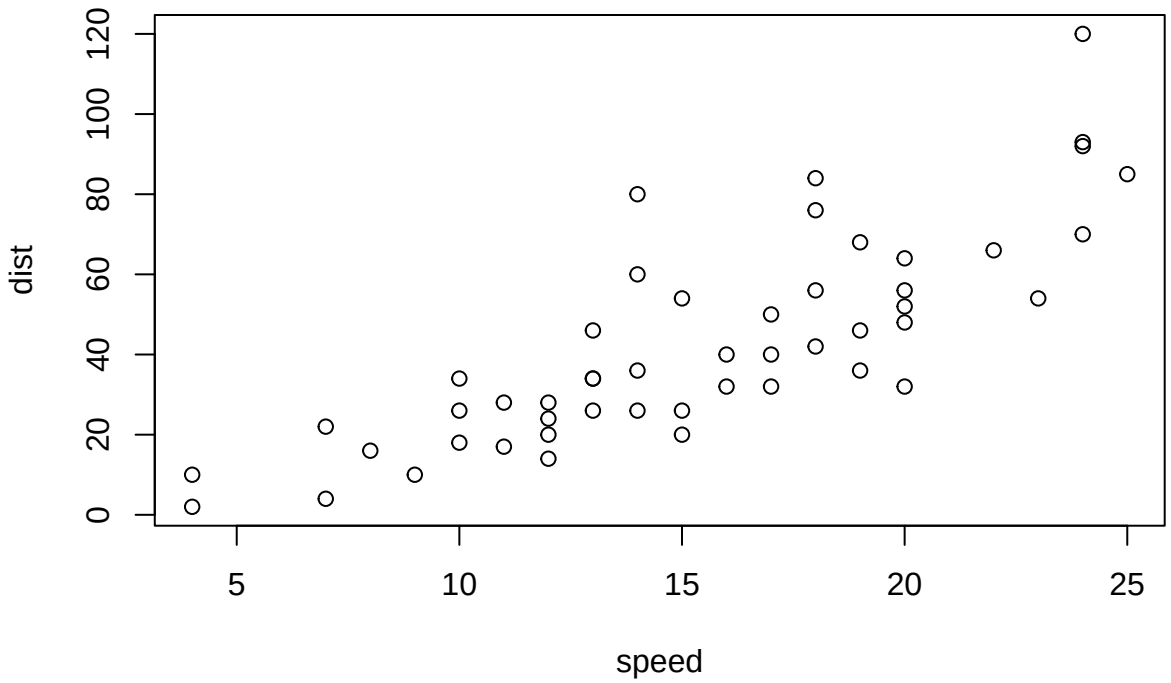
R Notebook

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

```
plot(cars)
```



Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

Estructuras de datos y su manipulación

- Tipos de Estructuras de Datos
  - Aspectos Básicos
  - Vectores
  - Factores
  - Matrices
  - Marcos de Datos
  - Listas

- Manipulación de Datos
  - Aspectos Básicos
  - reshape2
  - tidyr
  - dplyr
  - magrittr
  - Leer y guardar archivos
  - Referencias

Estructura de Datos: Vectores

Definición

La estructura más sencilla de R, contiene una fila de valores del mismo tipo (numérico o cadena de texto)

Se construye con `c()`

Los elementos en el vector se referencian con corchetes `[i]`

Ejemplo:

```
v <- c(5,10,15,20)
v+1
```

```
## [1]  6 11 16 21
```

```
v+v
```

```
## [1] 10 20 30 40
```

```
v <- c("a","b","c","d")
v[1]
```

```
## [1] "a"
```

```
v[1:3]
```

```
## [1] "a" "b" "c"
```

```
v[c(1,4)]
```

```
## [1] "a" "d"
```

Usos

Los vectores numéricos son útiles para cálculos sencillos:

`mean () sd() max() min() length ()`

Ejercicio Considera los vectores x y y:

```
x <- c(4,6,5,7,10,9,4,15)
y <- c(0,10,1,8,2,3,4,1)
```

Que pasa con:

```
x+y
```

```
## [1]  4 16  6 15 12 12  8 16
```

```r
x >7
```

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

```r
c(x,y)
```

```
##  [1]  4  6  5  7 10  9  4 15  0 10  1  8  2  3  4  1
```

```r
length(x)
```

```
## [1] 8
```

Estructuras de Datos: Factores

Definición

- Un vector cuyos valores están organizados en categorías
- Las categorías se llaman ***levels*** y son valores de texto
- Esta nueva capa de información es útil para calcular estadísticos descriptivos

Ejemplo

```r
meses_mix <- c("Enero","Febrero","Marzo","Marzo",
          "Abril","Enero","Abril","Mayo",
          "Junio","Agosto","Julio","Julio",
          "Noviembre","Febrero","May","Agosto",
          "Julio","Diciembre","Enero","Agosto","Septiembre",
          "Noviembre","Febrero","Abril")
fmeses_mix <- factor(meses_mix)
table(fmeses_mix)
```

```
## fmeses_mix
##      Abril     Agosto  Diciembre      Enero    Febrero      Julio      Junio
##          3          3          1          3          3          3          1
##      Marzo        May       Mayo  Noviembre Septiembre
##          2          1          1          2          1
```

Usos

```r
pesos <- rnorm(n=100,mean=50,sd=10)
fpesos <- cut(pesos,breaks=3)
table(fpesos)
```

```
## fpesos
##   (26.3,42]   (42,57.6] (57.6,73.3]
##          20          53          27
```

```r
fpesos <- cut(round(pesos), breaks=quantile(pesos, probs = seq(0, 1, 0.25)), labels=c("1stQ","2ndQ","3r
# advertencia: produce valores NA para los outliers
table(fpesos)
```

```
## fpesos
## 1stQ 2ndQ 3rdQ 4thQ
##   24   25   23   27
```

Ejercicio

Considera el factor:

```r
x <- factor(c("bajo", "alto", "medio", "alto", "alto" , "bajo", "medio"))
```

Ejercicio

Obtén la frecuencia de cada valor

Agrega un valor "muy alto" y haz que aparezca como nivel (tip: append(); de que tipo es levels()?)

Cambia los valores de "bajo" por "no satisfactorio" (tip: levels())

Nota extra: Evitemos los espacios en blanco en los nombres de las variables. Podemos usar guiones bajos "muy_alto" o separar usando mayúsculas "muyAlto". Evitemos usar acentos y caracteres especiales y la ñ.

https://platzi.com/blog/buena-practica-codigo-nombrar-elementos/

Soluciones

Agrega un valor "muy alto":

```r
#levels(x)
append(x=levels(x),values="muy_alto",after=length(x))
```

```
## [1] "alto"     "bajo"     "medio"    "muy_alto"
```

```r
#x
```

Cambia los valores de "bajo" por "no satisfactorio":

```r
levels(x)[2] <- "no_satisfactorio"
levels(x)
```

```
## [1] "alto"             "no_satisfactorio" "medio"
```

```r
#x
```

Estructuras de Datos: Matrices

Definición

Son estructuras que pueden contener información del mismo tipo (numérica o cadena de texto) en dos o más dimensiones (arreglos) Se construyen con: `matrix ( )`

```r
?matrix
```

Elementos en la matriz se referencian usando corchetes y comas: `[i,j]`

Ejemplo

```r
m <- matrix (1:12 , ncol =3)
m <- matrix(1:12,ncol=3,byrow=TRUE)
# m[3,4] # qué pasa? intentar ejecutar sin el #
m #para ver la matriz completa
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

```r
dim(m) # para saber sus dimensiones, concuerda?
```

```
## [1] 4 3
```

```r
m[4,3] #elemento de la fila 4, columna 3
```

```
## [1] 12
```

```r
m[,3] #todos los elementos de la columna 3
```

```
## [1]  3  6  9 12
```

```r
m[4,] #todos los elementos de la fila 4
```

```
## [1] 10 11 12
```

```r
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

```r
m+1
```

```
##      [,1] [,2] [,3]
## [1,]    2    3    4
## [2,]    5    6    7
## [3,]    8    9   10
## [4,]   11   12   13
```

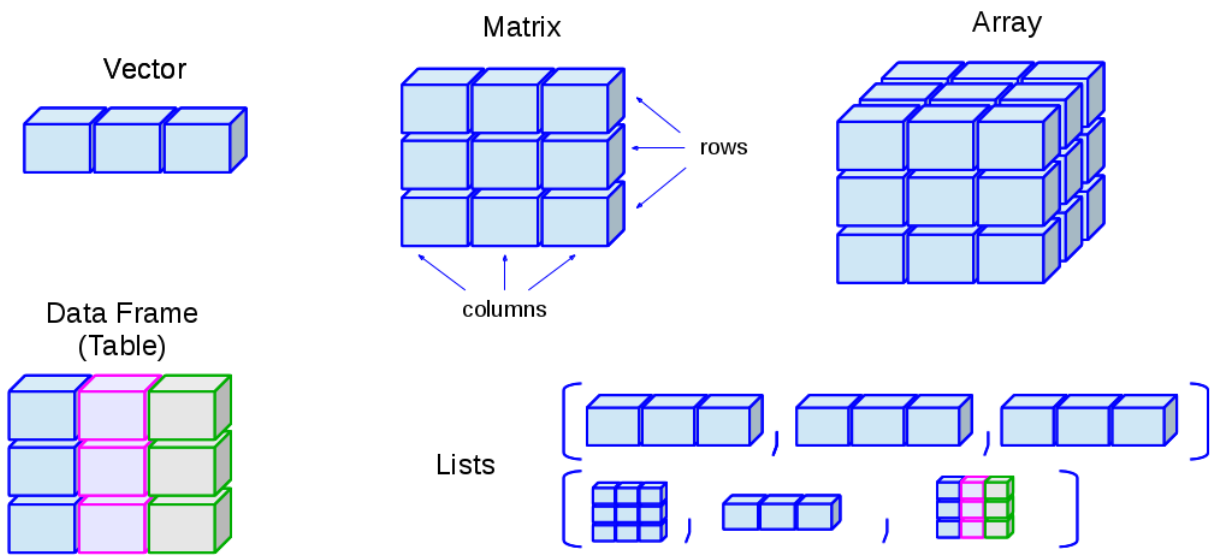
```r
m+m
```

```
##      [,1] [,2] [,3]
## [1,]    2    4    6
## [2,]    8   10   12
## [3,]   14   16   18
## [4,]   20   22   24
```

```r
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8    9
## [4,]   10   11   12
```

Arreglos (*Arrays*): matrices de n dimensiones `[i,j,k]`



Usos

```
m * m #multiplicacion por elemento
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    9
## [2,]   16   25   36
## [3,]   49   64   81
## [4,]  100  121  144
```

```
m %*% t(m) #multiplicacion de matrices > diag(m) #crea vector de una matriz
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   14   32   50   68
## [2,]   32   77  122  167
## [3,]   50  122  194  266
## [4,]   68  167  266  365
```

```
diag(m[3,]) #crea matriz de un vector > rowSums(m)
```

```
##      [,1] [,2] [,3]
## [1,]    7    0    0
## [2,]    0    8    0
## [3,]    0    0    9
```

```
colMeans(m)
```

```
## [1] 5.5 6.5 7.5
```

Ejercicio

- Crea un vector con 12 enteros
- Conviértelo en una matriz de 43 usando `matrix()`
- Cambia los nombres de las columnas a `x, y, z` y las filas a `a, b, c, d`

  tip: `colnames()` *y* `rownames()`
- Obtén una matriz de 3x3 a partir de la matriz previa

- Revisa las dimensiones de `m` con `dim()`

Soluciones

```
#Crea un vector con 12 enteros y conviértelo en una matriz de 4*3
m <- matrix (1:12 , nrow =4)
m
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

```
#Cambia los nombres de las columnas y filas
colnames(m) <- c("x","y","z")
rownames(m) <- c("a","b","c","d")
m
```

```
##   x y  z
## a 1 5  9
## b 2 6 10
## c 3 7 11
## d 4 8 12
```

```
#Obtén una matriz de 3*3
m[1:3,1:3]
```

```
##   x y  z
## a 1 5  9
## b 2 6 10
## c 3 7 11
```

```
#Dimensiones de m
dim(m)
```

```
## [1] 4 3
```

Estructuras de Datos: Marco de Datos

Definición:

Estructuras que pueden combinar argumentos numéricos y de cadena de texto dentro de la misma entidad.

Se construyen con: `data.frame()`

Se pueden referir las columnas (o componentes) usando corchetes o con el operador compacto `$`

Ejemplo:

```
df <- data.frame(distance=c(4,4,4,7,8,5), condition=c("a","a","a","b","b","b"))
df
```

```
##   distance condition
## 1        4         a
## 2        4         a
## 3        4         a
## 4        7         b
## 5        8         b
## 6        5         b
```

```
df[1]
```

```
##   distance
## 1        4
## 2        4
## 3        4
## 4        7
## 5        8
## 6        5
```

```
df["distance"]
```

```
##   distance
## 1        4
## 2        4
## 3        4
## 4        7
## 5        8
## 6        5
```

```
df["distance"][1,1]
```

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

```
df$condition
```

```
## [1] "a" "a" "a" "b" "b" "b"
```

```
df$condition[1]
```

```
## [1] "a"
```

```
head(df)
```

```
##   distance condition
## 1        4         a
## 2        4         a
## 3        4         a
## 4        7         b
## 5        8         b
## 6        5         b
```

```
tail(df)
```

```
##   distance condition
## 1        4         a
## 2        4         a
## 3        4         a
## 4        7         b
## 5        8         b
## 6        5         b
```

Usos:

- Muy similar a una hoja de cálculo estándar
- Uso de cabeceras (headers) significativos => componentes del dataframe `df`
- Se pueden modificar los nombres de filas y columnas
- Se pueden unir dfs con: `cbind ()` y `rbind ()`

Ejercicio:

- Crea un df con los siguientes datos

```
edad <- c(22, 25, 18, 20)
edad
```

```
## [1] 22 25 18 20
```

```
nombres <- c("Jaime", "Mateo", "Olivia", "Sandra")
nombres
```

```
## [1] "Jaime"  "Mateo"  "Olivia" "Sandra"
```

```
genero <- c("M", "M", "F", "F")
genero
```

```
## [1] "M" "M" "F" "F"
```

- Ordena los valores por edad
  tip: `order()`

Soluciones:

- Crea un df

```
df <- data.frame(edades=edad,names_df=nombres,genero=genero)
df
```

```
##   edades names_df genero
## 1     22    Jaime      M
## 2     25    Mateo      M
## 3     18   Olivia      F
## 4     20   Sandra      F
```

- Ordena los valores por edad (age)
  tip: `order()`

```
df[order(df$edades),]
```

```
##   edades names_df genero
## 3     18   Olivia      F
## 4     20   Sandra      F
## 1     22    Jaime      M
## 2     25    Mateo      M
```

Estructuras de Datos: Listas

Definición:

- Una estructura de datos que puede contener todos los demás tipos
- Se construye con `list ()`
- Cada componente se encuentra usando
  - `$` si los componentes tienen nombre
  - `[i]` para accesar al componente como elemento de lista
  - `[[i]]` para accesar al componente como vector

Ejemplo:

```r
vec <- 1:5
vec
```

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

```r
df <- data.frame(y = c(1:3), x = c("m", "m", "f"))
df
```

```
##   y x
## 1 1 m
## 2 2 m
## 3 3 f
```

```r
char <- "Hola!"
char
```

```
## [1] "Hola!"
```

```r
a.list <- list(vec, df, char)
a.list
```

```
## [[1]]
## [1] 1 2 3 4 5
##
## [[2]]
##   y x
## 1 1 m
## 2 2 m
## 3 3 f
##
## [[3]]
## [1] "Hola!"
```

```r
# a.list[1] + 1 # no funciona! porque?
a.list[[1]] + 1
```

```
## [1] 2 3 4 5 6
```

```r
str(a.list) # ver estructura de la lista
```

```
## List of 3
##  $ : int [1:5] 1 2 3 4 5
##  $ :'data.frame':    3 obs. of  2 variables:
##   ..$ y: int [1:3] 1 2 3
##   ..$ x: chr [1:3] "m" "m" "f"
##  $ : chr "Hola!"
```