| Olá alunos de R!

...

|====== | 3%

| Uma das grandes vantagens de usar uma linguagem de programação estatística como o R é a sua vasta coleção de ferramentas para gerar números aleatórios, também conhecida como simular valores.

...

|============ | 6%

| Esta lição pressupõe uma familiaridade com algumas distribuições de probabilidade comuns, mas esses tópicos só serão discutidos com relação à geração de números aleatórios. Mesmo que você não tenha experiência prévia com

| esses conceitos, você deve ser capaz de completar a lição e entender as principais ideias.

...

|=================== | 9%

| A primeira função que usaremos para gerar números aleatórios é sample(). Digite ?sample para ver a documentação.

> ?sample

| Essa é a resposta que eu esperava.

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

| Se você já leu, vamos ao que interessa.

...

|=============================== | 14%

| Vamos simular a rolagem de quatro dados de seis lados. Para isso digite: sample(1:6, 4, replace = TRUE).

> sample(1:6,4,replace=TRUE)

[1] 3 3 6 6

| Ótimo trabalho!

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

| Agora repita o comando para ver como seu resultado será diferente. (A probabilidade de dar exatamente o mesmo resultado é (1/6) ^ 4 = 0,00077, o que é bem pequeno!)

> sample(1:6,4,replace=TRUE)

[1] 5 4 6 2

| Bom trabalho!

|=========================================== | 20%

| sample(1:6, 4, replace = TRUE) diz ao R a selecionar aleatoriamente quatro números entre 1 e 6, com reposição. Amostragem com reposição significa simplesmente que cada número é "reposto" depois de ser selecionado, para que o

| mesmo número possa aparecer mais de uma vez. É isso que queremos aqui, já que o que dá em um dado não deve afetar o que dá em nenhum dos outros.

...

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

| Agora, faça uma amostragem de 10 números entre 1 e 20, SEM reposição. Para amostras sem reposição, simplesmente remova o argumento 'replace'.

> sample(1:20,10,replace=FALSE)

[1] 17 1 9 11 5 12 2 7 19 6

| Você está em um bom ritmo!

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

| Desta vez, perceba que nenhum número apareceu mais de uma vez na saída.

...

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

| LETTERS é uma variável pré-definida em R contendo um vetor de todas as 26 letras do alfabeto Inglês. Seja o conteúdo deste vetor.

> LETTERS

[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"

| Na mosca! Bom trabalho!

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

| A função sample() também pode ser usada para permutar ou reorganizar os elementos de um vetor. Por exemplo, digite sample(LETTERS) para permutar todas as 26 letras do alfabeto.

> sample(LETTERS)

[1] "O" "G" "T" "A" "R" "Z" "E" "N" "Q" "X" "K" "F" "V" "J" "I" "W" "C" "B" "P" "D" "M" "U" "S" "H" "Y" "L"

| Bom trabalho!

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

| Isto é a mesma coisa que tirar uma amostra de tamanho 26 de LETTERS, sem substituição. Quando o argumento 'size' para sample() não é especificado, o R gera uma amostra igual em tamanho ao vetor do qual você está amostrando.

...

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

| Agora, suponha que queremos simular 100 lançamentos de uma moeda viciada de dois lado. Esta moeda em particular tem uma probabilidade de 0,3 ser 'coroa' e uma probabilidade de 0,7 de ser 'cara'.

...

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

| Deixe o valor 0 representar as coroas e o valor 1 representar as caras. Use sample() para gerar uma amostra de tamanho 100 do vetor c(0,1), com substituição. Como a moeda é injusta, devemos especificar probabilidade aos

| valores 0 (coroas) e 1 (caras) com um quarto argumento, prob = c(0.3, 0.7). Atribua o resultado a uma nova variável chamada flips.

> flips <- sample(c(0,1),100,prob=c(0.3,0.7))

Error in sample.int(length(x), size, replace, prob) :

cannot take a sample larger than the population when 'replace = FALSE'

> flips <- sample(c(0,1),100,prob=c(0.3,0.7), replace=TRUE)

| Toda a prática está rendendo frutos!

|============================================================================================= | 43%

| Veja o conteúdo da variável flips.

> flips

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

| Excelente!

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

| Como definimos a probabilidade de que tirar uma cara é de 0.7, esperamos que aproximadamente 70 de nossas jogadas de moeda tenham o valor 1. Conte o número real de 1s contido nas inversões usando a função sum().

> sum(flips)

[1] 58

| Todo o trabalho duro está dando resultado!

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

| Uma jogada de moeda é um resultado binário (0 ou 1) e estamos realizando 100 testes independentes (coin flips). Podemos também usar outra função, a rbinom() para simular uma variável aleatória binomial. Veja a documentação

| para rbinom() digitando ?rbinom.

> ?rbinom

| Na mosca! Bom trabalho!

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

| Cada distribuição de probabilidade em R tem uma função r\*\*\* (para "sorteio aleatório"), função ad\*\*\* (para "densidade"), ap\*\*\* (para "probabilidade") e q\*\*\* (para "quantile"). Nós veremos algumas funções nesta lição, mas eu

| encorajo você a explorar as outras por conta própria.

...

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

| Uma variável aleatória binomial representa o número de 'sucessos' (caras) em um determinado número de

| 'tentativas' independentes (lançamentos de moeda). Portanto, podemos gerar uma única variável aleatória que

| representa o número de caras em 100 lançamentos de nossa moeda viciada usando rbinom(1, size = 100, prob =

| 0.7). Note que você só especifica a probabilidade de 'sucesso' (caras) e NÃO a probabilidade de 'falhas'

| (coroas). Digite este comando para ver.

> rbinom(1, size=100, prob=0.7)

[1] 69

| Você está indo muito bem!

|============================================================================================================================ | 57%

| De modo equivalente, se quisermos ver todos os 0s e 1s, usando esta mesma função podemos solicitar 100

| observações, cada uma de tamanho 1, com probabilidade de sucesso de 0.7. Experimente, atribuindo o resultado

| a uma nova variável chamada flips2. Parece um pouco confuso, mas se você errar eu te dou uma boa dica.

> flips2 <- rbinom(1, size=100, prob=0.7)

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

| opções.

| Digite rbinom() com n = 100, size = 1 e prob = 0.7 e atribua o resultado a flips2.

> flips2 <- rbinom(100, size=1, prob=0.7)

| Bom trabalho!

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

| Veja o conteúdo de flips2.

> flips2

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

[54] 1 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1

| Continue assim e você chegará lá!

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

| Agora use sum() para contar o número de 1s (caras) em flips2. Deve estar perto de 70 também!

> sum(flips2)

[1] 76

| Continue assim e você chegará lá!

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

| Semelhante a rbinom(), podemos usar o R para simular números aleatórios de muitas outras distribuições de

| probabilidade. Veja a documentação do rnorm() agora.

> ?rnorm

| Maravilha!

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

| A função de distribuição normal padrão tem média 0 e desvio padrão 1. Como você pode ver na documentação, os

| valores padrão para os argumentos 'mean' e 'sd' para rnorm() são 0 e 1, respectivamente. Assim, o rnorm(10)

| gerará 10 números aleatórios a partir de uma distribuição normal padrão. Tente aí.

> rnorm(10)

[1] -0.009171898 -1.007381767 0.472028516 0.982347783 -0.201369336 -1.178439443 -1.204797364 0.814138827 0.692494786 -0.063671847

| Você acertou!

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

| Agora faça o mesmo, mas com uma média de 100 e um desvio padrão de 25.

> ?rnorm

> rnorm(10, mean=100, sd=25)

[1] 54.99045 127.92137 55.56936 114.48847 99.11480 89.75835 75.51234 65.19187 97.27927 107.52484

| Todo o trabalho duro está dando resultado!

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

| Agora um problema mais difícil. E se quisermos simular 100 "grupos" de números aleatórios, cada um contendo

| 5 valores gerados a partir de uma distribuição de Poisson com média de 10? Vamos começar com um grupo de 5

| números, depois mostrarei como repetir a operação 100 vezes de maneira conveniente e compacta.

...

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

| Gere 5 valores aleatórios a partir de uma distribuição de Poisson com média 10. Veja a documentação de

| rpois() se precisar de ajuda.

> ?rpois

> rpois(5,10)

[1] 17 7 7 9 11

| Ótimo!

|============================================================================================================================================================================== | 80%

| Agora use a instrução replicate(100, rpois(5, 10)) para executar esta operação 100 vezes. Armazene o resultado em uma nova variável chamada my\_pois.

> my\_pois <- replicate(100, rpois(5,10))

| Você acertou!

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

| Dê uma olhada no conteúdo de my\_pois.

> my\_pois

[,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] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38]

[1,] 5 11 9 7 7 11 6 12 5 9 5 8 8 12 12 10 10 10 16 12 10 10 14 10 8 12 5 7 7 11 10 10 8 11 8 12 9 17

[2,] 11 8 17 9 10 11 4 10 11 4 7 9 17 11 15 6 8 8 10 10 11 4 6 12 10 12 10 5 8 12 14 12 11 9 13 11 10 9

[3,] 4 8 7 11 9 11 8 12 7 6 14 14 9 8 10 8 12 10 12 17 17 5 6 13 15 5 6 10 15 13 8 10 9 8 9 7 13 10

[4,] 8 13 9 4 7 8 5 14 11 9 19 7 9 15 8 10 15 9 8 14 8 16 8 9 6 13 11 8 11 8 11 14 7 21 16 4 10 3

[5,] 6 8 12 8 7 7 5 12 12 7 10 12 4 8 9 6 9 9 8 10 9 7 4 11 8 5 5 9 10 9 12 9 10 8 15 14 6 16

[,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74] [,75]

[1,] 8 7 5 16 13 9 18 6 10 15 13 16 8 14 10 8 6 9 14 12 13 12 9 4 12 4 17 9 13 14 10 7 9 17 7 8 17

[2,] 8 8 7 15 6 12 10 9 7 6 7 12 6 4 6 9 12 8 14 9 9 11 11 10 13 12 9 8 16 8 18 8 5 8 11 7 8

[3,] 8 10 11 6 15 6 9 9 6 5 11 12 14 8 11 11 11 10 8 3 10 12 7 12 7 10 10 9 11 7 15 6 5 10 11 9 11

[4,] 8 13 12 11 10 10 16 9 8 8 10 7 5 9 10 8 16 7 8 8 10 13 12 7 7 5 12 10 8 12 12 10 13 8 12 8 14

[5,] 13 11 8 13 13 12 7 14 9 12 8 14 8 9 10 8 11 14 8 11 9 14 11 8 9 11 16 9 9 8 13 16 6 5 10 9 12

[,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84] [,85] [,86] [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96] [,97] [,98] [,99] [,100]

[1,] 13 13 8 8 14 13 9 10 8 5 12 9 10 5 10 14 12 6 7 11 15 8 12 8 14

[2,] 6 8 10 8 17 6 13 9 7 8 11 12 7 7 7 9 15 15 10 5 8 14 6 6 7

[3,] 9 8 12 12 12 10 14 9 9 7 10 8 6 4 14 8 11 11 1 5 10 11 16 8 11

[4,] 11 6 13 10 7 8 12 16 15 13 17 9 4 10 6 7 11 12 14 8 18 13 12 12 10

[5,] 9 11 16 8 8 8 7 13 9 6 8 9 9 16 6 11 14 7 9 12 10 8 12 12 11

| Você acertou!

|========================================================================================================================================================================================== | 86%

| replicate() criou uma matriz, cada coluna contém 5 números aleatórios gerados a partir de uma distribuição de Poisson com média 10. Agora podemos encontrar a média de cada coluna em my\_pois usando a função colMeans().

| Armazene o resultado em uma variável chamada cm.

> cm <- colMeans()

Error in is.data.frame(x) : argument "x" is missing, with no default

> cm <- colMeans(my\_pois)

| Essa é a resposta que eu esperava.

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

| E vamos dar uma olhada na distribuição em um gráfico de colunas, trace um histograma com hist(cm).

> hist(cm)

| Excelente trabalho!

|====================================================================================================================================================================================================== | 91%

| Parece que nossas colunas são quase "normalmente distribuídas", certo? Esse é o Teorema do Limite Central, mas isso é uma lição para as aulas de estatística!

...

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

| Todas as distribuições de probabilidade padrão estão no R, incluindo exponencial (rexp()), qui-quadrado (rchisq()), gama (rgamma()), .... Bem, você já se ligou do padrão.

...

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

| Fazer simulações é praticamente um campo próprio dentro da estatística e nós apenas analisamos a superfície do que é possível. Existem muitas outras opções e funções que vocês podem explorar por conta própria. Este foi só o

| primeiro passo!

...

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

Confusa...

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

[1] "saved"

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

Seu resultado foi salvo!

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