Graphics in Matlab

Bruce A. Desmarais

Odum Institute for Research in Social Science

February 21, 2009

Welcome!

Who am I?

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra, simulation and statistical computing in Matlab and other software/languages

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra, simulation and statistical computing in Matlab and other software/languages
- No formal training in the instruction of Matlab (comments/suggestions will be much appreciated)

What will we cover?

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra, simulation and statistical computing in Matlab and other software/languages
- No formal training in the instruction of Matlab (comments/suggestions will be much appreciated)

What will we cover?

- Mechanics of Plotting in Matlab

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra, simulation and statistical computing in Matlab and other software/languages
- No formal training in the instruction of Matlab (comments/suggestions will be much appreciated)

What will we cover?

- Mechanics of Plotting in Matlab
- Many two-dimensional plot types

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra, simulation and statistical computing in Matlab and other software/languages
- No formal training in the instruction of Matlab (comments/suggestions will be much appreciated)

What will we cover?

- Mechanics of Plotting in Matlab
- Many two-dimensional plot types
- Some 3-D plots

Welcome!

Who am I?

- Bruce Desmarais; Red Sox Fan and PhD student in the UNC Poli Sci Dept.
- Extensive experience with matrix algebra, simulation and statistical computing in Matlab and other software/languages
- No formal training in the instruction of Matlab (comments/suggestions will be much appreciated)

What will we cover?

- Mechanics of Plotting in Matlab
- Many two-dimensional plot types
- Some 3-D plots
- Saving Plots

General Approach to Plotting in Matlab

General Approach to Plotting in Matlab

1. Do it with code, in a proper Matlab editor!!

General Approach to Plotting in Matlab

1. Do it with code, in a proper Matlab editor!!
2. Use pull-down menus to make post-hoc edits, then look up functions to include these in code

General Approach to Plotting in Matlab

1. Do it with code, in a proper Matlab editor!!
2. Use pull-down menus to make post-hoc edits, then look up functions to include these in code
3. Save code for replication

General Approach to Plotting in Matlab

1. Do it with code, in a proper Matlab editor!!
2. Use pull-down menus to make post-hoc edits, then look up functions to include these in code
3. Save code for replication
4. Save plot for inclusion in work

General Approach to Plotting in Matlab

1. Do it with code, in a proper Matlab editor!!
2. Use pull-down menus to make post-hoc edits, then look up functions to include these in code
3. Save code for replication
4. Save plot for inclusion in work
5. Use 'help *function*' to figure out the various arguments in a function

A Basic Scatter Plot

A Basic Scatter Plot

```
%Randomly Generate X
X = randn(100,1);
%Generate Y conditional on X
Y = X + rand(100,1);
```

A Basic Scatter Plot

```
%Randomly Generate X
X = randn(100,1);
%Generate Y conditional on X
Y = X + rand(100,1);

%Plot Y against X
plot(X,Y,'.')
```

A Basic Scatter Plot

```
%Randomly Generate X
X = randn(100,1);
%Generate Y conditional on X
Y = X + rand(100,1);

%Plot Y against X
plot(X,Y,'.')
```

This produces a simple, unlabeled scatter plot. There are many point-type arguments including:

A Basic Scatter Plot

```
%Randomly Generate X
X = randn(100,1);
%Generate Y conditional on X
Y = X + rand(100,1);

%Plot Y against X
plot(X,Y,'.')
```

This produces a simple, unlabeled scatter plot. There are many point-type arguments including:

b	blue	.	point	-	solid
g	green	o	circle	:	dotted
r	red	x	x-mark	-.	dashdot
c	cyan	+	plus	--	dashed

Titles, Axis Labels and Legends

All of these are added to plots through separate Matlab commands

- Title Usage: *title('text','FontSize',...,'FontName',...,'Color',...)*;

Titles, Axis Labels and Legends

All of these are added to plots through separate Matlab commands

- Title Usage: *title('text','FontSize',...,'FontName',...,'Color',...)*;
- Try This: *title('Y by X','FontSize',12,'FontName','Papyrus','Color','red')*;

Titles, Axis Labels and Legends

All of these are added to plots through separate Matlab commands

- Title Usage: *title('text','FontSize',...,'FontName',...,'Color',...)*;
- Try This: *title('Y by X','FontSize',12,'FontName','Papyrus','Color','red')*;
- X and Y axis usage: *ylabel('Text','FontSize',...,'FontName',...,'Color',...)*

Titles, Axis Labels and Legends

All of these are added to plots through separate Matlab commands

- Title Usage: *title('text','FontSize',...,'FontName',...,'Color',...)*;
- Try This: *title('Y by X','FontSize',12,'FontName','Papyrus','Color','red')*;
- X and Y axis usage: *ylabel('Text','FontSize',...,'FontName',...,'Color',...)*
- Try This: *ylabel('Y','FontSize',12,'FontName','Calibri')*

Titles, Axis Labels and Legends

All of these are added to plots through separate Matlab commands

- Title Usage: *title('text','FontSize',...,'FontName',...,'Color',...)*;
- Try This: *title('Y by X','FontSize',12,'FontName','Papyrus','Color','red')*;
- X and Y axis usage: *ylabel('Text','FontSize',...,'FontName',...,'Color',...)*
- Try This: *ylabel('Y','FontSize',12,'FontName','Calibri')*
- Legend Usage: *legend('string1','string2'...)* Caveat

Titles, Axis Labels and Legends

All of these are added to plots through separate Matlab commands

- Title Usage: *title('text','FontSize',...,'FontName',...,'Color',...)*;
- Try This: *title('Y by X','FontSize',12,'FontName','Papyrus','Color','red')*;
- X and Y axis usage: *ylabel('Text','FontSize',...,'FontName',...,'Color',...)*
- Try This: *ylabel('Y','FontSize',12,'FontName','Calibri')*
- Legend Usage: *legend('string1','string2'...)* Caveat
- Try This: *legend('xy')*

Titles, Axis Labels and Legends

All of these are added to plots through separate Matlab commands

- Title Usage: *title('text','FontSize',...,'FontName',...,'Color',...)*;
- Try This: *title('Y by X','FontSize',12,'FontName','Papyrus','Color','red')*;
- X and Y axis usage: *ylabel('Text','FontSize',...,'FontName',...,'Color',...)*
- Try This: *ylabel('Y','FontSize',12,'FontName','Calibri')*
- Legend Usage: *legend('string1','string2'...)* Caveat
- Try This: *legend('xy')*

Titles, Axis Labels and Legends

All of these are added to plots through separate Matlab commands

- Title Usage: *title('text','FontSize',...,'FontName',...,'Color',...)*;
- Try This: *title('Y by X','FontSize',12,'FontName','Papyrus','Color','red')*;
- X and Y axis usage: *ylabel('Text','FontSize',...,'FontName',...,'Color',...)*
- Try This: *ylabel('Y','FontSize',12,'FontName','Calibri')*
- Legend Usage: *legend('string1','string2'...)* Caveat
- Try This: *legend('xy')*

All of these commands can be used independently to construct a plot, but much more interesting and varied plots can be constructed by forcing inheritance relations.

The Same Plot Again I

```
% Create figure
figure1 = figure;

% Create axes
axes1 = axes('Parent',figure1,'FontName','Serif');
box('on');
hold('all');

% Create plot
plot(X,Y,'Parent',axes1,'Marker','.',
'LineStyle','none',...'DisplayName','XY');

% Create xlabel
xlabel('X','FontName','Serif');
```

The Same Plot Again II

```
% Create ylabel
ylabel('Y','FontSize',12,'FontName','Calibri');

% Create title
title('Y by X','FontSize',12,'FontName',
'Papyrus','Color','red');

% Create legend
legend(axes1,'show');
```

Histogram I

```
%Create Figure
figure1 = figure('Color','white');

% Create Histogram
hist(X);

% Identify color control
h = findobj(gca,'Type','patch');
set(h,'FaceColor','red')
```

The *set* command assigns generic parameters to different objects.

Histogram I

```
%Create Figure
figure1 = figure('Color','white');

% Create Histogram
hist(X);

% Identify color control
h = findobj(gca,'Type','patch');
set(h,'FaceColor','red')
```

The *set* command assigns generic parameters to different objects.
The *findobj* function finds objects with specified parameters.

Histogram II

```
% Create xlabel
xlabel('X');

% Create ylabel
ylabel('Frequency of X (Sample of 100)');

% Create title
title('Histogram of X','FontWeight','bold','FontSize',
     14,...'FontName','Aharoni');
```

Kernel Density I

Matlab Has no Kernel Density plot function...

Kernel Density I

Matlab Has no Kernel Density plot function...

```
% Create density of X
[f,xf] = ksdensity(X);

% Create figure
figure1 = figure('Color',[0.9059 0.9059 0.9059]);

% Create axes
axes('Parent',figure1);
box('on');
hold('all');
```

Kernel Density II

```
% Create plot
plot(xf,f,'LineWidth',2);

% Create title
title('Kernel Density Plot of X','FontWeight',
     'bold','FontSize',14,...
    'FontName','Times New Roman',...
    'FontAngle','italic');

% Create xlabel
xlabel('X');

% Create ylabel
ylabel('f(X)')
```

Matlab Colors

Up to now we've been specifying colors with simple names. The most general way to get any color on the spectrum is to use the RGB [Red Green Blue] triple representation.

RGB Value	Short Name	Long Name
[1 1 0]	y	yellow
[1 0 1]	m	magenta
[0 1 1]	c	cyan
[1 0 0]	r	red
[0 1 0]	g	green
[0 0 1]	b	blue
[1 1 1]	w	white
[0 0 0]	k	black

Points With Standard Errors I

Nice function to plot points with standard errors (Y-e,Y+e) (e.g. regression residuals)

Points With Standard Errors I

Nice function to plot points with standard errors (Y-e,Y+e) (e.g. regression residuals)
First create errors *e=rand(100,1)* (must be positive)

Points With Standard Errors I

Nice function to plot points with standard errors (Y-e,Y+e) (e.g. regression residuals)
First create errors *e=rand(100,1)* (must be positive)

```
% Create figure
figure1 = figure('Color',[0.9529 0.8706 0.7333]);

% Create axes
axes('Parent',figure1);
box('on');
hold('all');

% Create errorbar
errorbar(X(1:10),Y(1:10),e(1:10),'Marker','o',
'LineStyle','none',...'Color',[0.03922 0.1412 0.4157])
```

Points With Standard Errors II

```
% Create xlabel
xlabel('X','FontWeight','bold','FontName',
'Aharoni');

% Create ylabel
ylabel('Predicted Value','FontWeight','bold',
'FontName','Aharoni');

% Create title
title('Predicted Values of Y given X','FontSize',
18,'FontName','Aharoni');
```

Plot Two Lines I

Firs Create Inputs

Plot Two Lines I

Firs Create Inputs

```
x2 = 0:0.01:20;
y1 = 200*exp(-0.05*x2).*sin(x2);
y2 = 0.8*exp(-0.5*x2).*sin(10*x2);
```

```
% Create figure
figure1 = figure('Color',[0.9412 0.9412 0.9412]);

% Create Plot
plotyy(x2,y1,x2,y2,'plot');

% Create xlabel
xlabel('x2');

% Create title
title('y1 and y2 by x2');
```

Plot Two Lines II

Use the following code to add horizontal Y-Labels

```
% Create textbox
annotation(figure1,'textbox',[0.9232 0.4954 0.02778
    0.03604],'String',{'y2'},'LineStyle','none',
'EdgeColor',[1 1 1]);

% Create textbox
annotation(figure1,'textbox',[0.09406 0.5045
    0.02192 0.02445],'String',{'y1'},'FitBoxToText',
'off', 'LineStyle','none');
```

Object Positioning

Positioning in Matlab follows a Uniform Format

Object Positioning

Positioning in Matlab follows a Uniform Format

- Position is given as [left, bottom, width, height]

Object Positioning

Positioning in Matlab follows a Uniform Format

- Position is given as [left, bottom, width, height]
- Left and Bottom are the normalized distances from the left and bottom edges of the figure window

Object Positioning

Positioning in Matlab follows a Uniform Format

- Position is given as [left, bottom, width, height]
- Left and Bottom are the normalized distances from the left and bottom edges of the figure window
- Width and Height are the normalized dimensions

Object Positioning

Positioning in Matlab follows a Uniform Format

- Position is given as [left, bottom, width, height]
- Left and Bottom are the normalized distances from the left and bottom edges of the figure window
- Width and Height are the normalized dimensions
- Normalized means [0,1] proportion of total height and length

Object Positioning

Positioning in Matlab follows a Uniform Format

- Position is given as [left, bottom, width, height]
- Left and Bottom are the normalized distances from the left and bottom edges of the figure window
- Width and Height are the normalized dimensions
- Normalized means [0,1] proportion of total height and length
- In the annotation command, Normalized distance can be changed to true distance in measurement Units

Bar Graph I

First Create Appropriate Data

Bar Graph I

First Create Appropriate Data

```
grp = round(1+rand(100,1));
yb = ceil(10*(rand(100,1).^grp));
t = tabulate(yb);
bg1 = t(:,1);
bg2 = t(:,2);
yb1 = yb(find(grp==1));
yb2 = yb(find(grp==2));
t1 = tabulate(yb1);
t2 = tabulate(yb2);
gbv = t1(:,1);
gby1 = t1(:,2);
gby2 = t2(:,2);
```

Bar Graph II

Now Create a Bar Graph Using...

Bar Graph II

Now Create a Bar Graph Using...

```
% Create figure
figure1 = figure;

% Create bar
bar(t(:,1),t(:,2),'k')

% Create xlabel
xlabel('Yb');

% Create ylabel
ylabel('Frequency');

% Create title
title('Bar Graph of Yb');
```

Bar Graph III

Now Create a Grouped Bar Graph...

Bar Graph III

Now Create a Grouped Bar Graph...

```
% Create figure
figure1 = figure;

% Create multiple lines
% using matrix input
bar(gbv,[gby1 gby2])

% Create xlabel
xlabel('Yb');

% Create ylabel
ylabel('Frequency');
```

Bar Graph III

Now Create a Grouped Bar Graph...

```
% Create figure
figure1 = figure;

% Create multiple lines
% using matrix input
bar(gbv,[gby1 gby2])

% Create xlabel
xlabel('Yb');

% Create ylabel
ylabel('Frequency');

% Create title
title('Bar Graph of Yb
by Grp');

% Create legend
legend('grp = 1',
'grp = 2')
```

Pie Chart

The data vector gbv is in appropriate format

Pie Chart

The data vector gbv is in appropriate format
Now Create a Pie Chart...

Pie Chart

The data vector gbv is in appropriate format
Now Create a Pie Chart...

Pie Chart

The data vector gbv is in appropriate format
Now Create a Pie Chart...

1. First construct the cell array *labs* = {*'1' '2' '3' '4' '5' '6' '7' '8' '9' '10'*}

Pie Chart

The data vector gbv is in appropriate format
Now Create a Pie Chart...

1. First construct the cell array *labs* = {*'1' '2' '3' '4' '5' '6' '7' '8' '9' '10'*}
2. *labs* will serve as the labels for the wedges in the pie chart

Pie Chart

The data vector gbv is in appropriate format
Now Create a Pie Chart...

1. First construct the cell array *labs* = {*'1' '2' '3' '4' '5' '6' '7' '8' '9' '10'*}
2. *labs* will serve as the labels for the wedges in the pie chart
3. Must be the length of the input to the pie chart

Pie Chart

The data vector gbv is in appropriate format
Now Create a Pie Chart...

1. First construct the cell array *labs* = {*'1' '2' '3' '4' '5' '6' '7' '8' '9' '10'*}
2. *labs* will serve as the labels for the wedges in the pie chart
3. Must be the length of the input to the pie chart
4. Then issue the command *pie(gbv,labs)*

Pie Chart

The data vector gbv is in appropriate format
Now Create a Pie Chart...

1. First construct the cell array *labs* = {*'1' '2' '3' '4' '5' '6' '7' '8' '9' '10'*}
2. *labs* will serve as the labels for the wedges in the pie chart
3. Must be the length of the input to the pie chart
4. Then issue the command *pie(gbv,labs)*
5. The first argument is a vector giving frequencies

General 3D plot

```
%First Create Some Data
Z = 0:pi/50:10*pi;
X = sin(Z);
Y = cos(Z);

% Create figure
figure1 = figure;

% Create plot3
plot3(X1,Y1,Z1);

% Set Perspective
view([-37.5 22]);
```

General 3D plot

```
%First Create Some Data
Z = 0:pi/50:10*pi;
X = sin(Z);
Y = cos(Z);

% Create figure
figure1 = figure;

% Create plot3
plot3(X1,Y1,Z1);

% Set Perspective
view([-37.5 22]);

% Create xlabel
xlabel('X');

% Create ylabel
ylabel('Y');

% Create zlabel
zlabel('Z');

% Create title
title('Z by X and Y');
```

The View

The biggest change from 2D to 3D is in perspective

The View

The biggest change from 2D to 3D is in perspective

- Perspective is set as view([RZ RV])

The View

The biggest change from 2D to 3D is in perspective

- Perspective is set as view([RZ RV])
- RZ is the rotation about the Z-axis

The View

The biggest change from 2D to 3D is in perspective

- Perspective is set as view([RZ RV])
- RZ is the rotation about the Z-axis
- RV is vertical rotation

The View

The biggest change from 2D to 3D is in perspective

- Perspective is set as view([RZ RV])
- RZ is the rotation about the Z-axis
- RV is vertical rotation
- Best strategy is to play with values...

Bivariate Histogram

```
%First Create Some Data
x1 = randn(1000,1);
x2 = randn(1000,1)+x1.^2;

% Create Plot
hist3([x1 x2])

% Set Perspective
view([-147.5 18]);
```

Bivariate Histogram

```
%First Create Some Data
x1 = randn(1000,1);
x2 = randn(1000,1)+x1.^2;

% Create Plot
hist3([x1 x2])

% Set Perspective
view([-147.5 18]);

% Create title
title('Bivariate
Histogram
of x1 and x2');

% Create xlabel
xlabel('x1');

% Create zlabel
zlabel('Frequency');

% Create ylabel
ylabel('x2');
```

3D Surface Plot

```
%First Create Some Data
g1=-2:0.1:2;
g2=g1;
[g1m ,g2m] = meshgrid
(g1,g2);
fg1g2 = normpdf(g1m)
.*normpdf(g2m);

% Create Plot
surf(g1,g2,fg1g2)

% Set Perspective
view([-147.5 18]);
```

3D Surface Plot

```
%First Create Some Data
g1=-2:0.1:2;
g2=g1;
[g1m ,g2m] = meshgrid
(g1,g2);
fg1g2 = normpdf(g1m)
.*normpdf(g2m);

% Create Plot
surf(g1,g2,fg1g2)

% Set Perspective
view([-147.5 18]);
```

```
% Create title
title('Bivariate
Standard Normal
Density');

% Create xlabel
xlabel('g1');

% Create zlabel
zlabel('f(g1,g2)');

% Create ylabel
ylabel('g2');
```