**Problemas em equipe 02**

Estudantes: Eduardo Eiji Goto, Gustavo Hammerschmidt, João Vitor Andrioli.

**Parte 1 – Variáveis aleatórias discretas**

1. Uma variável aleatória discreta tem a distribuição de probabilidade dada por: *pX*(*x*) = *K*/*x*, para *x* = 1, 3, 5 e 7.
   1. Calcule o valor de *K* .

K/1 + K/3 + k/5 + k/7 = 1

(mmc: 3, 5, 7 = 105)

K \* 105 + K \* 105/3 + K \* 105/5 + K \*105/7 = 105

176 \* K = 105

K = 105/176 = 0.59659...

* 1. Apresentar a tabela que define *pX*(*x*).

|  |  |  |
| --- | --- | --- |
| *x* | *pX*(*x*) | *Px(X)* |
| 1 | K/1 | 0.59659 |
| 3 | K/3 | 0.19886 |
| 5 | K/5 | 0.11931 |
| 7 | K/7 | 0.08522 |

* 1. Calcular *E*[*X*].

E[X] = 1 \* 0.59659 + 3 \* 0.19886 + 5 \* 0.11931 + 7 \* 0.08522

E[X] = 2.38626

**Parte 2 – Distribuições Geométrica e Poisson**

1. A probabilidade de sucesso de um experimento é de 0,3 e ele deve ser realizado até que o resultado com sucesso seja alcançado. Qual a probabilidade que o experimento tenha que se realizar mais de 3 vezes?

Colocar como resposta os comandos do scypy e o valor do resultado.

Psucc[x] = 0.3

Pfrac[x] = 1 – Psucc[x] = 0.7

Pexec[3 < X] = Pfrac[x] ^ 3 = 0.7 \* 0.7 \* 0.7

Pexec[3 < X] = 0.343

=================================PYTHON

from scipy.stats import geom

print(1-geom.cdf(3, 0.3))

========================================

OUTPUT: 0.34299999999999997

1. Um departamento de qualidade realiza testes em hard disk selecionados aleatoriamente. A política definida é parar o processo de fabricação para calibração se um inspetor encontrar mais do que quatro defeitos em um disco. Qual é a probabilidade de o processo de fabricação ser interrompido se o número médio de defeitos é dois, e os defeitos são distribuídos segundo a distribuição de Poisson?

Colocar como resposta os comandos do scypy e o valor do resultado.

=================================PYTHON

from scipy.stats import poisson

print(1-poisson.cdf(4, 2))

========================================

OUTPUT: 0.052653017343711084

**Parte 3 – Simulação vetorial**

2.1 O arquivo “SimVet1.ipynb” contém o código Python para Jupyter Notebook para a simulação interativa e vetorial do problema do dado simulado na aula anterior. Preencher tabela seguinte com os tempos de simulação para os seguintes números de simulação.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Simulação interativa | | | | |
| Número de simulações | 100 | 1000 | 10000 | 100000 |
| Tempo de execução | 0.0087 | 0.0419 | 0.3236 | 3.3562 |
|  | | | | |
| Simulação vetorial | | | | |
| Número de simulações | 100 | 1000 | 10000 | 10000 |
| Tempo de execução | 0.0006 | 0.0005 | 0.0042 | 0.0130 |

Qual dos dois métodos é mais eficiente?

O vetorial, pois gera uma matriz com todos os dados primeiramente para, então, realizar os procedimentos lógicos.

2.2 O arquivo “SimVet1.ipynb” contém o código Python para Jupyter Notebook para a simulação interativa do problema da moeda apresentado na aula anterior. Contém também dicas e instruções para o algoritmo vetorial. Programar o algoritmo vetorial e copiar o código aqui. Caso não consiga finalizar, na próxima aula haverá tempo para finalizar.

==============Função Vetorial

import scipy.stats as st

def MoedaV(n):

    # Passo 1

    # Calcular o valo de m

    m = int(st.geom.ppf(0.9999, 0.5))

    # sortear n simulacoes do lancamento de uma moeda m vezes

    # 0 - coroa      1 - cara

    sorteio = np.random.randint(0,2,[n, m])

    # Passo 2

    # Usar np.argmax para encontrar um vetor com as posições da primeira cara sorteada em cada linha

    x = np.argmax(sorteio, 1) + 1

    # Passo 3 Passo 3

    # Contar quantas vezes a primeira cara saiu em um lançamento de número de ordem par e dividir pela quantidade de simulações.

    return np.sum(x % 2 == 0)/n

=========== Output Interativo

for n in [100,1000,10000,100000]:

    t1 = time.perf\_counter()

    probS = Moeda(n)

    t2 = time.perf\_counter()

    print(str(n) + '-'\*40)

    print('Probabilidade simulada:  {:.4f}'.format(probS))

    print('Probabilidade teórica: {:.4f}'.format(1/3))

    print('Tempo de simulação: {:.4f}'.format(t2-t1))

100----------------------------------------

Probabilidade simulada: 0.3500

Probabilidade teórica: 0.3333

Tempo de simulação: 0.0087

1000----------------------------------------

Probabilidade simulada: 0.3320

Probabilidade teórica: 0.3333

Tempo de simulação: 0.0419

10000----------------------------------------

Probabilidade simulada: 0.3353

Probabilidade teórica: 0.3333

Tempo de simulação: 0.3236

100000----------------------------------------

Probabilidade simulada: 0.3343

Probabilidade teórica: 0.3333

Tempo de simulação: 3.3562

**======================= Output Vetorial**

for n in [100,1000,10000,100000]:

    t1 = time.perf\_counter()

    probS = MoedaV(n)

    t2 = time.perf\_counter()

    print(str(n) + '-'\*40)

    print('Probabilidade simulada:  {:.4f}'.format(probS))

    print('Probabilidade teórica: {:.4f}'.format(1/3))

    print('Tempo de simulação: {:.4f}'.format(t2-t1))

100----------------------------------------

Probabilidade simulada: 0.4300

Probabilidade teórica: 0.3333

Tempo de simulação: 0.0006

1000----------------------------------------

Probabilidade simulada: 0.2940

Probabilidade teórica: 0.3333

Tempo de simulação: 0.0005

10000----------------------------------------

Probabilidade simulada: 0.3383

Probabilidade teórica: 0.3333

Tempo de simulação: 0.0042

100000----------------------------------------

Probabilidade simulada: 0.3305

Probabilidade teórica: 0.3333

Tempo de simulação: 0.0130