Intro sim

Manoel Galdino

2023-05-05

Introdução à Simulação

Simulação é uma ferramenta extremamente importante em probabilidade e estatística. Dois são seus principais usos. De um lado, é possível usar um modelo de probabilidade para recriar, artificialmente, no computador, amostras desse modelo, quantas vezes quisermos. Tudo se passa como se tivéssemos o poder de criar um mundo, totalmente controlado por nós, e podemos simular esse mundo e verificar o que acontece. Isso é útil quando não podemos ou não queremos verificar matematicamente o que deveria acontecer com base no modelo matemático de probabilidade. De outro lado, é possível usar simulações para aproximar quantidades matemáticas que não podemos ou não queremos calcular analiticamente. Por calcular analiticamente, é fazendo a conta nós mesmos. Um exemplo simples disso é, em vez de calcular os resultados de alguma análise combinatória usando as fórmulas de combinação e simularmos os valores e contar quantas combinações resultam.

No geral usamos simulação para o primeiro tipo de uso. Mas, aqui no curso, será útil que vocês verifiquem resultados matemáticos por meio de simulações. Assim, se escrevemos que $\sum_{i=1}^{10} i = 55$, vocês podem verificar com código no R essa soma, por exemplo, rodando: `sum(1:10)`. Sempre que vocês tiverem dúvida de algum passo matemático, podem fazer uma simulação para entender melhor o que está acontecendo. Isso é encorajado, para melhorar a intuição do que está acontecendo e lhe assegurar que de fato você está entendendo o que está acontecendo. Podemos usar simulações para aproximar quantidades. Um exemplo clássico em probabilidade é a fórmula de Stirling, dada por $n! \sim \sqrt{2\pi}n^{n+1/2}e*^{-n}$. Então, por exemplo, $10! = 10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1$ com os computadores modernos pode ser calculada diretamente para números não tão grandes. No caso, $10! = 3628800$. Ou pode ser aproximada pela fórmula de Stirling:

exemplo 1.1

```
# especificando semente, para simulação ser reproduzível
# número de amostras
stirling_aprox <- function(n) {
  sqrt(2*pi)*n^(n+1/2)*exp(-n)
}

print(stirling_aprox(10))
```

[1] 3598696

```
# razão da aproximação para o valor correto
stirling_aprox(10)/3628800
```

[1] 0.991704

```
# erro percentual
1 - stirling_aprox(10)/3628800 # 0,8%
```

[1] 0.00829596

Dito isso, voltemos ao primeiro caso, de simulação de modelos de probabilidade.

Comecemos lembrando que probabilidades podem ser interpretadas como frequências relativas de longo prazo. Portanto, a probabilidade de um evento pode ser aproximada por simulações, na medida em que aproximamos a frequência relativa com que o fenômeno acontece em nossas simulações. Vamos dar um exemplo simples desse tipo de aplicação.

Suponha que quero simular a probabilidade do número 6 sair em um dado de seis lados.

Exemplo 1.

```
# especificando semente, para simulação ser reproduzível
set.seed(234)
# número de amostras
n <- 10000

# 1000 amostras de uma lançamento de dado de 6 lados
resultado <- sample(1:6, n, TRUE)

# frequência relativade 6 é dada por número de 6 / total de amostras
prob_6 <- sum(resultado == 6)/n

# 16,89%
# 1/6 = 16.6666
```

Podemos também ver como a aproximação converge para o verdadeiro valor à medida que n cresce.

```
# especificando semente, para simulação ser reproduzível
set.seed(234)
# número de amostras

vec_amostra <- c(100, 1000, 10000, 100000, 1000000)

# lista vazia para armazenar os resultados das simulações
resultado_lista <- list()

# vetor vazio para armazenar a frequência relativa de 6
vec_prob6 <- numeric()

set.seed(234)
# loop sobre os tamanhos das amostrar
for ( i in 1:length(vec_amostra)) {
  # n amostras de uma lançamento de dado de 6 lados
resultado_lista[[i]] <- sample(1:6, vec_amostra[i], TRUE)

# frequência relativade 6 é dada por número de 6 / total de amostras
vec_prob6[i] <- sum(resultado_lista[[i]] == 6)/vec_amostra[i]

}

print(vec_prob6)
```

[1] 0.150000 0.189000 0.164700 0.169250 0.166257

Se supusermos que todos os números possuem a mesma chance de sair quando jogamos os dados, então, a distribuição conjunta de X e Y pode ser dada por:

```
library(knitr)
library(kableExtra)
```

Table 1: Tabela 2.26: Tabela representando a distribuição conjunta da soma (X) e o maior (Y) de dois lançamentos de um dado de quatro faces

(X,Y)	$P(X=x, Y=y)$
(2, 1)	0.0625
(3, 2)	0.1250
(4, 2)	0.0625
(4, 3)	0.1250
(5, 3)	0.1250
(5, 4)	0.1250
(6, 3)	0.0625
(6, 4)	0.1250
(7, 4)	0.1250
(8, 4)	0.0625

```
# Definir os valores de (x, y) e P(X = x, Y = y)
valores <- c("(2, 1)", "(3, 2)", "(4, 2)", "(4, 3)", "(5, 3)", "(5, 4)", "(6, 3)", "(6, 4)", "(7, 4)", "
probabilidades <- c(0.0625, 0.1250, 0.0625, 0.1250, 0.1250, 0.1250, 0.0625, 0.1250, 0.1250, 0.0625)

# Criar a tabela
tabela <- data.frame("(x, y)" = valores, "P(X = x, Y = y)" = probabilidades)

# Formatar a tabela
tabela_formatada <- kable(tabela, caption = "Tabela 2.26: Tabela representando a distribuição conjunta d
                          col.names = c("$(X,Y)$", "$P(X=x, Y=y)$")) %>%
  kable_styling(bootstrap_options = "striped")

# Imprimir a tabela
print(tabela_formatada)
```

como podemos ver, à medida que n cresce, a simulação converge para o verdadeiro valor do parâmetro, embora isso não seja linear. Por isso que é importante variar a configuração para ter certeza de que a simulação está próxima do verdadeiro valor do parâmetro.

De maneira geral, uma simulação envolve os seguintes passos.

Configuraçao

Definir o modelo de probabilidade, variáveis aleatórias e eventos a serem modelados. O modelo de probabilidade codifica todas as suposições do modelo. No exemplo acima, de que todos os lados têm a mesma probabilidade de sair, que existem apenas seis possibilidades, numeradas de $1-6$ e assim por diante.

Simular

Resumir

Análise de sensibilidade

Um dos resultados mais contraintuitivos de modelos de probabilidade está relacionado à chamada "falácia do jogador". A falácia é a crença de que, porque saiu um número par (ou preto) na última jogada em uma roleta no cassino, então aumenta a chance de sair ímpar (vermelho) na próxima rodada, para que os números se equilibrem. Porém, cada jogada é independente da anterior. Um problema relacionado é o chamado problema da ruína do jogador.

Exemplo 1.2. Ballot theorem (toerema da urna de votação? teorema da votação?)

Esse teorema, provado em 1878 por W. A. Whitworth e independentemente em 1887 por J. Bertrand estabelece a probabilidade de que, durante a apuração dos votos em uma urna, sempre haja mais votos contabilizados para o candidato P do que para o candidato Q, quando P possui p votos e Q possui q votos, com $p > q$. A fórmula para essa probabilidade é $(p-q)/(p+q)$.

Vamos verificar esse teorema por meio de simulação? Configuração: suponha que $p = .51$ e $q = .49$. O que sigifnica que é $(p-q)/(p+q) = .02/1 = 2\%$ Primeiro, faremos uma simulação, e depois repetiremos k vezes a simulação para obter a frequência relativa.

```
set.seed(234)
# número de votos na urna
n = 1000

# 1 é o voto para P, -1 o voto para Q.
# então geramos n votos e armazenamos em votos

# vamos usar rbinom, que gera 0 ou 1.
votos <- rbinom(n, 1, p=.51)

# transformado 0 em -1
votos <- ifelse(votos == 0, -1, votos)

# Agora, iremos contar o primeiro voto e registrar para quem vai. Depois o segundo e o terceiro etc. at
apuracao <- numeric()

# em seguida, contamos se em algum momento Q está na frente. Se estiver, então não aconteceu de P lider
contagem_lideranca <- numeric()
for ( i in  1:n) {
  apuracao[i] <- votos[i]
  contagem_lideranca[i] <- sum(apuracao)
}
# se a soma dos votos for positiva, tivemos mais votos 1 que -1. Se for negativa ou igual a zero, P não
# então, basta contar quantos casos aconteceram da soma dos votos apurados é menor ou igual a zero.

if (sum(contagem_lideranca <= 0)) print("p não liderou sempre")
```

[1] “p não liderou sempre”

```
#agora, vamos fazer a simulação mil vezes. Esperamos que aprox. 2% das vezes P lidere sempre.

# simulando k = 1000 vezes

apuracao <- numeric()
contagem_lideranca <- numeric()
freq_lideranca <- numeric() # armazanear se liderou ou não em cada sim k

for (k in 1:1000)  {
  for ( i in  1:n) {
  apuracao[i] <- votos[i]
  contagem_lideranca[i] <- sum(apuracao)
  }
  freq_lideranca[k] <- as.numeric(sum(contagem_lideranca <= 0))
  if( k%% 50 == 0) print(k) # para vermos a evolução da simulação a cada 50 passos
}
```

[1] 50 [1] 100 [1] 150 [1] 200 [1] 250 [1] 300 [1] 350 [1] 400 [1] 450 [1] 500 [1] 550 [1] 600 [1] 650 [1] 700 [1] 750

[1] 800 [1] 850 [1] 900 [1] 950 [1] 1000

```
# se a soma dos votos for positiva, tivemos mais votos 1 que -1. Se for negativa ou igual a zero, P não
# então, basta contar quantos casos aconteceram da soma dos votos apurados é menor ou igual a zero.

print(sum(freq_lideranca)/k)
```

[1] 0.281

```
# 0,281 ou 2,8%. Próximo dovalor verdadeiro de 2%. Se aumentarmos k, irá convergir para o valor verdade
```

Suponha agora um problema diferente. Um dos dois candidatos vai ter a primeira liderança. Após n votos apurados, qual a probabilidade de a contagem ter empatado pela primeira vez? Dito de outro modo, após uma das candidatas assumir a liderança, quanto "tempo" (isto é, após quantos votos apurados) em média devemos esperar até que uma reversão de liderança ocorra (ou pelo menos um empate)?

A intuição que as pessoas têm, talvez a partir da lei dos grandes números, é que deveríamos esperar que, se $p = q = 1/2$, ambas candidatas deveriam liderar por 50% do tempo em uma apuração suficientemente longa (com n votos grande) e deveria haver troca constante de liderança, em vez de uma candidata liderar po um longo tempo. Essa intuição é errada, como iremos mostrar agora por simulação. A leitora interessada na demonstração matemática utilizando apenas análise combinatória (portanto, apenas estatística básica) deve consultar o clássico livro de Feller (1968).