TEFE 2017

Aula 3 - Geração de dados simulados - versão Octave e Matlab

Definindo variáveis ¶

O Octave e Matlab são linguagens orientadas a objetos com tipagem fraca, e portanto variáveis podem ser atribuídas livremente sem especificar o tipo. Note que se o ponto e vírgula ao final dos comandos for omitido, o conteúdo da variável será mostrada na tela, como no caso da atribuição à variável **b** no exemplo abaixo.

```
>> a=10;
>> b=20
b =  20
>> c = (1:5);  % sequencia com 5 elementos (do 1 ate o 5)
```

Exibindo variáveis ¶

Os tipos e o conteúdo das variáveis podem ser exibidas utilizando-se do comando **whos**

É possível inspecionar variáveis individuais digitando apenas o nome da variável

---

```
>> whos
Variables in the current scope:
  Attr Name        Size                     Bytes  Class
  ==== ====        ====                     =====  =====
       a           1x1                          8  double
       b           1x1                          8  double
       c           1x5                         24  double

Total is 7 elements using 40 bytes


>> c
c =

  1   2   3   4   5
```

Ajuda ¶

Para obter a ajuda sobre alguma função basta digitar **help** ou **doc** antes do nome do função. A diferença entre essas duas opções de ajuda é que o comando **help** fornece uma ajuda na forma de texto sem formatação na própria janela de comandos, ao passo que o comando **doc** abre uma janela com texto formatado dentro de uma janela própria para isso.

```
>> help fprintf
>> doc fprintf
```

Matrizes¶

O Matlab e o Octave assumem de que todas as variáveis são matrizes (que podem ter até mesmo mais de duas dimensões). É por este motivo que o tamanho de uma variável contendo apenas um número é exibida como tendo dimensão 1x1. Em geral, todos os comandos e funções estão prontos para lidar com matrizes. Além disso, quando apenas um número é passado no parâmetro referente ao tamanho, essas funções geralmente assumem que se trata de uma matriz quadrada daquele tamanho.

```
>> d = zeros(3)
d =

  0   0   0
  0   0   0
  0   0   0

>> e = zeros(2,4)
e =

  0   0   0   0
  0   0   0   0
```

Colocando e acessando valores em matrizes¶

Aqui colocaremos valores predefinidos em matrizes e vetores:

```
>> % para atribuir os valores 290, 310, 290, 320, 280 e 330 ao vetor linha x
>> x = [290 310 290 320 280 330];
>> x
x =

  290   310   290   320   280   330

>> % o operador ' (aspas simples) pode ser usado para transpor a matriz
>> x'
ans =

  290
  310
  290
  320
  280
  330

>> % para ver o conteudo do quarto elemento do vetor x
>> x(4)
ans =  320

>> mean( x )
ans =  303.33
>> std( x )
ans =  19.664
```

```
>> % para atribuir os mesmos valores a um vetor coluna y
>> y = [290; 310; 290; 320; 280; 330]
y =

  290
  310
  290
  320
  280
  330

>> % para criar uma matriz 2x3 (o ponto e virgula indica a mudança de linha)
>> z = [290 310 290; 320 290 330]
z =

  290   310   290
  320   290   330

>> % a media por padrao e calculada ao longo da primeira dimensao nao unitaria
>> mean(z)
ans =

  305   300   310
>> % para saber como definir a dimensao a ser considerada no comando mean,
>> % use "help mean" ou "doc mean"
```

Numeros aleatórios¶

A geração de números aleatórios gaussianos pode ser feita com o comando **randn**, que gera dados com distribuição gaussiana de valor verdadeiro 0 e desvio-padrão 1.

O comando **rand** (sem o "n" no final) gera dados com distribuição uniforme entre 0 e 1.

```
>> randn(1,3)
ans =

  0.076685  -0.318310   0.189178

>> rand(1,4)
ans =

  0.72333   0.76562   0.93059   0.37179


>> % para gerar um dado com com valor verdadeiro 300 e desvio-padrao 50:
>> 300 + 50*randn
ans =  335.00

>> % para gerar 100 dados com essas caracteristicas
>> z = 300 + 50 * randn(100,1);
>> mean( z )
ans =  302.07
>> std( z )
ans =  50.332
```

Limpar variáveis¶

As variáveis na memória podem ser removidas pelo comando **clear**. É possível limpar apenas uma variável indicando o nome dela após o **clear** ou um conjunto de variáveis com nomes parecidos usando o ***.

```
>> clear
>> clear x
>> clear x*
```

Loops¶

Loops podem ser feitos pelo uso dos comandos **for...end** ou **while...end**. Abaixo exemplos do uso de loops com o comando for são apresentados. Para saber sobre o comando while veja a ajuda com o comando **doc while**. A indentação é opcional.

```
>> x0 = 300;
>> sigma = 50;
>> N = 6;
>> for i=1:6
       x(i) = x0 + sigma * randn;
    end
>> x
x =

  311.14   267.12   310.01   318.53   336.45   393.29
```

```
>> % gerar 1000 repeticoes de conjuntos de 32 dados guardando a media e o desvio-padrao amostral
>> nREP = 1e3;
>> N = 32;
>> x0 = 300;
>> sigma = 50;
>> xm = zeros( nREP,1 );
>> sx = zeros( nREP,1 );
>> for qR = 1 : nREP
      x = zeros(N,1);
      for i = 1 : N
        x(i) = x0 + sigma * randn;
      end
      xm(qR) = mean( x );
      sx(qR) = std( x );
   end
```

Visualização¶

Para mostrar as médias e os desvios-padrões calculados com a rotina anterior:

```
>> figure
>> plot( xm, '*' )
>> figure
>> plot( sx, '*r' )
```
