Using Matlab

ECE 303: Introduction to Communication Principles
Fall 2010

I. What is Matlab?

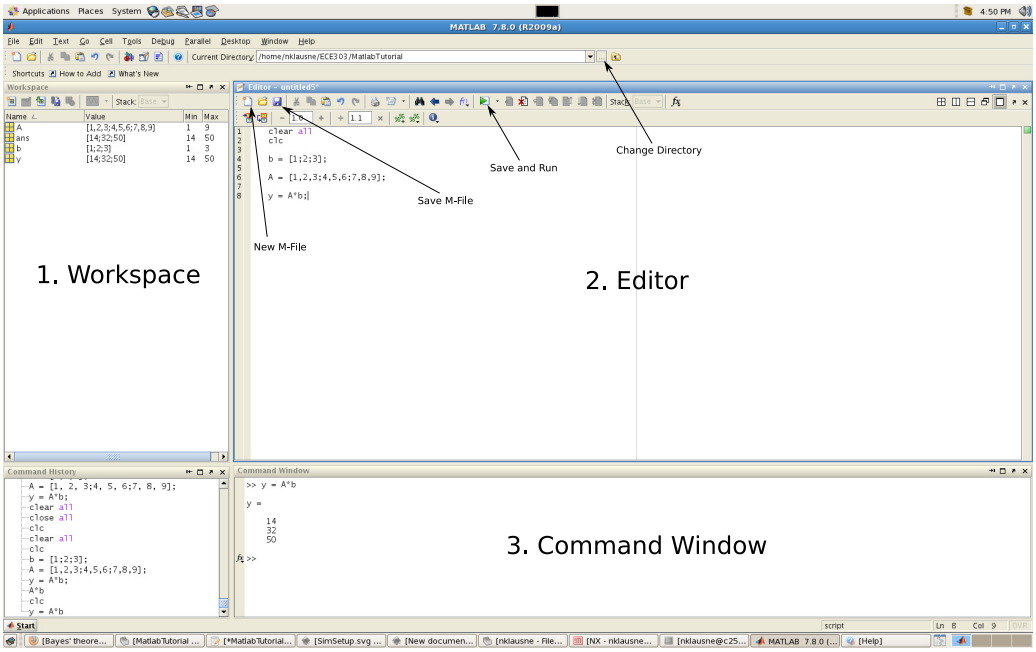
Matlab is an interactive, matrix-based programming tool with high-level computational power for technical computing. Matlab has hundreds of functions in the mail toolbox as well as in several other toolboxes that are application specific. The programming in Matlab is very simple because most of the Matlab programs can be written by calling the existing functions.

Typical uses include but are not limited to:

- Math and computation
- Algorithm development
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

II. Starting Matlab

On your PC click on the Matlab icon as a shortcut on the Desktop or in the Start menu. After loading, you should see something like this screen.



The main windows consist of:

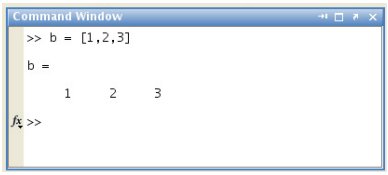
1) Workspace: Contains variables that have been created in the Command Window or Editor. You can view any variable in the Workspace by double-clicking it. You can clear the general variable 'XXX' by entering *clear XXX* in the Command Window. All variables can be cleared from the workspace by entering *clear all* in the Command Window.
2) Editor: A built-in text editor allowing you to enter the commands of your program. Commands are executed sequentially from top to bottom.
3) Command Window: Allows you to enter commands one at a time by entering the statement and hitting enter. The Command Window can be cleared by entering *clc* into the Command Window.

If you don't find any of these windows open when loading Matlab, they can be found in the 'Desktop' tab at the top. For example click *Desktop* → *Editor* to open the Editor window. Scripts generated in the Editor are known as M-files and have to be saved with a *.m* extension, e.g. *filename.m*. Matlab must be steered to the same directory as the script for the program to run. The directory (shown by *Current Directory:* at the top) can be changed by clicking the button shown by an arrow in the

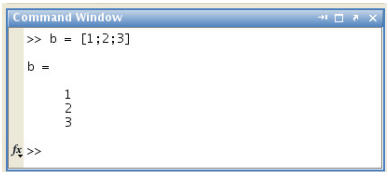
above figure and navigating to the directory where the M-file is saved. To run the script, click the button with a green arrow at the top of the Editor window.

III. Vectors and Matrices

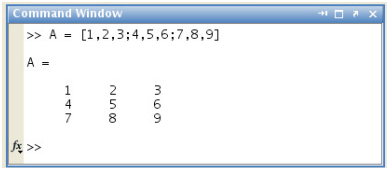
Creating vectors and matrices in Matlab is one of the easiest yet most fundamental parts of programming in this language. When creating a vector or matrix valued variable, spaces or commas (,) always delineate a new column, semicolons (;) always delineate rows, and the first and last entry in the matrix must be closed with square brackets ([and], respectively). To create a 1×3 row-vector enter the following command



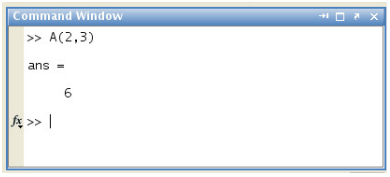
Likewise a 3×1 column vector



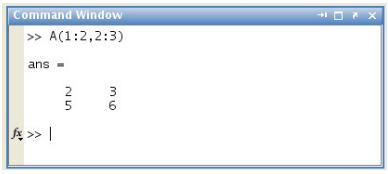
The following command creates a 3×3 square matrix



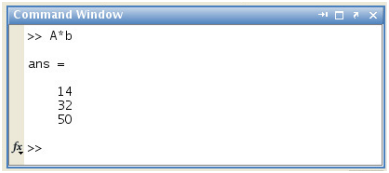
To reference an element in an array, type the name of the variable along with *(row,col)*. For example, the following command finds the element in the second row and third column



To reference more than one element, do the same but with *(row1:row2,col1:col2)*. For example, if we want rows 1 through 2 and columns 2 through 3 we type



Finally, to perform matrix multiplication we use the asterisk (*)



IV. Useful Commands and Functions

The following are a lists of basic commands, operators, symbols, and built-in Matlab functions that may be useful. If you need help finding or defining built-in Matlab functions, operators, symbols, etc., click *Help* → *Product Help* or navigate your web browser to www.mathworks.com.

TABLE I
General Purpose Commands

Command	Description
help	Online Documentation
lookfor	Keyword search through help entries
path	Control Matlab's search path
clear	Clear memory of variables and functions
disp	Desplay matrix or text
length	Length of vector
size	Size of matrix
save	Save workspace variables
load	Retrieve variables
chdir	Change current working directory
diary	Save test of Matlab session
dir	Directory listing
clc	Clear command window
format	Set output format
who(s)	List current variables
pwd	Show current working directory
more	Control paged output in window

TABLE II
Operators and Special Characters

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponentiation
...	Continuation
;	Delineates row in matrix, suppress statement output
'	Conjugate Transpose
.'	Non-conjugate Transpose
=	Assignment
%	Comment
= =	Equality
< (<=)	Less than (less than or equal to)
> (>=)	Greater than (greater than or equal to)
&	Logical AND
|	Logical OR
∼	Logical NOT

TABLE III
LANGUAGE CONSTRUCTS

Construct	Description
for	Repeat statements a number of times
while	Repeat statements until a condition is met
if	Conditionally execute statements
else, elseif	Used with if statements
end	Terminate scope of for, while, and if statements

TABLE IV
SPECIAL SYMBOLS

Symbol	Description
i,j	Imaginary unit in the complex number system
Inf	Infinity
NaN	Not a Number
pi	3.14159...

TABLE V
PLOTTING AND PRINTING

Command	Description
figure	Creates figure graphics object
plot	Plots a vector
subplot	Creates axes in tiled positions
hold on	Retains current plot so subsequent graphs can be added
xlabel	Adds text on the x-axis of the figure
ylabel	Adds text on the y-axis of the figure
title	Add graph title
legend	Add legend entries to the plot
axis	Manually control x and y axes
grid	Add grid on the figure
orient	Figure orientation
print	Print figure or model, save as image or M-file

V. EXAMPLE 1: CLASSIFICATION

Problem:

The FBI is interested in formulating a battery of questions to aid in identifying criminal minds from a randomly selected set of individuals in the population. The questions are designed so that for each individual the answers are independent in the sense that the answers to any subset of these questions is not affected by and does not affect the answers to any other subset of the questions.

A sample of 125 subjects with *known* criminal (class 1) or no criminal (class 2) backgrounds was chosen to design the classification scheme. Each subject was asked a battery of 10 independent questions. The subjects answer independently. Data on the results are summarized in the following table.

TABLE VI
SUBJECT QUESTIONNAIRE DATA

	Class 1			***Class 2***		
Question	Yes	No	Unclear	Yes	No	Unclear
1	42	22	5	20	31	5
2	34	27	8	16	37	3
3	15	45	9	33	19	4
4	19	44	6	31	18	7
5	22	43	4	23	28	5
6	41	13	15	14	37	5
7	9	52	8	31	17	8
8	40	26	3	13	38	5
9	48	12	9	27	24	5
10	20	37	12	35	16	5

Assuming that this data is representative of the entire population consisting of these groups, we would like to use the classification scheme derived based upon this data to identify the potential for criminal activity of a randomly selected individual from the population.

Three people were then interviewed and the result of each interview is a "profile" of answers to the questions. The following profiles were taken

TABLE VII
INDIVIDUAL PROFILES

Question	Subject 1	Subject 2	Subject 3
1	Y	N	Y
2	N	N	Y
3	Y	U	N
4	N	N	Y
5	Y	Y	U
6	U	Y	U
7	N	U	N
8	U	N	N
9	Y	N	Y
10	U	Y	Y

Classify each individual into one of the two groups depending on the likelihood for criminal activity.

Solution:

Let $\{A_i\}_{i=1}^{10}$ denote the set of events generated from the 10 answers given by any one subject. For the first subject for example, $A_1 = \{$"The subject has answered YES to Question 1"$\}$. What we are ultimately interested in is the *profile* of answers associated with any one individual which we will denote as A. Again, for the first person we have $A = \{$"The subject has answered YES to Question 1 *and* NO to Question 2 *and* $\ldots$ *and* UNCLEAR to Question 10"$\}$. Mathematically using set theory we write

$$A = \bigcap_{i=1}^{10} A_i$$

We will also let C_j denote the event that any subject belongs to class j for $j = 1, 2$. An intuitive method for deciding class membership is to associate the individual with class 1 when the following inequality is met

$$P\left(C_1|A\right) > P\left(C_2|A\right),$$

otherwise associate that person with class 2. Baye's theorem tells us that

$$P\left(C_j|A\right) = \frac{P\left(A|C_j\right)P\left(C_j\right)}{P\left(A\right)}$$

resulting in an equivalent test

$$\frac{P\left(A|C_1\right)}{P\left(A|C_2\right)} > \frac{P\left(C_2\right)}{P\left(C_1\right)}$$

The term on the left is typically referred to as a *likelihood ratio* and the term on the right is a threshold dependent on the *a priori* probabilities and independent of any answer one might give. From our assumptions about the independence of the questions in this survey, it follows that

$$P\left(A|C_j\right) = P\left(\bigcap_{i=1}^{10} A_i|C_j\right) = \prod_{i=1}^{10} P\left(A_i|C_j\right)$$

resulting in the final likelihood ratio test

$$\frac{\prod_{i=1}^{10} P\left(A_i|C_1\right)}{\prod_{i=1}^{10} P\left(A_i|C_2\right)} > \frac{P\left(C_2\right)}{P\left(C_1\right)}$$

To implement a basic program in Matlab, we would like to build a program that inputs a profile and outputs class membership and/or a likelihood value. We can do this in Matlab by building an M-file function. Building a function is like building any other M-file script but a function takes variables as inputs and returns new variables as outputs. To begin a function type *function [op1, op2, ...] = FUNCTIONNAME(ip1, ip2, ...)* at the beginning of the script and save it as *FUNCTIONNAME.m*

in the directory you want. Just like any other M-file, you cannot save the function in one directory and hope to use it in another. The following code implements *BayesianClassifier.m*, a function that takes a 10×1 vector of characters (the profile) and outputs class membership, the likelihood ratio, and a 10×1 vector of likelihood values we'll look at later.

```
function [class lr likelihood] = BayesianClassifier(answers)

%% Probability Model Based on Previously Surveyed Individuals

NumTotal = 125; % Total Number of Individuals
NumC1 = 69;     % Number of Individuals in Class 1
NumC2 = 56;     % Number of Individuals in Class 2

P_C1 = NumC1/NumTotal; % A-Priori Class 1 Probability
P_C2 = NumC2/NumTotal; % A-Priori Class 2 Probability

% This matrix represents the number of answers given Class 1. First column
% is Y, second N, third U.  Rows correspond to different questions.
Num_Ans_C1 = [42,22,5; 34,27,8; 15,45,9; 19,44,6; 22,43,4; 41,13,15; ...
              9,52,8; 40,26,3; 48,12,9; 20,37,12];

% Same as before but for class 2.
Num_Ans_C2 = [20,31,5; 16,37,3; 33,19,4; 31,18,7; 23,28,5; 14,37,5; ...
              31,17,8; 13,38,5; 27,24,5; 35,16,5];

%% Compute the Likelihood
% Initialize the likelihood probabilities
P_Ans_C1 = 1;
P_Ans_C2 = 1;

% For loop used to accumulate the likelihood based on the elements given in
% the input vector 'answer'.
for i = 1:10
    if answers(i) == 'Y'
        P_Ans_C1 = P_Ans_C1*(Num_Ans_C1(i,1)/NumC1);
        P_Ans_C2 = P_Ans_C2*(Num_Ans_C2(i,1)/NumC2);
    elseif answers(i) == 'N'
        P_Ans_C1 = P_Ans_C1*(Num_Ans_C1(i,2)/NumC1);
        P_Ans_C2 = P_Ans_C2*(Num_Ans_C2(i,2)/NumC2);
    else
        P_Ans_C1 = P_Ans_C1*(Num_Ans_C1(i,3)/NumC1);
        P_Ans_C2 = P_Ans_C2*(Num_Ans_C2(i,3)/NumC2);
    end
    likelihood(i) = (P_Ans_C1)/(P_Ans_C2);
end

% Find the likelihood ratio
lr = (P_Ans_C1)/(P_Ans_C2)

%% Determine Class Membership
if lr > P_C2/P_C1
    class = 1
else
    class = 2
end
```

As can be seen, the function begins by again defining the *a priori* probabilities and creating the matrices containing the number of Yes/No/Unclear answers. Here, these variables are "hard-wired" into the function which can easily be changed by defining them as inputs. The function then uses a *for* loop with an *if, elseif* statement to decide which of the elements in the two data arrays to reference and accumulates the likelihood values through multiplication. The sequence of statements progresses as follows:

1) The likelihood values for both classes are initialized at a value of 1
2) The for loop starts with a value of $i = 1$ (Question 1)
3) The if, elseif statement compares the first element of *answers* to 'Y', 'N', and 'U' to find what answer the subject has given for the first question
4) If the subject has answered yes to the first question, the function takes the first row and first column of the data array, divides by the number of people in that class, and accumulates the likelihood through multiplication (the second and third columns are used if the subject has answered no or unclear, respectively)
5) The for loop iterates to $i = 2$ (Question 2)

6) The if, elseif statement compares the second element of *answers* to 'Y', 'N', and 'U' to find what answer the subject has given for the second question
7) If the subject has answered yes to the second question, the function takes the second row and first column of the data array, divides by the number of people in that class, and accumulates the likelihood through multiplication (the second and third columns are used if the subject has answered no or unclear, respectively)
8) The for loop iterates to $i = 3$ (Question 3)
9) ...

After all 10 iterations of the for loop, the likelihood ratio is computed and compared to the threshold to decide class membership. In a separate script file, the following code then defines the answer vectors for all three subjects and calls the *BayesianClassifier* function.

```
%% First Subject

% Create a 10x1 vector consisting of the answers
answers1 = ['Y';'N';'Y';'N';'Y';'U';'N';'U';'Y';'U'];

% Call BayesianClassifier
disp('First Subject')
[class1 lr1 like1] = BayesianClassifier(answers1);

%% Second Subject

answers2 = ['N';'N';'U';'N';'Y';'Y';'U';'N';'N';'Y'];

disp('Second Subject')
[class2 lr2 like2] = BayesianClassifier(answers2);

%% Third Subject

answers3 = ['Y';'Y';'N';'Y';'U';'U';'N';'N';'Y';'Y'];

disp('Third Subject')
[class3 lr3 like3] = BayesianClassifier(answers3);
```

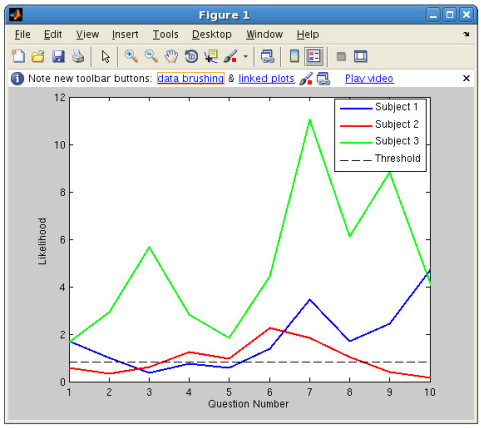
```
Command Window
First Subject

lr =

    4.7439

class =

     1

Second Subject

lr =

    0.1934

class =

     2

Third Subject

lr =

    4.0984

class =

     1

fx >>
```

The vector *likelihood* defined earlier in *BayesianClassifier.m* is a 10×1 vector showing the accumulated likelihood ratio as one starts with Question 1 and adds questions. Mathematically, the k^{th} element of this vector is described by equation

$$\frac{\prod_{i=1}^{k} P\left(A_i|C_1\right)}{\prod_{i=1}^{k} P\left(A_i|C_2\right)}$$

The following code is then used to graph this vector for all three individuals on the same plot using different colors (the *'LineWidth'* option is used to just make the lines a little wider). Also graphed on the same plot with a dotted line is the threshold $(P(C_2)/P(C_1) = 56/69)$. Note that *1:10* creates a row vector of integers from 1 to 10 and *ones(1,10)* creates a 1×10 row vector with each element equal to one.

```
%% Display likelihood vectors all on the same plot with different colors
figure;
plot(like1,'LineWidth',2);
hold on;
plot(like2,'r','LineWidth',2);
hold on;
plot(like3,'g','LineWidth',2);
hold on;
plot(1:10,(56/69)*ones(1,10),'k--')
xlabel('Question Number')
ylabel('Likelihood')
title('Likelihood vs. Question')
legend('Subject 1','Subject 2','Subject 3','Threshold')
```



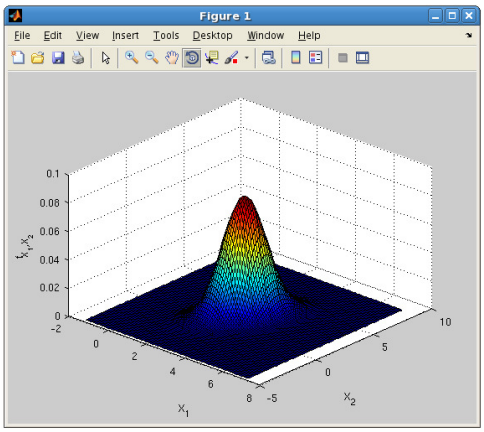
VI. Example 2: Bivariate Gaussian (Peebles, Sections 5.3 and 5.6)

Two random variables $X_1 \sim N(\mu_1, \sigma_1^2)$ and $X_2 \sim N(\mu_2, \sigma_2^2)$ are said to be bivariate normal with correlation coefficient ρ (a number between 0 and 1 with larger values implying greater dependence among the two random variables) if they share the joint density function

$$f_{X_1,X_2}(x_1,x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}}\exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{(x_1-\mu_1)^2}{\sigma_1^2}+\frac{(x_2-\mu_2)^2}{\sigma_2^2}-\frac{2\rho(x_1-\mu_1)(x_2-\mu_2)}{\sigma_1\sigma_2}\right]\right)$$

The following script generates a 3 dimensional view of the joint density using the *surface* plotting routine

```
% Define the Parameters of the Bivariate Distribution
mu1 = 3;
mu2 = 2;
sigma1 = sqrt(2);
sigma2 = sqrt(5);
rho = 0.8;

% Generate equally-spaced points between [mu-3sigma,mu+3sigma]
x1 = linspace(mu1-3*sigma1,mu1+3*sigma1,100);
x2 = linspace(mu2-3*sigma2,mu2+3*sigma2,100);

% Evaluate Density on the Grid
for i = 1:length(x1)
    for j = 1:length(x2)
        disp([i,j])
        f(i,j) = (1/(2*pi*sigma1*sigma2*sqrt(1-rho^2)))*exp( ...
            -(1/(2*(1-rho^2)))*((x1(i)-mu1)^2/sigma1 ...
            +(x2(j)-mu2)^2/sigma2-2*rho*(x1(i)-mu1)*(x2(j)-mu2)/ ...
            (sigma1*sigma2)));
    end
end

% 3D Graph of the Density
figure;surface(x1,x2,f)
xlabel('X_{1}')
ylabel('X_{2}')
zlabel('f_{X_{1},X_{2}}')
grid on
```



A generalization of the bivariate Gaussian distribution to more than two random variables is the multivariate Gaussian distribution which is parametrized by a mean vector, $\boldsymbol{\mu}$, and covariance matrix, R. For the bivariate case these can be written as

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

$$R = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

The following code generates 1000 realizations of two independent standard Normal random variables and defines the 2×1 mean vector and 2×2 covariance matrix

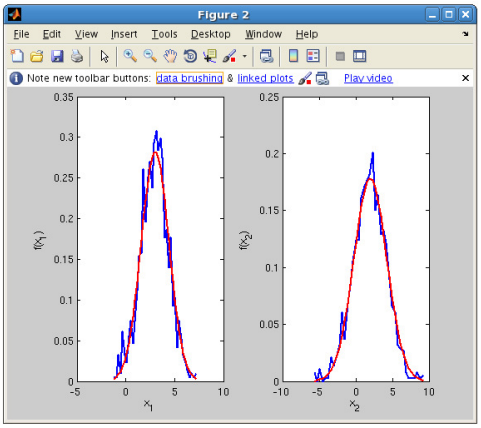
```
% Generate 1000 iid Normal Samples
z = randn(2,1000);

% Define Mean Vector and Covariance Matrix
mu = [mu1;mu2];
R = [sigma1^2,rho*sigma1*sigma2;rho*sigma1*sigma2,sigma2^2];
```

The script then produces the 2×2 matrix A known as a Cholesky decomposition of R which simply finds the lower triangular matrix A such that $R = AA^T$ (matrix A is like the square-root of R). Realizations of the standard Normal random variables are then transformed to two random variables from the bivariate Gaussian distribution described before through an affine transformation. The *hist* command is then used to partition the empirical sample space of these two random variables into 40 uniformly spaced bins whose locations are given by the variables *xout1* and *xout2*. The variables *n1* and *n2* give the number of realizations that fall into these bins which are normalized so that we may compare them to a pdf.

```
% Cholesky Decomposition
A = chol(R,'lower');

% Shift iid Samples in Mean and Correlate
for i = 1:size(z,2)
    y(:,i) = mu+A*z(:,i);
end

% Build Histogram of 1st RV with 40 bins
[n1,xout1] = hist(y(1,:),40);

% Normalize Histogram
n1 = n1/(sum(n1)*(xout1(2)-xout1(1)));

% Same for 2nd RV
[n2,xout2] = hist(y(2,:),40);
n2 = n2/(sum(n2)*(xout2(2)-xout2(1)));
```

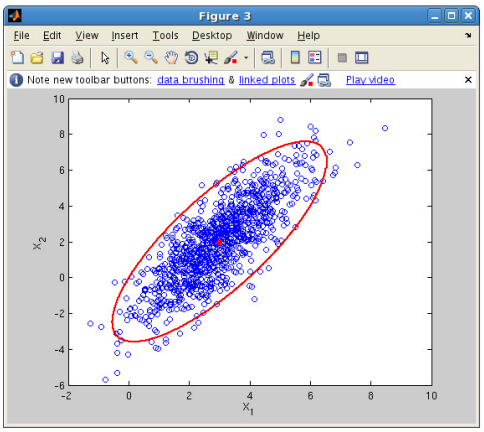
The following code then uses the *normpdf* command to generate the two marginal Gaussian densities which are then plotted along with the histograms using the *subplot* routine.

```
% Generate Marginal Densities
f1 = normpdf(xout1,mu1,sigma1);
f2 = normpdf(xout2,mu2,sigma2);

% Plot Histograms
figure;
subplot(1,2,1)
plot(xout1,n1,'LineWidth',2);
hold on;
plot(xout1,f1,'r','LineWidth',2)
xlabel('x_{1}')
ylabel('f(x_{1})')

subplot(1,2,2)
plot(xout2,n2,'LineWidth',2);
hold on;
plot(xout2,f2,'r','LineWidth',2)
xlabel('x_{2}')
ylabel('f(x_{2})')
```



Finally, we define a vector *theta* consisting of values ranging from 0 to 2π in increments of 0.001. Similar to taking standard Normal random variables and transforming them via an affine transformation, a circle with radius one is constructed (*cos(theta(i))* and *sin(theta(i))* describe the Cartesian coordinates for a circle in $\mathbb{R}^2$) which is then scaled, rotated, and translated according to the exact same affine transformation as before. Doing so results in the construction of what is known as a covariance ellipse and is essentially a horizontal "slice" of the 3 dimensional surface plot generated earlier. Note that 2.5 is multiplied to simply make the ellipse larger for illustrative purposes. A scatter plot of the realizations of these two random variables is then graphed along with the ellipse and mean (shown by a red curve and red dot, respectively).

```
% Generate Covariance Ellipse
theta = 2*pi*[0:0.001:1];
for i = 1:length(theta)
    x(:,i) = mu+2.5*A*[cos(theta(i));sin(theta(i))];
end

% Generate Scatter Plot
figure;
plot(y(1,:),y(2,:),'bo');
hold on;
plot(x(1,:),x(2,:),'r','LineWidth',2);
hold on;
plot(mu1,mu2,'ro','MarkerFaceColor','r')
xlabel('X_{1}')
ylabel('X_{2}')
title('Scatter Plot')
```



As can be observed, the realizations are centered about the mean and with high probability lie inside the covariance ellipse.