Tarea 1 | STAT NT

Luis Gagñevin 5.153.261-6

4/29/2021

1 Ejercicio 1

1.1 Parte 1: Vectores

1.1.1 Dado los siguientes vectores, indica a que tipo coercionan

```
w <- c(29, 1L, FALSE, "HOLA")
x <- c("Celeste pelela!", 33, NA)
y <- c(seq(3:25), 10L)
z <- paste(seq(3:25), 10L)
```

Las Coerciones ocurren del mas rigido al mas flexible (Logic -> Integer -> Numeric -> Character) Y en caso de combinaciones toma al mas flexible como su tipo, teniendo esto en cuenta:

W es un vector del tipo Character

W Tiene Elementos clasificados como: Logic, Integer, Numeric y Character, por lo que tomara el mas flexible, osea Character.

X es un vector del tipo Character

X Tiene Elementos clasificados como: Logic, Numeric y Character, por lo que tomara el mas flexible, osea Character.

Y es un vector de ltipo Integer

Y tiene Elementos clasificados solo como Integer

Z es un vector del tipo character

Aunque Z sea practicamente igual a y dentro de los (), el uso de paste concatena los vectores luego de convertirlos en texto, por lo que genera un vector con solo elementos del tipo Character

1.1.2 ¿Cual es la diferencia entre c(4,3,2,1) y 4:1?

Si bien su resultado se ve igual, la clasificacion de estos es distinta. Ya que 4:1 es tomado como una cadena entera, mientras que c(4,3,2,1) se toma como un vector numerico.

1.2 Parte 2: factor

Dado el siguiente factor x:

```
x <-
factor(
c(
"alto",
"bajo",
"medio",
"alto",
"muy alto",
"bajo",
"medio",
"alto",
"ALTO",
"MEDIO",
"BAJO",
"MUY ALTO",
"QUE LOCO",
"QUE LOCO",
"QUE LOCO",
"A",
"B",
"C",
"GUAU",
"GOL",
"MUY BAJO",
"MUY BAJO",
"MUY ALTO"
)
)
```

1.2.1 Genera un neuvo factor (llamalo xx) transformando el objeto x previamente generado de forma que quede como sigue:

[1] A B M A A B M A A M B A B B A Levels: B < M < A

Para reordenar y que quede de la misma forma, tomare los niveles que me sirven dejando como NA las palabras que no contienen ni “Alto, Medio o Bajo”.

Luego cambiare cada nivel por “alto”, “bajo” y “medio”

Luego elimino los NA y cambio los niveles por “A, B y M” dejandolo casi listo

Y para finalizar ordeno los niveles donde B sea el menor y A el mayor

```
xx <- factor(x, levels= c("alto","ALTO", "bajo", "BAJO", "medio",
                          "MEDIO", "muy alto", "MUY ALTO", "MUY BAJO"))
```

```
levels(xx)[levels(xx)=="ALTO"]<- "alto"
levels(xx)[levels(xx)=="MUY ALTO"]<- "alto"
levels(xx)[levels(xx)=="muy alto"]<- "alto"
levels(xx)[levels(xx)=="BAJO"]<- "bajo"
levels(xx)[levels(xx)=="MUY BAJO"]<- "bajo"
levels(xx)[levels(xx)=="muy bajo"]<- "bajo"
levels(xx)[levels(xx)=="MEDIO"]<- "medio"

xx<-xx[!is.na(xx)]
levels(xx)[levels(xx)=="alto"]<- "A"
levels(xx)[levels(xx)=="bajo"]<- "B"
levels(xx)[levels(xx)=="medio"]<- "M"

xx<- ordered(xx, levels=c("B","M","A"))
```

1.2.2. Generá el siguiente data.frame()

Para ello usá el vector xx que obtuviste en la parte anterior.

Para generar el data.frame() tome los valores unicos de x en una nueva variable, genere un df con los valores de la nueva variable y luego en una nueva df tomamos los niveles de x y los ordenamos haciendo match a los valores de las variables "Levels" para luego ordenarlo en un solo df, dejandolo de minusculas a mayusculas.

```
x2<-unique(x)
df<- data.frame(x2[1:15])
names(df)[1]<- "levels"

df2<-data.frame(levels(x))
df2$value<- c(1:15)
names(df2)[1]<- "levels"
df2<-df2[order(match(df2$levels, df$levels)),]
df$value<- df2[,2]
```

```
##      levels value
## 1      alto     2
## 2      bajo     5
## 3     medio    10
## 4  muy alto    12
## 5      ALTO     3
## 6     MEDIO    11
## 7      BAJO     6
## 8  MUY ALTO    13
## 9  QUE LOCO    15
## 10        A     1
## 11        B     4
## 12        C     7
## 13     GUAU     9
## 14      GOL     8
## 15 MUY BAJO    14
```

1.3 Parte 2: Listas

1.3.1 Genera una lista que se llame lista_t1 que contenga:

- Un vector numérico de longitud 4 (h).
- Una matriz de dimensión 4*3 (u).
- La palabra “chau” (palabra).
- Una secuencia diaria de fechas (clase Date) desde 2021/01/01 hasta 2021/12/30 (fecha)

```
h<- c(1,2,3,4)
u<- matrix(1:12, nrow=4)
palabra<- "chau"
fecha <- seq(as.Date('2021-1-1'), as.Date('2021-12-30'), by='day')
lista_t1<- list(h,u,palabra,fecha)
```

1.3.2 ¿Cual es el tercer elemento de la primera fila de la matriz m? ¿Que columna lo contiene?

```
lista_t1[[2]][3,1]
```

```
## [1] 3
```

1.3.3 ¿Cual es la diferencia entre hacer lista_t1[[2]][]<-0 y lista_t1[[2]] <- 0?

Al realizar lista_t1[[2]][] <-0 quedan todos los valores de la matriz iguales a 0

```
lista_t1[[2]][]<-0
```

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

Mientras que al realizar lista_t1[[2]] <- 0, cambia la matriz por el valor 0.

```
lista_t1[[2]]<-0
```

0

1.3.4 Iteracion

Iterá sobre el objeto lista_t1 y obtené la clase de cada elemento teniendo el cuenta que si la longitud de la clase del elemento es mayor a uno nos quedamos con el último elemento. Es decir, si class(x) es igual a c(“matrix”, .array“) el resultado debería ser “array”. A su vez retorná el resultado como clase list y como character

Para lograr esto, lo que realize es crear una funcion donde utiliza un for() con la variable i en la que mira cuales son las clases de cada parte de la lista y luego cuenta sus elementos, si estos son iguales a 1, coloca el valor correcto, si estos son mayores a 1 coloca el ultimo tipo de clase

```
clasificaciones <- function(lista){
  clases <- list()
for (i in 1:length(lista)) {
  cls<- class(lista[[i]])
  if (length(cls)==1) {
    clases[[i]]<- as.character(cls)
  }else{
    clases[[i]]<- as.character(cls[length(cls)])
  }
}
print(clases)
}
```

```
clasificaciones(lista_t1)
```

```
## [[1]]
## [1] "numeric"
##
## [[2]]
## [1] "array"
##
## [[3]]
## [1] "character"
##
## [[4]]
## [1] "Date"
```

1.3.5 Iteracion (2)

Utilizando las últimas 10 observaciones de el elemento "fecha" del objeto "lista_t1" escriba para cada fecha "La fecha en este momento es . . . ." donde ". . ." debe contener la fecha para valor de lista$fecha. Ejemplo: "La fecha en este momento es '2021-04-28' ". Hacelo de al menos 2 formas y que una de ellas sea utilizando un for. Obs: En este ejercicio NO imprimas los resultados.

Forma 1)

En esta forma utilizamos una funcion del tipo for() donde toma los valores de un data.frame con 10 fechas y los acomoda en una oracion

```
lista <- data.frame(tail(lista_t1[[4]], 10))
names(lista)[1] <- "fecha"

for (i in 1:10) {
  print(paste("La fecha en este momento es ", as.character(lista$fecha[i])))
}
```

Forma 2)

En esta forma utilizamos una funcion del tipo while donde toma los valores de un data.frame con 10 fechas y los acomoda en una oracion

```r
lista <- data.frame(tail(lista_t1[[4]], 10))
names(lista)[1] <- "fecha"

i=1
while (i <= 10) {
  print(paste("La fecha en este momento es ", as.character(lista$fecha[i])))
  i=i+1
}
```

1.4 Parte 3: Matrices

1.4.1 Genera una matriz A de dimension 4x3 y una matriz B de dimension 4x2 con numeros aleatorios usando alguna funcion predefinida en R

Elegimos los valores aleatorios utilizando la funcion sample()

```r
A<- matrix(sample(1:100,12), ncol=3)
B<- matrix(sample(1:100,8), ncol=2)
```

1.4.2 Calcula el producto elemento a elemento de la primera columna de la matriz A por la ultima columna de la matriz B

```r
A[,1]*B[,2]
```

```
## [1] 2128 1085   26 7221
```

1.4.3 Calcula el producto matricial entre D=AT B luego selecciona los elementos de la primera y tercera fila de la segunda columna (en un paso)

Para seleccionar ambos valores con un solo paso utilizamos un vector

```r
D<- t(A)%*%B
D[c(1,3),2]
```

```
## [1] 10460  4056
```

1.4.4 Usa las matrices A y B de forma tal de lograr una matriz C de dimension 4x5. Con la funcion Attributes inspecciona los atributos de C. Posteriormente renombra filas y columnas como “fila_1”, “fila_2” . . . “columna_1”, “columna_2”, vuelve a inspeccionar los atributos. Finalmente, generaliza y escribi una funcion que reciba como argumento una matriz y devuelva como resultado la misma matriz con columnas y filas con nombres.

Utilizamos la funcion cbind() para combinar ambas matrices y veo con el atributo que solo tiene el $dim

```r
c<- cbind(A,B)
attributes(c)
```

```
## $dim
## [1] 4 5
```

Utilizo como primer forma la funcion provideDimnames() y el atributo nos muestra los $dimnames de las columnas y filas en un formato parecido a una lista

```
c<- provideDimnames(c, sep="_", base=list("fila", "columna"))
attributes(c)
```

```
## $dim
## [1] 4 5
##
## $dimnames
## $dimnames[[1]]
## [1] "fila"   "fila_1" "fila_2" "fila_3"
##
## $dimnames[[2]]
## [1] "columna"   "columna_1" "columna_2" "columna_3" "columna_4"
```

Aca realizo una funcion general que hace un nested loop con dos for donde renombra todas las filas y columnas de una matriz segun su posicion

```
nombres <- function(matriz){
  filas<-dim(matriz)[1]
  columnas<- dim(matriz)[2]

  for (i in 1:filas) {
      for (j in 1:columnas) {
    rownames(matriz)[i]<- paste("fila_",i, sep="")

    colnames(matriz)[j]<- paste("columna_",j, sep="")
      }}

  return(matriz)
}
```

```
##        columna_1 columna_2 columna_3 columna_4 columna_5
## fila_1        28        64        11        57        76
## fila_2        31        10        38        20        35
## fila_3        13        48        75        89         2
## fila_4        83        61        20        77        87
```

1.4.5 Puntos Extra: Generaliza la funcion para que funcione con arrays de forma que renombre filas, columnas y matrices

```
renombre <- function(ary){
  ary <- provideDimnames(ary, sep="_", base=c("fila","columnas","matriz"), unique=TRUE)

  return(ary)
  }
```

```
##        columna columna_1 columna_2 columna_3 columna_4
## fila        28        64        11        57        76
## fila_1      31        10        38        20        35
## fila_2      13        48        75        89         2
## fila_3      83        61        20        77        87
```