

Introducción a



3. Manejo básico de datos

3.3. Data frame

Índice

3.3. Data frame

3.3.1. Data frame

3.3.2. Operaciones con variables (vectores)

3.3.3. Generación de secuencias

3.3.4. Missing

3.3.5. Ordenación

3.3.6. Acceso a elementos

datos.curso1.RData

```
'data.frame': 200 obs. of 11 variables:  
 $ ID      : num  137 174 200 23 39 90 40 115 72 27 ...  
 $ edad    : num  37 85 29 13 49 12 85 31 39 70 ...  
 $ sexo    : chr  "Mujer" "Mujer" "Hombre" "Hombre" ...  
 $ estado.civil : chr  "Casado" "Soltero" "Casado" "Divorciado" ...  
 $ nivel.estudios : chr  "Bajo" "Alto" "Bajo" "Alto" ...  
 $ peso     : num  59.6 60 79.2 80.8 80.8 ...  
 $ altura   : num  151 149 169 171 171 ...  
 $ fumador  : chr  "No" "No" "No" "Si" ...  
 $ diabetes : chr  "No" "Si" "Si" "Si" ...  
 $ cancer.mama : chr  "Si" "No" "Si" NA ...  
 $ cancer.prostata: chr  NA NA "Si" "Si" ...
```

	ID	edad	sexo	estado.civil	nivel.estudios	peso	altura	fumador	diabetes	cancer.mama	cancer.prostata
1	137	37	Mujer	Casado	Bajo	59.58221	150.7163	No	No	Si	<NA>
2	174	85	Mujer	Soltero	Alto	59.95427	149.2075	No	Si	No	<NA>
3	200	29	Hombre	Casado	Bajo	79.20674	168.9795	No	Si	Si	Si
4	23	13	Hombre	Divorciado	Alto	80.78347	171.1568	Si	Si	<NA>	Si
5	39	49	Hombre	Divorciado	Bajo	80.76036	170.5682	Si	Si	<NA>	No
6	90	12	Hombre	Casado	Alto	79.83426	170.8565	No	No	<NA>	Si
7	40	85	Hombre	Casado	Alto	80.69636	168.5586	No	No	<NA>	Si
8	115	31	Mujer	Soltero	Alto	61.28985	150.0667	Si	No	Si	<NA>
9	72	39	Mujer	Divorciado	Bajo	60.70871	150.4091	Si	Si	Si	<NA>
10	27	70	Hombre	Soltero	Medio	78.89874	167.8307	No	Si	<NA>	Si
11	19	24	Mujer	Divorciado	Alto	60.06984	149.5328	No	No	Si	<NA>
12	133	45	Hombre	Soltero	Medio	79.57263	170.7013	Si	Si	<NA>	Si
13	15	42	Mujer	Soltero	Bajo	60.39387	150.7226	Si	No	Si	<NA>
14	44	74	Mujer	Casado	Medio	60.33449	148.2137	No	Si	Si	<NA>
15	179	16	Mujer	Divorciado	Medio	60.01237	151.4407	Si	Si	Si	<NA>
16	148	6	Mujer	Soltero	Bajo	57.93049	149.7306	No	No	Si	<NA>
17	192	31	Mujer	Casado	Bajo	57.99527	148.3326	Si	Si	Si	<NA>
18	186	83	Mujer	Soltero	Bajo	60.44401	150.3675	Si	Si	Si	<NA>
19	18	32	Hombre	Divorciado	Alto	79.39825	168.9841	Si	No	<NA>	No
20	106	67	Hombre	Soltero	Medio	80.07338	170.6679	No	Si	<NA>	Si
21	86	54	Mujer	Divorciado	Medio	60.02079	150.1170	Si	Si	No	<NA>
22	55	18	Hombre	Soltero	Bajo	78.30574	170.9642	No	Si	<NA>	Si
23	20	84	Hombre	Casado	Bajo	80.35391	168.2667	Si	Si	<NA>	No
24	102	40	Mujer	Soltero	Bajo	60.34935	150.3949	Si	No	No	<NA>
25	140	6	Hombre	Divorciado	Bajo	79.47181	170.0953	Si	No	<NA>	Si
26	112	63	Mujer	Soltero	Medio	60.05865	150.2747	No	Si	No	<NA>
27	183	79	Hombre	Soltero	Alto	79.69627	169.4480	Si	Si	<NA>	Si
28	120	27	Mujer	Divorciado	Alto	58.60144	150.2412	No	Si	Si	<NA>
29	117	34	Hombre	Soltero	Alto	80.76927	168.8787	No	No	<NA>	Si
30	130	53	Hombre	Divorciado	Alto	79.05485	170.1690	Si	No	<NA>	Si
31	144	23	Mujer	Divorciado	Alto	59.16448	149.9304	No	No	Si	<NA>
32	41	66	Hombre	Casado	Medio	79.35361	170.4784	Si	No	<NA>	No
33	193	29	Mujer	Casado	Alto	58.42305	149.1501	Si	No	Si	<NA>
34	12	54	Hombre	Soltero	Medio	78.61265	169.0950	Si	Si	<NA>	Si
35	108	32	Mujer	Casado	Bajo	60.56363	150.0661	No	Si	Si	<NA>
36	17	7	Mujer	Soltero	Bajo	58.65925	148.7323	Si	Si	<NA>	<NA>
37	157	42	Hombre	Soltero	Bajo	79.90425	169.8202	No	Si	<NA>	Si
38	122	68	Mujer	Casado	Alto	61.06287	149.0318	No	No	Si	<NA>
39	25	15	Hombre	Soltero	Alto	81.42743	169.8150	No	Si	<NA>	Si
40	50	6	Hombre	Casado	Bajo	80.68350	168.3866	Si	No	<NA>	Si
41	146	38	Mujer	Soltero	Bajo	58.74115	150.0431	No	Si	Si	<NA>
42	29	28	Hombre	Casado	Medio	80.60408	170.8330	Si	No	<NA>	Si
43	187	27	Mujer	Soltero	Alto	60.28175	149.2774	No	No	Si	<NA>
44	64	7	Mujer	Divorciado	Bajo	60.67000	151.5040	No	No	Si	<NA>
45	185	9	Hombre	Divorciado	Alto	82.13408	169.4244	No	Si	<NA>	Si
46	31	85	Mujer	Divorciado	Medio	59.54810	150.3059	No	No	Si	<NA>
47	21	24	Hombre	Casado	Alto	79.06357	170.3569	Si	No	<NA>	Si
48	78	57	Mujer	Soltero	Medio	58.93866	149.9912	No	Si	Si	<NA>
49	11	30	Hombre	Divorciado	Alto	81.73699	169.1571	Si	Si	<NA>	Si
50	87	84	Hombre	Soltero	Alto	80.20169	170.2267	Si	Si	<NA>	Si
51	116	64	Hombre	Divorciado	Medio	80.71775	167.6929	No	No	<NA>	Si
52	47	81	Mujer	Casado	Bajo	61.21061	149.0885	No	Si	Si	<NA>
53	8	38	Hombre	Casado	Alto	80.21772	170.2392	Si	Si	<NA>	Si
54	94	73	Hombre	Divorciado	Alto	79.47415	168.2748	No	Si	<NA>	Si
55	92	54	Mujer	Casado	Bajo	59.50153	150.7325	No	No	Si	<NA>
56	173	19	Hombre	Soltero	Alto	79.17450	170.0500	Si	No	<NA>	Si
57	175	77	Hombre	Soltero	Medio	79.05946	169.3775	Si	Si	<NA>	Si
58	79	29	Mujer	Soltero	Bajo	59.24916	149.8326	No	Si	Si	<NA>
59	5	81	Hombre	Soltero	Alto	80.11593	168.1224	No	No	<NA>	Si
60	36	20	Hombre	Casado	Alto	80.11153	168.8877	Si	Si	<NA>	Si
61	110	41	Mujer	Casado	Medio	60.35693	148.6183	No	Si	No	<NA>
62	106	44	Hombre	Casado	Alto	79.43454	168.3474	Si	No	<NA>	No

3.3.1. Data frame

data frame= Como array, pero permiten elementos (columnas) de distintos tipos. El objeto más habitual para manejo de datos procedentes de experimentos.

Ejemplos

```
> my.data.frame <- data.frame(ID = c("paciente1", "paciente2", "paciente3"),  
edad= c(10, 25, 33), altura = c(NA, 150, 180),  
cancer = c(TRUE, TRUE, FALSE),  
bmi = c(22,25, 20))
```

```
>load("datos.curso1.RData")
```

3.3.1. Data frame

Podemos utilizar la función `data.frame()` para crear nuestras tablas de resultados (**YA LO HEMOS VISTO**)

Esa tabla se puede exportar como veremos

Acceso a elementos:

a) A una variable: `nombre.data.frame$"nombre .variable"`

```
> datos$"sexo"
```

```
> sexo.mirar <- datos$"sexo"
```

b) Notacion matricial (`Nombre.data.frame[Filas, Columnas]`)

```
> datos[,3]; datos[1:3,3]; datos [1:10,1:3]
```

3.3.1. Data frame

Acceso a elementos:

c) Tanto en Filas como en Columnas: vector de indices(posiciones), vector lógico, condición lógica sobre otros vectores (variables), etc

```
> datos [datos$"sexo"=="Mujer", ]
```

```
> datos [datos$"sexo"=="Mujer" & datos$"estado.civil"=="Casado" ,1:10 ]
```

```
> datos [datos$"sexo">%in%“Mujer” & datos$"estado.civil">%in%“Casado” ,1:10 ]
```

```
> datos.new <- datos [datos$"sexo"=="Mujer", ]
```

```
> datos.new2<-datos
```

```
> datos.new2[1 , 3] <- "Hombre"
```

Recodificaciones

```
> datos.new2[datos.new2$"ID"==200 , sexo ]<-"Mujer"  
> datos.new2$"sexo"[datos.new2$"ID"==200]<-"Mujer"
```

3.3.1. Data frame

(1) Añadir variables (columnas):

a) nombre.data.frame\$"nombre.nueva.variable" <- valores

```
> datos$"caso.cancer" <- c(1,1,1.....)
```

b) Utilizar la funcion cbind()

```
> datos.nuevos1 <- datos[ ,1:3]
```

```
> datos.union1 <- cbind(datos,datos.nuevos1)
```

c) Utilizar la funcion merge() (YA LA VEREMOS)

3.3.1. Data frame

(2) Añadir registros (filas):

- a) Utilizar la funcion rbind()

```
> datos.nuevos2 <- datos[1:5, ]
```

```
> datos.union2 <- rbind(datos,datos.nuevos2)
```

(3) Eliminar variables o registros:

- a) nombre.data.frame [-vector de posiciones de las filas a eliminar, - vector de posiciones de las columnas a eliminar]

```
> datos.nuevos3 <- datos[-c(2,3), ]
```

```
> datos.nuevos4 <- datos[-c(2,3) , -c(14,18)]
```

3.3.1. Data frame

(3) Eliminar variables o registros:

b) nombre.data.frame [! condiciones lógicas, ! Condiciones logicas]

```
> indice5 <- datos$"sexo"!="Mujer"  
> datos.nuevos5 <- datos[indice5, ]  
  
> datos.nuevos6 <- datos[ , names(datos)!="peso"]  
  
> indice7 <- which(datos$"sexo"=="Mujer")  
> datos.nuevos7 <- datos[-indice7, ]
```

3.3.1. Data frame

Visualización de los data frames:

```
> fix(datos)
```

```
> str(datos)
```

```
> str(datos)
'data.frame': 200 obs. of 11 variables:
 $ ID      : num 137 174 200 23 39 90 40 115 72 27 ...
 $ edad    : num 37 85 29 13 49 12 85 31 39 70 ...
 $ sexo    : chr "Mujer" "Mujer" "Hombre" "Hombre" ...
 $ estado.civil : chr "Casado" "Soltero" "Casado" "Divorciado" ...
 $ nivel.estudios: chr "Bajo" "Alto" "Bajo" "Alto" ...
 $ peso     : num 59.6 60 79.2 80.8 80.8 ...
 $ altura   : num 151 149 169 171 171 ...
 $ fumador  : chr "No" "No" "No" "Si" ...
 $ diabetes : chr "No" "Si" "Si" "Si" ...
 $ cancer.mama: chr "Si" "No" "Si" NA ...
 $ cancer.prostata: chr NA NA "Si" "Si" ...
```

```
> head(datos,3); tail(datos,3)
```

```
> head(datos,3)
  ID edad sexo estado.civil nivel.estudios     peso    altura fumador diabetes cancer.mama cancer.prostata
1 137   37 Mujer      Casado       Bajo 59.58221 150.7163      No      No        Si          <NA>
2 174   85 Mujer      Soltero      Alto 59.95427 149.2075      No      Si        No          <NA>
3 200   29 Hombre     Casado       Bajo 79.20674 168.9795      No      Si        Si          Si
> tail(datos,3)
  ID edad sexo estado.civil nivel.estudios     peso    altura fumador diabetes cancer.mama cancer.prostata
198  57   41 Hombre     Soltero      Medio 79.45077 168.4157      No      No        <NA>          No
199 169   44 Mujer      Soltero      Bajo 59.69171 149.0113      Si      No        No          <NA>
200 143   54 Mujer      Divorciado    Medio 61.02373 150.8797      Si      Si        Si          <NA>
```

Ejercicio 8

EJERCICIOS
“ejercicios.manejo.basico.datos.r”

3.3.1. Data frame

Otra forma de obtener subsets de data frames (sin incluir los NA):

```
> datos.mujer1 <- datos[datos$"sexo"%in%"Mujer", -c(6,7) ]
```

```
> datos.mujer2 <- subset(datos, sexo=="Mujer",select=-c(peso, altura))
```

	ID	edad	sexo	estado.civil	nivel.estudios	fumador	diabetes	cancer.mama	cancer.prostata
2	174	85	Mujer	Soltero	Alto	No	Si	No	<NA>
8	115	31	Mujer	Soltero	Alto	Si	No	Si	<NA>
9	72	39	Mujer	Divorciado	Bajo	Si	Si	Si	<NA>
11	19	24	Mujer	Divorciado	Alto	No	No	Si	<NA>
13	15	42	Mujer	Soltero	Bajo	Si	No	Si	<NA>
14	44	74	Mujer	Casado	Medio	No	Si	Si	<NA>

3.3.1. Data frame

Observaciones generales

```
> names(datos)
[1] "ID"          "edad"        "sexo"        "estado.civil"  "nivel.estudios" "peso"
[7] "altura"      "fumador"     "diabetes"    "cancer.mama"   "cancer.prostata"
> names(datos)[3]
[1] "sexo"
> names(datos)[3]<- "sex"
> nombres.variables<-names(datos)
```

```
> row.names(datos)
[1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15" "16" ...
> mujeres<-datos[datos$sexo%in%"Mujer",]
> row.names(mujeres)
[1] "1"  "2"  "8"  "9"  "11" "13" "14" "15" "16" "17"...
> row.names(mujeres)<-NULL
> row.names(mujeres)
[1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"
```

Índice

3.3. Data frame

3.3.1. Data frame

3.3.2. Operaciones con variables (vectores)

3.3.3. Generación de secuencias

3.3.4. Missing

3.3.5. Ordenación

3.3.6. Acceso a elementos

3.3.2. Operaciones con vectores

- A. Operaciones aritméticas
- B. Operadores comparativos y lógicos
- C. Operaciones con conjuntos

A. Operaciones aritméticas con vectores (I)

- suma **+**, resta **-**, multiplicación *****, división **/**
- potencia **^**, raíz cuadrada **sqrt**
- %/%** división entera, **%%** módulo: resto de la división entera
- logaritmos **log**, **log10**, **log2**, **logb(x, base)**, exponencial **exp**
- trigonométricas **sin**, **cos**, **tan**, **asin**, **acos**, **atan**
- otras:
 - max**, **min**, **range**, **pmax**, **pmin**, **mean**, **median**, **var**, **sd**, **quantile**.
 - sum**, **prod**, **diff**, **cumsum**, **cumprod**, **cummax**, **cummin**.

A. Operaciones aritméticas con vectores (II)

- **FUNDAMENTAL:** R puede operar sobre vectores enteros de un golpe.

```
> x <- 1:10
```

```
> y <- x/2; z <- x^2; w <- y + z
```

- Si un elemento es más corto, se "recicla". Es "intuitivo" si la operación es escalar con vector. Pero también ocurre en operaciones entre vectores.

```
> x + 15
```

```
> x2 <- 1:5
```

```
> x + x2
```

- El reciclado cuando (vector, escalar) suele ser lo que queremos. Entre vectores no siempre: cuidado. (R da un warning, de momento –S4 classes dan un error).

A. Operaciones aritméticas con vectores (III)

- Operar sobre vectores enteros es **MUCHO MEJOR** que usar loops:
 - Mucho más claro:
 - Es la forma natural de operar sobre objetos enteros.
 - Código más fácil de entender.
 - Más sencillo de modificar y mantener.
 - Más fácil hacer debugging.
 - Más rápido de escribir (y no sólo porque muchas menos líneas!).
 - Más eficiente (en tiempo y memoria).

B. Operadores comparativos y lógicos

- <, >, <=, >=, ==, !=
- !, &, |, xor() y los parecidos &&, ||

```
x <- 5; x < 5; x >= 5; x == 6; x != 5
y <- c(TRUE, FALSE); !y; z <- c(TRUE, TRUE)
xor(y, z)
y & z; y | z
```

- Las formas &&, || son "short-circuit operators" que se suelen usar dentro de if statements. Se aplican a vectores de longitud uno y sólo evalúan el segundo argumento si es preciso.

C. Operaciones de conjuntos

```
> x <- 1:5; y <- c(1, 3, 7:10)
```

```
> union(x, y)
```

```
> intersect(x, y)
```

```
> setdiff(y, x)
```

```
> v <- c("bcA1", "bcA2", "blX1")
```

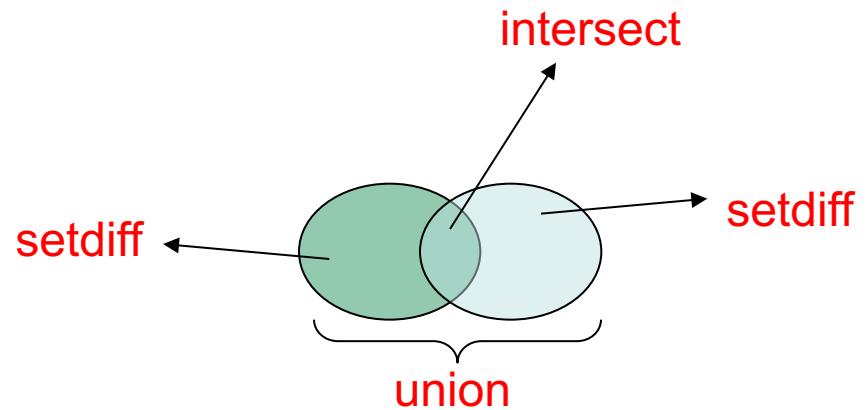
```
> w <- c("bcA2", "xA3")
```

```
> union(v, w)
```

```
> intersect(v, w)
```

```
> setdiff(w, v)
```

```
> setdiff(v, w)
```



3.3.3. Generación de secuencias

A. Secuencias aleatorias

A.1) NO basadas en una distribución de probabilidad definida

A.2) Basadas en una distribución de probabilidad definida

B. Secuencias NO aleatorias

A.Secuencias aleatorias (I)

A.1) NO Basadas en una distribución de probabilidad definida

- Muestrear un vector (imprescindible en bootstrapping y cross-validation):
 - **sample ()**

A. Secuencias aleatorias (II)

A.2) Basadas en una distribución de probabilidad definida

- Números aleatorios: usar **rFuncionDistribucion** (con sus parámetros).

```
> rnorm(10)
> rnorm(10, mean = 13, sd = 18)
> runif(15)
> runif(8, min = 3, max = 59)
```
- Existen muchas funciones de distribución: *lognormal*, *t*, *F*, *χ2*, *exponential*, *gamma*, *beta*, *poisson*, *binomial*, etc.
- Ver, por **ejemplo**, sección 8.1 (p. 34) de *An introduction to R* y p. 16 de *R para principiantes*, de E. Paradis. (Con esas funciones podemos también obtener la densidad, quantiles y función cumulativa de todas estas distribuciones).

B.Secuencias NO aleatorias

- Función seq()

```
> x <- seq(from = 2, to = 18, by = 2)
```

- Función rep()

```
> x <- rep( 1,10)
```

3.3.4. Missing

3.3.4. Missing

- **NA** es el código de “Not available”.

```
> v <- c(1,6,9,NA)  
> is.na(v);  
> which(is.na(v))
```

!!!!ATENCIÓN!!!!



```
> w <- v[!is.na(v)] # sin los valores perdidos  
> v == NA # !No funciona!
```

- **Sustituir NA** por, p.ej., 0:

```
> v[is.na(v)] <- 0
```

- El infinito y NaN (“not a number”) son diferentes de NA.

```
> is.infinite(-5/0); is.nan(0/0); is.na(5/0)
```

3.3.4. Missing

- Con algunas funciones
 - > **xna <- c(1, 2, 3, NA, 4); mean(xna) ## Esto da “NA”**
 - > **mean(xna, na.rm = TRUE)**
- Para “modelling functions” (ej. lm) lo mejor es usar:

na.omit

na.exclude==Esta última es más conveniente para generar predicciones, residuos, etc.

- Eliminar todos los NA:

```
> XNA <- matrix(c(1,2,NA,3,NA,4), nrow = 3)
> XNA
> X.no.na <- na.omit(XNA)
```

3.3.4. Missing

```
> sum(is.na(datos$sex))
[1] 0
```

```
> datos.con.missing<-datos
> datos.con.missing$peso[datos.con.missing$"peso"<60]<-NA
> sum(is.na(datos.con.missing$"peso"))
[1] 51
```

```
summary(datos.con.missing)
```

```
> summary(datos.con.missing)
      ID          edad         sexo        estado.civil    nivel.estudios      peso        altura
Min. : 1.00  Min. : 1.00  Length:200  Length:200  Length:200  Min.   :60.01  Min.   :147.8
1st Qu.: 50.75 1st Qu.:21.00  Class :character  Class :character  Class :character  1st Qu.:61.17  1st Qu.:150.1
Median :100.50 Median :42.00  Mode  :character  Mode  :character  Mode  :character  Median :79.39  Median :159.8
Mean   :100.50 Mean   :42.17                 Mode  :character  Mode  :character  Mean   :73.67  Mean   :159.8
3rd Qu.:150.25 3rd Qu.:63.00                 Mode  :character  Mode  :character  3rd Qu.:80.25  3rd Qu.:169.7
Max.   :200.00 Max.   :85.00                 Class :character  Class :character  Max.   :82.28  Max.   :172.9
                                         NA's   :51

      fumador        diabetes       cancer.mama      cancer.prostata
Length:200  Length:200  Length:200  Length:200
Class :character  Class :character  Class :character  Class :character
Mode  :character  Mode  :character  Mode  :character  Mode  :character
```

3.3.5. Ordenación

Ordenando vectores

➤ **order()**

```
> datos$"edad"  
[1] 37 85 29 13 49...
```

```
> order(datos$"edad")  
[1] 102 175 124 147 118...
```

➤ **sort()**

```
> sort(datos$“edad”)  
[1] 1 1 3 3 4...
```

```
> datos$“edad”[order(datos$“edad”)]  
[1] 1 1 3 3 4...
```

- **order** y **sort** admiten **decreasing** = TRUE.

Ordenando data frames

```
> head(datos)
  ID edad sexo estado.civil nivel.estudios peso altura fumador diabetes cancer.mama cancer.prostata
1 137  37 Mujer Casado      Bajo 59.58221 150.7163   No    No     Si      <NA>
2 174  85 Mujer Soltero    Alto 59.95427 149.2075   No    Si     No      <NA>
3 200  29 Hombre Casado    Bajo 79.20674 168.9795   No    Si     Si      Si
4  23  13 Hombre Divorciado Alto 80.78347 171.1568   Si    Si     <NA>    Si
5  39  49 Hombre Divorciado Bajo 80.76036 170.5682   Si    Si     <NA>    No
6  90  12 Hombre Casado    Alto 79.83426 170.8565   No    No     <NA>    Si
```

```
> datos.ordenados.ID <- datos[order(datos$"ID"), ]
```

```
> head(datos.ordenados.ID)
  ID edad sexo estado.civil nivel.estudios peso altura fumador diabetes cancer.mama cancer.prostata
167  1  21 Hombre Soltero    Alto 80.23501 171.0138   No    No     <NA>    Si
168  2  13 Mujer Soltero    Alto 61.92519 150.3046   Si    No     Si      <NA>
118  3   4 Hombre Soltero    Alto 80.13242 168.7405   No    Si     <NA>    Si
138  4   9 Mujer Soltero    Alto 61.17363 150.2084   Si    No     Si      <NA>
59   5  81 Hombre Soltero    Alto 80.11593 168.1224   No    No     <NA>    Si
162  6  66 Mujer Casado    Alto 58.91967 150.4848   No    Si     No     <NA>
```

```
> datos.ordenados.peso.altura <-
  datos[order(datos$"peso",datos$"altura"), ]
```

```
> head(datos.ordenados.peso.altura)
  ID edad sexo estado.civil nivel.estudios peso altura fumador diabetes cancer.mama cancer.prostata
137 104   6 Mujer Divorciado Medio 56.56025 148.9580   No    Si     Si      <NA>
119  67  75 Mujer Soltero    Bajo 57.30206 149.7838   No    Si     Si      <NA>
90  195  55 Mujer Divorciado Medio 57.58416 151.1169   No    No     No      <NA>
143  93  46 Mujer Soltero    Bajo 57.88490 148.9322   Si    No     Si      <NA>
16  148   6 Mujer Soltero    Bajo 57.93049 149.7306   No    No     Si      <NA>
17  192  31 Mujer Casado    Bajo 57.99527 148.3326   Si    Si     Si      <NA>
```

Ejercicios del 9 al 15

EJERCICIOS
“ejercicios.manejo.basico.datos.r”

3.3.6. Acceso a elementos

Vector []

Array [,]

Data.frame [,]

List [[]]

Vector [*]

Data.frame [* , *]

*vector de indices, vector lógico, condición lógica sobre otros vectores (variables), etc

Sustitución de elementos accediendo antes a ellos

Ejemplo para un vector:

```
> X <- c(1, 2, 3, NA, 4)
```

```
> X [ 4 ] <- 10
```

Otra alternativa (hay muchas más):

```
> Y<- X
```

```
> Y [ is.na(X) ] <- 10
```

```
> datos.new2<-datos
```

```
> datos.new2[1 , 3] <- "Hombre"
```

```
> datos.new2[datos.new2$"ID"==200 , sexo ]<-"Mujer"
```

```
> datos.new2$"sexo"[datos.new2$ID==200]<-"Mujer"
```

Ejercicio del 16

EJERCICIOS
“ejercicios.manejo.basico.datos.r”