Módulo IV

Taller de IA y CD

Maciel-Cruz, EJ

2025-06-12

Contenidos

1 Pre-procesado de Datos

2 Generalidades

El pre-procesado de datos consiste en el conjunto de acciones requeridas para que una base de datos pase de un formato complicado, ilegible o combinado, a una base de datos limpia, atómica, y útil para las pruebas que vamos a realizar.

En primera instancia, para realizar el pre-procesado, debemos de saber cómo se encuentra reestructurada nuestra base de datos. Para eso, debemos de cargar la base de datos que vamos a trabajar. Carguemos la base de datos provista al final del Módulo III y exploremos en el visualizador nativo de R. Recuerda que todos los enlaces están en esta liga.

```
pacman::p_load(tidyverse,GGally,esquisse,ggThemeAssist,summarytools)
```

Cargamos las librerías, la alternativa es usar "library()".

```
base_datos <- read_delim("BD R.csv", delim = ";",
                   escape_double = FALSE, trim_ws = TRUE)
```

```
Rows: 403 Columns: 20
-- Column specification --------------------------------------------------------
Delimiter: ";"
chr  (2): location, gender
dbl (18): id, chol, glu, hdl, ratio chol-hdl, glyhb, age, height, weight, ph...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

> **Tip**
>
> Recuerda que la base de datos debe de tener el nombre acorde a tu archivo que tienes. Puedes renombrar tu archivo, o lo correcto, poner el nombre correcto en R.

De entrada, al cargar nuestra base de datos, podemos observar que se menciona en la consola algunas características de la base, como son el número de filas, el número de columnas, cuáles variables son de tipo carácter y cuáles son de tipo double (es un tipo de codificación especial que usa R para considerar un dato tanto numérico como entero). Para explorar cada parte, podemos hacer los siguientes comandos:

```r
BD1 <- base_datos # Hacemos nuestro respaldo
colnames(BD1) # Visualizamos los nombres de las variables en el orden en que están en nuestra
```

```
 [1] "id"             "chol"           "glu"            "hdl"
 [5] "ratio chol-hdl" "glyhb"          "location"       "age"
 [9] "gender"         "height"         "weight"         "phys.act"
[13] "bp_manage"      "bp.1s"          "bp.1d"          "bp.2s"
[17] "bp.2d"          "waist"          "hip"            "sec.id"
```

Con el código anterior, podemos observar que tenemos un total de 20 variables (incluyendo nuestro número de Id), de forma habitual, R enumera las variables de izquierda a derecha y en cada salto de línea nos arroja el número con el que empieza. Por ejemplo, [9] corresponde a "gender". Si bien, esta función es útil para dar un vistazo a todas nuestras columnas y la posición de cada una de ellas, no nos arroja información sobre el tipo de dato de cada una de ellas, para ello, utilizamos lo siguiente:

```r
sapply(BD1, class)
```

```
        id          chol           glu           hdl ratio chol-hdl
 "numeric"     "numeric"     "numeric"     "numeric"     "numeric"
     glyhb      location           age        gender        height
 "numeric"   "character"     "numeric"   "character"     "numeric"
    weight       phys.act     bp_manage         bp.1s          bp.1d
 "numeric"     "numeric"     "numeric"     "numeric"     "numeric"
     bp.2s          bp.2d         waist           hip        sec.id
 "numeric"     "numeric"     "numeric"     "numeric"     "numeric"
```

Recordemos que la función de "class", nos otorgaba el tipo de dato por vector/variable que solicitemos, la función "sapply", lo que logra es que repite los comandos para todo. Dicho de otra forma, "sapply" hace que se aplique "class" a cada variable de la base de datos llamada "BD1".

Con este código, ahora tenemos la información sobre el tipo de dato de cada una de nuestras variables, aunque no la posición en nuestra base de datos. El conocer el tipo de dato es esencial, ya que algunas variables las codificamos como lógicas, pero aparecen como numéricas, o en el caso de "Id", no tiene sentido que sea numérico (ya que no realizaremos operaciones matemáticas), por lo que más adelante veremos la conversión de variables de una forma más sistemática.

Otra función bastante útil para un vistazo a cómo se encuentra conformada nuestra base de datos, y tener un resumen de nuestra información.

```
summary(BD1)
```

```
      id              chol            glu             hdl        
 Min.   : 1000   Min.   : 78.0   Min.   : 48.0   Min.   : 12.00  
 1st Qu.: 4792   1st Qu.:179.0   1st Qu.: 81.0   1st Qu.: 38.00  
 Median :15766   Median :204.0   Median : 89.0   Median : 46.00  
 Mean   :15978   Mean   :207.8   Mean   :106.7   Mean   : 50.45  
 3rd Qu.:20336   3rd Qu.:230.0   3rd Qu.:106.0   3rd Qu.: 59.00  
 Max.   :41756   Max.   :443.0   Max.   :385.0   Max.   :120.00  
                 NA's   :1                       NA's   :1       
 ratio chol-hdl       glyhb         location              age        
 Min.   : 1.500   Min.   : 2.68   Length:403         Min.   :19.00  
 1st Qu.: 3.200   1st Qu.: 4.38   Class :character   1st Qu.:34.00  
 Median : 4.200   Median : 4.84   Mode  :character   Median :45.00  
 Mean   : 4.522   Mean   : 5.59                      Mean   :46.85  
 3rd Qu.: 5.400   3rd Qu.: 5.60                      3rd Qu.:60.00  
 Max.   :19.300   Max.   :16.11                      Max.   :92.00  
 NA's   :1        NA's   :13                                        
    gender              height          weight          phys.act     
 Length:403         Min.   :52.00   Min.   :  0.0   Min.   :0.000  
 Class :character   1st Qu.:63.00   1st Qu.:146.0   1st Qu.:0.000  
 Mode  :character   Median :66.00   Median :177.0   Median :1.000  
                    Mean   :66.02   Mean   :181.7   Mean   :1.003  
                    3rd Qu.:69.00   3rd Qu.:207.0   3rd Qu.:2.000  
                    Max.   :76.00   Max.   :374.0   Max.   :2.000  
                    NA's   :5                       NA's   :12     
  bp_manage          bp.1s           bp.1d            bp.2s      
 Min.   :0.0000   Min.   : 90.0   Min.   : 48.00   Min.   : 92.0  
 1st Qu.:0.0000   1st Qu.:121.2   1st Qu.: 75.00   1st Qu.:129.0  
 Median :1.0000   Median :136.0   Median : 82.00   Median :140.0  
 Mean   :0.6501   Mean   :136.9   Mean   : 83.32   Mean   :144.4  
 3rd Qu.:1.0000   3rd Qu.:146.8   3rd Qu.: 90.00   3rd Qu.:158.0  
 Max.   :1.0000   Max.   :250.0   Max.   :124.00   Max.   :238.0  
                  NA's   :5       NA's   :5        NA's   :6      
    bp.2d            waist           hip             sec.id      
 Min.   : 49.00   Min.   :26.0   Min.   :30.00   Min.   : 1000  
 1st Qu.: 76.00   1st Qu.:33.0   1st Qu.:39.00   1st Qu.: 4792  
 Median : 85.00   Median :37.0   Median :42.00   Median :15766  
 Mean   : 85.17   Mean   :37.9   Mean   :43.04   Mean   :15978  
 3rd Qu.: 94.00   3rd Qu.:41.0   3rd Qu.:46.00   3rd Qu.:20336  
 Max.   :129.00   Max.   :56.0   Max.   :64.00   Max.   :41756  
 NA's   :6        NA's   :2      NA's   :2                      
```

En el resumen anterior, que se logra con la función "summary", tiene cómo objetivo otorgarnos estadística descriptiva básica tanto de las variables numéricas como de variables categóricas. Hay que notar que variables cómo "Location", nos aparece cómo "character", por el inicio de codificación que realizamos. Si fuera una variable de tipo factor, nos arrojaría frecuencias, posteriormente haremos la conversión y repetiremos el paso.

3 Exploración de Datos

En R, existen muchos métodos para obtener el mismo resultado, se pueden tomar muchas vías. Habitualmente, la vía que requiere menos líneas de código es la mejor, además de que requiere menos costos computacionales.

Los resúmenes pueden ser textuales (cómo los que hemos hecho previamente), o gráficos (p. e. histogramas), veremos a continuación algunos ejemplos.

3.1 Resumen textual de los datos

Podemos llevar a cabo estadística descriptiva empleando algunas funciones en la librería "summarytools". Por ejemplo, podemos utilizar lo siguiente:

```
freq(BD1)
```

```
Variable(s) ignored: id, chol, glu, hdl, ratio chol-hdl, glyhb, age, weight

Frequencies
BD1$location
Type: Character

                      Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
-------------------- ------ --------- -------------- --------- --------------
      Aguascalientes    179     44.42          44.42     44.42          44.42
             Jalisco    175     43.42          87.84     43.42          87.84
             Morelos     49     12.16         100.00     12.16         100.00
                <NA>      0                                0.00         100.00
               Total    403    100.00         100.00    100.00         100.00

BD1$gender
Type: Character

              Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
```

hombre	169	41.94	41.94	41.94	41.94
mujer	234	58.06	100.00	58.06	100.00
<NA>	0			0.00	100.00
Total	403	100.00	100.00	100.00	100.00

BD1$height
Type: Numeric

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
52	1	0.25	0.25	0.25	0.25
55	1	0.25	0.50	0.25	0.50
56	1	0.25	0.75	0.25	0.74
58	3	0.75	1.51	0.74	1.49
59	9	2.26	3.77	2.23	3.72
60	10	2.51	6.28	2.48	6.20
61	19	4.77	11.06	4.71	10.92
62	32	8.04	19.10	7.94	18.86
63	43	10.80	29.90	10.67	29.53
64	33	8.29	38.19	8.19	37.72
65	33	8.29	46.48	8.19	45.91
66	35	8.79	55.28	8.68	54.59
67	36	9.05	64.32	8.93	63.52
68	26	6.53	70.85	6.45	69.98
69	37	9.30	80.15	9.18	79.16
70	24	6.03	86.18	5.96	85.11
71	22	5.53	91.71	5.46	90.57
72	14	3.52	95.23	3.47	94.04
73	8	2.01	97.24	1.99	96.03
74	5	1.26	98.49	1.24	97.27
75	4	1.01	99.50	0.99	98.26
76	2	0.50	100.00	0.50	98.76
<NA>	5			1.24	100.00
Total	403	100.00	100.00	100.00	100.00

BD1$phys.act
Type: Numeric

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
0	103	26.34	26.34	25.56	25.56
1	184	47.06	73.40	45.66	71.22

```
          2    104     26.60          100.00     25.81          97.02
      <NA>     12                                 2.98         100.00
      Total    403    100.00          100.00    100.00         100.00
```

```
BD1$bp_manage
Type: Numeric
```

```
              Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
----------- ------ --------- -------------- --------- --------------
          0    141     34.99          34.99     34.99          34.99
          1    262     65.01         100.00     65.01         100.00
       <NA>      0                                0.00         100.00
      Total    403    100.00         100.00    100.00         100.00
```

Con el código anterior, de una forma muy sencilla, obtenemos una tabla de frecuencias y porcentajes (tanto relativos como acumulados), totales, y totales acumulados. Además de que nos arroja el número de datos omitidos. Todo esto, por variable…

Otra función extremadamente útil es la siguiente:

```
descr(BD1)
```

```
Non-numerical variable(s) ignored: location, gender
```

```
Descriptive Statistics
BD1
N: 403
```

```
                     age     bp.1d     bp.1s     bp.2d     bp.2s   bp_manage      chol       glu
----------------- -------- --------- --------- --------- --------- ----------- --------- ---------
             Mean    46.85     83.32    136.90     85.17    144.40        0.65    207.85    106.67
          Std.Dev    16.31     13.59     22.74     12.64     21.15        0.48     44.45     53.08
              Min    19.00     48.00     90.00     49.00     92.00        0.00     78.00     48.00
               Q1    34.00     75.00    121.00     76.00    129.00        0.00    179.00     81.00
           Median    45.00     82.00    136.00     85.00    140.00        1.00    204.00     89.00
               Q3    60.00     90.00    147.00     94.00    158.00        1.00    230.00    106.00
              Max    92.00    124.00    250.00    129.00    238.00        1.00    443.00    385.00
              MAD    19.27     11.86     20.76     13.34     17.79        0.00     37.06     17.79
              IQR    26.00     15.00     25.50     18.00     29.00        1.00     51.00     25.00
               CV     0.35      0.16      0.17      0.15      0.15        0.73      0.21      0.50
         Skewness     0.32      0.27      1.10      0.20      0.90       -0.63      0.92      2.75
      SE.Skewness     0.12      0.12      0.12      0.12      0.12        0.12      0.12      0.12
```

Kurtosis	-0.67	0.04	2.38	0.32	1.71	-1.61	2.54	8.10
N.Valid	403.00	398.00	398.00	397.00	397.00	403.00	402.00	403.00
N	403.00	403.00	403.00	403.00	403.00	403.00	403.00	403.00
Pct.Valid	100.00	98.76	98.76	98.51	98.51	100.00	99.75	100.00

Table: Table continues below

	glyhb	hdl	height	hip	id	phys.act	ratio chol-hdl
Mean	5.59	50.45	66.02	43.04	15978.31	1.00	4.52
Std.Dev	2.24	17.26	3.92	5.66	11881.12	0.73	1.73
Min	2.68	12.00	52.00	30.00	1000.00	0.00	1.50
Q1	4.38	38.00	63.00	39.00	4792.00	0.00	3.20
Median	4.84	46.00	66.00	42.00	15766.00	1.00	4.20
Q3	5.60	59.00	69.00	46.00	20337.00	2.00	5.40
Max	16.11	120.00	76.00	64.00	41756.00	2.00	19.30
MAD	0.83	14.83	4.45	4.45	8176.54	1.48	1.63
IQR	1.22	21.00	6.00	7.00	15543.50	2.00	2.20
CV	0.40	0.34	0.06	0.13	0.74	0.73	0.38
Skewness	2.23	1.19	0.03	0.80	0.81	0.00	2.20
SE.Skewness	0.12	0.12	0.12	0.12	0.12	0.12	0.12
Kurtosis	4.98	1.93	-0.21	0.83	0.06	-1.12	13.17
N.Valid	390.00	402.00	398.00	401.00	403.00	391.00	402.00
N	403.00	403.00	403.00	403.00	403.00	403.00	403.00
Pct.Valid	96.77	99.75	98.76	99.50	100.00	97.02	99.75

Table: Table continues below

	sec.id	waist	weight
Mean	15978.31	37.90	181.68
Std.Dev	11881.12	5.73	49.28
Min	1000.00	26.00	0.00
Q1	4792.00	33.00	146.00
Median	15766.00	37.00	177.00
Q3	20337.00	41.00	207.00
Max	41756.00	56.00	374.00
MAD	8176.54	5.93	44.48
IQR	15543.50	8.00	61.00

```
              CV       0.74     0.15     0.27
        Skewness       0.81     0.47     0.80
    SE.Skewness       0.12     0.12     0.12
        Kurtosis       0.06    -0.17     1.29
         N.Valid     403.00   401.00   403.00
               N     403.00   403.00   403.00
       Pct.Valid     100.00    99.50   100.00
```

La función "descr" nos arroja estadística descriptiva de los datos numéricos (recordemos que deben de estar clasificados en este tipo en R), de cada una de las variables. Se incluye la media, desviación estándar, rangos, cuartiles, etc.

En la documentación de summarytools es posible ver algunos argumentos modificadores y elegir estadísticos que deseemos.

Ahora, si bien, es muy útil la función, de forma general, no es lo habitual respecto al formato de visualización. Casi siempre trabajamos con tablas que tengan el estadístico en las columnas, y la variable en las filas, para lograr esto, se puede hacer de la siguiente manera:

```
descr(BD1$chol,
      transpose = F)
```

```
Error in match(x, table, nomatch = 0L): 'match' requires vector arguments
```

```
Warning in parse_call(mc = match.call(), var_name = (ncol(xx) == 1), var_label
= (ncol(xx) == : metadata extraction terminated unexpectedly; inspect results
carefully
```

```
Descriptive Statistics
BD1
N: 403

                    BD1$chol
----------------- ----------
             Mean     207.85
          Std.Dev      44.45
              Min      78.00
               Q1     179.00
           Median     204.00
               Q3     230.00
              Max     443.00
              MAD      37.06
```

```
            IQR      51.00
             CV       0.21
       Skewness       0.92
    SE.Skewness       0.12
       Kurtosis       2.54
        N.Valid     402.00
              N     403.00
      Pct.Valid      99.75
```

Lo que hace lo anterior, es transponer los datos (intercambiar lo ejes), mediante el argumento "transpose". Ahora, tenemos una tabla descriptiva como la vemos habitualmente en los artículos.

Otra situación, es que tal vez queramos elegir solo algunos estadísticos, como puede ser media, desviación estándar, mínimo, máximo, y mediana. Eso se logra de la siguiente manera:

```
descr(BD1,
      stats = c("mean", "sd", "min", "max", "med"),
      transpose = T)
```

```
Non-numerical variable(s) ignored: location, gender

Descriptive Statistics
BD1
N: 403
```

	Mean	Std.Dev	Min	Max	Median
age	46.85	16.31	19.00	92.00	45.00
bp.1d	83.32	13.59	48.00	124.00	82.00
bp.1s	136.90	22.74	90.00	250.00	136.00
bp.2d	85.17	12.64	49.00	129.00	85.00
bp.2s	144.40	21.15	92.00	238.00	140.00
bp_manage	0.65	0.48	0.00	1.00	1.00
chol	207.85	44.45	78.00	443.00	204.00
glu	106.67	53.08	48.00	385.00	89.00
glyhb	5.59	2.24	2.68	16.11	4.84
hdl	50.45	17.26	12.00	120.00	46.00
height	66.02	3.92	52.00	76.00	66.00
hip	43.04	5.66	30.00	64.00	42.00
id	15978.31	11881.12	1000.00	41756.00	15766.00
phys.act	1.00	0.73	0.00	2.00	1.00

```
ratio chol-hdl       4.52       1.73      1.50      19.30       4.20
        sec.id   15978.31   11881.12   1000.00   41756.00   15766.00
         waist      37.90       5.73     26.00      56.00      37.00
        weight     181.68      49.28      0.00     374.00     177.00
```

Con el código anterior, tenemos solo los estadísticos descriptivos que necesitamos.

3.2 Resumen gráfico de los datos

Los resúmenes gráficos de datos nos permiten darnos una idea general y rápida de cómo está estructurada nuestra base. Para hacer un resumen gráfico, hacemos lo siguiente:

```
view(dfSummary(BD1))
```

```
Switching method to 'browser'
```

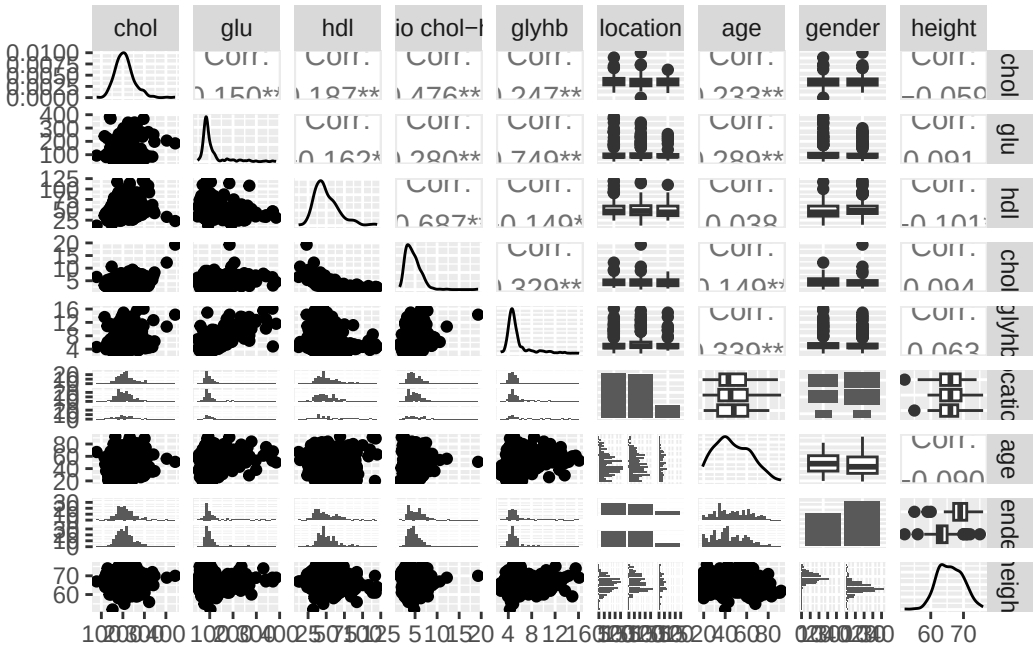
```
Output file written: C:\Users\Hack\AppData\Local\Temp\RtmpcrXKwR\file3e442860b22.html
```

Primeramente, hacemos un llamado a la función "view", que tiene cómo objeto visualizar en formato html lo que estamos pidiendo (no confundir con "View", con mayúscula). Luego solicitamos la función "dfSummary", que nos otorga un pequeño resumen gráfico y estadístico de las variables en nuestra base de datos.

Si bien, la información otorgada es similar a lo que hemos visto con otras funciones de "summarytools", no nos arroja todos los estadísticos, y en esta herramienta nos otorga un pequeño gráfico de cómo están distribuidas las observaciones por variable. Además, si presionamos el ícono de "Show in new window" y nos aparecerá en una ventana de nuestro navegador web. Revisemos cada variable.

Otra forma de hacer un análisis exploratorio extremadamente útil (el más útil que considero), y de forma simple, es el siguiente.

```
ggpairs(BD1[2:10])
```



Con el código anterior, solicitamos la función "ggpairs", que nos arroja una matriz de gráficos de dispersión, histogramas y correlaciones entre las variables numéricas. En este caso, seleccionamos las variables del 2 al 10 (es decir, quitamos el Id), para que no nos arroje un gráfico de dispersión con el folio.

Si deseamos incluir más variables, podemos hacerlo, pero hay que considerar que entre más variables incluyamos, más grande será la matriz y más difícil de visualizar.

> Tip
>
> Dependiendo del tamaño del monitor y definición, no hacer matrices de más de 10 variables, ya que no se puede apreciar muy bien las estadísticas.

4 Transformación de datos

La transformación de datos, en primera instancia consiste en la recodificación de variables, es decir, pasar de una variable numérica a lógica, o de carácter a numérica (si es el caso), entre otros casos. Para llevar a cabo eso, debemos considerar cómo está planeada la base de datos.

En la base de datos que estamos utilizando, tenemos variables de diversos tipos, como son numéricas, lógicas, factores, y de carácter. Algunas de estas variables no están correctamente codificadas, por lo que debemos hacer una transformación de datos para que estén en el tipo correcto.

Además, algunas variables tienen valores numéricos que corresponden a categorías, por ejemplo:

- "phys.act" tiene valores de 0 a 1, que corresponden a "Sedentario", "Bajo", y "Alto". Por lo tanto, es de tipo factor.
- "bp.manage" tiene valores de 0 y 1, que corresponden a "No", "Sí", respectivamente. Por lo que es de tipo lógico.

Con lo anterior podemos ver que tenemos que recodificar algunas variables para que sean del tipo de dato correcto, es decir, solo vamos a modificar el tipo de dato, no vamos a transformar el contenido aún.

4.1 Conversión del tipo de dato

En R, las conversiones de tipo de dato como las hemos manejado (p. e. de numérico binario a lógico), se les llama "coerción", estrictamente. Este método de conversión de datos tiene un flujo predeterminado, que va de los datos más restrictivos a los más flexibles, y no sucede de forma inversa (salvo algunas excepciones). El orden de coerción es el siguiente:

> lógico -> entero -> numérico -> cadena de texto `(logical -\> integer -\> numeric -\> character\)`

Es decir, de un T o F, podemos convertirlo a un dato entero. Un dato entero lo podemos convertir a numérico, pero el orden inverso hace que se pierda información. De igual forma, podemos hacer coerción de un dato lógico, entero o numérico a uno de tipo carácter.

No podemos transformar un carácter (digamos, "Positivo"), a un número o un valor lógico, a menos que sea en inglés, recordemos las palabras reservadas. Primero sería necesario hacer una transformación de datos sustituyendo valores o creando nuevas variables. Algunos casos, como un valor entero, si puede retornar la coerción directa a un valor lógico.

Las coerciones se logran mediante el prefijo "as." y agregamos el tipo de dato que queremos transformar (integer, numeric, factor, etc.), esto aplica para cualquier tipo de dato.

Cabe destacar que si intentamos hacer coerción de datos inversa, es decir, en contra del orden mencionado, se puede hacer, pero se pierde información. Hagamos lo siguiente:

```
V1 <- c("uno", "dos", "tres")
V1
```

```
[1] "uno"  "dos"  "tres"
```

```r
V2 <- as.integer(V1)
```

```
Warning: NAs introduced by coercion
```

```r
V2
```

```
[1] NA NA NA
```

Al imprimir en consola, obtenemos el contenido de V1, “uno” “dos” “tres”, si queremos forzar la conversión e intentamos ver el contenido, solo nos va a arrojar tres “NA”.

Con lo anterior, vamos a hacer algunas coerciones necesarias para trabajar nuestra base de datos. Vamos a escribir todo el código y luego lo explicaremos:

```r
BD2 <- BD1
BD2 <- BD2 %>%
  mutate(id=as.factor(id),
         location=as.factor(location),
         gender=as.factor(gender),
         phys.act=as.factor(phys.act),
         bp_manage=as.logical(bp_manage),
         )
sapply(BD2, class)
```

```
          id         chol          glu          hdl        ratio    chol-hdl 
    "factor"    "numeric"    "numeric"    "numeric"    "numeric" 
       glyhb     location          age       gender       height 
   "numeric"    "factor"    "numeric"    "factor"    "numeric" 
      weight     phys.act    bp_manage        bp.1s         bp.1d 
   "numeric"    "factor"    "logical"    "numeric"    "numeric" 
       bp.2s        bp.2d        waist          hip       sec.id 
   "numeric"    "numeric"    "numeric"    "numeric"    "numeric" 
```

Primeramente, creamos un respaldo de trabajo (primera línea), una BD2 que es una copia de BD1, y con esa trabajaremos.

> **Tip**
>
> Sugiero hacer respaldos de la base de forma inicial, luego con cada cambio que no estén seguros de lo que hace, eso evita tener que correr todo el código desde el inicio.

Luego, utilizamos BD2, operador de asignación, BD2 (es decir, que sobreescriba a la misma base de datos con los cambios que vamos a solicitar).

Luego, añadimos el operador de pipa "%/%", empleado por dplyr para anexar funciones. Luego utilizamos la función de "mutate" de dplyr, esta herramienta nos permite crear o modificar variables, para hacer el cambio de variable con el tipo de dato que deseamos.

Al hacer este cambio, utilizamos el signo de "=" en vez de operador de asignación.

> **Tip**
>
> El operador pipa viene instalado con el paquete "dplyr" o "tidyverse", para introducirlo, hay que poner "Ctrl" + "Shift" + "M", en Mac sería "Cmd" + "Shift" + "M"

Cabe mencionar, que ahorita pondremos a "location" como factor (con los niveles en orden alfabético), para poder listar más fácilmente, sin embargo, realmente no existe motivo para mostrarlo como factor.

Finalmente, revisamos los cambios de nuestra base de datos con la función de "sapply".

4.2 Recodificación de variables

Ahora, considerando que la variable phys.act tiene valores de 0 a 2, y que estos corresponden a "Sedentario", "Bajo", y "Alto", podemos recodificarla para que sea más fácil de interpretar. Para ello, utilizamos la función "recode_factor" del paquete "dplyr".

```
BD2 <- BD2 %>%
  mutate(phys.act = recode_factor(phys.act,
                                  `0` = "Sedentario",
                                  `1` = "Bajo",
                                  `2` = "Alto"))
freq(BD2$phys.act)
```

```
Frequencies
BD2$phys.act
Type: Factor

                  Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
---------------- ------ --------- -------------- --------- --------------
      Sedentario    103     26.34          26.34     25.56          25.56
            Bajo    184     47.06          73.40     45.66          71.22
            Alto    104     26.60         100.00     25.81          97.02
           <NA>     12                                2.98         100.00
           Total    403    100.00         100.00    100.00         100.00
```

Si hacemos un resumen nuevamente, ya podremos ver que phys.act es una variable con tres niveles.

> **Tip**
>
> Cabe mencionar que podemos hacer llamado de una sola variable de interés, esto se logra con el símbolo de peso o dólar "$".

Con el código anterior, podemos sustituir cualquier valor por otro que deseemos, en vez de 1, 2 y 3, podemos poner "Bajo", "Medio" y "Alto", por ejemplo. Depende del uso que se requiera, se puede recodificar como se desee.

4.3 Transformación de variables

Por último, para esta sección, se puede transformar la variable de peso y talla en el índice de masa corporal o IMC, ya sea sustituyendo la variable, o creando una nueva con "mutate"; sin embargo, hay que considerar que el peso y la altura se encuentran en libras y pulgadas, respectivamente, por lo que tenemos que hacer la transformación de dichas variables en primer lugar.

```
BD2 <- BD2 %>%
  mutate(weight = weight * 0.453592,
         height = height * 0.0254,
         "indice masa corporal" = weight/height^2) %>%
  relocate(`indice masa corporal`,
           .after=weight)

descr(BD2$`indice masa corporal`)
```

```
Descriptive Statistics
BD2$`indice masa corporal`
N: 403

                     indice masa corporal
---------------- ------------------------
            Mean                    29.39
         Std.Dev                     7.84
             Min                     0.00
              Q1                    23.81
          Median                    28.10
              Q3                    33.89
             Max                    64.20
```

```
              MAD                    7.39
              IQR                    9.99
               CV                    0.27
         Skewness                    0.85
      SE.Skewness                    0.12
         Kurtosis                    1.61
          N.Valid                  398.00
                N                  403.00
        Pct.Valid                   98.76
```

El código anterior, utiliza la función mutate nuevamente, y lo que hacemos es una operación aritmética para transformar el peso y la altura al sistema métrico, luego calculamos el IMC con la fórmula tradicional y se crea la variable, la cual se coloca al final de la base de datos..

> **!** Important
>
> Nótese de que usamos espacios en algunas variables nuevas, si bien, mencionamos que es una mala práctica, más adelante veremos herramientas que hacen limpieza de estos datos, ese es el motivo de que lo estemos creando de esa forma.

Con la función de "relocate" se posiciona después de la variable "weight" usando ".after".

Revisamos nuestra base para confirmar el resultado con función "descr".

4.4 Creación y eliminación de variables

Ya vimos un ejemplo de cómo crear una nueva variable con mutate. La utilidad de "dplyr" y R, consiste en crear todas las variables o modificarlas como lo necesitemos. En términos de eficiencia de recursos, es bastante útil, ya que a partir de unas cuántas variables numéricas podemos obtener muchas otras sin necesidad de realizar formularios demasiado extensos.

Ahora, también hay que considerar que las variables "waist" y "hip" se encuentran en el sistema imperial, por lo que debemos transformarlas también.

```
BD2 <- BD2 %>%
  mutate(waist = waist * 2.54,
         hip = hip * 2.54)

descr(BD2[, c("waist", "hip")])
```

```
Descriptive Statistics
BD2
```

```
N: 403

                      hip    waist
----------------- -------- --------
             Mean   109.32    96.27
          Std.Dev    14.37    14.55
              Min    76.20    66.04
               Q1    99.06    83.82
           Median   106.68    93.98
               Q3   116.84   104.14
              Max   162.56   142.24
              MAD    11.30    15.06
              IQR    17.78    20.32
               CV     0.13     0.15
         Skewness     0.80     0.47
      SE.Skewness     0.12     0.12
         Kurtosis     0.83    -0.17
          N.Valid   401.00   401.00
                N   403.00   403.00
        Pct.Valid    99.50    99.50
```

El código anterior, es muy similar a lo ya mencionado. Solamente mutamos o transformamos la variable de interés a través de una operación aritmética, y luego revisamos el resultado con la función "descr".

Nuevamente, el proceso es exactamente igual al anterior, lo único que cambia son las variables empleadas y las operaciones aritméticas.

Ahora, vamos a crear una variable lógica a partir de una numérica. De "glyhb", vamos a dicotomizar a "1" y "0", para dividir a los sujetos positivo y negativo para diabetes. Consideremos el punto de corte de 6.5 para la hemoglobina glucosilada.

```
BD2 <- BD2 %>%
  mutate(Diabetes = as.logical(ifelse(glyhb >= 6.5, 1, 0))) %>%
  relocate(Diabetes,
           .after = glyhb)

freq(BD2$Diabetes)
```

```
Frequencies
BD2$Diabetes
Type: Logical
```

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
FALSE	325	83.33	83.33	80.65	80.65
TRUE	65	16.67	100.00	16.13	96.77
<NA>	13			3.23	100.00
Total	403	100.00	100.00	100.00	100.00

En el código anterior, primeramente usamos la función "as.logical", para convertir todo lo que hagamos en un vector de este tipo. Luego utilizamos la función "ifelse", que nos permite crear una variable lógica a partir de una condición. Si la condición se cumple (es decir, si glyhb es mayor o igual a 6.5), asigna un valor de "1", de lo contrario asigna un valor de "0". Finalmente, usamos "relocate" para posicionar la nueva variable después de "glyhb".

Cotejamos nuestro resultado con la función de "freq".

4.5 Recodificación

La recodificación de variables consiste en cambiar los nombres de las variables (observaciones), por otros. En ocasiones puede que prefiramos nombres más significativos, o más cortos, o acortarlos para que nos sea de mayor utilidad. Vamos hacerlo con la variable "Lugar de origen".

Primeramente, veamos todas las observaciones que existen en la columna de interés, esto se logra así.

```
unique(BD2$location)
```

```
[1] Jalisco        Morelos        Aguascalientes
Levels: Aguascalientes Jalisco Morelos
```

El código anterior nuestra los diferentes nombres que existen en la columna, además de que si está codificado como factor, nos arroja en orden los niveles. Ya que tenemos esos datos, vamos a modificarlos.

```
BD2$location <- recode_factor(BD2$location,
                                      Aguascalientes="Ags",
                                      Jalisco="Jal",
                                      Morelos="Mor")
unique(BD2$location)
```

```
[1] Jal Mor Ags
Levels: Ags Jal Mor
```

En el ejemplo anterior, veíamos que tenemos nombres de estados, y queremos acortarlos. Utilizamos la función "recode_factor", seleccionamos la base de datos y la variable que queramos recodificar, y escribimos el nombre la variable, un signo "=", y luego entre paréntesis el nuevo nombre que deseamos.

Esto se puede repetir para cualquier variable, y se conserva el tipo de dato (es decir, factor). Otro punto, que al utilizar la función debemos mencionar todos los factores en orden de cómo queremos que aparezcan, aquí se sobreescribe el orden alfanumérico, y podemos elegir el orden de nuestros niveles.

Para eliminar una variable basta con realizar el siguiente código:

```
BD2 <- BD2%>%
       select(-sec.id)
colnames(BD2)
```

```
 [1] "id"                    "chol"                  "glu"
 [4] "hdl"                   "ratio chol-hdl"        "glyhb"
 [7] "Diabetes"              "location"              "age"
[10] "gender"                "height"                "weight"
[13] "indice masa corporal" "phys.act"              "bp_manage"
[16] "bp.1s"                 "bp.1d"                 "bp.2s"
[19] "bp.2d"                 "waist"                 "hip"
```

> **Tip**
>
> Recordemos que podemos concatenar una serie de variables para borrarlas de una vez.

4.6 Renombrar variables

En esta sección veremos cómo cambiar los nombres de las variables con las que estamos trabajando, es decir, los nombres de las columnas. Por ejemplo, digamos que no nos gustan los nombres en inglés, la forma en la lo podemos lograr es mediante la función "rename", veamos un ejemplo.

Primeramente, veamos los nombres nuestras columnas, recordemos la función de mostrar las columnas.

```
colnames(BD2)
```

```
 [1] "id"                    "chol"                  "glu"
 [4] "hdl"                   "ratio chol-hdl"        "glyhb"
 [7] "Diabetes"              "location"              "age"
[10] "gender"                "height"                "weight"
[13] "indice masa corporal" "phys.act"              "bp_manage"
[16] "bp.1s"                 "bp.1d"                 "bp.2s"
[19] "bp.2d"                 "waist"                 "hip"
```

Vamos a renombrar solamente las columnas de "chol" y "glu".

```
BD2 <- rename(BD2,
              colesterol=chol,
              glucosa=glu)
```

Con lo anterior, podemos ver que los nombres se han sobreescrito, pero no se modificó en absoluto su contenido. En esta función, la sintaxis es poner primero el nombre nuevo de la variable y luego el de la variable a sobreescribir.

Este método es de utilidad cuando las variables contienen nombres poco descriptivos, por ejemplo, la variable "Diabetes", puede significar muchas cosas, pero podemos agregar algún contexto, como es que sea "log" de lógico, o para la variable que colesterol, que sea "val" de valor.

Otra opción es dar una documentación adicional para los significados o contextos de las variables, esto es altamente utilizado cuando se trabajan con bases de datos grandes.

4.7 Limpieza de nombres

Otra función bastante útil, se encuentra incluida en el paquete llamado "janitor", dicho paquete hay que instalarlo y solo utilizaremos una función en todo el curso, motivo por el que no se mencionó previamente, además de que se utiliza una sola vez. Instalemos y carguemos el paquete.

```
pacman::p_load(janitor)
```

Ahora, la función que se mencionó, es de limpiar los nombres de las columnas, para que sean más fácilmente manipulables (sustituir espacios por guión bajo, usar solo minúsculas, quitar acentos, etc.). Si bien, es totalmente opcional, hace más ágil la manipulación de bases de datos.

```
BD2 <- BD2 %>%
  clean_names()
```

Y listo, así de simple "limpiamos" los nombres de nuestras columnas para que no tengamos que estar utilizando espacios ni comillas, ni nada de eso. Recordemos que el lenguaje de R es el inglés, por lo que también omiten los acentos y la letra "ñ".

Si no desean llevar este paso, no hay ningún problema, es totalmente opcional; sin embargo, por cuestiones del curso y para que el código sea reproducible, mantendremos la base de datos limpia usando "janitor" y continuaremos utilizando dichos nombres de las variables.

Vamos a terminar de cambiar los nombres de las variables, hagamos el siguiente código:

```
BD2 <- rename(BD2,
              col_hdl_ratio="ratio_chol_hdl",
              hb1ac=glyhb,
              diabetes_log=diabetes,
              lugar_origen=location,
              edad=age,
              sexo=gender,
              altura=height,
              peso=weight,
              imc=indice_masa_corporal,
              act_fisica_fac=phys_act,
              tx_ta=bp_manage,
              tas_1=bp_1s,
              tas_2=bp_2s,
              tad_1=bp_1d,
              tad_2=bp_2d,
              cintura=waist,
              cadera=hip
              )

colnames(BD2)
```

```
 [1] "id"             "colesterol"     "glucosa"        "hdl"
 [5] "col_hdl_ratio"  "hb1ac"          "diabetes_log"   "lugar_origen"
 [9] "edad"           "sexo"           "altura"         "peso"
[13] "imc"            "act_fisica_fac" "tx_ta"          "tas_1"
[17] "tad_1"          "tas_2"          "tad_2"          "cintura"
[21] "cadera"
```

4.8 Datos faltantes

La regla es encontrarnos con datos faltantes. Existen diferentes métodos para manejarlos, que puede ser no contarlos y hacer estadística con lo que se tenga, se puede calcular una media y emplearla para los datos vacíos, se pueden remover las filas que tengan datos vacíos (con el riesgo de perder mucha información), se puede hacer mención de que es información faltante como tal, o se puede hacer una recaptura manual a ver cuántos podemos obtener.

Sea cual sea la aproximación, el manejo de datos faltantes puede llegar a ser bastante lento y costoso, por lo que se mencionarán solo los métodos básicos de manejo.

En primera instancia, debemos de saber cuántos datos faltantes tenemos, podemos echar mano de la función "summary".

```
summary(BD2)
```

```
       id        colesterol       glucosa           hdl        
 1000   :  1   Min.   : 78.0   Min.   : 48.0   Min.   : 12.00  
 1001   :  1   1st Qu.:179.0   1st Qu.: 81.0   1st Qu.: 38.00  
 1002   :  1   Median :204.0   Median : 89.0   Median : 46.00  
 1003   :  1   Mean   :207.8   Mean   :106.7   Mean   : 50.45  
 1005   :  1   3rd Qu.:230.0   3rd Qu.:106.0   3rd Qu.: 59.00  
 1008   :  1   Max.   :443.0   Max.   :385.0   Max.   :120.00  
 (Other):397   NA's   :1                       NA's   :1       
 col_hdl_ratio        hb1ac       diabetes_log    lugar_origen      edad      
 Min.   : 1.500   Min.   : 2.68   Mode :logical   Ags:179      Min.   :19.00  
 1st Qu.: 3.200   1st Qu.: 4.38   FALSE:325       Jal:175      1st Qu.:34.00  
 Median : 4.200   Median : 4.84   TRUE :65        Mor: 49      Median :45.00  
 Mean   : 4.522   Mean   : 5.59   NA's :13                     Mean   :46.85  
 3rd Qu.: 5.400   3rd Qu.: 5.60                                3rd Qu.:60.00  
 Max.   :19.300   Max.   :16.11                                Max.   :92.00  
 NA's   :1        NA's   :13                                                  
     sexo         altura           peso             imc        
 hombre:169   Min.   :1.321   Min.   :  0.00   Min.   : 0.00  
 mujer :234   1st Qu.:1.600   1st Qu.: 66.22   1st Qu.:23.89  
              Median :1.676   Median : 80.29   Median :28.10  
              Mean   :1.677   Mean   : 82.41   Mean   :29.39  
              3rd Qu.:1.753   3rd Qu.: 93.89   3rd Qu.:33.88  
              Max.   :1.930   Max.   :169.64   Max.   :64.20  
              NA's   :5                        NA's   :5      
   act_fisica_fac   tx_ta            tas_1           tad_1       
 Sedentario:103    Mode :logical   Min.   : 90.0   Min.   : 48.00  
 Bajo      :184    FALSE:141       1st Qu.:121.2   1st Qu.: 75.00  
```

```
 Alto      :104    TRUE :262       Median :136.0   Median : 82.00
 NA's      : 12                    Mean   :136.9   Mean   : 83.32
                                   3rd Qu.:146.8   3rd Qu.: 90.00
                                   Max.   :250.0   Max.   :124.00
                                   NA's   :5       NA's   :5
    tas_2           tad_2           cintura          cadera
 Min.   : 92.0   Min.   : 49.00   Min.   : 66.04   Min.   : 76.20
 1st Qu.:129.0   1st Qu.: 76.00   1st Qu.: 83.82   1st Qu.: 99.06
 Median :140.0   Median : 85.00   Median : 93.98   Median :106.68
 Mean   :144.4   Mean   : 85.17   Mean   : 96.27   Mean   :109.32
 3rd Qu.:158.0   3rd Qu.: 94.00   3rd Qu.:104.14   3rd Qu.:116.84
 Max.   :238.0   Max.   :129.00   Max.   :142.24   Max.   :162.56
 NA's   :6       NA's   :6        NA's   :2        NA's   :2
```

De manera general, para cuestiones de estadística descriptiva y para la mayoría de los análisis de R, los valores omitidos no afectan, ya que se pueden calcular los descriptivos sin problema, además de que los análisis vienen con la función predeterminada de que manejen dichos datos de alguna forma

Dependiendo del análisis es lo que se hará, aunque de forma general, se lleva a cabo el análisis con la información con la que se cuenta.

Si por alguna razón nosotros no queremos trabajar con dichos valores, podemos utilizar un argumento que omita los valores omitidos (el nombre varía de función de función), que lo que hace es que, literalmente omite los datos ausentes de algún análisis. Para que sea práctico, vamos a hacer el siguiente ejemplo.

```
descr(BD2,
      stats = c("mean", "sd", "min", "max", "med", "n.valid"),
      transpose = T,
      headings  = F,
      order = "sort")
```

```
Non-numerical variable(s) ignored: id, diabetes_log, lugar_origen, sexo, act_fisica_fac, tx_
```

	Mean	Std.Dev	Min	Max	Median	N.Valid
altura	1.68	0.10	1.32	1.93	1.68	398.00
cadera	109.32	14.37	76.20	162.56	106.68	401.00
cintura	96.27	14.55	66.04	142.24	93.98	401.00
col_hdl_ratio	4.52	1.73	1.50	19.30	4.20	402.00
colesterol	207.85	44.45	78.00	443.00	204.00	402.00

edad	46.85	16.31	19.00	92.00	45.00	403.00
glucosa	106.67	53.08	48.00	385.00	89.00	403.00
hb1ac	5.59	2.24	2.68	16.11	4.84	390.00
hdl	50.45	17.26	12.00	120.00	46.00	402.00
imc	29.39	7.84	0.00	64.20	28.10	398.00
peso	82.41	22.35	0.00	169.64	80.29	403.00
tad_1	83.32	13.59	48.00	124.00	82.00	398.00
tad_2	85.17	12.64	49.00	129.00	85.00	397.00
tas_1	136.90	22.74	90.00	250.00	136.00	398.00
tas_2	144.40	21.15	92.00	238.00	140.00	397.00

En este código solicitamos algunas estadísticas descriptivas de nuestra BD2, además, agregamos el argumento "sort", el cual nos ordena alfabéticamente nuestras variables. Ya que el dato omitido se encuentra como un NA y no contiene un valor específico, no afecta en lo absoluto nuestras estadísticas.

Por otra parte, veamos el siguiente ejemplo.

```
freq(BD2[-1])
```

```
Variable(s) ignored: colesterol, glucosa, hdl, col_hdl_ratio, hb1ac, edad, peso, imc

Frequencies
BD2$diabetes_log
Type: Logical

              Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
----------- ------ --------- -------------- --------- --------------
     FALSE    325     83.33          83.33     80.65          80.65
      TRUE     65     16.67         100.00     16.13          96.77
      <NA>     13                                3.23         100.00
     Total    403    100.00         100.00    100.00         100.00

BD2$lugar_origen
Type: Factor

              Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
----------- ------ --------- -------------- --------- --------------
       Ags    179     44.42          44.42     44.42          44.42
       Jal    175     43.42          87.84     43.42          87.84
       Mor     49     12.16         100.00     12.16         100.00
      <NA>      0                                0.00         100.00
```

Total	403	100.00	100.00	100.00	100.00

BD2$sexo
Type: Factor

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
hombre	169	41.94	41.94	41.94	41.94
mujer	234	58.06	100.00	58.06	100.00
<NA>	0			0.00	100.00
Total	403	100.00	100.00	100.00	100.00

BD2$altura
Type: Numeric

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
1.3208	1	0.25	0.25	0.25	0.25
1.397	1	0.25	0.50	0.25	0.50
1.4224	1	0.25	0.75	0.25	0.74
1.4732	3	0.75	1.51	0.74	1.49
1.4986	9	2.26	3.77	2.23	3.72
1.524	10	2.51	6.28	2.48	6.20
1.5494	19	4.77	11.06	4.71	10.92
1.5748	32	8.04	19.10	7.94	18.86
1.6002	43	10.80	29.90	10.67	29.53
1.6256	33	8.29	38.19	8.19	37.72
1.651	33	8.29	46.48	8.19	45.91
1.6764	35	8.79	55.28	8.68	54.59
1.7018	36	9.05	64.32	8.93	63.52
1.7272	26	6.53	70.85	6.45	69.98
1.7526	37	9.30	80.15	9.18	79.16
1.778	24	6.03	86.18	5.96	85.11
1.8034	22	5.53	91.71	5.46	90.57
1.8288	14	3.52	95.23	3.47	94.04
1.8542	8	2.01	97.24	1.99	96.03
1.8796	5	1.26	98.49	1.24	97.27
1.905	4	1.01	99.50	0.99	98.26
1.9304	2	0.50	100.00	0.50	98.76
<NA>	5			1.24	100.00
Total	403	100.00	100.00	100.00	100.00

BD2$act_fisica_fac

```
Type: Factor
```

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
Sedentario	103	26.34	26.34	25.56	25.56
Bajo	184	47.06	73.40	45.66	71.22
Alto	104	26.60	100.00	25.81	97.02
<NA>	12			2.98	100.00
Total	403	100.00	100.00	100.00	100.00

```
BD2$tx_ta
Type: Logical
```

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
FALSE	141	34.99	34.99	34.99	34.99
TRUE	262	65.01	100.00	65.01	100.00
<NA>	0			0.00	100.00
Total	403	100.00	100.00	100.00	100.00

Ahora solicitamos frecuencias y porcentajes de las variables no numéricas. Nótese además que agregamos un argumento, al escribir adentro de corchetes y justo después de nuestra base de datos el texto "-1", estamos solicitando que no procese una columna, es decir, el folio. Esto es bastante útil si queremos quitar algunas variables, además, también podemos solicitar que solo trabaje algunas, es decir, si escribimos "2:5,7,8", sin el signo de "-", pedimos que haga la estadística descriptiva para las variables que están la posición 2 a 5, la 7 y la 8 (siempre y cuando cumplan el requisito de que sean factores o lógicos).

Como podemos ver, nuestra tabla de frecuencias y porcentajes arroja una fila llamada "<NA>", esto nos dice la cantidad de valores faltantes en dicha columna. Si por alguna razón, no queremos que aparezcan en nuestros informes, podemos hacer lo siguiente.

```
freq(BD2[-1],
     report.nas = F)
```

```
Variable(s) ignored: colesterol, glucosa, hdl, col_hdl_ratio, hb1ac, edad, peso, imc

Frequencies
BD2$diabetes_log
Type: Logical

              Freq        %   % Cum.
```

FALSE	325	83.33	83.33
TRUE	65	16.67	100.00
Total	390	100.00	100.00

BD2$lugar_origen
Type: Factor

	Freq	%	% Cum.
Ags	179	44.42	44.42
Jal	175	43.42	87.84
Mor	49	12.16	100.00
Total	403	100.00	100.00

BD2$sexo
Type: Factor

	Freq	%	% Cum.
hombre	169	41.94	41.94
mujer	234	58.06	100.00
Total	403	100.00	100.00

BD2$altura
Type: Numeric

	Freq	%	% Cum.
1.3208	1	0.25	0.25
1.397	1	0.25	0.50
1.4224	1	0.25	0.75
1.4732	3	0.75	1.51
1.4986	9	2.26	3.77
1.524	10	2.51	6.28
1.5494	19	4.77	11.06
1.5748	32	8.04	19.10
1.6002	43	10.80	29.90
1.6256	33	8.29	38.19
1.651	33	8.29	46.48
1.6764	35	8.79	55.28
1.7018	36	9.05	64.32
1.7272	26	6.53	70.85

```
      1.7526     37     9.30    80.15
       1.778     24     6.03    86.18
      1.8034     22     5.53    91.71
      1.8288     14     3.52    95.23
      1.8542      8     2.01    97.24
      1.8796      5     1.26    98.49
       1.905      4     1.01    99.50
      1.9304      2     0.50   100.00
       Total    398   100.00   100.00
```

```
BD2$act_fisica_fac
Type: Factor
```

```
                    Freq        %   % Cum.
---------------- ------ -------- --------
      Sedentario    103    26.34    26.34
            Bajo    184    47.06    73.40
            Alto    104    26.60   100.00
           Total    391   100.00   100.00
```

```
BD2$tx_ta
Type: Logical
```

```
              Freq        %   % Cum.
---------- ------ -------- --------
      FALSE    141    34.99    34.99
       TRUE    262    65.01   100.00
      Total    403   100.00   100.00
```

Y listo, con eso quitamos los valores omitidos y solo obtendremos los valores presentes. Hay que notar que ya no tenemos algunas columnas, como "% Valid", ya que describe el porcentaje de valores válidos (presentes), de dicha columna. De igual forma, podemos ordenar de forma alfabética si así lo deseamos.

Si deseamos quitar todos los valores NA de nuestra base de datos, podemos hacer lo siguiente:

```
BD3 <- BD2

BD3 <- BD3 %>%
  drop_na()

descr(BD3,
      stats = c("mean", "sd", "min", "max", "med", "n.valid"),
```

```
      transpose = T,
      headings  = F,
      order = "sort")
```

```
Non-numerical variable(s) ignored: id, diabetes_log, lugar_origen, sexo, act_fisica_fac, tx_
```

	Mean	Std.Dev	Min	Max	Median	N.Valid
altura	1.68	0.10	1.32	1.93	1.68	367.00
cadera	109.27	14.33	76.20	162.56	106.68	367.00
cintura	96.27	14.77	66.04	142.24	93.98	367.00
col_hdl_ratio	4.53	1.76	1.50	19.30	4.20	367.00
colesterol	207.25	44.25	78.00	443.00	203.00	367.00
edad	46.63	16.23	19.00	92.00	45.00	367.00
glucosa	107.74	54.48	48.00	385.00	90.00	367.00
hb1ac	5.62	2.27	2.68	16.11	4.86	367.00
hdl	50.26	17.13	12.00	120.00	46.00	367.00
imc	29.29	7.86	0.00	64.20	28.07	367.00
peso	82.31	22.48	0.00	169.64	79.83	367.00
tad_1	83.31	13.64	48.00	124.00	82.00	367.00
tad_2	85.21	12.75	49.00	129.00	85.00	367.00
tas_1	136.77	22.30	90.00	230.00	136.00	367.00
tas_2	144.44	21.50	92.00	238.00	140.00	367.00

Con el código anterior, "tiramos" los NA's de la base de datos, con lo que nos quedan 367 observaciones. Ya que realmente no deseo quitarlo, cree una copia de la base. Finalmente cotejo mi resultado con "descr".

> **Tip**
>
> Para algunos análisis es forzoso tener datos completos, por lo que se recomienda hacer arquitectura de datos antes de empezar la investigación.

4.9 Ordenar

Para ordenar y reordenar nuestras bases de datos, podemos hacer uso de diversas herramientas. Podemos reordenar nuestras variables acorde al nombre (alfabéticamente), podemos ordenar el listado de observaciones acorde a una variable (número de folio, por ejemplo), podemos ordenar nuestros resultados de alguna prueba o estadística (de mayor o menor, o de menor a mayor), etc. Además, ya vimos una forma de reinsertar alguna variable en otra posición, según lo necesitemos. Hagamos los siguientes ejercicios.

```
BD2 <- BD2 %>%
    relocate(diabetes_log,
           .after = id)
colnames(BD2)
```

```
 [1] "id"             "diabetes_log"   "colesterol"     "glucosa"
 [5] "hdl"            "col_hdl_ratio"  "hb1ac"          "lugar_origen"
 [9] "edad"           "sexo"           "altura"         "peso"
[13] "imc"            "act_fisica_fac" "tx_ta"          "tas_1"
[17] "tad_1"          "tas_2"          "tad_2"          "cintura"
[21] "cadera"
```

En el código anterior, vamos a reposicionar la variable "diabetes_log" (recordemos que ya limpiamos los nombres), y vamos a colocarla después de "id". Podemos revertir el proceso sustituyendo "folio", por "diabetes", que es en donde estaba.

Si por alguna razón queremos que sea la primera variable a mencionar, podemos utilizar ".before", en vez de ".after".

```
BD2 <- BD2 %>%
    relocate(diabetes_log,
           .before = id)
colnames(BD2)
```

```
 [1] "diabetes_log"   "id"             "colesterol"     "glucosa"
 [5] "hdl"            "col_hdl_ratio"  "hb1ac"          "lugar_origen"
 [9] "edad"           "sexo"           "altura"         "peso"
[13] "imc"            "act_fisica_fac" "tx_ta"          "tas_1"
[17] "tad_1"          "tas_2"          "tad_2"          "cintura"
[21] "cadera"
```

Y listo, ahora "diabetes_log" es la primera variable en nuestra base de datos. Vamos a revertir el proceso a como estaba.

```
BD2 <- BD2 %>%
    relocate(diabetes_log,
           .after = hb1ac)
colnames(BD2)
```

```
 [1] "id"             "colesterol"     "glucosa"        "hdl"
 [5] "col_hdl_ratio"  "hb1ac"          "diabetes_log"   "lugar_origen"
 [9] "edad"           "sexo"           "altura"         "peso"
[13] "imc"            "act_fisica_fac" "tx_ta"          "tas_1"
[17] "tad_1"          "tas_2"          "tad_2"          "cintura"
[21] "cadera"
```

Ahora, vamos a solicitar un listado de las columnas ordenadas por orden alfabético.

```
BD2 %>%
  colnames() %>%
  sort()
```

```
 [1] "act_fisica_fac" "altura"         "cadera"         "cintura"
 [5] "col_hdl_ratio"  "colesterol"     "diabetes_log"   "edad"
 [9] "glucosa"        "hb1ac"          "hdl"            "id"
[13] "imc"            "lugar_origen"   "peso"           "sexo"
[17] "tad_1"          "tad_2"          "tas_1"          "tas_2"
[21] "tx_ta"
```

Con el código anterior, podemos ordenar de forma alfabética nuestras variables. Si quisiéramos un orden inverso (es decir, de la Z a la A), podemos hacer lo siguiente.

```
BD2 %>%
  colnames() %>%
  sort(decreasing = T)
```

```
 [1] "tx_ta"          "tas_2"          "tas_1"          "tad_2"
 [5] "tad_1"          "sexo"           "peso"           "lugar_origen"
 [9] "imc"            "id"             "hdl"            "hb1ac"
[13] "glucosa"        "edad"           "diabetes_log"   "colesterol"
[17] "col_hdl_ratio"  "cintura"        "cadera"         "altura"
[21] "act_fisica_fac"
```

Solo con agregar el argumento “decreasing” como verdadero, nos invertirá el orden.

Si deseáramos realizar lo mismo con las observaciones por variable, podemos usar lo siguiente.

```r
BD2$lugar_origen %>%
  sort() %>%
  unique()
```

```
[1] Ags Jal Mor
Levels: Ags Jal Mor
```

Sin embargo, hay que considerar lo siguiente: primero, seleccionamos la base de datos de interés, así como variable a trabajar; posteriormente, utilizamos el operador de pipa, para enlazar el comando "sort" para ordenar los valores.

Si nos detuviéramos ahí, nos arrojaría cada una de las observaciones, para evitar eso, enlazamos otra función que es "unique", lo que hace es mostrar solamente los valores únicos. Lo anterior también es de utilidad para detectar algún error de tipado (por ejemplo, que hubiera algún "Agus" o "Afs", en vez de Ags).

Otra opción bastante viable, y más sencilla, de cierta forma, es utilizar las herramientas de summarytools. Podemos llevar a cabo un resumen de frecuencias de toda la base, o de un subset. Recordemos que el método es el siguiente.

```r
freq(BD2[-1])
```

```
Variable(s) ignored: colesterol, glucosa, hdl, col_hdl_ratio, hb1ac, edad, peso, imc


Frequencies
BD2$diabetes_log
Type: Logical

              Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
----------- ------ --------- -------------- --------- --------------
     FALSE    325     83.33          83.33     80.65          80.65
      TRUE     65     16.67         100.00     16.13          96.77
     <NA>      13                                3.23         100.00
     Total    403    100.00         100.00    100.00         100.00

BD2$lugar_origen
Type: Factor

              Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
----------- ------ --------- -------------- --------- --------------
       Ags    179     44.42          44.42     44.42          44.42
```

Jal	175	43.42	87.84	43.42	87.84
Mor	49	12.16	100.00	12.16	100.00
<NA>	0			0.00	100.00
Total	403	100.00	100.00	100.00	100.00

BD2$sexo
Type: Factor

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
hombre	169	41.94	41.94	41.94	41.94
mujer	234	58.06	100.00	58.06	100.00
<NA>	0			0.00	100.00
Total	403	100.00	100.00	100.00	100.00

BD2$altura
Type: Numeric

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
1.3208	1	0.25	0.25	0.25	0.25
1.397	1	0.25	0.50	0.25	0.50
1.4224	1	0.25	0.75	0.25	0.74
1.4732	3	0.75	1.51	0.74	1.49
1.4986	9	2.26	3.77	2.23	3.72
1.524	10	2.51	6.28	2.48	6.20
1.5494	19	4.77	11.06	4.71	10.92
1.5748	32	8.04	19.10	7.94	18.86
1.6002	43	10.80	29.90	10.67	29.53
1.6256	33	8.29	38.19	8.19	37.72
1.651	33	8.29	46.48	8.19	45.91
1.6764	35	8.79	55.28	8.68	54.59
1.7018	36	9.05	64.32	8.93	63.52
1.7272	26	6.53	70.85	6.45	69.98
1.7526	37	9.30	80.15	9.18	79.16
1.778	24	6.03	86.18	5.96	85.11
1.8034	22	5.53	91.71	5.46	90.57
1.8288	14	3.52	95.23	3.47	94.04
1.8542	8	2.01	97.24	1.99	96.03
1.8796	5	1.26	98.49	1.24	97.27
1.905	4	1.01	99.50	0.99	98.26
1.9304	2	0.50	100.00	0.50	98.76
<NA>	5			1.24	100.00

```
      Total    403    100.00          100.00    100.00          100.00
```

```
BD2$act_fisica_fac
Type: Factor
```

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
Sedentario	103	26.34	26.34	25.56	25.56
Bajo	184	47.06	73.40	45.66	71.22
Alto	104	26.60	100.00	25.81	97.02
<NA>	12			2.98	100.00
Total	403	100.00	100.00	100.00	100.00

```
BD2$tx_ta
Type: Logical
```

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
FALSE	141	34.99	34.99	34.99	34.99
TRUE	262	65.01	100.00	65.01	100.00
<NA>	0			0.00	100.00
Total	403	100.00	100.00	100.00	100.00

Recordemos eliminar la primera variable, que corresponde al folio.

4.10 Filtros y subsets

Si bien, "filter" y "subset", son funciones diferentes, son técnicamente iguales, la primera es del paquete de dplyr, mientras que el segundo es de R base. El primero gana en términos de eficiencia y facilidad del manejo, sobre todo para grandes volúmenes de datos, por lo que nos enfocaremos en esa función.

Para objetivo de la sección, llamaremos a filtros a todo condicional que reduzca de nuestra base de datos completa, un grupo de observaciones que cumplan dichas condiciones; mientras que llamaremos subset o subconjunto, a la creación de una nueva base que cumpla con los filtros que pidamos.

Es posible crear filtros de información, ya sea de tipo numérico (p. e. si queremos mayor que 100 para un biomarcador), o queremos que nos muestre todos las observaciones positivas para un operador lógico, o solo aquellos sujetos que cumplan un determinado caracter. Además, podemos filtrar ya sea columnas o variables, o filas. Incluso podemos agrupar casos.

Vamos a hacer el siguiente ejemplo.

```r
BD2 %>%
  group_by(diabetes_log) %>%
  summarize(Media_Colesterol = mean(colesterol, na.rm = TRUE))
```

```
# A tibble: 3 x 2
  diabetes_log Media_Colesterol
  <lgl>                   <dbl>
1 FALSE                    203.
2 TRUE                     229.
3 NA                       225.
```

Con lo anterior, primero llamamos nuestra base de datos, agrupamos por "diabetes_log", el cual tiene dos grupos ("T" y "F"), y pedimos la media de los valores de la colesterol.

Recordemos que no estamos sobreescribiendo y creando la nueva variable, solo la estamos generando sin guardarla. Finalmente, la función de "na.rm" nos permite omitir los valores omitidos, ya que esta función es sensible a ese dato y nos puede arrojar errores.

Naturalmente, podemos agrupar por dos o más variables, solo separando las comas. Podemos agrupar mediante puntos de corte, ya sea creando la variable previamente, o en este punto.

```r
BD2 %>%
  group_by(diabetes_log, lugar_origen="Ags", edad>40) %>%
  summarize("Niveles" =  mean(glucosa, na.rm = T))
```

```
`summarise()` has grouped output by 'diabetes_log', 'lugar_origen'. You can
override using the `.groups` argument.
```

```
# A tibble: 6 x 4
# Groups:   diabetes_log, lugar_origen [3]
  diabetes_log lugar_origen `edad > 40` Niveles
  <lgl>        <chr>        <lgl>         <dbl>
1 FALSE        Ags          FALSE          87.3
2 FALSE        Ags          TRUE           94.0
3 TRUE         Ags          FALSE         172.
4 TRUE         Ags          TRUE          191.
5 NA           Ags          FALSE          82
6 NA           Ags          TRUE           87.5
```

En el ejemplo anterior, creamos un primer grupo con la variable de "diabetes_log", luego hicimos un segundo grupo con la variable de "lugar_origen" para "Ags" y rematamos con un "edad" mayor de 40 años. Cabe destacar que para las variables podemos especificar que argumentos aritméticos o exactamente igual a algo, como el caso de Ags.

Cabe la posibilidad de que deseemos hacer todas las estadísticas que nos interesan de una vez para un determinado subconjunto de datos. Hagamos uso de "summarytools" para ello.

Tip: el paquete de "summarytools" es más eficiente para muchas funciones, te recomiendo usarlo siempre.

```
with(BD2,
     stby(data    = glucosa,
          INDICES = list(lugar_origen, sexo),
          FUN     = descr,
          stats   = c("mean", "sd", "min", "med", "max"))
)
```

```
Descriptive Statistics
BD2$glucosa
Group: lugar_origen = Ags, sexo = hombre
N: 75

                glucosa
------------- ---------
         Mean    113.61
      Std.Dev     72.37
          Min     48.00
       Median     88.00
          Max    385.00

Group: lugar_origen = Jal, sexo = hombre
N: 71

                glucosa
------------- ---------
         Mean    110.20
      Std.Dev     56.42
          Min     57.00
       Median     91.00
          Max    342.00

Group: lugar_origen = Mor, sexo = hombre
```

```
N: 23

               glucosa
------------- ---------
         Mean    114.09
      Std.Dev     55.98
          Min     77.00
       Median     91.00
          Max    255.00

Group: lugar_origen = Ags, sexo = mujer
N: 104

               glucosa
------------- ---------
         Mean    102.56
      Std.Dev     44.03
          Min     62.00
       Median     90.00
          Max    279.00

Group: lugar_origen = Jal, sexo = mujer
N: 104

               glucosa
------------- ---------
         Mean    104.30
      Std.Dev     46.83
          Min     52.00
       Median     88.00
          Max    299.00

Group: lugar_origen = Mor, sexo = mujer
N: 26

               glucosa
------------- ---------
         Mean     96.42
      Std.Dev     25.88
          Min     56.00
       Median     90.50
          Max    182.00
```

En este código, empezamos primeramente con la función “with”, de “summarytools”.

Luego elegimos nuestra base de datos, luego la función “data” solicita la variable a trabajar, la función “INDICES” nos pide los índices o grupos a realizar (aquí agregamos el argumento “list” para poder hacer dos o más grupos, es opcional), “FUN” es la función directa de “summarytools” (recordemos que “descr” es para numérico descriptivo, mientras que “freq” es para frecuencias), finalmente con “stats” concatenamos o combinamos las funciones estadísticas que queremos.

Vamos a invertir las agrupaciones, para ver qué nos convence más.

```
with(BD2,
     stby(data    = diabetes_log,
          INDICES = lugar_origen,
          FUN     = freq,
          order   = "freq")
)
```

```
Frequencies
BD2$diabetes_log
Type: Logical
Group: lugar_origen = Ags

              Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
---------- ------ --------- -------------- --------- --------------
     FALSE    149     84.18          84.18     83.24          83.24
      TRUE     28     15.82         100.00     15.64          98.88
      <NA>      2                                1.12         100.00
     Total    179    100.00         100.00    100.00         100.00

Group: lugar_origen = Jal

              Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
---------- ------ --------- -------------- --------- --------------
     FALSE    136     82.42          82.42     77.71          77.71
      TRUE     29     17.58         100.00     16.57          94.29
      <NA>     10                                5.71         100.00
     Total    175    100.00         100.00    100.00         100.00

Group: lugar_origen = Mor

              Freq   % Valid   % Valid Cum.   % Total   % Total Cum.
---------- ------ --------- -------------- --------- --------------
```

```
      FALSE     40     83.33          83.33     81.63          81.63
       TRUE      8     16.67         100.00     16.33          97.96
       <NA>      1                                2.04         100.00
      Total     49    100.00         100.00    100.00         100.00
```

Con esta nueva table puedo ver que "Jal" tiene mayor porcentaje y frecuencia de diabetes, pero "Mich" tiene porcentajes bastante similares. Dependiendo de las necesidades, podemos ajustar los grupos, orden, quitar NA's, etc.

Ahora, supongamos que de nuestra base de datos, tenemos que trabajar solamente con algunas variables para un determinado proyecto. Podemos crear subconjuntos solo con las variables que deseemos. Veamos un ejemplo.

```
metabolico <- BD2 %>%
  select(id,colesterol,glucosa,hdl,col_hdl_ratio,hb1ac,diabetes_log,lugar_origen,edad,sexo,i
summary(metabolico)
```

```
       id        colesterol       glucosa           hdl
 1000   :  1   Min.   : 78.0   Min.   : 48.0   Min.   : 12.00
 1001   :  1   1st Qu.:179.0   1st Qu.: 81.0   1st Qu.: 38.00
 1002   :  1   Median :204.0   Median : 89.0   Median : 46.00
 1003   :  1   Mean   :207.8   Mean   :106.7   Mean   : 50.45
 1005   :  1   3rd Qu.:230.0   3rd Qu.:106.0   3rd Qu.: 59.00
 1008   :  1   Max.   :443.0   Max.   :385.0   Max.   :120.00
 (Other):397   NA's   :1                       NA's   :1
 col_hdl_ratio        hb1ac       diabetes_log    lugar_origen      edad
 Min.   : 1.500   Min.   : 2.68   Mode :logical   Ags:179      Min.   :19.00
 1st Qu.: 3.200   1st Qu.: 4.38   FALSE:325       Jal:175      1st Qu.:34.00
 Median : 4.200   Median : 4.84   TRUE :65        Mor: 49      Median :45.00
 Mean   : 4.522   Mean   : 5.59   NA's :13                     Mean   :46.85
 3rd Qu.: 5.400   3rd Qu.: 5.60                                3rd Qu.:60.00
 Max.   :19.300   Max.   :16.11                                Max.   :92.00
 NA's   :1        NA's   :13
     sexo          imc           act_fisica_fac
 hombre:169   Min.   : 0.00   Sedentario:103
 mujer :234   1st Qu.:23.89   Bajo      :184
              Median :28.10   Alto      :104
              Mean   :29.39   NA's      : 12
              3rd Qu.:33.88
              Max.   :64.20
              NA's   :5
```

En el código anterior, creamos una base de datos llamada “metabolico”, que contiene las variables que nos interesan.

Ahora, también es posible haciendo subconjuntos si las variables cumplen determinadas condiciones.

```
glucosa_M100 <- BD2 %>%
  select(id,lugar_origen,imc,glucosa) %>%
  filter(glucosa>100)
glucosa_M100
```

```
# A tibble: 127 x 4
   id    lugar_origen   imc glucosa
   <fct> <fct>        <dbl>   <dbl>
 1 1250  Jal           37.4     206
 2 1254  Jal           48.9     299
 3 1301  Jal           33.3     101
 4 1304  Jal           28.2     330
 5 1314  Jal           32.0     107
 6 1316  Mor           36.3     206
 7 1502  Jal           29.8     193
 8 2004  Mor           35.2     223
 9 2762  Jal           49.4     111
10 2763  Jal           47.9     106
# i 117 more rows
```

Ahora tenemos un subconjunto de datos que solo incluye el id, lugar de origen, imc y valores de glucosa que sean mayores de 100. Si deseamos, podemos agregar otro filtro para que el imc sea menor de 25, por ejemplo.

```
glucosa_M100 <- BD2 %>%
  select(id,lugar_origen,imc,glucosa) %>%
  filter(glucosa>100, imc>25)
glucosa_M100
```

```
# A tibble: 90 x 4
   id    lugar_origen   imc glucosa
   <fct> <fct>        <dbl>   <dbl>
 1 1250  Jal           37.4     206
 2 1254  Jal           48.9     299
 3 1301  Jal           33.3     101
```

```
 4 1304  Jal           28.2     330
 5 1314  Jal           32.0     107
 6 1316  Mor           36.3     206
 7 1502  Jal           29.8     193
 8 2004  Mor           35.2     223
 9 2762  Jal           49.4     111
10 2763  Jal           47.9     106
# i 80 more rows
```

Ahora tenemos todos los individuos que tienen un imc mayor de 25 y glucosa mayor que 100.

5 Exportación de objetos y bases

Vamos a exportar nuestra base de datos para trabajarla en la siguiente sección:

```
pacman::p_load("writexl")
write_xlsx(BD2, "BD2_R.xlsx")
```

Invocamos la función de "write_xlsx", seleccionamos la base de datos que queremos exportar, que es BD2, y le ponemos su nombre a como deseemos, además, recordemos que debemos poner la extensión del archivo.