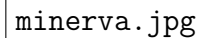


**UNIVERSIDAD DE EL SALVADOR  
FACULTAD MULTIDISCIPLINARIA DE OCCIDENTE  
DEPARTAMENTO DE MATEMÁTICA**



minerva.jpg

**Licenciatura en Estadística**

**Control Estadístico del Paquete R**

**”UNIDAD UNO”**

**Alumna:  
Martha Yoana Medina Sánchez**

**Fecha de elaboración  
Santa Ana - 31 de octubre de 2015**

## Práctica 03 - Tipos de objetos: factores, listas y hojas de datos, operadores y funciones que operan sobre ellos.

### 1. FACTORES NOMINALES Y ORDINALES.

#### FACTORES NOMINALES.

Ejemplo 1: Variables sexo (categórica) y edad en una muestra de 7 alumnos del curso.

```
# Supongamos que se obtuvieron los siguientes datos:
```

```
sexo <- c("M", "F", "F", "M", "F", "F", "M"); sexo
```

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

```
edad <- c(19, 20, 19, 22, 20, 21, 19); edad
```

```
## [1] 19 20 19 22 20 21 19
```

```
# Podemos construir un factor con los niveles o categorías de sexo
```

```
FactorSexo = factor(sexo); FactorSexo
```

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

```
## Levels: F M
```

```
# Se pueden ver los niveles o categorías del factor con:
```

```
levels(FactorSexo)
```

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

```
# Crear una tabla que contenga la media muestral por categoría de sexo
```

```
 #(nivel del factor):
```

```
mediaEdad <- tapply(edad, FactorSexo, mean); mediaEdad
```

```
## F M
```

```
## 20 20
```

```
# Note que el primer argumento debe ser un vector, que es del cual se  
# encontrará las medidas de resumen; el segundo es el factor que se está  
# considerando, mientras que en el tercero se especifica la medida de interés,  
# solamente puede hacerse una medida a la vez.
```

¿Qué tipo de objeto es la variable mediaEdad?:

```
is.vector(mediaEdad);  
  
## [1] FALSE  
  
is.matrix(mediaEdad);  
  
## [1] FALSE  
  
is.list(mediaEdad);  
  
## [1] FALSE  
  
is.table(mediaEdad);  
  
## [1] FALSE  
  
is.array(mediaEdad)  
  
## [1] TRUE
```

## FACTORES ORDINALES

Los niveles de los factores se almacenan en orden alfabético, o en el orden en que se especificaron en la función `factor()` si el usuario lo especifica.

```
factor(2)  
  
## [1] 2  
## Levels: 2
```

A veces existe una ordenación natural en los niveles de un factor, orden que deseamos tener en cuenta en los análisis estadísticos. La función `ordered()` crea este tipo de factores y su uso es idéntico al de la función `factor()`. Los factores creados por la función `factor()` los denominaremos nominales o simplemente factores cuando no haya lugar a confusión, y los creados por la función `ordered()` los denominaremos ordinales. En la mayoría de los casos la única diferencia entre ambos tipos de factores consiste en que los ordinales se imprimen indicando el orden de los niveles. Sin embargo, los contrastes generados por los dos tipos de factores para ajustar Modelos lineales, son diferentes.

```
ordered(2)  
  
## [1] 2  
## Levels: 2
```

## 2. CREACIÓN Y MANEJO DE LISTAS.

Ejemplo 1: Crear una Lista con cuatro componentes.

```
lista1<-list(padre="Pedro", madre="MarÃa", no.hijos=3, edad.hijos=c(4,7,9))
lista1

## $padre
## [1] "Pedro"
##
## $madre
## [1] "MarÃa"
##
## $no.hijos
## [1] 3
##
## $edad.hijos
## [1] 4 7 9
```

Revise algunos tipos como:

```
is.matrix(lista1); is.vector(lista1$edad.hijos)

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

Ejemplo 2: Acceso a las componentes de una lista:

```
lista1[1]

## $padre
## [1] "Pedro"

#accede a la componente como una lista (con etiqueta y valor)
```

```
lista1["padre"]

## $padre
## [1] "Pedro"

#el acceso es igual que con lista1[1]
```

```
lista1[[2]]

## [1] "MarÃa"

#accede al valor o valores de la componente segunda pero no muestra el nombre de
#la componente.
```

```
lista1["madre"]  
  
## $madre  
## [1] "MarÃa"  
  
#el acceso es igual que con lista1[[1]]
```

Ejemplo 3: Acceso a los elementos de la cuarta componente:

```
lista1[[4]][2]  
  
## [1] 7  
  
#Se indica el elemento a ingresar en el segundo corchete.
```

Ejemplo 4: Acceso de las componentes de una lista por su nombre:

```
lista1$"padre"  
  
## [1] "Pedro"
```

similar a

```
lista1["padre"]  
  
## $padre  
## [1] "Pedro"
```

Forma general:

Por ejemplo:

```
lista1$padre  
  
## [1] "Pedro"
```

equivale a

```
lista1[[1]]  
  
## [1] "Pedro"
```

y

```
lista1$edad.hijos[2]  
  
## [1] 7
```

equivale a

```
lista1[[4]][2]

## [1] 7
```

Ejemplo 5: Utilizar el nombre de la componente como Índice:

```
lista1[["madre"]]

## [1] "MarÃa"
```

se puede ver que equivale a

```
lista1$padre

## [1] "Pedro"
```

Tambi n es  til la forma:

```
x <- "madre"; lista1[x]

## $madre
## [1] "MarÃa"
```

Ejemplo 6: Creaci n de una sublista de una lista existente :

```
subLista <- lista1[4]; subLista

## $edad.hijos
## [1] 4 7 9
```

Ejemplo 7: Ampliaci n de una lista : *por ejemplo, la lista lista1 tiene 4 componentes y se le puede agregar una quinta componente con :*

```
lista1[5] <- list(sexo.hijos=c("F", "M", "F")); lista1

## $padre
## [1] "Pedro"
##
## $madre
## [1] "MarÃa"
##
## $no.hijos
## [1] 3
##
```

```
## $edad.hijos
## [1] 4 7 9
##
## [[5]]
## [1] "F" "M" "F"
```

Observe que no aparece el nombre del objeto agregado, pero usted puede modificar la estructura de la lista `lista1` con:

```
lista1 <- edit(lista1)
```

```
lista1[5] <- list(sexo.hijos=c("F", "M", "F")); lista1

## $padre
## [1] "Pedro"
##
## $madre
## [1] "MarÃa"
##
## $no.hijos
## [1] 3
##
## $edad.hijos
## [1] 4 7 9
##
## $sexo.hijos
## [1] "F" "M" "F"
```

**Nota:** Se puede aplicar la función `data.entry()` para modificar la estructura de una lista.

Ejemplo 8: Funciones que devuelven una lista.

Las funciones y expresiones de R devuelven un objeto como resultado, por tanto, si deben devolver varios objetos, previsiblemente de diferentes tipos, la forma usual es una lista con nombres. Por ejemplo, la función `eigen()` que calcula los autovalores y autovectores de una matriz simétrica.

```
eigen(40)

## $values
## [1] 40
##
## $vectors
##      [,1]
## [1,]    1
```

Ejecute las siguientes instrucciones:

```
S <- matrix(c(3, -sqrt(2), -sqrt(2), 2), nrow=2, ncol=2); S

##           [,1]      [,2]
## [1,]  3.000000 -1.414214
## [2,] -1.414214  2.000000

autovS <- eigen(S); autovS

## $values
## [1] 4 1
##
## $vectors
##           [,1]      [,2]
## [1,] -0.8164966 -0.5773503
## [2,]  0.5773503 -0.8164966
```

Observe que la función `eigen()` retorna una listade dos componentes, donde la componente

```
autovS$values

## [1] 4 1
```

es el vector de autovalores de `S` y la componente es la matriz de los

```
autovS$vectors

##           [,1]      [,2]
## [1,] -0.8164966 -0.5773503
## [2,]  0.5773503 -0.8164966
```

correspondientes autovectores. Si quisiéramos almacenar sólo los autovalores de podemos hacer lo siguiente:

```
evals <- eigen(S)$values; evals

## [1] 4 1
```

Ejemplo 9: Crear una matriz dando nombres a las filas y columnas

```
Notas <- matrix(c(2, 5, 7, 6, 8, 2, 4, 9, 10), ncol=3,
dimnames=list(c("Matematica","Algebra","Geometria"),
c("Juan","Jose","Rene"))); Notas

##           Juan Jose Rene
## Matematica    2    6    4
## Algebra       5    8    9
## Geometria     7    2   10

# Los nombres se dan primero para filas y luego para columnas.
```



### 3. CREACIÓN Y MANEJO DE HOJAS DE DATOS (DATA FRAME).

Una hoja de datos (data frame) es una lista que pertenece a la clase data.frame. Un data.frame puede considerarse como una matriz de datos. Hay restricciones en las listas que pueden pertenecer a esta clase, en particular:

- Los componentes deben ser vectores (numéricos, cadenas de caracteres, o lógico), factores, matrices numéricas, listas u otras hojas de datos.
- Las matrices, listas, y hojas de datos contribuyen a la nueva hoja de datos con tantas variables como columnas, elementos o variables posean, respectivamente.
- Los vectores numéricos y los factores se incluyensin modificar, los vectores no numéricos se fuerzan a factores cuyos niveles son los únicos valores que aparecen en el vector.
- Los vectores que constituyen la hoja de datos deben tener todos la misma longitud, y las matrices deben tener el mismo tamaño de filas.

Las hojas de datos pueden interpretarse, en muchos sentidos, como matrices cuyas columnas pueden tener diferentes modos y atributos. Pueden imprimirse en forma matricial y se pueden extraer sus filas o columnas mediante la indexación de matrices. En una hoja de datos cada columna corresponde a una variable y cada fila a un elemento del conjunto de observaciones.

Ejemplo 1: Creación de un data frame teniendo como columnas tres vectores:

#### En primer lugar generamos los tres vectores

El primer vector tendrá 20 elementos que se obtienen con reemplazamiento de una muestra aleatoria de valores lógicos.

```
log <- sample(c(TRUE, FALSE), size = 20, replace = T); log

## [1] FALSE TRUE FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE FALSE
## [12] FALSE FALSE TRUE TRUE FALSE FALSE FALSE FALSE TRUE

# Note que puede usar T en lugar de TRUE y F en lugar de FALSE.
```

El segundo vector tendrá 20 elementos de valores complejos cuya parte real proviene de una distribución Normal estándar y cuya parte imaginaria lo hace de una distribución Uniforme(0,1)

```
comp <- rnorm(20) + runif(20) * (1i); comp

## [1] -1.1724701+0.7197209i 1.4619989+0.5651870i 0.5020940+0.9616957i
## [4] 1.8654412+0.5409153i 0.0141955+0.2804807i 0.1516880+0.1775594i
## [7] -0.8354331+0.6953587i -0.2991425+0.9896466i -2.1139928+0.3545345i
## [10] -0.5736937+0.6745475i -0.8463461+0.1960985i 1.0436496+0.6496639i
## [13] 1.2660945+0.0935689i 0.2178955+0.5191883i -0.5353158+0.4409293i
## [16] -0.3413716+0.4909402i -1.9608042+0.6878372i -0.0915159+0.4199387i
## [19] -0.9123237+0.7862988i 1.1656568+0.8172013i
```

El tercer vector tendrá 20 elementos de una distribución Normal estándar

```
num <- rnorm(20, mean=0, sd=1); num

## [1] 0.659302973 1.637905616 0.046653312 0.637328733 0.075350420
## [6] 1.431484283 1.198338310 1.665349432 -2.223074957 1.825974121
## [11] 0.585395333 1.390275090 -1.725799792 0.912381976 -0.024667971
## [16] -0.817245733 0.316233864 0.006056304 2.027990139 1.206234720
```

**Crear un data frame compuesto por los tres vectores anteriores**

```
df1 <- data.frame(log, comp, num); df1

##      log      comp      num
## 1 FALSE -1.1724701+0.7197209i 0.659302973
## 2  TRUE  1.4619989+0.5651870i 1.637905616
## 3 FALSE  0.5020940+0.9616957i 0.046653312
## 4 FALSE  1.8654412+0.5409153i 0.637328733
## 5 FALSE  0.0141955+0.2804807i 0.075350420
## 6 FALSE  0.1516880+0.1775594i 1.431484283
## 7  TRUE -0.8354331+0.6953587i 1.198338310
## 8 FALSE -0.2991425+0.9896466i 1.665349432
## 9  TRUE -2.1139928+0.3545345i -2.223074957
## 10 FALSE -0.5736937+0.6745475i 1.825974121
## 11 FALSE -0.8463461+0.1960985i 0.585395333
## 12 FALSE  1.0436496+0.6496639i 1.390275090
## 13 FALSE  1.2660945+0.0935689i -1.725799792
## 14  TRUE  0.2178955+0.5191883i 0.912381976
## 15  TRUE -0.5353158+0.4409293i -0.024667971
## 16 FALSE -0.3413716+0.4909402i -0.817245733
## 17 FALSE -1.9608042+0.6878372i 0.316233864
## 18 FALSE -0.0915159+0.4199387i 0.006056304
## 19 FALSE -0.9123237+0.7862988i 2.027990139
## 20  TRUE  1.1656568+0.8172013i 1.206234720
```

**Crear un vector de nombres de los tres vectores anteriores**

```
nombres <- c("logico", "complejo", "numerico")
```

**Define los nombres de las columnas del data frame asignándoles el vector nombres**

```
names(df1) <- nombres; df1

##      logico      complejo      numerico
## 1  FALSE -1.1724701+0.7197209i 0.659302973
## 2   TRUE  1.4619989+0.5651870i 1.637905616
```

```
## 3  FALSE  0.5020940+0.9616957i  0.046653312
## 4  FALSE  1.8654412+0.5409153i  0.637328733
## 5  FALSE  0.0141955+0.2804807i  0.075350420
## 6  FALSE  0.1516880+0.1775594i  1.431484283
## 7   TRUE -0.8354331+0.6953587i  1.198338310
## 8  FALSE -0.2991425+0.9896466i  1.665349432
## 9   TRUE -2.1139928+0.3545345i -2.223074957
## 10 FALSE -0.5736937+0.6745475i  1.825974121
## 11 FALSE -0.8463461+0.1960985i  0.585395333
## 12 FALSE  1.0436496+0.6496639i  1.390275090
## 13 FALSE  1.2660945+0.0935689i -1.725799792
## 14  TRUE  0.2178955+0.5191883i  0.912381976
## 15  TRUE -0.5353158+0.4409293i -0.024667971
## 16 FALSE -0.3413716+0.4909402i -0.817245733
## 17 FALSE -1.9608042+0.6878372i  0.316233864
## 18 FALSE -0.0915159+0.4199387i  0.006056304
## 19 FALSE -0.9123237+0.7862988i  2.027990139
## 20  TRUE  1.1656568+0.8172013i  1.206234720
```

Define los nombres de las filas del data frame asignándoles un vector de 20 elementos correspondientes a las 20 primeras letras del abecedario

```
row.names(df1) <- letters[1:20]; df1

##   logico      complejo      numerico
## a  FALSE -1.1724701+0.7197209i  0.659302973
## b   TRUE  1.4619989+0.5651870i  1.637905616
## c  FALSE  0.5020940+0.9616957i  0.046653312
## d  FALSE  1.8654412+0.5409153i  0.637328733
## e  FALSE  0.0141955+0.2804807i  0.075350420
## f  FALSE  0.1516880+0.1775594i  1.431484283
## g   TRUE -0.8354331+0.6953587i  1.198338310
## h  FALSE -0.2991425+0.9896466i  1.665349432
## i   TRUE -2.1139928+0.3545345i -2.223074957
## j  FALSE -0.5736937+0.6745475i  1.825974121
## k  FALSE -0.8463461+0.1960985i  0.585395333
## l  FALSE  1.0436496+0.6496639i  1.390275090
## m  FALSE  1.2660945+0.0935689i -1.725799792
## n   TRUE  0.2178955+0.5191883i  0.912381976
## o   TRUE -0.5353158+0.4409293i -0.024667971
## p  FALSE -0.3413716+0.4909402i -0.817245733
## q  FALSE -1.9608042+0.6878372i  0.316233864
## r  FALSE -0.0915159+0.4199387i  0.006056304
## s  FALSE -0.9123237+0.7862988i  2.027990139
## t   TRUE  1.1656568+0.8172013i  1.206234720
```

Ejemplo 2: Vamos a crear la siguiente hoja de datos que tiene 4 variables o columnas:

```
edad <- c(18, 21, 45, 54); edad

## [1] 18 21 45 54

datos <- matrix(c(150, 160, 180, 205, 65, 68, 65, 69), ncol=2, dimnames=list(c(),
c("Estatura", "Peso"))); datos

##      Estatura Peso
## [1,]      150    65
## [2,]      160    68
## [3,]      180    65
## [4,]      205    69

sexo <- c("F", "M", "M", "M"); sexo

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

hoja1 <- data.frame(Edad=edad, datos, Sexo=sexo); hoja1

##   Edad Estatura Peso Sexo
## 1   18      150    65    F
## 2   21      160    68    M
## 3   45      180    65    M
## 4   54      205    69    M
```

Para editar o agregar datos, o componentes utilice:

```
fix(hoja1)
```

**Nota:** Puede forzar que una lista, cuyos componentes cumplan las restricciones para ser una hoja de datos, realmente lo sea, mediante la función `as.data.frame()`

## ACCESO A LAS COMPONENTE O VARIABLES DE UNA HOJA DE DATOS.

Normalmente para acceder a la componente o variable Edad de la hoja de datos se utilizará la expresión

```
hoja1$Edad

## [1] 18 21 45 54
```

, pero existe una forma más sencilla, consiste en conectar la hoja de datos para que se pueda hacer referencia a sus componentes directamente por su nombre.

### Conexión de listas o hojas de datos.

La función `search()` busca y presenta qué hojas de datos, listas o bibliotecas han sido conectadas o desconectadas. Teclee `search()`

```
search()

## [1] ".GlobalEnv"      "package:knitr"    "package:stats"
## [4] "package:graphics" "package:grDevices" "package:utils"
## [7] "package:datasets" "package:methods"  "Autoloads"
## [10] "package:base"
```

La función `attach()` es la función que permite conectar en la trayectoria de búsqueda no sólo directorios, listas y hojas de datos, sino también otros tipos de objetos. Teclee `attach(hoja1)` y luego `search()`.

Luego puede acceder a las componentes por su nombre:

```
hoja1$Edad

## [1] 18 21 45 54

hoja1$Peso

## [1] 65 68 65 69

hoja1$peso+1;hoja1

## numeric(0)
##      Edad Estatura Peso Sexo
## 1     18      150   65    F
## 2     21      160   68    M
## 3     45      180   65    M
## 4     54      205   69    M
```

Posteriormente podrá desconectar el objeto utilizando la función `detach()`, utilizando como argumento el número de posición o, preferiblemente, su nombre. Teclee `detach(hoja1)` y compruebe que la hoja de datos ha sido eliminada de la trayectoria de búsqueda con `search()`.

Pruebe si puede acceder a una componente *añadiendo un nombre, por ejemplo,*

## TRABAJO CON HOJAS DE DATOS

Una metodología de trabajo para tratar diferentes problemas utilizando el mismo directorio de trabajo es la siguiente:

- Reúna todas las variables de un mismo problema en una hoja de datos y dé le un nombre apropiado e informativo;
- Para analizar un problema, conecte, mediante `attach()`, la hoja de datos correspondiente (en la posición 2) y utilice el directorio de trabajo (en la posición 1) para los cálculos y variables temporales;

- Antes de terminar un análisis, añada las variables que deba conservar a la hoja de datos utilizando la forma \$ para la asignación y desconecte la hoja de datos mediante detach();
- Para finalizar, elimine del directorio de trabajo las variables que no desee conservar, para mantenerlo lo más limpio posible.

De este modo podrá analizar diferentes problemas utilizando el mismo directorio, aunque todos ellos compartan variables denominadas x, y o z, por ejemplo.