Sala de Entrenamiento 2: Básicos de R (Parte II)

Análisis de Métodos Cuantitativos
Semana 1
Diplomados Online

Septiembre 2018

R tiene una gran variedad de funciones, tales como suma, máximo y mínimo que calculan la suma, máximo y mínimo de todos los elementos del vector, respectivamente. La mayoría de las funciones en R se pueden aplicar a vectores, así como a otros tipos de objetos tales como como matrices y tablas de datos agrupados.

Algunas de las funciones, como suma, máximo y mínimo, producen sólo un número como resultado. Sin embargo, la mayoría de las funciones aplican sus operaciones a cada elemento del vector, y producen otro vector del mismo tamaño. Para obtener información sobre una función R, simplemente escriba un signo de interrogación seguido del nombre de la función, o su equivalente, utilice la función de ayuda.

Por ejemplo, log es el logaritmo natural. Para conocer todas sus caracterśticas escriba el comando (para ello necesitará acceso a internet):

```
> help(log)
```

Si aplicamos la función log al conjunto de números del 1 al 10 elevados al cuadrado, se tiene :

```
> log((1:10)^2)
```

```
 [1] 0.000000 1.386294 2.197225 2.772589 3.218876 3.583519 3.891820 4.158883
 [9] 4.394449 4.605170
```

Nótese que por convención común, si no se especifica una base para el log, entonces es el logaritmo natural, es decir, el logaritmo neperiano. Cualquier otra base debe ser especificada explícitamente. Por ejmplo, si quisiera aplicarse logaritmo base 10 a los números del 2 al 5, en R se debe hacer de la forma siguiente.

```
> log(10^(2:5),base=10)
```

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

1. Matrices y Arreglos

Una matriz en matemáticas es sólo un arreglo bidimensional de números. Las matrices se utilizan para muchos propósitos en la estadística teórica y práctica. Las matrices y también los arreglos de mayor dimensión se utilizan también para propósitos más simples, principalmente para representar tablas.

En R, la noción de matriz se extiende a elementos de cualquier tipo, por lo que podría tener, por ejemplo, una matriz de cadenas de caracteres. Las matrices y arreglos se representan como vectores con dimensiones:

```
> x<-1:12
> dim(x)<-c(3,4)
> x
```

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

La función *dim* establece o cambia el atributo de dimensión de x, haciendo que R trate el vector de 12 números como una matriz de 3×4. Observe que el almacenamiento se hace inuciando en la columna mayor; es decir, los elementos de la primera columna son seguidos por los de la segunda, etc.

Otra manera conveniente de crear matrices, es con el úso de la función *matrix*

```
> matrix(1:12,nrow=3,byrow=T)
```

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

Aquí *byrow* significa que el arreglo se hará por fila en lugar de columnas como en el caso anterior.

Tambien podemos darle nombre a la columnas y filas con los comandas *col.names* y *row.names*, respectivamente.

```
> x <- matrix(1:12,nrow=3,byrow=T)
> row.names(x)<-letters[1:3]
> x
```

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

Si se desea la matriz transpuesta, hacemos

```
> t(x)

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

Podemos "pegar"vectores en forma de columna o fila, con *cbind* y *rbind*, respectivamente.

```
> cbind(A=1:4,B=5:8,C=9:12)

     A B  C
[1,] 1 5  9
[2,] 2 6 10
[3,] 3 7 11
[4,] 4 8 12

> rbind(A=1:4,B=5:8,C=9:12)

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

2. Factores

Es común en los datos estadísticos variables categóricas, que indican alguna subdivisión de los datos, como la clase social, el diagnóstico primario, etapa de un tumor, etapa de Tanner de la pubertad, etc. Tales variables pueden ser especidicadas en R como *factores*.

Hay análisis en los que es esencial que R sea capaz de distinguir entre códigos categóricos y variables cuyos valores tienen una relación directa significado numérico

La terminología es que un factor tiene un conjunto de niveles - digamos cuatro niveles de concreción. Internamente, un factor de cuatro niveles consta de dos posiciones: a) un vector de números enteros entre 1 y 4, y b) un vector de caracteres de longitud 4 que contenga cadenas que describan los cuatro niveles. Veamos un ejemplo:

```
> dolor<- c(0,3,2,2,1)
> fdolor<- factor(dolor,levels=0:3)
> levels(fdolor) <- c("Ninguno","Bajo","Medio","Severo")
> fdolor
```

```
[1] Ninguno Severo  Medio   Medio   Bajo
Levels: Ninguno Bajo Medio Severo
```

La función *as.numeric* extrae la codificación numérica como números del 1 al 4 y *levels* extrae los nombres de los niveles. Observe que la codificación de entrada original en términos de números del 0 al 3 ha desaparecido; la representación interna de un factor siempre utiliza números que empiezan por 1.

```
> as.numeric(fdolor)

[1] 1 4 3 3 2

> levels(fdolor)

[1] "Ninguno" "Bajo"    "Medio"   "Severo"
```

3. Datos Agrupados y Dataframe

La manera natural de almacenar datos agrupados en un dataframe es tener los datos en un mismos vector y paralelos a éste se tiene un factor que indica qué datos son de qué grupo. Considere, por ejemplo, el siguiente conjunto de datos sobre el gasto energético de las mujeres delgadas y obesas, que lo encontramos cargando en R la liberia *ISwR* (Introductory Statistics with R)[1], y llamando los datos *energy*

```
> library("ISwR")
> energy

   expend stature
1    9.21   obese
2    7.53    lean
3    7.48    lean
4    8.08    lean
5    8.09    lean
6   10.15    lean
7    8.40    lean
8   10.88    lean
9    6.13    lean
10   7.90    lean
11  11.51   obese
12  12.79   obese
```

[1]Para trabajar con esta libería, primero debemos verificar si ya se encuentra previamente instalada en R, ,esto lo hacemos pulsando *Packages* en la consola de Files, Plosts, Packages, Help, Viewer, para luego verificar si se encuentra. En caso de no ser así, debemos en la consola de R tipear la función *install.packages("ISwR")*, con lo que comenzara el proceso de instalación. En esta liberia se encuentran varios grupos de datos que estaremos utilizando a lo largo del curso

```
13   7.05    lean
14  11.85   obese
15   9.97   obese
16   7.48    lean
17   8.79   obese
18   9.69   obese
19   9.68   obese
20   7.58    lean
21   9.19   obese
22   8.11    lean
```

A veces es deseable tener datos en un vector separado para cada grupo. Afortunadamente, es fácil extraerlos del dataframe. Por ejemplo, si quisieramos extraer el gasto de las personas obesas, lo hacemos de las siguiente forma:

```
> gasto.obesas<-energy$expend[energy$stature=="obese"]
> gasto.obesas

[1]  9.21 11.51 12.79 11.85  9.97  8.79  9.69  9.68  9.19
```

Si queremos presentar los dos vectores simultaneamente pero separados, hacemos:

```
> l<-split(energy$expend, energy$stature)
> l

$lean
 [1]  7.53  7.48  8.08  8.09 10.15  8.40 10.88  6.13  7.90  7.05  7.48  7.58
[13]  8.11

$obese
[1]  9.21 11.51 12.79 11.85  9.97  8.79  9.69  9.68  9.19
```

4. Aplicación Elemental: El Proceso de Bernoulli

Comenzamos nuestro estudio de probabilidad con el proceso de lanzar una moneda. La función que lanza una moneda en R se llama *rbinom*. Para lanzar una moneda justa 10 veces, usamos este comando:

```
> rbinom(10,1,1/2)

 [1] 0 0 0 1 1 1 0 0 0 1
```

Esta función tiene tres parámetros. El primer parámetro es el número de lanzamientos. El segundo parámetro selecciona el caso especial de esta función que implementa el proceso de Bernoulli. El proceso general se explicará luego. El tercer parámetro es el sesgo de la moneda. Cada vez que

llame a esta función obtendrá otra secuencia de 0 y 1. La función *rbinom* es una de las funciones R para generar números aleatorios. Veremos muchos más más más adelante en el curso.

Si utilizamos operaciones booleanas podemos determinar si se tiene caras o sellos en una realización del experimento de lanzar una moneda 10 veces. Para ello hacemos

```
> rbinom(10,1,1/2)==1

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

obteniendo un vector de valores booleanos en lugar de ceros y unos, con lo que estamos solicitando a R que nos diga si es cierto que en determinada entrada del vector se tiene una cara. Una vez realizado esto, podemos utilizar la siguiente instrucción para que R nos indique en qué posición de a obtenido una cara, así

```
> (1:10)[rbinom(10,1,1/2)==1]

[1] 3 5
```

Finalmente, cuando una expresión resulta especialmente complicada, podemos invocarla a través de una función en la cual sólo sea necesario especificar los parámetros. Por ejemplo, si quisieramos modelar en experimento de lanzar una moneda balanceada o no n veces, pudiesemos hacerlo a través de una función en la cual sólo se necesite especificar el número de lanzamientos y la probabilidad de obtener una cara, de la forma siguiente:

```
> caras<-function(n,p)(1:n)[rbinom(n,1,p)==1]
```

así, llamamos a la función caras para obtener la posición que ocupan las caras en una vector, que por ejemplo, recoge los resultados de 10 lanzamientos con una probabilidad de obtener una cara de 3/4,

```
> caras(10,3/4)

[1] 1 3 5 6
```

Con estos elementos básicos de R, ya podemos empezar a inspeccionar todas los elementos que no spermite hacer R, e iremos desarrollando otras funciones a medida que se avance en el curso. De igual manera, en la web se encuntran suficientes manuales de R que pueden consultar.