Graphics with MATLAB

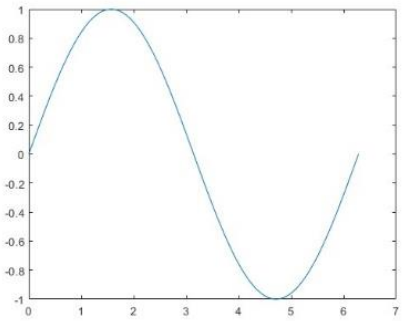
Basic Line Plot

Create x as a vector of linearly spaced values between 0 and 2π. Use an increment of $\pi/100$ between the values. Create y as sine values of x. Create a line plot of the data

Program

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```

Output



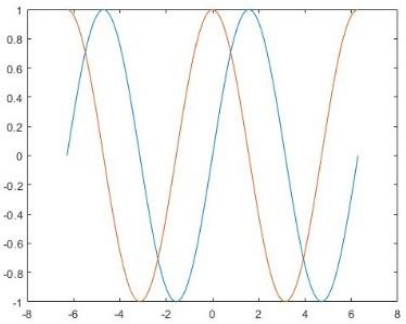
Multiple Line Plot

Define x as 100 linearly spaced values between -2π and 2π. Define y1 and y2 as sine and cosine values of x. Create a line plot of both sets of data.

Program

```
x = linspace(-2*pi,2*pi);
y1 = sin(x);
y2 = cos(x);
figure
plot(x,y1,x,y2)
```

Output



Line Plot from Matrix

Define Y as the 4-by-4 matrix returned by the magic function. Create a 2-D line plot of Y. MATLAB plots each matrix column as a separate line.

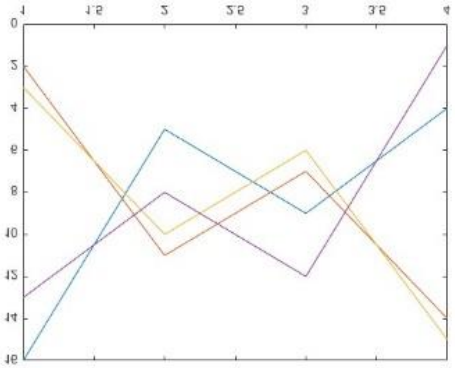
Program

```
Y = magic(4)
figure
plot(Y)
```

Output

```
Y =

    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```



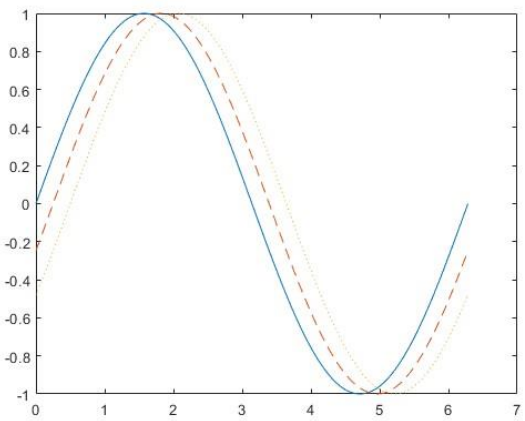
Line Style

Plot three sine curves with a small phase shift between each line. Use the default line style for the first line. Specify a dashed line style for the second line and a dotted line style for the third line.

Program

```
x = 0:pi/100:2*pi;
y1 = sin(x);
y2 = sin(x-0.25);
y3 = sin(x-0.5);
figure
plot(x,y1,x,y2,'--',x,y3,':')
```

Output



Specify Line Style, Colour, Marker

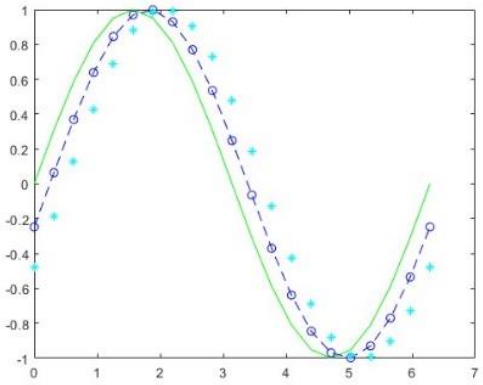
Plot three sine curves with a small phase shift between each line. Use a green line with no markers for the first sine curve. Use a blue dashed line with circle markers for the second sine curve. Use only cyan star markers for the third sine curve.

Program

```
x = 0:pi/10:2*pi;
y1 = sin(x);
y2 = sin(x-0.25);
y3 = sin(x-0.5);

figure
plot(x,y1,'g',x,y2,'b--o',x,y3,'c*')
```

Output



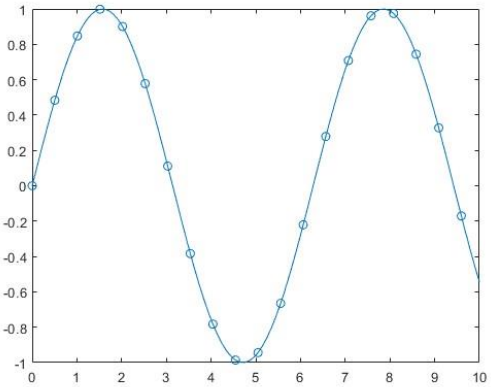
Display Markers at Specific Data Points

Create a line plot and display markers at every fifth data point by specifying a marker symbol and setting the MarkerIndices property as a name-value pair.

Program

```
x = linspace(0,10);
y = sin(x);
plot(x,y,'-o','MarkerIndices',1:5:length(y))
```

Output



Specify Line Width, Marker Size, and Marker Colour

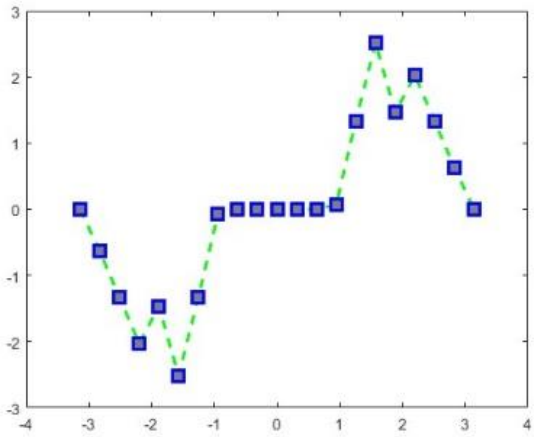
Create a line plot and use the LineSpec option to specify a dashed green line with square markers. Use Name,Value pairs to specify the line width, marker size, and marker colors. Set the marker edge color to blue and set the marker face color using an RGB color value.

Program

```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));

figure
plot(x,y,'--gs',...
    'LineWidth',2,...
    'MarkerSize',10,...
    'MarkerEdgeColor','b',...
    'MarkerFaceColor',[0.5,0.5,0.5])
```

Output



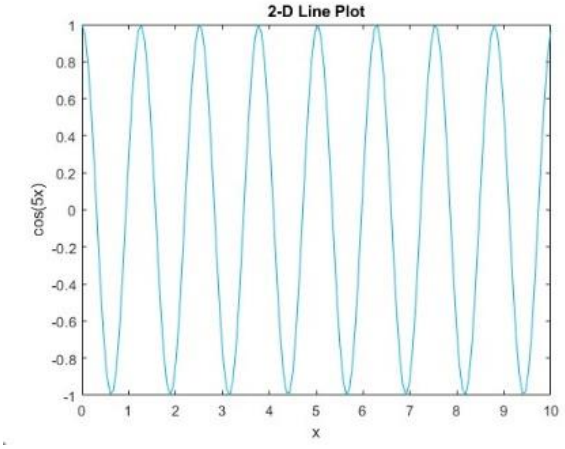
Add Title and Axis Labels

Use the linspace function to define x as a vector of 150 values between 0 and 10. Define y as cosine values of x. Create a 2-D line plot of the cosine curve. Change the line color to a shade of blue-green using an RGB color value. Add a title and axis labels to the graph using the title, xlabel, and ylabel functions.

Program

```
x = linspace(0,10,150);
y = cos(5*x);
figure
plot(x,y,'Color',[0,0.7,0.9])
title('2-D Line Plot')
xlabel('x')
ylabel('cos(5x)')
```

Output



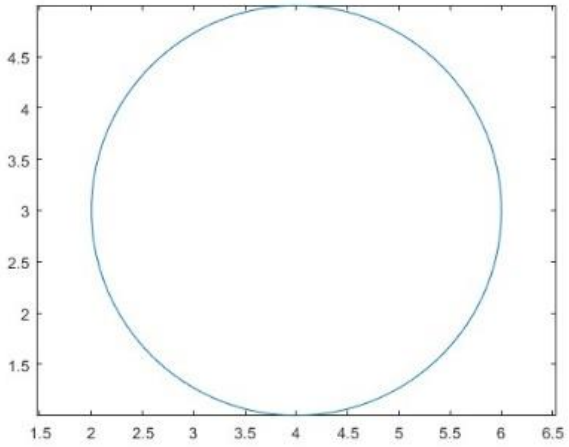
Plot Circle

Plot a circle centered at the point (4,3) with a radius equal to 2. Use axis equal to use equal data units along each coordinate direction.

Program

```
r = 2;
xc = 4;
yc = 3;
theta = linspace(0,2*pi);
x = r*cos(theta) + xc;
y = r*sin(theta) + yc;
plot(x,y)
axis equal
```

Output



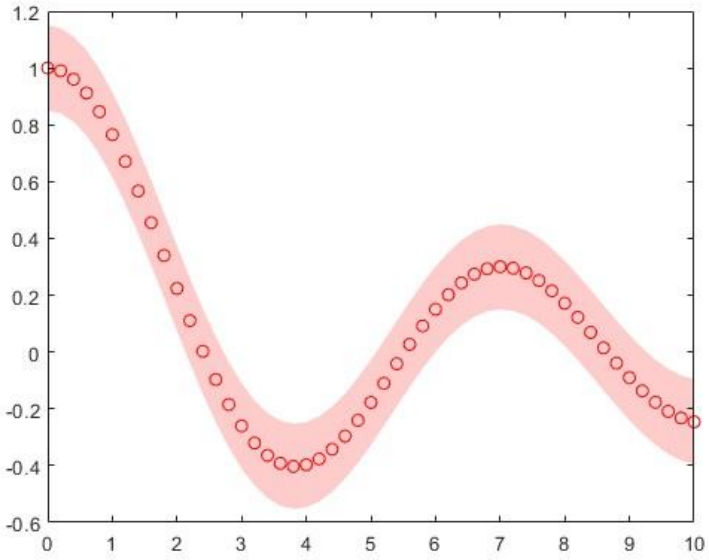
Line Plot with Confidence Bounds

Create a plot with confidence bounds using the fill function to draw the confidence bounds and the plot function to draw the data points. Use dot notation syntax object.PropertyName to customize the look of the plot.

Program

```
x = 0:0.2:10;
y = besselj(0, x);
xconf = [x x(end:-1:1)] ;
yconf = [y+0.15 y(end:-1:1)-0.15];
figure
p = fill(xconf,yconf,'red');
p.FaceColor = [1 0.8 0.8];
p.EdgeColor = 'none';
hold on
plot(x,y,'ro')
hold off
```

Output



Histogram

histogram(X) creates a histogram plot of X. The histogram function uses an automatic binning algorithm that returns bins with a uniform width, chosen to cover the range of elements in X and reveal the underlying shape of the distribution. histogram displays the bins as rectangular bars such that the height of each rectangle indicates the number of elements in the bin.

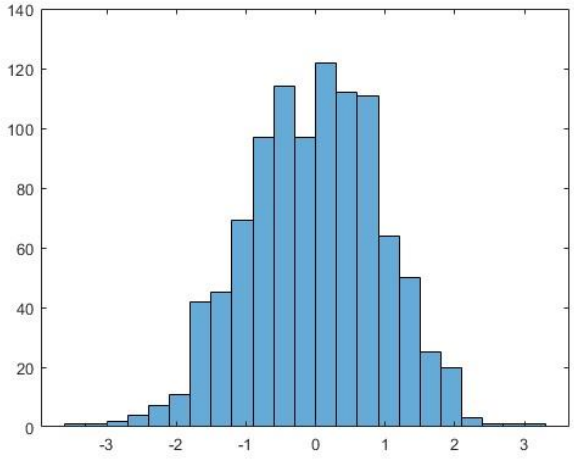
Basic Histogram

Generate 1000 random numbers and create a histogram. The `histogram` function automatically chooses an appropriate number of bins to cover the range of values in `x` and show the shape of the underlying distribution.

Program

```
x = randn(1000,1);
h = histogram(x)
```

Output



Specify Number of Histogram Bins

Plot a histogram of 1,000 random numbers sorted into 25 equally spaced bins.

Program

```
x = randn(1000,1);
nbins = 25;
h = histogram(x,nbins)
counts = h.Values
```

Output

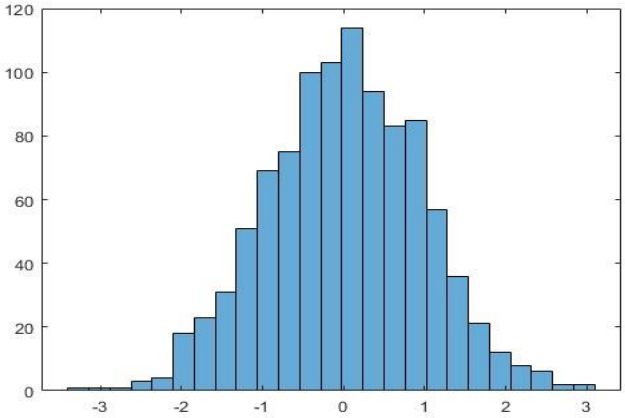
```
counts =

  Columns 1 through 21

     1     0     0     1     3     3    10    22    28    38    70    85   105    95   108   107    94    67    59    41    28

  Columns 22 through 25

    16    11     5     3
```



MoreBins

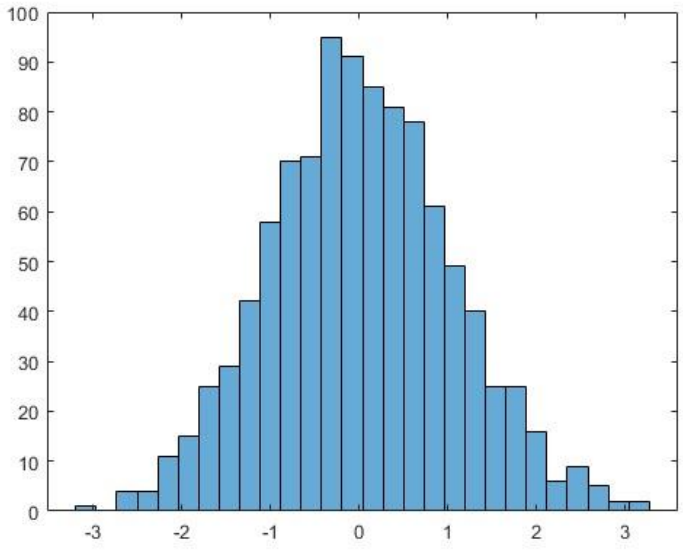
N = morebins(h) increases the number of bins in histogram h by 10% (rounded up to the nearest integer) and returns the new number of bins.

For bivariate histograms, this increases the bin count in both the *x* and *y* directions.

Program

```
x = randn(1000,1);
h = histogram(x)
morebins(h);
morebins(h)
```

Output



Box Chart

boxchart(ydata) creates a box chart, or box plot, for each column of the matrix ydata. If ydata is a vector, then boxchart creates a single box chart.

Each box chart displays the following information: the median, the lower and upper quartiles, any outliers (computed using the interquartile range), and the minimum and maximum values that are not outliers.

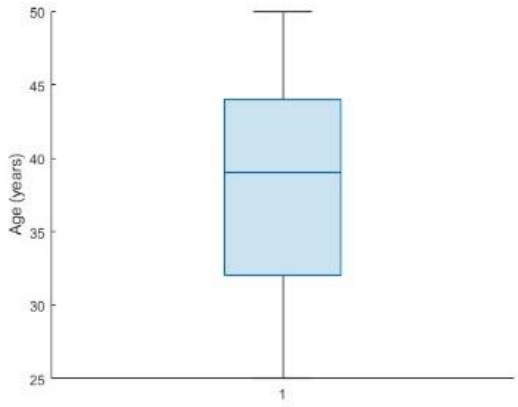
Basic Box Chart

Create a single box chart from a vector of ages. Use the box chart to visualize the distribution of ages. Load the patients data set. The Age variable contains the ages of 100 patients. Create a box chart to visualize the distribution of ages.

Program

```
load patients
boxchart(Age)
ylabel('Age (years)')
```

Output



The median patient age of 39 years is shown as the line inside the box. The lower and upper quartiles of 32 and 44 years are shown as the bottom and top edges of the box, respectively. The whiskers, or lines that extend below and above the box, have endpoints that correspond to the youngest and oldest patients. The youngest patient is 25 years old, and the oldest is 50 years old. The data set contains no outliers, which would be represented by small circles.

Create Box Chart from Matrix Data

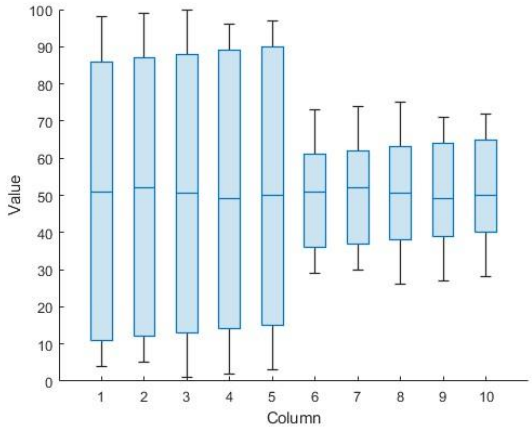
Use box charts to compare the distribution of values along the columns and the rows of a magic square. Create a magic square with 10 rows and 10 columns.

Program

Create a box chart for each column of the magic square. Each column has a similar median value (around 50). However, the first five columns of Y have greater interquartile ranges than the last five columns of Y. The interquartile range is the distance between the upper quartile (top edge of the box) and the lower quartile (bottom edge of the box).

```
Y = magic(10)
boxchart(Y)
xlabel('Column')
ylabel('Value')
```

Output

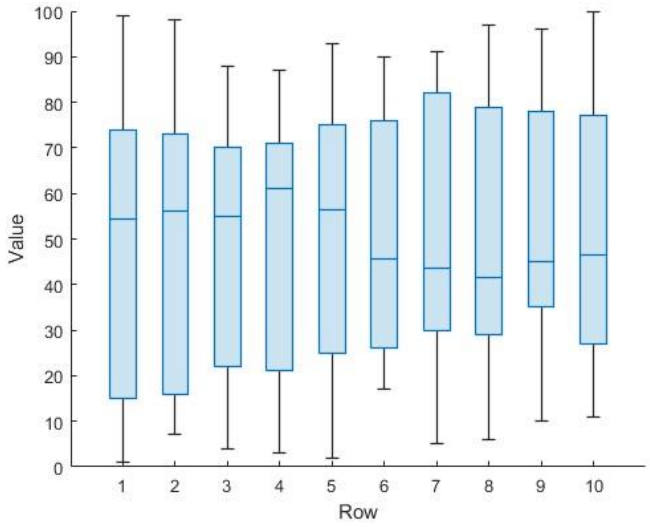


Program

Create a box chart for each row of the magic square. Each row has a similar interquartile range, but the median values differ across the rows.

```
Y = magic(10)
boxchart(Y')
xlabel('Row')
ylabel('Value')
```

Output



Swarm Chart

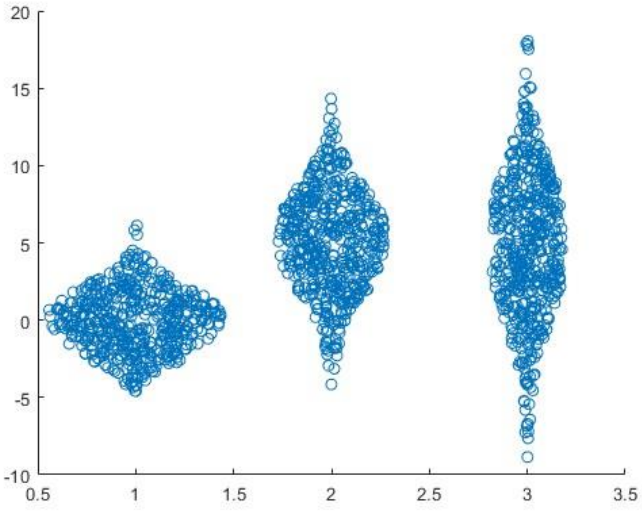
swarmchart(x,y) displays a swarm chart, which is a scatter plot with the points offset (jittered) in the x-dimension. The points form distinct shapes, and the outline of each shape is similar to a violin plot. Swarm charts help you to visualize discrete x data with the distribution of the y data. At each location in x, the points are jittered based on the kernel density estimate of y.

- To plot one set of points, specify x and y as vectors of equal length.
- To plot multiple sets of points on the same set of axes, specify at least one of x or y as a matrix.

Program

```
x = [ones(1,500) 2*ones(1,500) 3*ones(1,500)];
y1 = 2 * randn(1,500);
y2 = 3 * randn(1,500) + 5;
y3 = 5 * randn(1,500) + 5;
y = [y1 y2 y3];
swarmchart(x,y)
```

Output



Heatmap

heatmap(tbl,xvar,yvar) creates a heatmap from the table tbl. The xvar input indicates the table variable to display along the *x*-axis. The yvar input indicates the table variable to display along the *y*-axis. The default colors are based on a count aggregation, which totals the number of times each pair of *x* and *y* values appears together in the table.

Heatmap from Tabular Data

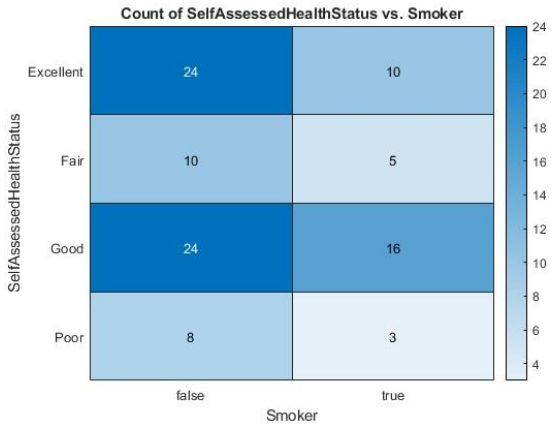
Create a heatmap from a table of data for medical patients.

Load the patients data set and create a table from a subset of the variables loaded into the workspace. Then create a heatmap that counts the total number of patients with the same set of Smoker and SelfAssessedHealthStatus values.

Program

```
load patients
tbl = table(LastName,Age,Gender,SelfAssessedHealthStatus,...
    Smoker,Weight,Location);
h = heatmap(tbl,'Smoker','SelfAssessedHealthStatus');
```

Output



Word Cloud

wordcloud(tbl,wordVar,sizeVar) creates a word cloud chart from the table tbl. The variables wordVar and sizeVar in the table specify the words and word sizes respectively.

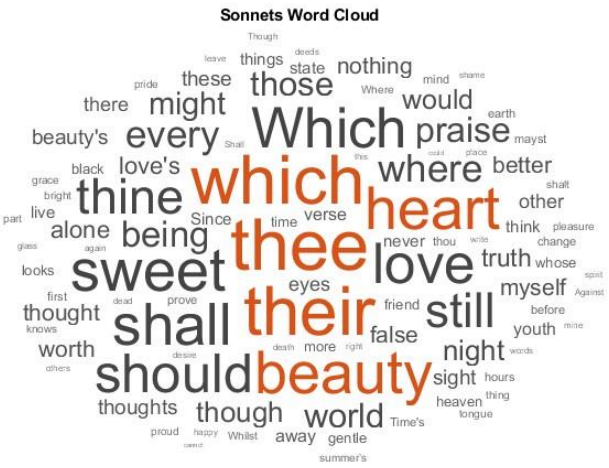
Program

Load the example data sonnetsTable. The table tbl contains a list of words in the variable Word, and the corresponding frequency counts in the variable Count. Plot the table data using wordcloud. Specify the words and corresponding word sizes to be the Word and Count variables respectively.

```
load sonnetsTable
head(tbl)
figure
wordcloud(tbl,'Word','Count');
title("Sonnets Word Cloud")
```

Output

Word	Count
{'''tis' }	1
{''Amen'' }	1
{''Fair' }	2
{''Gainst'}	1
{''Since' }	1
{''This' }	2
{''Thou' }	1
{''Thus' }	1



Pie chart

pie(X) draws a pie chart using the data in X. Each slice of the pie chart represents an element in X.

- If sum(X) ≤ 1, then the values in X directly specify the areas of the pie slices. pie draws only a partial pie if sum(X) < 1.
- If sum(X) > 1, then pie normalizes the values by X/sum(X) to determine the area of each slice of the pie.
- If X is of data type categorical, the slices correspond to categories. The area of each slice is the number of elements in the category divided by the number of elements in X.

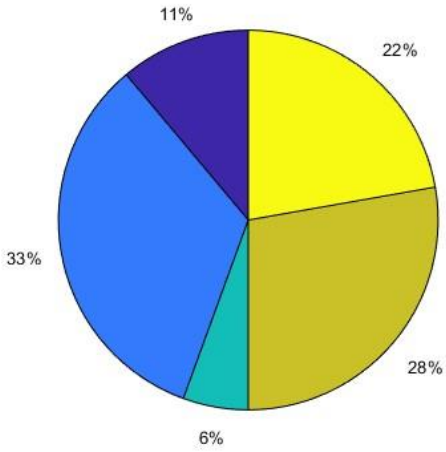
Basic Pie Chart

Create a pie chart of vector X.

Program

```
X = [1 3 0.5 2.5 2];
pie(X)
```

Output



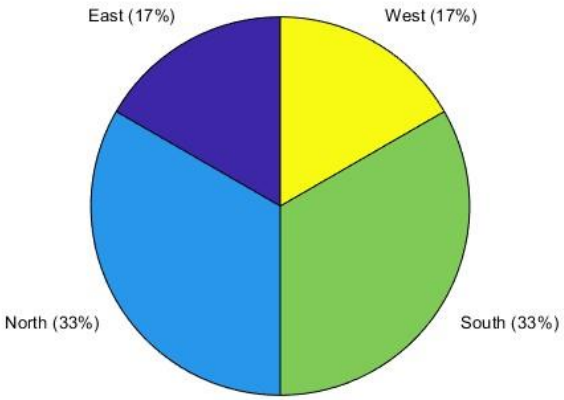
Categorical Pie Chart

Create a pie chart of categorical X

Program

```
X = categorical({'North','South','North','East','South','West'});
pie(X)
```

Output



Scatter Plot

scatter(x,y) creates a scatter plot with circular markers at the locations specified by the vectors x and y.

- To plot one set of coordinates, specify x and y as vectors of equal length.
- To plot multiple sets of coordinates on the same set of axes, specify at least one of x or y as a matrix.

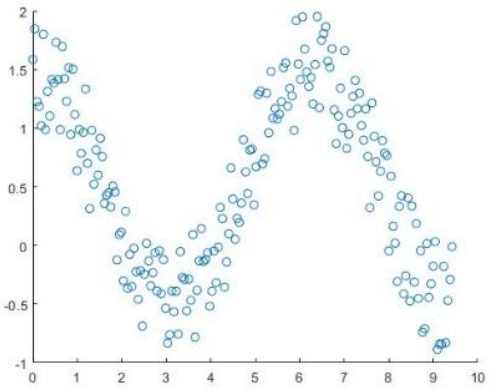
Create Scatter Plot

Create x as 200 equally spaced values between 0 and 3π. Create y as cosine values with random noise. Then, create a scatter plot.

Program

```
x = linspace(0,3*pi,200);
y = cos(x) + rand(1,200);
scatter(x,y)
```

Output



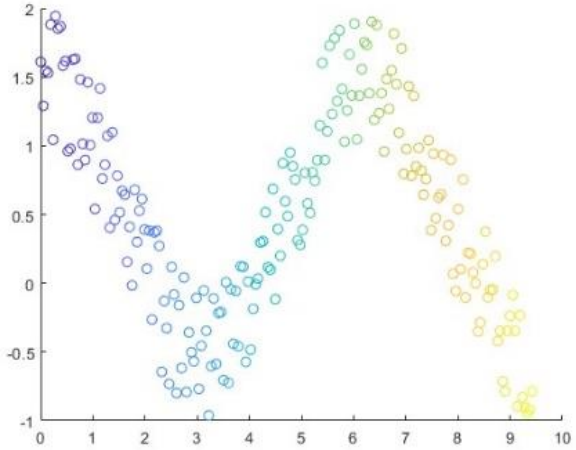
Colour of Circle

Create a scatter plot and vary the circle color.

Program

```
x = linspace(0,3*pi,200);
y = cos(x) + rand(1,200);
c = linspace(1,10,length(x));
scatter(x,y,[],c)
```

Output



Bubble Chart

bubblechart(x,y,sz) displays colored circular markers (bubbles) at the locations specified by the vectors x and y, with bubble sizes specified by sz.

- To plot one set of coordinates, specify x, y, and sz as vectors of equal length.
- To plot multiple sets of coordinates on the same set of axes, specify at least one of x, y, or sz as a matrix.

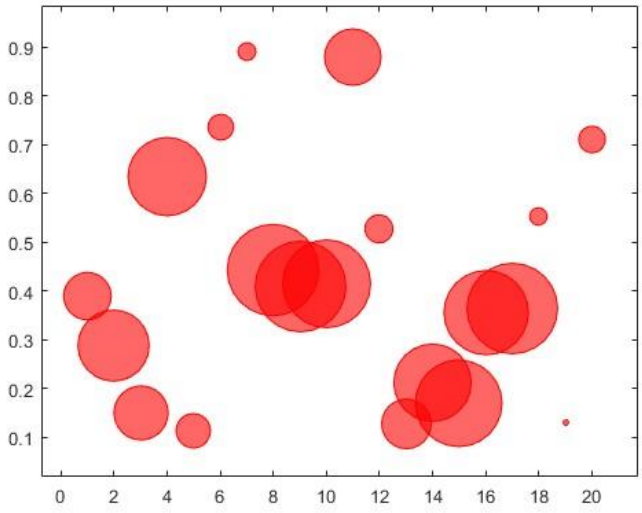
Create Bubble Chart with Specify Colour

Define the bubble coordinates as the vectors x and y. Define sz as a vector that specifies the bubble sizes. Then create a bubble chart of x and y, and specify the color as red. By default, the bubbles are partially transparent.

Program

```
x = 1:20;
y = rand(1,20);
sz = rand(1,20);
bubblechart(x,y,sz,'red');
```

Output



Bar Graph

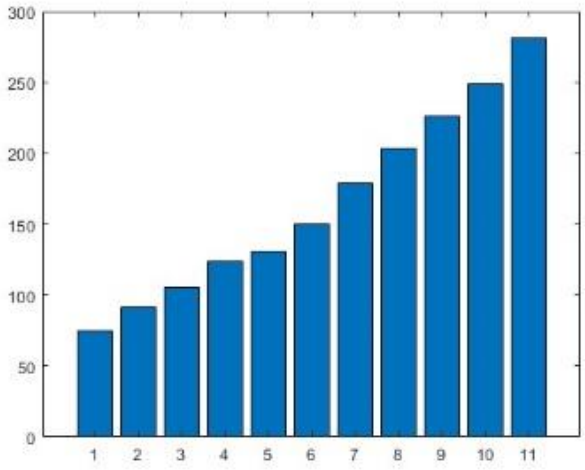
bar(y) creates a bar graph with one bar for each element in y.

- To plot a single series of bars, specify y as a vector of length *m*. The bars are positioned from 1 to *m* along the *x*-axis.
- To plot multiple series of bars, specify y as a matrix with one column for each series.

Basic Bar Graph Program

```
y = [75 91 105 123.5 131 150 179 203 226 249 281.5];
bar(y)
```

Output



Surface Plot

surf(X,Y,Z) creates a three-dimensional surface plot, which is a three-dimensional surface that has solid edge colors and solid face colors. The function plots the values in matrix Z as heights above a grid in the *x-y* plane defined by X and Y. The color of the surface varies according to the heights specified by Z.

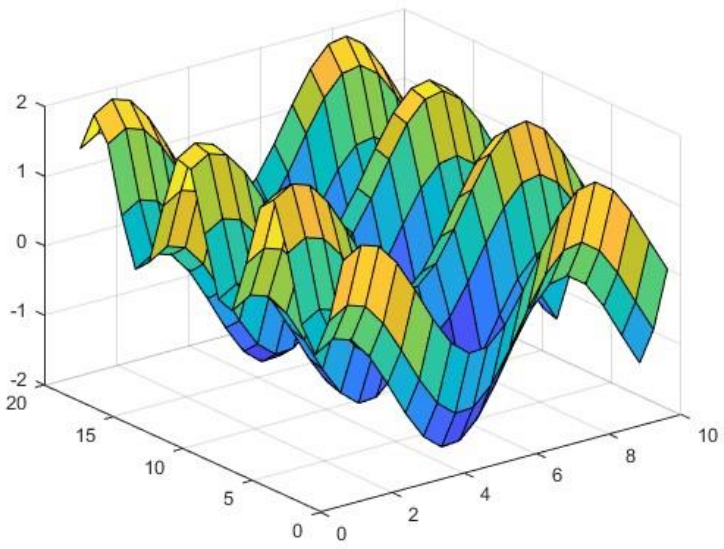
Basic Surface Plot

Create three matrices of the same size. Then plot them as a surface. The surface plot uses Z for both height and color.

Program

```
[X,Y] = meshgrid(1:0.5:10,1:20);
Z = sin(X) + cos(Y);
surf(X,Y,Z)
```

Output



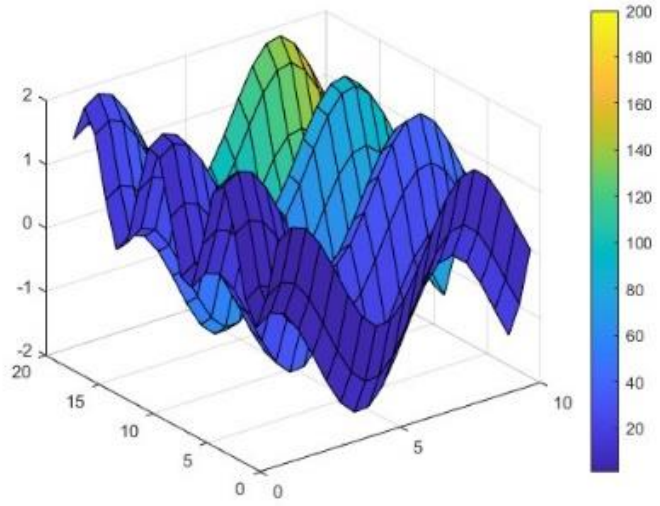
Specify Colourmap Colour for Surface Plot

Specify the colors for a surface plot by including a fourth matrix input, C. The surface plot uses Z for height and C for color. Specify the colors using a *colormap*, which uses single numbers to stand for colors on a spectrum. When you use a colormap, C is the same size as Z. Add a color bar to the graph to show how the data values in C correspond to the colors in the colormap.

Program

```
[X,Y] = meshgrid(1:0.5:10,1:20);
Z = sin(X) + cos(Y);
C = X.*Y;
surf(X,Y,Z,C)
colorbar
```

Output



Ribbon Plot

ribbon(Z) plots the columns of Z as three-dimensional ribbons of uniform width, where *y*-coordinates range from 1 to the number of rows in Z. Ribbons advance along the *x*-axis centered at unit intervals.

Basic Ribbon Plot

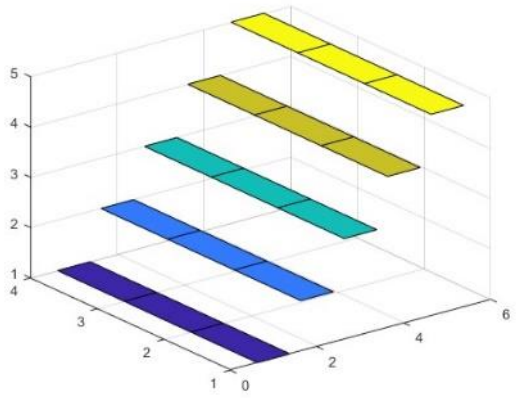
Create a plot with five ribbons at increasing heights. First, create a 5-by-5 matrix with elements corresponding to ribbon heights. Each column of Z represents one ribbon, plotted at a constant *x*-coordinate corresponding to the column number and with *y*-coordinates corresponding to the row numbers of Z.

Program

```
Z = repmat(1:5,4,1);
ribbon(Z)
```

Output

```
basic_ribbon
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
     1     2     3     4     5
```



Specify Ribbon Locations

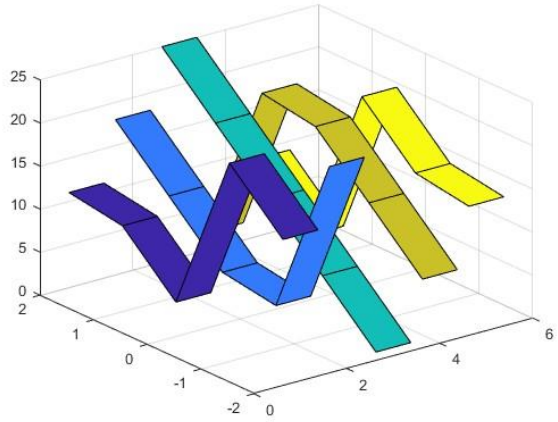
Create a 5-by-5 matrix with the magic function. Create a ribbon plot of the matrix and specify the *y*-coordinates so each ribbon is centered at 0.

Program

```
Z = magic(5);
Y = [-2 -1 0 1 2];
ribbon(Y,Z);
disp(Z)
```

Output

```
ribbon_location
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```



Ellipsoid

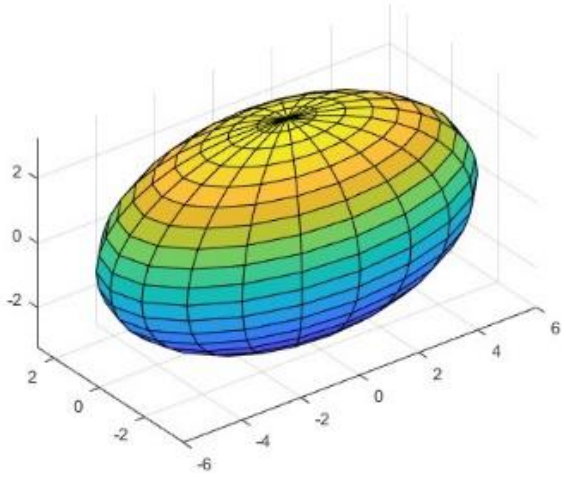
Create and Display Ellipsoid

Create and plot an ellipsoid with a center at (0, –0.5, 0) and semiaxis lengths (6, 3.25, 3.25). Use axis equal to use equal data units along each coordinate direction.

Program

```
ellipsoid(0,-0.5,0,6,3.25,3.25)
axis equal
```

Output



Sphere

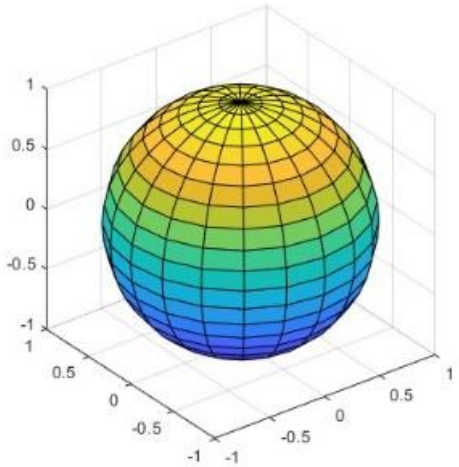
Display Unit Sphere

Create and plot a sphere with a radius equal to 1. Use axis equal to use equal data units along each coordinate direction.

Program

```
sphere
axis equal
```

Output



Specify Sphere Radius and Location

Specify the radius and location of a sphere by modifying the returned X, Y, and Z coordinates.

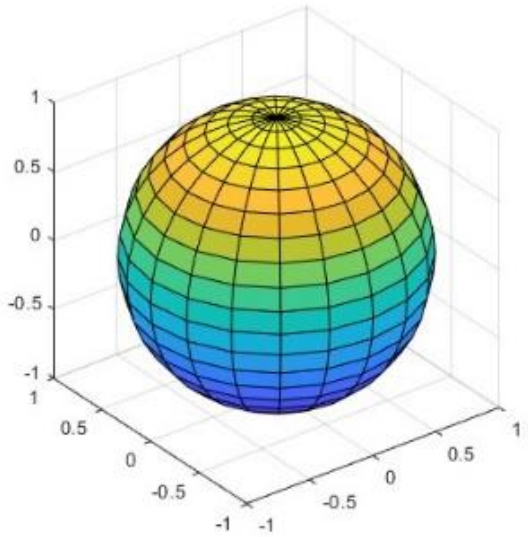
Define X, Y, and Z as coordinates of a unit sphere.

Program

Plot the unit sphere centered at the origin.

```
[X,Y,Z] = sphere;
%Plot the unit sphere centered at the origin.
surf(X,Y,Z)
axis equal
```

Output

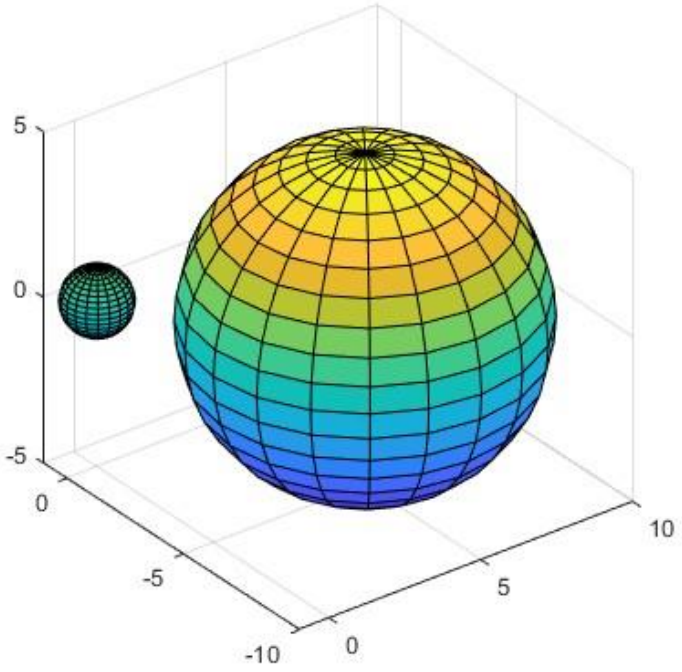


Program

Define `X2`, `Y2`, and `Z2` as coordinates of a sphere with a radius of 5 by multiplying the coordinates of the unit sphere. Plot the second sphere, centering it at `(5,-5,0)`.

```
[X,Y,Z] = sphere;
%Plot the unit sphere centered at the origin.
surf(X,Y,Z)
axis equal
%Define X2, Y2, and Z2 as coordinates of a sphere with a radius of 5 by
multiplying the coordinates of the unit sphere. Plot the second sphere, centering
it at (5,-5,0).
hold on
r = 5;
X2 = X * r;
Y2 = Y * r;
Z2 = Z * r;
surf(X2+5,Y2-5,Z2)
```

Output



Display Sphere with Different Number of Faces

Call the tiledlayout function to create a 2-by-2 tiled chart layout. Call the nexttile function to create the axes. Then, use the sphere function to plot three spheres with different numbers of faces into different tiles of the chart by specifying the axes.

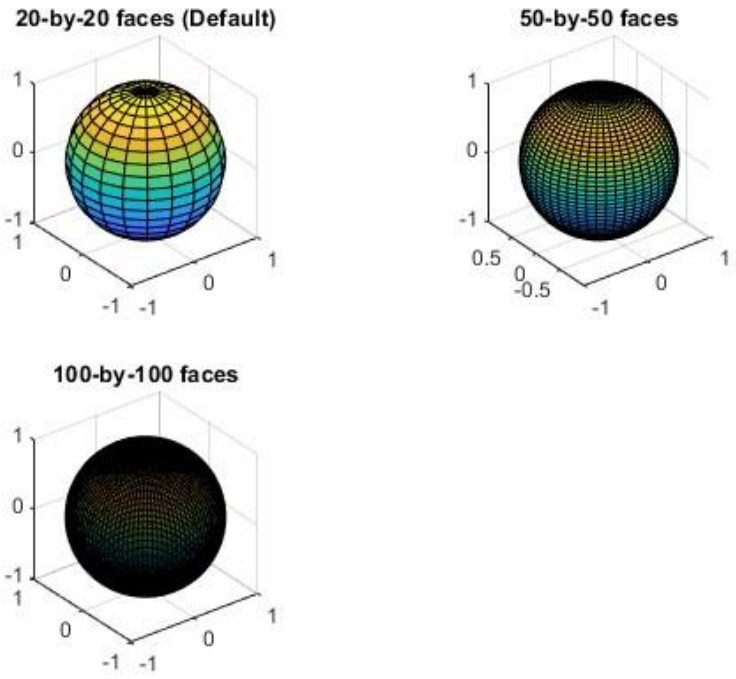
Program

```
tiledlayout(2,2);
ax1 = nexttile;
sphere(ax1);
axis equal
title('20-by-20 faces (Default)')

ax2 = nexttile;
sphere(ax2,50)
axis equal
title('50-by-50 faces')

ax3 = nexttile;
sphere(ax3,100)
axis equal
title('100-by-100 faces')
```

Output



Cylinder

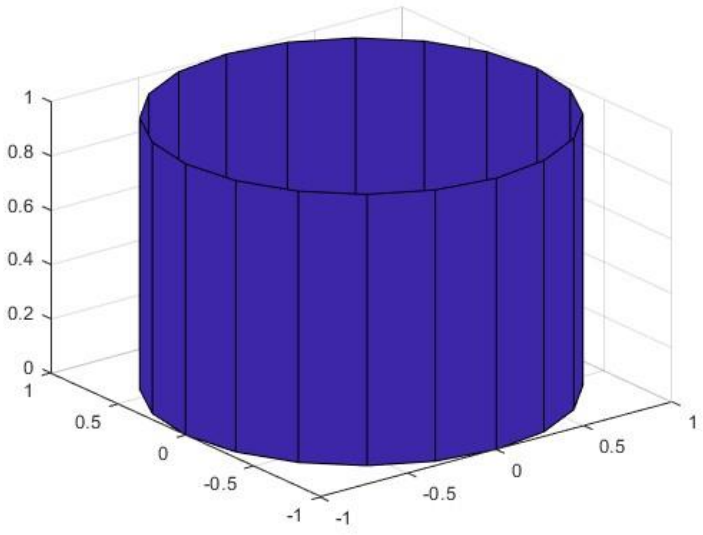
Display Unit Cylinder

Create and plot a cylinder with a radius equal to 1.

Program

```
cylinder
```

Output



Specify Cylinder Radius and Height

Specify the radius of a cylinder by including the input `r`. Then, specify the height of the cylinder by modifying the returned `Z` coordinate.

Define `X`, `Y`, and `Z` as coordinates of a cylinder with a radius of 4. Specify a height of 20 by modifying the `Z` coordinate. Plot the cylinder.

Program

```
r = 4;
[X,Y,Z] = cylinder(r);
h = 20;
Z = Z*h;
surf(X,Y,Z)
```

Output