Instituto Superior Técnico

Complementos de Probabilidade e Estatística

2º Semestre - 2019/2020

Licenciatura Em Matemática Aplicada e Computação

Projeto em R de Complementos de Probabilidade e Estatística

Grupo:
Alexandre Silva - 90004
Leandro Duarte - 93112
Diogo Santos - 93635
Vasco Pearson - 97015

Professora:
Isabel Rodrigues

29 de Maio de 2020

Conteúdo

1 Pergunta 1

Neste exercício consideramos uma variável aleatória $X \sim Poi(\lambda)$.

1.1 Alínea a)

Pretendemos gerar observações da v.a $X \sim Poi(\lambda)$ com base em três algoritmos diferentes:

1. Forma recursiva da função de probabilidade da Poisson;
2. $\{Y_i\}$ é uma sucessão de v.a's i.i.d a $Y \sim Exp(\lambda)$ e $X = max\{n : \sum_{j=1}^{n} Y_j \leq 1\}$ então $X \sim Poi(\lambda)$;
3. Gerador do programa R;

Para o primeiro caso, sabemos que a função de probabilidade de uma *Poisson* com parâmetro λ é dada por:

$$p_k = P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!}, \qquad k = 0, 1, 2, \dots \ , \ \ \lambda > 0$$

Logo podemos obter recursivamente p_{k+1} em função de p_k, pois:

$$\frac{p_{k+1}}{p_k} = \frac{e^{-\lambda}\lambda^{k+1}}{(k+1)!}\frac{k!}{e^{-\lambda}\lambda^k} = \frac{\lambda}{k+1}$$

Pelo que $p_{k+1} = \frac{\lambda}{k+1}p_k$.

O algoritmo baseado nesta recursividade torna-se agora simples de se gerar, à luz do Teorema da Tranformação Inversa:

1. Gerar um número pseudo-aleatório de $U \sim Unif(0, 1)$: u.
2. Iniciar as variáveis auxiliares $k = 0$, $p = e^{-\lambda}$ (corresponde a p_0) e $F = p$ (inicialização da função de distribuição);
3. Enquanto $u > F$, fazer $p = \frac{\lambda p}{k+1}$ (passar para p_{k+1}) , $F = F + p$ e $k = k + 1$. Quando $u \leq F$ retornar k.

De modo a se gerarem as 10000 observações pretendidas, recorreu-se à função *runif(10000,0,1)* que gera 10000 observações pseudo-aleatórias de uma $U \sim Unif(0, 1)$. Para percorrermos o ciclo descrito nos pontos 2 e 3, criou-se a função *PoissonRecursive* que recebe como parâmetro de entrada o valor u pseudo-aleatório e executa-o, com recurso à função *While*. Por fim, com o objetivo de aplicarmos a função às 10000 observações, recorremos à função *lapply* que aplica a função a cada elemento do vector em que se guardaram as uniformes geradas. Convém notar que esta função retorna uma lista de elementos e não um vector, pelo que se usou a função *unlist* do R para voltarmos a ter um vector. Os dados ficaram armazenados no vector *GeneratePoissonRecursive*.

Na segunda opção, sabemos que $X \sim Poi(\lambda)$ e $X = max\{n : \sum_{j=1}^{n} Y_j \leq 1\}$, onde $\{Y_i\}$ é uma sucessão de v.a's i.i.d a $Y \sim Exp(\lambda)$, ou seja, podemos interpretar a variável aleatória X como o máximo de números de variáveis exponenciais cuja soma não exceda 1.

Considerando $U_i \overset{i.i.d}{\sim} U(0,1)$ e sabendo que para gerar uma v.a. $Y \sim Exp(\lambda)$ pelo Teorema da Tranformação Inversa basta realizar $x = -\frac{\ln(u)}{\lambda}$, conseguimos reescrever X como:

$$X = max\{n : \sum_{j=1}^{n} -\frac{\ln(U_j)}{\lambda} \leq 1\}$$

$$X = max\{n : \sum_{j=1}^{n} \ln(U_j) \geq -\lambda\}$$

$$X = max\{n : \ln(\prod_{j=1}^{n} U_j) \geq -\lambda\}$$

$$X = max\{n : \prod_{j=1}^{n} U_j \geq e^{-\lambda}\}$$

Podemos então escrever o algoritmo em função da última equação como se segue:

1. Iniciar as variáveis auxiliares $i = 0$, $p = e^{-\lambda}$ e $m = 1$;
2. Se $m \geq p$, gerar um número pseudo-aleatório de $U \sim Unif(0,1)$: u_i, fazer $m \mapsto m \cdot u_i$ e $i \mapsto i + 1$.
3. Quando a condição não se verificar, retornar $i - 1$.

Este algoritmo, que gera uma observação da $Poi(3)$ está implementado no R na função *PoissonSum*. Para se gerar as 10000 observações recorreu-se a um *dummy vector*, ou seja, criou-se um vetor de zeros de tamanho 10000, para se usar a função *lapply* como no algoritmo anterior. Os dados ficaram guardados no vector *GeneratePoissonSum*.

Por último é pedido para usar o gerador do programa R, com recurso à função *dpois*, que recebe como input os valores do domínio que queremos representar (definimos de 0 a 25) e o outro *input* é o parâmetro da distribuição, neste caso $\lambda = 3$.

Após os algoritmos estarem criados, procedeu-se à comparação do tempo de geração das 3 alternativas, com recurso à função *Sys.time* do *R*. Os resultados obtidos encontram-se na tabela apresentada no início da próxima página. É de notar que os valores acima de $x = 11$ nunca foram gerados, pois $P(X > 11) = 1 - P(X \leq 11) = 7.138 \cdot 10^{-5}$, um valor muito pequeno e como tal é muito improvável ser gerado.

Algoritmo	**Recursividade**	**Soma de Exp.**	**Gerador do R**
T. de geração 1 (s)	0.1825118	0.2900121	0.03487492
T. de geração 2 (s)	0.1755278	0.2685809	0.02992296
T. de geração 3 (s)	0.1713719	0.2654449	0.02890801
T. de geração 4 (s)	0.1685488	0.266283	0.03019905
T. de geração 5 (s)	0.1685481	0.330008	0.03034186
Média (s)	0.1733018	0.28406578	0.03084936

Realizámos a mesma operação de calcularmos os tempos de geração das observações 5 vezes pois dependendo de como a máquina de cada utilizador se comporta no momento em que o algoritmo é executado influência o resultado e, como tal, decidimos que a média de algumas observações do tempo de execução era um melhor indicativo do tempo de geração das observações de cada algoritmo.

Podemos concluir, a partir dos valores apresentados na última linha da tabela, que o algoritmo que o R apresenta é substancialmente mais rápido que os outros dois, o que era de expectar, dado que linguagens como esta procuram sempre a solução mais eficiente. A função *rpois* baseia-se não em recursividade ou em soma de exponenciais, mas sim em gerar um desvio normal apropriado, transformá-lo igualmente de uma maneira apropriada num inteiro e aplicar uma correção de probabilidade baixa de modo a gerar a distribuição de Poisson. Esta questão foi levantada por *D.E. Knuth*, respondida em 1982 por *J.H. Ahrens* e *U. Dieter* e o algoritmo revelou-se muito mais rápido do que os métodos normalmente utilizados.

Também era expectável que o algoritmo 1 (recursividade) fosse mais eficiente que o algoritmo 2 (soma de exponenciais), visto que no segundo algoritmo de modo a gerar um valor pseudo-aleatório da Poisson é necessário gerar vários valores pseudo-aleatórios da Uniforme, e então o ciclo demora mais tempo a terminar.

É-nos também pedido para calcular a média, os quartis e a $P(2 < X < 8)$ para as 3 amostras geradas, e comparar com o expectável teórico da $Poi(3)$. O comando *summary* permite-nos obter a média e os quartis dos vetores que guardam as amostras, e para o caso teórico sabemos que se $X \sim Poi(\lambda)$ então $E(X) = \lambda$ logo neste caso $E(X) = 3$. Para calcularmos os quartis teoricamente, relembramos que um quantil de probabilidade p $(0 < p < 1)$ de uma v.a discreta X é dado por:

$$\chi_p : p \leq F_X(\chi_p) \leq p + P(X = \chi_p)$$

Dado que a função de distribuição de uma $Poi(\lambda)$ é dada por:

$$F_X(x) = \sum_{i=0}^{x} \frac{e^{-\lambda}\lambda^x}{x!}$$

então podemos observar que:

1. **Mediana**: $\chi_{1/2} = 3$ pois $\frac{1}{2} \leq F_X(3) \approx 0.647 \leq 0.724$

2. **Primeiro Quartil**: $\chi_{1/4} = 2$ pois $\frac{1}{4} \leq F_X(2) \approx 0.423 \leq 0.474$
3. **Terceiro Quartil**: $\chi_{3/4} = 4$ pois $\frac{3}{4} \leq F_X(4) \approx 0.815 \leq 0.918$

Para calcularmos a $P(2 < X < 8)$, notamos que $P(2 < X < 8) = P(3 \leq X \leq 7)$ e que para o caso teórico isto se resume a $F_X(7) - F_X(2)$. Para as amostras geradas computacionalmente criou-se um ciclo que soma a frequência dos valores no vetor entre 2 e 8 e divide esse valor por 10000, o número total de observações da amostra. Os valores foram guardados nas variáveis *Prob1*, *Prob2* e *Prob3*, com os números correspondentes a cada algoritmo.

No caso teórico:

$$P(2 < X < 8) = \sum_{i=0}^{7} \frac{e^{-3} \cdot 3^x}{x!} - \sum_{i=0}^{2} \frac{e^{-3} \cdot 3^x}{x!} = 0.5649$$

Apresentamos na tabela a seguir um resumo dos valores dos quartis, mediana, média e $P(2 < X < 8)$ obtidos através do comando *summary* e do ciclo descrito anteriormente para as amostras geradas e os valores teóricos calculados.

Algo./Teórico	**Recursividade**	**Soma de Exp.**	**Gerador R**	**Teórico**
1º Quartil	2	2	2	2
Mediana	3	3	3	3
3º Quartil	4	4	4	4
Média	2.983	2.978	2.983	3
$P(2 < X < 8)$	0.5656	0.5557	0.5656	0.5649

Como era esperado, os valores dos quartis e da mediana são iguais tanto na teoria como nas amostras geradas, o que era expectável à luz de sabermos que podemos descrever a *Poisson* à custa dos métodos apresentados.

É de notar a simetria obtida nos valores para as amostras geradas a partir da recursividade da *Poisson* e do gerador do R para esta distribuição. Apesar de funcionarem por métodos diferentes, ao estarmos a reiniciar a seed antes de executar cada um dos algoritmos, isto leva à mesma criação de observações idênticas, o que confirmamos com recurso à função *all.equal*. Tal não acontece com as observações geradas pelo outro algoritmo, pois são geradas mais valores pseudo-aleatórios da $U \sim Unif(0, 1)$ do que nos outros algoritmos.

Por fim é-nos pedido para avaliar, graficamente, qual das três amostras simuladas melhor se ajusta à distribuição $Poi(3)$ e, a partir disso e dos dados apresentados anteriormente, justificar qual dos algoritmos é mais eficiente. De modo a realizar a primeira tarefa, dado que a função de probabilidade de uma variável aleatória discreta se assemelha a um histograma, onde o topo de cada barra do histograma é o valor da função de probabilidade num x discreto, recorreu-se então ao comando *barplot* para se comparar, valor a valor, se a probabilidade (teórica) se aproximava da frequência relativa das observações geradas por cada algoritmo.

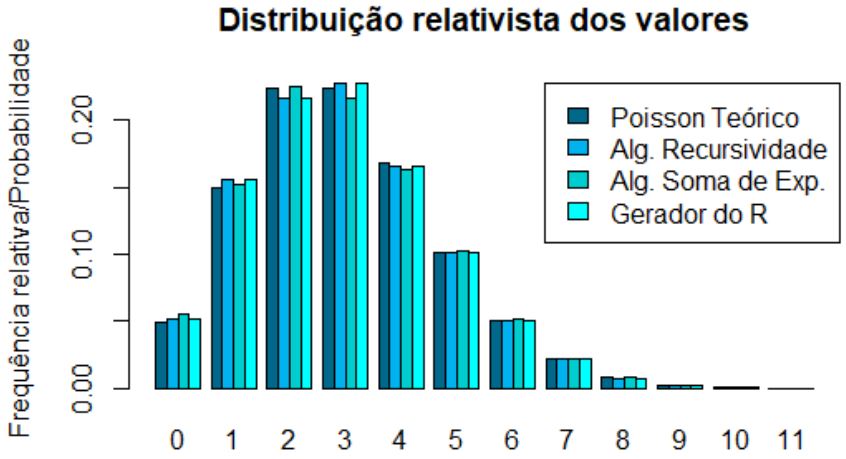

Figura 1: Representação gráfica da distribuição de Poisson e das observações geradas

Convém notar que deicidiu-se representar apenas até $x = 11$ devido a nenhum algoritmo ter gerados valores superiores a esse.

Como podemos observar todas as barras apresentam aproximadamente a mesma altura, o que significa que, a olho nu, todas as amostras geradas são um bom ajuste à teoria. É também importante notar que as colunas relativas à amostra gerada por recursividade e à amostra gerada pelo R são iguais, o que era esperado dado o que vimos acima sobre os valores gerados. Como os dados são bastante próximos uns dos outros, sentimos que apenas uma observação gráfica não seria a mais correta, e como tal decidimos optar por usar uma medida de comparação relativa, a soma dos quadrados dos desvios entre o valor teórico e os valores obtidos, isto é,

$$\sum_{x=0}^{11}(P(X = x) - Freq.Alg(x))^2$$

apresentando-se de seguida uma tabela com os dados obtidos.

Algoritmo	**Recursividade**	**Soma de Exponenciais**	**Gerador do R**
Soma Obtida	0.0001200148	0.000135269	0.0001200148

Dado que nos interessa o menor valor desta medida, pois representa a menor diferença da distância entre os valores teóricos e os obtidos pelas amostras geradas computacionalmente, então podemos concluir, apenas com base nos gráficos das funções e nesta medida, que dos 3 algoritmos utilizados tanto o recursivo como o do R são os melhores, por uma margem mínima.

Não só a medida de comparação relativa usada anteriormente é menor para o algoritmo da recursividade, o que evidencia uma melhor qualidade e eficiência

do algoritmo, como também os cálculos apresentados acima suportam este facto, onde apesar de todos os algoritmos serem bastante precisos em relação aos quartis e à mediana, o valor da média e da $P(2 < X < 8)$ aproximam-se mais do valor teórico no primeiro do que no segundo algoritmo. Aglomerando tanto esta eficiência das observações geradas com a eficiência temporal, descrita no início desta questão, onde o primeiro algoritmo executa mais rapidamente que o segundo, concluímos que o algoritmo da recursividade é mais eficiente que o algoritmo do máximo das somas das exponenciais.

1.2 Alínea b)

Pretendemos desenvolver um estudo de simulação de Monte Carlo para estimar $P = P(\bar{X} > 2)$, deste modo simulámos 500 amostras independentes de $x_i, i = 1, ..., 50$, com o algoritmo mais eficiente, sendo este o (i), já verificado em 1.a). A realização deste exercício foi efectuada, de duas maneiras:

1. sem redução da variância;
2. reduzindo a variância com variáveis antitéticas;

Para se gerarem as 500 amostras, utilizou-se os mesmos princípios que no exercício anterior, primeiramente gerou-se as amostras com variáveis pseudo-aleatórias de uma $U \sim Unif(0, 1)$, novamente recorrendo ao comando $runif(50, 0, 1)$, desta vez, apenas com 50 observações, colocado dentro de um ciclo, para se obter as amostras pretendidas. A semente de geração foi variada, optando-se por colocar $set.seed(3i)$ dentro do ciclo. Também foi utilizado o comando *rbind*, dentro deste mesmo ciclo, que simplesmente foi criando as linhas (cada linha é uma amostra de dimensão 50), donde se obteve uma matriz 500×50 (500 amostras cada uma de dimensão 50).

De seguida, realizou-se a transformação desta matriz, para as desejadas amostras, novamente, partindo dos princípios já explicados anteriormente, recorreu-se à função já criada, $PoissonRecursive$, para esta mesma realização. Deste modo, a cada observação foi aplicada esta função, com auxilio de um ciclo, que percorreu todas as amostras.

Do mesmo modo, foram geradas outras 500 amostras, com a condição de se usar variáveis antitéticas, que tinham como objetivo a redução da variância. Assim, foram consideradas estas variáveis, U e $1 - U$, que seguem ambas uma distribuição $Unif(0, 1)$ e irão preencher, cada uma, metade dos vetores, ou seja cada amostra será composta por 25 entradas de cada.

Contudo, precisamos de entender melhor o porquê de se usarem estas variáveis. Esta técnica de variáveis antitéticas permite a redução de variância, a partir de uma correlação negativa entre as estimativas. Considerando duas amostras que foram geradas, $Y_{1i}, i = 1, ..., n$ e $Y_{2i}, i = 1, ..., n$ identicamente distribuídas mas não independentes, que conduziram aos estimadores $\bar{Y_1}$ e $\bar{Y_2}$ de θ, então um possível

estimador centrado de θ é $\hat{\theta} = \frac{\bar{Y_1}+\bar{Y_2}}{2}$. Agora, com algumas manipulações da expressão da variância de θ, chegamos a

$$Var(\theta) = \frac{1}{2n} Var(Y_1)(1-\rho)$$

Assim, considerando $\bar{Y_1}$ e $\bar{Y_2}$ correlacionadas negativamente consegue-se reduzir a variância do estimador $\hat{\theta}$. Consideremos ainda a fórmula

$$\rho = \frac{Cov(Y_1, Y_2)}{\sqrt{Var(Y_1)Var(Y_2)}}$$

donde retiramos que para a correlação ser negativa basta a covariância sê-lo também. Assim se considerarmos duas v.a's com distribuição $Unif(0,1)$ tal que $Y_1 = U$ e $Y_2 = 1 - U$, temos $Cov(U, 1-U) = -Var(U) = -\frac{1}{12}$, o que justifica a utilização destas v.a's .

Para finalizar, realizou-se o procedimento idêntico efetuado para a matriz sem redução, para a obtenção das amostras com redução de variância.

Para calcular $P = P(\bar{X} > 2)$ foi utilizado o mesmo método, para as 2 matrizes (com e sem redução). Assim, recorreu-se a dois ciclos que percorrem cada entrada da matriz. Deste modo, o segundo ciclo irá percorrer os vetores, ou seja, cada amostra, donde se obteve a média de cada uma (através da soma dos valores de todas as observações dessa amostra e da divisão pelo número de observações, neste caso 50) e prosseguiu-se à criação de um vetor com os valores da média de cada amostra, denominado $Media1$ e $Media2$, para as amostras com e sem redução, respectivamente, que teve utilização mais à frente. Para o cálculo de P, recorreu-se à função if para verificar se a média da amostra obtida era superior ou igual a 3 (como se tratava de uma distribuição discreta) e assim obtivemos o números de vezes que se verificou, que dividimos pelo número total de amostras para a obtenção de $P1$ e $P2$ (P sem e com redução, respectivamente).

De seguida, foi pedido a estimativa pontual de P e para intervalar a aproximadamente 90%. Deste modo fez-se uso dos vetores com as médias ($Media1$ e $Media2$) e novamente, recorrendo a um ciclo for, realizou-se a soma de todas as médias e dividiu-se pelo número total de amostras, para a obtenção da estimativa pontual.

$$\hat{\theta} = \frac{\sum_{i=1}^{n} Y_i}{n}$$

.

De seguida, obteve-se o valor do desvio-padrão, recorrendo à função sd do R. Assim obteve-se:

$$sd(Media1) = 0.2595138$$

$$sd(Media2) = 0.06484211$$

Os valores necessários para intervalar são os presentes na seguinte expressão:

$$[\hat{\theta} - \Phi^{-1}(1 - \frac{\alpha}{2}); \hat{\theta} - \Phi^{-1}(1 - \frac{\alpha}{2})]$$

Deste modo obteve-se os intervalos de confiança IC_1 e IC_2 tais que:

$$IC_1 = [2.96067; 2.99885]$$

$$IC_2 = [2.97499; 2.984539$$

Para concluir, recorreu-se à função Var do R, para o calculo da variância e para verificar a redução do segundo caso:

$$Var(Media1) = 0.06734744$$

$$Var(Media2) = 0.004204499$$

Donde se retira uma clara redução dos valores da variância no segundo caso. Na teoria, o uso das variáveis antitéticas é melhor, pois requer um menor esforço computacional (visto que é necessário gerar menos valores, pois metade dos valores são obtidos através de $1 - U$), para além da já mencionada e confirmada, redução da variância. Conclui-se assim a melhor eficiência com o uso de variavéis antitéticas através de:

$$TE_W = n_W \times E_W = (\frac{\Phi^{-1}(1 - \frac{\alpha}{2})}{\epsilon})Var(W)E_W$$

$$TE_Z = n_Z \times E_Z = (\frac{\Phi^{-1}(1 - \frac{\alpha}{2})}{\epsilon})Var(Z)E_Z$$

Onde E_W e E_Z representa o esforço computacional, n_W e n_Z, o número de amostras (sendo neste caso, igual para ambos) e ϵ representa a eficiência. Dizemos que M_W é mais eficiente que M_Z se $TE_W < TE_Z$.

Assim, se $\begin{cases} E_W \approx E_Z \\ Var(W) < Var(Z) \end{cases} \implies M_W$ é mais eficiente.

2 Pergunta 2

Nesta questão pretendemos gerar com o método da aceitação-rejeição valores da v.a. X com função densidade de probabilidade $f_X(x) = \frac{3}{8}(1-x)^2, |x| \leq 1$ tomando como candidata a v.a. $Y \sim U(-1, 1)$. O algoritmo de aceitação-rejeição para simular valores de uma variável aleatória X com função densidade de probabilidade $f(x)$ consiste no seguinte:

Seja Y uma variável aleatória com função densidade de probabilidade $g(x)$ e

U uma variável aleatória $U(0,1)$ com U e Y independentes. Assume-se que $\forall x, 0 \leq \frac{f(x)}{cg(x)} \leq 1$, onde c é uma constante e $c \geq 1$.

1. Obter um valor y da variável aleatória Y (candidato a ser aceite/rejeitado para valor da variável aleatória X).
2. Obter um valor da $U(0,1) : u$.
3. Declarar y como um valor da variável aleatória X se $u \leq \frac{f(y)}{cg(y)}$, tomando $c = \max \frac{f(x)}{g(x)}$, caso contrário voltar ao passo 1.

Começamos por definir as funções densidade de probabilidade das variáveis X e Y, que são respetivamente $f(x) = \frac{3}{8}(1-x)^2, |x| \leq 1$ e $g(x) = \frac{1}{2}, |x| \leq 1$, e por desenhar os seus gráficos.

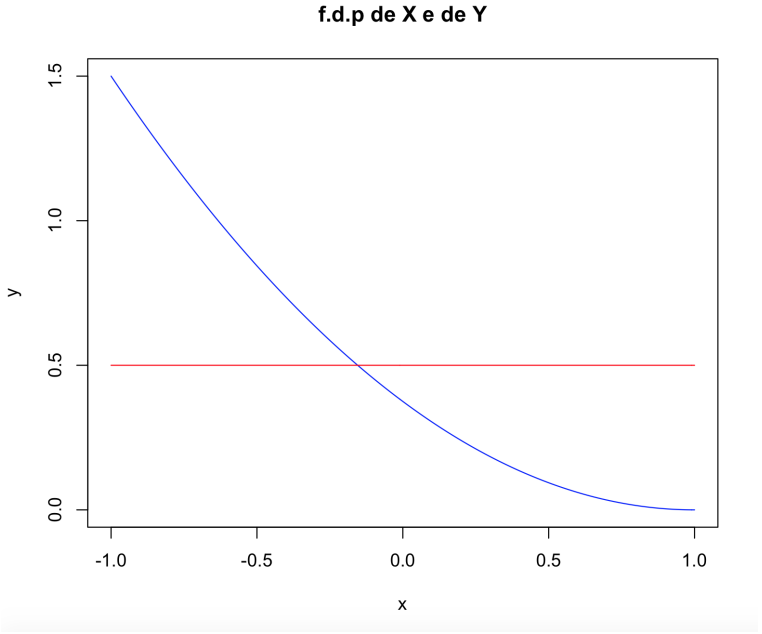

Figura 2: F.d.p de X (a azul) e f.d.p de Y (a vermelho)

Queremos agora encontrar um valor c que maximiza o quociente $\frac{f(x)}{g(x)}$.

$$\frac{f(x)}{g(x)} = \frac{\frac{3}{8}(1-x)^2}{\frac{1}{2}} = \frac{3}{4}(1-x)^2$$

$$\frac{d}{dx}(\frac{3}{4}(1-x)^2) = -\frac{3}{2}(1-x) \leq 0, \forall x \in [-1,1]$$

Como a derivada só se anula em $x = -1$ e é negativa no resto do domínio temos que o máximo da função ocorre em $x = -1$, ou seja $c = \frac{f(-1)}{g(-1)} = 3$. Confirmámos este resultado recorrendo ao comando max do R e obtemos novamente $c = 3$. Atribuímos então $c = 3$ no código e desenhámos os gráficos de $f(x)$ e de $cg(x)$. Deste modo foi

possível confirmar que $0 \leq \frac{f(x)}{cg(x)} \leq 1$, pois $f(x) \leq cg(x)$ e $f(x) \geq 0$, $cg(x) \geq 0$.

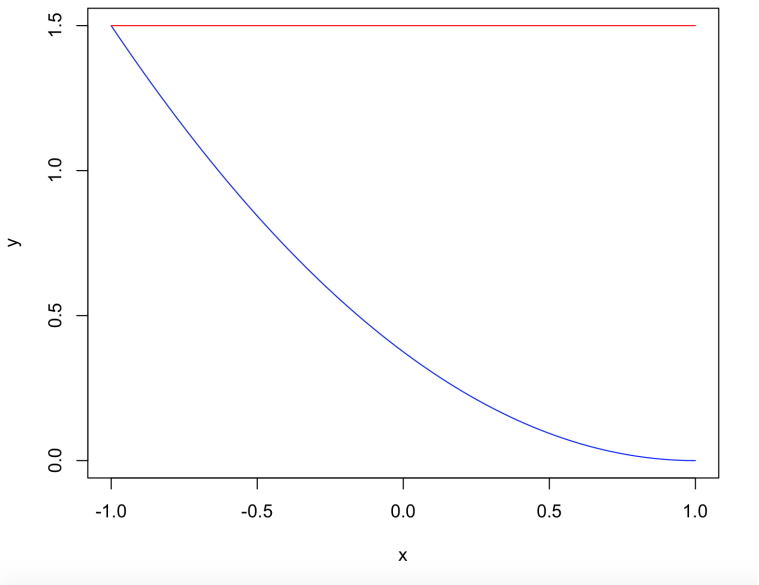

Figura 3: $f(x)$ (a azul) e $cg(x)$ (a vermelho)

Agora, como já temos o valor de c, podemos programar o algoritmo de aceitação-rejeição. Começámos por definir uma lista vazia à qual poderíamos ir adicionando os valores que fossem declarados como valores da variável aleatória X de acordo com as condições do algoritmo e criámos um ciclo for de forma a repetir o algoritmo 500 vezes, pois desta forma temos 500 pontos candidatos a Y.

O primeiro passo é obter um valor de Y, candidato a ser aceite ou rejeitado para valor da v.a. X. Fazemos isto pelo método da transformação inversa, que afirma que a função distribuição $F_X(x)$ de uma distribuição segue uma distribuição Uniforme(0,1). A função distribuição da v.a. Y é $F_Y(y) = \frac{y+1}{2}$, pois Y tem distribuição $U(-1, 1)$. Temos então

$$\frac{y+1}{2} = u \iff y = 2u - 1$$

onde u é um valor da $U(0, 1)$. Portanto a partir do comando runif(1) e da expressão obtida conseguimos gerar um valor da v.a. Y.

O segundo passo consiste em obter um valor u da $U(0, 1)$ o que já vimos conseguir fazer com o comando runif(1).

O terceiro e último passo é verificar se $u \leq \frac{f(y)}{cg(y)}$ e, se verifica, podemos declarar y como um valor da variável aleatória X e adicionar y à lista criada inicialmente para os valores da v.a. X.

Agora que temos o algoritmo feito, podemos facilmente verificar que 169 dos 500 valores candidatos da distribuição uniforme foram aceites, usando o comando length para ver o comprimento da lista devolvida pelo algoritmo. De forma a fazer uma análise mais completa dos resultados, desenhámos os seguintes gráficos:

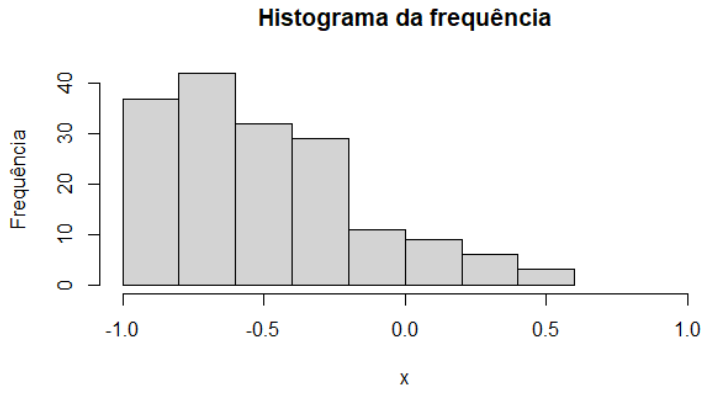

Figura 4: Histograma da frequência

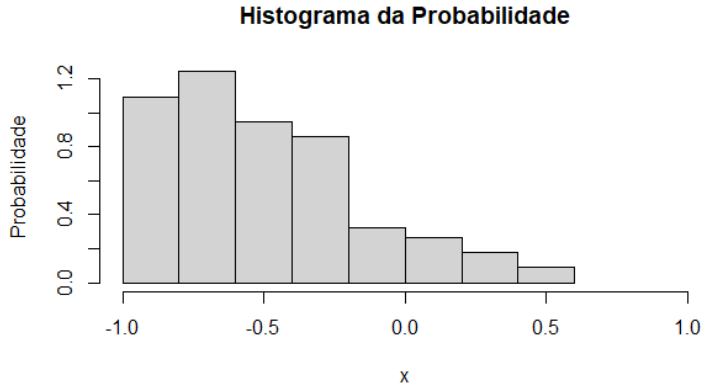

Figura 5: Histograma da probabilidade

O primeiro histograma representa a frequência absoluta de valores em cada intervalo de x apresentado e o segundo histograma representa a frequência relativa dos mesmos valores nos mesmos intervalos (a soma da área de todas as barras no segundo histograma é 1). Podemos agora desenhar o gráfico da função densidade de probabilidade da v.a. X por cima do histograma da probabilidade de forma a comparar o resultado obtido com o que era esperado.

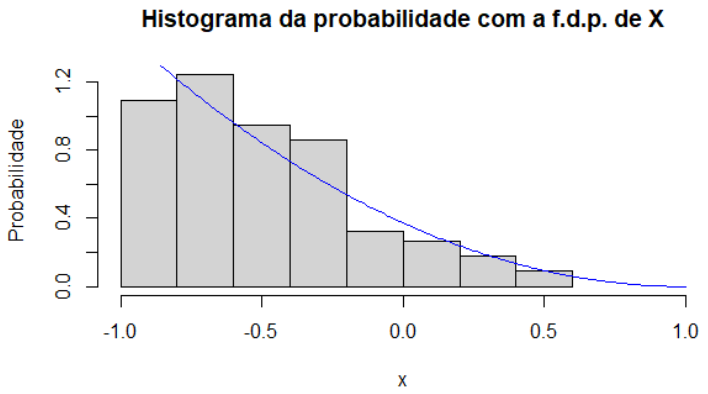

Figura 6: Histograma da probabilidade com a fdp de X

Deste modo podemos concluir que os valores obtidos pelo método da aceitação rejeição são muito próximos da distribuição X que queríamos gerar, e portanto dizemos que a amostra que foi aceite segue a distribuição X.

3 Pergunta 3

O **objetivo** deste exercício é *simular o valor-p* num *Teste de Sinais* que confronta as hipóteses nula $H_0 : \chi_{0.5} = 26.7406$ e alternativa $H_1 : \chi_{0.5} \neq 26.7406$, ou seja, testa o valor da *mediana* de uma amostra de dimensão 6 da variável aleatória $X \sim Gama(3, 0.1)$, utilizando o conjunto N de 500 amostras para o estudo.

O *Teste dos Sinais*, um *teste não paramétrico*, é eficiente nesta simulação devido à dimensão das amostras e ao elevado número destas utilizado. Nesta simulação foram utilizadas 500 amostras aleatórias X_i: $\underset{\sim}{X} = (X_1, ..., X_6)$, $i = \{1, 500\}$. Para *estatística de teste* utilizou-se $S_6 = \sum_{n=1}^{6} I(X_n - \chi_0)$, em que

$I(X_n - \chi_0) = \begin{cases} 1, X_n > \chi_0 \\ 0, X_n \leq \chi_0 \end{cases}$, sendo que sob H_0: $S_6 \sim Bin(6, 0.5)$ pois $P(X_n > \chi_0) = 0.5$ e $\chi_0 = 27.7406$; ou seja, trata-se do número de diferenças positivas entre as variáveis e a mediana assumida por hipótese.

Para o cálculo do **valor-p** utilizou-se dois processos diferentes:

1. O processo **usual**:
   Para cada amostra, após calcular o valor da estatística $E = S_6$ calculou-se:

$$valor - p = 2 \times min\{0.5, P(S_6 \geq E), P(S_6 \leq E)\}$$

   (0.5 para evitar os casos em que as outras probabilidades são superiores a este valor)

2. O processo do **Princípio da Verosimilhança Mínima**:
   Para cada amostra, o *valor-p* é a soma das probabilidades para os possíveis valores de $0 \leq S_6 \leq 6$ inferiores ou iguais à probabilidade de S_6 igualar a estatística de teste.

$$valor - p = \sum_{a=0}^{6} \left[P(S_6 = a) \middle| \{P(S_6 = a) \leq P(S_6 = E)\} \right]$$

Calculados os valores-p das 500 amostras reuniu-se os resultados em *Histogramas* e avaliou-se a veracidade da Hipótese nula.

Para alcançar os resultados foi utilizado o programa *RStudio* e elaborado um algoritmo com os seguintes processos:

1. Gerou-se 500 amostras de de dimensão 6 de variáveis aleatórias $X \sim Gama(3, 0.1)$. Através de um *ciclo for* adicionaram-se 500 linhas à matriz N (com a função *rbind*) com 6 variáveis aleatórias Gama em cada, geradas pelo gerador do *programa R* (com a função *rgamma*) e tendo implementado anteriormente a nossa semente (*set.seed(*3*)*).

2. Para o cálculo das 500 estatísticas percorreu-se as linhas da matriz N num *ciclo for*, e em cada linha somou-se os valores das colunas que são superiores a 27.7406 com um novo *ciclo for* sendo o resultado anexado ao *vetor EstTst* com a função *append*.

3. Percorreu-se o *vetor EstTst* num ciclo for e calculou-se os 500 *valores-p* de acordo com a fórmula, utilizando as funções *pbinom* e *dbinom* para calcular a distribuição e a probabilidade, respetivamente, das estatísticas de teste e por fim anexou-se os resultados ao *vetor pvaluesu*.

4. De novo, percorreu-se o *vetor EstTst* num *ciclo for* e, recorrendo também às funções de distribuição e probabilidade calculou-se os 500 $valores - p_{PVM}$. Desta feita, foi necessário usar novo *ciclo for* para somar apenas as probabilidades iguais ou inferiores a $P(S_6 = E)$ e anexou-se os resultados ao *vetor pvaluespvm*.

Com os resultados obtidos foram gerados dois *Histogramas*, um para cada processo de cálculo dos valores-p :

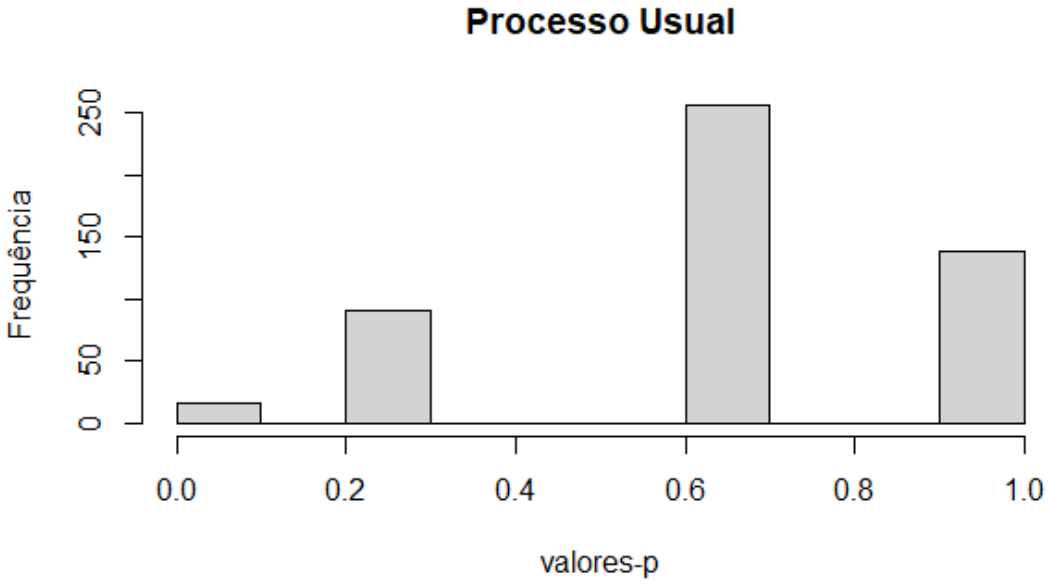

Figura 7: Histograma do cálculo de p-values pelo processo usual

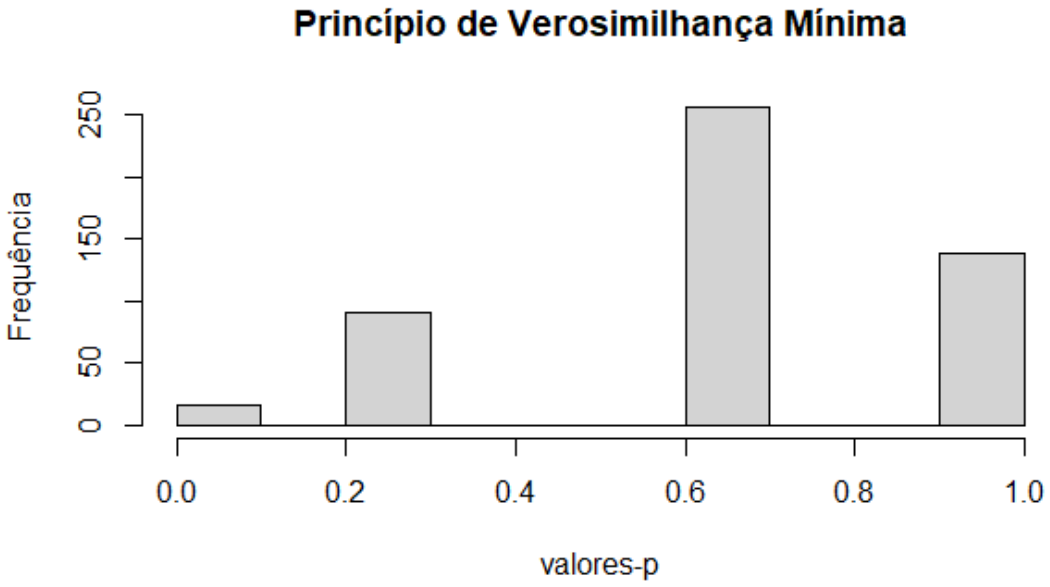

Figura 8: Histograma do cálculo de p-values pelo PVM

Através da análise de ambos os Histogramas verifica-se que em cerca de 400 amostras a *Hipótese Nula* não se rejeita para *níveis de significância* inferiores a 60% e destas, em aproximadamente 150 amostras o *valor-p* está entre os 90% e os 100%, logo não se rejeita para os valores usuais $(5\%, 10\%, 15\%)$.

Apesar de nas restantes amostras os valores-p serem bastante mais baixos acreditamos que tendo conta estes dados, **não se deve rejeitar a Hipótese Nula** de que a mediana de uma amostra de dimensão 6 da variável $X \sim Gama(3, 0.1)$ é 26.7406.

Além disso, podemos também afirmar que os processos *usual* e pelo *princípio da verosimilhança mínima* de cálculo de valores-p do teste de sinais originaram exatamente os mesmos resultados, o que gera mais confiança nestes.

Verificou-se que nos histogramas existem apenas 4 valores-p diferentes: isto verifica-se devido às propriedades da *distribuição binomial*, pois como a estatística de teste $S_6 \sim Bin(6, 0.5)$, a sua função probabilidade é $P(S_6 = k) = \binom{6}{k}(0.5)^6$ e esta é simétrica para os valores de k, ou seja $P(S_6 = k) = P(S_6 = 6 - k)$.

No processo de elaboração do algoritmo em R verificou-se também que o programa tem um ligeiro erro na devolução do valor das probabilidades de variáveis binomiais através da função *dbinom*. O comando *dbinom(1,6,0.5)==dbinom(5,6,0.5)* devolve *FALSE*, o que está **incorreto**. Devido a isto, no código desta questão, no cálculo dos 500 valores-p PVM evitou-se uma igualdade e usou-se o módulo da diferença de variáveis menor que 10^{-10} para evitar erros nos resultados.

4 Referências

- S.M. Ross (2009). *Introduction to Probability and Statistics for Engineers and Scientists*: Elsevier - Academic Press, 4rd ed.
- D. Kroese, T. Taimre, Z.I. Botev (2011). *Handbook of Monte Carlo Methods*: Wiley - Academic Press, 1st ed.
- Manuel Cabral Morais (2012). *Notas de Apoio de Probabilidade e Estatística.*
- Notas das Aulas de Complementos de Probabilidade e Estatística.