Lista 2: Simulações
FLP 0468/FLS 6183

Prof. Manoel Galdino
Gabriel Mardegan
Pedro Reis

Para entregar até: 13 de setembro de 2024

O envio dos resultados dos exercícios das listas, via Moodle, deverão conter: 1) um arquivo em formato ".Rmd", com o script completo, incluindo os códigos utilizados; 2) um arquivo em formato ".pdf" gerado no próprio R (obrigatório para a Pós e opcional para a Graduação), através do RMarkdown (linguagem de marcação).

Revisando um pouco mais: alguns teoremas importantes

Antes de entrarmos na revisão de simulações, precisamos revisar (ou, para quem nunca viu, ter uma breve introdução) dois teoremas[1] fundamentais na Estatística: a Lei dos Grandes Números e o Teorema do Limite Central. Esses teoremas nos permitem fazer inferências estatísticas confiáveis sobre uma população de interesse a partir de uma amostra aleatória coletada. Satisfeitas certas premissas, eles nos asseguram estabilidade e precisão nas nossas estimativas.

Lei dos Grandes Números

A *Lei dos Grandes Números* diz respeito ao comportamento da média de variáveis aleatórias (v.a.s) independente e identicamente distribuídas (i.i.d.)[2]. Há duas versões dessa Lei: a Fraca e a Forte. Comecemos pela Lei Fraca.

Lei Fraca dos Grandes Números

A *Lei Fraca dos Grandes Números* postula que a *média amostral* de um grande número de v.a.s i.i.d. *converge em probabilidade* à esperança da distribuição. Isto é, suponhamos uma distribuição Normal de média 0 e variância 10. Tiramos n amostras aleatórias dessa distribuição. O que a Lei Fraca nos diz é que, conforme amostramos mais (isto é, n maior), torna-se cada vez mais provável que a média amostral dessas variáveis seja muito próxima à média populacional (no caso, 0).

[1] Melhor dizendo, são conjuntos de teoremas, pois há mais de uma Lei dos Grandes Números e diversos Teoremas do Limite Central.

[2] Veremos melhor em próximas aulas, mas essa é a uma premissa importante do teorema. Resumidamente, uma v.a. é independente quando o conhecimento de um valor extraído não nos informa sobre o valor de outro valor extraído, e vice-versa; por sua vez, a v.a. é identicamente distribuída quando os valores da nossa amostra vieram da mesma distribuição de probabilidade.

Formalmente: assume-se uma sequência de v.a.s i.i.d. $X_1, X_2, \cdots, X_n$ com esperança $\mu = \mathbb{E}[X_i]$ e variância σ^2. A média amostral, por sua vez é:

$$\bar{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{1}$$

Assim, para qualquer $\epsilon > 0$ (isto é, uma constante maior que zero, mas, muito pequena, aproximando-se de zero):

$$\lim_{n\to\infty} \Pr\left(|\bar{X}_n - \mu| < \epsilon\right) = 1 \tag{2}$$

Isto é, conforme n vai ao infinito, a probabilidade de que a diferença absoluta entre a média amostral e a média populacional seja menor que uma constante ϵ qualquer se aproxima de 1. Em outras palavras, conforme aumentamos o n, torna-se cada vez mais provável que a média amostral *muito* próxima à média populacional – reparemos que, na Lei Fraca, a média amostral não se torna igual à média populacional, mas muito próxima com alta probabilidade. Podemos deduzir qual é a afirmação da Lei Forte.

Lei Forte dos Grandes Números

A *Lei Forte dos Grandes Números* vai além: ela afirma que, com o aumento de n, a média amostral converge ao valor da esperança da distribuição – isto é, não é mais uma probabilidade de alta proximidade entre a média amostral e a média populacional, trata-se de uma certeza cada vez maior de igualdade entre os dois valores. Formalmente:

$$\Pr\left(\lim_{n\to\infty} \bar{X}_n = \mu\right) = 1 \tag{3}$$

A diferença entre a equação 3 e a equação 2 é até sutil, mas a equação 3 faz uma afirmação mais forte do que "convergência em probabilidade". Isto é, aqui, com n ao infinito, a probabilidade de que a média amostral é igual à média populacional é igual a 1 – é quase certo.

As duas Leis dos Grandes Números já nos dizem muito sobre as propriedades da média amostral. Afinal, esses teoremas nos dizem que, conforme o tamanho da nossa amostra aumenta, a média amostral converge à média populacional; além disso, as Leis mostram que, com uma amostra grande, a variabilidade nas médias amostrais diminui. Ou seja, com uma amostra suficientemente grande, nossa estimativa da média populacional torna-se mais confiável, mais próxima do valor verdadeiro e mais precisa.

Porém, precisamos de mais: precisamos saber qual distribuição de probabilidade nossa variável aleatória assumirá, conforme o tamanho da amostra aumenta, pois, a partir de então, teremos os pilares fundamentais para fazermos inferências estatísticas confiáveis.

Teorema do Limite Central

Há diversos *Teoremas do Limite Central*, cada um assumindo condições diferentes para as variáveis aleatórias de interesse ou assumindo formas resultantes diferentes. Para efeitos pedagógicos, falaremos do Teorema Clássico. O *Teorema Clássico do Limite Central* afirma que, se $X_1, X_2, \cdots, X_n$ são v.a.s i.i.d. com média μ e variância σ^2, então, conforme o n de amostras aumenta, o resultado da soma dessas variáveis se aproxima de uma distribuição Normal.

Formalmente:

$$\sqrt{n}\left(\bar{X}_n - \mu\right) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2\right) \tag{4}$$

Isto é, pelas Leis dos Grandes Números, sabemos que, conforme n tende ao infinito, nossa média amostral converge (tanto em probabilidade quanto quase certamente) à esperança μ. Com isso, a distribuição da diferença entre a média amostral e a média populacional, multiplicada por um fator $\sqrt{n}$, se aproxima da distribuição Normal com média 0 e variância σ^2, conforme n aumenta.

Na prática, isso significa que, aplicando os dois conjuntos de teoremas, conforme aumentamos o tamanho da nossa amostra, a nossa média amostral $\bar{X}_n$ terá, aproximadamente, uma distribuição Normal com média μ e variância σ^2/n. Ou seja, nossa média amostral convergirá para a média populacional e ainda teremos uma medida de precisão da nossa estimativa.

Simulações na teoria e no R

O que são *simulações*? São modelos de processos ou fenômenos reais; eles nos permitem criar um ambiente controlado para estudarmos um fenômeno (que assumimos ter sido criado por determinadas funções e distribuições de probabilidade) e interagirmos com ele. Assim, podemos replicar o comportamento de um sistema, para compreendermos seu funcionamento e até pensar e ver como esse comportamento é afetado se aplicarmos outros fatores sobre ele.

Podemos facilmente criar simulações no R! Para isso, utilizaremos algumas funções de aleatorização e de distribuição Normal para gerarmos valores quaisquer para nossas variáveis. Mas, também aprenderemos como replicar, de forma padronizável, sucinta, e clara, diversas simulações de uma vez: para isso, aprenderemos (revisaremos) `for loops` e *funções*.

Como material de apoio, recomendamos os seguintes materiais:

- https://rstudio-education.github.io/hopr/ (Livro *Hands-On Programming with R*, em inglês)
- https://jonnyphillips.github.io/Ciencia_de_Dados/ (Material do curso ministrado pelo professor Jonathan Phillips no DCP/USP, em português, com as gravações das aulas)
- https://r4ds.hadley.nz/ (Livro *R for Data Science, 2nd edition*, em inglês)

Lançando um dado de 6 lados

Comecemos a implementação de simulações com um exemplo simples: o lançamento de um dado de 6 lados. Assumindo um dado não-viciado, sabemos que a probabilidade analítica de um lado cair para cima $\mathbb{P}[X = x]$ é $1/6$ ou aproximadamente 0.167. Visto que um dado vai de 1 a 6, sua esperança é:

$$\begin{aligned}
\mathbb{E}[X] &= \sum x \cdot \mathbb{P}[X = x] \\
&= \sum (1 \cdot \frac{1}{6}) + (2 \cdot \frac{1}{6}) + \cdots + (6 \cdot \frac{1}{6}) \\
&= 3.5
\end{aligned}$$

Podemos observar isso empiricamente? Para isso, não basta lançar o dado apenas uma vez, obviamente. Quantas vezes? Como a Lei dos Grandes Números mostra, com um número cada vez maior

de lançamentos, aproximar-nos-emos[3] dos valores dos parâmetros – nesse caso, da probabilidade de cada lado cair e de sua esperança.

Escolhamos lançar o dado 1000 vezes. A cada lançamento, guardaremos o resultado. Após lançarmos 1000 vezes, analisaremos as propriedades da v.a. dada. Como implementar isso no R? Visto que repetiremos uma ação (o lançamento) diversas vezes, um "instrumento" de programação útil para esse propósito é um `for loop`, uma função que repetirá um código até uma condição ser satisfeita (no caso, chegarmos ao número máximo de lançamentos). Vejamos o exemplo abaixo:

```
# Lançamento de um dado ------
lancamentos <- 1000 # número de lançamentos
resultado <- numeric(lancamentos) # vetor do tamanho do número de lançamentos
#resultado[1] # a primeira posição do vetor resultado
#resultado[2] # a segunda posição do vetor resultado...
for (lance in 1:lancamentos) { # for loop de 1 até o número de lançamentos
    resultado[lance] <- sample(1:6, 1) # lançar um D6 não-viciado
}
```

O que fizemos aqui?

- Primeiro, criamos um objeto (chamado "lancamentos"[4]) para guardar o número de lançamentos que faremos, será útil no futuro.
- Depois, criamos um outro objeto chamado "resultado" onde guardaremos o resultado de cada lançamento: como são resultados numéricos, criamos um vetor numérico vazio com a função `numeric()`. Ademais, queremos que esse vetor tenha um tamanho específico, 1000 lançamentos; em vez de escrever 1000 dentro da função, podemos usar o objeto numérico "lancamentos" criado logo acima.
- Agora, começamos o `loop`. Para cada lançamento (indexamos como `lance`) entre 1 e 1000 lançamentos, execute o seguinte código: i) Equiprobabilisticamente, extraia (com a função `sample()`) um número entre 1 e 6 – isto é, lançamos o dado e vemos qual lado caiu. ii) Tendo um lado qualquer como resultado, guarde no vetor `resultado`, na posição equivalente ao lance feito – isto é, no 200º lance, o resultado será guardado na 200ª posição desse vetor.

Vamos ter uma noção rápida do vetor criado. Executemos a função `head()` para vermos os primeiros resultados do vetor.

```
head(resultado, n = 10) # os 10 primeiros resultados

##  [1] 2 2 6 4 3 5 3 6 5 3
```

Com isso, vamos examinar as características dessa variável. Lembrando: a esperança do dado de 6 lados é 3.5 e a probabilidade de cada lado é 1/6. Isso se aproxima na realidade?

[3] Devemos reiterar que essa aproximação é quase certa de acordo com a Lei Forte dos Grandes Nùmeros, mas converge em probabilidade de acordo com a Lei Fraca.

[4] Evite usar ç ou acentos na escrita dos códigos.

```r
mean(resultado) # média dos resultados

## [1] 3.493

prop.table(table(resultado)) # proporção de cada lado obtido no lançamento

## resultado
##     1     2     3     4     5     6
## 0.166 0.166 0.180 0.154 0.165 0.169
```

Vemos que sim! Experimente, porém, diminuir o número de lançamentos (para 10 ou 100) e re-execute as funções posteriores; os números observados divergem do que sabemos analiticamente? E se aumentarmos o número de lançamentos (para 10000 ou 100 mil)[5]? Os números convergem para o esperado analiticamente? Esperamos que sim, dado o que sabemos sobre "grandes números".

A Lei dos Grandes Números e o Teorema do Limite Central na prática

Vamos falar mais sobre simulações! Os lançamentos de um dado de 6 lados podem ser entendidos como uma variável aleatória uniforme discreta, onde cada lado obtido é equiprovável. Podemos pensar e modelar outros fenômenos físicos ou sociais em termos de v.a.s com famílias de distribuição definidas: podemos pensar v.a.s Normais, Poisson, Bernoulli, Binomial etc.

Como dissemos, com simulações podemos criar populações de tamanho qualquer, com variáveis vindas de qualquer tipo de distribuição com parâmetros de valores quaisquer. No próximo exemplo, vamos criar uma população de 10 milhões de observações. Nessa população, temos três variáveis: `x1`, que foi gerada de uma distribuição Normal com média -10 e desvio-padrão 10; `x2`, gerada de uma distribuição Normal com média 30 e desvio-padrão 10. As duas primeiras variáveis foram geradas de distribuições diferentes, logo, são v.a.s independentes entre si, nenhuma é função da outra. Em contrapartida, a terceira variável `y` é uma função das outras duas; no caso, decidimos que ela seja o resultado da soma das duas variáveis independentes.

```r
library(tidyverse)
# LLN e CLT -----
tamanho_pop <- 1e7 # tamanho da nossa população

mu_1 <- -10 # média populacional do nosso x1
mu_2 <- 30  # média populacional do nosso x2
populacao <- tibble( # criar uma tabela
    x1 = rnorm(tamanho_pop, mu_1, 10), # x1 vem de uma dist Normal(-10, 10)
    x2 = rnorm(tamanho_pop, mu_2, 10), # x2 vem de uma dist Normal(30, 10)
    y = x1 + x2 # y é uma função de x1 e x2; no caso, é a soma das duas
)
```

Sabemos as esperanças e variâncias das variáveis `x1` e `x2`. Qual é a esperança e a variância de `y`,

[5] Devemos notar que, quanto maior o número de iterações, o computador será mais demandado, o que resultará na demora da execução das funções.

visto que é a soma de duas v.a.s independentes Normalmente distribuídas? Se:

$$X_1 \sim \mathcal{N}\left(\mu_{X_1}, \sigma^2_{X_1}\right)$$
$$X_2 \sim \mathcal{N}\left(\mu_{X_2}, \sigma^2_{X_2}\right)$$
$$Y = X_1 + X_2$$

Então, a variável resultado da soma é:

$$Y \sim \mathcal{N}\left(\mu_{X_1} + \mu_{X_2}, \sigma^2_{X_1} + \sigma^2_{X_2}\right)$$

Isto é, y é uma variável normalmente distribuída com média igual a $-10 + 30 = 20$ e variância igual a $10^2 + 10^2 = 200$ – logo, o desvio-padrão de y é $\sqrt{200} \approx 14.14$.

Certo, criamos nossa população, nossas variáveis de interesse, e sabemos os parâmetros (média e desvio-padrão populacionais) de cada uma. Mas, na vida real, precisamos extrair uma amostra aleatória para estimar esses parâmetros populacionais. Façamos uma amostra de 10 observações.

```
amostra_teste <- sample_n(populacao, 10) # amostra aleatória de 10 observações
head(amostra_teste, n = 10) # uma noção breve da amostra
```

```
## # A tibble: 10 x 3
##        x1    x2     y
##     <dbl> <dbl> <dbl>
##  1  -4.10  41.2 37.1
##  2 -10.1   49.1 39.1
##  3  -2.63  42.3 39.7
##  4  -4.98  48.9 43.9
##  5 -15.8   48.6 32.8
##  6   3.98  38.7 42.7
##  7 -11.3   36.0 24.6
##  8  18.0   24.7 42.7
##  9 -18.1   32.7 14.5
## 10 -32.4   24.7 -7.63
```

Temos uma amostra de 10 observações – reparemos que usamos `sample_n()` em vez de `sample()`, pois extraímos uma amostra de uma tabela inteira, não apenas de um vetor. Vamos estimar as médias e desvios-padrões amostrais. Será que os valores amostrais são próximos dos valores verdadeiros?

```
amostra_teste %>% # calculando médias e desvios-padrões amostrais
    summarise_all(list(media = mean, desviopad = sd)) %>%
    pivot_longer(everything()) %>% # "girar" a tabela
    separate(name, c("variavel", "estatistica")) %>% # separar variável em duas
    pivot_wider( # "girar" de volta a tabela
        names_from = estatistica,
        values_from = value
    )
```

```
## # A tibble: 3 x 3
##   variavel media desviopad
##   <chr>    <dbl>     <dbl>
```

```
## 1 x1       -7.75     13.5
## 2 x2       38.7       9.21
## 3 y        30.9      16.4
```

Nem um pouco! Tanto a média amostral quanto o desvio-padrão amostral das nossas variáveis divergem consideravelmente de seus valores verdadeiros. Porém, conforme vimos na revisão da Lei dos Grandes Números acima, conforme aumentamos o tamanho da amostra, a estimativa amostral se aproxima do parâmetro populacional. Então, vamos aumentar o tamanho da nossa amostra! Mas, qual será o tamanho? Vamos comparar amostras de diferentes tamanhos: 10, 25, 50, 100, 250, 500, 1000, e 10 mil observações.

No R, podemos criar um código para cada amostra. Mas, aprendemos anteriormente a usar o `for loop`. Podemos padronizar o código! Como queremos amostrar diversas tabelas (de diferentes tamanhos) de uma população, não podemos guardar os resultados do `for loop` num mero vetor, como fizemos com os lançamentos dos dados. Agora, precisamos de um armazenamento mais complexo: uma *lista*. A lista pode guardar objetos de diversos tipos num mesmo objeto, é o que a torna tão versátil. No caso, queremos guardar diversos objetos de mesmo tipo (tabelas), mas de tamanhos diferentes. Segue o código abaixo:

```
amostras <- c(10, 25, 50, 100, 250, 500, 1000, 10000) # Ns possíveis
lista_amostras <- list() # uma lista vazia para guardar cada amostragem
for (amostra in seq_along(amostras)) { # para cada amostra
    lista_amostras[[amostra]] <- sample_n( # extrair aleatoriamente
                                    populacao, # da população
                                    amostras[amostra] # amostra de tamanho N
    ) # e armazenar em uma lista cada amostra
}
```

O que fizemos aqui?

- Criamos um vetor contendo cada tamanho de amostra.
- Depois, criamos uma lista vazia para guardarmos cada conjunto de dados amostrado.
- Criamos o `loop`: para cada amostra no vetor de amostras, amostre aleatoriamente a nossa populacao, com a amostra de tamanho respectivo a cada iteração do `loop`.
- Após amostrarmos a população, armazenaremos essa amostra em um espaço da nossa lista – reparemos que, por ser uma lista, seu indexador não é `[ ]` como a de um vetor, mas `[[ ]]`.

Criada a lista, podemos usar cada uma individualmente para fazermos nossas estimativas e criarmos gráficos. Para extrairmos um espaço específico da nossa lista, precisamos executar o código `lista_amostras[[i]]`, onde `i` é um número específico respectivo a cada espaço da lista. Como criamos amostras de 8 tamanhos diferentes, nossa lista tem 8 espaços para serem extraídos. Como exemplo, vamos extrair a tabela da terceira posição da lista – a tabela com 50 observações.

```
lista_amostras[[3]]

## # A tibble: 50 x 3
```

```
##          x1    x2      y
##       <dbl> <dbl>  <dbl>
##  1 -17.3     30.0  12.7
##  2 -16.1     45.6  29.5
##  3 -15.3     22.1   6.76
##  4  -7.64    27.5  19.8
##  5   3.95    35.0  38.9
##  6 -23.1     40.5  17.4
##  7 -23.9     10.0 -13.9
##  8 -21.7     38.6  16.9
##  9  -0.0753  33.3  33.2
## 10 -16.7     25.1   8.42
## # i 40 more rows
```

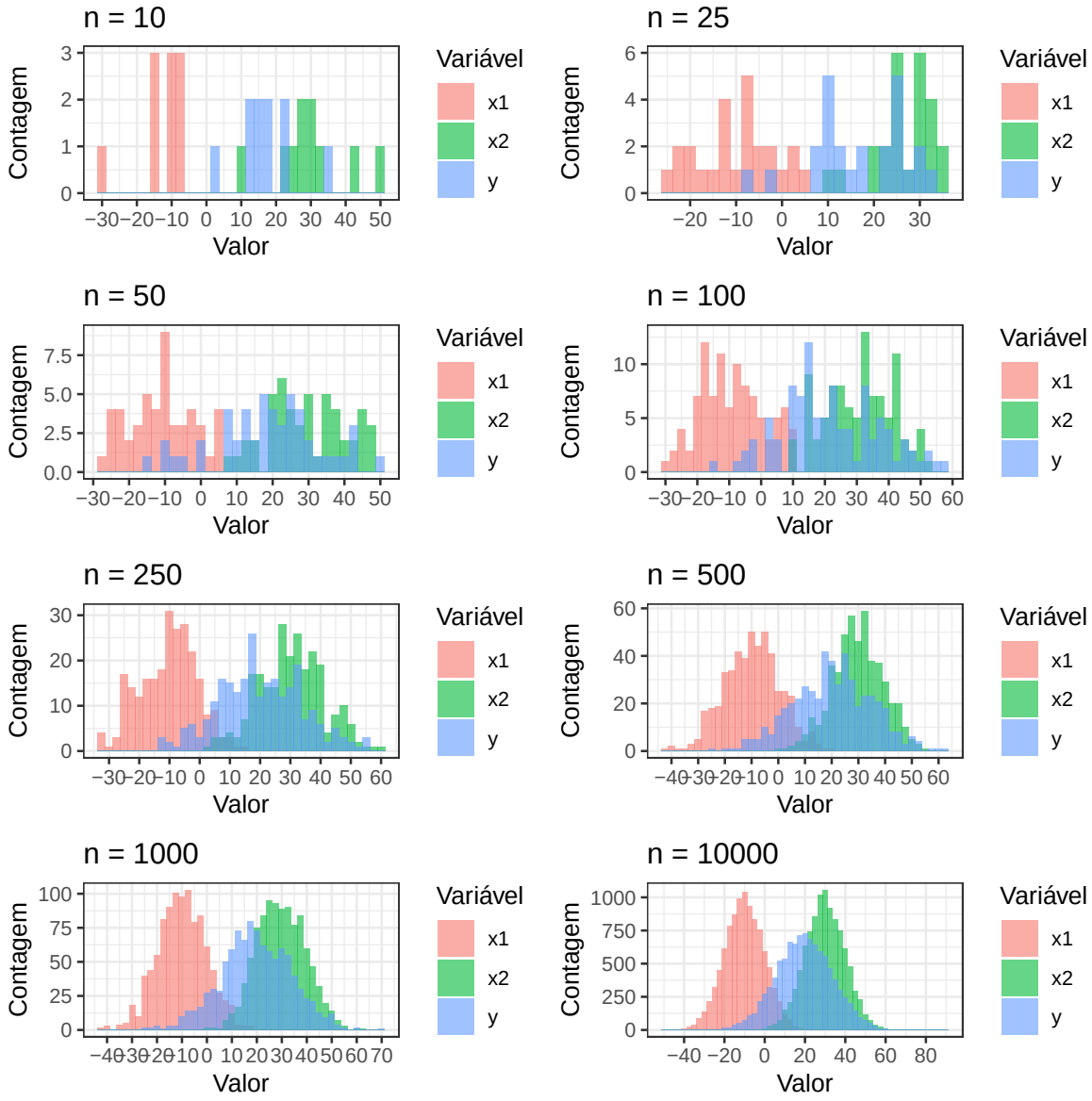
Ótimo! Com isso, podemos fazer os nossos cálculos e visualizações!

Para salientarmos a Lei dos Grandes Números nas amostras já coletadas, segue abaixo uma tabela com as médias amostrais das nossas variáveis por tamanho de cada amostra. Sabemos que a esperança (a média verdadeira) de x1 é -10 e o desvio-padrão é 10; a esperança de x2 é 30 e o desvio-padrão é 10; e, por ser a soma das duas variáveis anteriores, y possui uma esperança igual a 20 e um desvio-padrão de aproximadamente 14.14.

N	x1	x2	y
10	media = -12.62, sd = 6.86	media = 30.1, sd = 10.79	media = 17.48, sd = 8.8
25	media = -9.83, sd = 8.39	media = 26.49, sd = 6.47	media = 16.67, sd = 10.28
50	media = -11.14, sd = 8.99	media = 29.75, sd = 10.84	media = 18.61, sd = 14.48
100	media = -9.02, sd = 10.01	media = 30.97, sd = 10.45	media = 21.95, sd = 15.3
250	media = -10.15, sd = 9.36	media = 30.24, sd = 10.66	media = 20.1, sd = 13.85
500	media = -9.69, sd = 10.33	media = 29.75, sd = 9.52	media = 20.06, sd = 14.63
1000	media = -10.04, sd = 9.82	media = 29.72, sd = 9.75	media = 19.68, sd = 14.07
10000	media = -10.17, sd = 9.91	media = 29.85, sd = 9.96	media = 19.68, sd = 14.06

Como podemos ver, nas amostras menores, as médias amostrais e os desvios-padrões amostrais de cada variável desviam bastante de suas respectivas esperanças e desvios-padrões. Porém, conforme o tamanho da amostra aumenta, essas médias e desvios se aproximam cada vez mais de seus valores verdadeiros. Logo, uma amostra aleatória grande torna mais provável a identificação dos parâmetros das nossas variáveis de interesse.

Bom, sabemos as medidas-resumo das nossas variáveis por tamanho da amostra. Mas, e quanto as suas distribuições? Sabemos, pelo Teorema do Limite Central, que conforme o tamanho da amostra aumenta, a distribuição da nossa média amostral converge para uma distribuição Normal em torno da nossa média populacional. Na figura abaixo, estamos lidando com as variáveis inteiras, não apenas a média amostral; mas, podemos observar um comportamento similar:



A figura acima demonstra visualmente a ação conjunta do Teorema do Limite Central e da Lei dos Grandes Números, onde, conforme o tamanho da amostra aumenta, não só a média amostral de cada variável converge ao seu respectivo parâmetro, como a distribuição de cada variável se aproxima de uma distribuição Normal em torno de suas respectivas esperanças.

Passeio aleatório ou "andar do bêbado"

Para terminarmos o assunto sobre simulações, vamos olhar um exemplo clássico como material extra[6]: o *passeio aleatório* – também chamado de "andar do bêbado". O processo de passeio aleatório é um fenômeno estatístico onde um "passeio" consiste de uma série de passos aleatórios.

Pensemos no caso unidimensional, onde o "bêbado" começa de uma posição inicial $X_0 = 0$, e, a cada passo subsequente, o bêbado pode dar um passo para a esquerda ou um passo para a direita

[6]Logo, não se refletirá nos exercícios, mas, pode ajudar a pensar processos aleatórios e construir a intuição necessária para pensar a modelagem estatística dos dados.

com igual probabilidade. Definamos formalmente o processo do passeio:

$$X_n, \quad \text{a posição do bêbado após n passos.}$$
$$X_0 = 0, \quad \text{isto é, o bêbado começa na origem.}$$

Agora, a cada passo i, o bêbado se move da posição X_{i-1} para a posição X_i – ou seja, a posição futura do bêbado depende apenas da sua posição presente e não onde estava em posições passadas. Ademais, o movimento a cada passo é definido por:

$$X_i = X_{i-1} + e_i$$

Onde e_i é uma variável aleatória que assume o valor $+1$ com probabilidade $\frac{1}{2}$, e -1 com probabilidade $\frac{1}{2}$ – isto é, a partir da sua posição presente, o bêbado dará um passo à esquerda ou à direita, equiprobabilisticamente.

Como implementamos e visualizamos o passeio aleatório no R? Há diversas formas. Aprendemos como criar `for loops`; certamente, esse instrumento será importante, pois, queremos dar muitos passos com nosso bêbado. Mas, queremos ser capazes de guardar nosso `for loop` do passeio e poder customizar rapidamente o número de passos que o passeio terá (seja 10, 100, 1000, etc.). Para isso, vamos colocar nossos `loops` dentro de *funções*.

Uma função personalizada `function() { }` não precisa de um input; a cada execução, a função simplesmente reproduzirá o que está dentro dela. Mas, como queremos ditar facilmente o número de passos que o passeio terá, colocamos um `x` qualquer dentro de `function`; para facilitar a leitura do código, esse `x` terá o nome de `n_passos`. Para o nosso caso, seguiremos uma lógica similar ao do loop do lançamento de dados. Vejamos o código abaixo:

```
# Passeio aleatório, o "andar do bêbado" ------
fun_bebado <- function(n_passos) { # função personalizada de nome "fun_bebado"
    x <- numeric(n_passos + 1) # vetor vazio para armazenar o passeio
    for (i in 2:(n_passos + 1)) { # for loop começa no primeiro passo
        e <- sample(c(1, -1), 1) # um passo à "esquerda" ou à "direita"
        x[i] <- x[i - 1] + e # o passo futuro...
    }                        # depende apenas do presente

    return(x) # retorna o vetor do andar do bêbado
}
```

O que fizemos aqui? Muita coisa para o nosso conhecimento atual de R![7] O que torna essencial a documentação de cada passo que tomarmos no código. Dentro da nossa função, que nomeamos de `fun_bebado`:

- Criamos um vetor numérico vazio `x` para armazenarmos nosso passeio. Como partiremos de uma origem definida $X_0 = 0$, o tamanho do nosso vetor deve ser igual ao número de passos que daremos $+1$.

- Comecemos nosso `for loop` para o bêbado dar os passos i: como a primeira posição é a origem, então o loop de passos não pode ir da posição 1 (a origem) até o passo final, mas da

[7] Recomendamos que leiam os textos indicados no início da lista.

posição 2 (o primeiro passo dado) até o passo final.

- Dentro do loop, criamos nossa variável aleatória e do passo i: equiprobabilisticamente, o passo será à "esquerda" (1) ou à "direita" (-1).
- Com isso, a posição futura do bêbado `x[i]` dependerá da sua posição inicial `x[i - 1]` + e, a direção do próximo passo. Fica óbvio de ver que, na posição 1 (o primeiro passo), `x[i - 1]` é igual a 0 (a origem).
- Após enchermos nosso vetor `x` com o passeio de tamanho `n_passos`, retorne o seu resultado.

Então, criamos nossa função `fun_bebado` que aceita um número positivo qualquer referente ao número de passos do passeio aleatório. Vamos experimentá-la! Queremos um passeio aleatório de 15 passos. O resultado dessa função será um vetor que retorna a posição do bêbado a cada passo desde a origem.

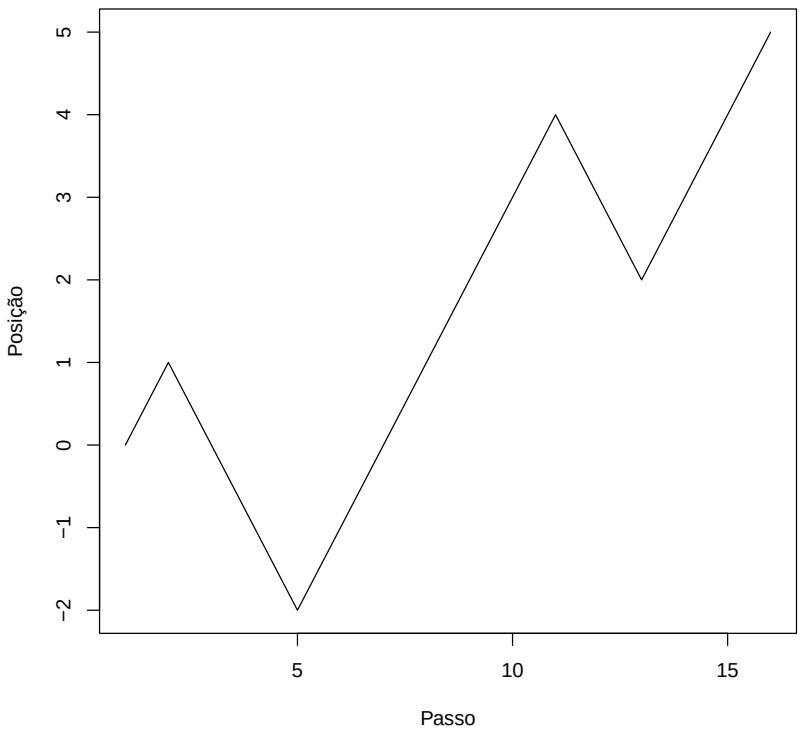
```
passeio_1 <- fun_bebado(15) # passeio aleatório de 15 passos
passeio_1

##  [1]  0  1  0 -1 -2 -1  0  1  2  3  4  3  2  3  4  5
```

Executamos nosso primeiro passeio de 15 passos. O que vemos aqui? O bêbado começou na posição 0, a origem. Deu diversos passos à esquerda e à direita, com a posição final sendo 5 passos à esquerda da origem.

Podemos visualizar esse passeio com uma função de visualização do R base – veremos adiante como visualizar passeios nas funções `ggplot()`.

```
plot(passeio_1, type = "l", xlab = "Passo", ylab = "Posição")
```



Simulamos um passeio aleatório! Um passeio cuja posição futura depende apenas da posição do presente e cuja direção depende de um "cara-ou-coroa". Logo, não conseguimos saber onde esse passeio irá parar a partir do nosso conhecimento sobre os passos anteriores (os dados históricos de sua trajetória).

Isso não significa que um passeio aleatório seja imprevisível ou que não possua estrutura. O que observamos acima foi uma possível trajetória de um objeto de interesse (como o crescimento de uma população ou o preço de uma ação na bolsa). Se executarmos a nossa função customizada diversas vezes, veremos diversos outros passeios com trajetórias diferentes. Assim, o processo de passeio aleatório nos permite pensar sobre o comportamento de sistemas complexos (que são condicionados por um número de fatores) e pensar sobre a incerteza em torno desse comportamento: para o nosso propósito, qual é o comportamento médio do passeio e qual o seu desvio (ou variância)?

Para isso, podemos criar diversos passeios com um grande número de passos e juntá-los num mesmo gráfico para pensá-los agregadamente. Mas, antes de implementarmos no R (algo que, a essa altura, temos uma noção de como fazer), vamos pensar as propriedades analíticas desse passeio aleatório simples, pois nos ajudará no futuro.

Bom, a essa altura, sabemos que nosso passeio aleatório pode ser definido como:

$$X_n = \sum_{i=1}^{n} e_i$$

Onde X_n é a posição final do bêbado após n passos; e_i é uma variável aleatória Bernoulli que assume, equiprobabilisticamente, resultados possíveis $\{-1, +1\}$. Logo, X_n é a somatória de todos os passos desde o primeiro passo até n passos. Sabemos também que o nosso bêbado começa da origem $X_0 = 0$.

Dado isso, o passeio aleatório possui esperança e variância definidas? Surpreendentemente, apesar da aparente imprevisibilidade do passeio, sim! Achemos sua esperança:

$$\begin{aligned}
\mathbb{E}[X_n] &= \mathbb{E}\left[\sum_{i=1}^{n} e_i\right] \\
&= \sum_{i=1}^{n} \mathbb{E}[e_i] && \text{(Usando as propriedades da somatória)} \\
&= \sum_{i=1}^{n} 0 && \text{(Afinal: } \mathbb{E}[e_i] = \textstyle\sum(-1)\cdot\frac{1}{2} + (+1)\cdot\frac{1}{2}\text{)} \\
&= 0
\end{aligned}$$

A esperança do passeio aleatório simples é definida! E é zero! Isso quer dizer que, i) se pensarmos os passeios como uma variável aleatória, sua média é igual a zero; ademais, ii) ao longo de muitos passos, é igualmente provável que o nosso bêbado esteja acima ou abaixo do ponto de origem. Em breve, observaremos isso empiricamente. Antes, precisamos saber a variância do nosso passeio aleatório, para termos uma noção mais qualificada da incerteza das trajetórias; como já sabemos sua esperança, tudo fica mais fácil:

$$\begin{aligned}
\text{Var}[X_n] &= \mathbb{E}\left[(X_n - E[X_n])^2\right] \\
&= \mathbb{E}\left[X_n^2\right] \quad \text{(Afinal: } \mathbb{E}[X_n] = 0\text{)}
\end{aligned}$$

Certo, descobrimos facilmente que a variância do nosso passeio aleatório é igual à esperança do quadrado da posição final do passeio. O que nos dá uma forma de saber a dispersão do andar do bêbado. Mas, podemos ir um pouco mais além para saber o que é de fato $\mathbb{E}\left[X_n^2\right]$. Antes, vamos ver o que é X_n^2. Prestemos atenção:

$$\begin{aligned} X_n^2 &= \left(\sum_{i=1}^{n} e_i\right)^2 \\ &= (e_1 + \cdots + e_n)^2 \\ &= (e_1^2 + \cdots + e_n^2) + 2 \cdot (e_1 e_2 + e_1 e_3 + \cdots + e_{n-1} e_n) \\ &= \sum_{i=1}^{n} e_i^2 + 2 \cdot \left(\sum_i \sum_j e_i e_j\right) \end{aligned}$$

Com isso, podemos calcular a esperança de X_n^2:

$$\begin{aligned} \mathbb{E}\left[X_n^2\right] &= \mathbb{E}\left[\sum_{i=1}^{n} e_i^2 + 2 \cdot \left(\sum_i \sum_j e_i e_j\right)\right] \\ &= \mathbb{E}\left[\sum_{i=1}^{n} e_i^2\right] + \mathbb{E}\left[2 \cdot \left(\sum_i \sum_j e_i e_j\right)\right] \\ &= \mathbb{E}\left[\sum_{i=1}^{n} e_i^2\right] + 2 \sum_i \sum_j \mathbb{E}\left[e_i e_j\right] \\ &= \mathbb{E}\left[\sum_{i=1}^{n} e_i^2\right] + 2 \sum_i \sum_j 0 \quad (\text{Pois, } \mathbb{E}\left[e_i e_j\right] = \mathbb{E}[e_i]\mathbb{E}[e_j] = 0 \cdot 0 = 0) \\ &= \mathbb{E}\left[\sum_{i=1}^{n} e_i^2\right] \\ &= n \quad (\text{Pois, } e_i^2 = (-1)^2 = 1^2 = 1 \text{ e } \sum 1 = n) \end{aligned}$$

Demoramos, mas, chegamos num resultado interessantíssimo: a variância do passeio aleatório é igual ao seu número de passos. Isso quer dizer que, em um processo de passeio aleatório simples, não obstante a sua média ser igual a zero, a dispersão em torno de sua média aumenta conforme o número de passos aumenta. Intuitivamente, faz muito sentido, isto é: as trajetórias possíveis aumentam conforme mais passos são dados, pois são cada vez mais caminhos e desvios que um "bêbado" (ou uma determinada dinâmica populacional) pode tomar[8].

Agora que sabemos as propriedades estatísticas do passeio aleatório, em vista dos teoremas revistos, queremos saber a dinâmica no longo prazo, conforme o número de passos e o número de amostras aumentam: queremos passeios mais longos e queremos saber as trajetórias de outros passeios. Podemos implementar no R! Assumimos que essas propriedades são atingidas no longo prazo, isto é, quanto mais iterações e quanto mais passos os passeios tiverem. Queremos números grandes: vamos definir que cada passeio terá 1000 passos e vamos simular 100 passeios. Ademais, queremos guardar cada passeio em um único objeto cujo elemento em comum entre cada passeio é a posição de origem ($X_0 = 0$); podemos fazer isso facilmente criando uma matriz vazia cujo número de linhas é igual ao número de passos mais 1 (como anteriormente), e o número de colunas é igual ao número

[8] Logo, o desvio-padrão do passeio é $\sqrt{n}$.

de passeios.

Já sabemos usar `for loops`. Para cada passeio, queremos executar nossa função de passeio aleatório `fun_bebado` de tamanho determinado (no caso, 1000) e armazenar cada resultado em cada coluna da matriz. Com a matriz preenchida, vamos transformá-la em tabela para facilitar a manipulação. Depois disso, podemos olhar um pedaço da tabela para ter noção do resultado. Observamos isso no código abaixo.

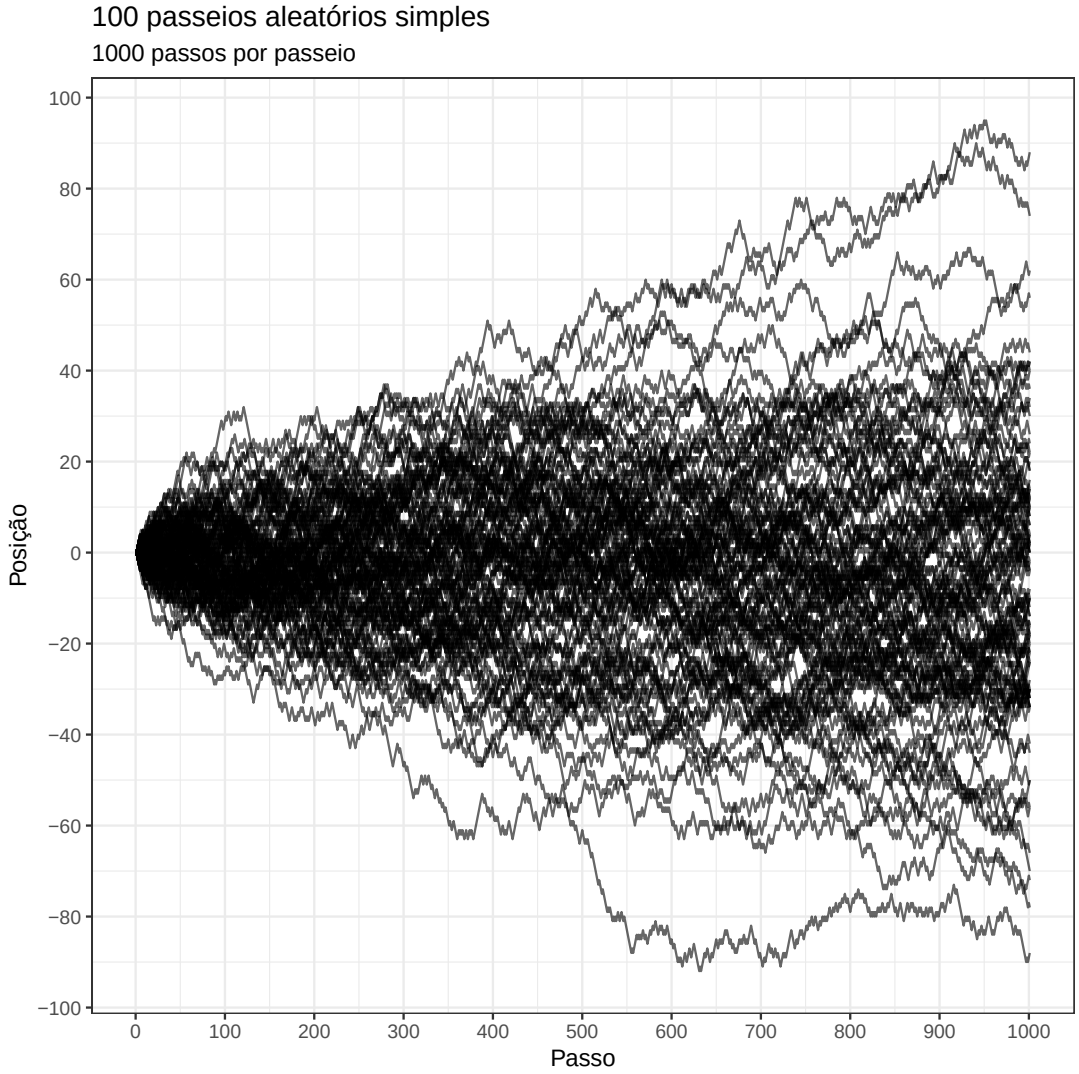
```
n_passos <- 1000 # 1000 passos
n_iteracoes <- 100 # 100 passeios
passeios <- matrix(0, nrow = n_passos + 1, ncol = n_iteracoes) # matriz vazia
for (j in 1:n_iteracoes) {
  passeios[, j] <- fun_bebado(n_passos) # cada coluna da matriz é um passeio
}
passeios <- as_tibble(passeios) # transformar a matriz em uma tabela
passeios[1:5, 1:10] # os primeiros 5 passos dos 10 primeiros passeios

## # A tibble: 5 x 10
##      V1    V2    V3    V4    V5    V6    V7    V8    V9   V10
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     0     0     0     0     0     0     0     0     0     0
## 2     1     1     1    -1    -1     1    -1     1    -1     1
## 3     0     2     2    -2     0     0    -2     0     0     0
## 4    -1     1     3    -1     1    -1    -3    -1     1    -1
## 5    -2     0     4    -2     2    -2    -2    -2     0    -2
```

Vemos no *output* os primeiros 5 passos dos 10 primeiros passeios da tabela. Vemos que todos começam no ponto de origem, mas tomam trajetórias iniciais diferentes (vemos que alguns passeios são similares nos primeiros passos). Vamos visualizar todos os passeios em um único gráfico. Isso é feito facilmente usando as funções `ggplot()` aprendidas.

```
passeios %>%
    mutate(id = row_number()) %>% # cada linha é a posição no passeio
    pivot_longer(-id) %>% # "girar" a tabela para colocar os passeios
    ggplot(aes(           # e os valores de cada passeio em duas colunas
        x = id, # o eixo x é o número de passos
        y = value, # posição atingida em cada passo
        group = name # cada linha é um passeio
        )
    ) +
    geom_line( # gráfico de linha
        colour = "black",
        alpha = 0.6 # transparência
    ) +
    labs(
        x = "Passo",
        y = "Posição",
        title = "100 passeios aleatórios simples",
        subtitle = "1000 passos por passeio"
    ) +
    scale_x_continuous(breaks = scales::breaks_extended(n = 10)) +
```

```
    scale_y_continuous(breaks = scales::breaks_extended(n = 10)) +
    theme_bw()
```



Uau! Uma demonstração empírica do que derivamos analiticamente sobre o passeio aleatório: cada passeio aleatório começou na mesma posição 0; conforme o número de passos aumenta, a dispersão em torno da média 0 aumenta; por fim, visto que a esperança do passeio é 0, observamos uma maior concentração de linhas em torno de 0, e menor concentração de linhas nas posições mais distantes de 0 – não à toa, esse passeio aleatório simples é chamado de "passeio aleatório simétrico". Se aumentarmos o número de passos e o número de passeios, isso fica cada vez mais claro. É um caos, mas um caos ordenado: podemos usar isso com um ponto de partida para modelar comportamentos de fenômenos e obter medidas pontuais e intervalos de incerteza.

Exercícios

Nestes exercícios, faremos simulações e analisaremos algumas propriedades teóricas de variáveis aleatórias. Em cada questão, explique detalhadamente tanto seu raciocínio e cálculos quanto seu código, etapa por etapa.

1 Começando a simular dados

1. Execute os seguintes códigos: `help(rnorm)` e `help(rbinom)` e leia a documentação de cada função. Nas suas palavras, do que se trata cada função `r_____`? Quais são seus parâmetros principais?

2. Pense em duas distribuições de probabilidade: i) distribuição Normal com média 0 e desvio-padrão 10, e ii) distribuição Binomial com dois resultados possíveis (0 e 1), 100 tentativas e probabilidade de sucesso igual a 0.7. Qual é a *esperança* e a *variância* dessas distribuições?

3. Gere dois vetores de 100 observações cada vindo das distribuições do item anterior. Guarde os dois vetores em objetos distintos e gere dois histogramas para cada.

4. Calcule a média e a variância amostrais dos vetores gerados anteriormente. Os valores obtidos são iguais aos valores teóricos obtidos no item 1.2? Por quê?

5. Gere dois novos vetores com os mesmos parâmetros do item 1.2, também com 100 observações cada. Calcule a média e a variância amostrais desses vetores. Os valores obtidos com esses vetores são iguais aos valores teóricos? Ademais, são iguais ou diferentes aos valores obtidos no item 1.4? Por quê?

2 Simulando a Normal múltiplas vezes

1. Aprendemos a como usar `for loops`. Crie um loop em que uma distribuição Normal de 10 observações, média 0 e desvio-padrão é gerada, sua média é calculada e armazenada em um *vetor*. Repita esse processo 10 vezes. Verifique se o vetor foi gerado corretamente: devemos ter 10 médias.

2. Agora, repita o item 2.1, mas, agora o *novo* loop deve rodar 50 vezes. Verifique se o *novo* vetor foi gerado corretamente: devemos ter um vetor com 50 médias.

3. Com os vetores gerados nos itens 2.1 e 2.2, gere um histograma para cada e calcule suas médias. Do que se trata cada histograma e cada média?

4. Agora, simule 10, 30, 50, 100, e 1000 médias da distribuição Normal do item 2.1. Você pode fazer `for loops` diferentes para cada simulação diferente ou você pode criar um único `for loop` e armazenar cada simulação em uma lista. Faça o procedimento que preferir – recomendamos fazer a versão reduzida para você praticar a escrita de códigos mais sucintos.

5. Calcule a média e crie um gráfico de densidade para cada simulação do item 2.4. O que você observa comparando cada gráfico e cada média? Mobilizando o que você aprendeu (e entendeu) sobre a Lei dos Grandes Números e o Teorema do Limite Central, como você entende o comportamento dessas médias e gráficos, como um todo?

3 Um pouco mais sobre variáveis aleatórias

Tendo as seguintes variáveis aleatórias, todas de tamanho 100:

$$U \sim Unif(0,1)$$
$$B \sim Binom(100, 0.6)$$

1. Quais distribuições são essas e quais são suas respectivas esperanças?

2. Com um loop, gere 10, 100 e 1000 médias para cada variável aleatória (funções `runif()` e `rbinom()`) e armazene cada iteração de cada variável em objetos apropriados (vetores ou listas) – não esqueça de nomear adequadamente cada objeto. Mostre que o código foi bem-sucedido em seu objetivo.

3. Calcule as médias e gere histogramas para cada iteração do loop criado no item anterior. Sabemos que essas variáveis não vieram de distribuições Normais. Mobilizando o Teorema do Limite Central, qual sentido você dá para o conjunto de histogramas?

CRÉDITO EXTRA: Por fim, para exercitar o cálculo da soma de duas v.a.s independentes Normalmente distribuídas (revisite a seção *A Lei dos Grandes Números e o Teorema do Limite Central na prática* da lista, se preciso), dadas as seguintes v.a.s independentes:

$$X \sim \mathcal{N}\left(\mu_X = 5, \sigma_X^2 = 30\right)$$
$$Z \sim \mathcal{N}\left(\mu_Z = 10, \sigma_Z^2 = 100\right)$$

4. Qual é a distribuição da seguinte variável:

$$Y = X + Z$$

5. E qual é a distribuição da seguinte variável? (Revise as propriedades da esperança e da variância para fazer a multiplicação de variáveis aleatórias)

$$W = 2 \cdot X - 3 \cdot Z$$

6. Por fim, qual é a distribuição da seguinte variável?

$$U = 0.5\frac{X}{Z}$$