Tarea26

Yimmy Eman

2022-07-17

Pregunta 1

Escribe bucles que calculen

a. La media para cada columna de mtcars.

```
output <- vector("double", ncol(mtcars))
names(output) <- names(mtcars)
for (i in names(mtcars)) {
  output[i] <- mean(mtcars[[i]])
}
output
```

```
##        mpg        cyl       disp         hp       drat         wt       qsec
##  20.090625   6.187500 230.721875 146.687500   3.596563   3.217250  17.848750
##         vs         am       gear       carb
##   0.437500   0.406250   3.687500   2.812500
```

b. El tipo de dato de cada columna de nycflights13::flights.

```
output <- vector("list", ncol(nycflights13::flights))
names(output) <- names(nycflights13::flights)
for (i in names(nycflights13::flights)) {
  output[[i]] <- class(nycflights13::flights[[i]])
}
output
```

```
## $year
## [1] "integer"
##
## $month
## [1] "integer"
##
## $day
## [1] "integer"
##
## $dep_time
## [1] "integer"
##
```

```
## $sched_dep_time
## [1] "integer"
##
## $dep_delay
## [1] "numeric"
##
## $arr_time
## [1] "integer"
##
## $sched_arr_time
## [1] "integer"
##
## $arr_delay
## [1] "numeric"
##
## $carrier
## [1] "character"
##
## $flight
## [1] "integer"
##
## $tailnum
## [1] "character"
##
## $origin
## [1] "character"
##
## $dest
## [1] "character"
##
## $air_time
## [1] "numeric"
##
## $distance
## [1] "numeric"
##
## $hour
## [1] "numeric"
##
## $minute
## [1] "numeric"
##
## $time_hour
## [1] "POSIXct" "POSIXt"
```

c. El número de valores únicos de cada columna deiris.

```
data("iris")
iris_uniq <- vector("double", ncol(iris))
names(iris_uniq) <- names(iris)
for (i in names(iris)) {
  iris_uniq[i] <- n_distinct(iris[[i]])
}
iris_uniq
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##           35           23           43           22            3
```

d. Genera 10 números aleatorios de una distribución normal de media -10, 0, 10 y 100 respectivamente.

```
n <- 10
mu <- c(-10, 0, 10, 100)
normals <- vector("list", length(mu))
for (i in seq_along(normals)) {
  normals[[i]] <- rnorm(n, mean = mu[i])
}
normals
```

```
## [[1]]
##  [1]  -9.121692  -9.600124 -10.267465  -8.587993  -9.802846 -10.419831
##  [7]  -9.761431 -10.349539 -11.001560 -11.871561
##
## [[2]]
##  [1]  0.528846360  0.896554963 -0.582769268 -0.416555123  0.003502675
##  [6] -0.923909263  0.179503068  0.486294689  0.280915114  1.493834391
##
## [[3]]
##  [1]  7.202033  8.607125 10.900601  9.529994  7.322510 11.160331 11.098792
##  [8] 12.370152  8.378712 10.739067
##
## [[4]]
##  [1]  99.56782  99.32774 100.81717 102.20171  99.97345  99.18179  99.09799
##  [8] 101.82594  98.75153  99.02542
```

Piensa acerca del tipo de dato que tiene que devolver cada bucle antes de empezar a programarlo.

Pregunta 2

Elimina el bucle for de cada ejemplo haciendo uso de una función que ya trabaje sobre vectores:

```
out <- ""
for (x in letters) {
  out <- stringr::str_c(out, x)
}

x <- sample(100)
sd <- 0
for (i in seq_along(x)) {
  sd <- sd + (x[i] - mean(x)) ^ 2}
sd <- sqrt(sd / (length(x) - 1))
x <- runif(100)
out <- vector("numeric", length(x))
out[1] <- x[1]
for (i in 2:length(x)) {
  out[i] <- out[i - 1] + x[i] }
```

Pregunta 3

Combina una función con tus habilidades de programación con bucles para: - Convertir la canción "99 bottles of beer on the wall" en una función. Generalízalo a cualquier número de recipientes conteniendo cualquier tipo de líquido en cualquier superficie posible. - Escribe un bucle que utilice la función prints() para escribir la letra de la canción "Alice the camel"

```
numbers <- c(
  "ten", "nine", "eight", "seven", "six", "five",
  "four", "three", "two", "one"
)
for (i in numbers) {
  cat(str_c("There were ", i, " in the bed\n"))
  cat("and the little one said\n")
  if (i == "one") {
    cat("I'm lonely...")
  } else {
    cat("Roll over, roll over\n")
    cat("So they all rolled over and one fell out.\n")
  }
  cat("\n")
}
```

```
## There were ten in the bed
## and the little one said
## Roll over, roll over
## So they all rolled over and one fell out.
##
## There were nine in the bed
## and the little one said
## Roll over, roll over
## So they all rolled over and one fell out.
##
## There were eight in the bed
## and the little one said
## Roll over, roll over
## So they all rolled over and one fell out.
##
## There were seven in the bed
## and the little one said
## Roll over, roll over
## So they all rolled over and one fell out.
##
## There were six in the bed
## and the little one said
## Roll over, roll over
## So they all rolled over and one fell out.
##
## There were five in the bed
## and the little one said
## Roll over, roll over
## So they all rolled over and one fell out.
##
```

```
## There were four in the bed
## and the little one said
## Roll over, roll over
## So they all rolled over and one fell out.
##
## There were three in the bed
## and the little one said
## Roll over, roll over
## So they all rolled over and one fell out.
##
## There were two in the bed
## and the little one said
## Roll over, roll over
## So they all rolled over and one fell out.
##
## There were one in the bed
## and the little one said
## I'm lonely...
```

Pregunta 4

Imagina que tienes un directorio lleno de archivos CSV que quieres leer con un patrón de nombre y estructura de datos común en su path: files <- dir("data/", pattern = "\.csv$", full.names = TRUE), y quieres leerlos todos haciendo uso de la funciónread_csv(). Escribe un bucle for que los cargue todos en un solo data frame.

```r
files <- dir("data/", pattern = "\\.csv$", full.names = TRUE)
files
```

```
## character(0)
```

```r
df_list <- vector("list", length(files))
for (i in seq_along(files)) {
  df_list[[i]] <- read_csv(files[[i]])
}
print(df_list)
```

```
## list()
```

Pregunta 5

¿Qué ocurre si usamos el bucle for (nm in names(x)) y x no tiene ningún nombre? ¿Y si solo tienen nombre alguno de ellos? ¿Y si algún nombre está repetido?

```r
x <- c(11, 12, 13)
print(names(x))
```

```
## NULL
```

```r
for (nm in names(x)) {
  print(nm)
  print(x[[nm]])
}
```

Pregunta 6

Escribe una función que imprima la media de cada columna numérica en un data frame junto con su nombre. Por ejemplo la llamada show_mean(iris) debería imprimir: Reto adicional: ¿cómo lo harías para alinear los números en columna a pesar de que los nombres de las variables tengan diferente longitud?

```r
#No me funciona show_mean(iris)
```

Pregunta 7

¿Qué hace este código y cómo funciona?

```r
# show_mean(iris)

# No funciona el codigo
```