MATLAB 科学计算语言与应用

Lecture 4: Advanced Methods

Outline

(1) **Probability and Statistics**

(2) Data Structures

(3) Images

(4) File I/O

Random Numbers

- Many probabilistic processes rely on random numbers
- MATLAB contains the common distributions built in
  - » `rand`
    - ➢ draws from the uniform distribution from 0 to 1
  - » `randn`
    - ➢ draws from the standard normal distribution (Gaussian)
  - » `random`
    - ➢ can give random numbers from many more distributions
    - ➢ see **help random**
- You can also seed the random number generators

```
rng( 'default' );
rng(sd);
rng( 'shuffle' );
```

Statistics

- Whenever analyzing data, you have to compute statistics

```
» scores = 100*rand(1,100);   % random data
```

- Built-in functions
  - ➢ mean, median, mode
    - **mean**: 求算数平均值，average;
    - **median**: 求中值，中位数。中值是一组数值中取值为中间的数值
    - **mode**: 返回向量、数组出现频率最多的数值。

      mode(X)，或 mode(X,1) 返回每列出现频率最高的值组成的行向量。 当有多个值有相等的频率时，mode返回这些值中最小的值。

      mode(X,2)返回每行出现频率最高的值组成的行向量。 当有多个值有相等的频率时，mode返回这些值中最小的值。

Statistics

- Built-in functions
  - **max, min**

    y=max(X)

    [y, k]=max(X)

    [y, k]=max(X, [], dim)

    y=max(A, B)

    - Try and check results

```
>> A=[12 1 -6 24;-4 23 12 0;2 -3 18 6;45 3 16 -7]
>> yx=max(A)
>> [yx,kx]=max(A)
>> [yx kx]=max(A,[],1)
>> [yx kx]=max(A,[],2)
>> A=[1 5 6;7 3 1;3 7 4]
>> B=[2 9 4;9 1 3;-1 0 3]
>> y=max(A,B)
```

Statistics

- 排序

  [Y, I]=sort(A, dim, ‘mode’)

  - Try and check results

  >> X=[1 12 23 7 9 -5 30]

  >> Y=sort(X)

  >> A=[0 9 2;7 3 1;-1 0 3]

  >> Y1=sort(A)

  >> Y2=sort(A,1,'descend')

  >> Y3=sort(A,2,'ascend')

  - try ‘**sortrows**’

- 求和、求积、累加、累乘

Y=**sum**(X)

Y=sum(A);

Y=**prod**(X)

Y=prod(A);

Y=sum(A, 2)

Y=prod(A, 2)

Y=**cumsum**(X)

Y=cumsum(A)

Y=cumsum(A, 2)

Y=**cumprod**(X);

Y=cumprod(A)

Y=cumprod(A, 2)

Statistics

- standard deviation（标准差）、correlation（相关系数）

d=std(X);

D=std(A);

D=std(A, flag, dim)

flag: =0 or =1, 两种标准差公式（除以N-1、除以N）

dim: =1 or =2，对列元素计算，对行元素计算

R=corrcoef(X, Y)

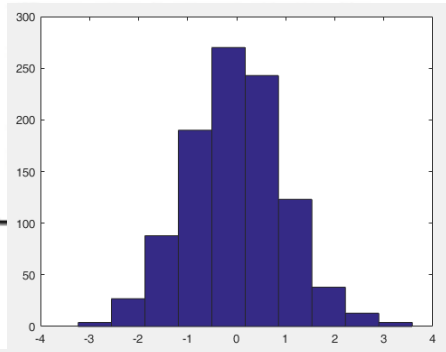
求两个长度相等向量之间的相关系数

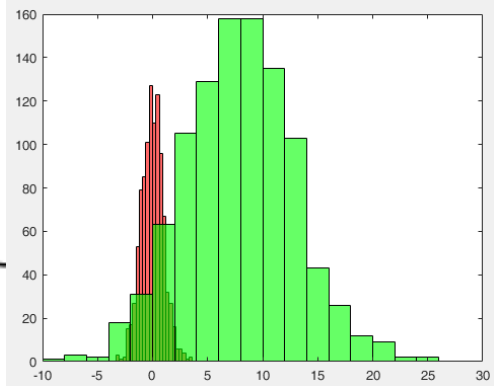
R=corrcoef(A)

求矩阵每列之间的相关系数

Functions: overloading

- We can alter the given distributions
  - » `y=rand(1,100)*10+5;`
    - ➢ gives 100 uniformly distributed numbers between 5 and 15
  - » `y=floor(rand(1,100)*10+6);`
    - ➢ gives 100 uniformly distributed integers between 6 and 15. floor or ceil is better to use here than round
    - ➢ you can also use `randi([6,15],1,100)`
  - » `y=randn(1,1000)`
  - » `y2=y*5+8`
    - ➢ increases std to 5 and makes the mean 8





Statistics-Plotting

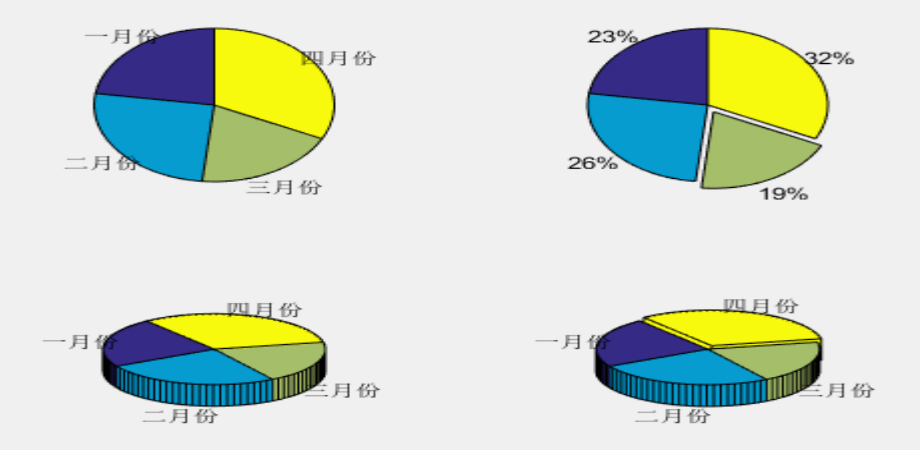
- 饼形图：显示各元素占总和的百分比

  pie(x, explode, ‘label’), pie3(x, explode, ‘label’)

  % explode是由0和 1组成的与x长度一致的向量，用于控制该部分是否从饼图中分离；label为标注。

例：已知一个服装店4个月的销售数据为x=[210 240 180 300]，分别用二维饼图和三维饼图显示数据。

```
>> x=[210 240 180 300];
>> subplot(2,2,1);pie(x,{'一月份','二月份','三月份','四月份'});
>> subplot(2,2,2);pie(x,[0 0 1 0]);
>> subplot(2,2,3);pie3(x,{'一月份','二月份','三月份','四月份'});
>> subplot(2,2,4);pie3(x,[0 0 0 1],{'一月份','二月份','三月份','四月份'});
```



Statistics-Plotting

- 直方图：统计、显示已知数据的分布

  histogram(X); %自动按默认方式计算直方图

  histogram(X, N)； %分成N段显示

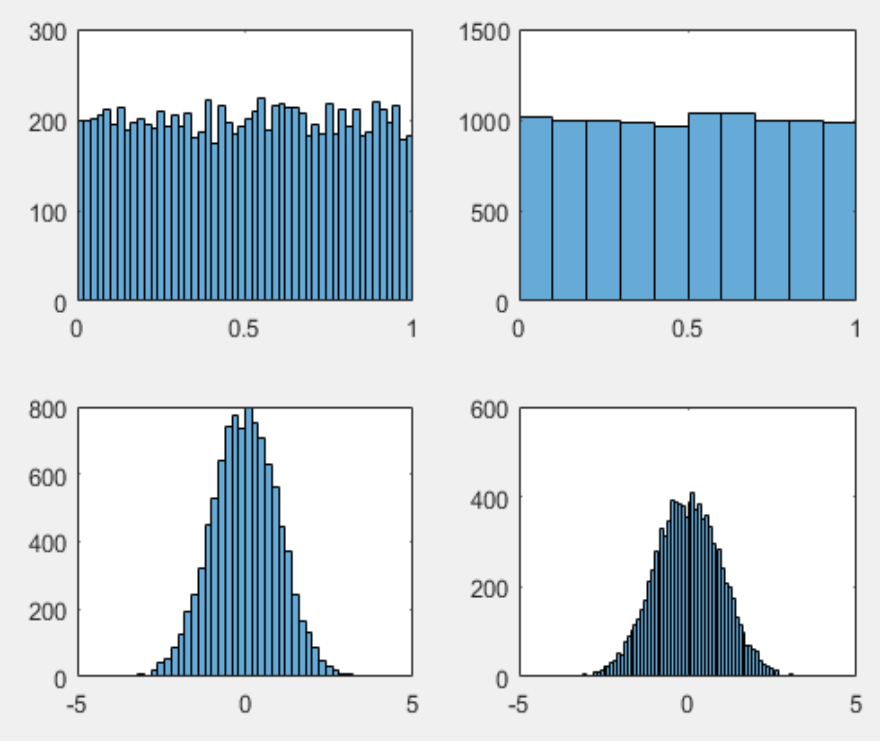
  histogram(X, edges-vector); % 按edges-vevtor定义的边缘分段

  histogram(…, ‘Normalization’, ‘mode’)% 定义归一化方式。

  N=histogram(…..) %N中保存直方图全部属性

例：绘制用rand(10000, 1)和randn(10000,1)命令产生的数据的直方图。

```
>> y1=rand(10000,1);
>> y2=randn(10000,1);
>> subplot(2,2,1);histogram(y1,50);
>> subplot(2,2,2);histogram(y1,[0:.1:1]);
>> subplot(2,2,3);histogram(y2);
>> subplot(2,2,4);histogram(y2,[-5:.1:5]);
>> figure;
>> N1=histogram(y1,10)；
>> N1.values    %每一段中数据个数
```



Exercise: Probability

- We will simulate Brownian motion（布朗运动） in 1 dimension. Call the script ‘brwn’
- Make a 10,001 element vector of zeros
- Write a loop to keep track of the particle’s position at each time
- Assume motion starts from position 0. To get the new position, pick a random number, and if it’s <0.5, go left; if it’s >0.5, go right. Count how many times each position is visited.
- Plot a 50 bin histogram of the positions.

Outline

(1) Probability and Statistics

(2) **Data Structures**

(3) Images

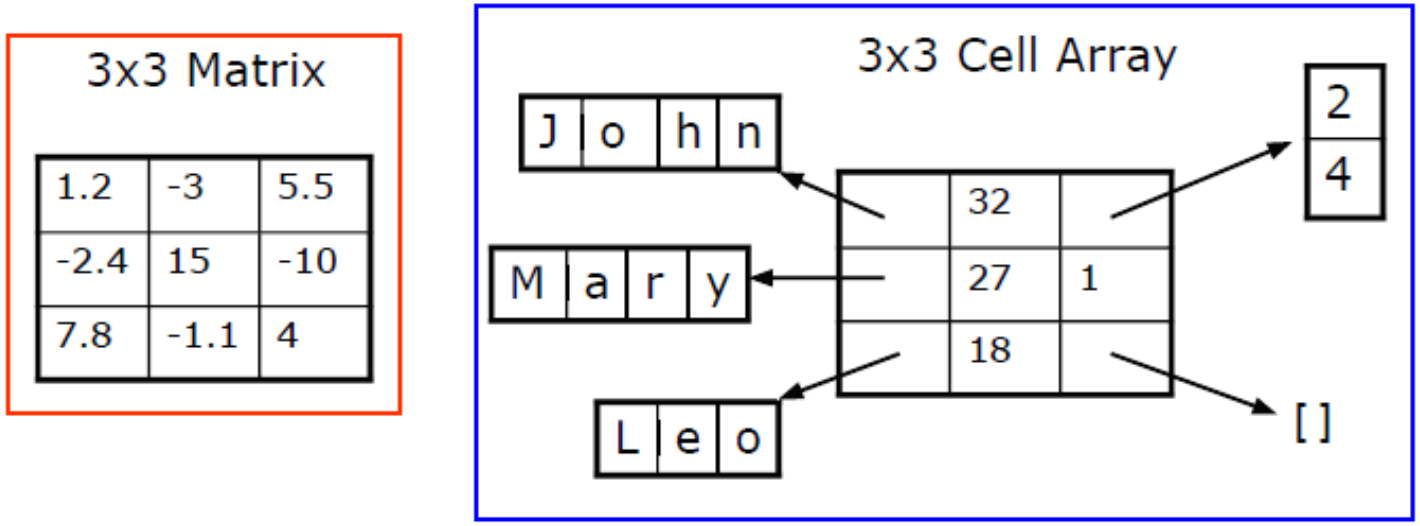
(4) File I/O

Advanced Data Structures

- We have used 2D matrices
  - ➢ Can have n-dimensions (e.g., RGB images)
  - ➢ Every element must be the same type (ex. integers, doubles, characters…)
  - ➢ Matrices are space-efficient and convenient for calculation
  - ➢ Large matrices with many zeros can be made sparse
    - – More on this later this lecture
- Sometimes, more complex data structures are more appropriate
  - ➢ Cell array: it's like an array, but elements don't have to be the same type
  - ➢ Structs: can bundle variable names and values into one structure
    - – Like object oriented programming in MATLAB

Cells: organization

• A cell is just like a matrix, but each field can contain anything (even other matrices):



• One cell can contain people's names, ages, and the ages of their children

• To do the same with matrices, you would need 3 variables and padding

Cells: initialization

- To initialize a cell, specify the size
  - `» a=cell(3,10);`
    - ➢ a will be a cell with 3 rows and 10 columns
- or do it manually, with curly braces {}
  - `» c={'hello world',[1 5 6 2],rand(3,2)};`
    - ➢ c is a cell with 1 row and 3 columns
- Each element of a cell can be anything
- To access a cell element, use curly braces {}
  - `» a{1,1}=[1 3 4 -10];`
  - `» a{2,1}='hello world 2';`
  - `» a{1,2}=c{3};`

```
>> A={[1+2i],'Matlab';1:6,{[1 2;3 4],'cell'}}
A =
  2×2 cell 数组
    [1.0000 + 2.0000i]    'Matlab'
    [1×6 double]    {1×2 cell}
```

```
>> B{1,1}=1+2i
>> B{1,2}='Matlab'
>> B{2,1}=1:6
>> B{2,2}={[1 2;3 4],'cell'}
B =
  2×2 cell 数组
    [1.0000 + 2.0000i]    'Matlab'
    [1×6 double]    {1×2 cell}
```

```
>> A=cell(2)
A =
  2×2 cell 数组

    []    []
    []    []
>> A{1,1}=1+2i
>> A{1,2}='Matlab'
>> A{2,1}=1:6
>> A{2,2}={[1 2;3 4],'cell'}
A =
  2×2 cell 数组
    [1.0000 + 2.0000i]    'Matlab'
    [1×6 double]    {1×2 cell}
```

Cell: Access

- 用{ }提取元胞数组的元素数据

```
>> a1=A{2,1}
a1 =
     1     2     3     4     5     6

>> a4=A{4}
a4 =
  1×2 cell 数组
    [2×2 double]    'cell'
```

- 用（）只能得到元胞类型，不能得到元胞数据

```
>> a4=A(4)
a4 =
  cell
    {1×2 cell}
```

Cell: Access

- 用**deal**函数提取多个元胞元素的数据

  [c1,c2,c3]=deal(A{1:3})

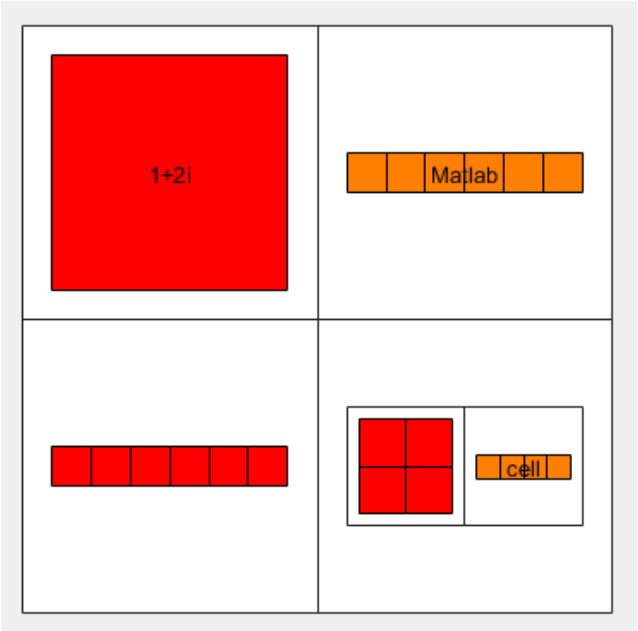
  [c1,c2,c3,c4]=deal(A{:})

- 用**celldisp**函数显示元胞数组中的详细数据内容

  比较输入**celldisp(A)**和直接输入A的区别

- 用**cellplot**函数以图形方式显示元胞结构

  >> cellplot(A)



Exercise: Cells

- Write a script called sentGen
- Make a 2x3 cell, and put three names into the first row, and adjectives into the second row
- Pick two random integers (values 1 to 3)
- Display a sentence of the form '[name] is [adjective].'
- Run the script a few times

Structs

- Structs allow you to name and bundle relevant variables
  - Like C-structs, which are containers with fields

- To initialize an empty struct:

```
» s=struct;
```

  - size(s) will be 1x1
  - initialization is optional but is recommended when using large structs

- To add fields

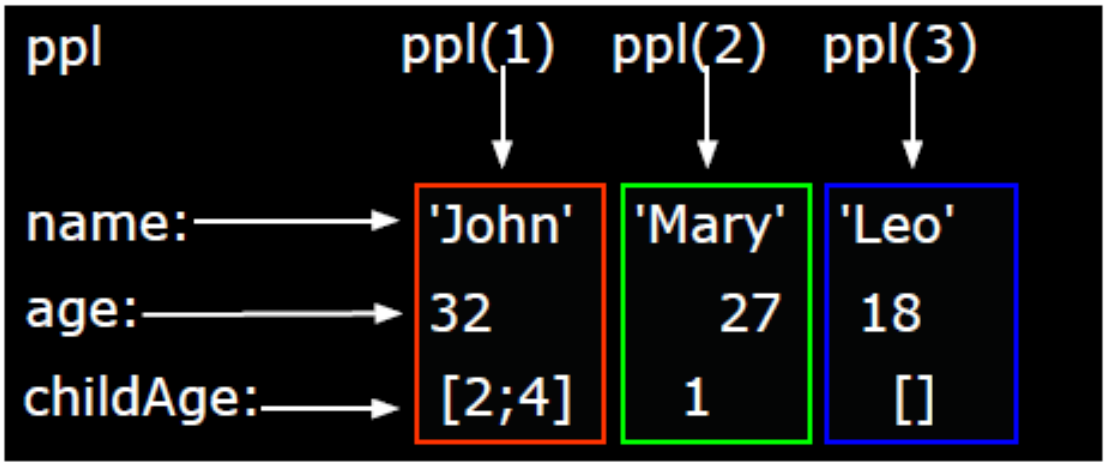
```
» s.name = 'Leo';
» s.age = 18;
» s.childAge = [];
```

  - Fields can be anything: matrix, cell, even struct
  - Useful for keeping variables together

- For more information, see **help struct**

Struct Arrays

- To initialize a struct array, give field, values pairs
  - » `ppl=struct('name',{'John','Mary','Leo'},...`
    `'age',{32,27,18},'childAge',{[2;4],1,[]});`
    - ➢ size(ppl)=1x3
    - ➢ every cell must have the same size
  - » `person=ppl(2);`
    - ➢ person is now a struct with fields name, age, children
    - ➢ the values of the fields are the second index into each cell
  - » `ppl(3)=s;`
    - ➢ adds struct (fields must match)
  - » `person.name`
    - ➢ returns 'Mary'
  - » `ppl(1).age`
    - ➢ returns 32



例：构建一个班级学生信息结构数组

- 有三个元素（每个元素对应一个学生）
- 每个元素（学生）有三个字段：姓名，学号，成绩

```
>> tongxin(1).name='Zhang san';
>> tongxin(1).ID='2020134001';
>> tongxin(1).score=[98 95 90 99 87];
>> tongxin(2).name='Li si';
>> tongxin(2).ID='2020134002';
>> tongxin(2).score=[88 95 91 90 97];
>> tongxin(3).name='Wang wu';
>> tongxin(3).ID='2020134003';
>> tongxin(3).score=[81 75 61 80 87];
```

```
>> tongxin

tongxin =

  包含以下字段的 1×3 struct 数组:

    name
    ID
    score
```

```
>> tongxin(1)=struct('name','Zhang san','ID','2020134001','score',[98 95 91 89]);
>> tongxin(2)=struct('name','Li si','ID','2020134002','score',[88 95 91 90 97]);
>> tongxin(3)=struct('name','Wang wu','ID','2020134003','score',[81 75 61 80 87]);
```

```
>> tongxin

tongxin =

  包含以下字段的 1×3 struct 数组:

    name
    ID
    score
```

```
>> tongxin(1)

ans =

  包含以下字段的 struct:

      name: 'Zhang san'
        ID: '2020134001'
     score: [98 95 91 89]
```

Structs: Access

- To access 1x1 struct fields, give name of the field
  - » `stu=s.name;`
  - » `a=s.age;`
    - ➢ 1x1 structs are useful when passing many variables to a function. Put them all in a struct, and pass the struct

- To access nx1 struct arrays, use indices
  - » `person=ppl(2);`
    - ➢ person is a struct with name, age, and child age
  - » `personName=ppl(2).name;`
    - ➢ personName is 'Mary'
  - » `a=[ppl.age];`
    - ➢ a is a 1x3 vector of the ages; this may not always work, the vectors must be able to be concatenated

- 使用“.”直接访问
- 使用函数getfield获取结构体内部数据
- 使用函数fieldnames获取结构体所有字段

```
>> name1=getfield(tongxin,{1},'name')
name1 =
Zhang san

>> s1=getfield(tongxin,{1},'score',{2})
s1 =
    95
```

```
>> name1=tongxin(1).name
name1 =
Zhang san

>> ID1=tongxin(1).ID
ID1 =
2020134001

>> score1=tongxin(1).score
score1 =
    98    95    91    89
```

```
>> x=fieldnames(tongxin)
x =
  3×1 cell 数组
    'name'
    'ID'
    'score'
```

```
>> whos tongxin x
  Name          Size          Bytes    Class     Attributes
  tongxin       1x3            1414    struct
  x             3x1             358    cell
```

- 使用setfield函数对结构体数据进行修改
- 使用函数rmfield删除结构体的字段

```
>> tongxin=setfield(tongxin,{1},'ID','2020234001')
>> tongxin(1)
ans =
  包含以下字段的 struct:
     name: 'Zhang san'
       ID: '2020234001'
    score: [98 95 91 89]
```

```
>> tongxin=rmfield(tongxin,'name')
tongxin =
  包含以下字段的 1×3 struct 数组:
    ID
    score

>> tongxin(1)
ans =
  包含以下字段的 struct:
       ID: '2020234001'
    score: [98 95 91 89]
```

Exercise: Structs

- Modify the script sentGen
- Create a struct array with a field “name” and a field “adj” containing the values from the previous cell array
- Do not create it from scratch! Use the previously defined cell array!
- Modify the display command to use the struct array
- Run the script a few times

例：NASA电池数据集

- 锂电池充放电实验数据，全生命周期，用于储能系统故障检测与健康状态监控算法研究

```
>> load B0007
>> fieldnames(B0007)
>> B0007.cycle
>> B0007.cycle(1)
>> B0007.cycle(1).data
```

Outline

(1) Probability and Statistics

(2) Data Structures

(3) **Images**

(4) File I/O

Handles

- Manipulate graphics objects using ‘handles’
  - » L=plot(1:10,rand(1,10));
    - ➢ gets the handle for the plotted line
  - » A=gca;
    - ➢ gets the handle for the current axis
  - » F=gcf;
    - ➢ gets the handle for the current figure
- To see the current property values, use get
  - » Ldata=get(L); %返回一个struct
  - » yVals=get(L, ‘YData’); %返回一个vector
- To change the properties, use set
  - » set(A,'FontName','Arial','XScale','log');
  - » set(L,'LineWidth',1.5,'Marker','*');
- Everything you see in a figure is completely customizable through handles

Reading/Writing Images

- Images can be imported as a matrix of pixel values

```
» im=imread('myPic.jpg');
» imshow(im);
```

- Matlab supports almost all image formats
  - ➢ jpeg, tiff, gif, bmp, png, …
  - ➢ see help imread for details (pixel format and types)
- To write an image, give:
  - ➢ rgb matrix (0 to 1 doubles, or 0 to 255 uint8)

```
» imwrite(rand(300,300,3),'t1.jpg');
```

  - ➢ indices and colormap

```
» imwrite(ceil(rand(200)*256),jet(256),'t2.jpg');
```

  - ➢ see help imwrite for more options

MATLAB's built-in images

```
AT3_1m4_01.tif          AT3_1m4_02.tif
AT3_1m4_03.tif          AT3_1m4_04.tif
AT3_1m4_05.tif          AT3_1m4_06.tif
AT3_1m4_07.tif          AT3_1m4_08.tif
AT3_1m4_09.tif          AT3_1m4_10.tif
    autumn.tif              bag.png
     blobs.png            board.tif
 cameraman.tif             canoe.tif
      cell.tif            circbw.tif
   circles.png           circuit.tif
     coins.png       concordaerial.png
concordorthophoto.png        eight.tif
    fabric.png         football.jpg
    forest.tif       gantrycrane.png
     glass.png           greens.jpg
   hestain.png             kids.tif
liftingbody.png            logo.tif
      m83.tif            mandi.tif
     moon.tif              mri.tif
  office_1.jpg          office_2.jpg
  office_3.jpg          office_4.jpg
  office_5.jpg          office_6.jpg
    onion.png            paper1.tif
    pears.png           peppers.png
  pillsetc.png              pout.tif
     rice.png           saturn.png
   shadow.tif       snowflakes.png
    spine.tif             tape.png
  testpat1.png            text.png
     tire.tif           tissue.png
    trees.tif  westconcordaerial.png
westconcordorthophoto.png
```

They are in C:\Program Files\MATLAB\R2014a\toolbox\images\imdata;

But you can read them in from your current directory:

```
>> im = imread('trees.tif');
>> imshow(im)
```

Outline

(1) Probability and Statistics

(2) Data Structures

(3) Images

(4) **File I/O**

Importing Data

- Matlab is a great environment for processing data. If you have a text file with some data:

```
jane joe jimmy
10 11 12
5 4 2
5 6 4
```

- To import data from files on your hard drive, use importdata

```
» a=importdata('textFile.txt');
```

➢ a is a struct with `data`, `textdata`, and `colheaders` fields

```
a =
            data: [3x3 double]
        textdata: {'jane'  'joe'  'jimmy'}
      colheaders: {'jane'  'joe'  'jimmy'}
```

```
» x=a.data;
» names=a.colheaders;
```

Importing Data

• With importdata, you can also specify delimiters（分隔符）.

For example, for comma separated values, use:

```
» a=importdata('filename', ',');
```

➢ The second argument tells matlab that the tokens of interest are separated by commas

• importdata is very robust, but sometimes it can have trouble. To read files with more control, use fscanf (similar to C/Java), textscan. See help for information on how to use these functions

Writing Excel Files

- Matlab contains specific functions for reading and writing Microsoft Excel files
- To write a matrix to an Excel file, use xlswrite

```
» xlswrite('randomNumbers',rand(10));
» xlswrite('randomNumbers',rand(10), 'Sheet1','C11:L20');
```

  - ➢ Sheet name and range optional

- You can also write a cell array if you have mixed data:

```
» C={'hello','goodbye';10,-2;-3,4};
» xlswrite('randomNumbers',C,'mixedData');
```

- See help xlswrite for more usage options

Reading Excel Files

- Reading excel files is equally easy
- To read from an Excel file, use xlsread

» [num,txt,raw]=xlsread('randomNumbers.xls');

➢ Reads the first sheet

➢ num contains numbers(matrix), txt contains strings(cell), raw is the entire cell array containing everything

» [num,txt,raw]=xlsread('randomNumbers.xls',... 'mixedData');

➢ Reads the mixedData sheet

» [num,txt,raw]=xlsread('randomNumbers.xls',-1);

➢ Opens the file in an Excel window and lets you click on the data you want!

- See help xlsread for even more fancy options

Reading ANY File

- You can read any file as binary data
- To read from a file, use fopen
  - » fid = fopen('fileName' , 'r');
    - ➢ Returns a handle to a file
  - » data = fread(fid, 10);
    - ➢ Reads the next 10 bytes from the file and stores them in data
  - » fseek(fid, 5, 0);
    - ➢ Moves forward 5 bytes from the current position
- See help fopen/fread/fwrite/ftell/fseek for even more fancy options