

# Clase\_\_2\_\_ejercicios

*Joshua Kock*

*2/1/2019*

## Contents

Exploracion de objetos con base R.	2
Funciones para manejo de datos dplyr	3
Filtrar filas con <code>Filter()</code> . . . . .	3
Organizar filas con <code>Arrange()</code> . . . . .	6
Seleccionar columnas con <code>Select()</code> . . . . .	6

```
library(tidyverse)
```

```
## -- Attaching packages ----- tidyverse 1.2.1 --
## v ggplot2 3.1.0      v purrr  0.2.5
## v tibble  2.0.1      v dplyr  0.7.8
## v tidyr   0.8.2      v stringr 1.3.1
## v readr   1.3.1      v forcats 0.3.0

## Warning: package 'tibble' was built under R version 3.5.2

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

“Data obtained from <http://biostat.mc.vanderbilt.edu/DataSets>”.

Revisar las variables en el siguiente enlace: Codebook: <http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/Cdiabetes.html>

```
diabetes <- read_csv("https://raw.githubusercontent.com/vizual-wanderer/6071402_Electiva_II/master/Base")
```

```
## Parsed with column specification:
## cols(
##   id = col_double(),
##   chol = col_double(),
##   stab.glu = col_double(),
##   hdl = col_double(),
##   ratio = col_double(),
##   glyhb = col_double(),
##   location = col_character(),
##   age = col_double(),
##   gender = col_character(),
##   height = col_double(),
##   weight = col_double(),
##   frame = col_character(),
##   bp.1s = col_double(),
##   bp.1d = col_double(),
##   bp.2s = col_double(),
##   bp.2d = col_double(),
##   waist = col_double(),
##   hip = col_double(),
```

```
## time.ppn = col_double()
## )
```

## Exploracion de objetos con base R.

Explorar el objeto con comando de base R.

```
str(diabetes)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 403 obs. of  19 variables:
## $ id      : num  1000 1001 1002 1003 1005 ...
## $ chol    : num  203 165 228 78 249 248 195 227 177 263 ...
## $ stab.glu: num  82 97 92 93 90 94 92 75 87 89 ...
## $ hdl     : num  56 24 37 12 28 69 41 44 49 40 ...
## $ ratio   : num  3.6 6.9 6.2 6.5 8.9 ...
## $ glyhb   : num  4.31 4.44 4.64 4.63 7.72 ...
## $ location: chr  "Buckingham" "Buckingham" "Buckingham" "Buckingham" ...
## $ age     : num  46 29 58 67 64 34 30 37 45 55 ...
## $ gender  : chr  "female" "female" "female" "male" ...
## $ height  : num  62 64 61 67 68 71 69 59 69 63 ...
## $ weight  : num  121 218 256 119 183 190 191 170 166 202 ...
## $ frame   : chr  "medium" "large" "large" "large" ...
## $ bp.1s   : num  118 112 190 110 138 132 161 NA 160 108 ...
## $ bp.1d   : num  59 68 92 50 80 86 112 NA 80 72 ...
## $ bp.2s   : num  NA NA 185 NA NA NA 161 NA 128 NA ...
## $ bp.2d   : num  NA NA 92 NA NA NA 112 NA 86 NA ...
## $ waist   : num  29 46 49 33 44 36 46 34 34 45 ...
## $ hip     : num  38 48 57 38 41 42 49 39 40 50 ...
## $ time.ppn: num  720 360 180 480 300 195 720 1020 300 240 ...
## - attr(*, "spec")=
## .. cols(
## ..   id = col_double(),
## ..   chol = col_double(),
## ..   stab.glu = col_double(),
## ..   hdl = col_double(),
## ..   ratio = col_double(),
## ..   glyhb = col_double(),
## ..   location = col_character(),
## ..   age = col_double(),
## ..   gender = col_character(),
## ..   height = col_double(),
## ..   weight = col_double(),
## ..   frame = col_character(),
## ..   bp.1s = col_double(),
## ..   bp.1d = col_double(),
## ..   bp.2s = col_double(),
## ..   bp.2d = col_double(),
## ..   waist = col_double(),
## ..   hip = col_double(),
## ..   time.ppn = col_double()
## .. )
```

```
names(diabetes)
```

```
## [1] "id"      "chol"    "stab.glu" "hdl"     "ratio"    "glyhb"
## [7] "location" "age"     "gender"   "height"  "weight"   "frame"
## [13] "bp.1s"    "bp.1d"   "bp.2s"    "bp.2d"   "waist"    "hip"
## [19] "time.ppn"
```

```
length(diabetes)
```

```
## [1] 19
```

## Funciones para manejo de datos dplyr

Vamos a revisar 3 funciones clave del paquete **dplyr** que permiten resolver la gran mayoría de los desafíos de manipulación de datos.

- Escoger observaciones por sus valores (**filter()**).
- Reordenar las filas (**arrange()**).
- Escoger las variables por sus nombres (**select()**).

Todos los verbos funcionan de manera similar: 1. El primer argumento es un dataframe. 2. Los argumentos subsiguientes describen qué hacer con el dataframe, usando los nombres de las variables (sin comillas). 3. El resultado es un nuevo dataframe/tibble.

Juntas, estas propiedades facilitan una cadena de varios pasos simples para lograr un resultado complejo. Vamos a sumergirnos y ver cómo funcionan estos verbos.

### Filtrar filas con **Filter()**

**filter()** permite escoger observaciones basadas en sus valores. El primer argumento es el nombre del dataframe. El segundo y subsiguientes argumentos son las expresiones que filtran el marco de datos. Por ejemplo, podemos seleccionar todos los vuelos los pacientes con hdl igual a 52 con:

Puedes leer mas: `?dplyr::filter()`

```
filter(diabetes, hdl == 52)
```

```
## # A tibble: 8 x 19
##       id chol stab.glu  hdl ratio glyhb location  age gender height
##   <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <chr>   <dbl> <chr>   <dbl>
## 1  1282   254     84   52  4.90  4.52 Bucking~   43 female    62
## 2  4758   185     84   52  3.60  5.28 Louisa    53 female    61
## 3  4842   179     70   52  3.40  3.98 Louisa    34 male      72
## 4 15813   237     96   52  4.60 NA      Bucking~   45 male      69
## 5 20332   178     64   52  3.40  4.10 Louisa    41 female    65
## 6 40784   226    279   52  4.30 10.1 Louisa    84 female    60
## 7 41039   179     85   52  3.40  4.05 Louisa    32 female    62
## 8 41752   199     76   52  3.80  4.49 Louisa    41 female    63
## # ... with 9 more variables: weight <dbl>, frame <chr>, bp.1s <dbl>,
## #   bp.1d <dbl>, bp.2s <dbl>, bp.2d <dbl>, waist <dbl>, hip <dbl>,
## #   time.ppn <dbl>
```

Cuando ejecuta esa línea de código, **dplyr** ejecuta la operación de filtrado y devuelve un dataframe. Las funciones **dplyr** nunca modifican sus entradas, por lo que si desea guardar el resultado, deberá utilizar el operador de asignación `<-`

```
pc_hdl_52 <- filter(diabetes, hdl == 52)
```

R imprime los resultados o los guarda en una variable. Si desea hacer ambas cosas, puede ajustar la asignación entre paréntesis:

```
(pc_hdl_52 <- filter(diabetes, hdl == 52))
```

```
## # A tibble: 8 x 19
##       id chol stab.glu  hdl ratio glyhb location  age gender height
##   <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl> <chr>    <dbl> <chr>    <dbl>
## 1  1282   254      84    52  4.90  4.52 Bucking~   43 female    62
## 2  4758   185      84    52  3.60  5.28 Louisa    53 female    61
## 3  4842   179      70    52  3.40  3.98 Louisa    34 male      72
## 4 15813   237      96    52  4.60  NA    Bucking~   45 male      69
## 5 20332   178      64    52  3.40  4.10 Louisa    41 female    65
## 6 40784   226     279    52  4.30 10.1  Louisa    84 female    60
## 7 41039   179      85    52  3.40  4.05 Louisa    32 female    62
## 8 41752   199      76    52  3.80  4.49 Louisa    41 female    63
## # ... with 9 more variables: weight <dbl>, frame <chr>, bp.1s <dbl>,
## #   bp.1d <dbl>, bp.2s <dbl>, bp.2d <dbl>, waist <dbl>, hip <dbl>,
## #   time.ppn <dbl>
```

## Comparaciones

Para utilizar el filtrado de manera efectiva, debe saber cómo seleccionar las observaciones que desee mediante los operadores de comparación. R proporciona anotaciones estándar: >, >=, <, <=, != (No es igual), y == (igual).

Cuando comienzas con R, el error más fácil de cometer es usar = en lugar de == al probar la igualdad. Cuando esto suceda, obtendrás un error informativo:

## Operadores lógicos

Los argumentos múltiples para `filter` se combinan con “and”: cada expresión debe ser verdadera para que una fila se incluya en la salida. Para otros tipos de combinaciones, deberá utilizar los operadores booleanos usted mismo: & es “and”, | es “OR”, y ! es “NOT”. La figura 5.1 muestra el conjunto completo de operaciones booleanas.

```
filter(diabetes, chol >= 200 | hdl <= 30)
```

```
## # A tibble: 234 x 19
##       id chol stab.glu  hdl ratio glyhb location  age gender height
##   <dbl> <dbl>    <dbl> <dbl> <dbl> <dbl> <chr>    <dbl> <chr>    <dbl>
## 1  1000   203      82    56  3.60  4.31 Bucking~   46 female    62
## 2  1001   165      97    24  6.90  4.44 Bucking~   29 female    64
## 3  1002   228      92    37  6.20  4.64 Bucking~   58 female    61
## 4  1003    78      93    12  6.5   4.63 Bucking~   67 male      67
## 5  1005   249      90    28  8.90  7.72 Bucking~   64 male      68
## 6  1008   248      94    69  3.60  4.81 Bucking~   34 male      71
## 7  1015   227      75    44  5.20  3.94 Bucking~   37 male      59
## 8  1022   263      89    40  6.60  5.78 Bucking~   55 female    63
## 9  1024   242      82    54  4.5   4.77 Louisa    60 female    65
## 10 1029   215     128    34  6.30  4.97 Louisa    38 female    58
## # ... with 224 more rows, and 9 more variables: weight <dbl>, frame <chr>,
## #   bp.1s <dbl>, bp.1d <dbl>, bp.2s <dbl>, bp.2d <dbl>, waist <dbl>,
## #   hip <dbl>, time.ppn <dbl>
```

Con la ejecucion del codigo anterior vemos que tenemos 234 pacientes con colesterol igual o mayor a 200 OR HDL menos o igual a 30. Si queremos que ambos criterios se utilicen es necesario utilizar & para buscar los pacientes que tengan colesterol de 200 o mas y hdl menor o igual a 30.

```
filter(diabetes, chol >= 200 & hdl <= 30)
```

```
## # A tibble: 9 x 19
##       id chol stab.glu  hdl ratio glyhb location  age gender height
##   <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <chr>   <dbl> <chr>   <dbl>
## 1  1005   249     90    28  8.90  7.72 Bucking~   64 male     68
## 2  1280   232     87    30  7.70  5.10 Bucking~   37 male     68
## 3  2778   443    185    23 19.3  14.3 Bucking~   51 female   70
## 4 15797   211    225    29  7.30 10.1 Bucking~   61 female   63
## 5 17814   224     85    30  7.5   5.26 Bucking~   36 male     69
## 6 20750   245    119    26  9.40  7.51 Louisa    36 male     66
## 7 21320   216    155    30  7.20  5.91 Louisa    38 male     68
## 8 21329   249     81    28  8.90  5.12 Louisa    51 female   65
## 9 41001   227     85    26  8.70  4.98 Louisa    58 male     70
## # ... with 9 more variables: weight <dbl>, frame <chr>, bp.1s <dbl>,
## #   bp.1d <dbl>, bp.2s <dbl>, bp.2d <dbl>, waist <dbl>, hip <dbl>,
## #   time.ppn <dbl>
```

Ejercicio:

1.Busca los pacientes masculinos con colesterol igual a 250 y 300.

```
filter(diabetes, gender == "male" & chol == 250 & chol == 300)
```

```
## # A tibble: 0 x 19
## # ... with 19 variables: id <dbl>, chol <dbl>, stab.glu <dbl>, hdl <dbl>,
## #   ratio <dbl>, glyhb <dbl>, location <chr>, age <dbl>, gender <chr>,
## #   height <dbl>, weight <dbl>, frame <chr>, bp.1s <dbl>, bp.1d <dbl>,
## #   bp.2s <dbl>, bp.2d <dbl>, waist <dbl>, hip <dbl>, time.ppn <dbl>
```

2.Busca las mujeres mayor a 45 con HDL menor a 30.

```
filter(diabetes, gender == "female", age > 45, hdl <= 30)
```

```
## # A tibble: 5 x 19
##       id chol stab.glu  hdl ratio glyhb location  age gender height
##   <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <chr>   <dbl> <chr>   <dbl>
## 1  1502   148    193    14 10.6   6.14 Bucking~   54 female   67
## 2  2778   443    185    23 19.3  14.3 Bucking~   51 female   70
## 3 15501   193    248    24  8     7.14 Bucking~   59 female   66
## 4 15797   211    225    29  7.30 10.1 Bucking~   61 female   63
## 5 21329   249     81    28  8.90  5.12 Louisa    51 female   65
## # ... with 9 more variables: weight <dbl>, frame <chr>, bp.1s <dbl>,
## #   bp.1d <dbl>, bp.2s <dbl>, bp.2d <dbl>, waist <dbl>, hip <dbl>,
## #   time.ppn <dbl>
```

3.Cual es el promedio de edad de los pacientes con con la HbA1c mayor a 6.5.

```
df_1 <- filter(diabetes, glyhb > 6.5)
mean(df_1$age)
```

```
## [1] 58.43077
```

## Organizar filas con Arrange()

`arrange()` funciona de forma similar a `filter()`, excepto que en lugar de seleccionar filas, cambia su orden. Se necesita un dataframe y los nombres de columna (o expresiones más complicadas) para ordenar. Si proporciona más de un nombre de columna, cada columna adicional se utilizará para romper empates en los valores de las columnas anteriores:

```
arrange(diabetes, chol, hdl, glyhb)
```

```
## # A tibble: 403 x 19
##       id chol stab.glu hdl ratio glyhb location age gender height
##   <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <chr>   <dbl> <chr>   <dbl>
## 1  1003    78     93    12  6.5  4.63 Bucking~  67 male     67
## 2 15012   115    239    36  3.20 13.6 Louisa   58 male     69
## 3 15017   118     95    39  3    4.71 Louisa   47 female   64
## 4 21357   122     82    43  2.80  3.98 Louisa   36 female   71
## 5  2004   128    223    24  5.30 10.9 Bucking~  60 male     67
## 6 12506   129    110    42  3.10  6.13 Bucking~  56 male     74
## 7 10001   132     99    34  3.90  4.01 Bucking~  21 female   65
## 8 20340   132     83    40  3.30  5.70 Louisa   28 female   68
## 9 21284   134    101    36  3.70  4.67 Bucking~  25 female   63
## 10 4501   134    105    42  3.20  4.29 Bucking~  48 male     70
## # ... with 393 more rows, and 9 more variables: weight <dbl>, frame <chr>,
## #   bp.1s <dbl>, bp.1d <dbl>, bp.2s <dbl>, bp.2d <dbl>, waist <dbl>,
## #   hip <dbl>, time.ppn <dbl>
```

utiliza `desc()` para cambiar el orden de la columna de mayor a menor.

```
arrange(diabetes, desc(chol))
```

```
## # A tibble: 403 x 19
##       id chol stab.glu hdl ratio glyhb location age gender height
##   <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <chr>   <dbl> <chr>   <dbl>
## 1  2778   443    185    23 19.3 14.3 Bucking~  51 female   70
## 2 20313   404    206    33 12.2 10.8 Louisa   56 male     69
## 3 12769   347    197    42  8.30  6.34 Bucking~  36 male     70
## 4 15800   342    251    48  7.10 12.7 Bucking~  63 female   65
## 5 41003   337     85    62  5.40  4.66 Louisa   35 male     72
## 6 41023   322     87    92  3.5  4.45 Louisa   43 female   56
## 7 12006   318    270   108  2.90  6.51 Louisa   60 female   65
## 8 40764   307     87    58  5.30  4.28 Louisa   49 male     67
## 9 21347   306     92    56  5.5  5.58 Louisa   74 male     69
## 10 17795   305     91    44  6.90  5.34 Bucking~  31 male     71
## # ... with 393 more rows, and 9 more variables: weight <dbl>, frame <chr>,
## #   bp.1s <dbl>, bp.1d <dbl>, bp.2s <dbl>, bp.2d <dbl>, waist <dbl>,
## #   hip <dbl>, time.ppn <dbl>
```

## Seleccionar columnas con Select()

No es raro obtener dataframe con cientos o incluso miles de variables. En este caso, el primer desafío a menudo se reduce a las variables en las que está realmente interesado. `Select()` le permite escoger rápidamente un subconjunto útil mediante operaciones basadas en los nombres de las variables.

`select()` no es muy útil con los datos de diabetes porque solo tenemos 19 variables, pero aún se puede entender la idea general: Ver `?select()` para más detalles.

```
select(diabetes, chol, hdl, glyhb)
```

```
## # A tibble: 403 x 3
##   chol    hdl glyhb
##   <dbl> <dbl> <dbl>
## 1   203    56  4.31
## 2   165    24  4.44
## 3   228    37  4.64
## 4    78    12  4.63
## 5   249    28  7.72
## 6   248    69  4.81
## 7   195    41  4.84
## 8   227    44  3.94
## 9   177    49  4.84
## 10  263    40  5.78
## # ... with 393 more rows
```

```
select( diabetes, chol:age)
```

```
## # A tibble: 403 x 7
##   chol stab.glu    hdl ratio glyhb location    age
##   <dbl>    <dbl> <dbl> <dbl> <dbl> <chr>    <dbl>
## 1   203      82    56  3.60  4.31 Buckingham 46
## 2   165      97    24  6.90  4.44 Buckingham 29
## 3   228      92    37  6.20  4.64 Buckingham 58
## 4    78      93    12  6.5   4.63 Buckingham 67
## 5   249      90    28  8.90  7.72 Buckingham 64
## 6   248      94    69  3.60  4.81 Buckingham 34
## 7   195      92    41  4.80  4.84 Buckingham 30
## 8   227      75    44  5.20  3.94 Buckingham 37
## 9   177      87    49  3.60  4.84 Buckingham 45
## 10  263      89    40  6.60  5.78 Buckingham 55
## # ... with 393 more rows
```

```
select(diabetes, -(chol:age))
```

```
## # A tibble: 403 x 12
##   id gender height weight frame bp.1s bp.1d bp.2s bp.2d waist  hip
##   <dbl> <chr>   <dbl>  <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1000 female    62   121 medi~ 118   59   NA   NA    29    38
## 2 1001 female    64   218 large 112   68   NA   NA    46    48
## 3 1002 female    61   256 large 190   92  185   92    49    57
## 4 1003 male      67   119 large 110   50   NA   NA    33    38
## 5 1005 male      68   183 medi~ 138   80   NA   NA    44    41
## 6 1008 male      71   190 large 132   86   NA   NA    36    42
## 7 1011 male      69   191 medi~ 161  112  161  112    46    49
## 8 1015 male      59   170 medi~  NA   NA   NA   NA    34    39
## 9 1016 male      69   166 large 160   80  128   86    34    40
## 10 1022 female    63   202 small 108   72   NA   NA    45    50
## # ... with 393 more rows, and 1 more variable: time.ppn <dbl>
```

Hay una serie de funciones de ayuda que puede utilizar dentro de `select()`: `+ starts_with("abc")`: coincide con los nombres que comienzan con "abc". `+ ends_with("xyz")`: coincide con los nombres que terminan con "xyz". `+ contains("ijk")`: coincide con los nombres que contienen "ijk". `+ coincidirias("(.) \\ 1")`: selecciona variables que coinciden con una expresión regular. Este coincide con cualquier variable que contenga caracteres repetidos. Aprenderás más sobre expresiones regulares en cadenas. `+ num_range("x",`

1:3): coincide con x1, x2 y x3.

Ejercicio: 1. Que pasa cuando incluyes el nombre de una variable en multiples ocasiones en `select()`?

```
select(diabetes, age, age)
```

```
## # A tibble: 403 x 1
##   age
##   <dbl>
## 1    46
## 2    29
## 3    58
## 4    67
## 5    64
## 6    34
## 7    30
## 8    37
## 9    45
## 10   55
## # ... with 393 more rows
```

2. Que hace la funcion `one_of()`? porque seria util en la conjugacion del siguiente vector?

```
?tidyselect::one_of()
vars <- c("age", "gender", "height", "chol", "hdl", "ratio")
vars
```

```
## [1] "age"      "gender" "height" "chol"    "hdl"     "ratio"
```

```
typeof(vars)
```

```
## [1] "character"
```

```
#hay como vars es un vector de caracteres no te va funcionar dentro de select
select(diabetes, one_of(vars))
```

```
## # A tibble: 403 x 6
##   age gender height chol hdl ratio
##   <dbl> <chr>   <dbl> <dbl> <dbl> <dbl>
## 1    46 female    62   203    56  3.60
## 2    29 female    64   165    24  6.90
## 3    58 female    61   228    37  6.20
## 4    67 male      67    78    12  6.5
## 5    64 male      68   249    28  8.90
## 6    34 male      71   248    69  3.60
## 7    30 male      69   195    41  4.80
## 8    37 male      59   227    44  5.20
## 9    45 male      69   177    49  3.60
## 10   55 female    63   263    40  6.60
## # ... with 393 more rows
```

3. Selecciona los ID de los pacientes que cumplen una tension normal (tanto sistolica, como diastolica), y tengan una HbA1C por encima de 7.

```
select(filter(diabetes, bp.1s <= 120, bp.1d <= 80, glyhb > 7), id)
```

```
## # A tibble: 6 x 1
##   id
##   <dbl>
```



## 1 2004  
## 2 4796  
## 3 15502  
## 4 20335  
## 5 40775  
## 6 41510