Simulação das propriedades da exponencial

Importar bibliotecas

```
import time
import numpy as np
import scipy.stats as st
```

Probabilidade de uma variável aleatória exponencial ser menor do que outra

Sejam X_1 e X_2 variáveis aleatórias independentes com média $mu_1 = 1/L_1$ e $mu_2 = 1/L_2$

- P$[X_1 \leq X_2]$ = $\frac{L_1}{L_1 + L_2}$

Podemos simular essa probabilidade com o seguinte algoritmo vetorial, implementado na função **pExpMenor**.

- Sortear array exp_1 com $nSim$ valores exponenciais com média mu_1.
- Sortear array exp_2 com $nSim$ valores exponenciais com média mu_2.
- Calcular vetor $menor$ com valores $True$ se $exp_1 < exp_2$.
- Retornar a quantidade de elementos no vetor $menor$ contendo valor True, dividida por $nSim$.

```
def pExpMenor(MU1, MU2, nSim):
    exp1 = st.expon.rvs(scale=MU1, size=nSim)
    exp2 = st.expon.rvs(scale=MU2, size=nSim)
    menor = exp1 < exp2
    return np.count_nonzero(menor) / nSim
```

O código a seguir compara o valor de probabildade simulado pela função **pExpMenor** com o valor teórico.

```
mu1 = 1000
mu2 = 500
nSim = 10000

probT = (1/mu1)/(1/mu1+1/mu2)
t1 = time.perf_counter()
probS = pExpMenor(mu1, mu2, nSim)
t2 = time.perf_counter()
print('Probabilidade simulada:  {:.4f}'.format(probS))
print('Probabilidade teórica: {:.4f}'.format(probT))
print('Tempo de simulação: {:.4f}'.format(t2-t1))
```

```
Probabilidade simulada:  0.3360
Probabilidade teórica: 0.3333
Tempo de simulação: 0.0071
```

Distribuição da soma de N variáveis aleatórias exponenciais

Sejam $X_1, X_2, \cdots, X_N$ variáveis aleatórias independentes com média $mu_1 = 1/L$
$X = X_1 + X_2 + \cdots + X_N$ tem distribuição Erlang com parâmetros N e L.

- $f_X(x) = \frac{L^N x^{N-1} e^{-Lx}}{\tau(N)}$
- $F_X(x) = 1 - \sum_{j=0}^{N-1} e^{-Lx} \frac{(Lx)^j}{j!}$

Com a biblioteca Scipy temos:

- P[$X \leq x$] = st.gamma.cdf(x, a=N, scale=MU)

Algoritmo interativo

Podemos calcular simular a CDF da soma de N variáveis aleatórias exponencias com média MU com o seguinte algoritmo interativo:

- Iniciar a variável $deuCerto$ com zero.
- Sortear o array EXP com N variáveis aleatórias com média MU.
- Calcular o array $soma$ contendo o somatório de EXP, ou seja, a soma de N variáveis aleatórias exponenciais com média MU.
- Incrementar $deuCerto$, e a soma for menor do que o valor x para o qual queremos calcular a CDF (passado com argumento).
- Retornar $deuCerto$ dividido por $nSim$.

```
def somaExpCDFI(x, N, MU, nSim):
    deuCerto = 0
    for i in range(nSim):
        EXP = st.expon.rvs(scale=MU, size=N)
        soma = sum(EXP)
        if soma <= x:
            deuCerto = deuCerto + 1
    return deuCerto/nSim
```

O código a seguir compara o valor de probabildade simulado pela função **somaExpCDFI** com o valor da CDF da variável gamma calculado pela biblioteca Scipy.

```
x = 25
N = 10
MU = 3
```

```
nSim = 100000

probT = st.gamma.cdf(x, a=N, scale=MU)
t1 = time.perf_counter()
probS = somaExpCDFI(x, N, MU, nSim)
t2 = time.perf_counter()
print('Probabilidade simulada:  {:.4f}'.format(probS))
print('Probabilidade teórica: {:.4f}'.format(probT))
print('Tempo de simulação: {:.4f}'.format(t2-t1))
```

```
Probabilidade simulada:  0.3265
Probabilidade teórica: 0.3255
Tempo de simulação: 14.8270
```

Algoritmo vetorial

Podemos calcular simular a CDF da soma de N variáveis aleatórias exponencias com média MU com o seguinte algoritmo vetorial implementado com a biblioteca numpy:

Sortear a matriz EXP com $nSim$ linhas e N colunas.

- Observação: Cada linha corresponde a uma simulação e contém os valores das N exponencias sorteadas com média MU.
- Dica: usar np.random.exponential(MU, [nSim, N])

Calcular o array $soma$ contendo o somatório de cada linha da matriz EXP, ou seja, cada elento tem a soma de N variáveis aleatórias exponenciais com média MU.

- Dica: usar np.sum(), observando que tem que somar as linhas.

Calcular o array $menor$ que contém $True$ para cada linha da matriz cuja soma menor seja do que o valor x para o qual queremos calcular a CDF (passado com argumento).

- Dica: (soma <= x)

Retornar a quantidade de elementos no vetor $menor$ contendo valor True, dividida por $nSim$.

- Dica: usar a função np.count_nonzero para contar a quantidade de elementos com valor True.

```
def somaExpCDFV(x, N, MU, nSim):
    # Gerar uma matriz nSim x N de valores exponenciais
    EXP = np.random.exponential(MU, [nSim, N])

    # Calcular a soma das variáveis exponenciais ao longo de cada linha
    soma = np.sum(EXP, axis=1)

    # Verificar onde a soma é menor ou igual a x
    menor = soma <= x
```

```
    # Retornar a proporção de casos onde a soma foi menor ou igual a x
    return np.count_nonzero(menor) / nSim
```

O código a seguir compara o valor de probabildade simulado pela função **somaExpCDFV** com o valor da CDF da variável gamma calculado pela biblioteca Scipy.

```
x = 25
N = 10
MU = 3
nSim = 100000

probT = st.gamma.cdf(x, a=N, scale=MU)
t1 = time.perf_counter()
probS = somaExpCDFV(x, N, MU, nSim)
t2 = time.perf_counter()
print('Probabilidade simulada:  {:.4f}'.format(probS))
print('Probabilidade teórica: {:.4f}'.format(probT))
print('Tempo de simulação: {:.4f}'.format(t2-t1))
```

```
Probabilidade simulada:  0.3245
Probabilidade teórica: 0.3255
Tempo de simulação: 0.0401
```

Entrega

Completar o algoritmo. Imprimir para PDF. Enviar no AVA.