TEFE 2017

Suplemento da 1ª aula do tópico 2 - versão Python

suplemento elaborado por Danilo Lessa Bernardineli

Esse suplemento aborda geração de números pseudo-aleatórios com distribuição uniforme, o uso do desvio-padrão para determinar a incerteza dos resultados de simulações, o uso de condições lógicas em matrizes e a criação de funções.

Início - incorporação das bibliotecas¶

Em Python, é sempre necessário incorporar as bibliotecas a serem utilizadas:

```
import numpy as np # Biblioteca para manipulação numérica
import matplotlib.pyplot as plt # Biblioteca para visualização e gráficos
```

Geração de números pseudo-aleatórios com distribuição uniforme¶

In [2]:

```
# A geração de números pseudo-aleatórios uniformes no Python
# é feita usando o np.random.rand
N = 1000
L = 1
x = (np.random.rand(N) - 0.5) * L
```

In [3]:

```
s0 = L / np.sqrt(12)
print("Desvio padrão calculado: %.2g" % np.std(x))
print("Desvio padrão teórico: %.2g" % s0)
print("Média: %.2g" % np.mean(x))
print("Desvio padrão da média: %.2g" % np.std(x) / np.sqrt(N) )
```

```
Média: 0.0048
Desvio padrão calculado: 0.3
Desvio padrão teórico: 0.29
Desvio padrão da média: 0.0093
```

In [4]:

```
plt.hist(x)
plt.title("Distribuição Uniforme, %s dados" % N)
plt.show()
```

Calculando a incerteza do desvio padrão obtido nas simulações¶

In [5]:

```
# Vamos fazer nREP=50 repetições de conjuntos com N=1000 dados
N=1000
nREP=50
L=1
x = (np.random.rand(N, nREP) - 0.5) * L

# Em seguida, vamos calcular os desvios-padrões, sx, de cada um desses conjuntos
# e depois o desvio-padrão dos valores de sx, que é a incerteza de cada valor de sx.
# Note o parametro axis
# quando ele não é declarado, a operação é sobre todos os elementos
# quando é declarado, ele opera no respectivo eixo da matriz multi-dimensional
# ou em outras palavras, axis=0 opera nas linhas (N nesse caso)
# e axis=1 opera nas colunas (nREP nesse caso)
sx = np.std(x, axis=0)

print("N: %s \t nREP: %s \t L: %s" % (N, nREP, L))
print("Desvio padrão dos desvios padrões: %.2g" % np.std(sx))
print("Média dos desvios padrões: %.3g" % np.mean(sx))
print("Desvio padrão da média dos desvios padrões: %.2g" % (np.std(sx) / nREP))
```

```
N: 1000       nREP: 50       L: 1
Desvio padrão do desvios padrões: 0.0044
Média dos desvios padrões: 0.288
Desvio padrão da média dos desvios padrões: 8.8e-05
```

Contagem de elementos dado uma condição¶

In [11]:

```
N = 1000
L = 1
x = (np.random.rand(N) - 0.5) * L

# valores_dentro é uma matriz multidimensional de verdadeiros e falsos
# a condição de verdadeiro é de que o valor absoluto seja menor que L/sqrt[12]
valores_dentro = np.abs(x) <= L / np.sqrt(12)

# cada item verdadeiro contribui em 1 para a soma, e cada falso contribui em 0
ndentro = np.sum(valores_dentro)
print("Proporção de x dentro de 1 desv. pad: %s" % (ndentro / N))
```

```
Proporção de x dentro de 1 desv. pad: 0.553
```

Funções no Python¶

*No Python, é possível fazer funções com argumentos opcionais ao igualar ele a um valor padrão após o nome da variável na definição da função. No exemplo abaixo, a função **calcular** tem com argumentos padrões **N=1000**, **nREP=50** e **L=1**, mas estes valores podem ser escolhidos na chamada na função.*

In [6]:

```
def calcular(N=1000, nREP=50, L=1):
    x = (np.random.rand(N, nREP) - 0.5) * L
    desvio_padroes = np.std(x, axis=0)

    sm = np.mean(desvio_padroes) # média dos desvios padrões
    s_sm = np.std(desvio_padroes) / nREP # desvio padrão da média dos desvios padrões

    return (sm, s_sm) # retorna uma lista de 2 elementos para quem chamou
```

In [7]:

```
# Para chamar a função calcular usando os valores padrões
calcular()
```

Out[7]:

```
(0.28750739955756122, 8.8294461597535577e-05)
```

In [8]:

```
# Para chamar a função calcular escolhendo os valores de N e nREP (os dois primeiros parâmetros da função)
calcular(3000, 100)
```

Out[8]:

```
(0.28811702766136094, 2.3415929966102642e-05)
```

In [9]:

```
# Para chamar a função calcular escolhendo os valores de L, N e nREP (em qualquer ordem)
calcular(L=3, N=1200, nREP=70)
```

Out[9]:

```
(0.86647665673782093, 0.00014894908006580842)
```

In [10]:

```
# Para chamar a função calcular registrando os valores obtidos nas variáveis media_sx e sigma_media_sx
media_sx, sigma_media_sx = calcular()
print("Desvio padrão médio: %.3g" % media_sx)
print("Desvio padrão da média do desvio padrão %.3g" % sigma_media_sx)
```

```
Desvio padrão médio: 0.289
Desvio padrão da média do desvio padrão 8.76e-05
```