Complementos

Sesión 3

Natalie Julian - www.nataliejulian.com

Estadística UC y Data Scientist en Zippedi Inc.

¿Qué puedo hacer con vectores de tipo character en R?

Función paste()

Supongamos que tenemos un vector de nombres y apellidos de algunas personas:

```
nombres<-c("Josefa", "Antonio", "Ariel", "Leonie", "Maria")
apellidos<-c("Ferreira", "Julian", "Prado", "Contreras", "Lopez")
```

Pudiéramos estar interesados en concatenar dicha información, es decir, en un mismo vector tener nombres y apellidos. Esto se puede realizar ´facilmente con la función `paste()`:

```
paste(nombres, apellidos) #Separa con espacio
[1] "Josefa Ferreira"  "Antonio Julian"   "Ariel Prado"      "Leonie Contreras"
[5] "Maria Lopez"

paste(nombres, apellidos, sep="-") #Separa con guion
[1] "Josefa-Ferreira"  "Antonio-Julian"   "Ariel-Prado"      "Leonie-Contreras"
[5] "Maria-Lopez"
```

¡Note que el orden es muy importante! Si no respetáramos el orden podríamos unir nombres con cualquier apellido! Y perderíamos toda nuestra información...

Función `substr()`

Suponga que tenemos un vector con la información del cumpleaños de la persona, con la siguiente estructura:

Dia/Mes/Año

El vector es el siguiente:

```
nacimientos<-c("25/04/1997", "13/03/2004", "04/01/1993", "22/11/1999")
```

Con la función `substr()` podemos extraer determinados caracteres del vector, pero debemos indicar los argumentos `start` (número del caracter a partir del que vamos a extraer) y `end` (número del caracter hasta el que extraeremos). Por ejemplo, probemos extraer desde el caracter 3 hasta el caracter 8 de cada elemento del vector nacimientos:

```
substr(nacimientos, start=3, stop=8)
[1] "/04/19" "/03/20" "/01/19" "/11/19"
```

Función substr()

Si quisiéramos extraer sólo el día de nacimiento es sencillo, extraemos los dos primeros caracteres:

```
substr(nacimientos, start=1, stop=2)
[1] "25" "13" "04" "22"
```

Si quisiéramos extraer sólo el año, tenemos que extraer desde el caracter 7 hasta el último, ¿cómo saber la cantidad de caracteres? Con la función `nchar()`:

```
nchar(nacimientos)  #Cantidad de caracteres de cada elemento del vector
[1] 10 10 10 10

substr(nacimientos, start=7, stop=nchar(nacimientos))
[1] "1997" "2004" "1993" "1999"
```

Si utilizamos `substr(nacimientos, start=7, stop=10)` también sirve. ¿Por qué?

Practica 1

- ¿Cómo extraerías el mes a partir del vector `nacimientos`?
- Como usted podrá notar, al utilizar la función `substr()` se extrae el mes en formato `character`. Si quisiéramos obtener estadísticas del mes, no podríamos pues está en este formato. ¿Qué función habría que utilizar para que el mes tenga formato numérico?

Practica en R

Practica 2

Para un evento se activó un link en el que los participantes podían inscribirse para participar. Los participantes inscritos vía link fueron los siguientes:

Natalie J. Vanesa R. Sergio M. Felipe C. Catalina B. Cristobal B.

Sucede que también había otra manera de inscribirse al evento, vía encuesta. Los participantes inscritos vía encuesta fueron los siguientes:

Catalina B. Fernanda C. Natalie J. Sergio B. Samuel C. Antonia K. Joaquin E.

Practica 2

La persona organizadora del evento necesita saber lo siguiente:

a) ¿Quiénes son todos los inscritos para el evento? ¿Cuántas personas son?
b) ¿Cuántas personas se inscribieron tanto vía link como vía encuesta?
c) ¿Cuáles son las personas que utilizaron sólo una vía para inscribirse?

Practica en R

Obteniendo verdades de los datos

Considere que Bastián tiene 5 años y Lorena tiene 10. Podemos definir objetos con sus edades respectivas:

```
Bastian<-5
Lorena<-10
```

Podemos obtener mucha información a partir de objetos creados, y para obtener esta información, debemos hacernos preguntas:

- ¿Es Bastián menor que Lorena?
- ¿Se cumple que la edad de Bastián es menor que 10 y que la edad de Lorena es menor a 25?
- ¿Es cierto que la edad de Bastián es mayor a 5 o que la edad de Lorena es menor o igual a 3?

Lenguaje lógico en R

Practique las siguientes pruebas lógicas en R:

Operador	Significado	Prueba lógica	Valor lógico
<	Menor que	`Bastian<Lorena`	TRUE
>	Mayor que	`Bastian>Lorena`	FALSE
≤	Menor o igual que	`Bastian<=Lorena`	TRUE
≥	Mayor o igual que	`Bastian>=Lorena`	FALSE
==	Igual que	`Bastian==Lorena`	FALSE
! =	Distinto a	`Bastian!=Lorena`	TRUE
&	Y	`(Bastian<10)&(Lorena<25)`	TRUE
|	O (inclusivo)	`(Bastian>5)|(Lorena<=3)`	FALSE

En el ejemplo del IMC trabajado en clases

El(la) especialista a cargo del estudio necesita conocer información para realizar el seguimiento de lxs pacientes, para lo cual, le realiza las siguientes preguntas:

- ¿Existen índices inferiores a 24? ¿Cuáles?
- ¿Qué índices cumplen que son mayores a 20 y menores que 25?
- ¿Existe algún índice que sea exactamente 24? ¿Y 27?
- ¿Existe algún índice que se repita? ¿Cuál(es)?

¿Cómo se interpreta el resultado de cada pregunta?

Pruebas lógicas en vectores

Podemos aplicar pruebas lógicas a vectores. En ese caso, vamos a obtener un vector de valores lógicos.

Recordemos que en clases definimos el vector de IMC como sigue:

```
IMC<-c(19.5,25,27,25,23,23,35)
```

Respondiendo preguntas

- ¿Existen índices de obesidad inferiores a 24? ¿Cuáles?

Lo anterior es equivalente a preguntarse cuántos valores `TRUE` aparecen al realizar la prueba lógica:

```
imc<24
[1] TRUE FALSE FALSE FALSE TRUE TRUE FALSE
```

`TRUE` indica que sí se cumple la prueba lógica. Por ejemplo, el primer elemento al que se realiza la prueba lógica es `TRUE`, esto indica, que el primer elemento del vector `imc` es menor que 24 (es decir, el primer paciente presenta un imc inferior a 24).

Función `which()`

El comando `which()` *selecciona sólo los elementos* `TRUE` *de una prueba lógica, ¡este comando es muy útil cuando queremos saber qué elementos de un vector cumplen una condición!*

```
which(imc<24)
[1] 1 5 6
```

Lo anterior indica que los pacientes 1, 5 y 6 presentan un imc menor a 24.

Para conocer los imc de estos pacientes, podemos realizar lo siguiente:

```
imc[which(imc<24)]
[1] 19.5 23.0 23.0
```

Función `table()`

Si también queremos conocer cuántos registros no cumplen la condición, podemos utilizar `table()`

```
table(imc<24)

FALSE  TRUE
    4     3
```

Lo anterior indica que existen 4 pacientes que no presentan un imc menor a 24 (es decir, poseen un imc mayor o igual a 24) y 3 pacientes que sí poseen imc inferior a 24.

Respondiendo preguntas

- ¿Qué índices cumplen que son mayores a 20 y menores que 25?

Necesitamos utilizar el operador **&** pues estamos pidiendo una condición y, simultáneamente, otra.

Respondiendo preguntas

- ¿Qué índices cumplen que son mayores a 20 y menores que 25?

```
(imc>20)&(imc<25)
[1] FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE

which((imc>20)&(imc<25))
[1] 5 6
```

Esto significa que los pacientes 5 y 6 posee un imc entre 20 y 25.

Tipo de vector: lógico

Conocemos ya los vectores de tipo `numeric` y `character`, un vector de valores lógicos es un vector de tipo `logical`:

```
class((imc>20)&(imc<25))
[1] "logical"
```

TRUE=1 y FALSE=0

Cuando utilizamos la función `as.numeric()` en un vector lógico obtenemos un vector con números 0 y 1:

```
(imc>20)&(imc<25)
[1] FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE

as.numeric((imc>20)&(imc<25))
[1] 0 0 0 0 1 1 0
```

El valor lógico `FALSE` es equivalente al valor `0` y el valor lógico `TRUE` es equivalente al valor 1. Dicho esto, podríamos perfectamente aplicar funciones matemáticas a vectores lógicos:

```
(imc>20)*(imc<25)
[1] 0 0 0 0 1 1 0

sum(imc<24)
[1] 3
```

Respondiendo preguntas

- ¿Existe algún índice que sea exactamente 24? ¿Y 27?

Respondiendo preguntas

- ¿Existe algún índice que sea exactamente 24? ¿Y 27?

```
imc==24
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE

which(imc==24)
integer(0)
```

No existe ningún paciente que posea un imc exactamente igual a 24.

Respondiendo preguntas

- ¿Existe algún índice de obesidad que sea exactamente 24? ¿Y 27?

```
imc==27
[1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE

sum(imc==27)
[1] 1
```

Existe tan sólo un paciente que posee un imc igual a 27.

Respondiendo preguntas

- ¿Existe algún índice que se repita? ¿cuál(es)?

```
table(imc)

imc
19.5   23   25   27   35
   1    2    2    1    1
```

Con la función `table()` podemos ver cómo se distribuyen los imc. Esta función es muy útil entrega un conteo de los valores registrados. Podemos ver que hay dos índices que se repiten, 23 y 25 se repiten dos veces cada uno. Es decir, existen dos pacientes cuyo imc es igual a 23 y dos pacientes cuyo imc es igual a 25.

Respuestas práctica

Respuestas práctica 1

```
substr(nacimientos, start=4, stop= nchar(nacimientos)-5) #extrae el mes
[1] "04" "03" "01" "11"

(meses<-as.numeric(substr(nacimientos, start=4, stop= nchar(nacimientos)-5))) #Lo asignamos a un objeto
[1]  4  3  1 11

class(meses)
[1] "numeric"
```

Cada vez que queramos cambiar el formato y que sea numérico debemos utilizar `as.numeric()`. Recordar que a los vectores numéricos podemos realizarles múltiples operaciones, suma, multiplicación, ordenarlos de menor a mayor, etcétera, pero a un vector en formato `character` no es posible realizarle dichas operaciones. Distinto tipo de vector, distinto tipo de tratamiento.

Respuestas practica 2: Función `union()`

Primero debemos definir dos vectores que contengan la información de los inscritos:

```
link<-c("Natalie J.", "Vanesa R.", "Sergio M.", "Felipe C.", "Catalina B.", "Cristobal B.")
encuesta<-c("Catalina B.", "Fernanda C.", "Natalie J.", "Sergio B.", "Samuel C.",  "Antonia K.", "Joaquin E.")
```

Note que hay ciertas personas que aparecen en ambos listados (*seguramente olvidaron que ya se habían inscrito!*). ¿Cómo obtener el listado de las personas pero sin repetir a ninguna? Con la función `union()`:

```
union(link, encuesta)
 [1] "Natalie J."   "Vanesa R."    "Sergio M."    "Felipe C."    "Catalina B."  "Cristobal B." "Fernanda C."
 [8] "Sergio B."    "Samuel C."    "Antonia K."   "Joaquin E."
```

La función `union()` obtiene la unión de los elementos de ambos vectores. Para saber el total de inscritos, basta con utilizar la función `length()`:

```
length(union(link, encuesta))
[1] 11
```

Hay 11 personas inscritas en el evento.

Respuestas practica 2: Función `intersect()`

Si queremos saber quiénes son las personas olvidadizas que se inscribieron dos veces, podemos obtener la intersección entre ambos vectores, es decir, cuáles son los elementos que se repiten en ambos vectores (que se inscribieron tanto vía link como vía encuesta):

```
intersect(link, encuesta)
[1] "Natalie J."  "Catalina B."
```

Natalie J. y Catalina B. fueron quienes se inscribieron dos veces. (*Ups, olvidé que ya me había inscrito!*)

Respuestas practica 2: Función `setdiff()`

Con la función `setdiff()` podemos saber qué elementos están en un vector pero no en el otro (en este caso sería quiénes se inscribieron en una vía pero no en la otra):

```
setdiff(link,encuesta) #Personas inscritas via link pero no via encuesta
[1] "Vanesa R."    "Sergio M."    "Felipe C."    "Cristobal B."

setdiff(encuesta, link) #Personas inscritas via encuesta pero no via link
[1] "Fernanda C." "Sergio B."   "Samuel C."   "Antonia K."  "Joaquin E."

union(setdiff(link,encuesta),setdiff(encuesta, link)) #Entrega todos los participantes que se registraron en sólo una vía
[1] "Vanesa R."    "Sergio M."    "Felipe C."    "Cristobal B." "Fernanda C."  "Sergio B."    "Samuel C."
[8] "Antonia K."   "Joaquin E."
```