| Se a sua versão do dplyr não for pelo menos 0.4.0, você deve pressionar a tecla Esc agora, reinstalar dplyr e continuar esta lição de onde parou.

...

|=============================== | 15%

| A primeira etapa do trabalho com dados no dplyr é carregar os dados no que os autores do pacote chamam de data.frame\_tbl ou tbl\_df. Use o código a seguir para criar um novo tbl\_df chamado cran: cran <- tbl\_df(mydf).

> cran <- tbl\_df(my\_df)

Error in as\_tibble(data, .name\_repair = "check\_unique") :

object 'my\_df' not found

> cran <- tbl\_df(mydf)

| Maravilha!

|=================================== | 16%

| O dataset novo, chamado cran, é uma especialização do data.frame. Se quiser, veja as classes deste dataset.

...

|====================================== | 18%

| E se quiser saber mais sobre a classe tbl\_df, digite o comando ?tbl\_df.

...

|========================================== | 19%

| Para evitar confusão e manter as coisas funcionando sem problemas, vamos remover o data.frame original de sua área de trabalho com rm("mydf").

> rm('mydf')

| Ótimo trabalho!

|============================================= | 21%

| A principal vantagem de usar um tbl\_df ao invés de um data.frame regular é a impressão na tela. Vamos ver o que isso significa. Digite cran para imprimir nosso tbl\_df no console.

> cran

# A tibble: 225,468 × 11

X date time size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 1 2014-07-08 00:54:41 80589 3.1.0 x86\_64 mingw32 htmltools 0.2.4 US 1

2 2 2014-07-08 00:59:53 321767 3.1.0 x86\_64 mingw32 tseries 0.10-32 US 2

3 3 2014-07-08 00:47:13 748063 3.1.0 x86\_64 linux-gnu party 1.0-15 US 3

4 4 2014-07-08 00:48:05 606104 3.1.0 x86\_64 linux-gnu Hmisc 3.14-4 US 3

5 5 2014-07-08 00:46:50 79825 3.0.2 x86\_64 linux-gnu digest 0.6.4 CA 4

6 6 2014-07-08 00:48:04 77681 3.1.0 x86\_64 linux-gnu randomForest 4.6-7 US 3

7 7 2014-07-08 00:48:35 393754 3.1.0 x86\_64 linux-gnu plyr 1.8.1 US 3

8 8 2014-07-08 00:47:30 28216 3.0.2 x86\_64 linux-gnu whisker 0.3-2 US 5

9 9 2014-07-08 00:54:58 5928 NA NA NA Rcpp 0.10.4 CN 6

10 10 2014-07-08 00:15:35 2206029 3.0.2 x86\_64 linux-gnu hflights 0.1 US 7

# … with 225,458 more rows

| Excelente trabalho!

|================================================= | 23%

| Essa saída é muito mais informativa e compacta do que obteríamos se imprimíssemos o quadro de dados original (mydf) no console.

...

|==================================================== | 24%

| Primeiro, mostramos a classe e as dimensões do conjunto de dados. Logo abaixo disso, temos uma prévia dos dados. Em vez de tentar imprimir o conjunto de dados inteiro, dplyr nos mostra apenas as primeiras 10 linhas de dados

| e apenas quantas colunas se encaixam perfeitamente em nosso console. Na parte inferior, vemos os nomes e as classes de quaisquer variáveis que não se encaixam em nossa tela.

...

|======================================================== | 26%

| De acordo com a vinheta "Introduction to dplyr" escrita pelos autores do pacote, "A filosofia dplyr é ter pequenas funções que fazem uma coisa bem feita." Especificamente, o dplyr fornece cinco 'funções principais' que

| cobrem a maioria das tarefas fundamentais de manipulação de dados select(), filter(), arrange(), mutate() e summarize().

...

|=========================================================== | 27%

| Digite ?select para puxar a documentação para a primeira dessas funções principais.

> ?select

| Excelente trabalho!

|=============================================================== | 29%

| O help para as outras funções estão acessíveis da mesma maneira.

...

|================================================================== | 31%

| Como muitas vezes pode ser o caso, particularmente com conjuntos de dados maiores, estamos interessados

| apenas em algumas das variáveis. Use select(cran, ip\_id, package, country) para selecionar apenas as

| variáveis/colunas ip\_id, package e country do dataset do cran.

> select(cran, ip\_id, package, country)

# A tibble: 225,468 × 3

ip\_id package country

<int> <chr> <chr>

1 1 htmltools US

2 2 tseries US

3 3 party US

4 3 Hmisc US

5 4 digest CA

6 3 randomForest US

7 3 plyr US

8 5 whisker US

9 6 Rcpp CN

10 7 hflights US

# … with 225,458 more rows

| Todo o trabalho duro está dando resultado!

|====================================================================== | 32%

| A primeira coisa a notar é que não precisamos digitar cran$ip\_id, cran$package e cran$country, como

| normalmente faríamos ao nos referirmos a colunas de um data.frame. A função select() sabe que estamos nos

| referindo às colunas do conjunto de dados do cran.

...

|========================================================================== | 34%

| Além disso, observe que as colunas são retornadas para nós na ordem que especificamos, embora ip\_id seja a

| coluna mais à direita no dataset original.

...

|============================================================================= | 35%

| Lembre-se que no R, o operador `:` fornece uma notação compacta para criar uma sequência de números. Por

| exemplo, digite 5:20 para relembrar.

> 5:20

[1] 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

| Mantenha esse bom nível!

|================================================================================ | 37%

| Normalmente, essa notação é reservada para números, mas select() permite que você especifique uma sequência

| de colunas dessa maneira, o que pode economizar um monte de digitação. Use select(cran, r\_arch:country) para

| selecionar todas as colunas a partir de r\_arch e terminando com country.

> select(cran, r\_arch:country)

# A tibble: 225,468 × 5

r\_arch r\_os package version country

<chr> <chr> <chr> <chr> <chr>

1 x86\_64 mingw32 htmltools 0.2.4 US

2 x86\_64 mingw32 tseries 0.10-32 US

3 x86\_64 linux-gnu party 1.0-15 US

4 x86\_64 linux-gnu Hmisc 3.14-4 US

5 x86\_64 linux-gnu digest 0.6.4 CA

6 x86\_64 linux-gnu randomForest 4.6-7 US

7 x86\_64 linux-gnu plyr 1.8.1 US

8 x86\_64 linux-gnu whisker 0.3-2 US

9 NA NA Rcpp 0.10.4 CN

10 x86\_64 linux-gnu hflights 0.1 US

# … with 225,458 more rows

| Todo o trabalho duro está dando resultado!

|==================================================================================== | 39%

| Também podemos selecionar as mesmas colunas na ordem inversa. Tenta aí.

> select(cran, country:r\_arch)

# A tibble: 225,468 × 5

country version package r\_os r\_arch

<chr> <chr> <chr> <chr> <chr>

1 US 0.2.4 htmltools mingw32 x86\_64

2 US 0.10-32 tseries mingw32 x86\_64

3 US 1.0-15 party linux-gnu x86\_64

4 US 3.14-4 Hmisc linux-gnu x86\_64

5 CA 0.6.4 digest linux-gnu x86\_64

6 US 4.6-7 randomForest linux-gnu x86\_64

7 US 1.8.1 plyr linux-gnu x86\_64

8 US 0.3-2 whisker linux-gnu x86\_64

9 CN 0.10.4 Rcpp NA NA

10 US 0.1 hflights linux-gnu x86\_64

# … with 225,458 more rows

| Você está em um bom ritmo!

|======================================================================================== | 40%

| Imprima o conjunto de dados inteiro novamente, apenas para se lembrar de como ele é. Você pode fazer isso a

| qualquer momento durante a aula.

> cran

# A tibble: 225,468 × 11

X date time size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 1 2014-07-08 00:54:41 80589 3.1.0 x86\_64 mingw32 htmltools 0.2.4 US 1

2 2 2014-07-08 00:59:53 321767 3.1.0 x86\_64 mingw32 tseries 0.10-32 US 2

3 3 2014-07-08 00:47:13 748063 3.1.0 x86\_64 linux-gnu party 1.0-15 US 3

4 4 2014-07-08 00:48:05 606104 3.1.0 x86\_64 linux-gnu Hmisc 3.14-4 US 3

5 5 2014-07-08 00:46:50 79825 3.0.2 x86\_64 linux-gnu digest 0.6.4 CA 4

6 6 2014-07-08 00:48:04 77681 3.1.0 x86\_64 linux-gnu randomForest 4.6-7 US 3

7 7 2014-07-08 00:48:35 393754 3.1.0 x86\_64 linux-gnu plyr 1.8.1 US 3

8 8 2014-07-08 00:47:30 28216 3.0.2 x86\_64 linux-gnu whisker 0.3-2 US 5

9 9 2014-07-08 00:54:58 5928 NA NA NA Rcpp 0.10.4 CN 6

10 10 2014-07-08 00:15:35 2206029 3.0.2 x86\_64 linux-gnu hflights 0.1 US 7

# … with 225,458 more rows

| Perseverança é a resposta.

|=========================================================================================== | 42%

| Em vez de especificar as colunas que queremos manter, também podemos especificar as colunas que queremos

| descartar. Para ver como isso funciona, use select(cran, -time) para omitir a coluna de tempo.

> select(cran, -time)

# A tibble: 225,468 × 10

X date size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 1 2014-07-08 80589 3.1.0 x86\_64 mingw32 htmltools 0.2.4 US 1

2 2 2014-07-08 321767 3.1.0 x86\_64 mingw32 tseries 0.10-32 US 2

3 3 2014-07-08 748063 3.1.0 x86\_64 linux-gnu party 1.0-15 US 3

4 4 2014-07-08 606104 3.1.0 x86\_64 linux-gnu Hmisc 3.14-4 US 3

5 5 2014-07-08 79825 3.0.2 x86\_64 linux-gnu digest 0.6.4 CA 4

6 6 2014-07-08 77681 3.1.0 x86\_64 linux-gnu randomForest 4.6-7 US 3

7 7 2014-07-08 393754 3.1.0 x86\_64 linux-gnu plyr 1.8.1 US 3

8 8 2014-07-08 28216 3.0.2 x86\_64 linux-gnu whisker 0.3-2 US 5

9 9 2014-07-08 5928 NA NA NA Rcpp 0.10.4 CN 6

10 10 2014-07-08 2206029 3.0.2 x86\_64 linux-gnu hflights 0.1 US 7

# … with 225,458 more rows

| Todo o trabalho duro está dando resultado!

|============================================================================================== | 44%

| O sinal negativo na frente de time informa ao select() que NÃO queremos a coluna de tempo. Agora, vamos

| combinar estratégias para omitir todas as colunas do X através do size(X:size). Para ver como isso pode

| funcionar, vamos ver um exemplo numérico com -5:20.

>

> select(cran, -5:20)

Error in `select()`:

! Can't subset columns past the end.

ℹ Locations 12, 13, 14, 15, 16, … don't exist.

ℹ There are only 11 columns.

Run `rlang::last\_error()` to see where the error occurred.

Warning message:

In x:y : numerical expression has 10 elements: only the first used

> size(-5:20)

Error in size(-5:20) : could not find function "size"

> select(cran, -5:20)

Error in `select()`:

! Can't subset columns past the end.

ℹ Locations 12, 13, 14, 15, 16, … don't exist.

ℹ There are only 11 columns.

Run `rlang::last\_error()` to see where the error occurred.

Warning message:

In x:y : numerical expression has 10 elements: only the first used

> select(cran, 5:20)

Error in `select()`:

! Can't subset columns past the end.

ℹ Locations 12, 13, 14, 15, 16, … don't exist.

ℹ There are only 11 columns.

Run `rlang::last\_error()` to see where the error occurred.

> select(cran, 5:7)

# A tibble: 225,468 × 3

r\_version r\_arch r\_os

<chr> <chr> <chr>

1 3.1.0 x86\_64 mingw32

2 3.1.0 x86\_64 mingw32

3 3.1.0 x86\_64 linux-gnu

4 3.1.0 x86\_64 linux-gnu

5 3.0.2 x86\_64 linux-gnu

6 3.1.0 x86\_64 linux-gnu

7 3.1.0 x86\_64 linux-gnu

8 3.0.2 x86\_64 linux-gnu

9 NA NA NA

10 3.0.2 x86\_64 linux-gnu

# … with 225,458 more rows

| Mas uma vez. Você consegue! Ou digite info() para mais opções.

| Veja o que -5:20 nos dá.

> -5:20

[1] -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

| Você está em um bom ritmo!

|================================================================================================== | 45%

| Opa! Isso nos dá um vetor de números de -5 a 20, o que não é o que queremos. Em vez disso, queremos negar

| toda a seqüência de números de 5 a 20, de modo a obtermos -5, -6, -7, ..., -18, -19, -20. Tente a mesma

| coisa, com -(5:20), é com parênteses! Desso modo o R saberá que nós queremos que primeiro surja a sequência

| de números, depois aplique o sinal negativo à coisa toda.

> -(5:20)

[1] -5 -6 -7 -8 -9 -10 -11 -12 -13 -14 -15 -16 -17 -18 -19 -20

| Essa é a resposta que eu esperava.

|====================================================================================================== | 47%

| Use este padrão para omitir todas as colunas X:size usando select().

> select(cran, -(5:20))

Error in `select()`:

! Can't subset columns past the end.

ℹ Locations 12, 13, 14, 15, 16, … don't exist.

ℹ There are only 11 columns.

Run `rlang::last\_error()` to see where the error occurred.

> select(cran, -(5:6))

# A tibble: 225,468 × 9

X date time size r\_os package version country ip\_id

<int> <chr> <chr> <int> <chr> <chr> <chr> <chr> <int>

1 1 2014-07-08 00:54:41 80589 mingw32 htmltools 0.2.4 US 1

2 2 2014-07-08 00:59:53 321767 mingw32 tseries 0.10-32 US 2

3 3 2014-07-08 00:47:13 748063 linux-gnu party 1.0-15 US 3

4 4 2014-07-08 00:48:05 606104 linux-gnu Hmisc 3.14-4 US 3

5 5 2014-07-08 00:46:50 79825 linux-gnu digest 0.6.4 CA 4

6 6 2014-07-08 00:48:04 77681 linux-gnu randomForest 4.6-7 US 3

7 7 2014-07-08 00:48:35 393754 linux-gnu plyr 1.8.1 US 3

8 8 2014-07-08 00:47:30 28216 linux-gnu whisker 0.3-2 US 5

9 9 2014-07-08 00:54:58 5928 NA Rcpp 0.10.4 CN 6

10 10 2014-07-08 00:15:35 2206029 linux-gnu hflights 0.1 US 7

# … with 225,458 more rows

| Você quase acertou. Tente novamente. Ou digite info() para mais opções.

| Use select(cran, -(X:size)) para omitir todas as colunas de X através do tamanho.

> select(cran, -(X:size))

# A tibble: 225,468 × 7

r\_version r\_arch r\_os package version country ip\_id

<chr> <chr> <chr> <chr> <chr> <chr> <int>

1 3.1.0 x86\_64 mingw32 htmltools 0.2.4 US 1

2 3.1.0 x86\_64 mingw32 tseries 0.10-32 US 2

3 3.1.0 x86\_64 linux-gnu party 1.0-15 US 3

4 3.1.0 x86\_64 linux-gnu Hmisc 3.14-4 US 3

5 3.0.2 x86\_64 linux-gnu digest 0.6.4 CA 4

6 3.1.0 x86\_64 linux-gnu randomForest 4.6-7 US 3

7 3.1.0 x86\_64 linux-gnu plyr 1.8.1 US 3

8 3.0.2 x86\_64 linux-gnu whisker 0.3-2 US 5

9 NA NA NA Rcpp 0.10.4 CN 6

10 3.0.2 x86\_64 linux-gnu hflights 0.1 US 7

# … with 225,458 more rows

| Continue assim e você chegará lá!

|========================================================================================================= | 48%

| Agora que você sabe como selecionar um subconjunto de colunas usando select (), uma próxima pergunta natural

| é "Como eu seleciono um subconjunto de linhas?" É aí que entra a função filter().

...

|============================================================================================================ | 50%

| Use filter(cran, package == "swirl") para selecionar todas as linhas para as quais a variável do pacote é

| igual a "swirl". Certifique-se de usar dois sinais de igual!

> filter(cran, package == 'swirl')

# A tibble: 820 × 11

X date time size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 27 2014-07-08 00:17:16 105350 3.0.2 x86\_64 mingw32 swirl 2.2.9 US 20

2 156 2014-07-08 00:22:53 41261 3.1.0 x86\_64 linux-gnu swirl 2.2.9 US 66

3 358 2014-07-08 00:13:42 105335 2.15.2 x86\_64 mingw32 swirl 2.2.9 CA 115

4 593 2014-07-08 00:59:45 105465 3.1.0 x86\_64 darwin13.1.0 swirl 2.2.9 MX 162

5 831 2014-07-08 00:55:27 105335 3.0.3 x86\_64 mingw32 swirl 2.2.9 US 57

6 997 2014-07-08 00:33:06 41261 3.1.0 x86\_64 mingw32 swirl 2.2.9 US 70

7 1023 2014-07-08 00:35:36 106393 3.1.0 x86\_64 mingw32 swirl 2.2.9 BR 248

8 1144 2014-07-08 00:00:39 106534 3.0.2 x86\_64 linux-gnu swirl 2.2.9 US 261

9 1402 2014-07-08 00:41:41 41261 3.1.0 i386 mingw32 swirl 2.2.9 US 234

10 1424 2014-07-08 00:44:49 106393 3.1.0 x86\_64 linux-gnu swirl 2.2.9 US 301

# … with 810 more rows

| Está correto!

|================================================================================================================ | 52%

| Novamente, note que filter() reconhece 'package' como uma coluna de cran, sem que você tenha que especificar

| explicitamente o cran$package.

...

|==================================================================================================================== | 53%

| O operador == pergunta se o valor à esquerda é igual ao valor à direita. Se sim, então retorna TRUE. Se não,

| FALSE. Neste caso, package é um vetor inteiro (coluna) de valores, então package == "swirl" retorna um vetor

| de TRUEs e FALSEs. Em seguida, filter() retorna apenas as linhas do cran correspondendo aos TRUEs.

...

|======================================================================================================================= | 55%

| Você pode especificar quantas condições desejar, separadas por vírgulas. Por exemplo, filter(cran, r\_version

| == "3.1.1", country == "US") retornará todas as linhas de referência correspondentes aos downloads dos

| usuários nos EUA que executam a versão R 3.1.1. Experimente.

> filter(cran, r\_version == '3.1.1', country == 'US')

# A tibble: 1,588 × 11

X date time size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 2216 2014-07-08 00:48:58 385112 3.1.1 x86\_64 darwin13.1.0 colorspace 1.2-4 US 191

2 17332 2014-07-08 03:39:57 197459 3.1.1 x86\_64 darwin13.1.0 httr 0.3 US 1704

3 17465 2014-07-08 03:25:38 23259 3.1.1 x86\_64 darwin13.1.0 snow 0.3-13 US 62

4 18844 2014-07-08 03:59:17 190594 3.1.1 x86\_64 darwin13.1.0 maxLik 1.2-0 US 1533

5 30182 2014-07-08 04:13:15 77683 3.1.1 i386 mingw32 randomForest 4.6-7 US 646

6 30193 2014-07-08 04:06:26 2351969 3.1.1 i386 mingw32 ggplot2 1.0.0 US 8

7 30195 2014-07-08 04:07:09 299080 3.1.1 i386 mingw32 fExtremes 3010.81 US 2010

8 30217 2014-07-08 04:32:04 568036 3.1.1 i386 mingw32 rJava 0.9-6 US 98

9 30245 2014-07-08 04:10:41 526858 3.1.1 i386 mingw32 LPCM 0.44-8 US 8

10 30354 2014-07-08 04:32:51 1763717 3.1.1 i386 mingw32 mgcv 1.8-1 US 2122

# … with 1,578 more rows

| Você acertou!

|========================================================================================================================== | 56%

| As condições transmitidas para filter() podem fazer uso de qualquer um dos operadores de comparação padrão.

| A documentação desses operadores é feita com "Comparison" (que é um C maiúsculo).

>

> ?Comparison

| Você acertou!

|============================================================================================================================== | 58%

| Edite sua chamada anterior para filter() para retornar as linhas correspondentes aos usuários em "IN"

| (Índia) executando uma versão R que seja menor ou igual a "3.0.2". A seta para cima no teclado pode ser útil

| aqui. Não esqueça suas aspas duplas!

> filter(cran, country == 'India', r\_version <= "3.0.2")

# A tibble: 0 × 11

# … with 11 variables: X <int>, date <chr>, time <chr>, size <int>, r\_version <chr>, r\_arch <chr>, r\_os <chr>,

# package <chr>, version <chr>, country <chr>, ip\_id <int>

| Está perto... posso sentir! Tente novamente. Ou digite info() para mais opções.

| filter(cran, r\_version <= "3.0.2", country == "IN") retornará todas as linhas para as quais r\_version é

| menor ou igual a "3.0.2" e country é igual a "IN".

> filter(cran, country == 'IN', r\_version <= "3.0.2")

# A tibble: 4,139 × 11

X date time size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 348 2014-07-08 00:44:04 10218907 3.0.0 x86\_64 mingw32 BH 1.54.0-2 IN 112

2 9990 2014-07-08 02:11:32 397497 3.0.2 x86\_64 linux-gnu equateIRT 1.1 IN 1054

3 9991 2014-07-08 02:11:32 119199 3.0.2 x86\_64 linux-gnu ggdendro 0.1-14 IN 1054

4 9992 2014-07-08 02:11:33 81779 3.0.2 x86\_64 linux-gnu dfcrm 0.2-2 IN 1054

5 10022 2014-07-08 02:19:45 1557078 2.15.0 x86\_64 mingw32 RcppArmadillo 0.4.320.0 IN 1060

6 10023 2014-07-08 02:19:46 1184285 2.15.1 i686 linux-gnu forecast 5.4 IN 1060

7 10189 2014-07-08 02:38:06 908854 3.0.2 x86\_64 linux-gnu editrules 2.7.2 IN 1054

8 10199 2014-07-08 02:38:28 178436 3.0.2 x86\_64 linux-gnu energy 1.6.1 IN 1054

9 10200 2014-07-08 02:38:29 51811 3.0.2 x86\_64 linux-gnu ENmisc 1.2-7 IN 1054

10 10201 2014-07-08 02:38:29 65245 3.0.2 x86\_64 linux-gnu entropy 1.2.0 IN 1054

# … with 4,129 more rows

| Toda a prática está rendendo frutos!

|================================================================================================================================== | 60%

| Nossas últimas duas chamadas para filter() solicitaram todas as linhas para as quais alguma condição E outra

| condição foram TRUE. Também podemos solicitar linhas para as quais UMA condição OU outra condição são

| VERDADEIRAS. Por exemplo, filter(cran, country == "US" | country == "IN") nos dará todas as linhas para as

| quais a variável country é igual a "US" ou "IN". Tenta aí.

> filter(cran, country == 'IN' | country == 'US')

# A tibble: 95,283 × 11

X date time size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 1 2014-07-08 00:54:41 80589 3.1.0 x86\_64 mingw32 htmltools 0.2.4 US 1

2 2 2014-07-08 00:59:53 321767 3.1.0 x86\_64 mingw32 tseries 0.10-32 US 2

3 3 2014-07-08 00:47:13 748063 3.1.0 x86\_64 linux-gnu party 1.0-15 US 3

4 4 2014-07-08 00:48:05 606104 3.1.0 x86\_64 linux-gnu Hmisc 3.14-4 US 3

5 6 2014-07-08 00:48:04 77681 3.1.0 x86\_64 linux-gnu randomForest 4.6-7 US 3

6 7 2014-07-08 00:48:35 393754 3.1.0 x86\_64 linux-gnu plyr 1.8.1 US 3

7 8 2014-07-08 00:47:30 28216 3.0.2 x86\_64 linux-gnu whisker 0.3-2 US 5

8 10 2014-07-08 00:15:35 2206029 3.0.2 x86\_64 linux-gnu hflights 0.1 US 7

9 11 2014-07-08 00:15:25 526858 3.0.2 x86\_64 linux-gnu LPCM 0.44-8 US 8

10 12 2014-07-08 00:14:45 2351969 2.14.1 x86\_64 linux-gnu ggplot2 1.0.0 US 8

# … with 95,273 more rows

| Excelente trabalho!

|===================================================================================================================================== | 61%

| Agora, use filter() para buscar todas as linhas cujo size é estritamente maior que (>) 100500 (sem aspas, já

| que o size é numérico) e r\_os é igual a "linux-gnu". Dica: Você está passando três argumentos para filter():

| o nome do conjunto de dados, a primeira condição e a segunda condição.

> filter(cran, size > 100500 | country == 'US')

# A tibble: 176,669 × 11

X date time size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 1 2014-07-08 00:54:41 80589 3.1.0 x86\_64 mingw32 htmltools 0.2.4 US 1

2 2 2014-07-08 00:59:53 321767 3.1.0 x86\_64 mingw32 tseries 0.10-32 US 2

3 3 2014-07-08 00:47:13 748063 3.1.0 x86\_64 linux-gnu party 1.0-15 US 3

4 4 2014-07-08 00:48:05 606104 3.1.0 x86\_64 linux-gnu Hmisc 3.14-4 US 3

5 6 2014-07-08 00:48:04 77681 3.1.0 x86\_64 linux-gnu randomForest 4.6-7 US 3

6 7 2014-07-08 00:48:35 393754 3.1.0 x86\_64 linux-gnu plyr 1.8.1 US 3

7 8 2014-07-08 00:47:30 28216 3.0.2 x86\_64 linux-gnu whisker 0.3-2 US 5

8 10 2014-07-08 00:15:35 2206029 3.0.2 x86\_64 linux-gnu hflights 0.1 US 7

9 11 2014-07-08 00:15:25 526858 3.0.2 x86\_64 linux-gnu LPCM 0.44-8 US 8

10 12 2014-07-08 00:14:45 2351969 2.14.1 x86\_64 linux-gnu ggplot2 1.0.0 US 8

# … with 176,659 more rows

| Você quase acertou. Tente novamente. Ou digite info() para mais opções.

| Isso é complicado né! filter(cran, size > 100500, r\_os == "linux-gnu") nos dará todas as linhas cujo tamanho

| é estritamente maior que 100500 e r\_os é "linux-gnu".

> filter(cran, size > 100500, r\_os=='linux-gnu')

# A tibble: 33,683 × 11

X date time size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 3 2014-07-08 00:47:13 748063 3.1.0 x86\_64 linux-gnu party 1.0-15 US 3

2 4 2014-07-08 00:48:05 606104 3.1.0 x86\_64 linux-gnu Hmisc 3.14-4 US 3

3 7 2014-07-08 00:48:35 393754 3.1.0 x86\_64 linux-gnu plyr 1.8.1 US 3

4 10 2014-07-08 00:15:35 2206029 3.0.2 x86\_64 linux-gnu hflights 0.1 US 7

5 11 2014-07-08 00:15:25 526858 3.0.2 x86\_64 linux-gnu LPCM 0.44-8 US 8

6 12 2014-07-08 00:14:45 2351969 2.14.1 x86\_64 linux-gnu ggplot2 1.0.0 US 8

7 14 2014-07-08 00:15:35 3097729 3.0.2 x86\_64 linux-gnu Rcpp 0.9.7 VE 10

8 15 2014-07-08 00:14:37 568036 3.1.0 x86\_64 linux-gnu rJava 0.9-6 US 11

9 16 2014-07-08 00:15:50 1600441 3.1.0 x86\_64 linux-gnu RSQLite 0.11.4 US 7

10 18 2014-07-08 00:26:59 186685 3.1.0 x86\_64 linux-gnu ipred 0.9-3 DE 13

# … with 33,673 more rows

| Todo o trabalho duro está dando resultado!

|======================================================================================================================================== | 63%

| Finalmente, queremos obter apenas as linhas para as quais a r\_version não está faltando. R representa

| valores ausentes com NA e esses valores ausentes podem ser detectados usando a função is.na().

...

|============================================================================================================================================ | 65%

| Para ver como isso funciona, digite is.na(c(3, 5, NA, 10)).

> is.na(3,5,NA,10)

Error in is.na(3, 5, NA, 10) :

4 arguments passed to 'is.na' which requires 1

> is.na(c(3,5,NA,10))

[1] FALSE FALSE TRUE FALSE

| Sua dedicação é inspiradora!

|================================================================================================================================================ | 66%

| Agora, coloque um ponto de exclamação (!) Antes de is.na() para mudar todos os TRUEs para FALSEs e todos os

| FALSEs para TRUEs, dizendo-nos o que NÃO é NA: !is.na (c (3, 5, NA, 10)).

> !is.na(c(3,5,NA,10))

[1] TRUE TRUE FALSE TRUE

| Essa é a resposta que eu esperava.

|=================================================================================================================================================== | 68%

| Ok, pronto para colocar tudo isso junto? Use filter() para retornar todas as linhas do cran para as quais

| r\_version NÃO é NA. Dica: Você precisará usar !is.na() como parte de seu segundo argumento para filter().

> filter(cran, size > 100500, r\_os=='linux-gnu', !is\_na(R.version))

Error in `filter()`:

! Problem while computing `..3 = !is\_na(R.version)`.

Caused by error in `is\_na()`:

! could not find function "is\_na"

Run `rlang::last\_error()` to see where the error occurred.

> filter(cran, size > 100500, r\_os=='linux-gnu', !is.na(R.version))

Error in `filter()`:

! Problem while computing `..3 = !is.na(R.version)`.

✖ Input `..3` must be of size 225468 or 1, not size 15.

Run `rlang::last\_error()` to see where the error occurred.

> filter(cran, size > 100500, r\_os=='linux-gnu', !is.na(r\_version))

# A tibble: 33,683 × 11

X date time size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 3 2014-07-08 00:47:13 748063 3.1.0 x86\_64 linux-gnu party 1.0-15 US 3

2 4 2014-07-08 00:48:05 606104 3.1.0 x86\_64 linux-gnu Hmisc 3.14-4 US 3

3 7 2014-07-08 00:48:35 393754 3.1.0 x86\_64 linux-gnu plyr 1.8.1 US 3

4 10 2014-07-08 00:15:35 2206029 3.0.2 x86\_64 linux-gnu hflights 0.1 US 7

5 11 2014-07-08 00:15:25 526858 3.0.2 x86\_64 linux-gnu LPCM 0.44-8 US 8

6 12 2014-07-08 00:14:45 2351969 2.14.1 x86\_64 linux-gnu ggplot2 1.0.0 US 8

7 14 2014-07-08 00:15:35 3097729 3.0.2 x86\_64 linux-gnu Rcpp 0.9.7 VE 10

8 15 2014-07-08 00:14:37 568036 3.1.0 x86\_64 linux-gnu rJava 0.9-6 US 11

9 16 2014-07-08 00:15:50 1600441 3.1.0 x86\_64 linux-gnu RSQLite 0.11.4 US 7

10 18 2014-07-08 00:26:59 186685 3.1.0 x86\_64 linux-gnu ipred 0.9-3 DE 13

# … with 33,673 more rows

| Essa não é a resposta esperada, mas tente novamente. Ou digite info() para mais opções.

| Este é outro complicado. filter(cran, !is.na(r\_version)) retornará todas as linhas do cran para as quais

| r\_version NÃO é NA.

> filter(cran, !is.na(r\_version))

# A tibble: 207,205 × 11

X date time size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 1 2014-07-08 00:54:41 80589 3.1.0 x86\_64 mingw32 htmltools 0.2.4 US 1

2 2 2014-07-08 00:59:53 321767 3.1.0 x86\_64 mingw32 tseries 0.10-32 US 2

3 3 2014-07-08 00:47:13 748063 3.1.0 x86\_64 linux-gnu party 1.0-15 US 3

4 4 2014-07-08 00:48:05 606104 3.1.0 x86\_64 linux-gnu Hmisc 3.14-4 US 3

5 5 2014-07-08 00:46:50 79825 3.0.2 x86\_64 linux-gnu digest 0.6.4 CA 4

6 6 2014-07-08 00:48:04 77681 3.1.0 x86\_64 linux-gnu randomForest 4.6-7 US 3

7 7 2014-07-08 00:48:35 393754 3.1.0 x86\_64 linux-gnu plyr 1.8.1 US 3

8 8 2014-07-08 00:47:30 28216 3.0.2 x86\_64 linux-gnu whisker 0.3-2 US 5

9 10 2014-07-08 00:15:35 2206029 3.0.2 x86\_64 linux-gnu hflights 0.1 US 7

10 11 2014-07-08 00:15:25 526858 3.0.2 x86\_64 linux-gnu LPCM 0.44-8 US 8

# … with 207,195 more rows

| Ótimo trabalho!

|====================================================================================================================================================== | 69%

| Vimos como selecionar um subconjunto de colunas e linhas do nosso conjunto de dados usando select() e

| filter(), respectivamente. Saiba que select() também era a capacidade de ordenar nossas colunas selecionadas

| em qualquer ordem que desejássemos.

...

|========================================================================================================================================================== | 71%

| Às vezes, queremos ordenar as linhas de um dataset de acordo com os valores de uma variável específica. Este

| é o trabalho de arrange().

...

|============================================================================================================================================================== | 73%

| Para ver como o arrange() funciona, vamos primeiro pegar um subconjunto do cran. selecione (com select)

| todas as colunas de size a ip\_id e armazene o resultado em cran2.

> cran2 <- select(cran, size:ip\_id)

| Está correto!

|================================================================================================================================================================= | 74%

| Agora, para ordenar as linhas de cran2 para que ip\_id esteja em ordem crescente (de pequeno a grande),

| digite arrange(cran2, ip\_id). Você pode querer maximizar seu console para ver ip\_id, que é a última coluna.

> arrange(cran2, ip\_id)

# A tibble: 225,468 × 8

size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 80589 3.1.0 x86\_64 mingw32 htmltools 0.2.4 US 1

2 180562 3.0.2 x86\_64 mingw32 yaml 2.1.13 US 1

3 190120 3.1.0 i386 mingw32 babel 0.2-6 US 1

4 321767 3.1.0 x86\_64 mingw32 tseries 0.10-32 US 2

5 52281 3.0.3 x86\_64 darwin10.8.0 quadprog 1.5-5 US 2

6 876702 3.1.0 x86\_64 linux-gnu zoo 1.7-11 US 2

7 321764 3.0.2 x86\_64 linux-gnu tseries 0.10-32 US 2

8 876702 3.1.0 x86\_64 linux-gnu zoo 1.7-11 US 2

9 321768 3.1.0 x86\_64 mingw32 tseries 0.10-32 US 2

10 784093 3.1.0 x86\_64 linux-gnu strucchange 1.5-0 US 2

# … with 225,458 more rows

| Você é muito bom, amig@!

|==================================================================================================================================================================== | 76%

| Para fazer o mesmo, mas em ordem decrescente, mude o segundo argumento para desc(ip\_id), onde desc()

| significa 'descendente'. Vai lá.

> arrange(cran2, desc(ip\_id))

# A tibble: 225,468 × 8

size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 5933 NA NA NA CPE 1.4.2 CN 13859

2 569241 3.1.0 x86\_64 mingw32 multcompView 0.1-5 US 13858

3 228444 3.1.0 x86\_64 mingw32 tourr 0.5.3 NZ 13857

4 308962 3.1.0 x86\_64 darwin13.1.0 ctv 0.7-9 CN 13856

5 950964 3.0.3 i386 mingw32 knitr 1.6 CA 13855

6 80185 3.0.3 i386 mingw32 htmltools 0.2.4 CA 13855

7 1431750 3.0.3 i386 mingw32 shiny 0.10.0 CA 13855

8 2189695 3.1.0 x86\_64 mingw32 RMySQL 0.9-3 US 13854

9 4818024 3.1.0 i386 mingw32 igraph 0.7.1 US 13853

10 197495 3.1.0 x86\_64 mingw32 coda 0.16-1 US 13852

# … with 225,458 more rows

| Ótimo trabalho!

|======================================================================================================================================================================== | 77%

| Nós também podemos organizar os dados de acordo com os valores de múltiplas variáveis. Por exemplo,

| arrange(cran2, package, ip\_id) primeiro organiza por nome dos pacotes (crescente alfabeticamente), depois

| por ip\_id. Isso significa que, se houver várias linhas com o mesmo valor para o pacote, elas serão

| classificadas por ip\_id (numericamente ascendente). Digite arrange(cran2, package, ip\_id) agora.

> arange(cran2, package, ip\_id)

Error in arange(cran2, package, ip\_id) : could not find function "arange"

> arrange(cran2, package, ip\_id)

# A tibble: 225,468 × 8

size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 71677 3.0.3 x86\_64 darwin10.8.0 A3 0.9.2 CN 1003

2 71672 3.1.0 x86\_64 linux-gnu A3 0.9.2 US 1015

3 71677 3.1.0 x86\_64 mingw32 A3 0.9.2 IN 1054

4 70438 3.0.1 x86\_64 darwin10.8.0 A3 0.9.2 CN 1513

5 71677 NA NA NA A3 0.9.2 BR 1526

6 71892 3.0.2 x86\_64 linux-gnu A3 0.9.2 IN 1542

7 71677 3.1.0 x86\_64 linux-gnu A3 0.9.2 ZA 2925

8 71672 3.1.0 x86\_64 mingw32 A3 0.9.2 IL 3889

9 71677 3.0.3 x86\_64 mingw32 A3 0.9.2 DE 3917

10 71672 3.1.0 x86\_64 mingw32 A3 0.9.2 US 4219

# … with 225,458 more rows

| Você acertou!

|============================================================================================================================================================================ | 79%

| Ordene o dataset cran2 pelas três variáveis a seguir, nesta ordem: country (crescente), r\_version

| (decrescente) e ip\_id (ascendente).

> arrange(cran2, country, desc(r\_version), ip\_id)

# A tibble: 225,468 × 8

size r\_version r\_arch r\_os package version country ip\_id

<int> <chr> <chr> <chr> <chr> <chr> <chr> <int>

1 1556858 3.1.1 i386 mingw32 RcppArmadillo 0.4.320.0 A1 2843

2 1823512 3.1.0 x86\_64 linux-gnu mgcv 1.8-1 A1 2843

3 15732 3.1.0 i686 linux-gnu grnn 0.1.0 A1 3146

4 3014840 3.1.0 x86\_64 mingw32 Rcpp 0.11.2 A1 3146

5 660087 3.1.0 i386 mingw32 xts 0.9-7 A1 3146

6 522261 3.1.0 i386 mingw32 FNN 1.1 A1 3146

7 522263 3.1.0 i386 mingw32 FNN 1.1 A1 3146

8 1676627 3.1.0 x86\_64 linux-gnu rgeos 0.3-5 A1 3146

9 2118530 3.1.0 x86\_64 linux-gnu spacetime 1.1-0 A1 3146

10 2217180 3.1.0 x86\_64 mingw32 gstat 1.0-19 A1 3146

# … with 225,458 more rows

| Na mosca! Bom trabalho!

|=============================================================================================================================================================================== | 81%

| Para ilustrar a próxima função das principais do dplyr, vamos pegar outro subconjunto de nossos dados. Use

| select() para pegar 3 colunas de cran. Eu quero as colunas ip\_id, package e size (nessa ordem) - e armazene

| o resultado em uma nova variável chamada cran3.

> cran3 <- select(cran, ip\_id, package, size)

| Ótimo!

|================================================================================================================================================================================== | 82%

| Dê uma olhada no cran3 agora.

> cran3

# A tibble: 225,468 × 3

ip\_id package size

<int> <chr> <int>

1 1 htmltools 80589

2 2 tseries 321767

3 3 party 748063

4 3 Hmisc 606104

5 4 digest 79825

6 3 randomForest 77681

7 3 plyr 393754

8 5 whisker 28216

9 6 Rcpp 5928

10 7 hflights 2206029

# … with 225,458 more rows

| Você está indo muito bem!

|====================================================================================================================================================================================== | 84%

| É comum criar uma nova variável com base no valor de uma ou mais variáveis de um dataset. A função mutate()

| faz essa mágica.

...

|========================================================================================================================================================================================== | 85%

| A variável size representa o tamanho do download em bytes, que são unidades de memória do computador.

| Atualmente, megabytes (MB) são a unidade de medida mais comum. Um megabyte é igual a 2 ^ 20 bytes. Isso é 2

| elevado a 20, que é de aproximadamente um milhão de bytes!

...

|============================================================================================================================================================================================= | 87%

| Queremos adicionar uma coluna chamada size\_mb que contém o tamanho do download em megabytes. Aqui está o

| código para fazer isso: mutate(cran3, size\_mb = size / 2 ^ 20)

> mutate(cran3, size\_mb = size/2^20)

# A tibble: 225,468 × 4

ip\_id package size size\_mb

<int> <chr> <int> <dbl>

1 1 htmltools 80589 0.0769

2 2 tseries 321767 0.307

3 3 party 748063 0.713

4 3 Hmisc 606104 0.578

5 4 digest 79825 0.0761

6 3 randomForest 77681 0.0741

7 3 plyr 393754 0.376

8 5 whisker 28216 0.0269

9 6 Rcpp 5928 0.00565

10 7 hflights 2206029 2.10

# … with 225,458 more rows

| Você acertou!

|================================================================================================================================================================================================ | 89%

| Uma unidade de medida ainda maior é um gigabyte (GB), que é igual a 2 ^ 10 megabytes. Podemos também

| adicionar outra coluna para o tamanho do download em gigabytes!

...

|==================================================================================================================================================================================================== | 90%

| Um recurso muito interessante de mutate () é que você pode usar o valor calculado para sua segunda coluna

| (size\_mb) para criar uma terceira coluna, tudo na mesma linha de código. Para ver isso em ação, repita

| exatamente o mesmo comando acima, exceto adicionar um terceiro argumento criando uma coluna chamada size\_gb

| que é igual a size\_mb / 2 ^ 10.

> mutate(cran3, size\_mb = size/2^20, size\_gb = size\_mb/2^10)

# A tibble: 225,468 × 5

ip\_id package size size\_mb size\_gb

<int> <chr> <int> <dbl> <dbl>

1 1 htmltools 80589 0.0769 0.0000751

2 2 tseries 321767 0.307 0.000300

3 3 party 748063 0.713 0.000697

4 3 Hmisc 606104 0.578 0.000564

5 4 digest 79825 0.0761 0.0000743

6 3 randomForest 77681 0.0741 0.0000723

7 3 plyr 393754 0.376 0.000367

8 5 whisker 28216 0.0269 0.0000263

9 6 Rcpp 5928 0.00565 0.00000552

10 7 hflights 2206029 2.10 0.00205

# … with 225,458 more rows

| Continue assim e você chegará lá!

|======================================================================================================================================================================================================== | 92%

| Vamos fazer mais um para praticar. Finja que descobrimos uma falha no sistema que forneceu os valores

| originais para a variável de tamanho. Todos os valores no cran3 são 1000 bytes menores do que deveriam ser.

| Usando o cran3, crie apenas uma nova coluna chamada correct\_size que contenha o tamanho correto.

> mutate(cran3, correct\_size = size-1000)

# A tibble: 225,468 × 4

ip\_id package size correct\_size

<int> <chr> <int> <dbl>

1 1 htmltools 80589 79589

2 2 tseries 321767 320767

3 3 party 748063 747063

4 3 Hmisc 606104 605104

5 4 digest 79825 78825

6 3 randomForest 77681 76681

7 3 plyr 393754 392754

8 5 whisker 28216 27216

9 6 Rcpp 5928 4928

10 7 hflights 2206029 2205029

# … with 225,458 more rows

| Você quase acertou. Tente novamente. Ou digite info() para mais opções.

| mutate(cran3, correct\_size = size + 1000) criará uma nova coluna que é igual aos tamanhos originais mais

| 1000.

> mutate(cran3, correct\_size = size+1000)

# A tibble: 225,468 × 4

ip\_id package size correct\_size

<int> <chr> <int> <dbl>

1 1 htmltools 80589 81589

2 2 tseries 321767 322767

3 3 party 748063 749063

4 3 Hmisc 606104 607104

5 4 digest 79825 80825

6 3 randomForest 77681 78681

7 3 plyr 393754 394754

8 5 whisker 28216 29216

9 6 Rcpp 5928 6928

10 7 hflights 2206029 2207029

# … with 225,458 more rows

| Todo o trabalho duro está dando resultado!

|=========================================================================================================================================================================================================== | 94%

| O último dos cinco comandos principais de dplyr, é o summarize(). Ele agrupa o conjunto de dados em uma

| única linha. Digamos que estamos interessados em saber o tamanho médio de download. summarize(cran,

| avg\_bytes = mean(size)) irá gerar o valor médio da variável size. Aqui nós escolhemos rotular o resultado

| 'avg\_bytes', mas poderíamos ter nomeado com qualquer outro nome. Vai lá, tenta isso.

> summarize(cran, avg\_bytes = mean(size))

# A tibble: 1 × 1

avg\_bytes

<dbl>

1 844086.

| Todo o trabalho duro está dando resultado!

|============================================================================================================================================================================================================== | 95%

| Isso não é particularmente interessante. summary() é mais útil ao trabalhar com dados que foram agrupados

| pelos valores de uma variável específica.

...

|================================================================================================================================================================================================================== | 97%

| A ideia é que summarize() pode fornecer o valor solicitado para CADA grupo em seu conjunto de dados. Mas

| isso deixamos para ver em sala.

...

|====================================================================================================================================================================================================================== | 98%

| Nesta lição, você aprendeu como manipular dados usando as cinco principais funções do dplyr. Ainda existe

| muitos outros recursos úteis do dplyr para facilitar muito a sua vida como analista de dados.

...

|=========================================================================================================================================================================================================================| 100%

| Gostaria de informar ao professor sobre a conclusão desta lição

1: Sim

2: Não

Selection: 1

| Qual o código da sua turma? (exemplo FIAP-01IA)

24IA

| Qual seu código de aluno?

344154

| Qual seu nome?

Diego Cohen

| O que achou deste exercício?

OK

[1] "Tentando submeter ao professor, tentativa 1 ... (max 5) ..."

[1] "saved"

#############################################################################################################

Seu resultado foi salvo!

#############################################################################################################