Olá meus queridos alunos. Que tal vermos um pouco sobre vetores?

...

|=== | 3%

| A estrutura de dados mais simples e mais comum em R é o vetor, o termo correto é vetor atômico - por ser

| originalmente indivisível. Observação: no R temos a impressão de que podemos dividir vetores atômicos, o que

| acontece na verdade é que um novo vetor é criado quando necessário ;-).

...

|===== | 5%

| Um vetor contém apenas um tipo de dados, ou seja, é um estrutura de dados homogênea. Uma lista pode conter

| vários tipos de dados, por isso é considerada uma estrutura de dados heterogênea. Exploraremos nesta lição

| os vetores antes de chegarmos às listas.

...

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

| Se você fez a lição de sequencias numéricas, viu que já lidamos com vetores numéricos, que são um tipo de

| vetor atômico. Outros tipos de vetores atômicos incluem lógico, caractere, inteiro e complexo (ou

| imaginário). Nesta lição, examinaremos mais de perto os vetores lógicos e de caracteres.

...

|========== | 10%

| Os vetores lógicos podem conter os valores TRUE, FALSE e NA (para 'não disponível'). Esses valores são

| gerados como resultado de 'condições' lógicas - ou testes lógicos. Vamos testar algumas condições simples.

...

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

| Primeiro, use a função c() e crie um vetor numérico chamado num\_vect que contenha os valores 0.5, 55, -10 e

| 6.

> num\_vec <- c(0.5,55,-10,6)

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

| Lembre-se de que a função c() é usada para criar um vetor. Se você esquecer como usá-lo, digite ?c para

| acessar o arquivo de ajuda. Não esqueça de atribuir o resultado a uma nova variável chamada num\_vect.

> num\_vect <- c(0.5,55,-10,6)

| Todo o trabalho duro está dando resultado!

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

| Agora, crie uma variável chamada tf que obtenha o resultado de num\_vect < 1, que é lido como 'num\_vect é

| menor que 1'.

> tf <- num\_vect<1

| Você está em um bom ritmo!

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

| Como você acha que a variável tf foi criada?

1: um vetor de 4 valores lógicos

2: um único valor lógico

Selection: 1

| Excelente trabalho!

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

| Imprima o conteúdo de tf agora.

> tf

[1] TRUE FALSE TRUE FALSE

| Ótimo trabalho!

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

| A instrução num\_vect < 1 é uma condição e tf nos informa se cada elemento correspondente de nosso numérico

| vetorial num\_vect satisfaz essa condição.

...

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

| O primeiro elemento de num\_vect é 0.5, pois é menor que 1 e, portanto, a instrução 0.5 < 1 é TRUE. O segundo

| elemento de num\_vect é 55, que é maior que 1, portanto, a instrução 55 < 1 é FALSE. A mesma lógica se aplica

| ao terceiro e quarto elementos.

...

|============================ | 28%

| Vamos tentar uma variação. Digite num\_vect >= 6 sem atribuir o resultado a uma nova variável para exibir o

| resultado no console.

> num\_vect >=6

[1] FALSE TRUE FALSE TRUE

| Você está indo muito bem!

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

| Desta vez, estamos perguntando se cada elemento individual de num\_vect é maior que OU igual a 6. Como

| somente 55 e 6 são maiores ou iguais a 6, o segundo e quarto elementos do resultado são TRUE e o primeiro e

| terceiro elementos são FALSE.

...

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

| Os símbolos `<` e `>=` nesses exemplos são chamados de 'operadores lógicos'. Outros operadores lógicos

| incluem `>`, `<=`, `==` para igualdade exata, e `!=` para desigualdade.

...

|==================================== | 36%

| Se tivermos duas expressões lógicas, A e B, podemos perguntar se pelo menos um é verdadeiro com A | B

| (lógico 'ou' também conhecido por 'união') ou se ambos são TRUE com A & B (lógica 'e' também conhecido por

| 'interseção'). Por fim, !A é a negação de A e é TRUE quando A é FALSE e vice-versa.

...

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

| É uma boa ideia passar algum tempo brincando com várias combinações desses operadores lógicos até ficar

| confortável com o uso deles. Vamos fazer alguns exemplos aqui para você começar.

...

|========================================= | 41%

| Tente o seu melhor para prever o resultado de cada uma das seguintes afirmações. Você pode usar lápis e

| papel, se for te ajudar. Se você ficar empacado, apenas adivinhe e você tem 50% de chance de obter a

| resposta certa!

...

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

| (3 > 5) & (4 == 4)

1: TRUE

2: FALSE

Selection: 2

| Excelente trabalho!

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

| (TRUE == TRUE) | (TRUE == FALSE)

1: TRUE

2: FALSE

Selection: 1

| Maravilha!

|================================================= | 49%

| ((111 >= 111) | !(TRUE)) & ((4 + 1) == 5)

1: TRUE

2: FALSE

Selection: 1

| Você acertou!

|==================================================== | 51%

| Não se preocupe se você achar que isso é complicado. Fis essa conta para ser mesmo. Trabalhar com

| declarações lógicas em R requer prática, mas seus esforços serão recompensados em lições futuras (por

| exemplo, estruturas de subconjuntos e controle).

...

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

| Os vetores de caracteres também são muito comuns em R. Aspas duplas ou simples são usadas para distinguir

| vetores de texto, como no exemplo a seguir.

...

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

| Crie um vetor de caractere que contenha as seguintes palavras: "My", "name", "is". Lembre-se de incluir cada

| palavra em seu próprio conjunto de aspas, de modo que R saiba que elas são cadeias de caracteres. Armazene o

| vetor em uma variável chamada my\_char.

> my\_char = c('My', 'name', 'is')

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

| Digite my\_char <- c("My", "name", "is") para criar uma nova variável chamada my\_char que contenha um vetor

| de caracteres de comprimento 3. Certifique-se de que as vírgulas que separam as palavras estejam FORA das

| aspas ou o R vai entender que as vígulas são parte das palavras.

> my\_char <- c('My', 'name', 'is')

| Todo o trabalho duro está dando resultado!

|============================================================ | 59%

| Imprima o conteúdo de my\_char para ver como que esta variável foi criada.

> my\_char

[1] "My" "name" "is"

| Você acertou!

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

| Show! my\_char é um vetor de caracteres de comprimento 3. Digamos que queremos unir os elementos de my\_char

| em uma cadeia de caracteres contínua (isto é, um vetor de caracteres de comprimento 1). Podemos fazer isso

| usando a função paste().

...

|================================================================= | 64%

| Digite paste(my\_char, collapse = " ") agora. Certifique-se de que existe um espaço entre as aspas duplas no

| argumento `collapse`. Você verá porque esse argumento em um segundo."

> paste(my\_char, collapse = " ")

[1] "My name is"

| Está correto!

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

| O argumento `collapse` para a função paste() diz ao R que quando unimos os elementos do vetor do caractere

| my\_char, gostaríamos de separá-los com espaços simples.

...

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

| Parece que estamos esquecendo alguma coisa .... Ah, sim! Seu nome!

...

|========================================================================= | 72%

| Para adicionar (ou `concatenar`) seu nome ao final de my\_char, use a função c() assim: c(my\_char,

| "your\_name\_here"). Coloque seu nome entre aspas duplas onde coloquei "your\_name\_here". Experimente agora,

| armazenando o resultado em uma nova variável chamada my\_name.

> my\_name <- paste(c(my\_char, 'Diego'), collapse = " ")

| Não é bem isso. Tente novamente. Ou digite info() para mais opções.

| Coloque seu nome no final do vetor my\_char usando a função c(). Certifique-se de atribuir o resultado a uma

| nova variável chamada my\_name. Se o seu nome fosse 'Swirl', você digitaria my\_name <- c(my\_char, "Swirl").

> my\_name <- c(my\_char, 'Diego')

| Você está indo muito bem!

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

| Dê uma olhada no conteúdo de my\_name.

> my\_name

[1] "My" "name" "is" "Diego"

| Maravilha!

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

| Agora, use a função paste() mais uma vez para unir as palavras em my\_name em uma única cadeia de caracteres.

| Não esqueça do argumento collapse = " "!

> paste(my\_name, collapse = ' ')

[1] "My name is Diego"

| Sua dedicação é inspiradora!

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

| Neste exemplo, usamos a função paste() para juntar os elementos de um único vetor de caractere. paste()

| também pode ser usado para unir os elementos de vários vetores de caracteres.

...

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

| No caso mais simples, podemos unir dois vetores de caracteres que são de comprimento 1 (isto é, juntar duas

| palavras). Digite paste("Hello", "world!", sep = " "), onde o argumento `sep` diz ao R que queremos separar

| os elementos unidos com um único espaço.

>

> paste('Hello', 'world', sep=' ')

[1] "Hello world"

| Tente novamente. Acertar de primeira não tem graça mesmo! Ou digite info() para mais opções.

| Digite paste("Hello", "world!", sep = " ") para unir as duas palavras "Hello" e "world", separadas por um

| único espaço. Deve haver um único espaço entre as aspas duplas no argumento `sep` para a função paste().

> paste("Hello", "world!", sep = " ")

[1] "Hello world!"

| Você acertou!

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

| Para um exemplo um pouco mais complicado, podemos unir dois vetores, cada um com o comprimento 3. Use

| paste() para unir o vetor inteiro 1:3 com o vetor de caracteres c("X", "Y", "Z"). Desta vez, use sep = ""

| para não deixar espaço entre os elementos unidos.

> paste(1:3,c('X','Y',Z), sep='')

Error in paste(1:3, c("X", "Y", Z), sep = "") : object 'Z' not found

> paste(1:3,c('X','Y','Z'), sep='')

[1] "1X" "2Y" "3Z"

| Continue assim e você chegará lá!

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

| O que você acha que acontecerá se os vetores tiverem comprimentos diferentes?

...

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

| Reciclagem de vetores! Digite paste(LETTERS, 1:4, sep = "-"), onde LETTERS é uma variável pré-definida em R

| contendo um vetor de caracteres de todas as 26 letras maiúsculas do alfabeto.

> paste(LETTERS, 1:4, sep='-')

[1] "A-1" "B-2" "C-3" "D-4" "E-1" "F-2" "G-3" "H-4" "I-1" "J-2" "K-3" "L-4" "M-1" "N-2" "O-3" "P-4" "Q-1"

[18] "R-2" "S-3" "T-4" "U-1" "V-2" "W-3" "X-4" "Y-1" "Z-2"

| Essa é a resposta que eu esperava.

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

| Como o vetor de caracteres LETTERS é maior que o vetor numérico 1:4, o R simplesmente recicla, ou repete,

| 1:4 até coincidir com o comprimento de LETTERS.

...

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

| Também vale a pena notar que o vetor numérico 1:4 é 'convertido' em um vetor de caracteres pela função

| paste().

...

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

| Discutiremos a conversão (ou coerção) em outra lição, mas tudo o que ela realmente quer dizer é que os

| números 1, 2, 3 e 4 na saída acima não são mais números para o R, mas sim caracteres '1', '2', '3' e '4'.

...

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

Ótimo!

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

[1] "saved"

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

Seu resultado foi salvo!

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

| Está correto!

| Você chegou ao final desta lição! Retornando ao menu principal...