| 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: 3

| Tentando carregar as dependências da lição...

| O pacote ‘openssl’ carregou corretamente!

| O pacote ‘jsonlite’ carregou corretamente!

| | 0%

| Oi Alunos!!! Nesta lição, você aprenderá como criar sequências de números em R.

...

|========= | 4%

| A maneira mais simples de criar uma sequência de números em R é usando o operador `:`. Digite 1:20 para ver como funciona.

> 1:20

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

| Excelente!

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

| Isso nos deu cada número inteiro entre (e incluindo) 1 e 20. Também poderíamos usá-lo para criar uma sequência de números reais. Por exemplo, digite pi:10.

> pi:10

[1] 3.141593 4.141593 5.141593 6.141593 7.141593 8.141593 9.141593

| Na mosca! Bom trabalho!

|=========================== | 12%

| O resultado é um vetor de números reais começando com pi (3.142...) e aumentando em incrementos de 1. O limite superior de 10 nunca é atingido, uma vez que o próximo número em nossa sequência seria maior que 10.

...

|==================================== | 17%

| O que acontece se fizermos 15:1? Tente para descobrir.

> 15:1

[1] 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

| Sua dedicação é inspiradora!

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

| Decresceu de 15 para 1 em decrementos de 1! É improvável que desejemos esse comportamento na maioria dos programas, mas é bom saber disso para usar em um programa ou outro.

...

|====================================================== | 25%

| Lembre-se que se você tiver dúvidas sobre uma função R em particular, você pode acessar sua documentação com um ponto de interrogação seguido pelo nome da função: ?function\_name\_here. No entanto, no caso de um operador como

| o dois pontos usado acima, você deve colocar o símbolo em crases como neste exemplo: ?`:`. (Se você não tiver uma tecla de crase, você pode usar aspas normais neste exercício.)

...

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

| Veja a documentação para `:` agora.

> ?':'

| Bom trabalho!

|======================================================================== | 33%

| Frequentemente, desejamos mais controle sobre uma sequência que estamos criando do que o operador `:` nos oferece. A função seq() serve para esse propósito.

...

|================================================================================= | 38%

| O uso mais básico de seq() faz exatamente a mesma coisa que o operador `:`. Tente seq(1, 20) para ver o que

| acontece.

> seq(1,20)

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

| Ótimo!

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

| Isso nos dá a mesma saída que 1:20. No entanto, digamos que, em vez disso, queremos um vetor de números que

| varia de 0 a 10, incrementado em 0,5. O comando seq(0, 10, by=0.5) faz exatamente isso. Experimente.

> seq(0,10,by=0.5)

[1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0 9.5 10.0

| Você está em um bom ritmo!

|=================================================================================================== | 46%

| Pode ser que não nos importemos com o incremento e queremos apenas uma sequência de 30 números entre 5 e 10.

| seq(5, 10, length=30) faz o truque, UAU! Tente agora e armazene o resultado em uma nova variável chamada

| my\_seq.

> my\_seq <- seq(5,10,length=30)

| Sua dedicação é inspiradora!

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

| Para confirmar que my\_seq tem comprimento 30, podemos usar a função length(). Experimente usar esta função

| na variável que você acabou de criar.

> my\_seq

[1] 5.000000 5.172414 5.344828 5.517241 5.689655 5.862069 6.034483 6.206897 6.379310 6.551724

[11] 6.724138 6.896552 7.068966 7.241379 7.413793 7.586207 7.758621 7.931034 8.103448 8.275862

[21] 8.448276 8.620690 8.793103 8.965517 9.137931 9.310345 9.482759 9.655172 9.827586 10.000000

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

| Use length(my\_seq) para ver o tamanho da variável my\_seq.

> length(my\_sec)

Error: object 'my\_sec' not found

> length(my\_seq)

[1] 30

| Toda a prática está rendendo frutos!

|===================================================================================================================== | 54%

| Vamos fingir que não sabemos o comprimento de my\_seq, mas queremos gerar uma seqüência de inteiros de 1 a N,

| onde N representa o comprimento do vetor my\_seq. Em outras palavras, queremos um novo vetor (1, 2, 3, ...)

| com o mesmo comprimento que my\_seq.

...

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

| Existem várias maneiras de fazer isso. Uma possibilidade é combinar o operador `:` e a função length()

| assim: 1:length(my\_seq). Tente assim e veja se funciona.

> 1:length(my\_seq)

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

| Excelente trabalho!

|======================================================================================================================================= | 62%

| Outra opção é usar seq(along.with = my\_seq). Opa, gostei disso! Agora faça você e veja com seus próprios

| olhos.

> seq(along.with=my\_seq)

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

| Todo o trabalho duro está dando resultado!

|================================================================================================================================================ | 67%

| No entanto, como é o caso de muitas tarefas comuns, o R tem uma função interna separada para esse propósito

| chamada seq\_along(). Digite seq\_along(my\_seq) para vê-lo em ação.

> seq\_along(my\_sec)

Error: object 'my\_sec' not found

> seq\_along(my\_seq)

[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

| Bom trabalho!

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

| Muitas vezes existem várias abordagens para resolver o mesmo problema, ainda mais em R. Simples abordagens

| que envolvem menos digitação são geralmente melhores. Também é importante que seu código seja legível, para

| que você e outras pessoas possam descobrir o que está acontecendo sem muito trabalho.

...

|================================================================================================================================================================== | 75%

| Se o R tiver uma função interna para uma tarefa específica, é provável que essa função seja altamente

| otimizada para essa finalidade e seja sua melhor opção. À medida que você se tornar um programador R mais

| avançado, criará suas próprias funções para executar tarefas quando não houver opções melhores. Vamos

| explorar escrevendo suas próprias funções em lições futuras.

...

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

| Mais uma função relacionada à criação de seqüências de números é rep(), que significa `replicar`. Vamos dar

| uma olhada em alguns exemplos e usos.

...

|==================================================================================================================================================================================== | 83%

| Se estivermos interessados em criar um vetor que contenha 40 zeros, podemos usar rep(0, times = 40). Vai lá,

| digita este comando.

> rep(0, times=40)

[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

| Continue assim e você chegará lá!

|============================================================================================================================================================================================= | 88%

| Se ao invés de zeros, nós quisermos que a saída seja 10 repetições do vetor (0, 1, 2), podemos fazer

| rep(c(0, 1, 2), times = 10). Vai fundo e veja o resultado.

> rep(c(1,2), times=10)

[1] 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2

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

| Tente rep(c(0, 1, 2), times = 10) para ver uma variação do mesmo comando. Certifique-se de usar a função c()

| para informar R que os números 0, 1 e 2 formam um vetor.

> rep(c(0,1,2), times=10)

[1] 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2 0 1 2

| Perseverança é a resposta.

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

| Finalmente, digamos que, em vez de repetir o vetor (0, 1, 2) várias vezes, queremos que nosso vetor contenha

| 10 zeros, depois 10 uns, depois 10 dois. Nós podemos fazer isso com o argumento `each`. Tente rep(c(0, 1,

| 2), each = 10).

> rep(c(0,1,2), each = 10)

[1] 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2

| Na mosca! Bom trabalho!

|=============================================================================================================================================================================================================== | 96%

| Por enquanto é isso. Vimos bastante sobre como criar sequencias numéricas hoje.

...

|========================================================================================================================================================================================================================| 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?

Ótimo!

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

[1] "saved"

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

Seu resultado foi salvo!

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