Métodos Probabilísticos para Engenharia Informática
Ano Letivo: 2020/2021
Turma P4
Tomé Lopes Carvalho - 97939
Gonçalo Fernandes Machado - 98359

Relatório PL2: Probabilidades e Variáveis Aleatórias

1. a)

```
p1 = 0.002;                                 %Prob. Componente 1 ser defeituosa
p2 = 0.005;                                 %Prob. Componente 2 ser defeituosa
pa = 0.01;                                  %Prob. montagem ser defeituosa
N = 1e4;                                    %Nº de experiências

n = 8;                                      %Número de brinquedos
comp1 = rand(n, N) < p1;                    %Componentes 1 para cada caixa
comp2 = rand(n, N) < p2;                    %Componentes 2 para cada caixa
caixasPorMontar = comp1 | comp2;            %União dos defeitos dos comp1 e comp2
defMontMatriz = rand(n, N) < pa;            %Matriz de brinquedos que serão montados
                                            %defeituosamente
caixas = caixasPorMontar | defMontMatriz; %União dos defeitos de montagem
nCaixasDef = sum(sum(caixas) >= 1);         %Número de caixas com 1 ou mais defeitos
probSimA = nCaixasDef/N;                    % ~0.13
%Algumas variáveis criadas serão usadas para alíneas posteriores
```

No início, são geradas matrizes que representam a existência de defeitos nos 2 componentes que serão utilizados na montagem dos brinquedos, tendo em conta as probabilidades enunciadas (*p1* = 0,2%, *p2* = 0,5%).
São matrizes *n* x *N*, em que cada coluna representa uma caixa (experiência) com *n* (8) brinquedos.
O resultado da união dos defeitos, a matriz *caixasPorMontar*, obtida através da operação "OR" entre *comp1* e *comp2* representa a existência de 1 ou 2 componentes defeituosos em cada brinquedo.
De seguida, é gerada a matriz de ocorrência de montagem defeituosa, *defMontMatriz*, do mesmo modo.
A variável *caixas* é, então, a união de todos os defeitos (*caixasPorMontar* | *defMontMatriz*).
Para obter o número de caixas com brinquedos defeituosos, *nCaixasDef*, determina-se o número de colunas de *caixas* com soma >= 1 (com 1 ou mais brinquedos defeituosos).
A probabilidade desejada, *probSimA*, é igual ao quociente entre o número de caixas defeituosas e o número total de caixas.
Resultado: *probSimA* = 0,13 (aproximadamente).

1. b)

```
defSoMont = 0;
for col = 1:N                               %Para cada caixa
    for row = 1:n                           %Para cada brinquedo
        if caixasPorMontar(row, col) == 0  %Se não tiver componentes defeituosos
            defSoMont = defSoMont + defMontMatriz(row, col); %Adiciona 1 quando é
        end                                                   %defeituoso por montagem
    end
end
avgDefMontPorCaixa = defSoMont/nCaixasDef; %Média de defeitos apenas de montagem
```

Para obter o número médio de brinquedos defeituosos apenas por montagem (*avgDefMontPorCaixa*) quando ocorre A e n = 8, temos de calcular o quociente entre o número de defeitos apenas por montagem (*defSoMont*) total e o número de caixas defeituosas (*nCaixasDef*).
Para cada brinquedo, verifica-se cumpre as seguintes condições:

1. Não tem componentes defeituosos (O valor correspondente a esse brinquedo na matriz *caixasPorMontar* é igual a 0)
2. A sua montagem é defeituosa (O valor correspondente a esse brinquedo na matriz *defMontMatriz* é igual a 1)

Caso as cumpra, é incrementada por 1 a variável *defSoMont* (utiliza-se apenas uma condição no código, pois ao adicionar *defMontMatriz*(*row*, *col*) será adicionado 1 se existir um defeito ou 0 se não existir um defeito).
Resultado: *avgDefMontPorCaixa* = 0,63 (aproximadamente).

2. a)

```
nCaixasBoas = sum(sum(caixas) == 0);            %Número de caixas com 0 defeitos
probSimB = nCaixasBoas/N;
%Os resultados são coerentes se a soma de probSimA e probSimB for igual a 1
```

Essencialmente, pretende-se determinar o contrário da alínea 1. a).
Na variável *nCaixasBoas* conta-se o número de caixas com 0 defeitos (número de colunas com apenas zeros na matriz *caixas*).
A probabilidade desejada é igual ao quociente entre *nCaixasBoas* e o número total de caixas.
Resultado: *probSimB* = 0,87, aproximadamente.
Tal como esperado, *probSimA* + *probSimB* é sempre igual a 1, uma vez que A e B são acontecimentos complementares.

2. b)

P(1) = 0,002; P(2) = 0,005; P(a) = 0,01
P(1 v 2) = P(Cd): Probabilidade de pelo menos um dos componentes de um brinquedo ser defeituoso
P(1 v 2) = P(1) + P(2) - P(1 ∧ 2) = 0,002+0,005 - 0,002*0,005 = 0,00699
P(Cd) = 0,00699
P(Cd v a) = P(d): Probabilidade de um brinquedo ser defeituoso
P(Cd v a) = P(Cd) + P(a) - P(Cd ∧ a) = 0,00699+0,01 - 0,00699*0,01 = 0,0169201
P(d) = 0,0169201

Função binomial de probabilidade: P(X = x) = nchoosek(n, x) * p^x * (1-p)^(n-x)
P(B) = P(X = 0) = nchoosek(8, 0) * (0,0169201)^0 * (1-0,0169201)^8 = 0,8724

Alguns resultados de P(B) por simulação, para comparação (N = 1e7):

1. 0,8722
2. 0,8724
3. 0,8724
4. 0,8723

Tomando o valor teórico como correto, conclui-se que a probabilidade por simulação foi calculada corretamente.

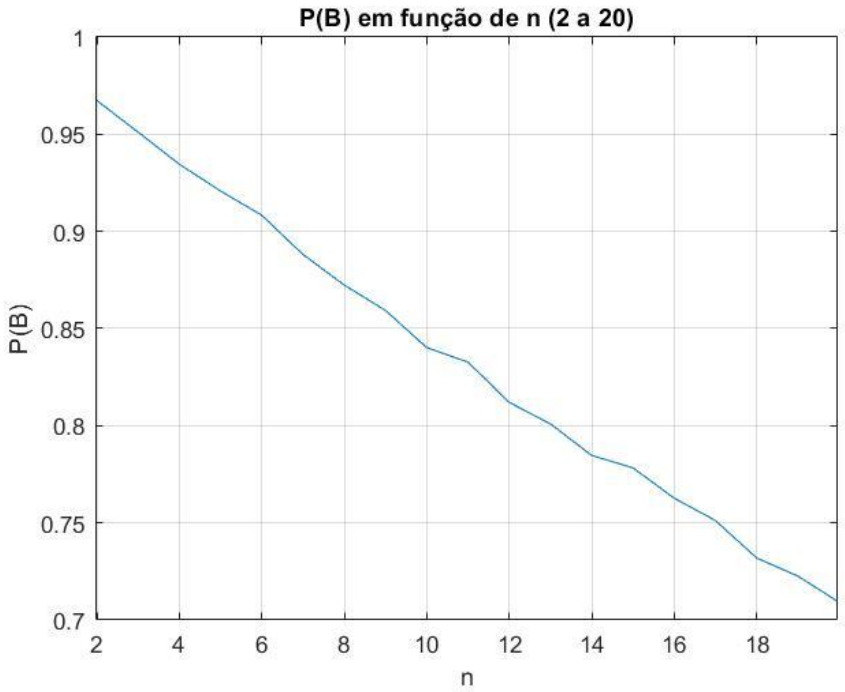
2. c)

```
function [p] = probNdef(p1, p2, pa, n, N, nDef)
    comp1 = rand(n, N) < p1;                    %Componentes 1 para cada caixa
    comp2 = rand(n, N) < p2;                    %Componentes 2 para cada caixa
    caixasPorMontar = comp1 | comp2;            %União dos defeitos dos comp1 e comp2
    defeitosMontagem = rand(n, N) < pa;         %Matriz de brinquedos que serão
                                                %montados defeituosamente
    caixas = caixasPorMontar | defeitosMontagem;%União dos defeitos de montagem
    sucessos = sum(sum(caixas) == nDef);        %Número de caixas com nDef defeituosos
    p = sucessos/N;                             %Rácio entre o nº de caixas com nDef
                                                %defeituosos e o nº total de caixas
end
```

Para calcular a probabilidade de uma caixa com *n* brinquedos (*n* de 2 a 20) não possuir nenhum brinquedo com defeito, e de maneira a simplificar o código, criámos a função *probNdef*, que, aceitando como argumentos as probabilidades de erro de componentes e montagem, número de brinquedos por caixa, número de experiências e número de brinquedos defeituosos *nDef* e retorna a probabilidade simulada de uma caixa ter exatamente *nDef* brinquedos defeituosos.
Funciona de maneira semelhante ao código utilizado em 1. a), quanto a gerar as caixas. Calcula o número de caixas com *nDef* defeituosos (*sucessos*).
A probabilidade desejada é obtida através do quociente entre o número de *sucessos* e o número total de caixas, *N*.

```
probSimB_array = zeros(1, 19);
%Array em que serão colocados os valores da probabilidade de existirem 0
%defeitos em caixas com 2 a 20 brinquedos
for i  = 2:20
    probSimB_array(i-1) = probNdef(p1, p2, pa, i, N, 0);
end
plot(2:20, probSimB_array);
title('P(B) em função de n (2 a 20)');
xlabel('n');
ylabel('P(B)');
grid on;
```

Começamos por inicializar o array, pois já sabemos o tamanho que o array vai ter e para otimizar o programa. Percorremos o array, colocando em cada index a probabilidade de existirem 0 brinquedos defeituosos com uma caixa com n brinquedos.
O gráfico que se obtém encontra-se abaixo.



Podemos observar no gráfico que quanto maior for o número de brinquedos na caixa, menor é a probabilidade de a caixa ter 0 brinquedos com defeito. Isto deve-se ao facto de cada brinquedo ter uma probabilidade igual de possuir defeitos, e quanto maior o número de brinquedos, maior é a probabilidade de pelo menos 1 ter defeitos.

2. d)

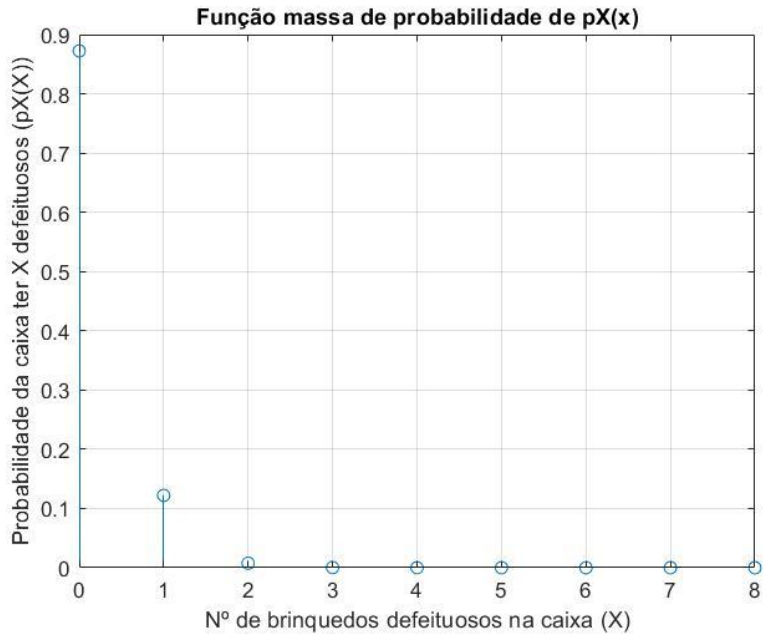
Por análise do gráfico, cada caixa deve ter no máximo 6 brinquedos se a empresa quiser garantir que a probabilidade de cada caixa não ter brinquedos defeituosos seja de pelo menos 90%.

3. a)

```
pX = zeros(1, 9); %Array da função massa de probabilidade de pX(x)
for X = 0:8
    pX(X+1) = probNdef(p1, p2, pa, n, N, X);
end
figure(2);
stem(0:8, pX);
title('Função massa de probabilidade de pX(x)');
xlabel('Nº de brinquedos na caixa (X)');
ylabel('Probabilidade da caixa ter X defeituosos (pX(X))');
grid on;
%Verifica-se que o resultado de 2 - a) é igual ao resultado obtido nesta
%alínea para X = 0 (quando não existe nenhum brinquedo defeituoso)
%Sendo assim, os resultados obtidos são consistentes.
```

Foi criado o array *pX* que representa a função massa de probabilidade **pX**(x) (**nota**: o valor de *pX* correspondente ao índice *n* corresponde ao valor da função de massa de probabilidade para *n-1*, uma vez que o primeiro valor de X é 0, porém, em MatLab, os índices começam em 1).

Representaram-se graficamente os resultados:



Dado que a probabilidade de um brinquedo ser defeituoso é apenas cerca de 1,7%, verifica-se que a probabilidade rapidamente decresce com o aumento de número de brinquedos defeituosos considerado.
Para X = 0, o resultado foi idêntico ao da alínea 2. a). Sendo assim, os resultados são consistentes.

3. b)

```
prob3b_array = zeros(1, 9);
prob3b_array(1) = pX(1);
for X = 1:8
    prob3b_array(X+1) = prob3b_array(X) + pX(X+1);
end
prob3b = prob3b_array(8) - prob3b_array(2);
```

A probabilidade de X >= 2, com base em pX(x), é essencialmente a probabilidade de numa caixa com 8 brinquedos, existirem 2 ou mais brinquedos defeituosos. Para isso, utilizamos o array *pX(x)* da alínea anterior, e criamos um novo array (*prob3b_array*), onde é calculado a função de distribuição acumulada para cada X.
Para obtermos a probabilidade desejada, subtraímos à probabilidade total, que seria 1 teoricamente, mas sendo uma simulação pode ser diferente de 1, a probabilidade de ter 0 ou 1 brinquedos defeituosos, tendo obtido um resultado de 0.0071, aproximadamente.
Dada o número de brinquedos na caixa ser baixo e a probabilidade de um brinquedo ter defeito, é natural que a probabilidade de existir 2 ou mais brinquedos defeituosos numa caixa com 8 brinquedos seja quase nula.

3. c)

```
E = 0;
for i = 0:n
    E = E + i*pX(i+1);
end

var = 0;
for i = 0:n
    var = var + ((i - E)^2)*pX(i+1);
end

desvioPadrao = var^(1/2);
```

$$\mathrm{E}[X] = \sum_i x_i p(x_i)$$

$$\mathrm{var}(X) = \sigma^2 = \sum_i [x_i - E(X)]^2 \, p(x_i)$$

O valor esperado foi calculado através da fórmula para casos discretos e a variância foi calculada através da definição. O desvio padrão é dado pela raiz quadrada da variância
Resultados: *E* = 0,14; *var* = 0,13; *desvioPadrao* = 0,37; Valores aproximados.

3. d)

Foi repetido o código das alíneas anteriores, mas com *n* = 16.
Os valores obtidos foram, aproximadamente, os seguintes:
E = 0,27; *var* = 0,28; *desvioPadrao* = 0,53.

Para uma caixa de 16 brinquedos, a probabilidade de existirem *i* (*i* entre 1 e 16) brinquedos defeituosos é superior à probabilidade de existirem *i* brinquedos defeituosos numa caixa com apenas 8 brinquedos.
Isto deve-se ao facto de que com mais brinquedos na caixa, existem mais hipóteses para a colocação de um defeituoso na caixa. Também é por este motivo que diminui a probabilidade de não existirem defeituosos ao aumentar o número de brinquedos na caixa.
Seguindo a mesma lógica, a probabilidade de existirem 2 ou mais brinquedos defeituosos numa caixa aumentam. Do mesmo modo, o valor esperado aumenta.
Existindo mais valores possíveis para o número de brinquedos defeituosos, é possível estarem mais distantes da média, causando o aumento da variância e, consequentemente, do desvio padrão.

4. a)

```
n = 20;                                  %Número de brinquedos
pa = 0.001;                              %processo de montagem melhorado
comp1_4 = rand(n, N) < p1;               %Componentes 1 para cada caixa
comp2_4 = rand(n, N) < p2;               %Componentes 2 para cada caixa
caixasPorMontar_4 = comp1_4 | comp2_4;   %União dos defeitos dos comp1 e comp2
defMontMatriz_4 = rand(n, N) < pa;       %Matriz de brinquedos que serão montados
defeituosamente
caixasEx4 = caixasPorMontar_4 | defMontMatriz_4; %União dos defeitos de montagem
nCaixasCom = 0;
nCaixasDefCom = 0;

for i = 1:N
    if caixasEx4(1, i) == 0          %se o primeiro brinquedo da caixa não for defeituoso
        nCaixasCom = nCaixasCom + 1; %incrementar o número de caixas comercializadas
        if sum(caixasEx4(:,i)) >= 1  %se houver um defeituoso na caixa + comercializada
            nCaixasDefCom = nCaixasDefCom + 1;  %incrementar o número de caixas
        End                                     %defeituosas comercializadas
    end
end

probCaixaCom = nCaixasCom/N;
probCaixaDefCom = nCaixasDefCom/N;
```

No início, são geradas as caixas da mesma maneira que nas alíneas anteriores, mas com probabilidades alteradas.
De seguida, utilizando o ciclo *for*, analisa-se o primeiro brinquedo de cada caixa. Se este não for defeituoso, comercializa-se a caixa (é incrementada a variável *nCaixasCom*).
Como a probabilidade de um brinquedo ser defeituoso não é afetada pela ordem dos brinquedos, analisar o 1º brinquedo ou selecionar um ao acaso levará ao mesmo resultado, pelo que decidimos não utilizar a função *randperm* (experimentando utilizar *randperm*(20, 1) ao invés de 1 na condição *if* produz o mesmo resultado).
Verifica-se se a caixa comercializada é defeituosa ao verificar se existe pelo menos um brinquedo defeituoso na caixa inteira (*caixasEx4*(:,*i*)). Caso seja, incrementa-se a variável que representa o número de caixas defeituosas comercializadas, *nCaixasDefCom*.
Averigua-se a probabilidade de uma caixa ser comercializada (*probCaixaCom*) através do quociente entre o número de caixas comercializadas (*nCaixasCom*) e o número total de caixas (*N*).
Calcula-se a probabilidade de uma caixa defeituosa ser comercializada (*probCaixaDefCom*) ao dividir o número de defeituosas comercializadas (*nCaixasDefCom*) pelo número total de caixas.

Resultados:
probCom = 0,99; *probCaixaDefCom* = 0,14 (valores aproximados)
A probabilidade de uma caixa ser comercializada quando m = 1 é ~99%.
Verifica-se que ~14% das caixas comercializadas terão pelo menos um brinquedo defeituoso.
Conclui-se que uma amostra m = 1 de brinquedos não permite atingir o objetivo desejado (90% das caixas comercializadas não terem brinquedos defeituosos).

4. b)

```
caixasBoasCom4b = 0;    %rácio de caixas boas comercializadas por caixas comercializadas
m = 1;            %Torna a condição do while verdadeira, já que MatLab não tem "do while"
while caixasBoasCom4b < 0.9
    nCaixasCom4b = 0;     %Número de caixas comercializadas
    nCaixasDefCom4b = 0;  %Número de caixas defeituosas comercializadas
    m = m + 1;            %Começa por m = 2, pois já se sabe que m = 1 não é suficiente
    for i = 1:N           %Para cada caixa comercializada
        brinquedosAnalisados = caixasEx4(1:m, i);  %Serão analisados os m primeiros
                                                   %brinquedos da caixa
        if sum(brinquedosAnalisados) == 0          %Se não existirem defeituosos nos
                                                   %brinquedos selecionados da caixa:
            nCaixasCom4b = nCaixasCom4b + 1;       %incrementar o número de caixas
                                                   %comercializadas
            if sum(caixasEx4(:,i)) >= 1            %se houver um brinquedo defeituoso na
                                                   %caixa comercializada
                nCaixasDefCom4b = nCaixasDefCom4b + 1;  %incrementar o número de caixas
                                                        %defeituosas comercializadas
            end
        end
    end

    caixasBoasCom4b = 1 - nCaixasDefCom4b/N;  %Rácio entre o nº de caixas comercializadas
                                              %sem defeitos e o nº total de caixas
                                              %comercializadas
end
%Conclui-se que o objetivo é atingido quando m é, no mínimo, 7.
```

A variável *caixasCom2Boas* é a probabilidade das caixas comercializadas não possuírem nenhum brinquedo defeituoso.
É usado um ciclo *while*, para o código correr enquanto não for satisfeita a condição, que neste caso é até a probabilidade atingir os 90%.
Neste ciclo o número de brinquedos na amostra começa por ser 2, pois verificámos na alínea anterior que quando *m* = 1 não é satisfeita esta condição.
Utilizando a matriz *caixasEx4*, escolhemos de cada caixa os *m* primeiros brinquedos, e se entre os brinquedos selecionados não existir nenhum brinquedo com defeito a caixa é comercializada, tal como na alínea 4 - a).
Posteriormente, verifica-se das caixas comercializadas, quais possuem pelo menos 1 brinquedo com defeito, e calcula-se a probabilidade (*caixasCom2Def*). Finalmente calculamos a probabilidade das caixas comercializadas não possuírem nenhum brinquedo defeituoso e subtraímos a 1, pois este evento é o complementar do evento que pretendemos saber.
Por simulação, verificamos que o menor valor de *m* para que a condição se verifique é 7.