Modulo 1: Herramientas Big Data

Herramientas de Análisis: Programación en R

Leandro Gutierrez

03/05/2024

EJERCICIO 1

Para el ejercicio 1, utilizaremos los datos los datos `millas` que hay el package `datos`. Estos datos consisten en 238 filas y 11 columnas que describen el consumo de combustible de 38 modelos de coche populares.

Puedes consultar más sobre los datos en la ayuda: `?millas`.

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

```
?millas
```

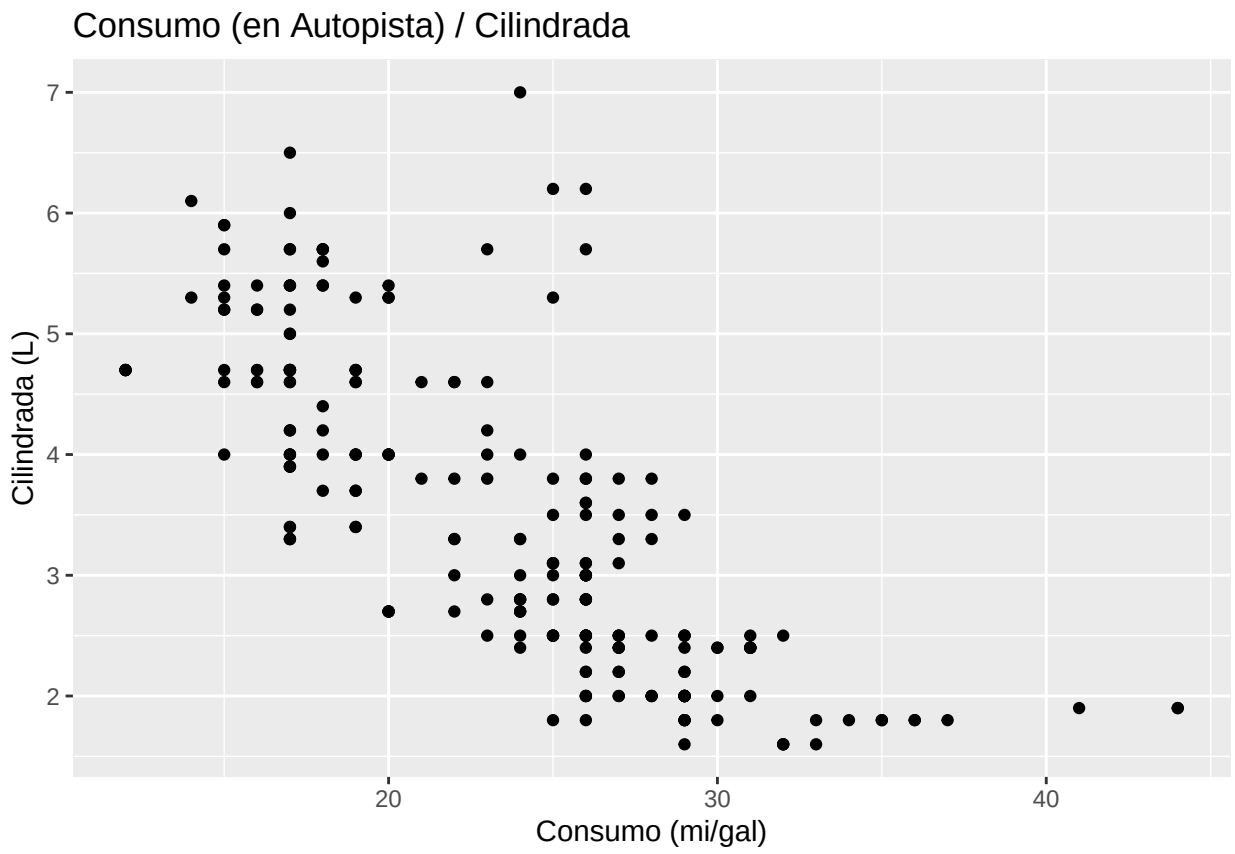
EJERCICIO 1.1.

A partir de los datos de **millas**, dibuja un gráfico de dispersión de puntos que muestre las millas recorridas en autopista por galón de combustible consumido (**autopista**) respecto a la **cilindrada** del motor de cada automóvil. No olvides añadir títulos al gráfico y a los ejes x e y.

Solución

```
e1 <- ggplot(data = millas) +
    geom_point(mapping = aes(x = autopista, y = cilindrada)) +
    labs(title = "Consumo (en Autopista) / Cilindrada"
                  , x = "Consumo (mi/gal)", y = "Cilindrada (L)")

e1
```



EJERCICIO 1.2.

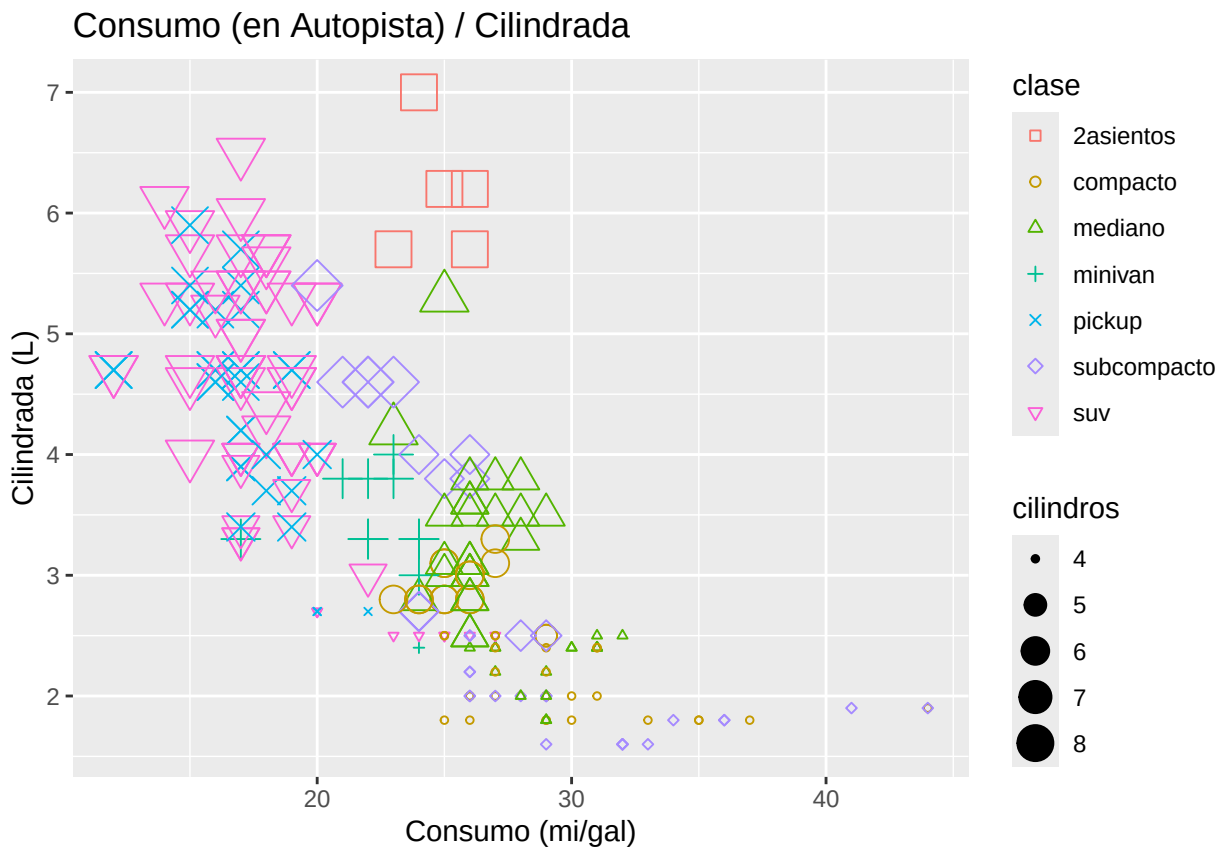
A partir del gráfico del ejercicio 1.1., escoge una columna para cada uno de los siguientes parámetros estéticos: `color`, `size` y `shape`. Comenta algún aspecto relevante que hayas descubierto sobre los coches a partir del gráfico.

> Truco: Observa que puedes seleccionar tanto columnas numéricas como de tipo carácter o factor. Si lo crees interesante, puedes utilizar la misma columna para distintos parámetros del gráfico .

Solución

```
e2 <- e1 + aes(color = clase, size = cilindros, shape = clase) +
    scale_shape_manual(values=seq(0,7))

e2
```



Respuesta

Parece evidenciarse una relación inversa entre *Consumo en Autopista* (medido en millas/galón) y *Cilindrada* del motor (Litros). Los vehículos con motores mas grandes generalmente consumen mas combustible. Del análisis por *Clase* de vehículo y su relación con el *Consumo* surje que *compactos* y *subcompactos* tienen la mejor performance, tambien hay que mencionar que son las clases que suelen tener el tamaño de motor mas pequeño. Las *suv's* y *pickups*, con motores mas grandes, tienden a tener el peor desempeño, en coherencia con el analisis anterior. Los vehículos de *2 asientos* a pesar de tener alta *Cilindrada* mantiene un buen rendimiento. Se observa practicamente una linea recta en el gráfico de dispersión, dato que podía ser útil para modelar el *Consumo de Combustible* en función de la *Cilindrada* del motor.

EJERCICIO 1.3.

Transforma el siguiente vector de tipo `factor` a tipo `numeric` de forma que el valor final mostrado sea exactamente el mismo en ambos vectores, pero con formato distinto. Para ello utiliza `as.character()` y `as.numeric()`.

¿Qué sucede si sólo utilizas `as.numeric()` directamente sobre la columna factor?

```
vec <- factor(c("8", "5", "9", "8", "1", "7"))
print(vec) # valor mostrado
```

```
## [1] 8 5 9 8 1 7
## Levels: 1 5 7 8 9
```

```r
vec3 <- as.numeric(as.character(vec))
print(vec3) # valor esperado
```

```
## [1] 8 5 9 8 1 7
```

```r
class(vec3)
```

```
## [1] "numeric"
```

```r
vec2 <- as.numeric(vec)
print(vec2) # resultado incorrecto
```

```
## [1] 4 2 5 4 1 3
```

Respuesta

`as.numeric()` devuelve por cada elemento del factor, el integer que representa el orden del elemento en el vector ordenado (Levels), mientras que `as.character()` toma el valor real del elemento del factor. Es por lo tanto la combinación `as.numeric(as.character(vec))` la correcta para convertir el vector de tipo `factor` a `numeric` y conservar los valores del vector original.

EJERCICIO 1.4.

Es millas un objeto de la clase *data.frame* o *matrix*?

¿Y el siguiente objeto `obj`?

```r
class(millas)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```r
obj <- cbind(millas$cilindrada, millas$cilindros)
class(obj)
```

```
## [1] "matrix" "array"
```

Respuesta

`millas` es un objeto tipo *tbl_df*, subclase de *data.frame*. Por otro lado el objeto `obj` pertenece a la clase *matrix*, la cual es a su vez una subclase de *array* de 2 dimensiones .

EJERCICIO 1.5.

Crea una función que tome un vector de tipo integer como input y retorne un objeto de tipo lista que contega los siguientes 4 elementos:

1. El último valor del vector

2. Los elementos de las posiciones impares.
3. Todos los elementos excepto el primero.
4. Solo números impares (y no valores faltantes).

```
input <- c(1, 4, 2, -10, 44, 55, -13, 98, 99)

analizar <- function(v) {
  r <- list(v[length(v)], v[seq(1, length(v), 2)], v[-1], v[v %% 2 != 0])
  return(r)
}

res <- analizar(input)
res
```

```
## [[1]]
## [1] 99
##
## [[2]]
## [1]   1   2  44 -13  99
##
## [[3]]
## [1]   4   2 -10  44  55 -13  98  99
##
## [[4]]
## [1]   1  55 -13  99
```

EJERCICIO 1.6.

Busca un ejemplo de objeto x en el que la expresión `x[-which(x > 0)]` no devuelve el mismo resultado que `x[x <= 0]`

```
x <- c(-1, -2, -3, -4)

res1 <- x[-which(x > 0)]
res1
```

```
## numeric(0)
```

```
res2 <- x[x <= 0]
res2
```

```
## [1] -1 -2 -3 -4
```

```
x <- c(1, NaN)

res1 <- x[-which(x > 0)]
res1
```

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

```
res2 <- x[x <= 0]
res2
```

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

Respuesta

Se puede apreciar que en el primer caso, donde ninguno de los elementos del vector satisface la condicion `which(x > 0)`, al estar filtrando por indice con el simbolo *menos* (-) el resultset termina siendo vacio. Quedando la expresion `x[-which(x > 0)]` equivalente a utilizar `x[-0]`. Comportamiento que no se manifiesta al utilizar el filtrado por indice convencional `x[x <= 0]`. Además se puede observar un tratamiento diferente en el análisis de elementos del tipo `NaN` (Not A Number).

EJERCICIO 1.7.

Añade a millas una nueva columna llamada "fabr_mod" que contenga la concatenación del nombre del fabricante, un guion "-" y el modelo del coche. Presenta la nueva columna mediante la función head().

```
head(millas)
```

```
## # A tibble: 6 x 11
##   fabricante modelo cilindrada  anio cilindros transmision traccion ciudad
##   <chr>      <chr>       <dbl> <int>     <int> <chr>       <chr>     <int>
## 1 audi       a4            1.8  1999         4 auto(l5)    d            18
## 2 audi       a4            1.8  1999         4 manual(m5)  d            21
## 3 audi       a4            2    2008         4 manual(m6)  d            20
## 4 audi       a4            2    2008         4 auto(av)    d            21
## 5 audi       a4            2.8  1999         6 auto(l5)    d            16
## 6 audi       a4            2.8  1999         6 manual(m5)  d            18
## # i 3 more variables: autopista <int>, combustible <chr>, clase <chr>
```

```
millas$fabr_mod <- paste(millas$fabricante, millas$modelo, sep="-")

head(millas$fabr_mod, 20)
```

```
##  [1] "audi-a4"                     "audi-a4"
##  [3] "audi-a4"                     "audi-a4"
##  [5] "audi-a4"                     "audi-a4"
##  [7] "audi-a4"                     "audi-a4 quattro"
##  [9] "audi-a4 quattro"             "audi-a4 quattro"
## [11] "audi-a4 quattro"             "audi-a4 quattro"
## [13] "audi-a4 quattro"             "audi-a4 quattro"
## [15] "audi-a4 quattro"             "audi-a6 quattro"
## [17] "audi-a6 quattro"             "audi-a6 quattro"
## [19] "chevrolet-c1500 suburban 2wd" "chevrolet-c1500 suburban 2wd"
```

EJERCICIO 1.8.

Selecciona todos los coches de `millas` que cumplan con todas todas las condiciones siguientes:

- La marca es distinta a “dodge”
- Tiene tracción en las cuatro ruedas
- Han estado fabricados antes del 2008
- Las millas/galón, o bién en ciudad, o bién en carretera, no llegan a 12 millas/galón.

¿Cuantos coches has encontrado?

```
res <- millas[millas$fabricante != "dodge"
            & millas$traccion == "4"
            & millas$anio < 2008
            & (millas$ciudad < 12 | millas$autopista < 12)
    ,]
res
```

```
## # A tibble: 5 x 12
##   fabricante modelo       cilindrada  anio cilindros transmision traccion ciudad
##   <chr>      <chr>             <dbl> <int>     <int> <chr>       <chr>     <int>
## 1 chevrolet  k1500 tahoe~        5.7  1999         8 auto(l4)    4            11
## 2 ford       f150 pickup~        5.4  1999         8 auto(l4)    4            11
## 3 land rover range rover         4    1999         8 auto(l4)    4            11
## 4 land rover range rover         4.6  1999         8 auto(l4)    4            11
## 5 toyota     land cruise~        4.7  1999         8 auto(l4)    4            11
## # i 4 more variables: autopista <int>, combustible <chr>, clase <chr>,
## #   fabr_mod <chr>
```

```
nrow(res)
```

```
## [1] 5
```

Respuesta

Con las condiciones solicitadas solo se encuentran **5 vehículos**. Nota: tanto para la condicion `Año de fabricación antes del 2008` como para `millas/galón menores a 12` se utilizan limites de intervalo superior abiertos.

EJERCICIO 1.9.

Añade una nueva columna “vol_por_cil” a `obj` del ejercicio 1.4. que contenga el ratio de la cilindrada sobre el número de cilindros. Presenta el summary de la nueva columna.

```
obj <- cbind(millas$cilindrada, millas$cilindros)

colnames(obj) <- c('cilindrada', 'cilindros')
vol_por_cil <- obj[,'cilindrada'] / obj[,'cilindros']

obj <- cbind(obj, vol_por_cil)
colnames(obj) <- c('cilindrada', 'cilindros', 'vol_por_cil')

summary(obj)
```

```
##    cilindrada      cilindros      vol_por_cil
##  Min.   :1.600   Min.   :4.000   Min.   :0.4000
##  1st Qu.:2.400   1st Qu.:4.000   1st Qu.:0.5000
##  Median :3.300   Median :6.000   Median :0.5875
##  Mean   :3.472   Mean   :5.889   Mean   :0.5780
##  3rd Qu.:4.600   3rd Qu.:8.000   3rd Qu.:0.6500
##  Max.   :7.000   Max.   :8.000   Max.   :0.8750
```

EJERCICIO 1.10.

Modifica los valores de la columna "vol_por_cil" del objeto `obj` del ejercicio 1.9. asignando NA a los valores de esta columna que sean superiores a 0.7.

Presenta los datos con un summary del nuevo objeto `obj`. ¿Cuántos valores NA se han creado en esta columna?

```
obj[,'vol_por_cil'] <- ifelse(obj[,'vol_por_cil'] > 0.7, NA, obj[,'vol_por_cil'])

summary(obj)
```

```
##    cilindrada      cilindros      vol_por_cil
##  Min.   :1.600   Min.   :4.000   Min.   :0.4000
##  1st Qu.:2.400   1st Qu.:4.000   1st Qu.:0.5000
##  Median :3.300   Median :6.000   Median :0.5750
##  Mean   :3.472   Mean   :5.889   Mean   :0.5644
##  3rd Qu.:4.600   3rd Qu.:8.000   3rd Qu.:0.6250
##  Max.   :7.000   Max.   :8.000   Max.   :0.7000
##                                  NA's   :18
```

Respuesta

Se puede observar en el output de la función `summary` que el valor **Max.** para la columna `vol_por_cil` ahora es **0.7**, y que el conteo de **NA's** es de **18** elementos.