



## HOMEWORK II

NOME COMPLETO: VINICIUS ALEXANDRE GOMES E MATHEUS PARENTE REIS

NUMERO DE MATRICULA: 568594 E 571954

## QUESTÃO 1

Em um restaurante muito frequentado, aproximadamente 70% dos clientes pedem uma sobremesa após o prato principal. Seja  $X$  a variável aleatória que representa o número de clientes que pedem sobremesa em uma amostra aleatória de  $n = 50$  clientes.

1. Determine a função de distribuição de  $X$ .
2. Construa os gráficos da função massa de probabilidade (PMF) e da função distribuição acumulada (CDF) de  $X$ .
3. Calcule o valor esperado, a variância e o desvio padrão dev  $X$ .
4. Calcule a probabilidade de:
  - (a)  $P(X \geq 20)$ .
  - (b)  $P(30 < X < 43)$ .
  - (c)  $P(X = 31)$ .
5. Suponha que o restaurante estoque sobremesas com base na demanda esperada. Como o uso da distribuição de  $X$  poderia ajudar a reduzir desperdício e evitar falta de produtos.
6. Como mudanças em  $p$  (por exemplo, sobremesa se torna mais popular,  $p = 0.8$ ) ou em  $n$  (número de clientes) afetariam a forma e as probabilidades de  $X$ ?

## SOLUÇÃO DA QUESTÃO 1

Na questão 1 é necessário identificar qual seria a melhor distribuição que representa o número de clientes que pedem sobremesa em um restaurante e a partir dessa distribuição realizar os cálculos solicitados.

1. Com base na informação dada de que 70% dos clientes pedem a sobremesa e tomando que a escolha de cada cliente é independente dos demais, encontra-se que a distribuição que mais se encaixa para o caso é uma Binomial, logo  $X \sim Bin(50, 0.7)$ .
2. Construindo os gráficos com R

```
1 # Parametros
2 p <- 0.7
3 n <- 50
4
5 # Plot
6 x <- 0:50
7
8 # PMF
9 plot(x, dbinom(x, size= n, prob= p), type='h',
10       ylab = 'PMF(X)',
11       xlab = 'X')
12
13 # CDF
14 plot(x, pbinom(x, size= n, prob= p), type='h',
15       ylab = 'CDF(X)',
16       xlab = 'X')
```

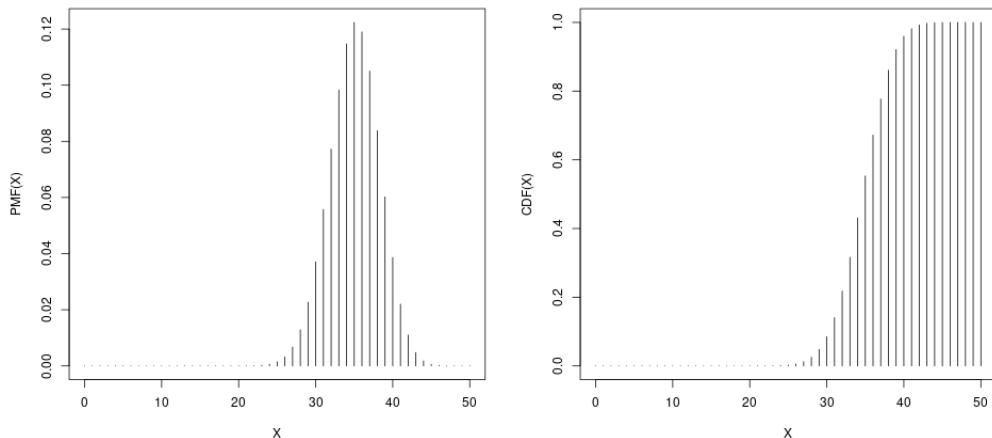


Figura 1: Gráfico Questão 1

A partir do gráfico da PMF (Figura 1) se observa que o numero de clientes que pedem sobremesa se concentra entre 30 e 40 clientes diariamente, o que pode ser

um dado. No mínimo por dia cerca de 25 clientes pedem sobremesa e raramente esse numero passa de 44 clientes.

3. Para conseguir os valores solicitados utiliza-se as seguintes fórmulas:

$$Valor\ Esperado = E[X] = np$$

$$Variância = \sigma^2 = np(1 - p)$$

$$Desvio\ padrao = \sigma = \sqrt{np(1 - p)}$$

Realizando os cálculos:

$$E[X] = 50 \cdot 0.7 = 35\ Clientes$$

$$\sigma^2 = 50 \cdot 0.7 \cdot (1 - 0.7) = 35 \cdot 0.3 = 10.5$$

$$\sigma = \sqrt{10.5} = 3.24$$

4. (a)  $P(X \geq 20)$

Desenvolvendo o cálculo:

$$P(X \geq 20) = 1 - P(X < 20) = 1 - [P(X = 19) + P(X = 18) + \dots + P(X = 0)]$$

Este resultado é obtido calculando a probabilidade de  $P(X \geq 20)$  não acontecer e a subtraindo da probabilidade total, dessa forma diminui-se o número de casos individuais a serem calculados.

Por conta da extensão do cálculo, sera apresentado apenas o resultado obtido ao utilizar o R.

```

1 #Parametros
2 p <- 0.7
3 n <- 50
4
5 #Calculo
6 s <- 1 - pbisec(19, size = n, prob = p)
7 #A função pbisec permite calcular a probabilidade
    #cumulativa de  $P(X=0)$  até  $P(X=19)$  diretamente
8 s
9
10 #Saída
11 [1] 0.9999972

```

- (b)  $P(30 < X < 43)$  Desenvolvendo o cálculo:

$$P(30 < X < 43) = P(X < 43) - P(X \leq 30) = P(X \leq 42) - P(X \leq 30)$$

Dessa maneira calcula-se o valor em R utilizando a função de probabilidade cumulativa.

```

1 #Parametros
2 p <- 0.7
3 n <- 50
4
5 #Calculo
6 s <- pbinom(42, size = n, prob = p) - pbinom(30, size = n,
      prob = p)
7
8 s
9
10 #Saída
11 [1] 0.9079332

```

Alternativamente pode-se realizar o cálculo da seguinte forma:

$$P(30 < X < 43) = P(X = 31) + P(X = 32) + P(X = 33) + \dots + P(X = 42)$$

(c)  $P(X = 31)$  Desenvolvendo o cálculo:

$$P(X = 31) = \binom{50}{31} (0.7)^{31} (0.3)^{19} = \frac{50!}{19!31!} (0.7)^{31} (0.3)^{19} = 0.05575$$

Agora calculando o valor com R:

```

1 #Parametros
2 p <- 0.7
3 n <- 50
4
5 #Calculo
6 s <- dbinom(31, size = n, prob = p)
7
8 s
9
10 #Saída
11 [1] 0.05575728

```

Foram obtidos resultados semelhantes como esperado.

5. Ajudaria a ter uma base de quantas sobremesas precisariam ser estocadas em média, permitindo que tanto a falta de estoque quanto o desperdício sejam mitigados, considerando que a média de clientes que pedem sobremesa não se altere. Também sendo possível levar em consideração o desvio padrão, sendo um valor base diante da média, de quantas sobremesas irão faltar ou ser desperdiçadas.
6. Um aumento na popularidade da sobremesa faria com que a distribuição de X se tornasse mais densa para valores de x mais altos (deslocando os gráficos da PMF e CDF para a direita, Figura 1), aumentando também sua média e reduzindo seu desvio padrão e variância. Realisticamente, haveria um aumento geral da quantidade de clientes que pedem sobremesas e, em relação ao item 5, seria necessário um estoque maior.

## QUESTÃO 2

Um site realiza uma pesquisa online e oferece uma recompensa a um usuário selecionado aleatoriamente que responde a uma série de perguntas. Cada um dos 10 milhões de visitantes diários tem, independentemente, probabilidade  $p = 10^{-7}$  de ganhar a recompensa.

1. Encontre uma aproximação simples e adequada para a função de massa de probabilidade (PMF) do número de vencedores em um dia,  $X$ . Justifique claramente se essa aproximação é apropriada para os valores dados de  $n$  and  $p$ .
2. Calcule o valor esperado,  $E[X]$ , e a variância  $\text{Var}(X)$ , usando tanto a distribuição exata quanto a aproximada. Comente sobre a semelhança entre os resultados.
3. Suponha que você ganhe a recompensa, mas que possa haver outros vencedores. Seja  $W \sim \text{Pois}(1)$  o número de vencedores além de você. Se houver vários vencedores, o prêmio é sorteado aleatoriamente entre todos eles. Encontre a probabilidade de que você realmente receba o prêmio.
4. Gere um grande número de simulações diárias para o número de vencedores. Crie uma comparação visual entre os resultados empíricos e a aproximação considerada no item 1. Descreva brevemente o que a visualização indica sobre a qualidade da aproximação.

## SOLUÇÃO DA QUESTÃO 2

A questão 2 pede que se analise um site que possui um sistema de sorteio que elege um ganhador seguindo uma distribuição Binomial com 10 milhões de visitantes, e uma probabilidade de  $10^{-7}$  de receberem a recompensa individualmente.

1. É possível simplificar e aproximar o valor da PMF esperado com uma distribuição de Poisson,  $X \sim \text{Poisson}(\lambda)$ , pois no caso há um numero de amostras muito alto com uma probabilidade muito pequena.

$$\lambda = n \cdot p = 10^{-7} \cdot 10^7 = 1$$

PMF aproximada:

$$P(X = k) = \frac{e^{-\lambda} \lambda^k}{k!} = \frac{e^{-1} 1^k}{k!} = \frac{e^{-1}}{k!}$$

Gráfico em R:

```

1 #Parametros
2 p <- 10^-7
3 n <- 10^7
4
5 #Plot
6 x <- 0:10
7
8 #PMF
9 plot(x, dbinom(x, size= n, prob= p), type='h',
10      ylab = 'PMF(X) Exata',
11      xlab = 'X')
12
13 #PMF
14 plot(x, dpois(x, 1), type='h',
15       ylab = 'PMF(X) Aproximada',
16       xlab = 'X')
```

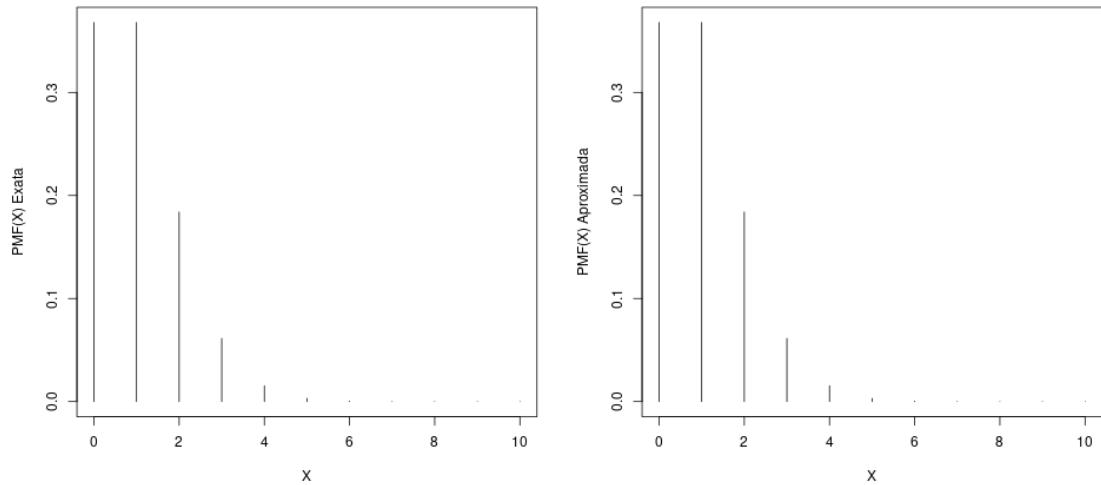


Figura 2: Gráfico Exata vs Aproximada

Pode-se observar que os gráficos (Figura 2) condizem aos mesmos valores. Sendo válida a aproximação como esperado.

2. Realizando primeiro os cálculos para distribuição exata e então para a aproximada:

(a) Distribuição Exata

$$E[X] = n \cdot p = 10^{-7} \cdot 10^7 = 1$$

$$Var(X) = n \cdot p \cdot (1 - p) = 10^{-7} \cdot 10^7 \cdot (1 - 10^{-7}) \approx 10^{-7} \cdot 10^7 = 1$$

(b) Distribuição Aproximada

$$E[X] = Var(X) = \lambda = 1$$

(c) Conclusão

Encontra-se que para ambas distribuições os resultados são iguais e de valor 1.

Realizando os cálculos com R

```
1 #Parametros
2 p <- 10^-7
3 n <- 10^7
4
5 #Calculo
6 e_exato <- p*n
7 variancia_exato <- p*n*(1-p)
8
9 lambda <- n*p
10 e_aprox <- lambda
11 variancia_aprox <- lambda
12
13 e_exato
14 variancia_exato
15
16 e_aprox
17 variancia_aprox
18
19 #Saida
20 [1] 1
21 [1] 0.9999999
22 [1] 1
23 [1] 1
```

Os valores obtidos são condizentes com os valores calculados diretamente.

3. Sendo Z o evento de obter o premio entre outros W candidatos. Considera-se que todos possuam a mesma chance de ganhar, com isso basta dividir a probabilidade total pelo numero de vencedores, dado por  $W+1$ .

$$P(Z|W) = \frac{1}{W+1}$$

Dessa forma, o valor de Z depende de quantos vencedores forem selecionados. Para conseguir uma probabilidade geral utiliza-se uma média de quantos candidatos são selecionados.

$$\begin{aligned} P(Z) &= E\left[\frac{1}{W+1}\right] = \sum_{k=0}^{\infty} \frac{1}{k+1} P(W=k) = \\ &= \sum_{k=0}^{\infty} \frac{1}{k+1} \cdot \frac{e^{-1}1^k}{k!} = e^{-1} \sum_{k=0}^{\infty} \frac{1}{k+1} \cdot \frac{1}{k!} = \\ &= e^{-1} \sum_{k=0}^{\infty} \frac{1}{(k+1)!} = e^{-1} \sum_{k=1}^{\infty} \frac{1}{k!} = e^{-1} \left[ \left( \sum_{k=0}^{\infty} \frac{1}{k!} \right) - 1 \right] = \\ &= e^{-1}(e-1) = 1 - e^{-1} = 0.632 \end{aligned}$$

Logo, a chance de ganhar o prêmio é de 63,2%. Em R:

```

1 #Parametros
2 lamb <- 1
3 k <- 0:20
4
5 #Funcoes
6 pmf <- dpois(k, lambda = lamb) #pmf de poisson(1)
7
8 P_ganhar <- sum(1 / (k + 1) * pmf) # 1/(W + 1)
9
10 P_ganhar
11
12 #Saida
13 [1] 0.6321206

```

Resultando no valor esperado.

4. A partir da geração de 100000 simulações os dados obtidos foram:

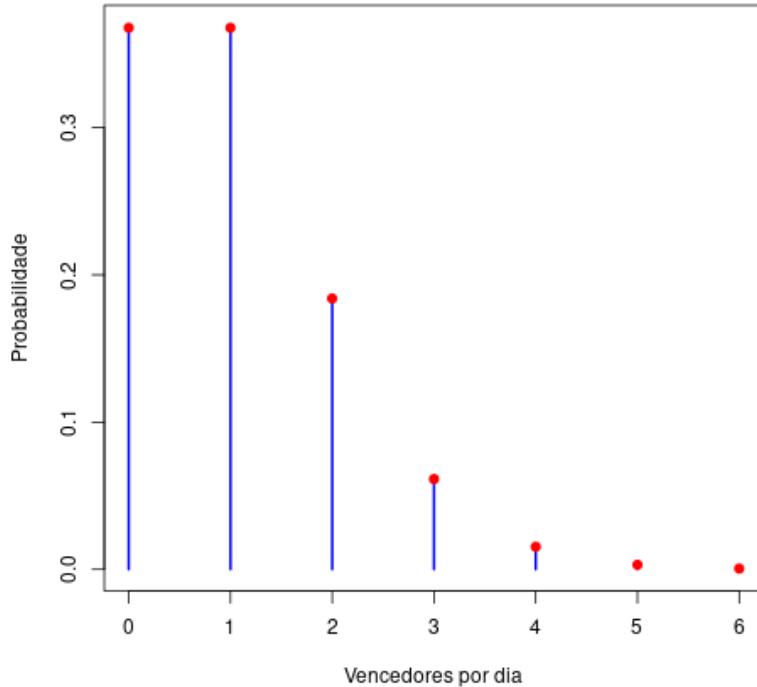


Figura 3: Simulacao Questao 2

No gráfico da Figura 3, as retas azuis representam a probabilidade do número de vencedores por dia obtidos das simulações, enquanto os pontos vermelhos representam os valores esperados pela aproximação com Poisson. A partir do gráfico é possível se observar que a aproximação é suficiente para ter valores confiáveis para um grande número de amostras.

Demonstrando a importância da quantidade de simulações nessa analise, o gráfico da Figura 4 possui apenas 10 simulações.

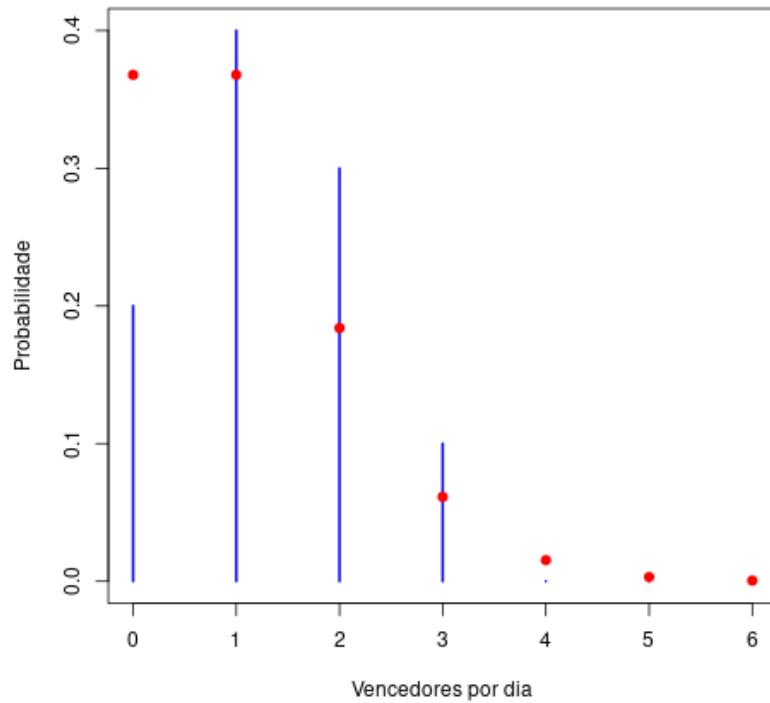


Figura 4: Simulacao 2 Questao 2

Comparando os gráficos (Figuras 3 e 4) se torna clara a importância do numero de simulações para a precisão da aproximação, onde apenas 10 simulações não apresentam uma aproximação satisfatória, resultando em distorções em certas partes do gráfico que se desviam do valores esperados.

Código Utilizado em R:

```
1 #Seed da Simulacao
2 set.seed(123)
3
4 #Parametros
5 N <- 10^1 #Numero de Simulacoes
6 n <- 10^7
7 p <- 10^-7
8 lambda <- n * p
9
10 #Simulacao da binomial
11 X <- rbinom(N, n, p)
12
13 #Probabilidades da Binomial, apenas resultados de 0 a 6, os
   demais se tornam irrelevantes para a analise
14 k <- 0:6
15 p_bin <- table(factor(X[X <= 6], levels = k))
16 p_bin <- p_bin / sum(p_bin)
17
18 #Grafico da Binomial
19 plot(k, p_bin,
20       type = "h",
21       lwd = 2,
22       col = "blue",
23       ylim = c(0, max(p_bin, dpois(k, lambda))),
24       xlab = "Vencedores por dia",
25       ylab = "Probabilidade",
26       yaxt = "n")
27 axis(2)
28
29 # pontos da Poisson
30 points(k, dpois(k, lambda),
31          pch = 19,
32          col = "red")
```

## QUESTÃO 3

Você é responsável por monitorar a temperatura de uma CPU multicore em uma unidade de processamento embarcada. Sob carga normal, a temperatura da CPU apresenta flutuações devido a mudanças na carga de trabalho, nas condições ambientais e na eficiência do sistema de resfriamento. Testes mostram que a temperatura em regime estacionário da CPU segue uma distribuição normal com temperatura média  $\mu = 62^\circ\text{C}$  e desvio-padrão  $\sigma = 3,5^\circ\text{C}$ . Sua tarefa é simular medições de temperatura da CPU e analisar suas propriedades estatísticas.

1. Crie uma função que gere valores com distribuição normal usando a transformação de Box-Muller<sup>1</sup>, a partir de entradas aleatórias uniformes. Especificamente:

- (a) Gere duas variáveis aleatórias uniformes independentes:  $U_1, U_2 \sim \text{Unif}(0, 1)$ .
- (b) Calcule dois valores normais padrão usando as fórmulas de Box-Muller:

$$Z_1 = \sqrt{-2 \ln(U_1)} \cos(2\pi U_2), \quad Z_2 = \sqrt{-2 \ln(U_1)} \sin(2\pi U_2)$$

$Z_1$  e  $Z_2$  são variáveis aleatórias independentes com distribuição normal padrão. Concatene-as para formar um vetor  $Z$  de valores normais padrão.

- (c) Converta cada valor normal padrão para a distribuição de temperatura da CPU:

$$T = 62 + 3.5 Z$$

2. Use seu gerador de números aleatórios para gerar 1.000 medições de temperatura da CPU.

Gere mais 1.000 valores de temperatura utilizando o gerador de números aleatórios normal embutido do R, com a mesma média e desvio-padrão.

3. Para ambos os conjuntos de dados simulados, calcule:

- (a) Média amostral.
- (b) Desvio-padrão amostral.
- (c) Temperatura mínima e máxima observada.
- (d) Probabilidade empírica e teórica  $P(T > 68)$ .
- (e) Probabilidade empírica e teórica  $P(60 < T < 65)$ .
- (f) Probabilidade teórica  $P(T > 75)$ .

Algum dos conjuntos de dados simulados (1.000 amostras) contém valores acima de  $75^\circ\text{C}$ ? Caso não, explique por que eventos raros requerem tamanhos de amostra grandes para serem observados.

4. Visualize os resultados criando:

---

<sup>1</sup> [https://en.wikipedia.org/wiki/Box-Muller\\_transform](https://en.wikipedia.org/wiki/Box-Muller_transform)

- (a) Um histograma das temperaturas simuladas da CPU (pode plotar os dois conjuntos de dados separadamente ou sobrepostos).
- (b) A função densidade de probabilidade (PDF) normal teórica (média 62 °C, desvio padrão 3,5 °C) sobreposta ao histograma.
5. Discuta seus resultados respondendo às seguintes perguntas: As distribuições empíricas da temperatura da CPU se assemelham à curva normal teórica? Quão próximas estão a média amostral e o desvio-padrão amostral dos valores esperados 62 °C e 3,5 °C? Há diferenças perceptíveis entre o conjunto de dados gerado com seu RNG manual e o produzido pelo RNG embutido do R? Como essa simulação pode ajudar na avaliação de estratégias de resfriamento ou de escalonamento dinâmico de clock? Por que geradores de números aleatórios uniformes são a base dos sistemas de RNG?

### SOLUÇÃO DA QUESTÃO 3

1. Esse item consiste na implementação prática da função que será utilizada nos próximos itens para analisar o comportamento estatístico simulado desse sistema. Portanto, os itens a seguir irão descrever essa construção em R, acompanhados da explicação:

- (a) Geração das duas variáveis aleatórias pertencentes ao intervalo (0, 1]:

```

1 U1 <- runif(n)
2 U2 <- runif(n)
```

A função `runif(n)` gera  $n$  variáveis aleatórias distribuídas uniformemente, ou seja,  $U \sim \text{Unif}(0,1)$ . Para que haja essa transformação, U1 define o Raio e U2 o ângulo das coordenadas polares da variável Z.

- (b) Cálculo dos valores padronizados conforme as Fórmulas de Box-Muller:

```

1 Z1 <- sqrt(-2 * log(U1)) * cos(2 * pi * U2)
2 Z2 <- sqrt(-2 * log(U1)) * sin(2 * pi * U2)
3
4 Z <- c(Z1, Z2)
```

Segundo Box-Muller, a distinção dos valores  $Z_1$  e  $Z_2$  reside na projeção em eixos ortogonais, assegurando valores estatisticamente independentes e igualmente válidos para a simulação. Uma vez que cada entrada  $U_1$  e  $U_2$  geram pares de amostras, devemos salvá-los em uma lista  $Z$  por meio da função `c(a,b,...)` que combina(ou concatena) esses valores, criando um **vetor**.

- (c) Convertendo o valor padronizado para Temperatura(C°)

```

1 T <- 62 + (3.5 * Z)
2
3 return(T)
```

Devido a propriedade de vetorização da linguagem, esse trecho de código é capaz de executar a fórmula aritmética em todos os elementos de  $Z$ , armazenando-os em  $T$ . Logo em seguida, esse **vetor** é retornado como a saída da função.

Ao final, implementa-se a seguinte função:

```

1 random_box_muller <- function(n) {
2   # (a) Gere duas variaveis aleatorias uniformes
3   # independentes:
4   U1 <- runif(n)
5   U2 <- runif(n)
6
7   # (b) Calcule dois valores normais padrao usando as
8   # formulas de Box-Muller
9   Z1 <- sqrt(-2 * log(U1)) * cos(2 * pi * U2)
10  Z2 <- sqrt(-2 * log(U1)) * sin(2 * pi * U2)
11
12  Z <- c(Z1, Z2)
13
14  # (c) Converta cada valor normal padrao para a
15  # distribuicao de temperatura da CPU
16  T <- 62 + (3.5 * Z)
17
18  return(T)
19}
20
21 if (sys.nframe() == 0) {
22   amostras <- random_box_muller(5)
23   print(amostras)
24}
```

Por fim, incluiu-se uma estrutura condicional baseada no método `sys.frame()`. Em linhas gerais, essa verificação indica se o código foi executado como principal ou foi referenciado em outro código por meio da função `source()`. Portanto, tem-se a saída:

```

1 [1] 63.46439 59.27323 64.17676 60.03350 60.15860 58.70230
2      59.28586 65.15255
2 [9] 66.19021 67.84876
```

À primeira vista, valores relativamente plausíveis, uma vez que estão distribuídos em torno da média teórica ( $\mu = 62$ ), respeitando a dispersão esperada seguindo o desvio padrão ( $\sigma = 3.5$ ).

2. Com a função Box-Muller já implementada, este item requer a geração de amostras para validação estatística.

A fim de obter um total de 1000 amostras, é necessário passar como parâmetro o número 500, uma vez que esses 500 pares resultam em um total de 1000 amostras.

Ademais, como objeto de estudo, comparando com a metodologia Box-Muller, utiliza-se a função nativa da linguagem que também gera uma distribuição aleatória desses valores. O nome da função é `rnorm()` e recebe como parâmetros o número de amostras, a média e desvio padrão da distribuição normal desejada.

Dessa forma:

```

1 source("1.R")
2
3 # 500 pares de numeros aleatorios nos retorna um total de 1000
4 # amostras, utilizando a funcao ja implementada no Item 1:
5 amostras_autoral <- random_box_muller(500)
6
7 # Gerando os numeros aleatoriamente com a funcao built-in do R
8 amostras_nativo <- rnorm(n = 1000, mean = 62, sd = 3.5)
9
10 if (sys.nframe() == 0) {
11   amostras_autoral
12   amostras_nativo
13 }
```

Reiterando: para utilizar a função de Box-Muller implementada anteriormente, utiliza-se a função `source()`, sendo o parâmetro o nome do código presente no item 1 como uma *string*. Ademais, a saída:

```

1 [1] 57.05889 62.68565 63.27337 65.70774 62.27397 61.43627
2           63.15186 64.95026
3 [9] 63.02543 60.47394 59.78331 59.93015 58.70922 57.16636
4           60.82682 62.12003
5 .
6 .
7 [985] 66.83025 63.61659 59.03819 56.53984 57.83500 62.00132
8           59.06200 62.00703
9 [993] 62.57725 66.14356 58.67293 65.77378 59.88360 66.32880
10          60.89094 61.05035
```

3. Em posse de dois conjuntos de valores aleatórios, realiza-se uma análise das suas características em comparação com o teórico. Desta forma, valida-se o comportamento estatístico tanto da metodologia, quanto da função nativa.

Segue abaixo a implementação do código com o cálculo das métricas solicitadas que, por serem de simples interpretação e aquisição, podem ser apresentadas a baixo:

```

1 # (a) Media amostral
2 media_autoral <- mean(amostras_autoral)
3 media_nativo <- mean(amostras_nativo)
4
5 # (b) Desvio-padrão amostral
6 sd_autoral <- sd(amostras_autoral)
7 sd_nativo <- sd(amostras_nativo)
8
9 # (c) Temperatura mínima e máxima observada
10 min_max_autoral <- range(amostras_autoral)
11 min_max_nativo <- range(amostras_nativo)

```

Por sua vez, a saída:

```

1 [1] ----- Item (a) (Medias) -----
2 [1] Box-Muller: 62.1402 | Nativo: 62.0063
3 [1] ----- Item (b) (Desvios) -----
4 [1] Box-Muller: 3.5339 | Nativo: 3.5029
5 [1] ----- Item (c) (Mins e Maxs)-----
6 [1] Box-Muller: Min 51.2563 / Max 71.2040
7 [1] Nativo: Min 50.4321 / Max 72.1941

```

Nota-se um comportamento bem semelhante dos dados provindos de ambas as amostras.

Ademais, os item (d), (e) e (f) aprofundam-se no cálculo das possibilidades, buscando comparar o comportamento empírico(observado) com o teórico(esperado). Para o cálculo da probabilidade na curva normal, utilizou-se `pnorm()`, fundamentadas nas propriedades da CDF, onde  $F(x) = P(X \leq x)$ :

- **Probabilidade do Evento Complementar:** Para calcular a probabilidade de a temperatura ser maior que um valor, utiliza-se o fato de que a soma das probabilidades é 1. Logo:

$$P(X > k) = 1 - P(X \leq k)$$

- **Probabilidade em um Intervalo:** Para calcular a probabilidade da temperatura pertencente a um intervalo, subtraí-se a área acumulada desses valores:

$$P(a < X < b) = P(X \leq b) - P(X \leq a)$$

Para obter o valor da probabilidade teórica, utiliza-se a função nativa `pnorm()`. Esta função calcula a área sob a curva normal, recebendo como parâmetros o valor a qual se procura a probabilidade acumulada, a média e o desvio padrão.

Por outro lado, a fim de calcular a probabilidade empírica, analisa-se o vetor de dados gerados. O algoritmo contabiliza iterativamente quantas amostras satisfazem a condição estipulada e divide essa soma pelo tamanho total da amostra, resultando em sua probabilidade. Como os dados são gerados aleatoriamente, cada execução ocasiona em diferentes valores. Apesar disso, os valores demonstraram-se bem plausíveis.

Segue o código:

```

1 # (d) Probabilidade teorica e empirica de P(T > 68):
2 # Te rica: Formula matematica
3 prob_teorica_d <- 1 - pnorm(68, mean = 62, sd = 3.5)
4
5 # Empirica: Quantos valores no meu vetor sao > 68?
6 prob_empirica_d <- mean(amostras_autoral > 68)
7
8 # (e) Probabilidade empirica e teorica P(60 < T < 65):
9 # Teorica
10 prob_teorica_e <- pnorm(65, 62, 3.5) - pnorm(60, 62, 3.5)
11
12 # Empirica: Quantos estao entre 60 e 65?
13 prob_empirica_e <- mean(amostras_autoral > 60 & amostras_
14 autoral < 65)
15
16 # (f) Probabilidade teorica P (T > 75) e verificacao de
17 # existencia
18 prob_teorica_f <- 1 - pnorm(75, 62, 3.5)
19
20 # Algum conjunto contem valores acima de 75?
21 existe_acima_75 <- any(amostras_autoral > 75)

```

E como saída:

```

1 [1] ----- Item (d) -----
2 [1] Empirica: 0.0430 (4.30%)
3 [1] Teorica: 0.0432 (4.32%)
4 [1] ----- Item (e)-----
5 [1] Empirica: 0.5400
6 [1] Teorica: 0.5205
7 [1] ----- Item (f) -----
8 [1] Probabilidade Teorica: 0.000102
9 [1] Algum valor > 75 nao foi gerado.

```

Ademais, detalhando o item (f), tal número não foi gerado devido a sua baixa probabilidade, certa de 1 a cada 10.000 tentativas. Tal probabilidade se fundamenta na distância do número 75 até a média 62, cerca de 4 desvios padrões de distância.

4. Utilizam-se as funções nativas de visualização do R para a construção dos gráficos. A função `hist()` é responsável pelos histogramas, sendo configurada com o parâmetro `prob = TRUE` para exibir a densidade de probabilidade e `breaks = 50` para aumentar a resolução das barras, dessa forma, facilitando na visualização do comportamento das amostras.

Para representar a distribuição teórica, emprega-se a função `curve()` em conjunto com `dnorm()`, que desenha a curva teórica sobreposta ao gráfico existente, permitindo a comparação.

O armazenamento do resultado é gerenciado pela função `pdf()`, que redireciona a saída gráfica para o diretório relativo `../graficos/`. O processo é concluído com o comando `dev.off()`, que finaliza a escrita do arquivo:

```

1 source("2.R")
2 pdf("../graficos/grafico_questao_3.pdf", width = 12, height =
   10)
3 par(mfrow = c(2, 2))
4
5 # (a) Um histograma das temperaturas simuladas da CPU (pode
   plotar os dois con-
6 # juntos de dados separadamente ou sobrepostos)
7 # Grafico 1: Histograma Box-Muller
8 hist(amostras_autoral,
9       prob = TRUE,
10      breaks = 50,
11      main = "(a) □Box-Muller",
12      xlab = "Temperatura □(C)",
13      ylab = "Densidade",
14      col = "lightblue",
15      border = "white",
16      ylim = c(0, 0.12))
17
18 # Grafico 2: Histrograma R Nativo
19 hist(amostras_nativo,
20       prob = TRUE,
21       breaks = 50,
22       main = "(a) □R □Nativo",
23       xlab = "Temperatura □C)",
24       ylab = "Densidade",
25       col = "lightgreen",
26       border = "white",
27       ylim = c(0, 0.12))
28
29 # (b) A funcao densidade de probabilidade normal teorica
   sobreposta ao histograma
30 # Grafico 3: Box-Muller + Curva
31 hist(amostras_autoral,
32       prob = TRUE,
33       breaks = 50,
```

```

34     main = "(b) Box-Muller+Teorico",
35     xlab = "Temperatura(C)",
36     ylab = "Densidade",
37     col = "lightblue",
38     border = "white",
39     ylim = c(0, 0.12))
40
41 curve(dnorm(x, mean = 62, sd = 3.5),
42        add = TRUE,
43        col = "red",
44        lwd = 2)
45 legend("topright", legend = "Teorico", col = "red", lwd = 2,
46        bty = "n")
47
48 # Grafico 4: R Nativo + Curva
49 hist(amostras_nativo,
50       prob = TRUE,
51       breaks = 50,
52       main = "(b) R+Teorico",
53       xlab = "Temperatura(C)",
54       ylab = "Densidade",
55       col = "lightgreen",
56       border = "white",
57       ylim = c(0, 0.12))
58 curve(dnorm(x, mean = 62, sd = 3.5),
59        add = TRUE,
60        col = "red",
61        lwd = 2)
62 legend("topright", legend = "Teorico", col = "red", lwd = 2,
63        bty = "n")
64 dev.off()

```

E foram gerados os seguintes gráficos: Percebe-se que ambos os conjuntos de dados, gerados tanto pelo algoritmo de Box-Muller quanto pela função nativa, apresentaram um comportamento estatístico semelhante. Ao serem comparados com a curva teórica da distribuição normal, essa análise visual validou a eficácia da implementação da metodologia Box-Muller.

A análise dos dados gerados permite responder às questões fundamentais sobre a validade e a aplicabilidade da simulação:

- **Aderência à Curva Normal Teórica:** Como discutido anteriormente, essa metodologia provou, visualmente, aderir satisfatoriamente a curva teórica, validando sua usabilidade apesar das limitações estocásticas de um conjunto finito.
- **Proximidade dos Parâmetros Estatísticos:** Calculado anteriormente, os valores de média e desvio padrão calculados foram próximos aos dados esperados

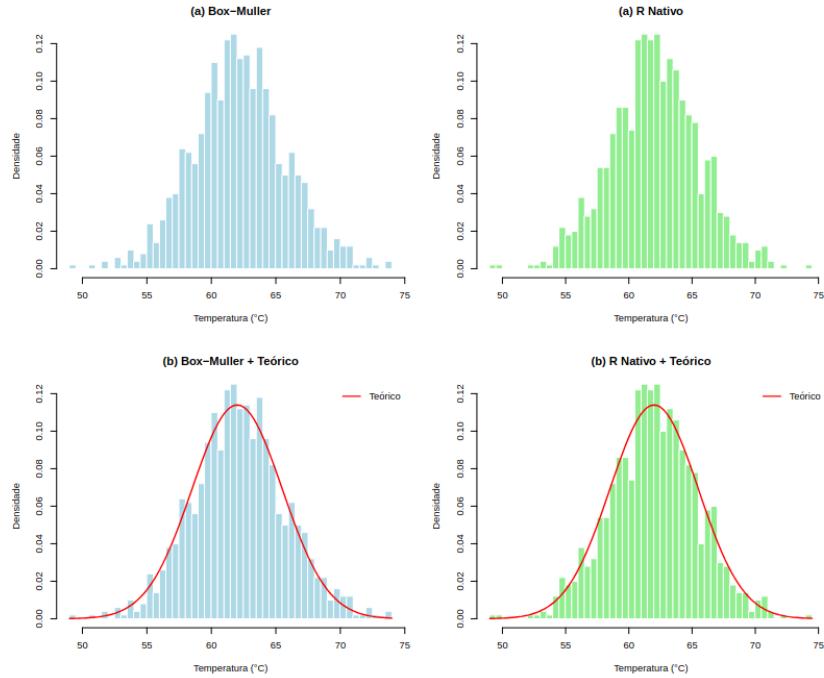


Figura 5: Simulacao Questao 3

$$(\mu = 62^\circ\text{C} \text{ e } \sigma = 3.5^\circ\text{C})$$

- **Comparação entre RNG Manual e Nativo:** Não foram observadas diferenças estatísticas entre o conjunto gerado via Box-Muller e o produzido pela função `rnorm()` do R.
- **Aplicação em Estratégias de Resfriamento e Escalonamento:** Tendo essa simulação, é possível prever e dimensionar os dissipadores de calor não apenas para temperatura média, mas sim com base nos desvios padrões em torno da média, garantindo uma taxa de sucesso com base na normal. Ademais, o próprio sistema consegue prever esse superaquecimento, promovendo medidas de segurança como a redução do clock.
- **O Papel dos Geradores Uniformes:** São a forma mais básica de aleatoriedade. Tendo essa função, aplicações mais complexas conseguem ser formadas a partir dela, aplicando esses intervalos aleatórios em locais estratégicos. No caso do Box-Muller,  $U_1$  e  $U_2$  servem, respectivamente, para indicar em que região do Raio e do Ângulo aquelas variáveis estão inseridas nas coordenadas polares.