

## GUÍA Nro. 4

### Estadísticos de Orden - Gráficos Cuantil-Cuantil

---

#### Ejercicio 1:

a) La función `sort` en su forma más simple toma un vector como argumento y devuelve el vector ordenado. El vector a ser ordenado puede ser numérico o carácter y si hay un atributo de nombres éstos son conservados.

```
> datos <- c(0.2, 3.4, 2.8, 4.5, 3.3, 2.8, 2.8, 3.3)
> names(datos) <- c("b", "d", "g", "l", "a", "r", "d", "i")
```

Recuerde: para generar un vector en el que cada componente tiene un nombre se utiliza la función `names`.

```
> sort(datos)
  b   g   r   d   a   i   d   l
0.2 2.8 2.8 2.8 3.3 3.3 3.4 4.5
```

¿Qué ocurre con el orden de los empates?

b) El segundo argumento de `sort`, es el argumento, `partial`, especifica un índice o conjunto de índices que representan el orden del estadístico de orden garantizado a ser correcto en el resultado. Hallemos la mediana de un conjunto grande de datos de una  $N(0,1)$ .

```
> set.seed(14)
> x<-rnorm(100001)
> sort(x,partial=50001)[50001]
[1] -0.002208236
```

Verificamos el resultado

```
> median(x)
[1] -0.002208236
```

c) Pruebe dando un vector de índices al argumento `partial`.

```
sort(x,partial=c(1,50001))[c(1,50001)]
```

d) ¿Qué efecto produce la función `set.seed`?

### Ejercicio 2:

También se puede ordenar un vector mediante la función `sort.list(x)`. A continuación se muestra un ejemplo.

```
> x<-c(2,1,3,5,7,6,10,8,11,20,19,18,17)
> sort.list(x)
[1] 2 1 3 4 6 5 8 7 9 13 12 11 10
> x[sort.list(x)]
[1] 1 2 3 5 6 7 8 10 11 17 18 19 20
```

a)

- ¿Qué devuelve `sort.list(x)`?
- ¿Qué devuelve `x[sort.list(x)]`?
- Explique cómo se obtiene el vector ordenado y construya su propio ejemplo.
- ¿Qué devuelve `x[sort.list(-x)]`?
- ¿Qué devuelve `x[sort.list(x)==(length(x)+1)/2]`?

b) Halle la mediana, los cuartos y los cuartiles de `x` utilizando la función `sort.list`.

c) Repita b) para un conjunto de 20 datos.

d) Construya una matriz `X` de 5 filas y 3 columnas y un vector `y` de longitud 5. Ordene el vector `y` en orden creciente y reordene las filas de la matriz `X` de manera que se mantenga la correspondencia original con los valores de `y`.

**Sugerencia:** Utilice el siguiente ejemplo del help como guía

```
# sort a matrix according to its 1st column
mat[ sort.list(mat[,1]), ]
```

### Ejercicio 3: Índices y valores de un vector.

a) Construya un vector con nombres

```
> set.seed(14)
> x1<-round(rnorm(10),1)

> names(x1)<-letters[1:10]
> x1
  a  b c d  e  f  g h  i  j
-1.9 0.2 0 1 0.4 0.7 -2.5 1 -0.6 -0.6
```

y estudie qué realizan las siguientes instrucciones:

```
names(x1)[seq(along=x1)[x1== -2.5]]
names(x1)[x1== -2.5]
```

- b) Halle los índices que corresponden a valores mayores que 0.5 en el vector "x1".
- c) Halle las componentes del vector "x1" que son mayores que 3.
- d) Para los datos dados en "hills", halle donde está la carrera en la que se trepa 2100 pies. Sugerencias:

```
> library(MASS)
> help("hills")
> colinas <-hills
mire que objeto ha creado en Object Explorer
utilice la función row.names.
```

**Ejercicio 4:** El orden (o rango) de cada componente de un vector puede obtenerse mediante la función `rank`.

- a) Compare el tratamiento que da cada una de las funciones, `sort.list` y `rank` a los empates.

```
> x <- c(5,2,8,5)
> sort.list(x)
[1] 2 1 4 3
> rank(x)
[1] 2.5 1.0 4.0 2.5
```

- b) La siguiente es una forma de obtener un vector ordenado en forma decreciente, obtenga el mismo resultado de otra manera.

```
> rev(x)
[1] 5 8 2 5
> rev(sort(x))
[1] 8 5 5 2
```

**Ejercicio 5:** ORDENACIÓN POR VARIAS CLAVES

- a) La función `order` toma una cantidad arbitraria de argumentos y devuelve un vector de índices que ordena el primer argumento en orden creciente, dentro de los empates del primer arg. se ordena el segundo y así siguiendo. Por ejemplo sirve para ordenar a los empleados por edad y dentro de la edad por salario. Vea que obtiene mediante el siguiente ejemplo.

```
age <- c(25,30,28,32,30,30,25,25,32,25,30)
salary <- c(120,400,500,350,150,170,130,130,300,400,550)
No <- c(15512,14672,12330,18285,18232, 17625,
```

```
42324,53421,78541, 98073,65747)
ord <- order(age,salary)

cbind(age[ord],salary[ord],No[ord])
aaa <- cbind(age[ord],salary[ord],No[ord])
dimnames(aaa)<-
list(letters[1:11],c("edad","salario","No"))

data.class(aaa)
aaa*aaa
aaa%*%t(aaa)

aaa<-data.frame(age=age[ord],sal=salary[ord],No=No[ord])
data.class(aaa)
aaa*aaa
aaa%*%t(aaa)
```

b) Estudie qué se obtiene mediante la instrucción siguiente

```
state.name[rev(order(state.x77[,"Area"]))][1:10]
```

c) Con referencia a `state.x77` obtenga los 10 estados con mayor porcentaje de graduados en la escuela secundaria ("`HS Grad`") y dichos porcentajes.

d) Obtenga los 10 estados con mayor porcentaje de analfabetos y dentro de cada porcentaje ordene de acuerdo al porcentaje de asesinatos.

e) Ídem d) con el menor porcentaje de analfabetos y compare.

**Ejercicio 6** Las funciones `sort`, `sort.list`, `rank` y `order` tienen un argumento `na.last` cuyo valor determina el manejo de los valores faltantes. Indique cuáles son esos valores y qué ocurre en cada caso.

**Ejercicio 7:**

A partir de la siguiente definición general de cuantil para un conjunto de  $n$  datos:

**$x_p$  es el  $p$ -ésimo cuantil si**

$$P(X < x_p) \leq p \text{ y } P(X > x_p) \leq 1 - p$$

O sea, **la Proporción(datos <  $x_p$ )  $\leq p$  y la Proporción(datos >  $x_p$ )  $\leq 1 - p$**

a) muestre que el dato en la posición  $i$ ,  $x_{(i)}$ , es el  $p$ -cuantil para todo  $p$  que satisface

$$\frac{i-1}{n} \leq p \leq \frac{i}{n}$$

b) La definición tradicional ubica al cuartil en la posición

$$i = (n+1) / 4$$

es decir que  $p = \frac{1}{4}$  y  $i = (n+1) p$ . Muestre que esta definición satisface la condición dada en a).

c) Del `help` de la función `quantile` del S-PLUS obtenemos

The algorithm linearly interpolates between order statistics of  $x$ , assuming that the  $i$ th order statistic is the  $(i-1)/(\text{length}(x)-1)$  quantile.

es decir que asigna al dato en la posición  $i$  el cuartil

$$p = (i - 1) / (n-1)$$

de esta manera obtiene valores entre cero y uno para los cuantiles de un conjunto de datos. Muestre que esta asignación satisface la condición dada en a).

d) Muestre que la mediana y los cuartos satisfacen la condición dada en a).

### Ejercicio 8

Del `help` de la función `ppoints` del S-PLUS obtenemos:

```
ppoints(n, a=0.5)
```

#### VALUE:

vector of length  $n$  containing the values  $(1:n - a) / (n+1-2*a)$ .

La función `ppoints(n, a=0.5)` da un vector de probabilidades (p-points).

a) Verifique que si esas probabilidades son asignadas a los estadísticos de orden 1, 2, 3, ...,  $n$  se satisfacen las condiciones dadas en el ejercicio 7 a), cuando  $0 \leq a \leq 1$ .

b) Analice las siguientes instrucciones

```
> n <- 12
```

```
> ppoints(n, 3/8) # hemos cambiado el valor por defecto
```

```
[1] 0.05102041 0.13265306 0.21428571 0.29591837 0.37755102
     0.45918367
[7] 0.54081633 0.62244898 0.70408163 0.78571429 0.86734694
     0.94897959
```

Observe que se obtiene lo mismo con:

```
> a <- 3/8
> (1:n - a)/(n + 1 - 2 * a)
[1] 0.05102041 0.13265306 0.21428571 0.29591837 0.37755102
     0.45918367
[7] 0.54081633 0.62244898 0.70408163 0.78571429 0.86734694
     0.94897959

> ppoints(n) # por defecto a=0.5
[1] 0.04166667 0.12500000 0.20833333 0.29166667 0.37500000
     0.45833333
[7] 0.54166667 0.62500000 0.70833333 0.79166667 0.87500000
     0.95833333
```

Observe que se obtiene lo mismo con:

```
> a <- 1/2
> (1:n - a)/(n + 1 - 2 * a)
[1] 0.04166667 0.12500000 0.20833333 0.29166667 0.37500000
     0.45833333
[7] 0.54166667 0.62500000 0.70833333 0.79166667 0.87500000
     0.95833333
```

### Ejercicio 9:

a) Vea qué obtiene al aplicar las siguientes funciones a `x` dado por

```
x <- seq(1,120,0.5):
```

```
summary, max, min, range, quantile
```

b) Vea que ocurre cuando hay un dato faltante,

```
> y<- c(x[1:10], NA ,x[11:239])
con
> quantile(y, c(0.05,0.10,0.90,0.95))
> quantile(y, c(0.05,0.10,0.90,0.95),na.rm=TRUE)
```

### Ejercicio 10

a) Estudie que realiza la función `qqplot` en los siguientes ejemplos

```
> bb <- qqplot(1:10, 20:11)
> bb
$x:
 [1] 1 2 3 4 5 6 7 8 9 10

$y:
 [1] 11 12 13 14 15 16 17 18 19 20
```

```
> bb <- qqplot(1:10, 1:20)
> bb
$x:
 [1] 1 2 3 4 5 6 7 8 9 10

$y:
 [1] 1.000000 3.111111 5.222222 7.333333 9.444444
     11.555556 13.666667
 [8] 15.777778 17.888889 20.000000
```

b) Compare la distribución de dos conjuntos de datos, cada uno correspondiente a una variable, mediante el gráfico cuantil-cuantil

```
> qqplot(x,y)
```

### Ejercicio 11.

a) Verifique que la función `qqnorm` toma los cuantiles definidos por la función `ppoints` con el valor por defecto `a=0.5`

```
> aa <- qqnorm(1:n, plot = F)
> aa
$x:
 [1] -1.7316644 -1.1503494 -0.8122178 -0.5485223 -0.3186394 -0.1046335
 [7] 0.1046335 0.3186394 0.5485223 0.8122178 1.1503494 1.7316644

$y:
 [1] 1 2 3 4 5 6 7 8 9 10 11 12

> qqnorm(ppoints(1:n))
 [1] -1.7316644 -1.1503494 -0.8122178 -0.5485223 -0.3186394 -0.1046335
 [7] 0.1046335 0.3186394 0.5485223 0.8122178 1.1503494 1.7316644
```

Construya el gráfico `plot(aa$x,aa$y)` y compare `qqnorm`

```
plot(aa$x,aa$y)
```

```
qqnorm(1:n)
```

c) Compare la distribución de cada uno de los conjuntos de datos anteriores con la distribución normal utilizando `qqnorm` y `qqline`.

d) También puede comparar la distribución de un conjunto de datos con la de cualquier distribución. Por ejemplo con la t con 2 grados de libertad.

```
plot(qt(ppoints(x),2),sort(x))
```

Pruebe con varias distribuciones, puede utilizar por ejemplo `qchisq`, `qexp`, `qf` ó `qgamma`.

**Ejercicio 12:** Analice cada una de las variables que aparecen en el conjunto de datos dados en "hills"

En R

```
> library(MASS)  
> help(hills)
```

Para cada una de ellas

- halle medidas resumen
- observe su distribución (histogramas, tallo-hoja)
- compare con distribuciones teóricas