Manipulación de datos en R

Clase 2

Joaquin Cavieres G.

Estudiante doctorado

j.cavieres.g@gmail.com

24 de octubre de 2019

Subconjunto de vectores

```
x <- c(3, 4, 2, 2, 10, 7)
> x[1]                    # Parentesis cuadrado accede al indice del vector.
[1] 3
> x[2]
[1] 4
> x[1:5]
[1] 1 4 2 2 10
> x[c(2,5])
[1] 4 10
```

Operadores boleanos

- $<$ Menor que
- $<$ Mayor que
- $<=$ Menor o igual que
- $=>$ Mayor o igual que
- $==$ Igual a
- $!=$ No igual a
- & Y
- | O
- ! No

Operadores boleanos

Ejemplos con operadores boleanos

```
x <- 3
```

```
> x == 3
```

```
[1] TRUE
```

```
> x < -1
```

```
[1] FALSE
```

```
> x > 0 & x < 10
```

```
[1] TRUE
```

Operadores boleanos

```
x <- 1 : 5

> x == 3

[1] FALSE FALSE TRUE FALSE FALSE

> x < 10

[1] TRUE TRUE TRUE TRUE TRUE

> x > 2 & x <= 4

[1] FALSE FALSE TRUE TRUE FALSE
```

Operaciones lógicas

```
día <- c(‘‘Lunes’’, ‘‘Martes’’, ‘‘Miércoles’’, ‘‘Jueves’’,
‘‘Viernes’’, ‘‘Sábado’’, ‘‘Domingo’’)

lluvia <- c(rep(‘‘Si’’, 6), ’’No’’)

nieve <- c(rep(‘‘No’’, 3), ‘‘Si’’, rep(‘‘No’’,3))

> lluvia == ‘‘Si’’

[1] TRUE TRUE TRUE TRUE TRUE TRUE FALSE

> lluvia !=’’No’’

[1] TRUE TRUE TRUE TRUE TRUE TRUE FALSE

> nieve == ‘‘Si’’

[1] FALSE FALSE FALSE TRUE FALSE FALSE FALSE
```

Operaciones lógicas

¿Cuantos días llueven a la semana?

```
> sum(lluvia==''Si'')
```

```
[1] 6
```

¿Cuantos TRUE/FALSE son representados numericamente?

```
> as.numeric(lluvia==''Si'')
```

```
[1] 1 1 1 1 1 1 0
```

Operaciones lógicas

Subconjuntos de vectores de acuerdo a operaciones lógicas

```
> día[lluvia==''Si'']
```

```
[1] ''Lunes'', ''Martes'', ''Miércoles'', ''Jueves'',
''Viernes'', ''Sábado''
```

```
> día[nieve==''Si'']
```

```
[1] ''Miércoles''
```

¿Que día llevo un paraguas si llueve o nieva?

```
> día[nieve == ''Si'' & nieve == ''Si'']
```

```
[1] ''Miércoles''
```

Operaciones lógicas

Las funciones a continuación pueden servir de ayuda para responder preguntas sobre observaciones especificas.

- Retornar lo índices correspondientes a la pregunta lógica

```
> which(lluvia==''Si'')
```

```
[1] 1 2 3 4 5 6
```

```
> any(lluvia == ''Si'']        # ¿Alguno de los días llueve?
```

```
[1] TRUE
```

```
> all(lluvia == ''Si'']        # ¿Todos los días llueve?
```

```
[1] FALSE
```

Ejercicios

```
y <- c(3, 2, 15, -1, 22, 1, 9, 10, 17, 5)
```

Realice los siguientes cálculos de acuerdo al vector "y"

- Obtenga el primer y último valor
- Mostrar los valores que son mayores a la media de "y"
- ¿Son todos los valores positivos?
- ¿Alguno de los valores de "y" es igual a la media o la mediana?

DATA FRAMES

Esto es conveniente para almacenar en un objeto una colección de datos.

```
> nyears <- length(co2)          # Número de años con observaciones
```

```
> years <- seq(from=1959, length=nyears)
```

```
> co2Data <- data.frame(years, co2)     # "years" y "co2" deben ser iguales
```

Utilizamos la función `head()` para vizualizar la data creada.

```
> head(co2Data)
```

	years	co2
1	1959	316.00
2	1960	315.91
3	1961	317.63
4	1962	318.46
5	1963	319.52

DATA FRAMES

Es mucho más comodo almacenar nuestros datos en el directorio.

```
> year
```

DATA FRAMES

Es mucho más comodo almacenar nuestros datos en el directorio.

```
> year

Error: object ‘‘year’’ not found

> ls()
```

DATA FRAMES

Es mucho más comodo almacenar nuestros datos en el directorio.

```
> year

Error: object ‘‘year’’ not found

> ls()

[1] ‘‘co2’’ ‘‘co2Data’’ ‘‘day’’
```

DATA FRAMES

Es mucho más comodo almacenar nuestros datos en el directorio.

```
> year
```

```
Error: object ‘‘year’’ not found
```

```
> ls()
```

```
[1] ‘‘co2’’ ‘‘co2Data’’ ‘‘day’’
```

Debemos usar el operador $ para extraer la variable que queremos desde el `data.frame`

```
> co2Data$year
```

DATA FRAMES

Es mucho más comodo almacenar nuestros datos en el directorio.

```
> year

Error: object ‘‘year’’ not found

> ls()

[1] ‘‘co2’’ ‘‘co2Data’’ ‘‘day’’
```

Debemos usar el operador $ para extraer la variable que queremos desde el `data.frame`

```
> co2Data$year

[1] 1959 1960 1961....
```

Extraer elementos de un data.frame

```
indice <- c(2, 3, 5, 7, 8, 9, 15, 21, 23, 26)

peso <- c(14.8, 21, 19.7, 23.2, 16, 16.1, 20, 29.3, 17.8,
21.2)

condición <- c("good", "fair", "fair", "poor", "fair",
"good", "good", "fair", "fair", "poor")
```

Extraer elementos de un `data.frame`

```
indice <- c(2, 3, 5, 7, 8, 9, 15, 21, 23, 26)

peso <- c(14.8, 21, 19.7, 23.2, 16, 16.1, 20, 29.3, 17.8,
21.2)

condición <- c("good", "fair", "fair", "poor", "fair",
"good", "good", "fair", "fair", "poor")
```

¿Como uno estos 3 vectores creados?

Extraer elementos de un data.frame

```
indice <- c(2, 3, 5, 7, 8, 9, 15, 21, 23, 26)

peso <- c(14.8, 21, 19.7, 23.2, 16, 16.1, 20, 29.3, 17.8,
21.2)

condición <- c("good", "fair", "fair", "poor", "fair",
"good", "good", "fair", "fair", "poor")
```

¿Como uno estos 3 vectores creados?

```
exampleData <- data.frame(indice, peso, condición)

> head(exampleData)
```

Extraer elementos de un `data.frame`

	indice	peso	condición
1	2	14.8	good
2	3	21.0	fair

Extraer elementos de un `data.frame`

Extraer la columna con el nombre “peso”.

Extraer elementos de un `data.frame`

Extraer la columna con el nombre “peso”.

```
> exampleData$peso
```

Extraer elementos de un `data.frame`

Extraer la columna con el nombre “peso”.

```
> exampleData$peso
```

```
[1] 14.8, 21, 19.7, 23.2, 16, 16.1, 20, 29.3, 17.8,
21.2.
```

Extraer elementos de un `data.frame`

Extraer la columna con el nombre "peso".

```
> exampleData$peso
```

```
[1] 14.8, 21, 19.7, 23.2, 16, 16.1, 20, 29.3, 17.8,
21.2.
```

Cambiar los valores de los elementos del vector "peso" **NO** cambia los valores de `> exampleData$peso`.

Extraer elementos de un `data.frame`

Extraer la columna con el nombre "peso".

```
> exampleData$peso
```

```
[1] 14.8, 21, 19.7, 23.2, 16, 16.1, 20, 29.3, 17.8,
21.2.
```

Cambiar los valores de los elementos del vector "peso" **NO** cambia los valores de `> exampleData$peso`.

```
> peso <- rep(20, 10) # repetir 20 veces el numero 10.
```

```
[1] 20 20 20 20 20 20 20 20 20 20.
```

```
> exampleData$peso
```

Extraer elementos de un `data.frame`

Extraer la columna con el nombre "peso".

```
> exampleData$peso
```

```
[1] 14.8, 21, 19.7, 23.2, 16, 16.1, 20, 29.3, 17.8,
21.2.
```

Cambiar los valores de los elementos del vector "peso" **NO** cambia los valores de `> exampleData$peso`.

```
> peso <- rep(20, 10) # repetir 20 veces el numero 10.
```

```
[1] 20 20 20 20 20 20 20 20 20 20.
```

```
> exampleData$peso
```

```
[1] 14.8, 21, 19.7, 23.2, 16, 16.1, 20, 29.3, 17.8,
21.2.
```

Extraer elementos de un `data.frame`

¿Como puedo elegir una fila y una columna de un `data.frame`?

`objeto[fila, columna]`

Extraigamos la información relacionada al "peso" del `data.frame`.

```
> exampleData[,2]
```

```
[1] 14.8 21.0 19.7 23.2 16.0 16.1 20.0 29.3 17.8 21.2
```

Podemos excluir columnas del `data.frame`

```
> exampleData[,-1]
```

	peso	condición
1	14.8	good
2	21.0	fair

Extraer elementos de un `data.frame`

Extraer la primera fila.

```
> exampleData[1,]
```

	indice	peso	condición
1	2	14.8	good

Extraer la primera y tercera fila.

```
> exampleData[c(1,3), ]
```

	indice	peso	condición
1	2	14.8	good
4	7	23.2	poor

Extraer elementos de un `data.frame`

Extraer la segunda y tercera columna.

```
> exampleData[,2:3]
```

	peso	condición
1	14.8	good
2	21.0	fair

Extraer la columna por su nombre

```
> exampleData[, c(``indice'', ``condicion'')]
```

	indice	condición
1	2	good
2	3	fair

Extraer elementos de un `data.frame`

Usar operadores lógicos para extraer información.

```
> exampleData[c(4,8), ] # Filas 4 y 8.
```

	indice	peso	condición
4	7	23.2	good
8	21	29.3	fair

```
> exampleData[exampleData$peso > 22 ]
```

	indice	peso	condición
4	7	23.2	good
8	21	29.3	fair

Extraer elementos de un `data.frame`

Mas de una declaración condicional

```
> exampleData[exampleData$peso < 20 & exampleData$condición ==
‘‘fair’’,]
```

	indice	peso	condición
3	5	19.7	fair
5	8	16.0	fair
9	23	17.8	fair

```
> exampleData[exampleData$peso < 15 | exampleData$peso > 25,]
```

	indice	peso	condición
1	2	14.8	good
8	21	29,3	fair

Dimesión de los datos

- La función `length` entrega la dimensión del vector, pero en un `data.frame` esta función retorna el numero de columnas.

```
> length(exampleData)
```

```
[1] 3
```

- La función `dim` entrega la dimensión de filas y columnas.

```
> dim(exampleData)
```

```
[1] 10 3.
```

- `nrow` y `ncol` entrega la dimensión individual de cada uno.

```
> nrow(exampleData)
```

```
[1] 10.
```

```
> ncol(exampleData)
```

```
[1] 3.
```

Ejercicios

De acuerdo a:

```
pacientes <- data.frame(id = c(31, 62, 50, 99, 53, 75, 54
, 58, 4, 74),
edad = c(12, 18, 20, 17, 14, 8, 12, 24, 24, 21),
sexo = c("M", "F", "F", "M", "F", "M", "M", "F", "F", "M") )
```

- Mostrar edades mayor a 20
- Usar un operador lógico para mostrar las variables “id” y “sexo”
- Mostrar sólo las observaciones de hombres

Valores perdidos (NA)

Por ejemplo, si tenemos el vector:

Valores perdidos (NA)

Por ejemplo, si tenemos el vector:

```
capturas <- c(20, 35, 14, NA, 53, 22, 45)
```

Valores perdidos (NA)

Por ejemplo, si tenemos el vector:

```
capturas <- c(20, 35, 14, NA, 53, 22, 45)
```

- Se puede apreciar que existen `NA's`, que significan valores "perdidos", y pueden corresponder a valores no observados en una determinada muestra.

Valores perdidos (NA)

Por ejemplo, si tenemos el vector:

```
capturas <- c(20, 35, 14, NA, 53, 22, 45)
```

- Se puede apreciar que existen `NA's`, que significan valores "perdidos", y pueden corresponder a valores no observados en una determinada muestra.
- Esto puede generar problemas si quieremos hacer algun cálculo sobre los valores del vector, por ejemplo:

Valores perdidos (NA)

Por ejemplo, si tenemos el vector:

```
capturas <- c(20, 35, 14, NA, 53, 22, 45)
```

- Se puede apreciar que existen `NA's`, que significan valores "perdidos", y pueden corresponder a valores no observados en una determinada muestra.
- Esto puede generar problemas si quieremos hacer algun cálculo sobre los valores del vector, por ejemplo:

```
> mean(capturas)
```

Valores perdidos (NA)

Por ejemplo, si tenemos el vector:

```
capturas <- c(20, 35, 14, NA, 53, 22, 45)
```

- Se puede apreciar que existen `NA's`, que significan valores "perdidos", y pueden corresponder a valores no observados en una determinada muestra.
- Esto puede generar problemas si quieremos hacer algun cálculo sobre los valores del vector, por ejemplo:

```
> mean(capturas)
```

```
[1] NA
```

Valores perdidos (NA)

Por ejemplo, si tenemos el vector:

```
capturas <- c(20, 35, 14, NA, 53, 22, 45)
```

- Se puede apreciar que existen `NA's`, que significan valores "perdidos", y pueden corresponder a valores no observados en una determinada muestra.
- Esto puede generar problemas si quieremos hacer algun cálculo sobre los valores del vector, por ejemplo:

```
> mean(capturas)
```

```
[1] NA
```

- Para remover los `Na's`, escribimos:

Valores perdidos (NA)

Por ejemplo, si tenemos el vector:

```
capturas <- c(20, 35, 14, NA, 53, 22, 45)
```

- Se puede apreciar que existen `NA's`, que significan valores "perdidos", y pueden corresponder a valores no observados en una determinada muestra.
- Esto puede generar problemas si quieremos hacer algun cálculo sobre los valores del vector, por ejemplo:

```
> mean(capturas)
```

```
[1] NA.
```

- Para remover los `Na's`, escribimos:

```
> mean(capturas, na.rm=TRUE)
```

Valores perdidos (NA)

Por ejemplo, si tenemos el vector:

```
capturas <- c(20, 35, 14, NA, 53, 22, 45)
```

- Se puede apreciar que existen `NA's`, que significan valores "perdidos", y pueden corresponder a valores no observados en una determinada muestra.
- Esto puede generar problemas si quieremos hacer algun cálculo sobre los valores del vector, por ejemplo:

```
> mean(capturas)
```

```
[1] NA.
```

- Para remover los `Na's`, escribimos:

```
> mean(capturas, na.rm=TRUE)
```

```
[1] 31.5.
```

Valores perdidos (NA)

- Omitir los `Na's`.

Valores perdidos (NA)

- Omitir los `Na's`.

```
> na.omit(capturas)
```

Valores perdidos (NA)

- Omitir los `Na's`.

```
> na.omit(capturas)
```

```
[1] 20, 35, 14, 53, 22, 45.
```

Valores perdidos (NA)

- Omitir los `Na's`.

```
> na.omit(capturas)
```

```
[1] 20, 35, 14, 53, 22, 45.
```

- También existen las funciones `na.exclude()`, `na.fail()`, `na.pass()`

Valores perdidos (NA)

- Omitir los `Na's`.

```
> na.omit(capturas)
```

```
[1] 20, 35, 14, 53, 22, 45.
```

- También existen las funciones `na.exclude()`, `na.fail()`, `na.pass()`
- `>!is.na()` es otra forma de manejar los `NA's` en un vector.

Valores perdidos (NA)

- Omitir los `Na's`.

```
> na.omit(capturas)
```

```
[1] 20, 35, 14, 53, 22, 45.
```

- También existen las funciones `na.exclude()`, `na.fail()`, `na.pass()`
- `>!is.na()` es otra forma de manejar los `NA's` en un vector.

```
> capturas[!is.na(capturas)]
```

Valores perdidos (NA)

- Omitir los `Na's`.

```
> na.omit(capturas)
```

```
[1] 20, 35, 14, 53, 22, 45.
```

- También existen las funciones `na.exclude()`, `na.fail()`, `na.pass()`
- `>!is.na()` es otra forma de manejar los `NA's` en un vector.

```
> capturas[!is.na(capturas)]
```

```
[1] 20, 35, 14, 53, 22, 45.
```