MÓDULO 1: HERRAMIENTAS BIG DATA

HERRAMIENTAS DE ANALISIS: PROGRAMACIÓN EN R - MICROACTIVIDADES 2

Elizabeth Evelin Peredo Mescco

25-12-2022

1 EJERCICIO 2

Cargamos datos de nuevo.

```
library(datos)
suppressPackageStartupMessages(library(tidyverse))
```

```
?millas
```

```
## starting httpd help server ... done
```

1.1 EJERCICIO 2.1.

Escribe un bucle `for` que guarda en nuevo data frame, la media de las columnas numéricas (de tipo integer o numeric) de `millas` . Presenta mediante `print` el data frame de resumen.

```
print(str(millas[1,]))
```

```
## tibble [1 × 11] (S3: tbl_df/tbl/data.frame)
##  $ fabricante : chr "audi"
##  $ modelo     : chr "a4"
##  $ cilindrada : num 1.8
##  $ anio       : int 1999
##  $ cilindros  : int 4
##  $ transmision: chr "auto(l5)"
##  $ traccion   : chr "d"
##  $ ciudad     : int 18
##  $ autopista  : int 29
##  $ combustible: chr "p"
##  $ clase      : chr "compacto"
## NULL
```

```
# Solución:
r<-list()
for (i in 1:ncol(millas)) {
  if (class(millas[[i]])=="integer" || class(millas[[i]])=="numeric" ) {
     r[names(millas[i])]<-mean(millas[[i]])
  }else{
  }
}
print(r)
```

```
## $cilindrada
## [1] 3.471795
##
## $anio
## [1] 2003.5
##
## $cilindros
## [1] 5.888889
##
## $ciudad
## [1] 16.85897
##
## $autopista
## [1] 23.44017
```

1.2 EJERCICIO 2.2.

Haz lo mismo que en 2.1 pero utilizando `sapply()` en vez del bucle for.

```
# Solución:
sapply(select_if(millas, is.numeric), mean)
```

```
##  cilindrada        anio   cilindros      ciudad   autopista
##    3.471795 2003.500000    5.888889   16.858974   23.440171
```

1.3 EJERCICIO 2.3.

Explica la diferencia entre la función `if()` e `ifelse()`. Pon un ejemplo de uso de ambas.

```
# Solución:
num <- 1:20
r1<-array()
print("IF ELSE")
```

```
## [1] "IF ELSE"
```

```
#if (condición){sentencia}
for (x in num) {
  if(x%% 2 == 0){
    r1[x]<-("Par")
    }else{
      r1[x]<-("Impar")
    }
}
print(r1)
```

```
##  [1] "Impar" "Par"   "Impar" "Par"   "Impar" "Par"   "Impar" "Par"   "Impar"
## [10] "Par"   "Impar" "Par"   "Impar" "Par"   "Impar" "Par"   "Impar" "Par"
## [19] "Impar" "Par"
```

```
#La función if solo considera el primer componente cuando se le pasa un vector, por eso en este caso se uso fo
r para poder realizar la operación y luego se creo un vector "r1"


#-----------------------------------------------------
print("IFELSE")
```

```
## [1] "IFELSE"
```

```
#ifelse(vector, valor_si_TRUE, valor_si_FALSE)
r2<-ifelse(num %% 2 == 0, "Par", "Impar")
print(r2)
```

```
##  [1] "Impar" "Par"   "Impar" "Par"   "Impar" "Par"   "Impar" "Par"   "Impar"
## [10] "Par"   "Impar" "Par"   "Impar" "Par"   "Impar" "Par"   "Impar" "Par"
## [19] "Impar" "Par"
```

```
#La función ifelse realiza la verificación de cada componente y devuelve un vector
```

1.4 EJERCICIO 2.4.

¿Qué parámetros son imprescindibles especificar cuando se leen datos de ancho fijo mediante: `read.fwf()` ?

Explica qué significan y pon un ejemplo.

```
# Solución:

#Crea data frame
table <- data.frame(
  Nombre=c("Ana", "Pedro", "Gabi", "Rodrigo", "Aurelia"),
  Edad=c(3, 3, 6, 6, 8))

write.table(x = table, file = "read.txt", sep = ",",
            row.names = FALSE, col.names = FALSE)

txt <- read.fwf(
  "./read.txt",
  widths=c(15,2),
  header = FALSE,
  sep = ",")

txt
```

V1 <chr>	V2 <int>	V3 <lgl>
"Ana"	3	*NA*
"Pedro"	3	*NA*
"Gabi"	6	*NA*
"Rodrigo"	6	*NA*
"Aurelia"	8	*NA*

5 rows

```
# FILE: La ruta al archivo fuente o la conexión.
# WIDTHS: Un vector de enteros que indican el ancho de cada columna.
# HEADER (VERDADERO/FALSO): Indica si el archivo tiene una fila de encabezado.
# SEP (separador): separador utilizado en la fila del encabezado.
```

1.5 EJERCICIO 2.5.

Calcula la media de millas/galón en autopista para cada `clase` de coche de `millas` .

Presenta la tabla obtenida.

```
# Solución:
millas5<-select(millas,clase, ciudad,autopista)
millas5<-group_by(millas5,clase)
millas5<-summarise(millas5, mean_autopista = mean(autopista))
print(millas5)
```

```
## # A tibble: 7 × 2
##   clase       mean_autopista
##   <chr>                <dbl>
## 1 2asientos             24.8
## 2 compacto              28.3
## 3 mediano               27.3
## 4 minivan               22.4
## 5 pickup                16.9
## 6 subcompacto           28.1
## 7 suv                   18.1
```

1.6 EJERCICIO 2.6.

Incorpora la media de calculada en 2.5. en el data frame `millas` como una nueva columna llamada “autopista_clase”.

> Utiliza la funcion `merge()` para juntar el objeto obtenido en 2.5 con `millas` .

Presenta el summary() de la nueva columna.

```
# Solución:
# LEFT JOIN
# merge(x = df_1, y = df_2, all.x = TRUE)
merge(millas,millas5, all.x = TRUE)
```

clase <chr>	fabricante <chr>	modelo <chr>	cilindrada <dbl>	an... <int>	cilindros <int>	transmision <chr>	traccion <chr>	ciudad <int>	autopista <int>
2asientos	chevrolet	corvette	7.0	2008	8	manual(m6)	t	15	24
2asientos	chevrolet	corvette	6.2	2008	8	manual(m6)	t	16	26
2asientos	chevrolet	corvette	6.2	2008	8	auto(s6)	t	15	25
2asientos	chevrolet	corvette	5.7	1999	8	auto(l4)	t	15	23
2asientos	chevrolet	corvette	5.7	1999	8	manual(m6)	t	16	26
compacto	audi	a4	1.8	1999	4	auto(l5)	d	18	29
compacto	audi	a4	1.8	1999	4	manual(m5)	d	21	29
compacto	audi	a4	2.0	2008	4	manual(m6)	d	20	31
compacto	audi	a4	2.0	2008	4	auto(av)	d	21	30
compacto	audi	a4	2.8	1999	6	auto(l5)	d	16	26

1-10 of 234 rows | 1-10 of 12 columns Previous 1 2 3 4 5 6 … 24 Next

1.7 EJERCICIO 2.7.

Utiliza las funciones del package dplyr: `group_by()` y `mutate()` para realizar el mismo calculo que en 2.5. y 2.6. sin necesidad de utilizar `merge()`. Llama a la nueva columna "autopista_clase_dplyr"

> Truco: Utiliza el siguiente ejemplo:
> `datos %>% group_by(var_seg) %>% mutate(nueva_variable=mean(variable))`

Haz un `summary()` para verificar que el resultado es el mismo que en 2.6.

```
# Solución:
millas %>%
  group_by(clase) %>%
  mutate(mean_autopista = mean(autopista)) %>%
  summary()
```

```
##   fabricante           modelo            cilindrada        anio
##  Length:234         Length:234         Min.   :1.600   Min.   :1999
##  Class :character   Class :character   1st Qu.:2.400   1st Qu.:1999
##  Mode  :character   Mode  :character   Median :3.300   Median :2004
##                                        Mean   :3.472   Mean   :2004
##                                        3rd Qu.:4.600   3rd Qu.:2008
##                                        Max.   :7.000   Max.   :2008
##    cilindros     transmision          traccion             ciudad
##  Min.   :4.000   Length:234         Length:234         Min.   : 9.00
##  1st Qu.:4.000   Class :character   Class :character   1st Qu.:14.00
##  Median :6.000   Mode  :character   Mode  :character   Median :17.00
##  Mean   :5.889                                         Mean   :16.86
##  3rd Qu.:8.000                                         3rd Qu.:19.00
##  Max.   :8.000                                         Max.   :35.00
##    autopista     combustible           clase           mean_autopista
##  Min.   :12.00   Length:234         Length:234         Min.   :16.88
##  1st Qu.:18.00   Class :character   Class :character   1st Qu.:18.13
##  Median :24.00   Mode  :character   Mode  :character   Median :27.29
##  Mean   :23.44                                         Mean   :23.44
##  3rd Qu.:27.00                                         3rd Qu.:28.14
##  Max.   :44.00                                         Max.   :28.30
```

1.8 EJERCICIO 2.8.

Analiza si `millas` tiene registros duplicados y en caso afirmativo crea un nuevo data frame que contenga una única copia de cada fila.

```
# Solución:
# Cantidad de filas en el Dataframe es 234
nrow(millas)
```

```
## [1] 234
```

```
# Vector de duplicados
millas8_1<-millas[duplicated(millas), ]
millas8_1
```

fabricante <chr>	modelo <chr>	cilindrada <dbl>	an... <int>	cilindros <int>	transmision <chr>	traccion <chr>	ciudad <int>	autopista <int>
chevrolet	c1500 suburban 2wd	5.3	2008	8	auto(l4)	t	14	20
dodge	caravan 2wd	3.3	1999	6	auto(l4)	d	16	22
dodge	caravan 2wd	3.3	2008	6	auto(l4)	d	17	24
dodge	dakota pickup 4wd	4.7	2008	8	auto(l5)	4	14	19
dodge	durango 4wd	4.7	2008	8	auto(l5)	4	13	17
dodge	ram 1500 pickup 4wd	4.7	2008	8	auto(l5)	4	13	17
dodge	ram 1500 pickup 4wd	4.7	2008	8	manual(m6)	4	12	16
ford	explorer 4wd	4.0	1999	6	auto(l5)	4	14	17
honda	civic	1.6	1999	4	auto(l4)	d	24	32

9 rows | 1-9 of 11 columns

```
# Cantidad de filas duplicadas es 9
nrow(millas[duplicated(millas), ])
```

```
## [1] 9
```

```
# Al eiminar los duplicados el número de filas en el Dataframe es 225
millas8_2<-millas[!duplicated(millas), ]
nrow(millas8_2)
```

```
## [1] 225
```

```
print(millas8_2)
```

```
## # A tibble: 225 × 11
##    fabrica…¹ modelo cilin…²  anio cilin…³ trans…⁴ tracc…⁵ ciudad autop…⁶ combu…⁷
##    <chr>     <chr>    <dbl> <int>   <int> <chr>   <chr>    <int>   <int> <chr>
##  1 audi      a4         1.8  1999       4 auto(l… d           18      29 p
##  2 audi      a4         1.8  1999       4 manual… d           21      29 p
##  3 audi      a4         2    2008       4 manual… d           20      31 p
##  4 audi      a4         2    2008       4 auto(a… d           21      30 p
##  5 audi      a4         2.8  1999       6 auto(l… d           16      26 p
##  6 audi      a4         2.8  1999       6 manual… d           18      26 p
##  7 audi      a4         3.1  2008       6 auto(a… d           18      27 p
##  8 audi      a4 qu…     1.8  1999       4 manual… 4           18      26 p
##  9 audi      a4 qu…     1.8  1999       4 auto(l… 4           16      25 p
## 10 audi      a4 qu…     2    2008       4 manual… 4           20      28 p
## # … with 215 more rows, 1 more variable: clase <chr>, and abbreviated variable
## #   names ¹fabricante, ²cilindrada, ³cilindros, ⁴transmision, ⁵traccion,
## #   ⁶autopista, ⁷combustible
```

1.9 EJERCICIO 2.9.

Crea una función que tenga como input la fecha de tu nacimiento (en formato date) y devuelva tu edad en años.

```
# Solución:
# Con la libreria "lubridate" se puede realizar varias funciones en relación a las fechas, en este caso se pue
de poner el formato de ingresar el date "dmy()" = dia/mes/año
# library(lubridate)

edad <- function(date){
   year<-floor(time_length(ymd(Sys.Date()) - dmy(date),
                     unit = "year"))
   return(year)
}

edad("29-10-1997")
```

```
## [1] 25
```

```
#fecha = readline(prompt = "Ingresa la fecha de nacimiento (yyyy-mm-dd) : ")
#edad(fecha)
```

1.10 EJERCICIO 2.10.

Explica porqué el resultado de fechahora_1 y fechahora_2 son distintos en la siguiente expresión:

```
library(lubridate)
Sys.setlocale(locale="es_ES.UTF-8")
```

```
## [1] "LC_COLLATE=es_ES.UTF-8;LC_CTYPE=es_ES.UTF-8;LC_MONETARY=es_ES.UTF-8;LC_NUMERIC=C;LC_TIME=es_ES.UTF-8"
```

```
fechahora <- ymd_hms("2020-03-28 15:11:23", tz = "Europe/Madrid")
fechahora_1 <- fechahora + dhours(24)
fechahora_2 <- fechahora + hours(24)

print(fechahora_1)
```

```
## [1] "2020-03-29 16:11:23 CEST"
```

```
print(fechahora_2)
```

```
## [1] "2020-03-29 15:11:23 CEST"
```

```
# DURATIONS: (dhours)
# Las duraciones miden la cantidad exacta de tiempo que transcurre entre dos instantes. Esto puede generar res
ultados inesperados en relación con las horas del reloj si se produce un segundo bisiesto, un año bisiesto o u
n cambio en el horario de verano (DST) en el intervalo.

# PERIODS: (hours)
# Los períodos miden el cambio en el tiempo del reloj que ocurre entre dos instantes. Los períodos proporciona
n predicciones sólidas de la hora del reloj en presencia de segundos bisiestos, años bisiestos y cambios en el
horario de verano.
```