

Introdução à programação com R

Filtros



Tópicos desta aula

- Operadores
- Mais sobre `Data.frames`

Operadores

Operações lógicas

Uma operação lógica é um teste que retorna **verdadeiro** ou **falso**. No R (e em outras linguagens de programação), esses valores dois valores recebem uma classe especial: `logical`.

O verdadeiro no R vai ser representado pelo valor `TRUE` e o falso pelo valor `FALSE`. Esses nomes no R são **reservados**, isto é, você não pode chamar nenhum objeto de `TRUE` ou `FALSE`.

```
TRUE <- 1  
## Error in TRUE <- 1 : invalid (do_set) left-hand side to assignment
```

Valores lógicos

Checando a classe desses valores, vemos que são lógicos (também conhecidos como valores binários ou booleanos). Eles são os únicos possíveis valores dessa classe.

```
class(TRUE)
```

```
## [1] "logical"
```

```
class(FALSE)
```

```
## [1] "logical"
```

Agora que conhecemos o `TRUE` e `FALSE`, podemos explorar os teste lógicos.

Igualdades

Começando pela igualdade: vamos testar se um valor é igual ao outro. Para isso, usamos o operador `==`.

```
# Testes com resultado verdadeiro  
1 == 1
```

```
## [1] TRUE
```

```
"a" == "a"
```

```
## [1] TRUE
```

```
# Testes com resultado falso  
1 == 2
```

```
## [1] FALSE
```

```
"a" == "b"
```

```
## [1] FALSE
```

Diferenças

Também podemos testar se dois valores são diferentes. Para isso, usamos o operador `!=`.

```
# Testes com resultado falso  
1 != 1
```

```
## [1] FALSE
```

```
"a" != "a"
```

```
## [1] FALSE
```

```
# Testes com resultado verdadeiro  
1 != 2
```

```
## [1] TRUE
```

```
"a" != "b"
```

```
## [1] TRUE
```

Desigualdades

Para comparar se um valor é maior que outro, temos à disposição 4 operadores:

```
# Maior  
3 > 3
```

```
## [1] FALSE
```

```
3 > 2
```

```
## [1] TRUE
```

```
# Maior ou igual  
3 > 4
```

```
## [1] FALSE
```

```
3 >= 3
```

```
## [1] TRUE
```



```
# Menor
```

```
3 < 3
```

```
## [1] FALSE
```

```
3 < 4
```

```
## [1] TRUE
```

```
# Menor ou igual
```

```
3 < 2
```

```
## [1] FALSE
```

```
3 <= 3
```

```
## [1] TRUE
```

Pertence

Um outro operador muito útil é o `%in%`. Com ele, podemos verificar se um valor está dentro de um conjunto de valores (vetor).

```
3 %in% c(1, 2, 3)
```

```
## [1] TRUE
```

```
"a" %in% c("b", "c")
```

```
## [1] FALSE
```

Filtros

Os testes lógicos fazem parte de uma operação muito comum na manipulação de base de dados: os **filtros**. No Excel, por exemplo, quando você filtra uma planilha, o que está sendo feito por trás é um teste lógico.

Falamos anteriormente que cada coluna das nossas bases de dados será representada dentro do R como um vetor. O comportamento que explica a importância dos testes lógicos na hora de filtrar uma base está ilustrado abaixo:

```
minha_coluna <- c(1, 3, 0, 10, -1, 5, 20)
minha_coluna > 3
```

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

```
minha_coluna[minha_coluna > 3]
```

```
## [1] 10  5 20
```

Muitas coisas aconteceram no código anterior, vamos por partes.

Primeiro, na operação `minha_coluna > 3` o R fez um excelente uso do comportamento de reciclagem. No fundo, o que ele fez foi transformar (reciclar) o valor 3 no vetor `c(3, 3, 3, 3, 3, 3, 3)` e testar se `c(1, 3, 0, 10, -1, 5, 20) > c(3, 3, 3, 3, 3, 3, 3)`.

Como os operadores lógicos também são vetorizados (fazem operações elemento a elemento), os testes realizados foram `1 > 3`, `3 > 3`, `0 > 3`, `10 > 3`, `-1 > 3`, `5 > 3` e, finalmente, `20 > 3`. Cada um desses testes tem o seu próprio resultado. Por isso a saída de `minha_coluna > 3` é um vetor de verdadeiros e falsos, respectivos a cada um desses 7 testes.

A segunda operação traz a grande novidade aqui: podemos usar os valores `TRUE` e `FALSE` para selecionar elementos de um vetor!

A regra é a seguinte: **retornar** as posições que receberem `TRUE`, **não retornar** as posições que receberem `FALSE`.

Portanto, a segunda operação é equivalente a:

```
minha_coluna[c(FALSE, FALSE, FALSE, TRUE, FALSE, TRUE, TRUE)]
```

```
## [1] 10 5 20
```

O vetor lógico filtra o vetor `minha_coluna`, retornando apenas os valores maiores que 3, já que foi esse o teste lógico que fizemos.

Essa é a *mágica* que acontece por trás de filtros no R. Na prática, não precisaremos usar colchetes, não lembraremos da reciclagem e nem veremos a cara dos `TRUE` e `FALSE`. Mas conhecer esse processo é muito importante, principalmente para encontrar problemas de código ou de base.

Mais sobre data frames

Chegou a hora de usarmos tudo o que aprendemos na seção anterior para explorarmos ao máximo o nosso objeto favorito: o *data frame*.

Para isso, continuaremos a usar o `mtcars`.

```
mtcars
```

```
##           mpg cyl  disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710      22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive  21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant         18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360      14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D       24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230        22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280        19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
```

Acessando as colunas

Lembrando que cada coluna de um *data frame* é um vetor, podemos usar o operador `$` para acessar cada uma de suas colunas.

```
mtcars$mpg
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4 10.4  
## [17] 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7 15.0 21.4
```

```
mtcars$cyl
```

```
## [1] 6 6 4 6 8 6 8 4 4 6 6 8 8 8 8 8 8 4 4 4 4 8 8 8 8 4 4 4 8 6 8 4
```

```
mtcars$wt
```

```
## [1] 2.620 2.875 2.320 3.215 3.440 3.460 3.570 3.190 3.150 3.440 3.440 4.070 3.730  
## [14] 3.780 5.250 5.424 5.345 2.200 1.615 1.835 2.465 3.520 3.435 3.840 3.845 1.935  
## [27] 2.140 1.513 3.170 2.770 3.570 2.780
```

Dimensões

A classe *data frame* possui uma característica especial: seus objetos possuem duas **dimensões**.

```
class(mtcars)
```

```
## [1] "data.frame"
```

```
dim(mtcars)
```

```
## [1] 32 11
```

O resultado do código `dim(mtcars)` nos diz que a primeira dimensão tem comprimento 32 e a segunda dimensão tem comprimento 11. Em outras palavras: a base `mtcars` tem 32 linhas e 11 colunas.

Subsetting

Ter duas dimensões significa que devemos usar dois índices para acessar os valores de um *data frame* (fazer *subsetting*). Para isso, ainda usamos o colchete, mas agora com dois argumentos: `[linha, coluna]`.

```
mtcars[2, 3]
```

```
## [1] 160
```

O código acima está nos devolvendo o valor presente na segunda linha da terceira coluna da base `mtcars`.

Também podemos pegar todos as linhas de uma coluna ou todas as colunas de uma linha deixando um dos argumentos vazio:

```
# Todas as linhas da coluna 1  
mtcars[,1]
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.4 10.4  
## [17] 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.7 15.0 21.4
```

```
# Todas as colunas da linha 1  
mtcars[1,]
```

```
##           mpg cyl disp  hp drat   wt  qsec vs am gear carb  
## Mazda RX4  21   6  160 110   3.9 2.62 16.46  0  1    4    4
```

Seleccionando columnas

Podemos usar o *subsetting* para seleccionar columnas:

```
mtcars[, c(1, 2)]
```

##		mpg	cyl
##	Mazda RX4	21.0	6
##	Mazda RX4 Wag	21.0	6
##	Datsun 710	22.8	4
##	Hornet 4 Drive	21.4	6
##	Hornet Sportabout	18.7	8
##	Valiant	18.1	6
##	Duster 360	14.3	8
##	Merc 240D	24.4	4
##	Merc 230	22.8	4
##	Merc 280	19.2	6

```
mtcars[, c("mpg", "am")]
```

```
##           mpg am
## Mazda RX4      21.0  1
## Mazda RX4 Wag  21.0  1
## Datsun 710     22.8  1
## Hornet 4 Drive  21.4  0
## Hornet Sportabout 18.7  0
## Valiant        18.1  0
## Duster 360     14.3  0
## Merc 240D      24.4  0
## Merc 230       22.8  0
## Merc 280       19.2  0
```

Nos dois exemplos, exibimos apenas as 5 primeiras linhas do *data frame*.

Selecionando colunas

- Existe mais de uma forma!
- Também podemos selecionar colunas usando o pacote `{dplyr}` e a função `select()` [leia mais aqui](#). Para isso, primeiramente precisaremos carregar o pacote `dplyr`, com a função `library()`, para que a função `select()` fique disponível:

```
library(dplyr)
```

```
##  
## Attaching package: 'dplyr'  
  
## The following objects are masked from 'package:stats':  
##  
##     filter, lag  
  
## The following objects are masked from 'package:base':  
##  
##     intersect, setdiff, setequal, union
```

```
# estrutura:  
# select(nome_do_dataset, quais_colunas_para_selecionar)
```

Selecionando columnas (cont.)

Selecionando **uma** coluna

```
select(mtcars, hp)
```

##	hp
## Mazda RX4	110
## Mazda RX4 Wag	110
## Datsun 710	93
## Hornet 4 Drive	110
## Hornet Sportabout	175
## Valiant	105
## Duster 360	245
## Merc 240D	62
## Merc 230	95
## Merc 280	123
## Merc 280C	123
## Merc 450SE	180
## Merc 450SL	180
## Merc 450SLC	180
## Cadillac Fleetwood	205
## Lincoln Continental	215
## Chrysler Imperial	230
## Fiat 128	66
## Honda Civic	52
## Toyota Corolla	65
## Toyota Corona	97
## Dodge Challenger	150

Selecionando colunas (cont.)

Selecionando **duas** colunas (ou mais)

```
select(mtcars, hp, drat)
```

##	hp	drat
## Mazda RX4	110	3.90
## Mazda RX4 Wag	110	3.90
## Datsun 710	93	3.85
## Hornet 4 Drive	110	3.08
## Hornet Sportabout	175	3.15
## Valiant	105	2.76
## Duster 360	245	3.21
## Merc 240D	62	3.69
## Merc 230	95	3.92
## Merc 280	123	3.92
## Merc 280C	123	3.92
## Merc 450SE	180	3.07
## Merc 450SL	180	3.07
## Merc 450SLC	180	3.07
## Cadillac Fleetwood	205	2.93
## Lincoln Continental	215	3.00
## Chrysler Imperial	230	3.23
## Fiat 128	66	4.08
## Honda Civic	52	4.93
## Toyota Corolla	65	4.22
## Toyota Corona	97	3.70
## Dodge Challenger	150	2.76

Selecionando columnas (cont.)

Selecionando todas as columnas, exceto alguma

```
select(mtcars, -hp) # selecionando todas as columnas, exceto alguma
```

##	mpg	cyl	disp	drat	wt	qsec	vs	am	gear	carb
## Mazda RX4	21.0	6	160.0	3.90	2.620	16.46	0	1	4	4
## Mazda RX4 Wag	21.0	6	160.0	3.90	2.875	17.02	0	1	4	4
## Datsun 710	22.8	4	108.0	3.85	2.320	18.61	1	1	4	1
## Hornet 4 Drive	21.4	6	258.0	3.08	3.215	19.44	1	0	3	1
## Hornet Sportabout	18.7	8	360.0	3.15	3.440	17.02	0	0	3	2
## Valiant	18.1	6	225.0	2.76	3.460	20.22	1	0	3	1
## Duster 360	14.3	8	360.0	3.21	3.570	15.84	0	0	3	4
## Merc 240D	24.4	4	146.7	3.69	3.190	20.00	1	0	4	2
## Merc 230	22.8	4	140.8	3.92	3.150	22.90	1	0	4	2
## Merc 280	19.2	6	167.6	3.92	3.440	18.30	1	0	4	4
## Merc 280C	17.8	6	167.6	3.92	3.440	18.90	1	0	4	4
## Merc 450SE	16.4	8	275.8	3.07	4.070	17.40	0	0	3	3
## Merc 450SL	17.3	8	275.8	3.07	3.730	17.60	0	0	3	3
## Merc 450SLC	15.2	8	275.8	3.07	3.780	18.00	0	0	3	3
## Cadillac Fleetwood	10.4	8	472.0	2.93	5.250	17.98	0	0	3	4
## Lincoln Continental	10.4	8	460.0	3.00	5.424	17.82	0	0	3	4
## Chrysler Imperial	14.7	8	440.0	3.23	5.345	17.42	0	0	3	4
## Fiat 128	32.4	4	78.7	4.08	2.200	19.47	1	1	4	1
## Honda Civic	30.4	4	75.7	4.93	1.615	18.52	1	1	4	2
## Toyota Corolla	33.9	4	71.1	4.22	1.835	19.90	1	1	4	1
## Toyota Corona	21.5	4	120.1	3.70	2.465	20.01	1	0	3	1
## Dodge Challenger	15.5	8	318.0	2.76	3.520	16.87	0	0	3	2

Filtrando linhas

Também podemos usar o *subsetting* para filtrar linhas:

```
mtcars$cyl == 6
```

```
## [1] TRUE TRUE FALSE TRUE FALSE TRUE FALSE FALSE FALSE TRUE TRUE FALSE FALSE
## [14] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [27] FALSE FALSE FALSE TRUE FALSE FALSE
```

```
mtcars[mtcars$cyl == 6, ]
```

```
##           mpg cyl  disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4      21.0   6 160.0 110 3.90 2.620 16.46 0  1    4    4
## Mazda RX4 Wag  21.0   6 160.0 110 3.90 2.875 17.02 0  1    4    4
## Hornet 4 Drive  21.4   6 258.0 110 3.08 3.215 19.44 1  0    3    1
## Valiant        18.1   6 225.0 105 2.76 3.460 20.22 1  0    3    1
## Merc 280       19.2   6 167.6 123 3.92 3.440 18.30 1  0    4    4
## Merc 280C      17.8   6 167.6 123 3.92 3.440 18.90 1  0    4    4
## Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.50 0  1    5    6
```

O código `mtcars$cyl == 6` nos diz em quais linhas estão os carros com 4 cilindros. Quando usamos o vetor de `TRUE` e `FALSE` resultante dentro do *subsetting* das linhas em `mtcars[mtcars$cyl == 6,]`, o R nos devolve todos as colunas dos carros com 4 cilindros. A regra é a seguinte: linha com `TRUE` é retornada, linha com `FALSE` não.

Outro exemplo:

```
mtcars[mtcars$mpg > 20, ]
```

##		mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
##	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
##	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
##	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
##	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
##	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
##	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
##	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
##	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
##	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
##	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
##	Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
##	Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
##	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
##	Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

Filtrando linhas com o pacote {dplyr}

- Também podemos selecionar colunas usando o pacote {dplyr} e a função `filter()` [leia mais aqui](#). Para isso, primeiramente precisaremos carregar o pacote `dplyr`, com a função `library()`, para que a função `filter()` fique disponível:

```
library(dplyr)
# estrutura:
# filter(nome_do_dataset, condicoes_para_filtrar)
```

- A seguir, apresentamos os mesmos exemplos anteriores, adaptados para funcionar com a função `filter()`.

Filtrando linhas com o pacote {dplyr} (cont.)

```
# em R base:  
# mtcars[mtcars$cyl == 6, ]  
# com dplyr  
filter(mtcars, cyl == 6)
```

##		mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
##	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
##	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
##	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
##	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
##	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
##	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
##	Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6

Filtrando linhas com o pacote {dplyr} (cont.)

```
# em R base:  
# mtcars[mtcars$mpg > 20, ]  
# com dplyr  
filter(mtcars, mpg > 20)
```

##		mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
##	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
##	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
##	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
##	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
##	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
##	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
##	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
##	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
##	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
##	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
##	Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
##	Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
##	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
##	Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

Filtrando linhas com o pacote {dplyr} (cont.)

- É possível combinar filtros. A vírgula neste contexto funciona como o &:

```
# Combinando filtros!  
# exemplo abaixo: filtrar carros que tenham 4 cilindros, e mais de 20 mpg  
filter(mtcars, cyl == 6, mpg > 20)
```

##		mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
##	Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
##	Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
##	Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1

Filtrando linhas com o pacote {dplyr} (cont.)

- Podemos combinar filtros usando o que aprendemos com os operadores: podemos também usar o OU |

```
# Combinando filtros!  
# exemplo abaixo: filtrar carros que tenham 4 cilindros OU mais de 20 mpg  
filter(mtcars, cyl == 6 | mpg > 20)
```

##		mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
##	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
##	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
##	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
##	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
##	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
##	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
##	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
##	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
##	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
##	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
##	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
##	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
##	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
##	Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
##	Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
##	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
##	Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
##	Ford Torino 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

Filtrando linhas com o pacote {dplyr} (cont.)

- Podemos combinar filtros usando o que aprendemos com os operadores: podemos também usar o NOT !

```
# Podemos também usar o NOT: !  
# exemplo abaixo: filtrar carros que NÃO tenham 4 cilindros  
filter(mtcars, ! cyl == 6)
```

##	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
## Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
## Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
## Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
## Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
## Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
## Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
## Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
## Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
## Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
## Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
## Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
## Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
## Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
## Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
## Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
## Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
## AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
## Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
## Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
## Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1

Filtrando linhas com o pacote {dplyr} (cont.)

- Podemos usar também o operador `%in%`:

```
# Queremos filtrar os carros que tenham 4 ou 6 cilindros  
filter(mtcars, cyl %in% c(4,6))
```

##		mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
##	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
##	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
##	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
##	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
##	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
##	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
##	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
##	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
##	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
##	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
##	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
##	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
##	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
##	Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
##	Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
##	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
##	Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
##	Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2