dcc
CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE CHILE

CC1000
Herramientas Computacionales para Ingeniería y Ciencias

Laboratorio #11 – Manejo de Datos en R

Francisco J. Gutiérrez
frgutier@dcc.uchile.cl

TIPOS DE OBJETOS

- **Vector**: secuencia de elementos, todos del mismo tipo
- **Factor**: vector especial para manejar datos categóricos (no numéricos)
- **Arreglo**: colección de elementos de datos, organizados en varias dimensiones (por ejemplo, cubo de 3 x 3 x 3)
- **Matriz**: arreglo de dos dimensiones
- **Lista**: secuencia de elementos, que pueden ser de cualquier tipo
- **Data Frame**: lista de vectores del mismo largo

PROPIEDADES DE OBJETOS

Todo objeto `x` tiene al menos dos propiedades:

- **`mode(x)`**, que indica el tipo de dato del objeto (por ejemplo, `numeric`, `logical`, `character`)
- **`length(x)`**, que corresponde al tamaño del objeto

```
> x <- 1:10
> mode(x)
[1] "numeric"
> length(x)
[1] 10
>
> x <- c("Ana", NA, "Carlos", "Daniel", "Elena")
> mode(x)
[1] "character"
> length(x)
[1] 5
```

PROPIEDADES DE OBJETOS

Existen las funciones **`as.xxxx(objeto)`**, que permiten cambiar el *modo* de un objeto:

```
> x <- 0:9
> digitos <- as.character(x)
> digitos
 [1] "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
>
> d <- as.integer(digitos)
> d
 [1] 0 1 2 3 4 5 6 7 8 9
```

También existen las funciones **`is.xxxx(objeto)`**, que retornan `TRUE` si el `objeto` es del tipo indicado en la función

MANEJO DE VECTORES

Podemos asignar valores en posiciones mayores que el largo actual:

```
> x <- integer(15)     #vector de largo 15, relleno con 0s
> x
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
>
> e <- integer()       #vector de largo 0
> e[3] <- 17
> e
[1] NA NA 17
> e[10] <- 3           #ojo que como inicialmente no habian datos, todo es NA
> e
 [1] NA NA 17 NA NA NA NA NA NA  3
```

MANEJO DE VECTORES

Podemos cambiar elementos usando el complemento:

```
> e
 [1] NA NA 17 NA NA NA NA NA NA  3
> e[-2] <- 4            #asignar 4 a todo, menos la posicion 2
> e
 [1]  4 NA  4  4  4  4  4  4  4  4
> mode(e)
[1] "numeric"
>
> e[2] <- 1.3
> e
 [1] 4.0 1.3 4.0 4.0 4.0 4.0 4.0 4.0 4.0 4.0
> mode(e)               #como agregamos un decimal, el vector ya no es integer
[1] "numeric"
>
> e[c(1,5,8)] <- "abc"
> e
 [1] "abc" "1.3" "4"   "4"   "abc" "4"   "4"   "abc" "4"   "4"
> mode(e)               #como agregamos texto, el vector ya no es numeric
[1] "character"
```

MANEJO DE FACTORES

Podemos crear un **`factor`**, a partir de un vector de datos categóricos:

```
> x <- c("rojo", "azul", "rojo", "rojo", "azul")
> fact_x <- factor(x)
> fact_x
[1] rojo azul rojo rojo azul
Levels: azul rojo
>
> levels(fact_x)     #siempre se muestran en orden alfabetico
[1] "azul" "rojo"
>
> y <- c("rojo1", "azul", "rojo2", "rojo11", "azul")
> fact_y <- factor(y)
> fact_y            #no hay repetidos
[1] rojo1  azul   rojo2  rojo11 azul
Levels: azul rojo1 rojo11 rojo2
```

Los **`Levels`** de un factor son las categorías identificadas

MANEJO DE FACTORES

Los factores se usan para calcular, de manera agrupada, funciones sobre las categorías:

```
> fact_x
[1] rojo azul rojo rojo azul
Levels: azul rojo
>
> datos <- 1:5
> tapply(datos, fact_x, sum)
azul rojo
   7    8
>
> tapply(datos, fact_x, mean)
    azul     rojo
3.500000 2.666667
```

Ojo que el vector de datos tiene que ser del mismo tamaño que el vector usado para crear el factor!

MANEJO DE FACTORES

La función **`tapply()`** recibe como segundo parámetro una lista de factores:

```
> fact_x
[1] rojo azul rojo rojo azul
Levels: azul rojo
>
> genero <- c("H", "H", "F", "F", "F")
> fact_g <- factor(genero)
>
> tapply(datos, list(fact_x,fact_g), sum)
     F H
azul 5 2
rojo 7 1
```

LISTAS Y DATA FRAMES

Una **lista** es una estructura que almacena elementos:

```
> p <- list(nombre="Juan", num.hijos=2, edades.hijos=c(4,7))
> p
$nombre
[1] "Juan"

$num.hijos
[1] 2

$edades.hijos
[1] 4 7
```

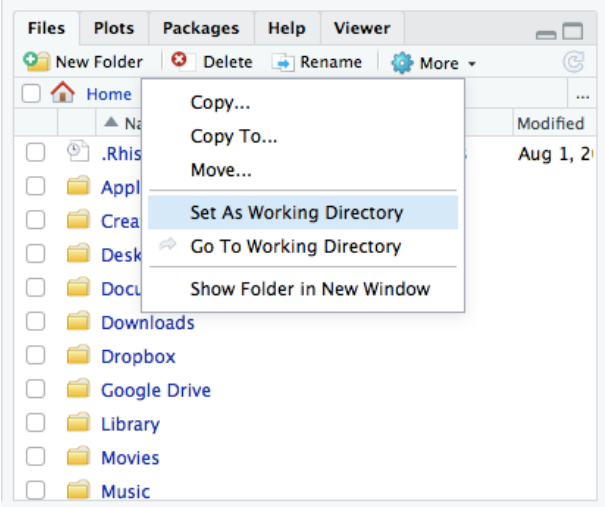
Un **Data Frame** es una lista donde las columnas son del mismo largo, incluyendo nombres para cada una:

```
> mtcars
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
```

MANEJANDO DATOS EN R

Podemos cargar datos desde un archivo a R, guardando la información en un **Data Frame**:

1. Setear el directorio de trabajo donde está guardado el archivo de datos



o bien, usar el comando **`setwd(...)`**

MANEJANDO DATOS EN R

2. Leer el archivo usando el comando **`read.table(…)`**
   - La primera línea (encabezado) contiene le nombre de las columnas
   - Las siguientes *n – 1* filas contienen los registros de datos

```
> setwd("~/Desktop/CC1000")
> data <- read.table("houses.txt")   #En este archivo tenemos los datos
> data
   Price Floor Area Rooms Age Cent.heat
01 52.00   111  830     5 6.2        no
02 54.75   128  710     5 7.5        no
03 57.50   101 1000     5 4.2        no
04 57.50   131  690     6 8.8        no
05 59.75    93  900     5 1.9       yes
```

MANEJANDO DATOS EN R

3. Usamos el operador **$** para acceder a una componente en particular:

```
> data
   Price Floor Area Rooms Age Cent.heat
01 52.00   111  830     5 6.2        no
02 54.75   128  710     5 7.5        no
03 57.50   101 1000     5 4.2        no
04 57.50   131  690     6 8.8        no
05 59.75    93  900     5 1.9       yes
>
> data$Price #vector con todos los elementos de la columna "Price"
[1] 52.00 54.75 57.50 57.50 59.75
```

MANEJANDO DATOS EN R

4. Usamos la función **`subset(dataset,condicion)`** para filtrar el dataset y quedarnos con los registros que cumplan la condición especificada:

```
> data
   Price Floor Area Rooms Age Cent.heat
01 52.00   111  830     5 6.2        no
02 54.75   128  710     5 7.5        no
03 57.50   101 1000     5 4.2        no
04 57.50   131  690     6 8.8        no
05 59.75    93  900     5 1.9       yes
>
> data2 <- subset(data, data$Area > 700)    #filtramos el dataset por la columna Area
> data2
   Price Floor Area Rooms Age Cent.heat
01 52.00   111  830     5 6.2        no
02 54.75   128  710     5 7.5        no
03 57.50   101 1000     5 4.2        no
05 59.75    93  900     5 1.9       yes
```

DISTRIBUCIÓN DE FRECUENCIAS

La función **`table()`** nos permite calcular la frecuencia de cada categoría de un factor

```
> table(painters$School)

 A  B  C  D  E  F  G  H
10  6  6 10  7  4  7  4
>
>
> table(fact_cyl)
fact_cyl
 4  6  8
11  7 14
>
>
> table(crabs$sex, crabs$sp)

    B  O
F 50 50
M 50 50
```

El paquete `MASS` contiene varios datasets, como `painters` y `crabs`

Para usar estos datasets, primero debemos cargar el paquete `MASS`, usando el comando **`library(MASS)`**

ESTADÍSTICAS SOBRE CATEGORÍAS

La función **`table()`** sólo cuenta ocurrencias. Si queremos calcular estadísticas sobre cruces de categorías, debemos usar **`tapply()`**

```
> tapply(crabs$FL, list(crabs$sp, crabs$sex), mean)
       F      M
B 13.270 14.842
O 17.594 16.626
>
>
> tapply(painters$Composition, painters$School, mean)
       A        B        C        D        E        F        G        H
10.40000 12.16667 13.16667  9.10000 13.57143  7.25000 13.85714 14.00000
```

FRECUENCIAS RELATIVAS

La **frecuencia relativa** indica la proporción de una determinada ocurrencia en la muestra observada

La frecuencia se calcula usando **`table(variable)`** y el tamaño de muestra con **`nrow(dataset)`**:

```
> frec_painters <- table(painters$School)
> frec_painters

 A  B  C  D  E  F  G  H
10  6  6 10  7  4  7  4
>
>
> frec_relativa <- frec_painters/nrow(painters)
> frec_relativa

         A          B          C          D          E          F          G
0.18518519 0.11111111 0.11111111 0.18518519 0.12962963 0.07407407 0.12962963
         H
0.07407407
```

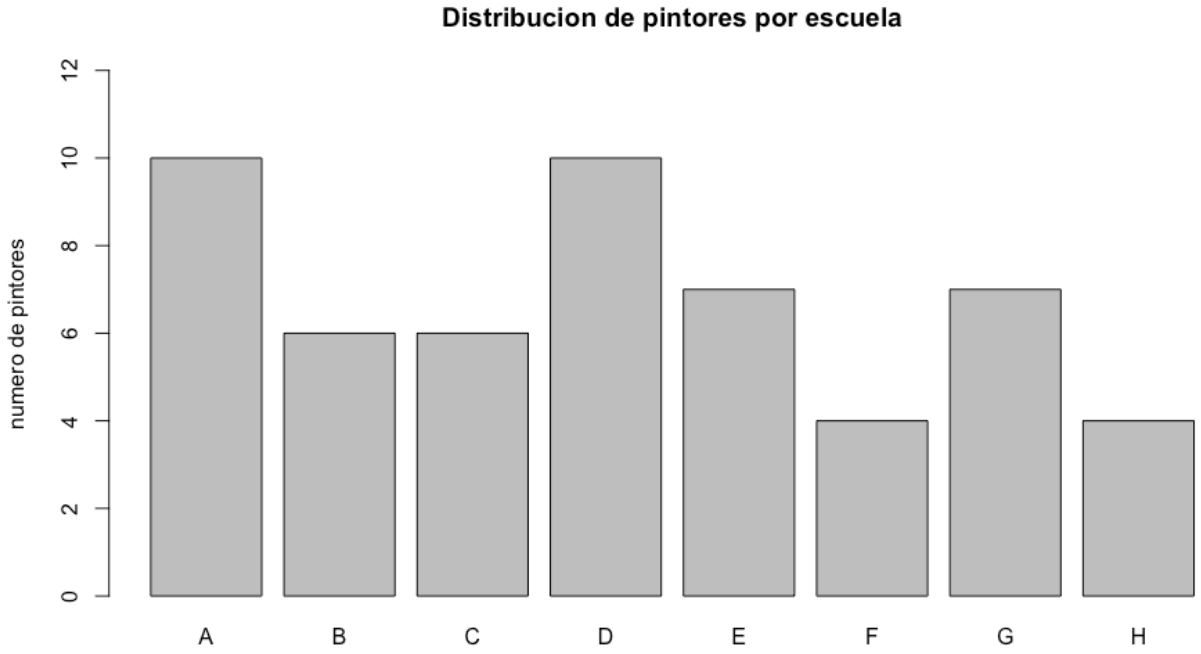
GRAFICANDO FRECUENCIAS

Usamos un **gráfico de barras (`barplot`)**

```
>
> frec_painters <- table(painters$School)
> barplot(frec_painters)
>
>
>
>
>
>
>
>
>
>
>
>
>
>
```

GRAFICANDO FRECUENCIAS

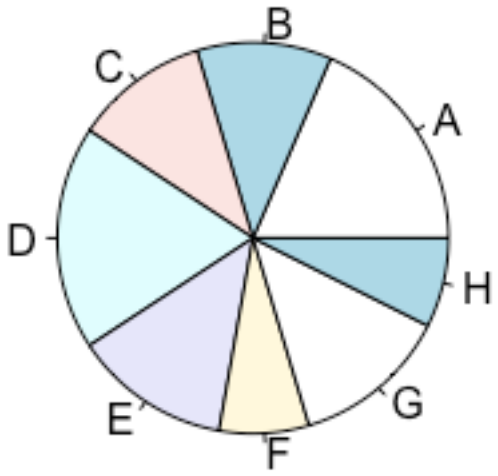
```
> barplot(frec_painters, ylab="numero de pintores", ylim=c(0,12),
main="Distribucion de pintores por escuela")
```

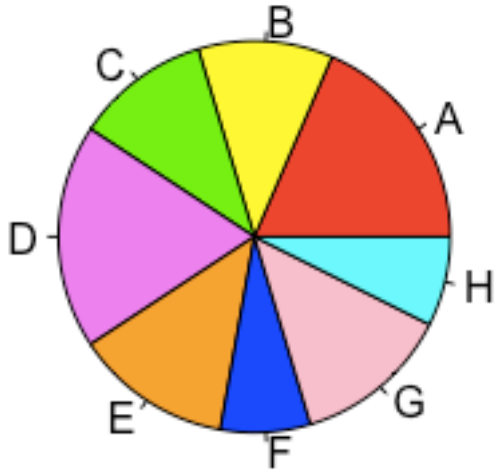


GRAFICANDO FRECUENCIAS

```
> pie(frec_painters)
```

```
> colores <- c("red", "yellow", "green", "violet",
"orange", "blue", "pink", "cyan")
> pie(frec_painters, col=colores)
```





GRAFICANDO FRECUENCIAS

También podemos usar la función **`hist()`** para crear un **histograma**:

```
> hist(crabs$FL, right=FALSE)
```

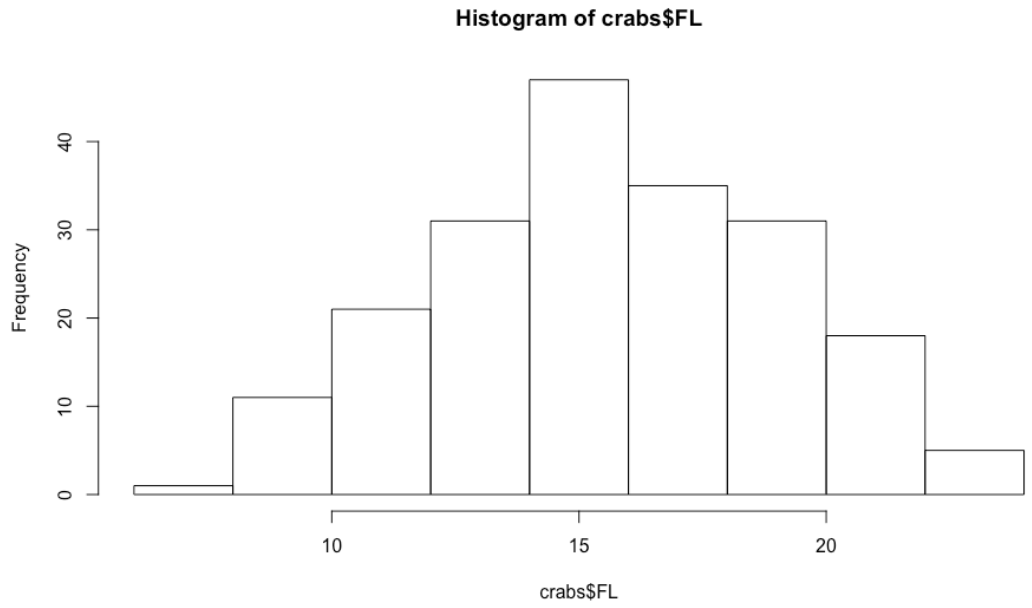
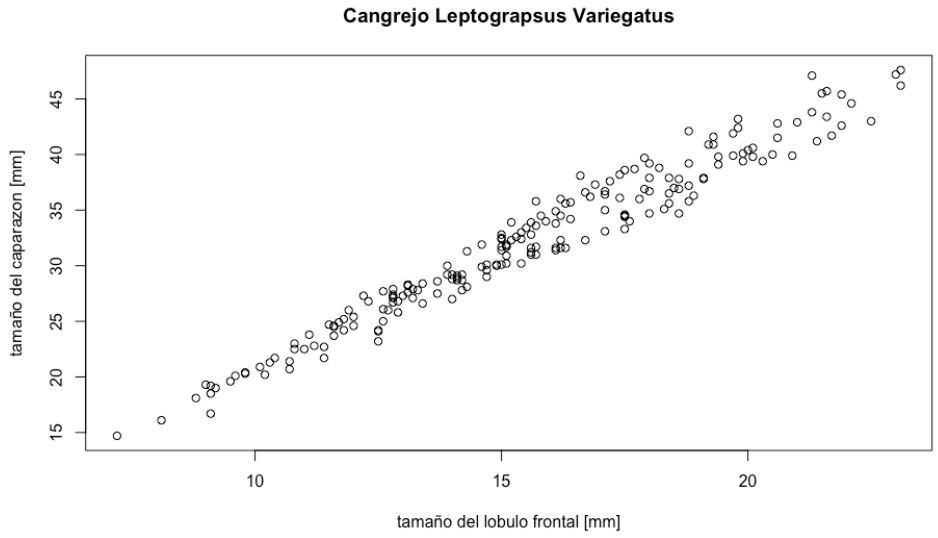


GRÁFICO DE DISPERSIÓN (SCATTER)

Para ver la relación entre dos variables numéricas, podemos usar un **gráfico de dispersión**:

```
> plot(crabs$FL, crabs$CL, xlab="tamaño del lobulo frontal [mm]",
ylab="tamaño del caparazon [mm]", main="Cangrejo Leptograpsus Var
iegatus")
```



ACTIVIDAD DE PREPARACIÓN

1. ¿En qué se diferencia un **vector** de un **factor**? ¿Para qué sirve cada tipo de estructura? Explique con ejemplos concretos.

2. ¿En qué se diferencia un gráfico generado con la función **plot**, con otro generado con la función **barplot**? ?Y con la función **hist**? ¿En qué tipo de problemas usaría cada uno? Explique con ejemplos concretos.