Introduction to MATLAB

Sajjad Bahrami

University of California, Riverside

Winter 2018

Topics

- **A Brief Introduction**
- **MATLAB Working Environment**
- **MATLAB MCMC Toolbox**

Introduction

- Stands for **MAT**rix **LAB**oratory
- The matrix-based MATLAB language lets you express math directly
- Interpreted language
- Scientific programming environment
- Very good tool for the manipulation of matrices
- Great visualisation capabilities
- Loads of built-in functions
- Easy to learn and simple to use
- In this tutorial, we get familiar how to use MATLAB in Statistics

How can you access to MATLAB?

- Licensed programming language, not free!
- UCR members have access to MATLAB
- Create an account in mathworks.com to download MATLAB
- Then, visit the mysoftware.ucr.edu and enter your UCR NetID and password
- Available software packages will be listed if you are enrolled in at least **one** class for the current or upcoming quarter
- Click on Annual License Key for MATLAB
- You are asked for your reason for downloading MATLAB (e.g. indicate a course for which you need MATLAB, then click continue)
- An email is sent to your Rmail containing Activation Code

Some Problems and Advantages

Problems:

- The algorithms are **proprietary**, which means you can not see the code of most of the algorithms you are using. (also makes it difficult/impossible for 3rd parties to extend the functionality of MATLAB)
- Quite **expensive**
- Mathworks puts restrictions on code **portability**, the ability to run your code on someone else's computer. (can be a nuisance considering that MATLAB releases a new version every 6 months)
- **Slower** than other programming languages

Advantages:

- Has a solid amount of functions
- Simulink is a product for which there is no good alternative yet
- Easier for beginners, because the package includes all you need, while for example in Python you need to install extra packages and an IDE
- Has a large scientific community; it is used on many universities

A Simple Comparison

- As you can see, there is a very subtle mistake in line 2 of the Python code. In the original code, row is a 1-d array. It looks like a row vector, but it doesn't have enough dimensionality to say if it is a row or a column – it is just a 1-d array. Since there is no second dimension, the transpose in the third line has no effect. MATLAB, on the other hand, makes no artificial distinction between scalar, 1-d, 2-d, and multidimensional arrays. (**MATLAB easier to work with**)

Python

```
>>> import numpy as np

# Create row vector
>>> row = np.array([1, 2, 3])
>>> row
array([1, 2, 3])

# Transpose
>>> col = row.T

# Compute inner product
>>> inner = np.dot(row,col)
>>> inner
14

# Compute outer product
>>> outer = np.dot(col,row)
>>> outer
14
```

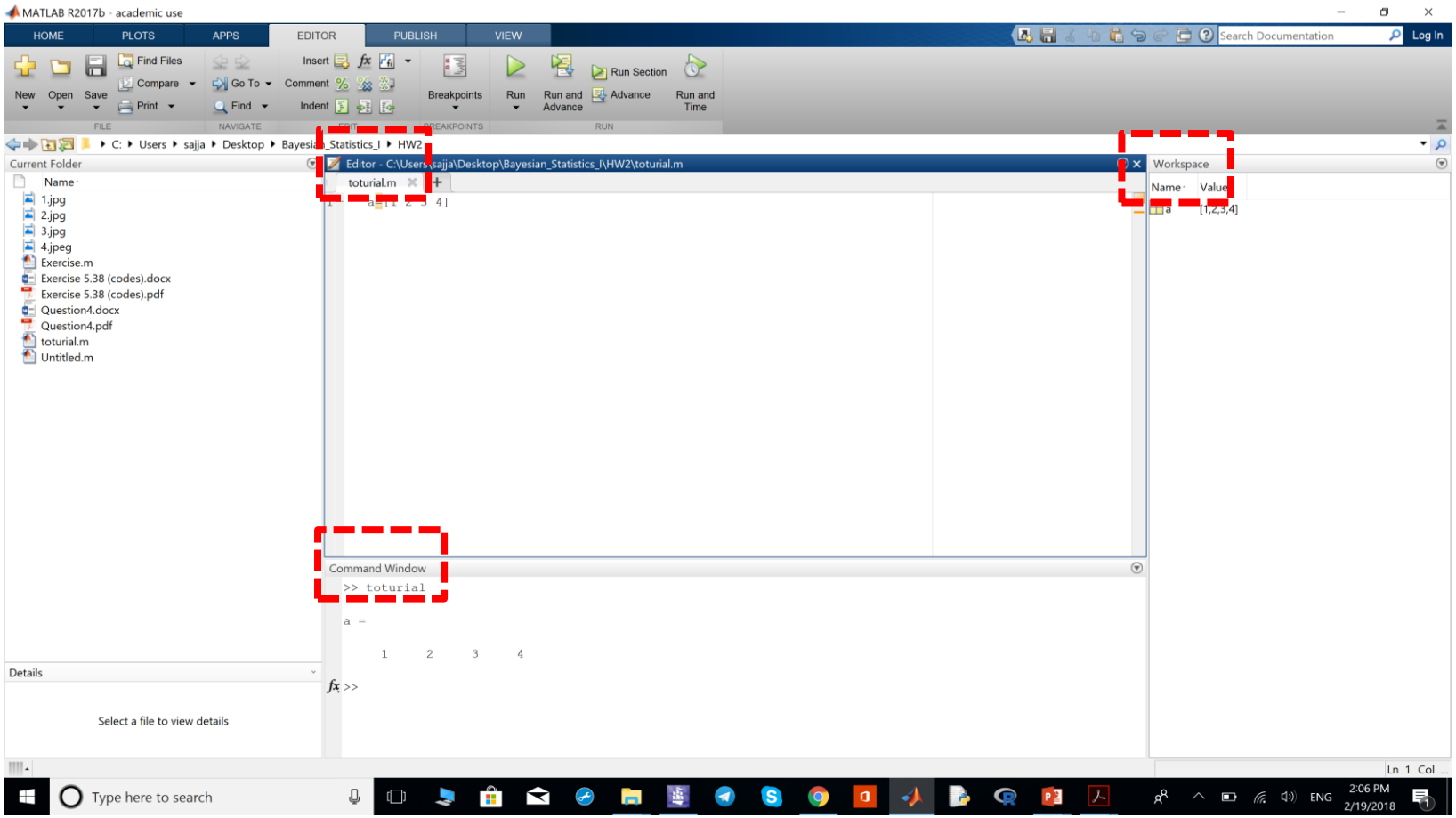
MATLAB

```
% Create row vector
>> row = [1 2 3]
row =
     1     2     3

% Transpose
>> col = row';

% Compute inner product
>> inner = row*col
inner =
    14

% Compute outer product
>> outer = col*row
outer =
     1     2     3
     2     4     6
     3     6     9
```

MATLAB Working Environment

- Workspace: View and make changes to the contents of the workspace
- Command Window: Run MATLAB statements (commands)
- M-file Editor: Creating, Editing, Debugging and Running Files

MATLAB Working Environment



Matrices

- While other programming languages work with numbers one at a time, MATLAB allows you to work with entire matrices quickly and easily

A few basic conventions for entering matrices:

- Separate the elements of a row with blanks or commas
- Use a semicolon, ; , to indicate the end of each row
- Surround the entire list of elements with square brackets, [ ]

```
>> a=[1 2 3;4 5 6]
a =
   1   2   3
   4   5   6
```

Matrices

- The element in row i and column j of A is denoted by A(i,j)
- The Colon Operator:
  - 1:10 is a row vector containing the integers from 1 to 10. To obtain no unit spacing, specify an increment, e.g. 100:-7:50
  - Subscript expressions involving colons refer to portions of a matrix, e.g. A(1:k,j) is the first k elements of the jth column of A
- Concatenating Matrices:

  B=[A A+32; A+48 A+16]

- Deleting rows or columns:

```
>> a=[1 2 3;4 5 6]
a =
   1   2   3
   4   5   6
>> a(:,2)=[]
a =
   1   3
   4   6
```

Matrices

- Some matrix functions: sum(A), A’, diag(A), zeros(4,4), ones(4,4), rand(4,4), randn(4,4), det(A), inv(A), eig(A)
- Some mathematical functions: abs, sqrt, exp, and sin
- For a list of mathematical and matrix functions, type:

>> help elfun (Elementary math functions)

>> help specfun (Specialized math functions)

>> help elmat (Elementary matrices and matrix manipulation)

Operators for Matrices

- Operators

  + - * / ^

+	Addition
-	Subtraction
.*	Element-by-element multiplication
./	Element-by-element division
.\	Element-by-element left division
.^	Element-by-element power
.'	Unconjugated array transpose

Multivariate Data

- MATLAB uses column-oriented analysis for multivariate statistical data. Each column in a data set represents a variable and each row an observation. The (i,j)th element is the ith observation of the jth variable, for example:

```
>> a=[1 2 3;4 5 6]

a =

    1    2    3

    4    5    6

>> mean(a)

ans =

   2.5000    3.5000    4.5000
```

Flow Control

- if, else, and else if

```
>> if rem(n,2) ~= 0
        M = 0
    elseif rem(n,4) ~= 0
        M = 1
    else
        M = 2
    end
```

- switch and case

```
>> n = input('Enter a number: ‘);
 switch n
 case -1
 disp('negative one’)
case 0
 disp('zero’)
case 1
 disp('positive one’)
otherwise disp('other value’)
end
```

Flow Control

- For

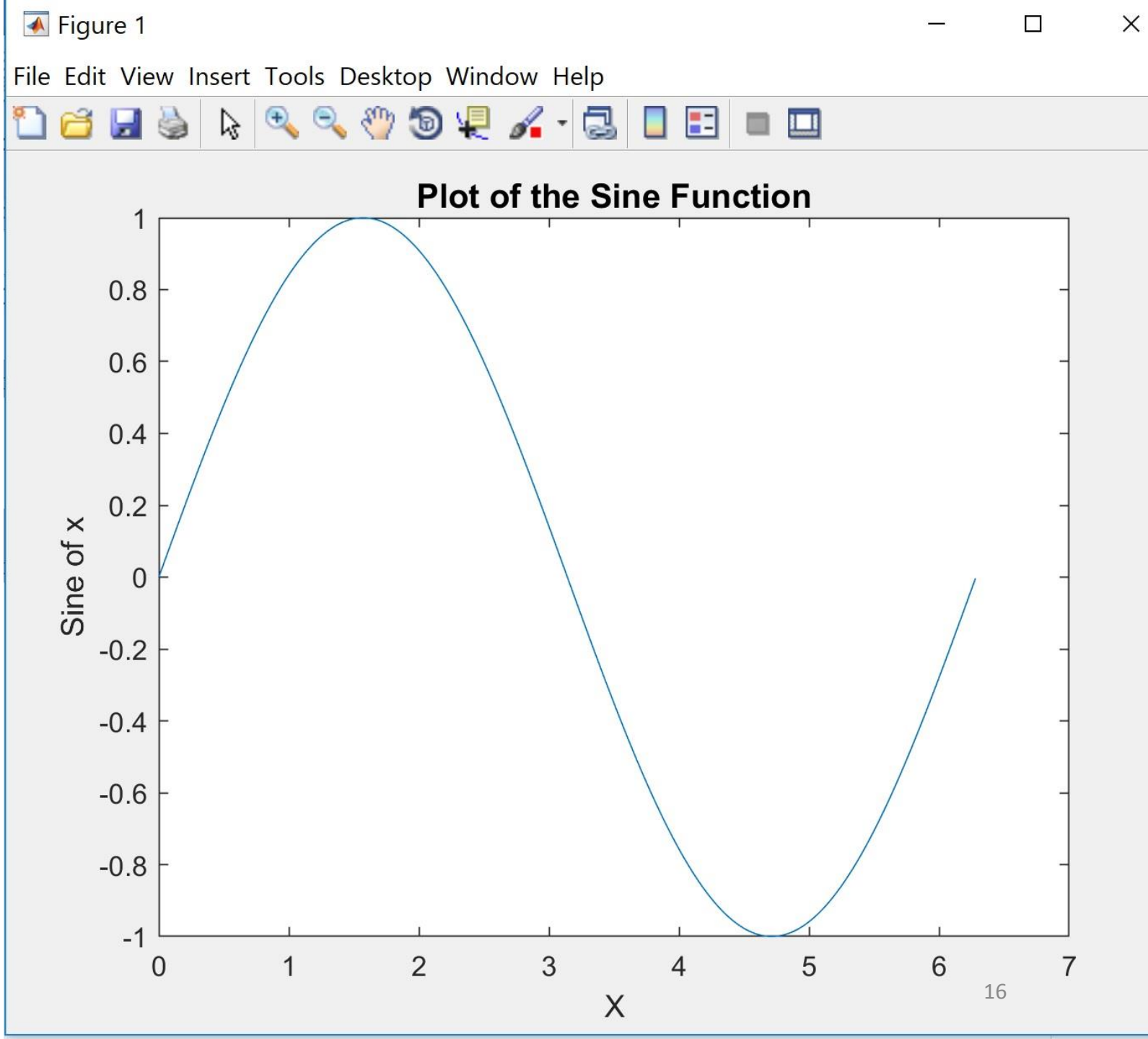
```
>> for i = 1:m
        for j = 1:n
            H(i,j) = 1/(i+j);
        end
    end
```

- While

```
>> while abs(M-.975)~=0
W2=logical(gam1<i);
M=mean(W2);
i=i+.05;
end
gam_en =i-.05
```

Basic Plotting

```
>> x = 0:.01:2*pi;
y = sin(x);
plot(x,y)
xlabel('X')
ylabel('Sine of x')
title('Plot of the Sine Function','FontSize',12)
```



Importing Information

- **Command Line Import:**

  The most elementary method for importing external information.

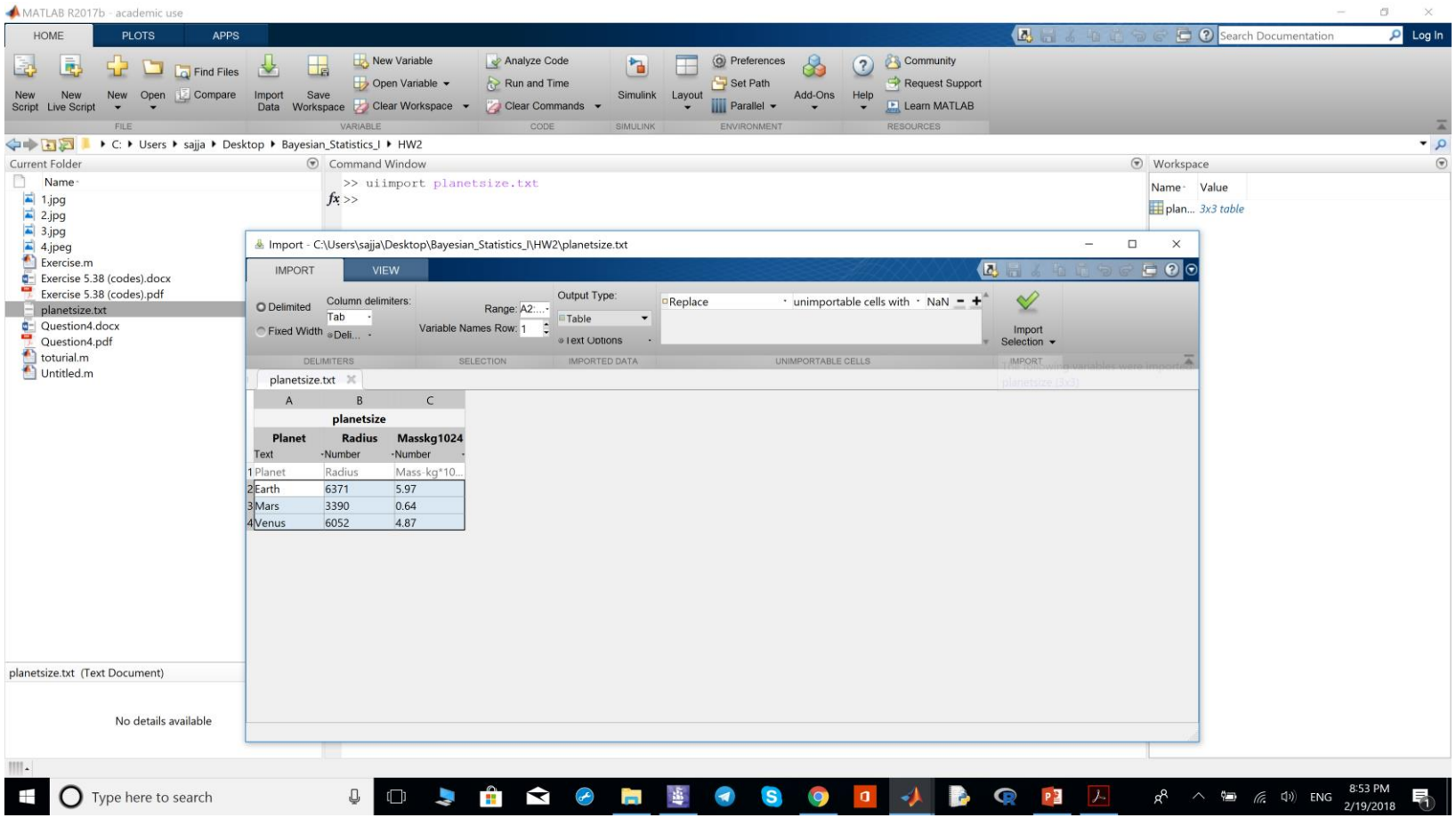
  ```
  >> earthradius = 6371;
  ```

- **The Import Wizard:**

  For convenient importation of data from external files. This tool can be activated by executing the `uiimport` command at a MATLAB command line prompt. This utility can be used for importing both text and numerical data contained within the same data file, but entries have to be in a matrix format with specified column separators.

  ```
  >> uiimport planetsize.txt
  ```

Importing Information



Importing Information

- **Import Functions:**
  - >> planets1 = **csvread**('planets1.txt’)

    Imports numeric data with comma-separated values (csv) and creates a matrix.
  - >> planets2 = **dlmread**('planets2.txt', ';')

    Similar to csvread but more flexible, allowing the delimiter to be specified by any character rather than restricting it to be a comma. It creates a matrix.
  - >> **load** planets3.txt

    Similar to csvread and dlmread but the separators can be blank spaces. It creates a matrix.
  - Other functions like **fscanf** (a lower level import function similar to the C language function), **textread** (similar to the primitive fscanf but will allow data variables to be defined as part of the import process.), **aviread**, **imread**, **xlsread** (for importing specialized types of binary files.).

Importing Information

- **M-file scripts**

  If data are already present within a text file **containing only legitimate MATLAB command syntax**, then that data can be imported by giving the file name an extension ".m" if not already present, and then typing that file name (without the ".m" extension) in the MATLAB command window

Exporting Information

- **diary**: The simplest way to export data to an external file. It creates an external file that can be edited with a text editor.

```
>> diary filename.txt
>> format short e  #set the display format to exponential form
>> planetinfo  #name of the variable
>> diary off
```

- **dlmwrite:** The dlmwrite function allows you to write external data files in which the delimiter can be specified.

```
>> dlmwrite('filename.txt',name of variable, ';')
```

- Other functions like **save** (This utility is a primitive function that will save an array in an external file with columns separated by blank space), **fprintf** (low level function that is equivalent to the C language function of the same name)

Basic Data Analysis Functions

Basic Data Analysis Function Summary	
Function	**Description**
cumprod	Cumulative product of elements.
cumsum	Cumulative sum of elements.
cumtrapz	Cumulative trapezoidal numerical integration.
diff	Difference function and approximate derivative.
max	Largest component.
mean	Average or mean value.
median	Median value.
min	Smallest component.
prod	Product of elements.
sort	Sort array elements in ascending or descending order.
sortrows	Sort rows in ascending order.
std	Standard deviation.
sum	Sum of elements.
trapz	Trapezoidal numerical integration.

Covariance and Correlation Coefficient Function Summary	
Function	**Description**
cov	Variance of vector - measure of spread or dispersion of sample variable. Covariance of matrix - measure of strength of linear relationships between variables.
corrcoef	Correlation coefficient - normalized measure of linear relationship strength between variables.

Preprocessing

- **Missing Values**

  You should remove NaN (The special value, NaN, stands for Not-a-Number in MATLAB. Any of these arithmetic operations will produce a NaN: zero/zero, zero*infinity, infinity/infinity, infinity-infinity)s from data before performing statistical computations

Code	Description
i = find(~isnan(x)); x = x(i)	Find indices of elements in vector that are not NaNs, then keep only the non-NaN elements.
x = x(find(~isnan(x)))	Remove NaNs from vector.
x = x(~isnan(x));	Remove NaNs from vector (faster).
x(isnan(x)) = [];	Remove NaNs from vector.
X(any(isnan(X)'),:) = [];	Remove any rows of matrix X containing NaNs.

Preprocessing

- **Removing Outliers (a value that is more than three scaled median absolute deviations (MAD) away from the median)**

  You can remove outliers or misplaced data points from a data set in the same manner as NaNs. For example:

```
>> y=[1 2 3 90]
    y =
    1    2    3   90
>> y(isoutlier(y))=[]
   y =
   1    2    3
```

- There are a lot of more things to learn about statistical computing using MATLAB like statistics toolbox and so forth!
  - More information on https://www.mathworks.com/help/stats/index.html.

- Next, we see results for one of the problems in Homework 1 using MATLAB. Then, we introduce MATLAB MCMC Toolbox.

Example:

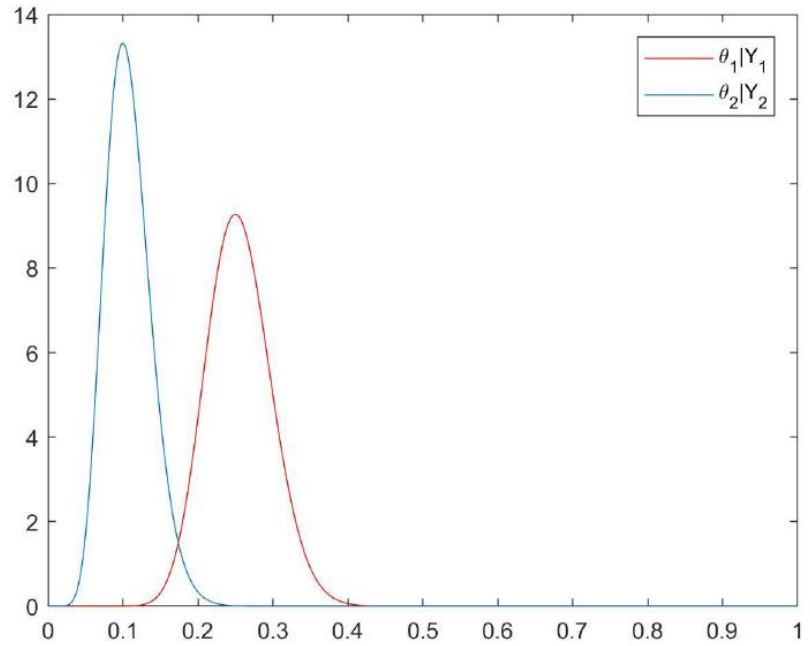
Exercise 3.7: Consider a two-binomial problem with independent Beta priors. Obtain the joint posterior for (θ1,θ2), to numerically **approximate the posterior probability that θ1 > θ2**, and to obtain a numerical approximation to the joint predictive density for a pair of future binomials, one from each population. We assume that the future values ($\tilde{y}1, \tilde{y}2$) are independent of the data ($y1,y2$) given the parameters (θ1,θ2). Consider numerically **approximating the predictive probability that $\tilde{y}1 > \tilde{y}2 + 10$**. **Find 95% probability interval for $\gamma = \theta1 - \theta2$**.

Example:

```
%Posterior probability approximation that Theta_1>Theta_2

n=10000; %number of observations
a1=1;a2=1;b1=1;b2=1; %parameters of prior distributions Theta_1 and Theta_2
y1=25; y2=10; %data
n1=100;n2=100;m1=100;m2=100; %number of trials for each sampling bionomial
distribution

pd1 = makedist('Beta',26,76); %plotting posterior pdfs
pd2 = makedist('Beta',11,91);
X=[0:.0005:1];
B1=pdf(pd1,X);
B2=pdf(pd2,X);
plot(X,B1,'r');
hold on;
plot(X,B2);
legend('\theta_1|Y_1','\theta_2|Y_2');

t1=betarnd(a1+y1,b1+n1-y1,[1,n]); %samples of posterior distribution for
Theta_1
t2=betarnd(a2+y2,b2+n2-y2,[1,n]); %samples of posterior distribution for
Theta_2
gam1 = t1-t2;
w1=logical(gam1>0); %new Bernoulli RV with probability of success
Pr(t1>t2|y1,y2)
ProbEst1 = mean(w1) %estimate of posterior probability that Theta_1 > Theta_2
SD_est1=std(w1)/sqrt(length(w1)) %estimate of standard error for estimated
Prob_est1
```

Example:

Results:




```
ProbEst1 =
    0.9973

SD_est1 =
   5.1894e-04
```

Example:

```
%obtaining 95% probability interval [gam_in,gam_en]
j=-1;
N=10;
while abs(N-.025)~=0
W1=logical(gam1<j);
N=mean(W1);
j=j+.000005;
end
gam_in =j-.05

i=gam_in;
M=10;
while abs(M-.975)~=0
W2=logical(gam1<i);
M=mean(W2);
i=i+.000005;
end
gam_en =i-.05
```

Results:

```
gam_in =
   -0.0054

gam_en =
    0.2008
```

Example:

```
%predictive probability approximation that y1_new>y_new + 10

N1=m1*ones(1,n);
N2=m2*ones(1,n);
y1New = binornd(N1,t1);
y2New = binornd(N2,t2); %predictive samples
gam2 = y1New-y2New-10;
w2=logical(gam2>0); %%new Bernoulli RV with probability of success
Pr(y1_new>y2_new+10)
ProbEst2 = mean(w2) %estimate of probability that y1_new > y2_new+10
SD_est2=std(w2)/sqrt(length(w2)) %estimate of standard error for estimated
Prob_est1
```

Results:

```
ProbEst2 =
    0.7154

SD_est2 =
    0.0045
```

MCMC Toolbox for MATLAB

- This toolbox provides tools to generate and analyze Metropolis-Hastings MCMC chain using multivariate Gaussian proposal distribution. No additional MATLAB toolboxes are used. However, a quite recent version of MATLAB is needed.
- The code can do the following
  - Produce MCMC chain for user written -2*log(likelihood) and -2*log(prior) functions. These will be equal to sum-of-squares functions when using Gaussian likelihood and prior.
  - Do plots and statistical analyses based on the chain, such as basic statistics, convergence diagnostics, chain timeseries plots, 2 dimensional clouds of points, kernel densities, and histograms.
  - Calculate densities, cumulative distributions, quantiles, and random variates for some useful common statistical distributions without using Mathworks own statistics toolbox.

Main Functions in the MCMC Toolbox

- **mcmcrun.m**

  MATLAB function for the MCMC run. The user provides her own MATLAB function to calculate the "sum-of-squares" function for the likelihood part, e.g. a function that calculates minus twice the log likelihood, -2log(p(θ;data)). Optionally a prior "sum-of-squares" function can also be given, returning -2log(p(θ)).

- **mcmcplot.m**

  This function plots some useful plots of the generated chain, such as chain time series, 2 dimensional marginal plots, kernel density estimates, and histograms.

- **mcmcpred.m**

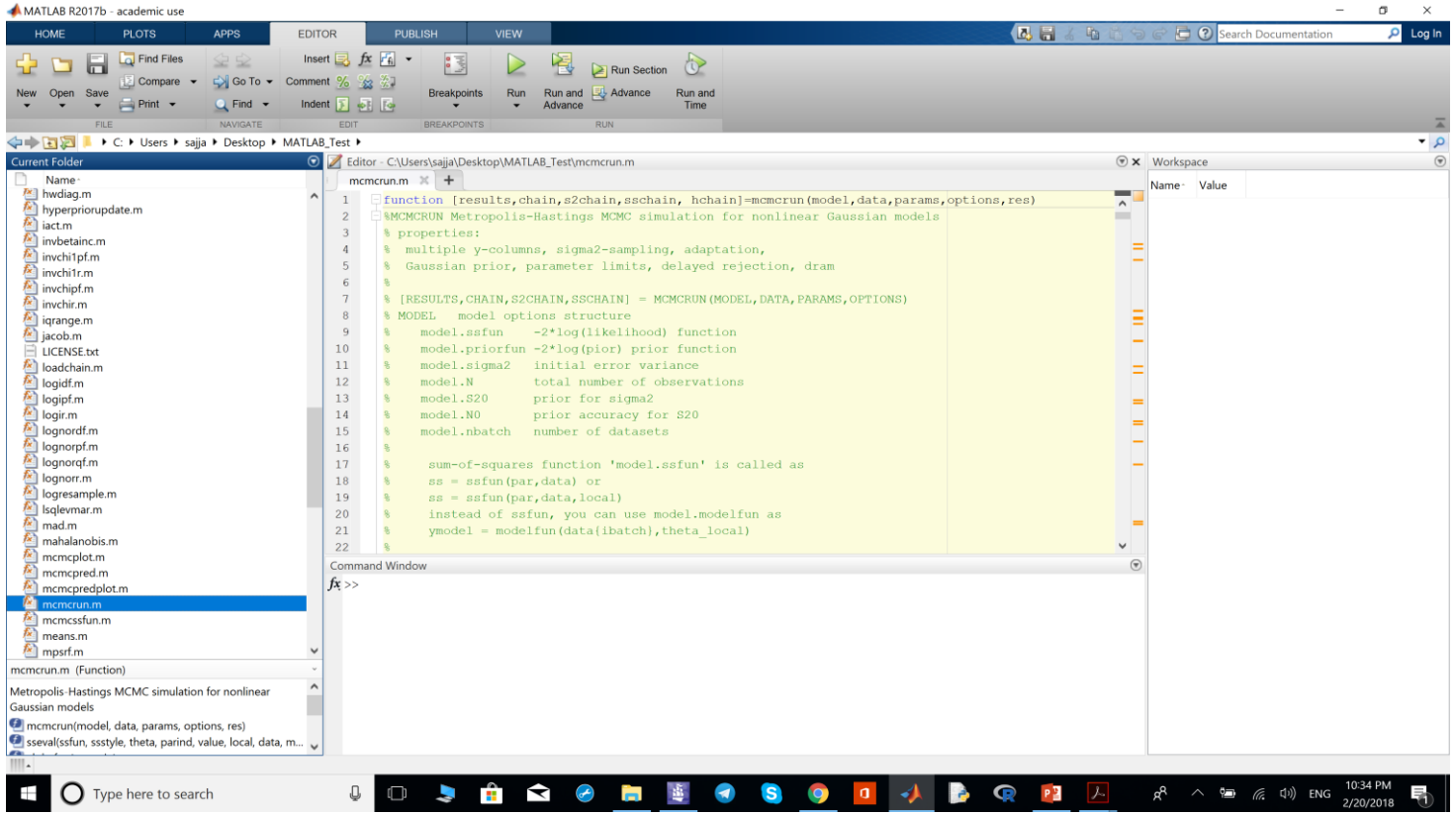
  For certain types of models is useful to plot predictive envelopes of model functions by sampling parameter values from the generated chain. This functions calls the model function repeatedly while sampling the unknowns from the chain. It calculates probability regions with respect to a "time" variable of the model.

The Toolbox Files

- You can download toolbox files and MATLAB scripts for some examples from http://helios.fmi.fi/~lainema/mcmc/ .

- Save toolbox files to a separate directory, and add the MATLAB path to it.

- For more details about each function and its inputs and outputs in this toolbox, you can click on its MATLAB file in MATLAB directory.



An Example Using MATLAB MCMC Toolbox

- **Dose response:** The classical binary beetle data is analyzed using complementary log-log regression and MCMC.
- Table shows numbers of beetles dead after five hours exposure to gaseous carbon disulphide at various concentrations (data from Bliss, 1935).

Beetle mortality data.

Dose, x_i ($\log_{10}\mathrm{CS}_2\mathrm{mgl}^{-1}$)	Number of beetles, n_i	Number killed, y_i
1.6907	59	6
1.7242	60	13
1.7552	62	18
1.7842	56	28
1.8113	63	52
1.8369	59	53
1.8610	62	61
1.8839	60	60

An Example Using MATLAB MCMC Toolbox

- -2log(likelihood) function:

```
function ss = beetless(theta,data)
% Beatle mortality example binomial -2*log(likelihood) function
% -2 log(likelihood) = -2*sum( y log(p) + (n-y) log(1-p) )

dose = data(:,1);
n    = data(:,2);
y    = data(:,3);

global BEETLE_LINK

switch BEETLE_LINK
 case 1
  % fitted probability from logistic model
  p  = 1./(1+exp(theta(1) + theta(2).*dose));
 case 2
  % loglog model
  p = 1-exp(-exp(theta(1)+theta(2).*dose));
 case 3
  % probit model
  p = nordf(theta(1) + theta(2).*dose);
end
```

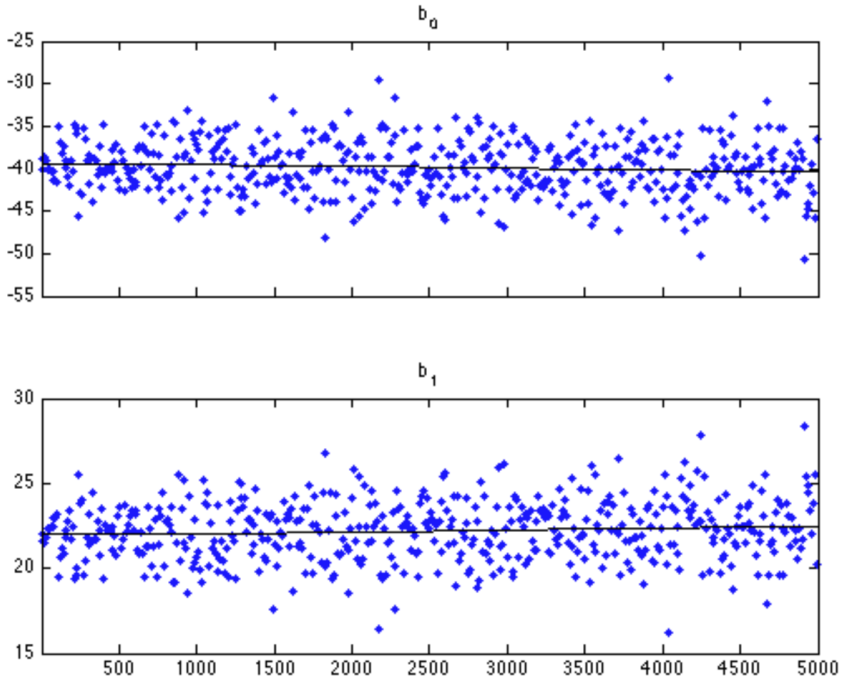
An Example Using MATLAB MCMC Toolbox

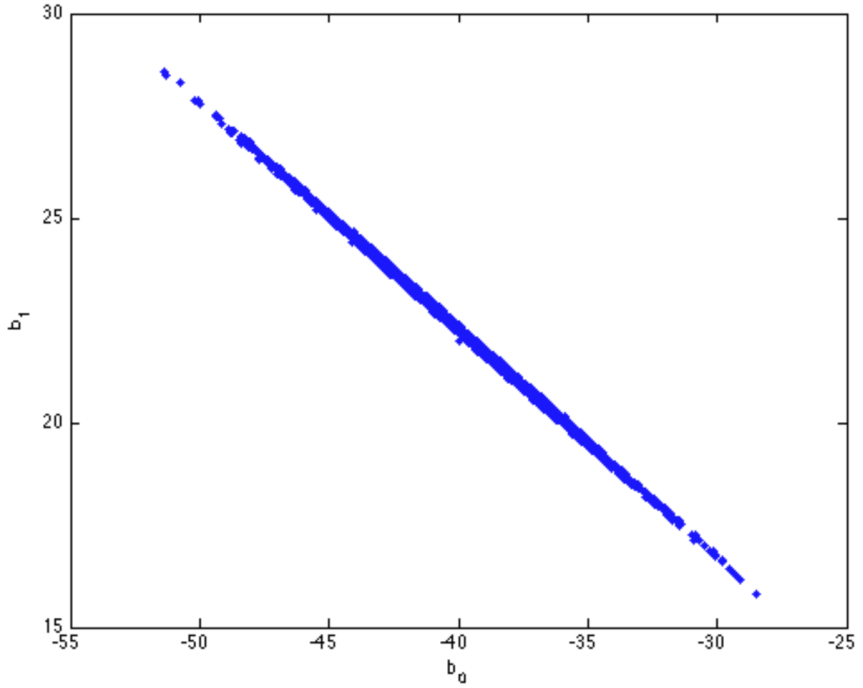
```
clear model data params options

data = [
% dose   n  y
  1.6907 59  6
  1.7242 60 13
  1.7552 62 18
  1.7842 56 28
  1.8113 63 52
  1.8369 59 53
  1.8610 62 61
  1.8839 60 60
    ];

global BEETLE_LINK
BEETLE_LINK = 2; % 1=logit, 2=loglog, 3=probit

% the "sum-of-squares" is now -2log(likelihood) of the binomial model
model.ssfun = @beetless;

% initial values and model function according to the link function
switch BEETLE_LINK
 case 1
  b = [ 60,-35]; % logit
  modelfun = @(d,th) 1./(1+exp(th(1)+th(2).*d));
  label = 'Beetle data with logit link';
 case 2
  b = [-40, 22]; % logog
  modelfun = @(d,th) 1-exp(-exp(th(1)+th(2).*d));
  label = 'Beetle data with loglog link';
 case 3
  b = [-35, 20]; % probit
  modelfun = @(d,th) nordf(th(1)+th(2).*d);
  label = 'Beetle data with probit link';
end
```

```
% model parameters
params = {
    {'b_0', b(1)}
    {'b_1', b(2)}
    };

options.nsimu = 5000;

[res,chain] = mcmcrun(model,data,params, options);

% plot the chain
figure
mcmcplot(chain,[],res)
figure
mcmcplot(chain,[],res,'pairs')

% sample the predicted mean respose
out = mcmcpred(res,chain,[],linspace(1.5,2)',modelfun,500);
figure
mcmcpredplot(out)
hold on % add data points
plot(data(:,1),data(:,3)./data(:,2),'ok')
hold off
title(label)
ylabel('proportion killed')
xlabel('log dose')
```

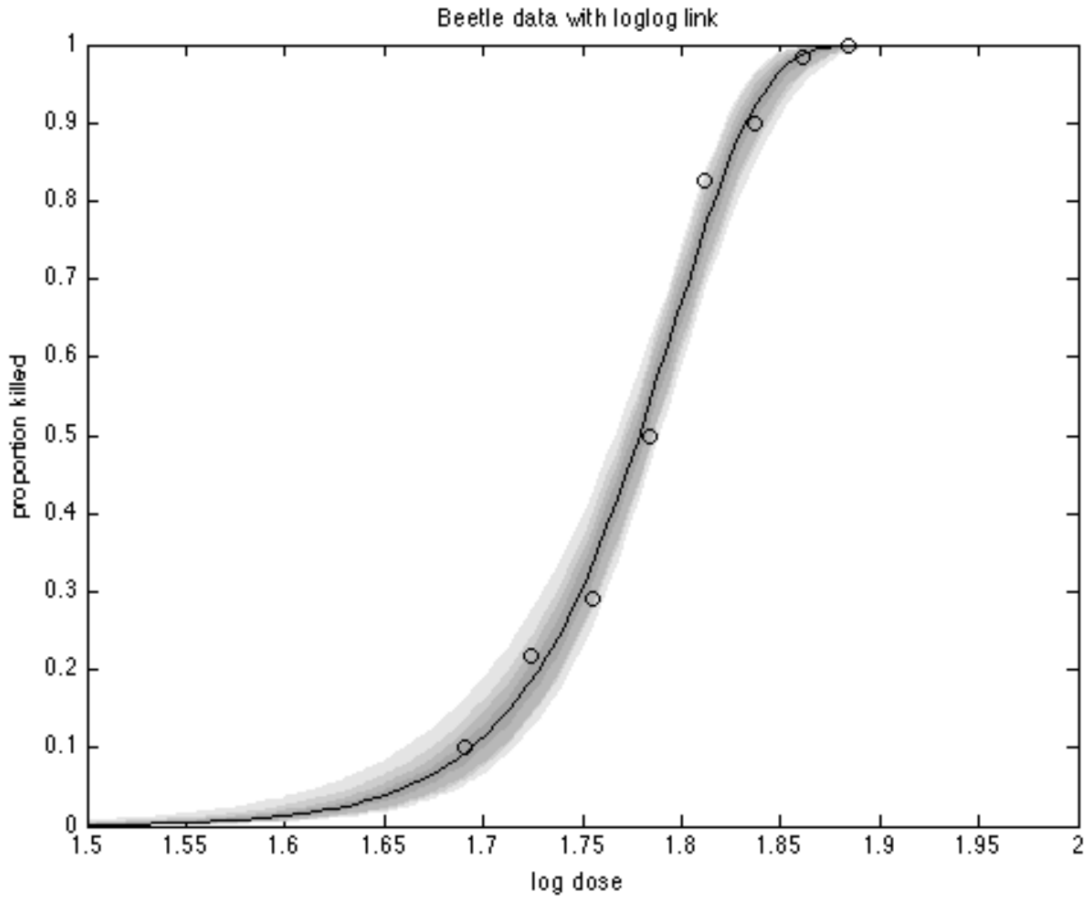
An Example Using MATLAB MCMC Toolbox

```
Sampling these parameters:
name   start [min,max] N(mu,s^2)
b_0: -40 [-Inf,Inf] N(0,Inf)
b_1: 22 [-Inf,Inf] N(0,Inf)
```





An Example Using MATLAB MCMC Toolbox



References

1. https://www.mathworks.com/products/MATLAB/MATLAB-vs-python.html
2. http://www.pyzo.org/python_vs_MATLAB.html
3. http://www.ams.sunysb.edu/~hahn
4. https://stat.utexas.edu/training/software-tutorials#MATLAB
5. http://helios.fmi.fi
6. Annette .J. Dobson, An Introduction to Generalized Linear Models, Second Edition: Chapman & Hall 2001.

Thank You!