Ejercicios r4ds

ronny hdez-mora

5/25/2019

Parte III

Capítulo 15

Funciones

##2 Hacer función propia para la varianza

```
secuencia <- c(10, 15, 30, 28, NA, 8)

varianza <- function(x) {
  media <- mean(secuencia, na.rm = TRUE)
  total <- length(secuencia) - 1
  sumatoria <- sum(cuadrado <- (x - media)^2)
  return(sumatoria / total)
}
```

##5 Hacer funcion para tomar dos vectores y

```
secuencia_a <- c(10, 15, 30, 28, NA, 8)
secuencia_b <- c(10, NA, 30, 28, 17, 8)

both_na <- function(x, y) {
  return(paste("La posición de x es:", which(is.na(x)), ", y",
               "la posición de y es:", which(is.na(y))))
}

both_na(secuencia_a, secuencia_b)
```

```
## [1] "La posición de x es: 5 , y la posición de y es: 2"
```

##6 ¿Qué hacen las siguientes funciones?

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

Functions are for humans and computers

##1 Leer el código de las funciones y hacer nombres

```
# Tomar en cuenta lo que hace doble o simple & |
a <- c(1:5)
b <- c(1:5)

c <- c(6:10)
d <- c(6:10)
```

```
a == b & c == d
```

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

```
a == b && c == d
```

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

```
5 > 4 & 6 > 3
```

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

```
5 > 4 && 6 > 3
```

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

```
5 < 4 & 6 > 3
```

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

```
5 < 4 && 6 > 3
```

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

```
5 < 4 | 6 > 3
```

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

```
5 < 4 || 6 > 3
```

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

```
a <- function(x, y, op) {
  switch (op,
          plus = x + y,
          minus = x - y,
          times = x * y,
          divide = x / y,
          stop("Unknown op!")
  )
}
```

Escribir funcion que diga buenos dias, buenas tardes o buenas noches de acuerdo a la hora del dia.

```
saludo <- function(hora = lubridate::now()){
  if (hour(hora) <= 12) {
    print("¡Buenos días!")
  } else if (hour(hora) > 12) {
    print("¡Buenas tardes!")
  } else {
    print("¡Buenas noches!")
  }
}
```

##3 Escribir funcion que si numero es divisible por 3 imprime ## fizz, si es divisible por tres y cinco imprime fizzbuzz. Lo ## demas imprime el numero

```
fizzbuzz <- function(x) {
  if(x %% 3 == 0 & x %% 5 == 0) {
    print("fizzbuzz")
  } else if(x %% 3 == 0) {
    print("fizz")
  } else {
    return(x)
  }
}
```

Una funcion de ejemplo con dot-dot-dot

```
comas <- function(...) {
  stringr::str_c(..., collapse = ",")
}

comas(letters[1:5])
```

```
## [1] "a,b,c,d,e"
```

Capítulo 17

Uno de los primeros ejemplos de loops:

```
df <- tibble(
  a = rnorm(10),
  b = rnorm(10),
  c = rnorm(10),
  d = rnorm(10)
)

output <- vector("double", ncol(df))

for (i in seq_along(df)) {
  output[[i]] <- median(df[[i]])
}
```

Ejercicios

1 Generar loops para:

Obtener media de cada columna en mtcars

```
# Estructura para almacenar valores
medias <- vector("double", ncol(mtcars))

# Loop
for (i in seq_along(mtcars)) {
  medias[[i]] <- mean(mtcars[[i]])
}
```

```r
# Mostrar resultados
print(medias)
```

```
##  [1]  20.090625   6.187500 230.721875 146.687500   3.596563   3.217250
##  [7]  17.848750   0.437500   0.406250   3.687500   2.812500
```

Determinar el tipo de cada columna en `fligths13::fligths`

```r
# Crear objeto con datos
vuelos <- nycflights13::flights

# Estructura para almancenar datos
# tipo_columnas <- vector("character", ncol(vuelos))

# Hay que hacer estructura de lista porque clase de columna
# time_hour tiene dos: "POSIXct" "POSIXt" y eso causa problemas
# de espacio en la estructura de vector
tipo_columnas <- list()

# Loop
for (i in seq_along(vuelos)) {
  tipo_columnas[[i]] <- class(vuelos[[i]])
}

# Mostrar resultados
tipo_columnas
```

```
## [[1]]
## [1] "integer"
##
## [[2]]
## [1] "integer"
##
## [[3]]
## [1] "integer"
##
## [[4]]
## [1] "integer"
##
## [[5]]
## [1] "integer"
##
## [[6]]
## [1] "numeric"
##
## [[7]]
## [1] "integer"
##
## [[8]]
## [1] "integer"
##
## [[9]]
## [1] "numeric"
```

```
##
## [[10]]
## [1] "character"
##
## [[11]]
## [1] "integer"
##
## [[12]]
## [1] "character"
##
## [[13]]
## [1] "character"
##
## [[14]]
## [1] "character"
##
## [[15]]
## [1] "numeric"
##
## [[16]]
## [1] "numeric"
##
## [[17]]
## [1] "numeric"
##
## [[18]]
## [1] "numeric"
##
## [[19]]
## [1] "POSIXct" "POSIXt"
```

Computar el número de valores únicos en cada columna de iris

Generar