Bem vindo ao swirl! Por favor, faça o login. Se você já esteve aqui antes, use o mesmo nome que você usou anteriormente. Se você é novo aqui, identifique-se com um nome original.

Como devo chamá-l@? Diego

| Obrigado, Diego. Permita-me explicar algumas coisinhas antes de começarmos nossa primeira lição. Antes de tudo, você precisa saber que quando você vir '...', isso significa que você deve pressionar Enter quando você

| terminar de ler e estiver pronto para continuar.

... <-- Esse é o sinal de que você deve pressionar Enter para continuar

| Da mesma forma, quando você vir 'ANSWER:', o prompt do R (>), ou quando lhe for pedido para selecionar itens de uma lista, isso significa que é a sua vez de dar uma resposta, e depois pressionar Enter para continuar.

Selecione 1, 2 ou 3 e pressione Enter

1: Continuar.

2: Avançar.

3: Vamos adiante!

Selection: 1

| Você pode sair do swirl e retornar ao prompt do R (>) a qualquer momento pressionando a tecla Esc. Se você já estiver no prompt, digite bye() para sair e salvar o seu progresso. Quando você sair corretamente, você verá uma

| breve mensagem lhe informando isso.

| Quando você estiver no prompt do R (>):

| -- Digitar skip() permite que você pule a questão atual.

| -- Digitar play() permite que você experimente com o R por conta própria; o swirl vai ignorar o que você fizer...

| -- ATÉ que você digite nxt() para ganhar a atenção do swirl novamente.

| -- Ao digitar bye() o swirl será fechado. O seu progresso será salvo.

| -- Ao digitar main(), você retornará ao menu principal do swirl.

| -- Ao digitar info(), essas informaçães serão apresentadas novamente.

| Vamos começar!

...

| Por favor, selecione um curso ou digite 0 para sair do swirl.

1: Aprenda R no R

2: Leve-me para o repositório de cursos do swirl!

Selection: 1

| Por favor, selecione uma lição ou digite 0 para retornar ao menu do curso.

1: Comandos basicos do R 2: Logicos 3: Sequencias numericas 4: Vetores 5: Valores ausentes 6: Data e Hora 7: Filtrando vetores 8: Matrizes e tabelas

9: Amostras e Simulacoes 10: Funcoes 11: Manipulacao com dplyr 12: Graficos Basicos 13: K Medias 14: Graficos Analiticos

Selection: 8

| Tentando carregar as dependências da lição...

| O pacote ‘openssl’ carregou corretamente!

| O pacote ‘jsonlite’ carregou corretamente!

| | 0%

| Nesta lição, abordaremos matrizes e tabelas. No R, o principal tipo de dados para tabelas é o 'data.frame'. Ambos representam tipos de dados 'retangulares', o que significa que eles são usados para armazenar dados

| tabulares, com linhas e colunas.

...

|====== | 3%

| A principal diferença, como você verá, é que as matrizes podem conter apenas um único tipo de dados, enquanto as tabelas podem armazenar muitos tipos diferentes.

...

|=========== | 5%

| Por isso as matrizes são consideradas estruturas de dados homegêneas e as tabelas, heterogêneas.

...

|================= | 8%

| Vamos criar um vetor contendo os números de 1 a 20 usando o operador `:`. Armazene o resultado em uma variável chamada my\_vector.

> my\_vector <- 1:20

| Continue assim e você chegará lá!

|======================= | 11%

| Veja o conteúdo do vetor que você acabou de criar.

> my\_vector

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

| Perseverança é a resposta.

|============================ | 13%

| A função dim() nos diz as 'dimensões' de um objeto. O que acontece se fizermos dim(my\_vector)? Tenta aí'.

> dim(my\_vector)

NULL

| Todo o trabalho duro está dando resultado!

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

| Me parece que não deu muito certo ... ou melhor: deu, mas não serve para muita coisa ... Como my\_vector é um vetor, ele não tem um atributo `dim` (por isso é apenas NULL), mas podemos encontrar seu tamanho usando a função

| length(). Tente isso agora.

> length(my\_vector)

[1] 20

| Mantenha esse bom nível!

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

| Ah! Isso é o que nós queríamos. Mas, o que acontece se dermos ao my\_vector um atributo `dim`? Vamos tentar! Digite dim(my\_vector) <- c(4, 5).

> dim(my\_vector) <- c(4,5)

| Excelente!

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

| Tudo bem se esse último comando pareceu um pouco estranho para você, eu sei. E deveria!

...

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

| A função dim() permite que você obtenha OU defina o atributo `dim` para um objeto R. Neste caso, atribuímos o valor c(4, 5) ao atributo `dim` de my\_vector.

...

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

| Digite dim(my\_vector) para confirmar que definimos corretamente o atributo `dim`.

> dim(my\_vector)

[1] 4 5

| Excelente trabalho!

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

| Outra maneira de ver isso é chamando a função attributes() em my\_vector. Tente agora.

> attributes(my\_vector)

$dim

[1] 4 5

| Mantenha esse bom nível!

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

| Você se lembra das aulas de matemática? Assim como nas aulas de matemática, ao lidar com um objeto bidimensional (pense em uma matriz), o primeiro número é o número de linhas e o segundo é o número de colunas. Portanto,

| apenas demos ao my\_vector a informação que ele possui 4 linhas e 5 colunas.

...

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

| Mas espere! Isso não parece mais como um vetor. Bem, e não é! Agora é uma matriz. Veja o conteúdo de my\_vector agora com seus próprios olhos.

> my\_vector

[,1] [,2] [,3] [,4] [,5]

[1,] 1 5 9 13 17

[2,] 2 6 10 14 18

[3,] 3 7 11 15 19

[4,] 4 8 12 16 20

| Excelente trabalho!

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

| Agora, vamos confirmar que é realmente uma matriz usando a função class(). Digite class(my\_vector) para ver se é verdade.

> class(my\_vector)

[1] "matrix" "array"

| Você está indo muito bem!

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

| Poxa! 'É isso mesmo! Agora my\_vector é uma matriz. Devemos armazená-lo em uma variável mais adequada que nos ajude a lembrar o que é. Armazene o valor de my\_vector em uma nova variável chamada my\_matrix.

> my\_matrix <- my\_vector

| Todo o trabalho duro está dando resultado!

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

| O exemplo que usamos até agora foi concebido para ilustrar o fato de que uma matriz é simplesmente um vetor atômico com um atributo de dimensão. Um método mais direto de criar a mesma matriz usa a função matrix().

...

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

| Chame o arquivo de ajuda para a função matrix() agora usando o `?`.

> ?matrix()

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

| O comando ?matrix deve fazer o que espero.

> ?matrix

| Está correto!

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

| Agora, olhe a documentação para a função matrix e veja se você descobre como criar uma matriz contendo os mesmos números (1-20) e dimensões (4 linhas, 5 colunas) chamando a função matrix(). Armazene o resultado em uma

| variável chamada my\_matrix2.

> my\_matrix2 <- matrix(data=c(0:20),nrow=1, ncol=5)

Warning message:

In matrix(data = c(0:20), nrow = 1, ncol = 5) :

data length [21] is not a sub-multiple or multiple of the number of columns [5]

| Boa tentativa, mas não é exatamente o que estou esperando. Tente novamente. Ou digite info() para mais opções.

| Chame a função matrix() com três argumentos - 1:20, o número de linhas e o número de colunas. Certifique-se de especificar argumentos por seus nomes e armazene o resultado em my\_matrix2 (não em my\_matrix).

> my\_matrix2 <- matrix(data=c(1:20),nrow=1, ncol=5)

Warning message:

In matrix(data = c(1:20), nrow = 1, ncol = 5) :

data length differs from size of matrix: [20 != 1 x 5]

| Isso não é exatamente o que espero. Tente novamente. Ou digite info() para mais opções.

| Chame a função matrix() com três argumentos - 1:20, o número de linhas e o número de colunas. Certifique-se de especificar argumentos por seus nomes e armazene o resultado em my\_matrix2 (não em my\_matrix).

> my\_matrix2 <- matrix(c(1:20),nrow = 4, ncol=5)

| Na mosca! Bom trabalho!

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

| Finalmente, vamos confirmar que my\_matrix e my\_matrix2 são realmente idênticas. A função identical() nos dirá se os dois primeiros argumentos são iguais. Experimente.

> identical(my\_matrix, my\_matrix2)

[1] TRUE

| Você acertou!

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

| Agora, imagine que os números em nossa tabela representam algumas medidas de um experimento clínico, em que cada linha representa um paciente e cada coluna representa uma variável para a qual as medidas foram tomadas.

...

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

| Podemos marcar as linhas para sabermos quais números pertencem a cada paciente no experimento. Uma maneira de fazer isso é adicionar uma coluna à matriz, que contém os nomes de todas as quatro pessoas.

...

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

| Vamos começar criando um novo vetor de caracteres contendo os nomes de nossos pacientes - Bill, Gina, Kelly e Sean. Lembre-se que as aspas duplas dizem ao R que algo é uma cadeia de caracteres. Armazene o resultado em uma

| variável chamada patients.

> patients <- c('Bill', 'Gina', 'Kelly', 'Sean')

| Você é muito bom, amig@!

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

| Agora vamos usar a função cbind() para "combinar colunas". Não se preocupe em armazenar o resultado em uma nova variável, veremos o resultado no console. Apenas chame cbind() com dois argumentos - o vetor patients e a matriz

| my\_matrix.

> cbind(patients, my\_matrix)

patients

[1,] "Bill" "1" "5" "9" "13" "17"

[2,] "Gina" "2" "6" "10" "14" "18"

[3,] "Kelly" "3" "7" "11" "15" "19"

[4,] "Sean" "4" "8" "12" "16" "20"

| Você é muito bom, amig@!

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

| Algo está estranho no nosso resultado! Hummm ... Parece que combinar o vetor de caracteres com a nossa matriz de números fez com que tudo fosse colocado entre aspas duplas. Isso significa que ficamos com uma matriz de

| caracteres, o que não é bom.

...

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

| Se você se lembrar do início desta lição, eu disse que as matrizes só podem conter um tipo de dados. Portanto, quando tentamos combinar um vetor de caracteres com uma matriz numérica, o R foi forçado a "converter" os números

| para caracteres, daí as aspas duplas.

...

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

| Isso é chamado de "conversão implícita", porque não pedimos por isso. Apenas aconteceu. Mas por que não converteu os nomes de nossos pacientes em números? Vou deixar você refletir sobre essa questão por conta própria.

...

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

| Então, ainda nos resta a questão de como incluir os nomes de nossos pacientes na tabela sem destruir a integridade de nossos dados numéricos. Digite o seguinte comando - my\_data <- data.frame(patients, my\_matrix)

> my\_data <- data.frame(patients, my\_matrix)

| Todo o trabalho duro está dando resultado!

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

| Agora, veja o conteúdo de my\_data para ver o que criamos.

> my\_data

patients X1 X2 X3 X4 X5

1 Bill 1 5 9 13 17

2 Gina 2 6 10 14 18

3 Kelly 3 7 11 15 19

4 Sean 4 8 12 16 20

| Está correto!

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

| Parece que a função data.frame() nos permitiu armazenar nosso vetor de caracteres de nomes ao lado de nossa matriz de números. Isso é exatamente o que estávamos esperando!

...

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

| Nos bastidores, a função data.frame() pega qualquer número de argumentos e retorna um único objeto tabela de classe `data.frame` que é composto dos objetos originais.

...

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

| Vamos confirmar isso chamando a função class() em nossa tabela recém-criada.

> class(my\_data)

[1] "data.frame"

| Está correto!

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

| Também é possível atribuir nomes às linhas e colunas de uma tabela, o que seria outra maneira de determinar qual linha de valores em nossa tabela pertence a cada paciente.

...

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

| No entanto, como já resolvemos esse problema, vamos fazer algo diferente, atribuiremos nomes às colunas da nossa tabela, para que saibamos que tipo de medida cada coluna representa.

...

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

| Como temos seis colunas (incluindo nomes de pacientes), primeiro precisamos criar um vetor contendo um elemento para cada coluna. Crie um vetor de caracteres chamado cnames que contém os seguintes valores (em ordem) -

| "paciente", "idade", "peso", "pressao", "batimentos", "teste".

> cnames('paciente', 'cidade', 'peso', 'pressao', 'batimentos', 'teste')

Error in cnames("paciente", "cidade", "peso", "pressao", "batimentos", :

could not find function "cnames"

> cnames <- c('paciente', 'cidade', 'peso', 'pressao', 'batimentos', 'teste')

| Quase! Tente novamente. Ou digite info() para mais opções.

| Certifique-se de que todos os nomes estejam em minúsculas, entre aspas duplas e separados por vírgulas. Não esqueça de usar a função c().

> cnames <- c("paciente", "idade", "peso", "pressao", "batimentos", "teste")

| Continue assim e você chegará lá!

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

| Agora, use a função colnames() para definir o atributo `colnames` para a nossa tabela. Vai ficar semelhante ao modo como usamos a função dim() anteriormente nesta lição.

> colnames(my\_data) <- cnames

| Você acertou!

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

| Vamos ver se isso deu certo. Imprima o conteúdo de my\_data.

> my\_data

paciente idade peso pressao batimentos teste

1 Bill 1 5 9 13 17

2 Gina 2 6 10 14 18

3 Kelly 3 7 11 15 19

4 Sean 4 8 12 16 20

| Ótimo!

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

| Nesta lição, você aprendeu o básico para trabalhar com duas estruturas de dados muito importantes e comuns - matrizes e tabelas (data.frame). Há muito mais a aprender que serão vistos em tópicos mais avançados,

| principalmente em relação a tabelas.

...

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

| Gostaria de informar ao professor sobre a conclusão desta lição

1: Não

2: Sim

Selection: 2

| 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?

Bom!

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

[1] "saved"

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

Seu resultado foi salvo!

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