PRACTICA_03

Nelson de Jesus Magaña Godinez

3/8/2022

Contents

1. FACTORES NOMINALES Y ORDINALES.

Un factor es un vector utilizado para especificar una clasificación discreta de los elementos de otro vector de igual longitud. En R existen factores nominales y factores ordinales. Los factores son útiles a la hora de querer hacer contrastes o de calcular medidas de resúmenes para variables numéricas en distintos niveles de una segunda variable la cual es no numérica

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 categorias 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 mediamuestral 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 encontrarán las medidas de resumen
```

¿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
```

La variable mediaEdad es un arreglo

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 ello se hizo explícitamente. 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 al ajustar Modelos lineales, son diferentes.

2. CREACIÓN Y MANEJO DE LISTAS.

Una lista es un objeto que contiene una colección ordenada de objetos de diferente tipo (vector,matriz, arreglo, función, o lista), conocidos como componentes. Se construye con la función list(), que tiene la forma general siguiente:

Lista <- list(nombre1 = objeto1, nombre2 = objeto2, . . . , nombren = objeton)

Si omite los nombres, las componentes sólo estarán numeradas. Las componentes pueden accederse por su número o posición, ya que siempre están numeradas, o también pueden referirse por su nombre, si lo tienen.

- *Ejemplo 1:* Crear una Lista con cuatro componentes.

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

```
## $padre
## [1] "Pedro"
##
## $madre
## [1] "Maria"
```

```
##
## $no.hijos
## [1] 3
##
## $edad.hijos
## [1] 4 7 9
```

```
# Tipo de objetos dentro de la lista
is.matrix(lista1)
```

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

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

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

- *Ejemplo 2:* Acceso a las componentes de una lista:

```
lista1[1] # accede a la componente como una lista (con etiqueta y valor)
```

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

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

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

```
lista1[[2]] # accede al valor o valores de la componente segunda pero no muestra el nombre de la compon
```

```
## [1] "Maria"
```

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

```
## $madre
## [1] "Maria"
```

- *Ejemplo 3:* Acceso a los elementos de la cuarta componente: lista1[[4]][2]

```
lista1[[4]][2] # (se indica el elemento a ingresar en el segundo corchete)
```

```
## [1] 7
```

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

```
lista1$padre # similar a
```

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

```
lista1["padre"]
```

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

Forma general: Nombre_de_lista$nombre_de_componente Por ejemplo:

```
lista1$padre
```

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

```
lista1[[1]]
```

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

```
lista1$edad.hijos[2]
```

```
## [1] 7
```

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

```
## [1] 7
```

- *Ejemplo 5:* Utilizar el nombre de la componente como índice: lista1[[“nombre”]] se puede ver que equivale a lista1$nombre También es útil la forma: x <- “nombre”; lista1[x]

- *Ejemplo 6:* Creación de una sublista de una lista existente:

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

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

- *Ejemplo 7:* Ampliación de una lista: por ejemplo, la lista lista1 tiene 4 componentes y se le puede agregár una quinta componente con:

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

```
## $padre
## [1] "Pedro"
##
## $madre
## [1] "Maria"
##
## $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 <- edit(lista1)
lista1
```

```
## $padre
## [1] "Pedro"
##
## $madre
## [1] "Maria"
##
## $no.hijos
## [1] 3
##
## $edad.hijos
## [1] 4 7 9
##
## [[5]]
## [1] "F" "M" "F"
```

```
#data.entry(lista1) # Muestra una tabla modificable de valores
```

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. Ejecute las siguientes instrucciones: S <- matrix(c(3, -sqrt(2), -sqrt(2), 2), nrow=2, ncol=2); S autovS <- eigen(S); autovS

```
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
```

```
autoS <- eigen(S);
autoS
```

```
## eigen() decomposition
## $values
## [1] 4 1
##
## $vectors
##            [,1]       [,2]
## [1,] -0.8164966 -0.5773503
## [2,]  0.5773503 -0.8164966
```

Observe que la función eigen() retorna una lista de dos componentes, donde la componente

```r
autoS$values # es el vector de autovalores de S y la componente
```

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

```r
autoS$vectors #es la matriz de los correspondientes autovectores.
```

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

```r
#Si quisiéramos almacenar sólo los autovalores de S, podemos hacer lo siguiente:
evals <- eigen(S)$values;
evals
```

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

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

```r
Notas <- matrix(c(2,5,7,6,8,2,4,9,10), ncol = 3, dimnames =list( c("Matematica", "Algerbra", "Geometria
Notas # Los nombres se dan primero para filas y luego para columnas.
```

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

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 lógicos), 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 incluyen sin 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.

```r
log <- sample(c(TRUE, FALSE), size = 20, replace = T);
log # Note que puede usar T en lugar de TRUE y F en lugar de FALSE.
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE
## [13]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE 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)

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

```
##  [1] -0.4974688+0.6958652i  1.8451515+0.0781217i -0.2786587+0.6395862i
##  [4]  0.4454174+0.9647808i  0.5184027+0.1090252i  0.4026810+0.0275609i
##  [7] -1.7734328+0.7801552i -1.3784038+0.9633083i  0.2235775+0.1293954i
## [10]  0.2659357+0.1888439i -0.8057193+0.1264682i -1.1923626+0.1958722i
## [13]  0.1699153+0.9879657i  1.6500525+0.0854174i  2.8904124+0.8358809i
## [16]  3.0515979+0.1081006i -1.0267463+0.9671219i  0.1706237+0.9094222i
## [19] -0.6137539+0.0197806i -0.9345274+0.9067711i
```

El tercer vector tendrá 20 elementos de una distribución Normal estándar num <- rnorm(20, mean=0, sd=1); num

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

```
##  [1] -1.39492981  2.13200513  0.60527483 -1.14137417  0.70424860 -0.77382941
##  [7]  0.16672122 -0.86290923  0.59860837 -0.97484039 -1.38901573  0.01187549
## [13] -1.31828639 -0.04664092  1.09817192 -0.21670577  0.01926030 -0.00168024
## [19]  1.21254461  1.58131825
```

Crear un data frame compuesto por los tres vectores anteriores

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

```
##      log                  comp         num
## 1  FALSE -0.4974688+0.6958652i -1.39492981
## 2  FALSE  1.8451515+0.0781217i  2.13200513
## 3  FALSE -0.2786587+0.6395862i  0.60527483
## 4  FALSE  0.4454174+0.9647808i -1.14137417
## 5  FALSE  0.5184027+0.1090252i  0.70424860
## 6  FALSE  0.4026810+0.0275609i -0.77382941
## 7   TRUE -1.7734328+0.7801552i  0.16672122
## 8   TRUE -1.3784038+0.9633083i -0.86290923
## 9  FALSE  0.2235775+0.1293954i  0.59860837
## 10 FALSE  0.2659357+0.1888439i -0.97484039
## 11  TRUE -0.8057193+0.1264682i -1.38901573
## 12 FALSE -1.1923626+0.1958722i  0.01187549
## 13  TRUE  0.1699153+0.9879657i -1.31828639
## 14  TRUE  1.6500525+0.0854174i -0.04664092
```

```
## 15  TRUE  2.8904124+0.8358809i  1.09817192
## 16 FALSE  3.0515979+0.1081006i -0.21670577
## 17  TRUE -1.0267463+0.9671219i  0.01926030
## 18  TRUE  0.1706237+0.9094222i -0.00168024
## 19  TRUE -0.6137539+0.0197806i  1.21254461
## 20 FALSE -0.9345274+0.9067711i  1.58131825
```

Crear un vector de nombres de los tres vectores anteriores

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

```
## [1] "logico"   "complejo" "numerico"
```

Define los nombres de las columnas del data frame asignándoles el vector nombres names(df1) <- nombres; df1

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

```
##    logico              complejo    numerico
## 1   FALSE -0.4974688+0.6958652i -1.39492981
## 2   FALSE  1.8451515+0.0781217i  2.13200513
## 3   FALSE -0.2786587+0.6395862i  0.60527483
## 4   FALSE  0.4454174+0.9647808i -1.14137417
## 5   FALSE  0.5184027+0.1090252i  0.70424860
## 6   FALSE  0.4026810+0.0275609i -0.77382941
## 7    TRUE -1.7734328+0.7801552i  0.16672122
## 8    TRUE -1.3784038+0.9633083i -0.86290923
## 9   FALSE  0.2235775+0.1293954i  0.59860837
## 10  FALSE  0.2659357+0.1888439i -0.97484039
## 11   TRUE -0.8057193+0.1264682i -1.38901573
## 12  FALSE -1.1923626+0.1958722i  0.01187549
## 13   TRUE  0.1699153+0.9879657i -1.31828639
## 14   TRUE  1.6500525+0.0854174i -0.04664092
## 15   TRUE  2.8904124+0.8358809i  1.09817192
## 16  FALSE  3.0515979+0.1081006i -0.21670577
## 17   TRUE -1.0267463+0.9671219i  0.01926030
## 18   TRUE  0.1706237+0.9094222i -0.00168024
## 19   TRUE -0.6137539+0.0197806i  1.21254461
## 20  FALSE -0.9345274+0.9067711i  1.58131825
```

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 -0.4974688+0.6958652i -1.39492981
## b  FALSE  1.8451515+0.0781217i  2.13200513
## c  FALSE -0.2786587+0.6395862i  0.60527483
```

```
## d  FALSE  0.4454174+0.9647808i -1.14137417
## e  FALSE  0.5184027+0.1090252i  0.70424860
## f  FALSE  0.4026810+0.0275609i -0.77382941
## g   TRUE -1.7734328+0.7801552i  0.16672122
## h   TRUE -1.3784038+0.9633083i -0.86290923
## i  FALSE  0.2235775+0.1293954i  0.59860837
## j  FALSE  0.2659357+0.1888439i -0.97484039
## k   TRUE -0.8057193+0.1264682i -1.38901573
## l  FALSE -1.1923626+0.1958722i  0.01187549
## m   TRUE  0.1699153+0.9879657i -1.31828639
## n   TRUE  1.6500525+0.0854174i -0.04664092
## o   TRUE  2.8904124+0.8358809i  1.09817192
## p  FALSE  3.0515979+0.1081006i -0.21670577
## q   TRUE -1.0267463+0.9671219i  0.01926030
## r   TRUE  0.1706237+0.9094222i -0.00168024
## s   TRUE -0.6137539+0.0197806i  1.21254461
## t  FALSE -0.9345274+0.9067711i  1.58131825
```

- *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
```

```
#fix(hoja1)
```

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 expresion hoja1$Edad, 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()

Si no ha realizado ninguna conexión o desconexión su valor es:
[1] ".GlobalEnv" "package:methods" "package:stats"
[4] "package:graphics" "package:grDevices" "package:utils"
[7] "package:datasets" "Autoloads" "package:base" donde .GlobalEnv corresponde al espacio de trabajo.

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: Edad hoja1$Peso <- Peso+1; hoja1

```
attach(hoja1)
search()
```

```
##  [1] ".GlobalEnv"        "hoja1"             "package:stats"
##  [4] "package:graphics"  "package:grDevices" "package:utils"
##  [7] "package:datasets"  "package:methods"   "Autoloads"
## [10] "package:base"
```

```
Edad
```

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

```
hoja1$Peso <- Peso+1;
hoja1
```

```
##   Edad Estatura Peso Sexo
## 1   18      150   66    F
## 2   21      160   69    M
## 3   45      180   66    M
## 4   54      205   70    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 su puede acceder a una componente sólo con su nombre, por ejemplo, Teclee Edad

```
detach(hoja1)
search()
```

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

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.