Principios y Herramientas de Programación

Dra. Jessica Andrea Carballido

jac@cs.uns.edu.ar

Dpto. de Ciencias e Ingeniería de la Computación

UNIVERSIDAD NACIONAL DEL SUR

```
    5
ames(sort(apply(ejemplo,1,sum)))[1]
1] "d"
> apply(ejemplo,1,sum)
a b c d e f g h
7 6 5 4 5 6 7 8
> sort(apply(ejemplo,1,sum))
d c e b f a g h
4 5 5 6 6 7 7 8
sort(apply(ejemplo,1,sum))[1]


y(ejemplo,1,sum))
```

Sobre funciones como argumento

```
miResumen=function(x){c(min(x),max(x),mean(x))}
```

```
> set.seed(5)
> V=sample(1:100,12)
> V
 [1] 66 57 79 75 41 85 94 71 1
> M=matrix(V,3)
> M
     [,1] [,2] [,3] [,4]
[1,]   66   75   94    3
[2,]   57   41   71   38
[3,]   79   85   19   58
> apply(M,1,miResumen)
     [,1]  [,2]  [,3]
[1,]  3.0 38.00 19.00
[2,] 94.0 71.00 85.00
[3,] 59.5 51.75 60.25
>
```

Apply recibe cualquier función
que tome un vector como dato de entrada

Resultado de aplicar "miResumen"
a la última fila de M

Corolario: las funciones pasadas como argumento pueden ser tanto predefinidas como definidas por el programador.

Sobre funciones como argumento

```
primerMultTres=function(x){x[x%%3==0][1]}
```

```
> V=c(5,3,6,2,9,12,21,10,3)
> M=matrix(V,3, byrow=T)
> M
     [,1] [,2] [,3]
[1,]    5    3    6
[2,]    2    9   12
[3,]   21   10    3
> apply(M,1,primerMultTres)
[1]  3  9 21
> apply(M,2,primerMultTres)
[1] 21  3  6
> M[3,]=2
> M
     [,1] [,2] [,3]
[1,]    5    3    6
[2,]    2    9   12
[3,]    2    2    2
> apply(M,1,primerMultTres)
[1]  3  9 NA
>
```

Retorna el primer valor en el arreglo que sea múltiplo de 3

apply recibe cualquier función que reciba un vector como dato de entrada

Corolario: las funciones pasadas como argumento pueden ser tanto predefinidas como definidas por el programador.

Sobre funciones como argumento

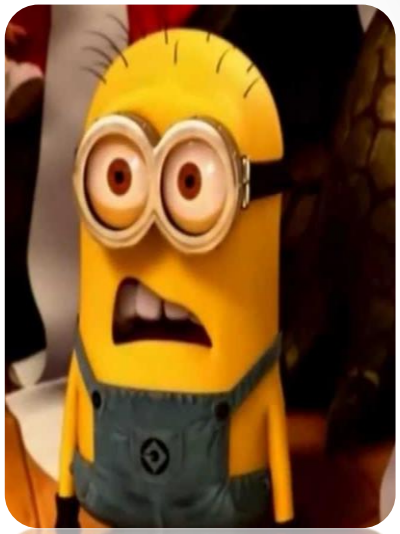
```
> minmax=function(x){c(min(x),max(x))}

> sexo=sample(c("hombre","mujer"),20,replace=T)
> sexo
 [1] "mujer"  "mujer"  "mujer"  "mujer"  "mujer"  "mujer"  "mujer"  "mujer"
 [9] "hombre" "hombre" "mujer"  "mujer"  "mujer"  "mujer"  "hombre" "hombre"
[17] "mujer"  "hombre" "hombre" "hombre"

> set.seed(0)
> edades=round(runif(20,1,99))
> edades
 [1] 89 27 37 57 90 21 89 94 66 63  7 21 18 68 39 76 50 71 98 38
> tapply(edades,sexo,minmax)
$hombre
[1] 38 98

$mujer
[1]  7 94
```

tapply también recibe **cualquier** función que tome un vector como dato de entrada

Corolario: las funciones pasadas como argumento pueden ser tanto predefinidas
como definidas por el programador.

Polimorfismo: sobrecarga

```
> mas5=function(x){x+5}
> mas5(2)
[1] 7
> mas5(w)
[1]  6  7  8  9 10
> m
     col1 col2 col3
[1,]   60   99   41
[2,]   19   40   16
[3,]   34   53   92
> mas5(m)
     col1 col2 col3
[1,]   65  104   46
[2,]   24   45   21
[3,]   39   58   97
>
```



Podemos definir en C una función como esta?

Repaso

Creación: c(), :, rep, seq

Generación de muestras aleatorias:

- Sample:

```
> xy=c("malo", "reg", "bueno")
> sample(xy, 10)
Error en sample(xy, 10) :
  cannot take a sample larger than the population when 'replace = FALSE'
> sample(xy, 3)
[1] "bueno" "malo"  "reg"
> pp=c(0.1,0.1,0.8)
> sample(xy, 10, replace=TRUE, prob=pp)
 [1] "bueno" "bueno" "reg"   "bueno" "reg"   "bueno" "bueno" "bueno" "bueno" "bueno"
>
```

- Con distribuciones de probabilidad:

r*dist* donde *dist* puede ser uniforme, normal, exponencial, poisson, etc.

Por ejemplo rnorm(n, mean=0, sd=1)

Repaso

Formas de acceder a los elementos de un vector con un vector índice de:

- **Valores lógicos**

Source on Save

```
1  x <- c(10,20, NA, 4, NA, 2)
2  sum(x)/length(x) # da error por los NA
3  (i <- is.na(x))
4  i ; !i
5  x[!i] # Usamos el vector negado de indices para extraer los valores numericos de x
6  nuevo.x <- x[!i]
7  sum(nuevo.x)/length(nuevo.x)
8  # NOTA: Se obtiene el mismo resultado con mean(x, na.rm = TRUE)
```

- **Cadenas de caracteres**

```
9
10 x=c(50,65,65,80,70)
11 names(x) <- c("Ana", "Beto", "Carlos", "Diana", "Eduardo")
12 x[c("Eduardo","Carlos")]
```

Repaso

Formas de acceder a los elementos de un vector con un vector índice de:

- **Enteros positivos**

```
14  x = runif(100) # 100 numeros aleatorios entre 0 y 1
15  round(x[20:30], 3) # Redondear los elementos 20 al 30
16  i = c(20:25, 28, 35:49) # 20 al 25, 28 y del 35 al 49
17  round(x[i], 2)
18
```

- **Enteros negativos**

```
19  x[c(-1,-5,-80)]
20  x[-(20:30)]
21
```

No se pueden mezclar!

```
[91] 0.03011694 0.05199271 0.23300202 0.13490076 0.21002436 0.066
> x[-1:3]
Error en x[-1:3] :
  solamente 0's pueden ser mezclados con subscritos negativos
> x[c(-1,-2,4)]
Error en x[c(-1, -2, 4)] :
  solamente 0's pueden ser mezclados con subscritos negativos
```

Repaso

Operaciones entre vectores:

Las operaciones binarias se aplican entre pares de elementos.

Por ejemplo x+y genera un vector con las sumas x_i+y_i

Funciones vectoriales:

length() longitud
sum() suma de las componentes del vector
prod() producto de las componentes del vector
cumsum(), **cumprod**() suma y producto acumulados
max(), **min**() máximo y mínimo del vector
cummax(), **cummin**() máximo y mínimo acumulados
sort() ordena el vector
diff() calcula la diferencia entre las componentes

```
> a=round(runif(10, 1, 10))
> a
 [1]  7 10  8 10 10  8  2  4  9  8
> cumsum(a)
 [1]  7 17 25 35 45 53 55 59 68 76
> cummax(a)
 [1]  7 10 10 10 10 10 10 10 10 10
> cummin(a)
 [1] 7 7 7 7 7 7 2 2 2 2
```

Repaso

Funciones vectoriales (cont.):

which(x == a) vector de los índices de x para los cuales
la comparación es verdadera
which.max(x) índice del mayor elemento
which.min(x) índice del menor elemento
range(x) valores del mínimo y el máximo de x
mean(x) promedio de los elementos de x
median(x) mediana de los elementos de x
round(x, n) redondea los elementos de x a n decimales (n=0 por def.)
rank(x) rango de los elementos de x
order(x) vector de permutaciones para ordenar a x
unique(x) vector con las componentes de x sin repeticiones

Repaso

Funciones vectoriales (cont.):

```
R> a <- c(4.1, 3.2, 6.1, 3.1)
R> order(a)
[1] 4 2 1 3
```

Indica que para ordenar debo tomar primero el 4to elemento, luego el 2do elemento, luego el 1ro y finalmente el 3ro.

which.min(x) índice del menor elemento
range(x) valores del mínimo y el máximo de
mean(x) promedio de los elementos de x
median(x)
round(x,

```
R> rank(a)
[1] 3 2 4 1
```

Indica en que orden se encuentra cada elemento en el vector ordenado.

rank(x) rango de los elementos de x
order(x) vector de permutaciones para ordenar a x
unique(x) vector con las componentes de x sin repeticiones

Repaso

```
> x=1:6
> diag(x)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    0    0    0    0    0
[2,]    0    2    0    0    0    0
[3,]    0    0    3    0    0    0
[4,]    0    0    0    4    0    0
[5,]    0    0    0    0    5    0
[6,]    0    0    0    0    0    6
> dim(x)=c(2,3)
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Creación de matrices: matrix(), cbind(), rbind(), diag(), dim()

Funciones:

ncol(x) Número de columnas de x.

nrow(x) Número de filas de x.

t(x) Transpuesta de x

cbind(...) Combina secuencias de vectores/matrices por columnas.

rbind(...) Combina secuencias de vectores/matrices por filas.

diag(x) **Extrae diagonal de matriz** o crea una matriz diagonal a partir de un vector.

col(x) Crea una matriz con elemento ij igual al valor j

row(x) Crea una matriz con elemento ij igual al valor i

apply(x,margin,FUN) Aplica la función FUN a la dimensión especificada en *margin*: 1 indica filas, 2 indica columnas.

Obtener el mayor valor en la diagonal de una matriz: max(diag(m))

Repaso

Data.frames

- Las componentes deben ser vectores (numéricos, carácter o lógicos), factores, matrices numéricas u otros *data.frames*.
- La longitud de los vectores debe ser la misma y se pegan como columnas.
- A los datos que no son numéricos, la estructura de *data.frame* los considera factores, con tantos niveles como valores distintos encuentre.

```r
# Repaso
nuevo=na.omit(studentdata); attach(nuevo)
nuevo[Drink=="pop",]
subset(nuevo,Drink=="pop")

nuevo[Drink=="milk" & Gender=="male",c("Height","Job")]
subset(nuevo,Drink=="milk"& Gender=="male", select=c(Height,Job))

# CUANTOS?
nrow(subset(nuevo, Drink=="water" & Job!=0 & Height>70))

# resumen
table(Gender, Drink)
#            Drink
# Gender    milk pop water
# female    55   94    215
# male      42   60     93

dim(nuevo) #[1] 559  11
sum(table(Gender, Drink)) # [1] 559
nrow(subset(nuevo, Drink=="pop"))
tapply(Height,Gender,max)
# female   male
# 84     79
tapply(Shoes,Drink,mean)
# milk      pop    water
# 12.53093 14.18182 17.25325
apply(nuevo[,c(2,4,9,10)],2,max)
# Height   Shoes Haircut     Job
# 84     164     180      80
```

ógicos),

o

subset: retorna un sub-grupo de filas del 1er argumento, de acuerdo al criterio mostrado en el 2do argumento, con todas las columnas salvo que se indiquen solo algunas con el 3er argumento

Repaso

file.choose()

Cargar datos desde archivo:

- read.table() o read.csv() # para cargar un dataframe
- load() # para cargar objetos previamente guardados con save()

Guardar datos:

- write.table() o write.csv()
- save() # guardar un dato “*.rdata”

Para guardar todos los datos:

- save.image() # se guarda como ".RData"
- save.image(file="nombre.RData")

El archivo ".RData" se carga al iniciar R.

Espero que hayan disfrutado
de esta pequeña muestra
de lo que es el arte de programar.

GOOD LUCK FOR

EXAMS