 4) == 0

La función is_directory() verifica si la ruta en x es un directorio. La función is_readable() verifica si la ruta en x es legible, lo que significa que el archivo existe y el usuario tiene permiso para abrirlo. Estas funciones son útiles aunque son cortas porque sus nombres hacen mucho más claro lo que está haciendo el código.