PROGRAMACIÓ

CADENES DE CARÀCTERS EN LLENGUATGE R

UB-UPC Grau en Estadística

Robert Joan Arinyo

Departament de Ciències de la Computació

UPC

Índex

Caràcters/cadenes, per què?

Quina diferència hi ha entre les informacions que donen les següents paraules?

```
Barcelona          barcelona
```

I en el cas?

```
mas          Mas
```

I quin significat tenen els dígits d'un número de telèfon?

```
934 320 607
```

Tipus de dades

En Informàtica un tipus de dades resta definit per dos conceptes:

1. Un conjunt finit de valors representats d'una manera prefixada.

2. Un conjunt d'operacions definides sobre el conjunt de valors.

Classes de tipus de dades

El llenguatge **R** ofereix dades dels tipus

1. Simples o escalars
   - enter: subconjunt dels $\mathbb{Z}$ de la matemàtica
   - real: subconjunt dels $\mathbb{R}$ de la matemàtica
   - booleà: booleans de la matemàtica
   - **caràcter**: conjunt de caràcters de l'escriptura
2. Compostos
   - cadenes de caràcters
   - vectors de caràcters
   - llistes (`list`) que contenen caràcters
   - matrius (`table`) que contenen caràcters
   - taules (`data frames`) que contenen caràcters

Caràcters 1/2

- Un **caràcter** és una informació que representa un símbol dels elements emprats en l'escriptura.
- Inclou les lletres minúscules, majúscules, digits àrabs, símbols especials i símbols de puntuació.
- Hi ha diverses maneres de codificar els caràcters: taula ASCII, Unicode UTF8, etc.

Caràcters (2/2)

En llenguatge **R** la informació tipus caràcter es codifica escrivint el caràcter entre dos símbols cometes dobles (").

Exemples són:

```
"a"     "3"     "$"     ";"
```

Cadenes

- Una **cadena** és un tipus de dades compost tal que representa una seqüència de caràcters delimitada.
- Una cadena es representa
  1. Com una constant:

     ```
     "Aixo 13 es **-2 una KadeNa"
     ```

  2. Amb un identificador:

     ```
     c = "Aixo 13 es **-2 una KadeNa"
     ```

     Veurem que `c` no és exactament una variable.

- Un cas especial és la cadena que no conté cap caràcter, la cadena buida: `""`.

Operacions (1/5)

Només presentarem les operacions bàsiques.
Siguin `c1` i `c2` dues cadenes de caràcters, les quals poden tenir un únic caràcter,

Concatenació i assignació: `c3 <- paste(c1, c2)`

Per exemple

```
c3 <- paste("Hola", "lluna")
```

és la cadena `"Hola lluna"`

El caràcter separador es controla amb un paràmetre

```
c3 <- paste("Hola", "lluna", sep="**")
```

`c3` és la cadena `"Hola**lluna"`

Operacions (2/5)

Comparació lexicogràfica (diccionari): resten definides les expressions de relació

```
c1 < c2
c1 <= c2
c1 == c2
c1 >= c2
c1 > c2
```

Segons l'ordre de definició dels caràcters a la taula ASCII, UTF8, etc

Operacions (3/5)

Llargària d'una cadena:

Sigui `c` una cadena de caràcters. La funció

```
nchar(c)
```

retorna un valor enter que indica el nombre de caràcters de la cadena.

`nchar("abc")` és 3

`nchar("")` és zero

Operacions (4/5)

- Una **subcadena** és un subconjunt de caràcters **contigus** d'una cadena. Si `c` és una cadena, la funció

  ```
  substring(c, inici, fi)
  ```

  denota la subcadena delimitada pels índexs

  `inici`: posició on comença la subcadena
  `fi`: posició on acaba la subcadena

  Si `c = "PaulJohnGeorgeAndRingo"`

  ```
  substring(c, 5, 8)
  ```

  denota la subcadena `"John"`

Operacions (5/5)

L'operador lògic `%in%` és força útil i determina si una cadena està inclosa en alguna component d'un vector de cadenes.

Experimenta amb el següent segment de codi

```
> cadena <- c("ab", "123", "*??*")
> "*??*" %in% cadena
[1] TRUE
> "*?" %in% cadena
[1] FALSE
>
```

Operacions E/S (1/3)

Les cadenes es poden entrar al programa des del teclat de manera similar a com es fa amb la informació numèrica.

Lectura d'una total d'`m` cadenes pel teclat

```
scan(n=m, what='character', quite=TRUE)
```

Cada cadena s'acaba amb un `ENTER`.

Per exemple, la sentència

```
cadena <- scan(n=3,what='character',quite=TRUE)
```

Genera una interacció al teclat com la següent

```
1: ab
2: 123
3: *-*
```

`cadena` és un vector de tres components amb valors

`"ab", "123", "*-*"`

Operacions E/S (2/3)

Les cadenes es poden escriure al terminal de manera similar a com es fa amb la informació numèrica.

Escriptura d'una cadena definida a la pròpia escriptura. La línia

```
cat("Escriu aquesta cadena \n")
```

Genera en la pantalla la sortida

```
> cat("Escriu aquesta cadena \n")
Escriu aquesta cadena
>
```

Si `c1, c2, ...` són cadenes, l'escriptura serà

```
cat(c1, c2, "\n")
```

Exerimenta què passa quan s'elimina el codi `\n`

Operacions E/S (3/3)

Una forma flexible i pràctica de llegir una cadena de llargada variable acabada amb la tecla `ENTER` és la comanda `readline()`.

El format del codi és

```
cadena <- readline(prompt="Entra la cadena : ")
```

Immutabilitat (1/2)

Les cadenes de caràcters, encara que siguin representades mitjançant un nom simbòlic, **NO** són variables, són constants. Per tant, el codi

```
cadena <- "Una cadena de lletres"
cadena[5] <- "K"
cat(cadena)
```

Escriurà?

```
"Una Kadena de lletres"
```

Immutabilitat (2/2)

- No. No és possible canviar o extreure el valor d'un caràcter d'una cadena predefinida mitjançant un índex com si fos un vector.

- El canvi proposat cal fer-lo creant cadenes noves

```
c  <- "Una cadena de lletres"
c1 <- substring(c, 1, 4)
c2 <- substring(c, 6, nchar(c))
c3 <- paste("K", c2, sep="")
c4 <- paste(c1, c3, sep="")
```

Prova-ho!

Les funcions `sub()` i `gsub()` no funcionen a la meva instal.lació

Transformacions (1/2)

Transformació d'un vector de cadenes en una cadena única

Si `v` és un vector de cadenes, la sentència

```
cadena <- paste(v, collapse="")
```

genera una cadena amb totes les cadenes que emmagatzemava `v`.

Per exemple

```
> v <- c('a','b','c')
> c <- paste(v, collapse="")
> c
[1] "abc"
```

Transformacions (2/2)

La sintaxi general de la transformació d'una cadena en un vector de cadenes és

```
strsplit(cadena, separador)[[1]]
```

Alguns dels diversos trencaments possibles són

- Trencament en un vector de caràcters

```
> c <- "Tant de bo fos sempre primavera"
> v <- strsplit(c, "")[[1]]
> v
[1] "T" "a" "n" "t" " " "d" "e" " " "b" "o" "
"r" "e" " " "p" "r" "i" "m" "a" "v" "e" "r" "a
```

- Trencament en un vector de cadenes

```
> c <- "Tant de bo fos sempre primavera"
> v <- strsplit(c, "e")[[1]]
> v
[1] "Tant d" " bo fos s" "mpr" " primav" "ra"
```

Recorregut de cadenes (1/7)

Donada una frase pel CEE acabada en un caràcter $ dissenya un programa que mostri totes les lletres de la frase que siguin majúscules.

Recorregut de cadenes (2/7)

1. S : `c1c2c3 ... ci ... cn$`
   S': `c1, c2, c3, ... ci, ... cn, $`
2. Representació
   `c` : representerà el text
   `i` : índex del caràcter en tractament
3. Esquema: Recorregut
4. Caracterització
   - primer : `i = 1`
   - següent : `i = i + 1`
   - acabament :c[i] == $ Primer que no cal tractar

Recorregut de cadenes (3/7)

- Programa: solució 1

```
majuscules = c("A","B","C","D","E","F","G","H","I",
               "J","K","L","M","N","O","P","Q","R",
               "S","T","V","W","X","Y","Z")

cadena <- readline(prompt="Text acabat en $ : ")
cadena <- strsplit(cadena, "")[[1]]

i <- 1
while (cadena[i] != "$"){
  if (cadena[i] %in% majuscules){
    cat(" lletra : ", cadena[i], "\n")
  }
  i <- i + 1
}
cat("\n")
```

Recorregut de cadenes (4/7)

1. S : `c1c2c3 ... ci ... cn`
   S': `c1, c2, c3, ... ci, ... cn`
2. Representació
   `c`: Representarà tot el text
   `i` : índex del caràcter en tractament
3. Esquema: Recorregut
4. Caracterització
   - primer : `i = 0`
   - següent : `i = i + 1`
   - acabament : `i <= length(S)` Unitats que cal tractar

Notis que el sentinella $ no té cap utilitat.

Recorregut de cadenes (5/7)

- Programa: Solució 2

```
majuscules = c("A","B","C","D","E","F","G","H","I",
               "J","K","L","M","N","O","P","Q","R",
               "S","T","V","W","X","Y","Z")

cadena <- readline(prompt="Text : ")
cadena <- strsplit(cadena, "")[[1]]

n <- length(cadena)
i <- 1
while (i <= n){
  if (cadena[i] %in% majuscules){
    cat(" lletra : ", cadena[i], "\n")
  }
  i <- i + 1
}
cat("\n")
```

Recorregut de cadenes (6/7)

- Programa: Solució 3

```
majuscules = c("A","B","C","D","E","F","G","H","I",
               "J","K","L","M","N","O","P","Q","R",
               "S","T","V","W","X","Y","Z")

cadena <- readline(prompt="Text : ")
cadena <- strsplit(cadena, "")[[1]]

n <- length(cadena)
for(i in 1:n){
  if (cadena[i] %in% majuscules){
    cat(" lletra : ", cadena[i], "\n")
  }
}
cat("\n")
```

Recorregut de cadenes (7/7)

- Programa: Solució 4

```
# Determinacio de majuscules emprant l'ordre
# lexicografic del llenguatge R
#
# ("A" <= cadena[i]) & (cadena[i] <= "Z")

cadena <- readline(prompt="Text : ")
cadena <- strsplit(cadena, "")[[1]]

n <- length(cadena)
for(i in 1:n){
  if (("A" <= cadena[i]) & (cadena[i] <= "Z")){
    cat(" lletra : ", cadena[i], "\n")
  }
}
cat("\n")
```

Cerca en cadenes (1/3)

Donat pel CEE un text, dissenya un programa que decideixi si el text contè cap símbol ’;’ (punt-i-coma).

Cerca en cadenes (2/3)

1. S : `c1c2c3 ... ci ... cn`
   S': `c1, c2, c3, ... ci, ... cn`
2. Representació
   `c` : cadena de caràcters
   `i` : índex del caràcter en tractament
3. Esquema: Cerca
4. Caracterització
   - primer : i = 0
   - següent : i = i + 1
   - acabament : i < length(text) Encara resten valors
5. Propietat cercada: `c[i] == ';'`

Cerca en cadenes (3/3)

```
frase <- readline(prompt="Entra una frase : ")

v <- strsplit(frase, "")[[1]]
n <-length(v)

i <- 1
while (!(v[i] == ";") & i < n){
  i <- i + 1
}
if ( v[i] == ";"){
  cat("punt-i-coma trobat \n")
}else{
  cat("no s'ha trobat cap punt-i-coma \n" )
}
```

Conclusió

- El tractament de cadenes de caràcters és de cabdal importància en la programació de computadors en general i en el camp de l'estadística en particular.

- El llenguatge **R** ofereix un conjunt d'eines que en facilita la programació.