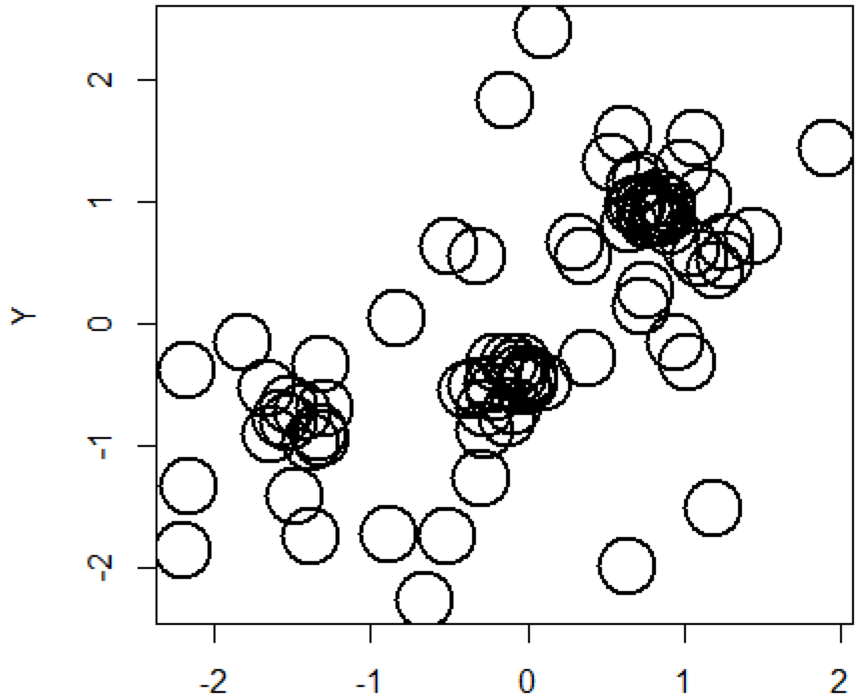
Introduction to K-means Clustering

K-means clustering: Algorithm

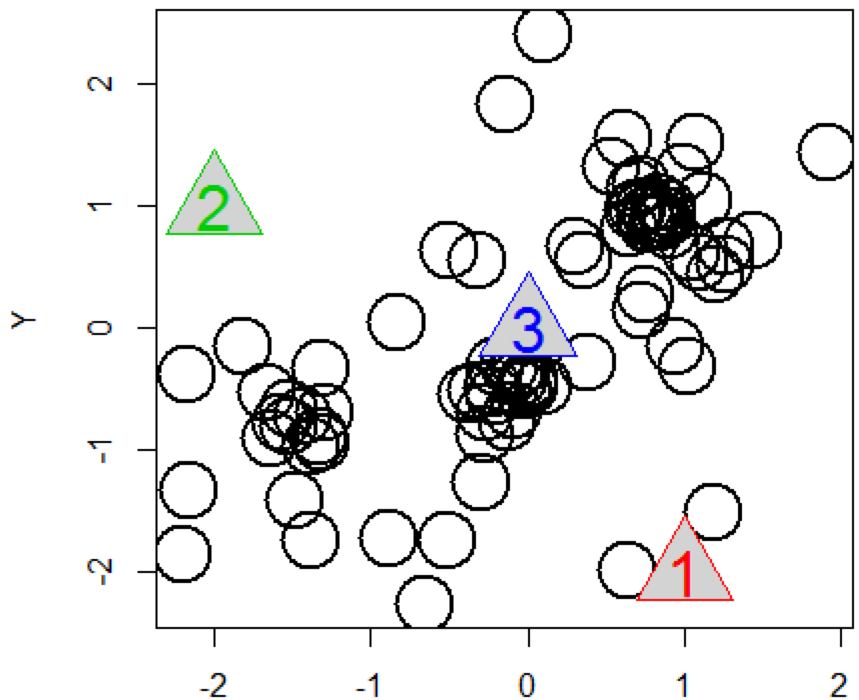
- Pre-requisites
  1. Get points in multi-dimensional space.
     - table, matrix, rectangular dataset
  2. Specify the number of clusters
     - Weakest point in algorithm (makes algorithm non-deterministic)
  3. Get a random center for each cluster
     - Another weak point in the algorithm
- Repeat until convergence:
  1. For each point, determine its closest cluster center and assign that point to that cluster
  2. Determine the centroid (mean) for each cluster of points

K-Means Clustering (0)



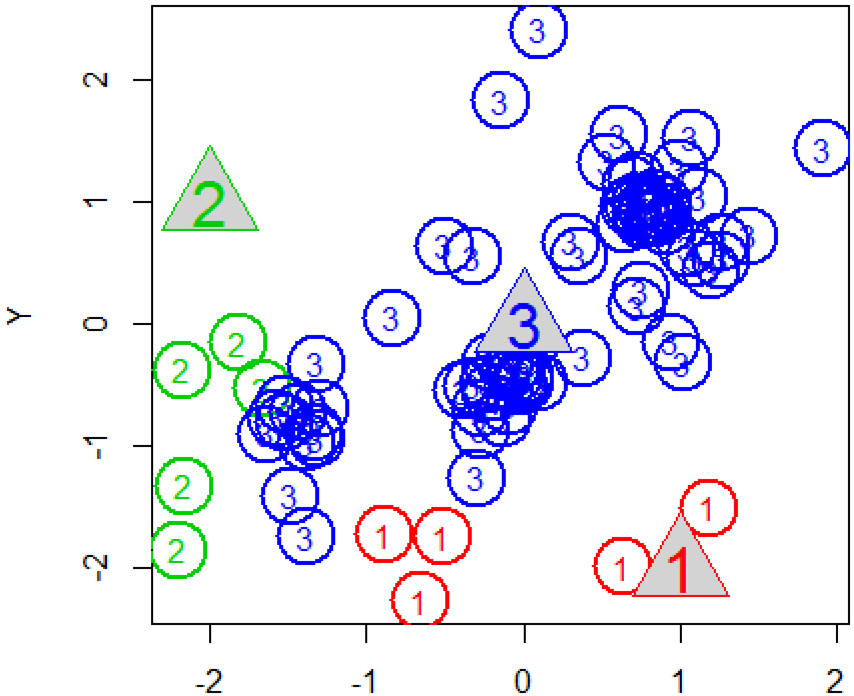
- Clustering starts by getting the data and representing the data as points in space. In this example the space is 2-dimensional.
- Each point describes an observation. An observation is an individual item.
- The dimensions are attributes that describe the item.

K-Means Clustering (1)



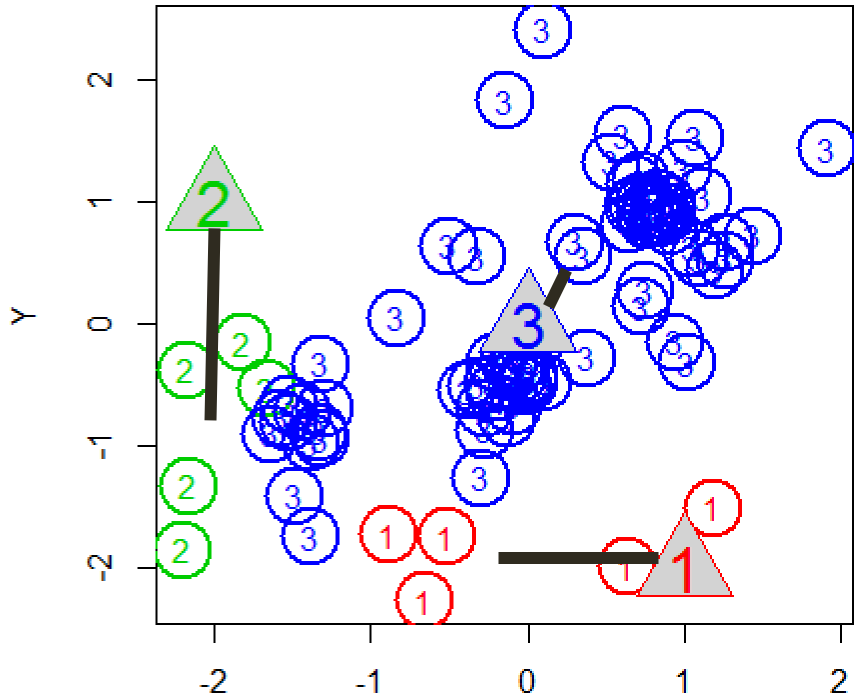
- Clustering continues by guessing, presuming, or specifying a number of clusters.
- Each centroid represents a cluster.
- The centroid positions are determined randomly. The centroids should be within the bounds of the points.

K-Means Clustering (2)



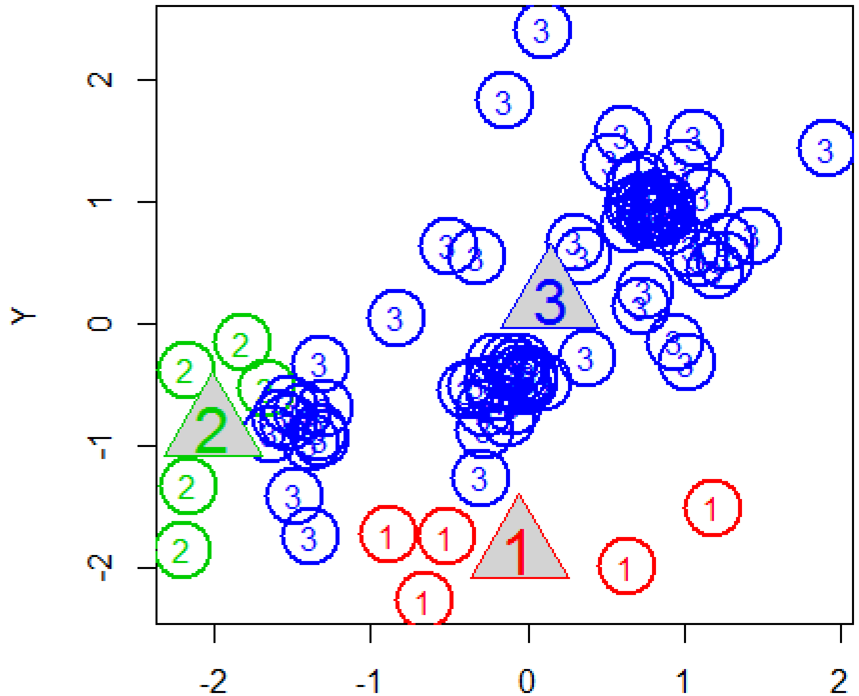
- Clustering continues by assigning each point to a cluster.
- For each point, the algorithm measures the distance to each centroid.
- For each point, the smallest distance to a centroid indicates the assignment.

K-Means Clustering (2)



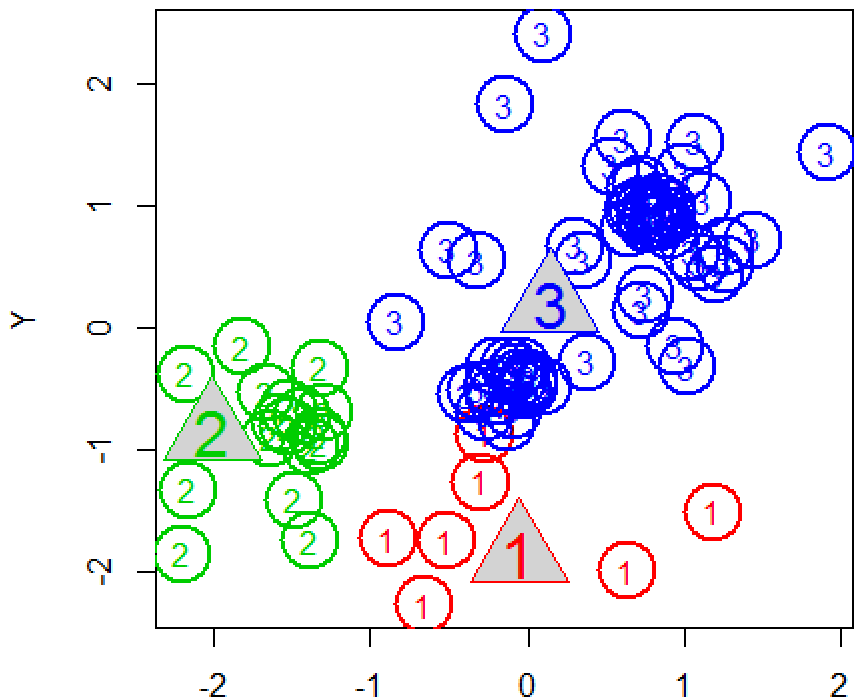
- Clustering continues by moving each centroid to the center of its cluster.

K-Means Clustering (3)



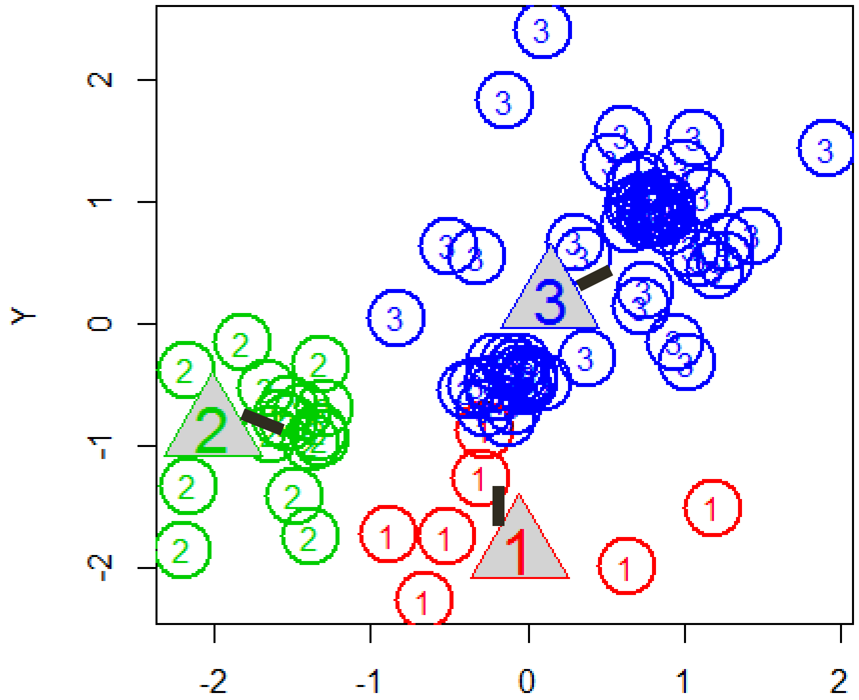
- Clustering continues by moving each centroid to the center of its cluster.

K-Means Clustering (4)



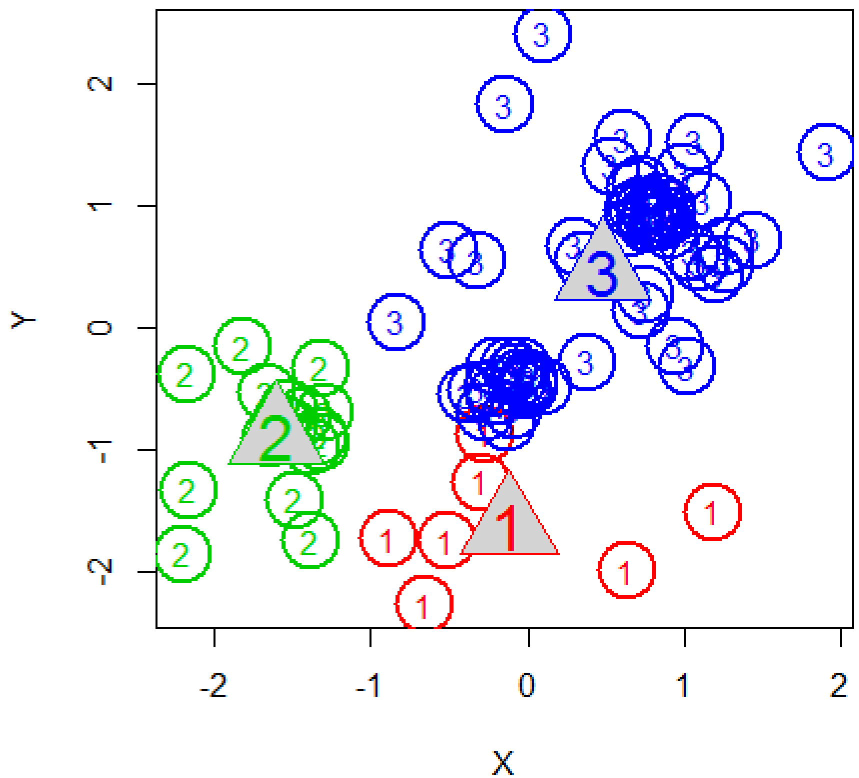
- Clustering continues by assigning each point to a cluster.
- For each point, the algorithm measures the distance to each centroid.
- For each point, the smallest distance to a centroid indicates the assignment.

K-Means Clustering (4)

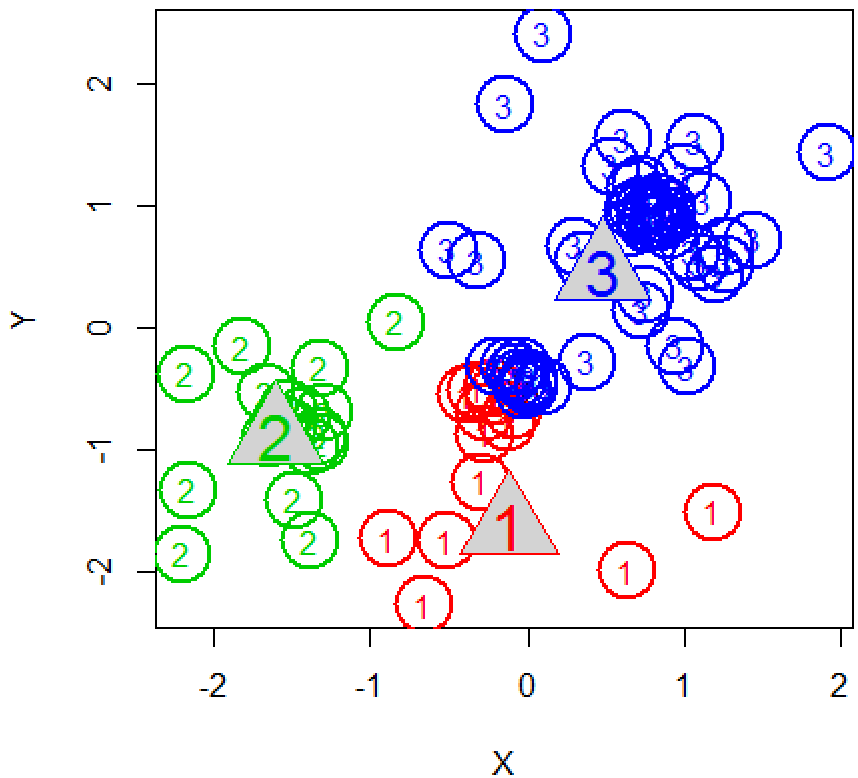


- Clustering continues by assigning each point to a cluster.
- For each point, the algorithm measures the distance to each centroid.
- For each point, the smallest distance to a centroid indicates the assignment.

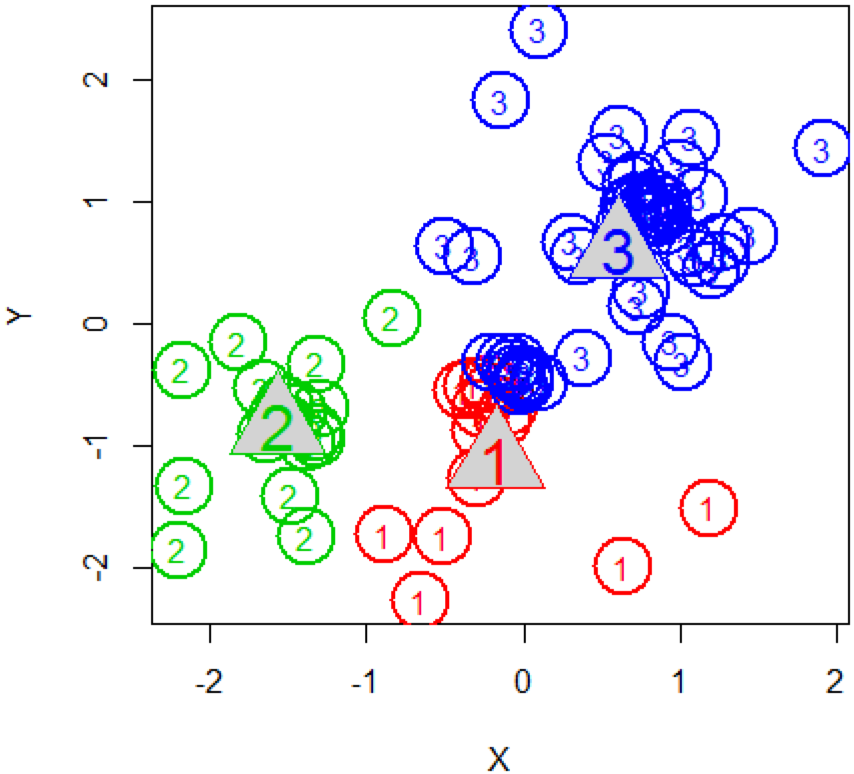
K-Means Clustering (5)



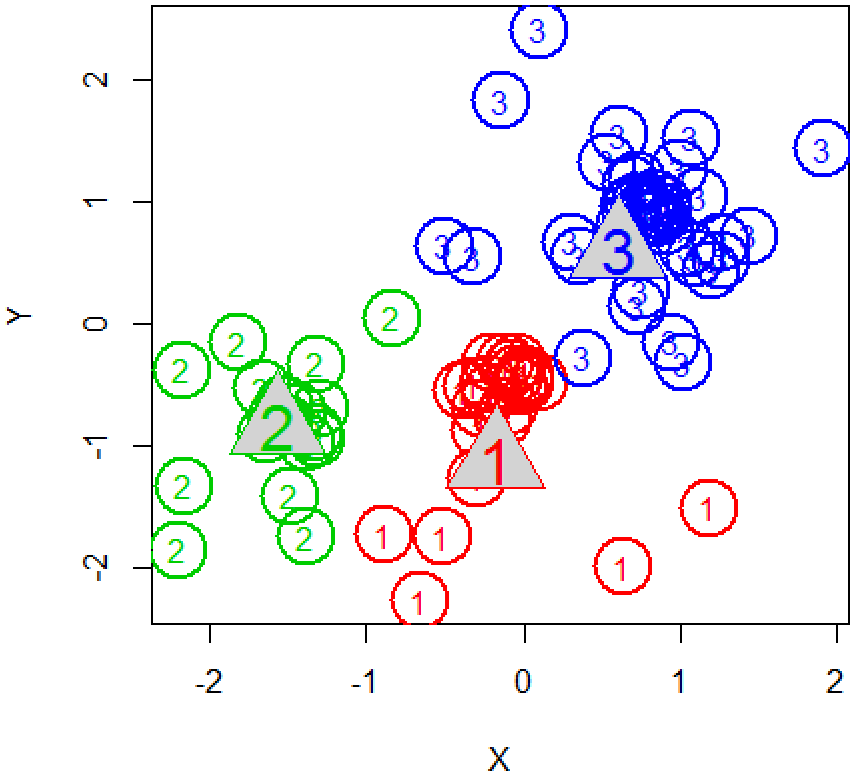
K-Means Clustering (6)



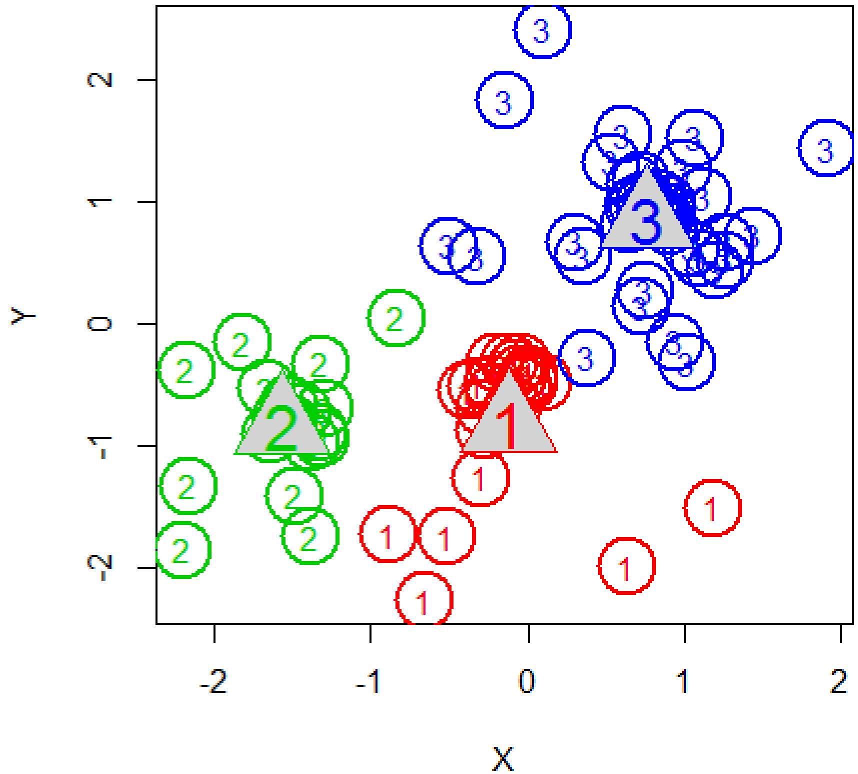
K-Means Clustering (7)



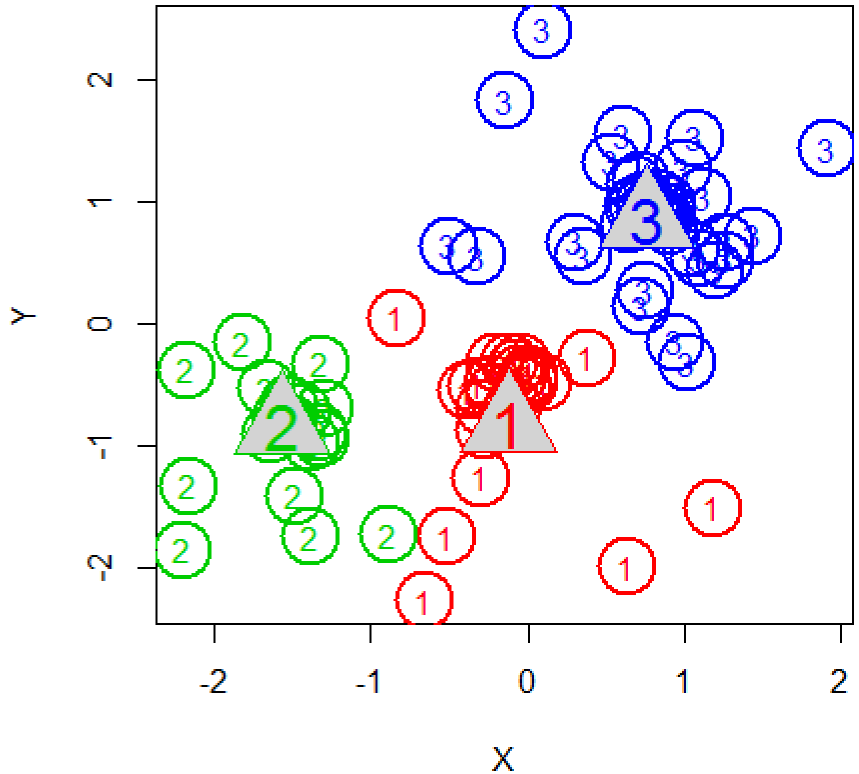
K-Means Clustering (8)



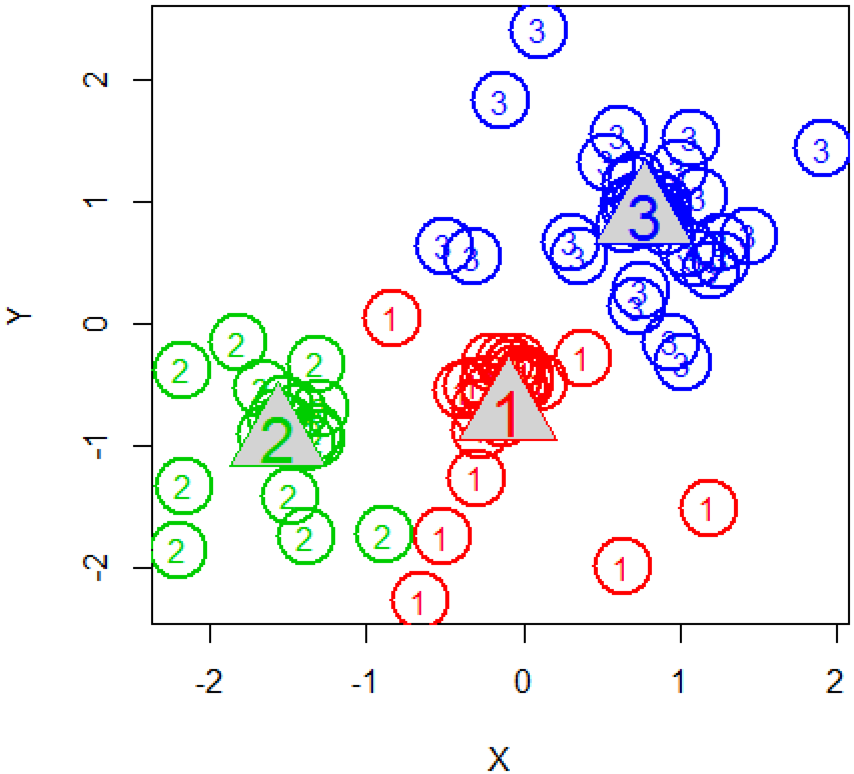
K-Means Clustering (9)



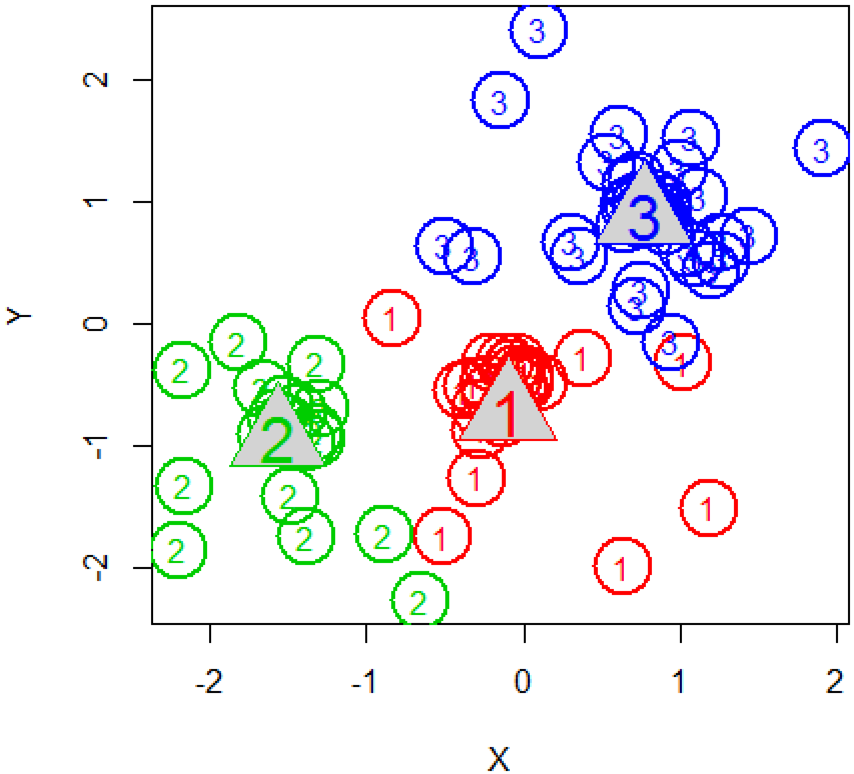
K-Means Clustering (10)



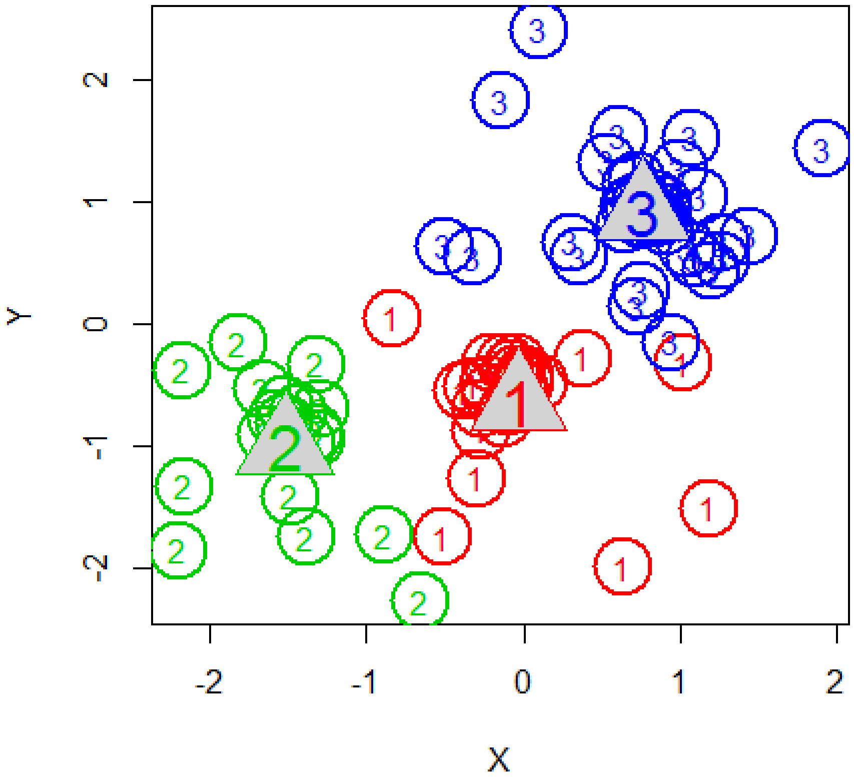
K-Means Clustering (11)



K-Means Clustering (12)



K-Means Clustering (13)



K-means Demo

- KMeansDemo

K-means

- Lessons learned:
  - Normalizations are important to put data on equal terms
  - Initial centroid number and placement is an art.
  - Categorical Data must be binarized
  - K-means is unsupervised because we do not tell the algorithm what outcome was observed or what outcome is desired.

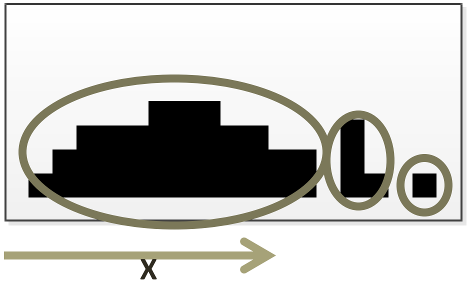
Introduction to K-means Clustering

Dimensions in Clustering

Clustering: Dimensions (1)

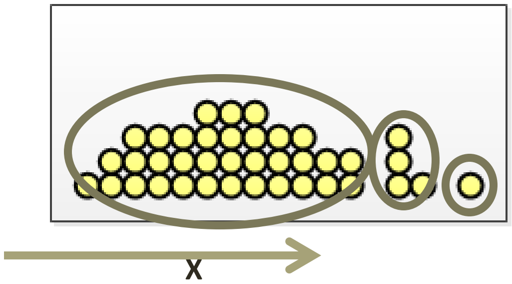
Where are the three clusters?

Clustering: Dimensions (2)



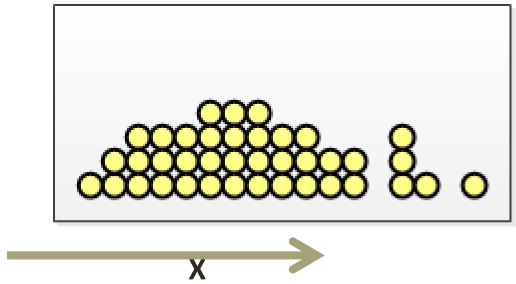
Simple assignment based on a 1D distribution

Clustering: Dimensions (3)



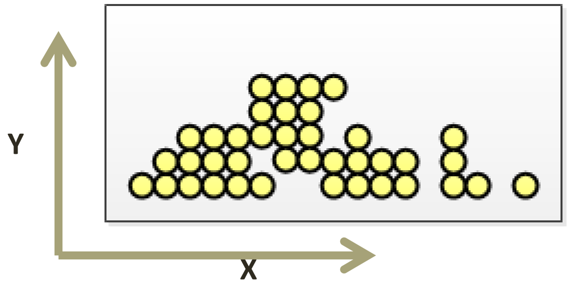
Simple assignment based on a 1D distribution

Clustering: Dimensions (4)



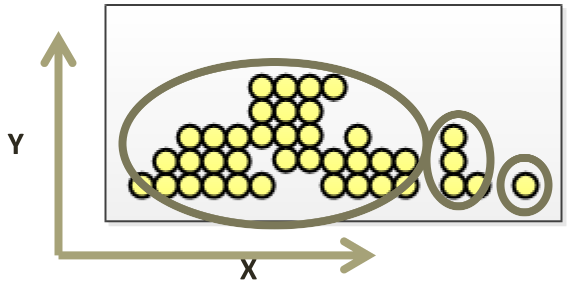
What if this was not a 1D distribution?

Clustering: Dimensions (5)



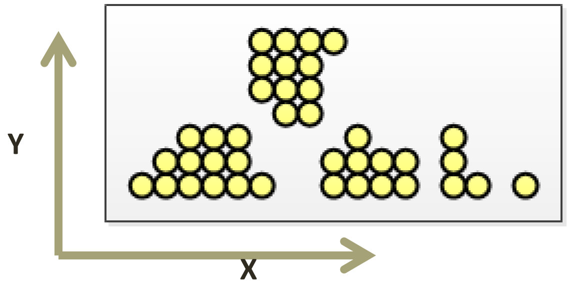
The distribution is in 2D. Some points differ in the 2^{nd} D

Clustering: Dimensions (6)



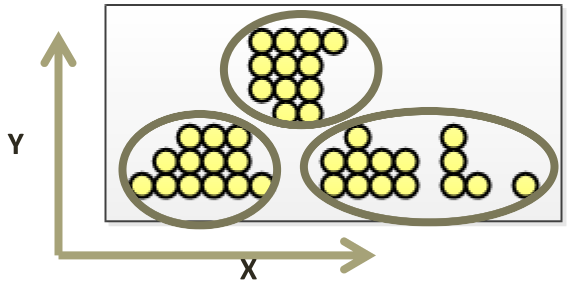
If the difference is minor, we still get the same clusters

Clustering: Dimensions (7)



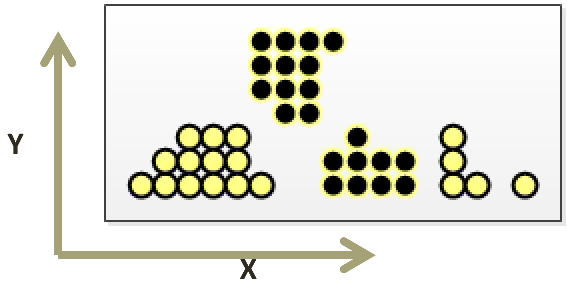
The difference could be significant

Clustering: Dimensions (8)



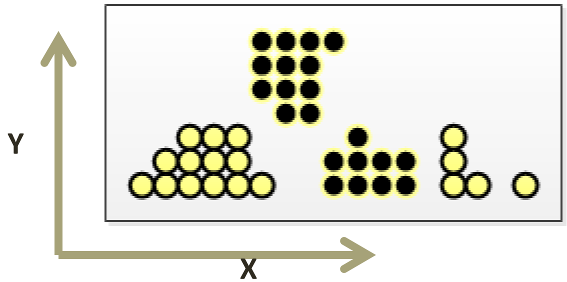
A big difference in the 2nd D can lead to different clusters

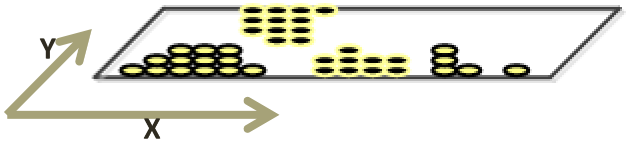
Clustering: Dimensions (9)



We can introduce another D by color coding. This is a Boolean Dimension

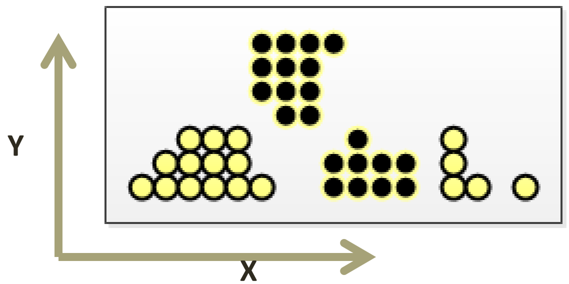
Clustering: Dimensions (10)

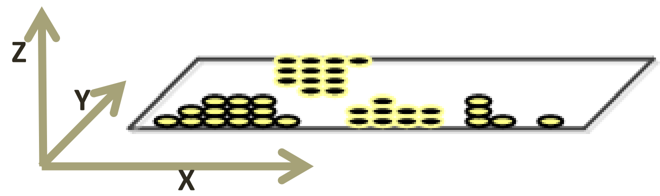




Create a 3rd Dimansion

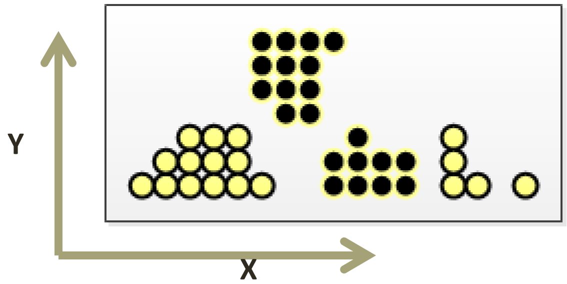
Clustering: Dimensions (11)

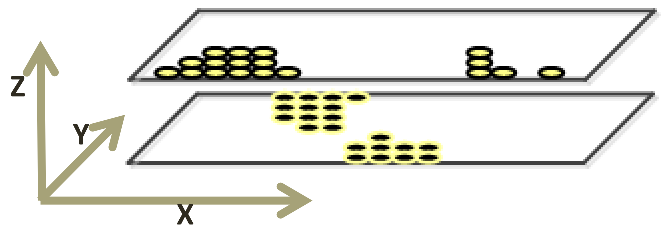




Create a 3rd Dimansion

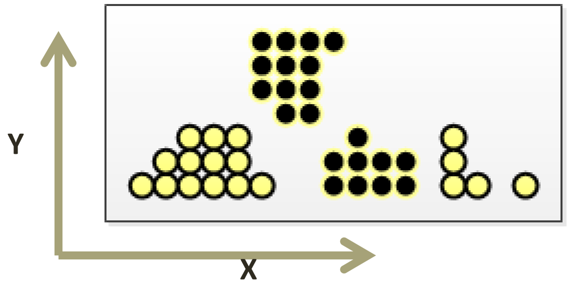
Clustering: Dimensions (12)

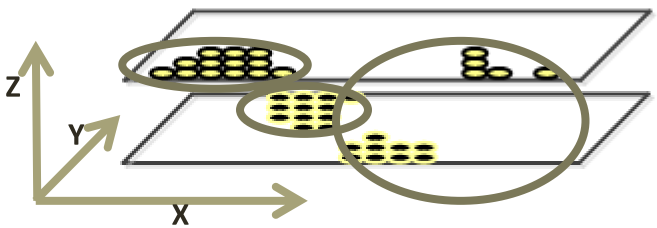


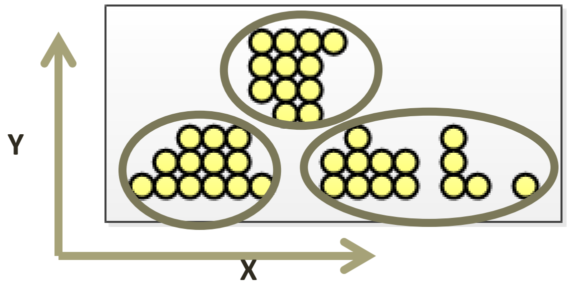


Where are the 3 clusters now?

Clustering: Dimensions (13)

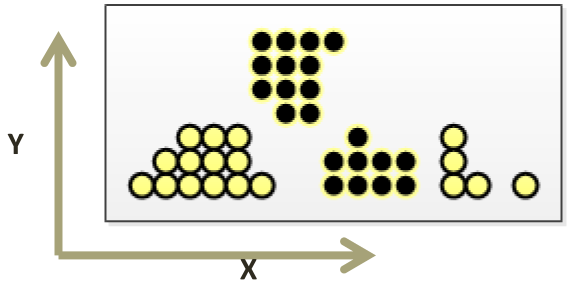


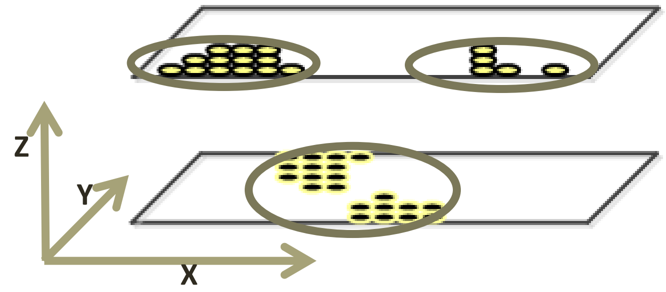


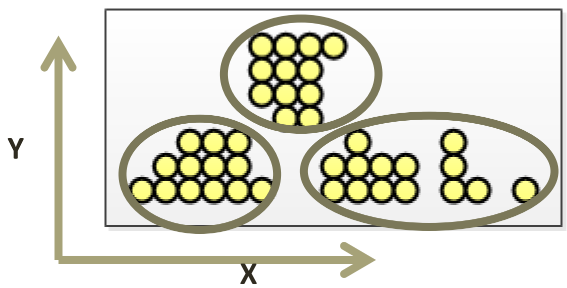


If the 3rd is small, then the clustering is the same as in 2D

Clustering: Dimensions (14)







If the 3rd is big, then the clustering differs from 2D

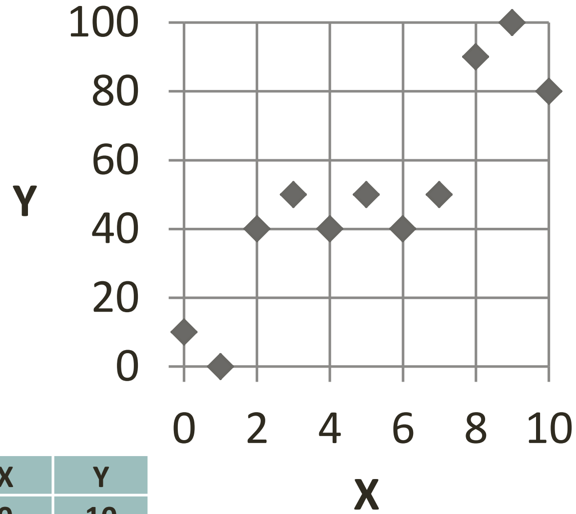
Dimensions in Clustering

Normalization in Clustering

Normalization of a linear relationship (1)

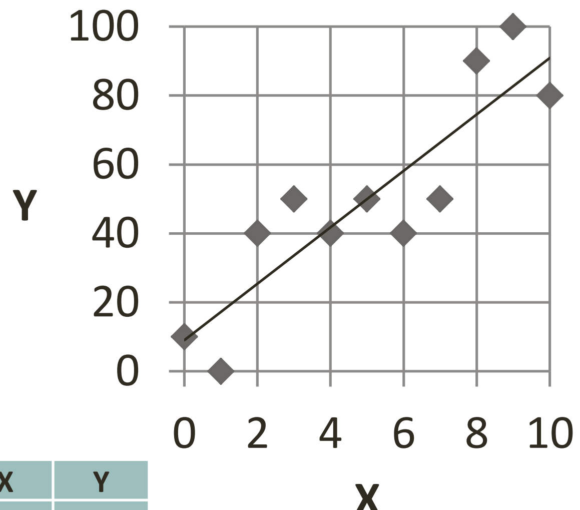
X	Y
0	10
1	0
2	40
3	50
4	40
5	50
6	40
7	50
8	90
9	100
10	80

Normalization of a linear relationship (2)



X	Y
0	10
1	0
2	40
3	50
4	40
5	50
6	40
7	50
8	90
9	100
10	80

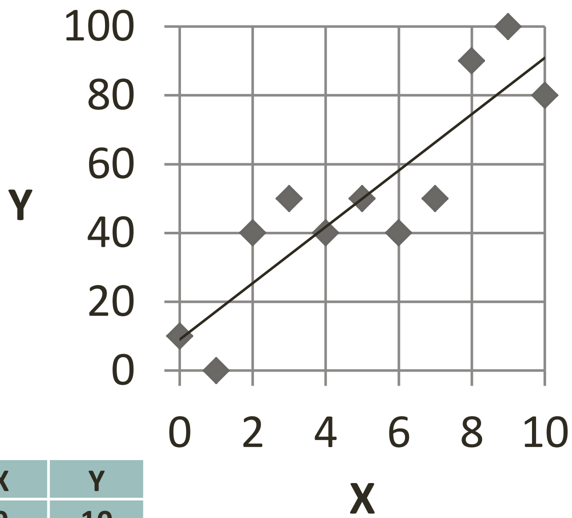
Normalization of a linear relationship (3)



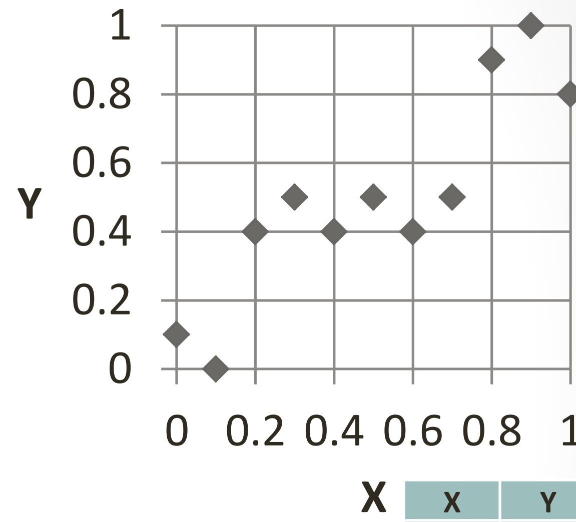
X	Y
0	10
1	0
2	40
3	50
4	40
5	50
6	40
7	50
8	90
9	100
10	80

Y = 10 + 8*X

Normalization of a linear relationship (4)



Normalize

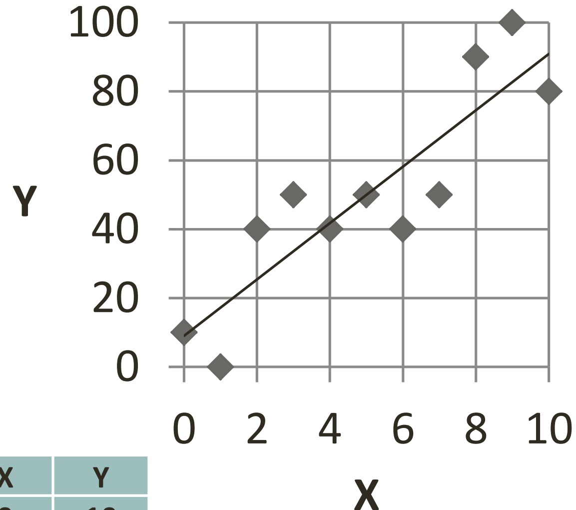


Y = 10 + 8*X

X	Y
0	10
1	0
2	40
3	50
4	40
5	50
6	40
7	50
8	90
9	100
10	80

X	Y
0	0.1
0.1	0
0.2	0.4
0.3	0.5
0.4	0.4
0.5	0.5
0.6	0.4
0.7	0.5
0.8	0.9
0.9	1
1	0.8

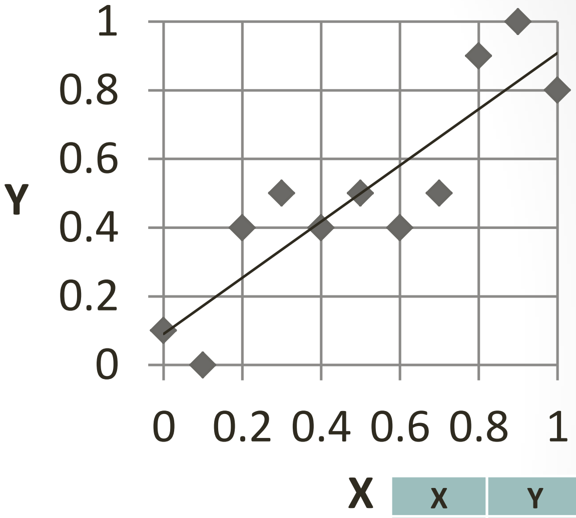
Normalization of a linear relationship (5)



X	Y
0	10
1	0
2	40
3	50
4	40
5	50
6	40
7	50
8	90
9	100
10	80

Y = 10 + 8*X

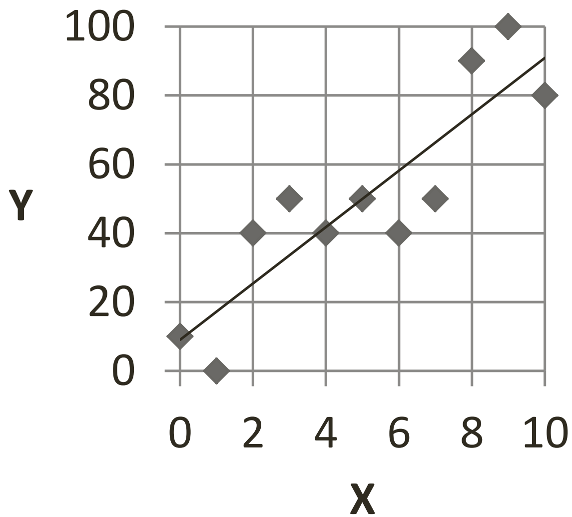
Normalize



Y = 0.1 + 0.8*X

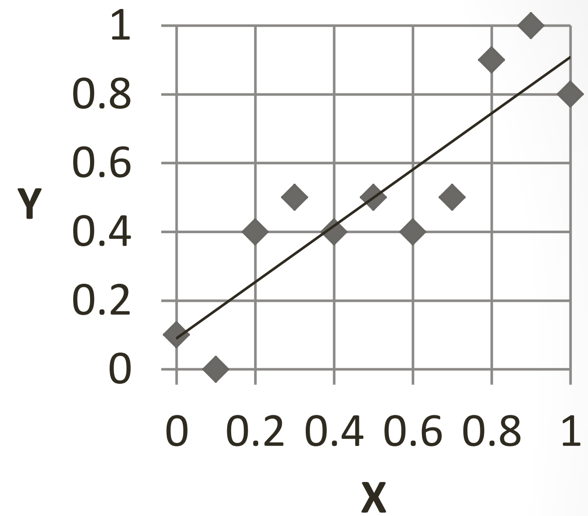
X	Y
0	0.1
0.1	0
0.2	0.4
0.3	0.5
0.4	0.4
0.5	0.5
0.6	0.4
0.7	0.5
0.8	0.9
0.9	1
1	0.8

Normalization of a linear relationship (6)



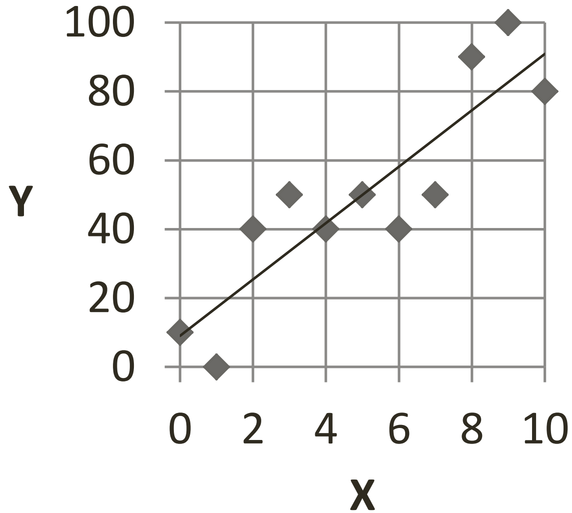
Y = 10 + 8*X

Normalize



Y = 0.1 + 0.8*X

Normalization of a linear relationship (7)



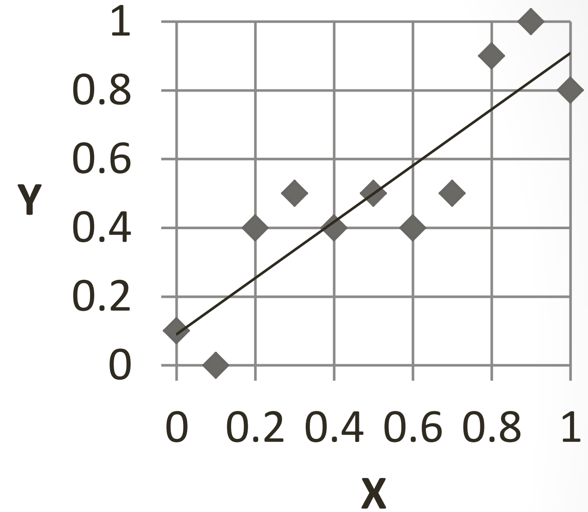
Y = 10 + 8*X

Normalize

Normalize Input
X = 2 -> X' = 0.2

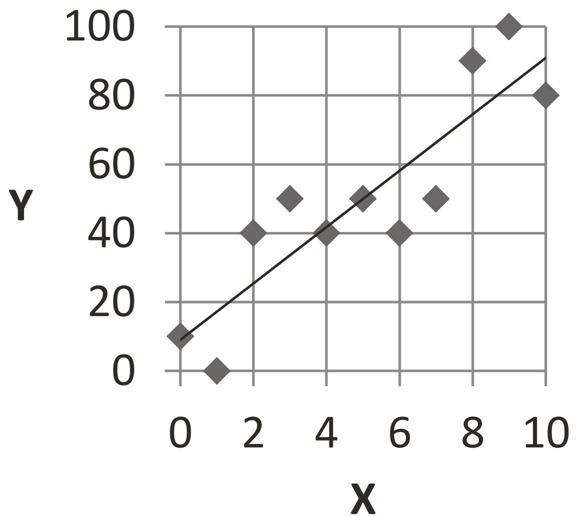
Predict Output
X' = 0.2 -> Y'= 0.26

Denormalize Output
Y'= 0.26 -> Y = 26



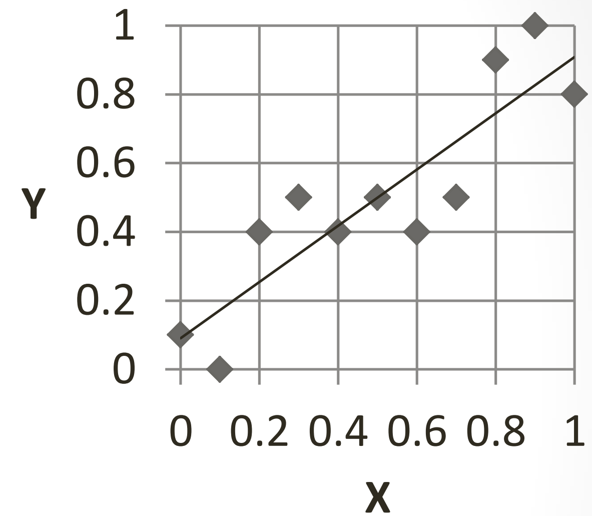
Y = 0.1 + 0.8*X

Normalization of a linear relationship (8)



Y = 10 + 8*X

Normalize



Y = 0.1 + 0.8*X

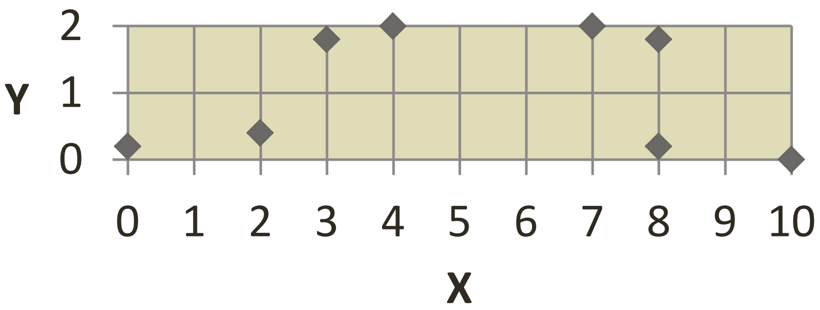
Normalize Input
X = 2 -> X’ = 0.2

Predict Output
X’ = 0.2 -> Y’= 0.26

Denormalize Output
Y’= 0.26 -> Y = 26

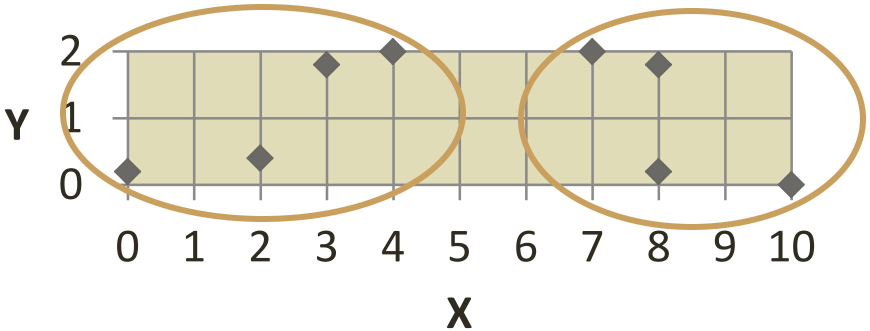
Prediction in Original Space:
X = 2 -> Y = 26

Normalization of a non-linear relationship (1)



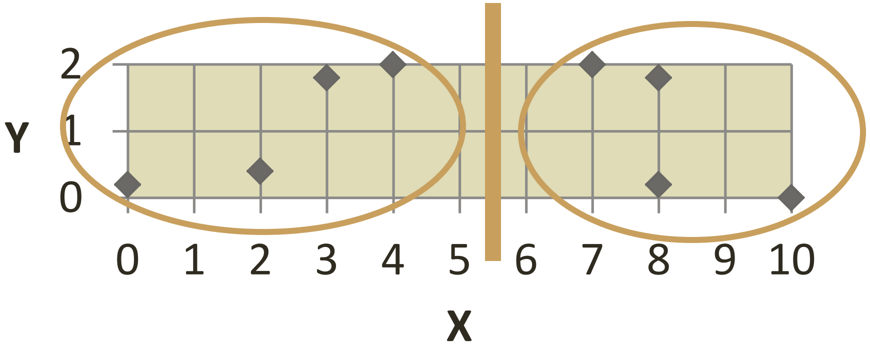
Original data in 2D:
Find 2 clusters

Normalization of a non-linear relationship (2)



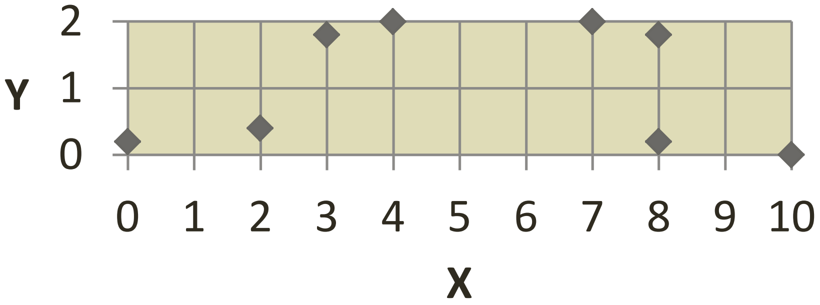
Found 2 Clusters

Normalization of a non-linear relationship (3)



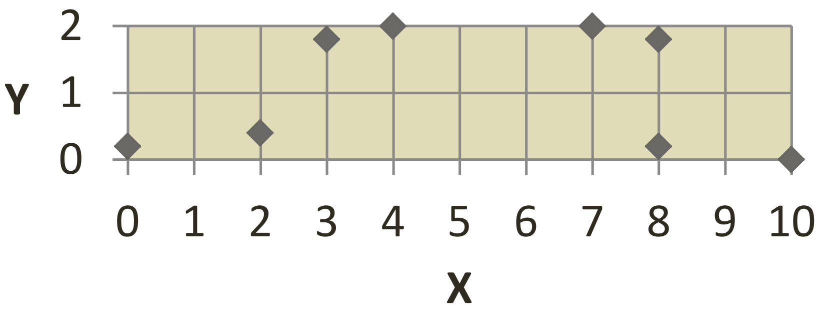
Clusters segment the image

Normalization of a non-linear relationship (4)

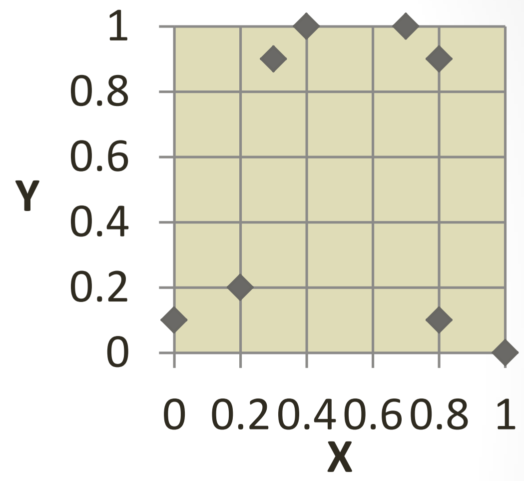


Non-normalized 2D data

Normalization of a non-linear relationship (5)



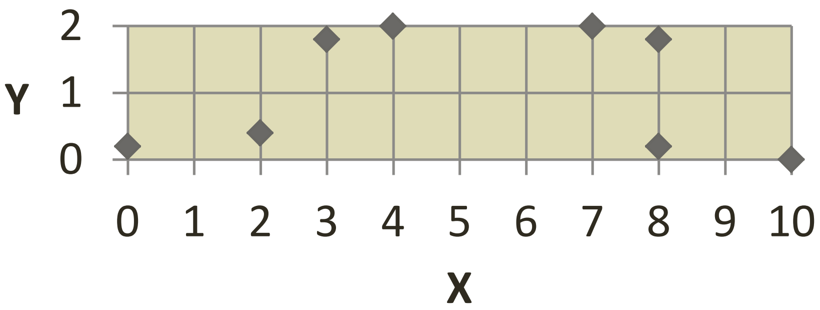
Normalize



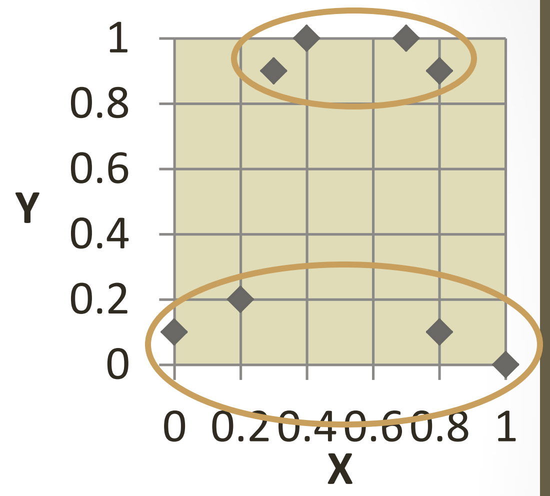
Non-normalized 2D data

Normalize the data:
Search for 2 Clusters

Normalization of a non-linear relationship (6)



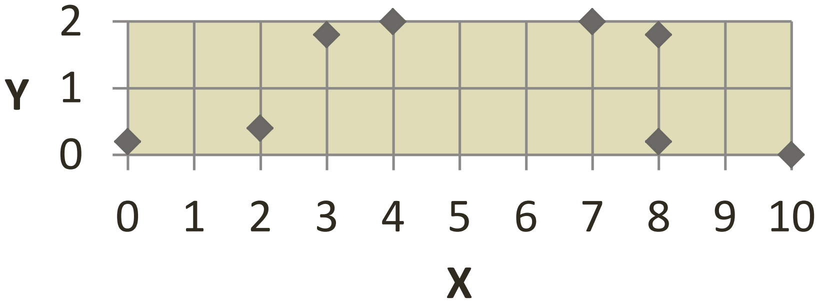
Normalize



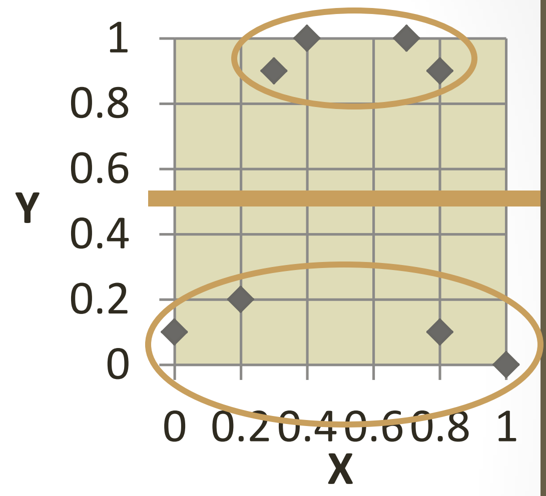
Non-normalized 2D data

Found 2 Clusters in the normalized data

Normalization of a non-linear relationship (6)



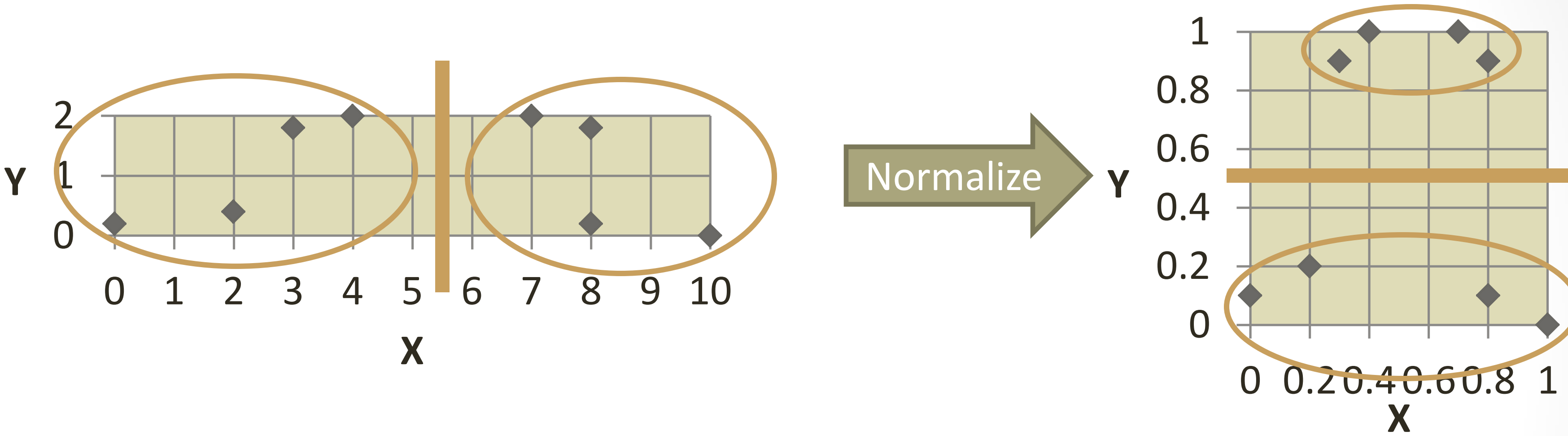
Normalize



Non-normalized 2D data

Clusters Segment the Image

Normalization of a non-linear relationship (7)



Clustering before normalization

Clustering after normalization

Normalization of Linear and Non-Linear Outcomes

- Non-linear (Normalization can change outcome):
- K-Means
- Neural Net

- Linear (Normalization should not change outcome):
- Logistic Regression
- Linear Regression
- Mixture of Gaussians

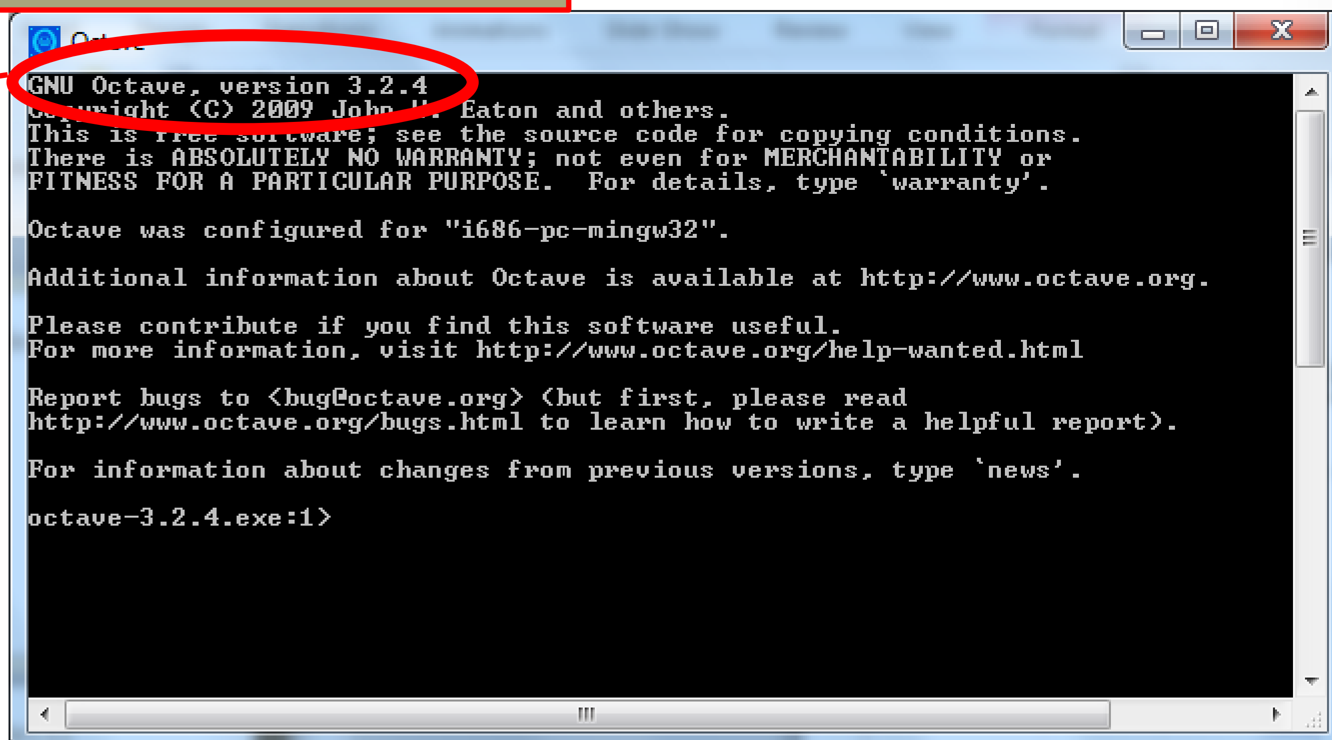
Normalization in Clustering

Introduction to Octave

Octave Console (0)

Start Octave

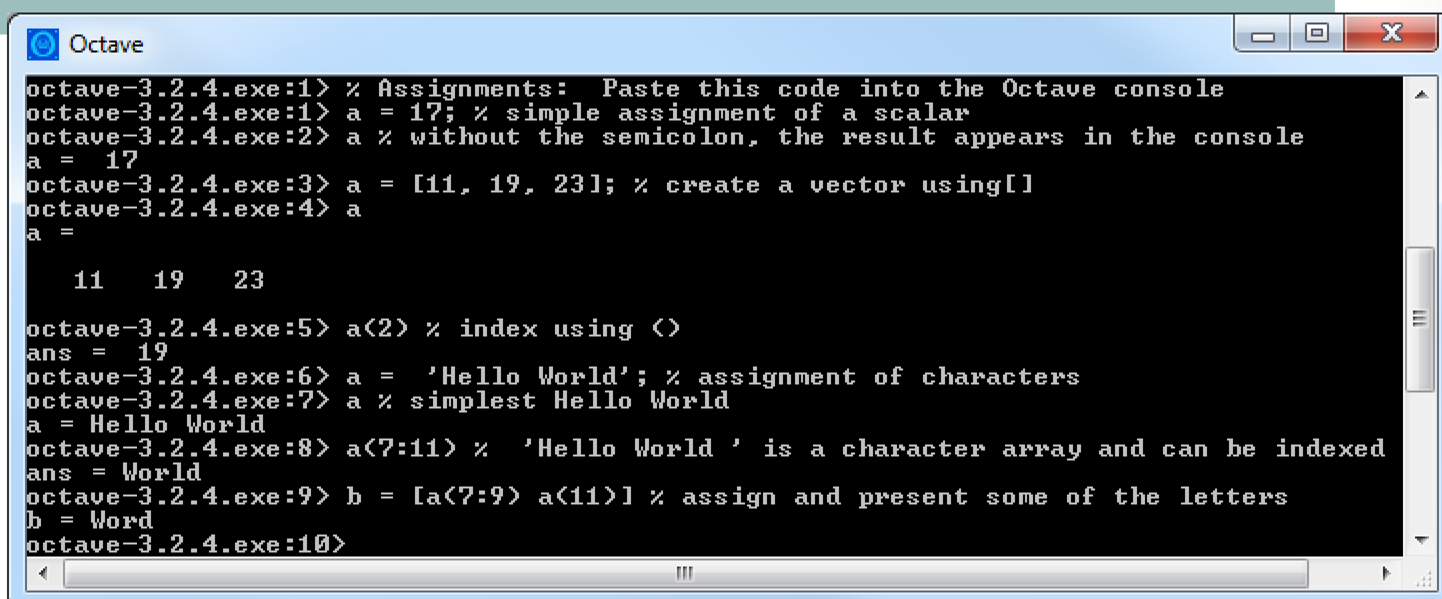
Octave console opens. Note version number



Octave Console (1)

% Assignments: Paste this code into the Octave console

```
a = 17; % simple assignment of a scalar
a % without the semicolon, the result appears in the console
a = [11, 19, 23]; % create a vector using[]
a
a(2) % index using ()
a =  'Hello World'; % assignment of characters
a % simplest Hello World
a(7:11) %  'Hello World ' is a character array and can be indexed
b = [a(7:9) a(11)] % assign and present some of the letters
```



```
octave-3.2.4.exe:1> % Assignments:  Paste this code into the Octave console
octave-3.2.4.exe:1> a = 17; % simple assignment of a scalar
octave-3.2.4.exe:2> a % without the semicolon, the result appears in the console
a =  17
octave-3.2.4.exe:3> a = [11, 19, 23]; % create a vector using[]
octave-3.2.4.exe:4> a
a =

   11   19   23

octave-3.2.4.exe:5> a(2) % index using ()
ans =  19
octave-3.2.4.exe:6> a =  'Hello World'; % assignment of characters
octave-3.2.4.exe:7> a % simplest Hello World
a = Hello World
octave-3.2.4.exe:8> a(7:11) %  'Hello World ' is a character array and can be indexed
ans = World
octave-3.2.4.exe:9> b = [a(7:9) a(11)] % assign and present some of the letters
b = Word
octave-3.2.4.exe:10>
```

Octave Console (2)

% Basic variables are matrices.   Paste these lines into Octave

a = 19;

a

size(a) % the result is the size in two dimensions

a(3,2) = -7; % Assign a value in a new dimension

a % Present the matrix

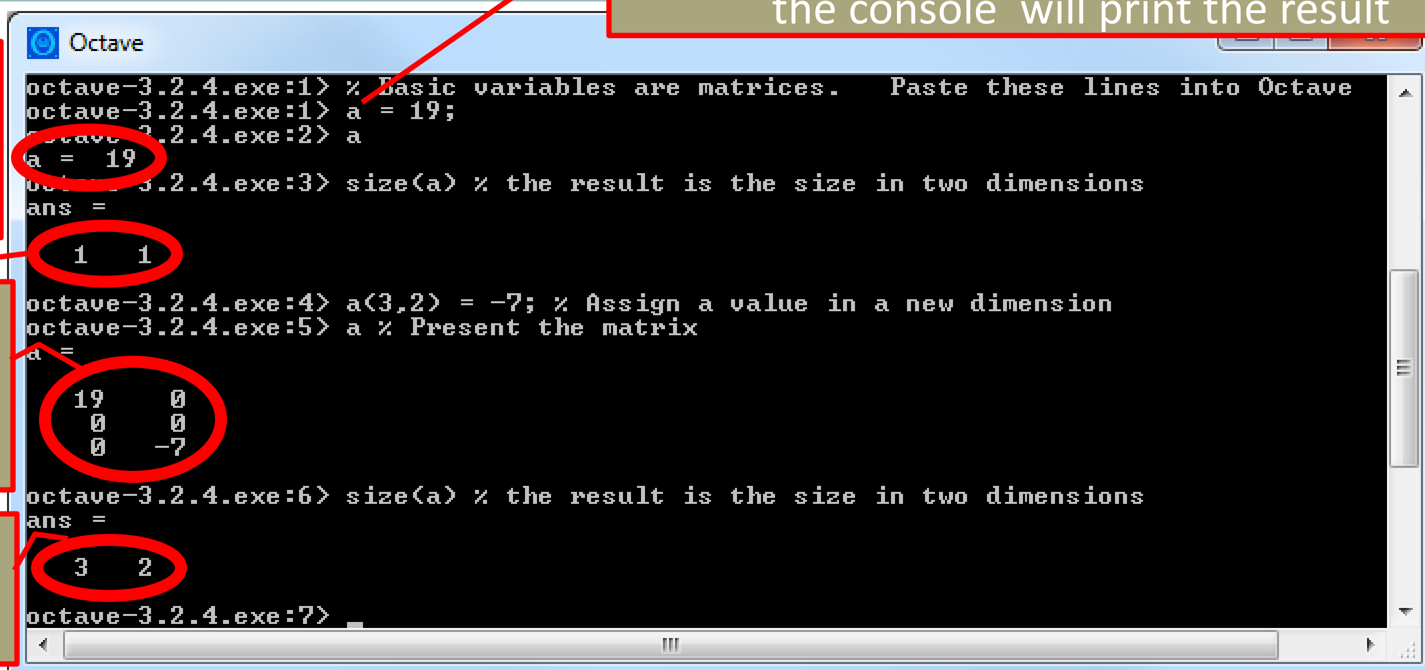
size(a) % the result is the size in two dimen

If the statement has no semi colon (;), then the console will print the result

a = 19
Results in a 1 X 1 matrix

Extending the matrix fills in zeros

a is now a
3 X 2 matrix



Octave Console (3)

% Distinguish between a row vector and a column vector.

a = [11, 19, 23];

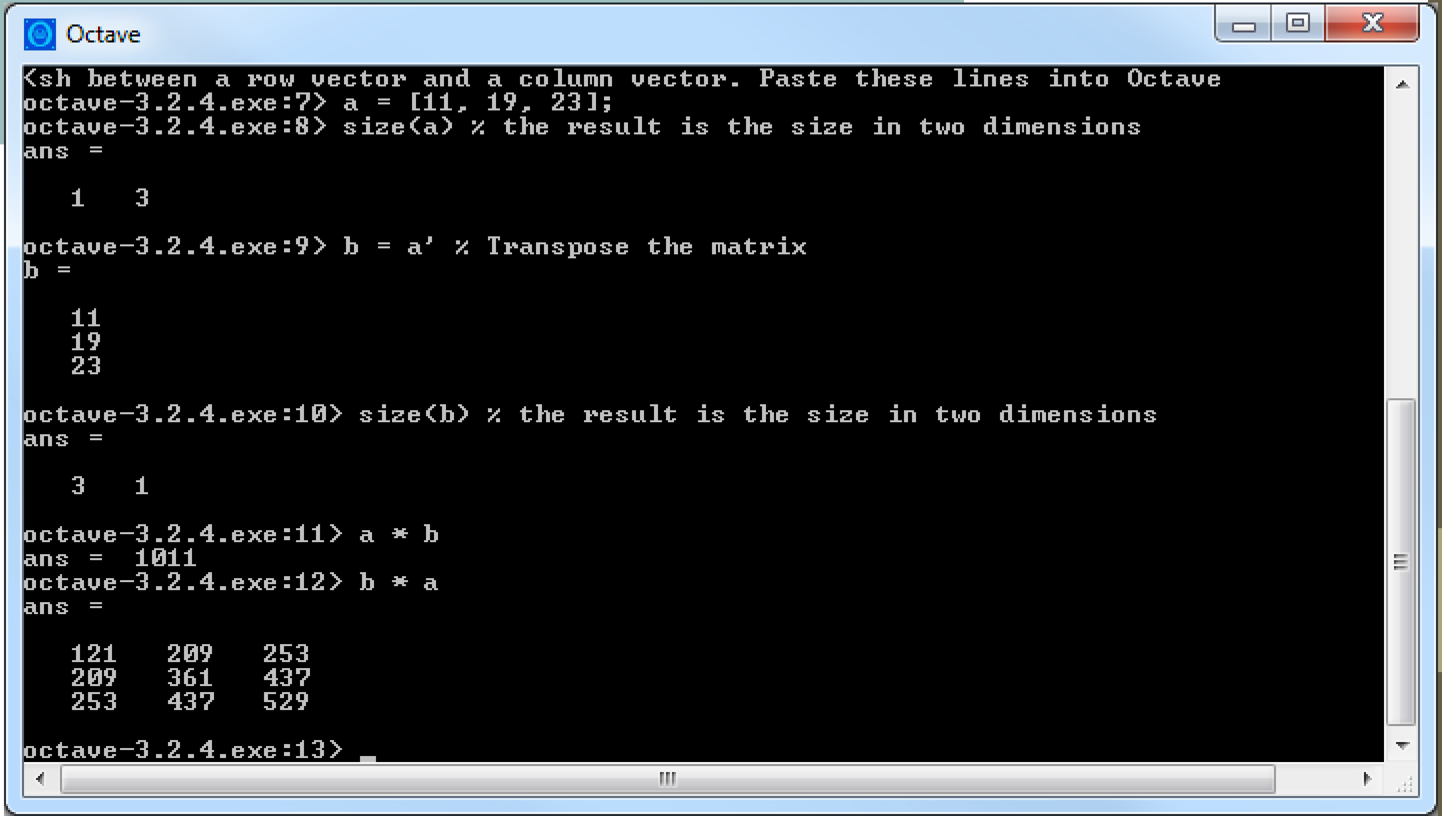
size(a) % the result is the size in two dimensions

b = a' % Transpose the matrix

size(b) % the result is the size in two dimensions

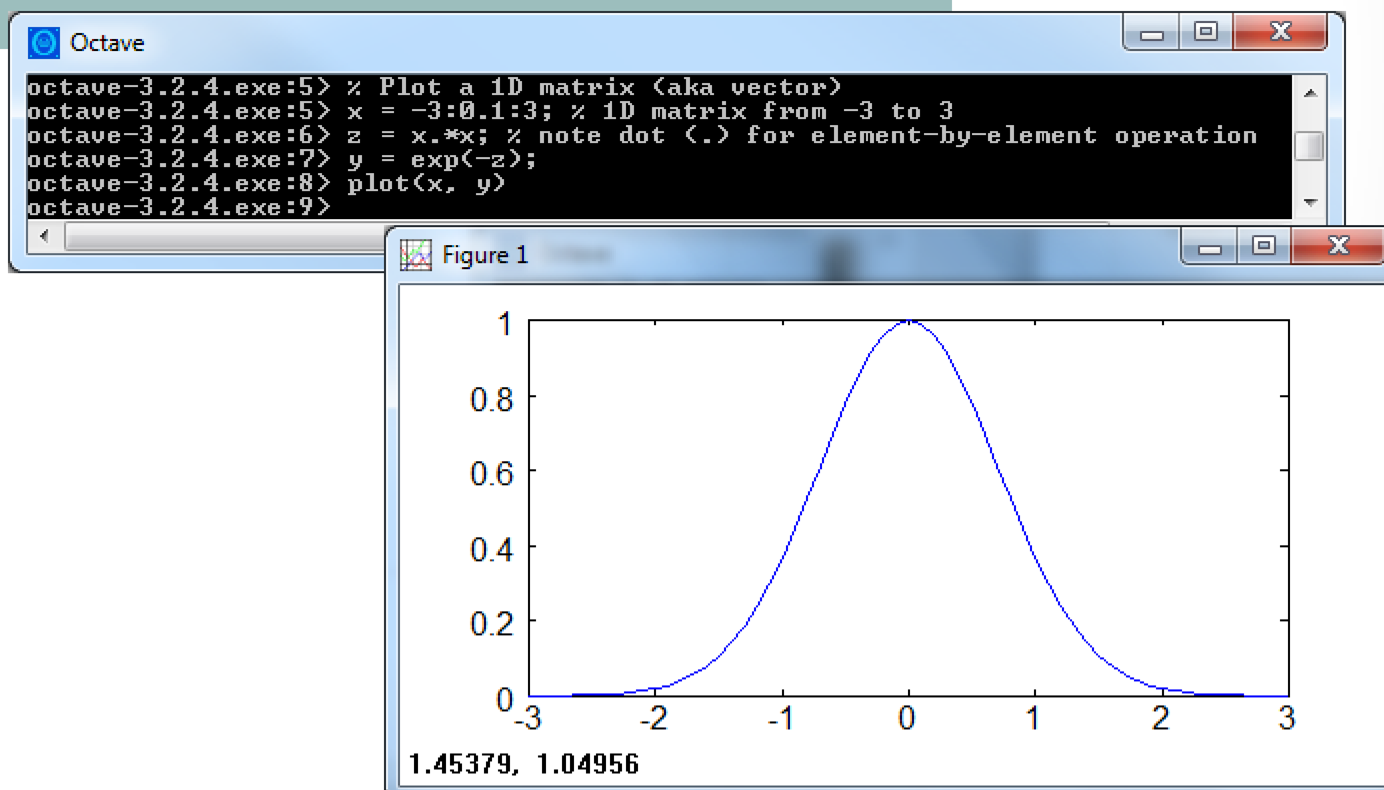
a * b

b * a



Octave Console (4)

```
% Plot a 1D matrix (aka vector)
    x = -3:0.1:3; % 1D matrix from -3 to 3
    z = x.*x; % note dot (.) for element-by-element operation
    y = exp(-z);
    plot(x, y)
```



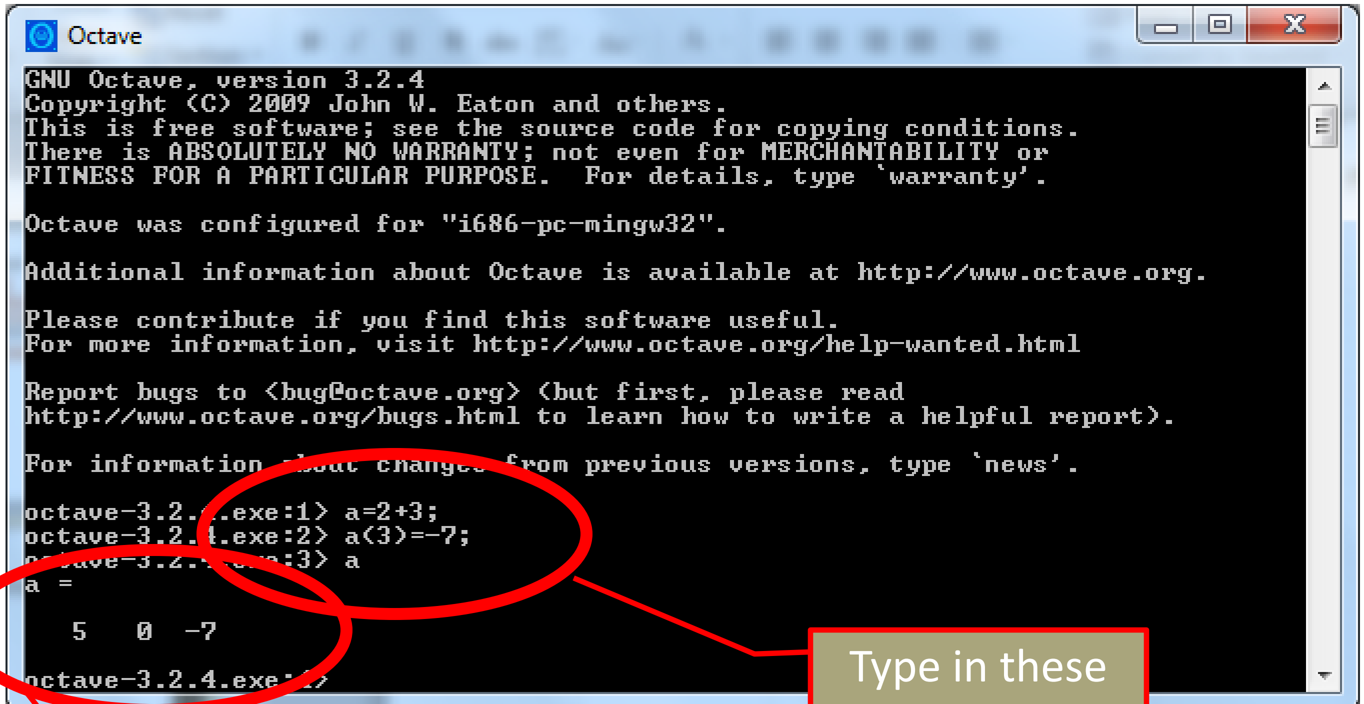
Binary operators in Octave

- Some common operator symbols are for use in matrix algebra and these operators do not do what many people expect.
  - ‘*’ matrix multiplication; Use “.*” for element-by-element multiplication
  - ‘^’ matrix power; Use”.^” for element-by-element power
  - ‘/’ matrix right division; Use “./” for element-by-element division

Octave m-files (0)

- How to create and use m-files in Octave

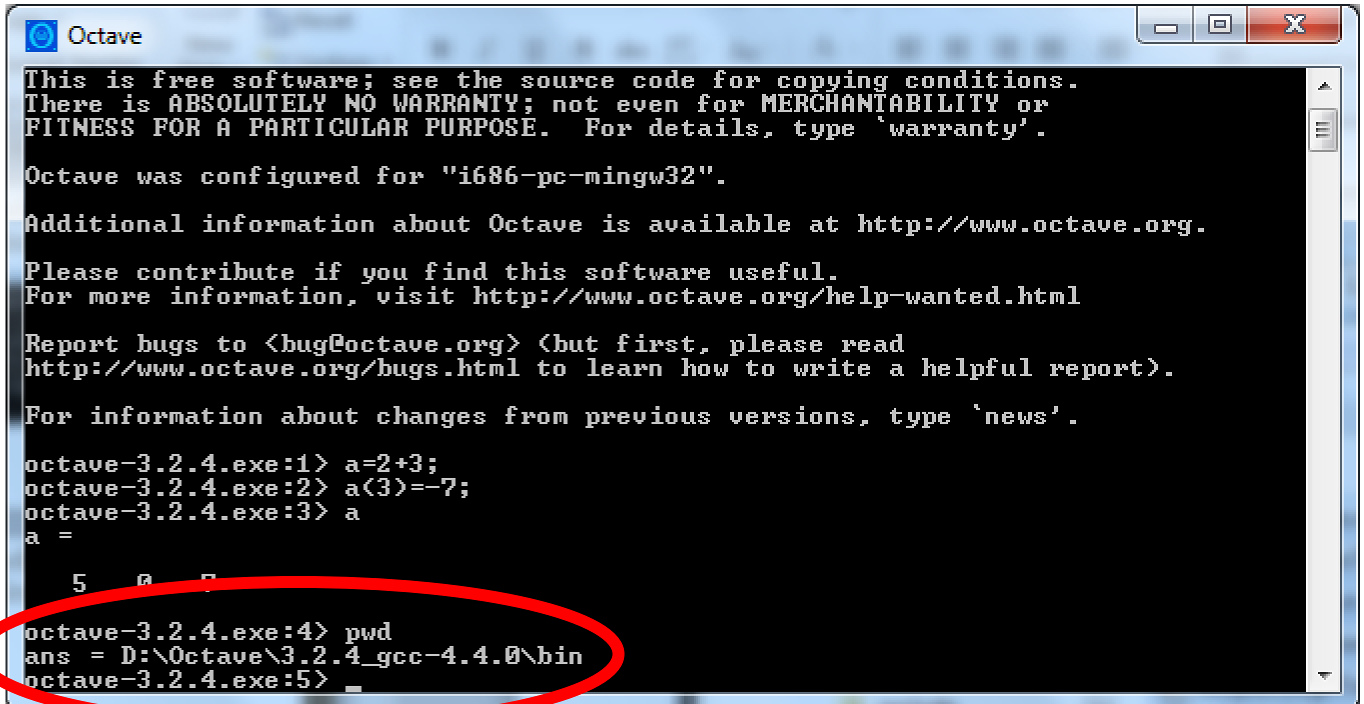
Octave m-files (1): Start Octave and Test it



Type in these commands

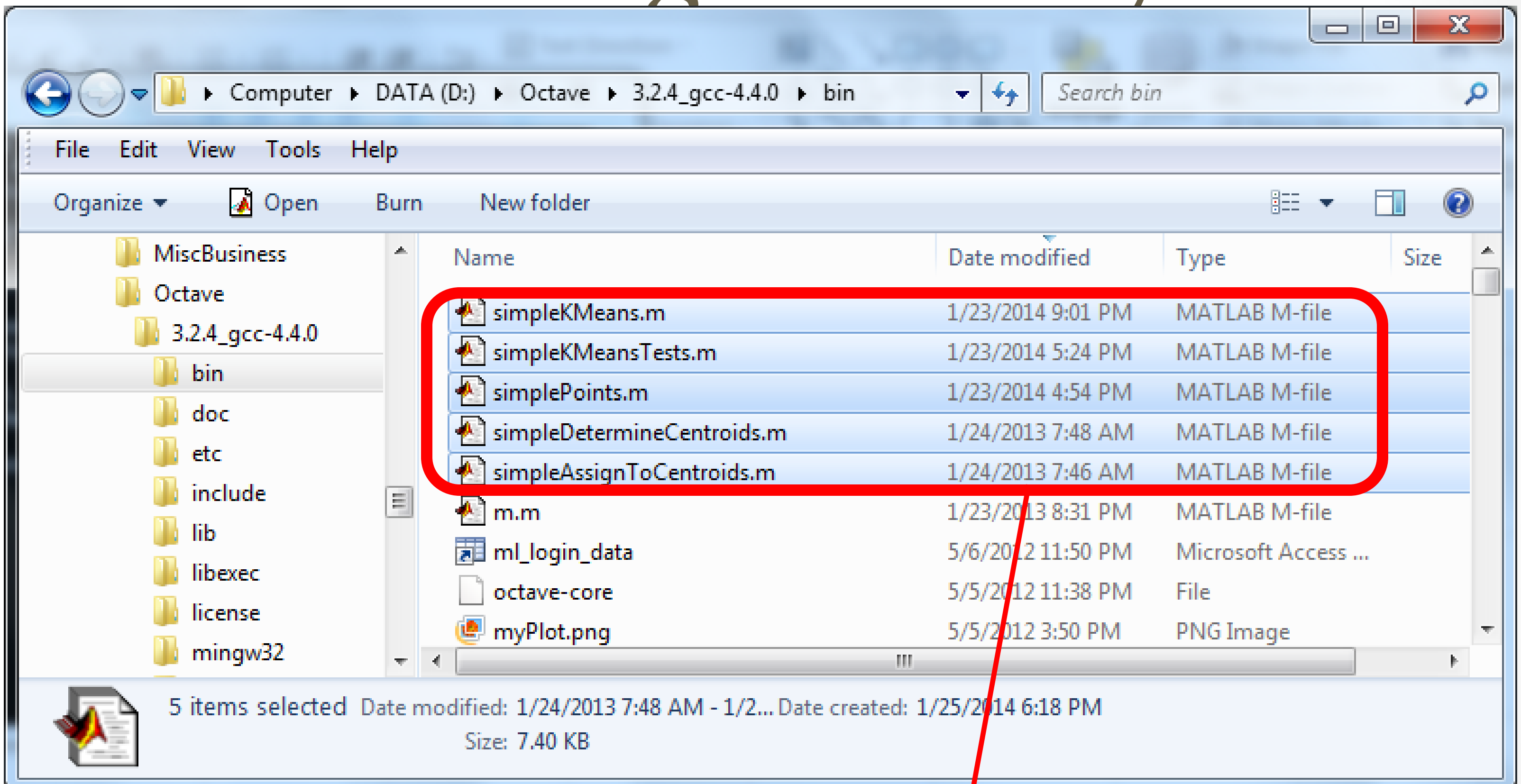
Understand why you get this response

Octave m-files (2): Determine working directory



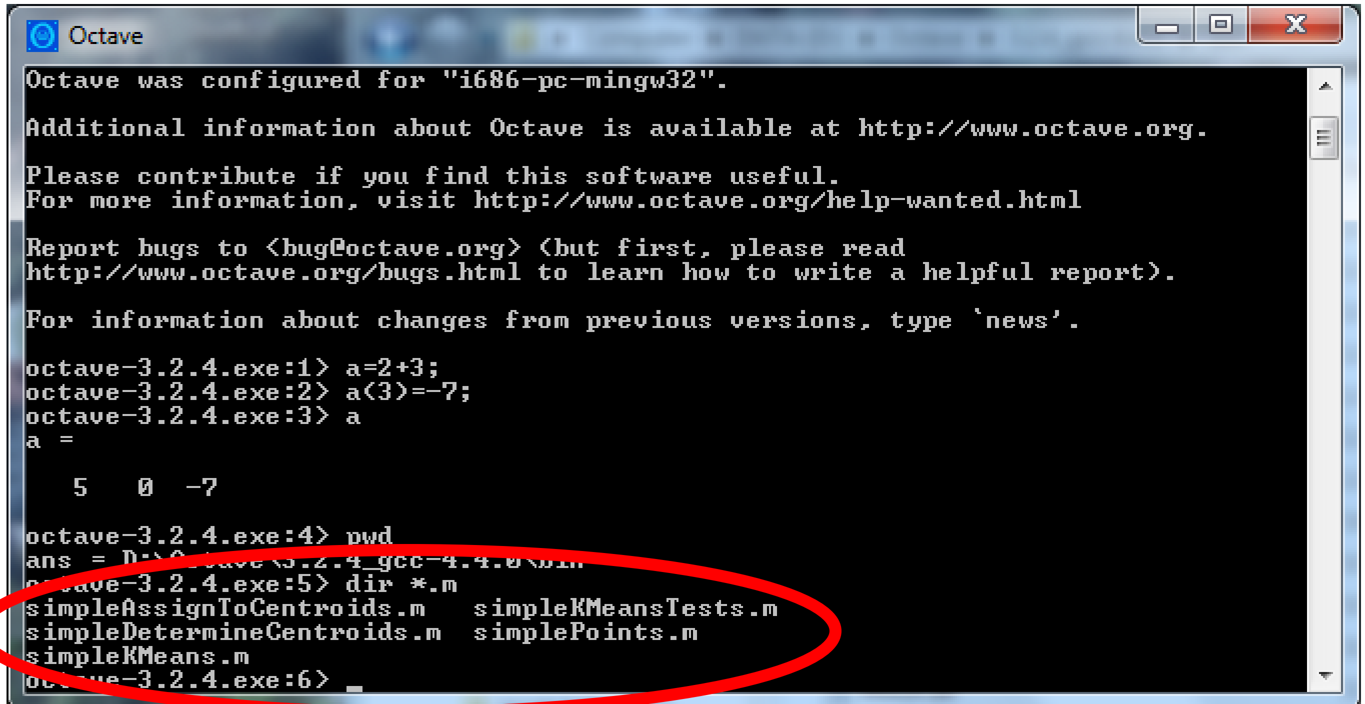
Use the pwd command to determine your working directory! Here, my working directory is: ***D:\Octave\3.2.4_gcc-4.4.0\bin***
Your working directory may be different.

Octave m-files (3): Place m-files in Working Directory



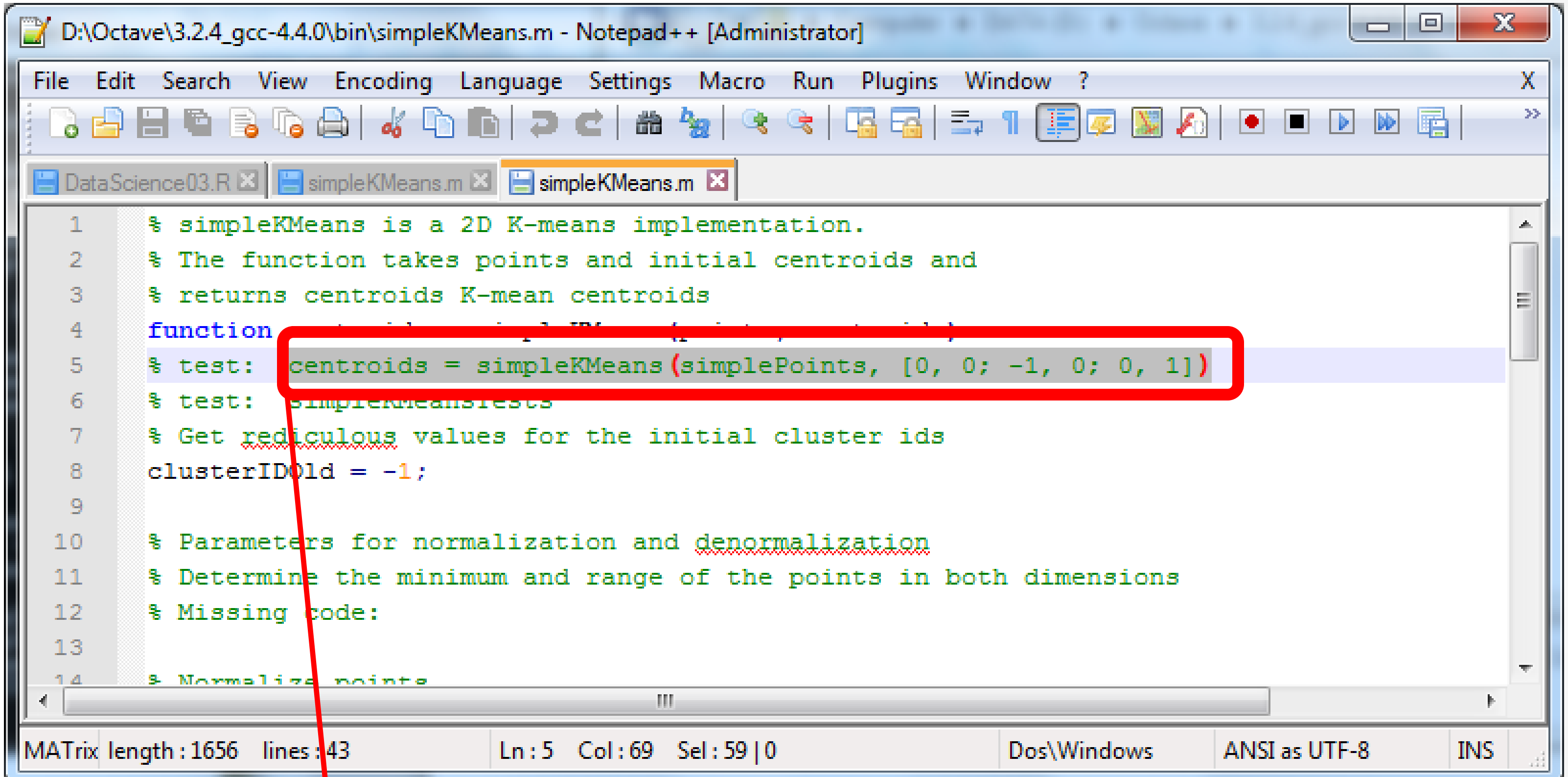
Take these 5 m-files from catalyst and place them in your working directory.

Octave m-files (4): Verify m-files in Working Directory



Type in ***dir *.m*** into the Octave console. The m-files in your working directory will be listed. These m-files must include the m-files from catalyst.

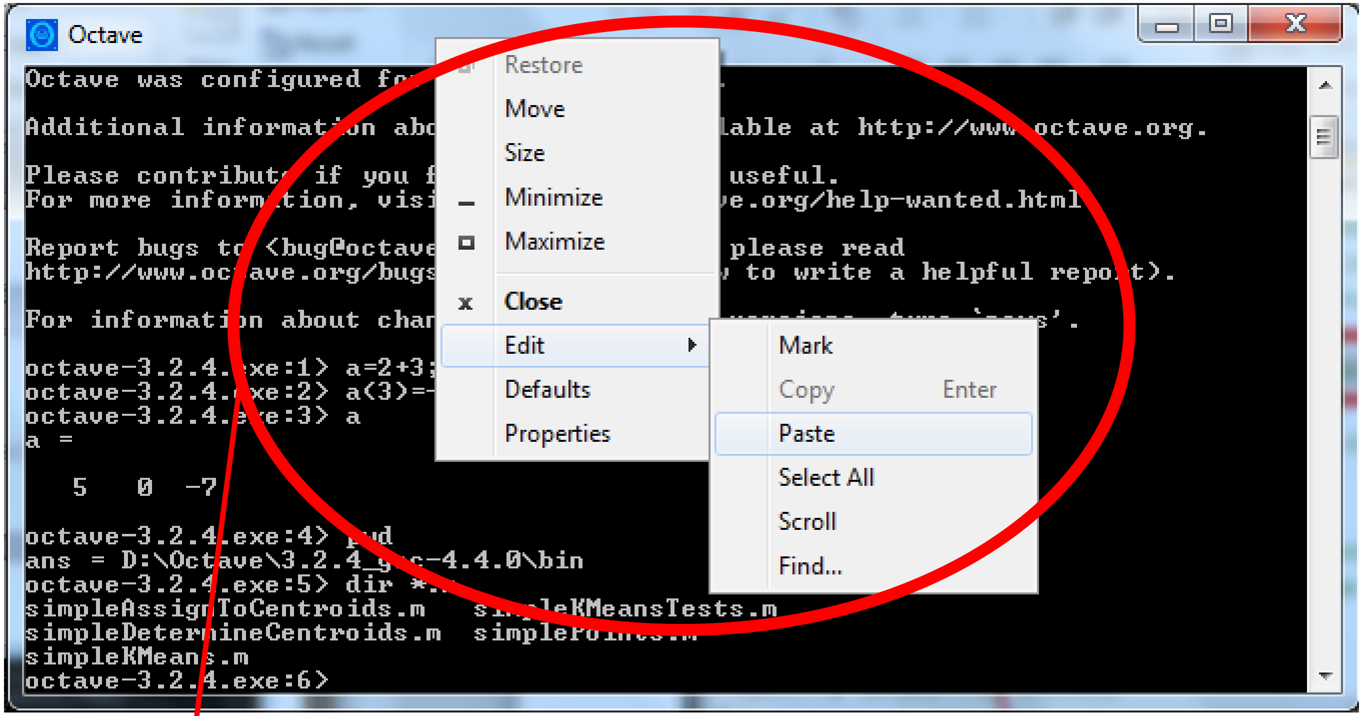
Octave m-files (5): Open m-file with editor



Use an editor, like Notepad++ to open simpleKMeans.m and then copy the following text:

centroids = simpleKMeans(simplePoints, [0, 0; -1, 0; 0, 1])

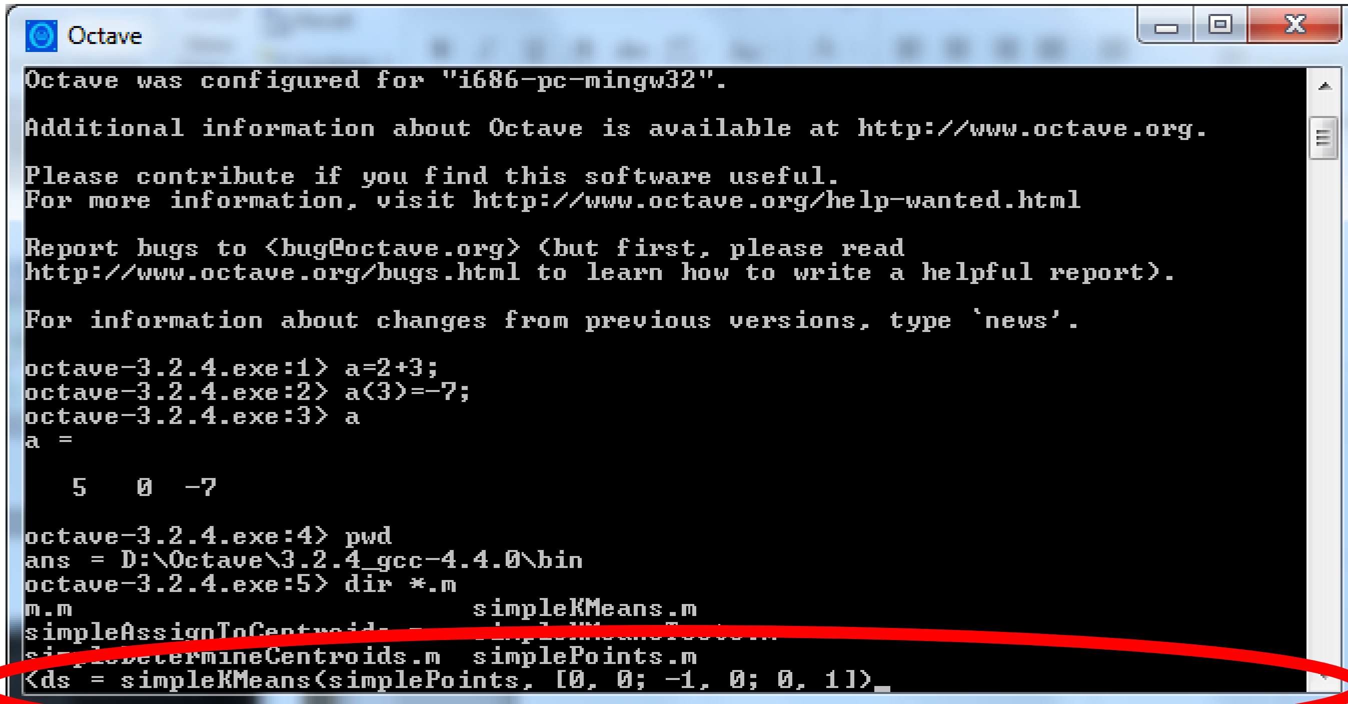
Octave m-files (6): Paste command into Octave console



Paste the text from the clipboard into the Octave console. Alternately, type in:

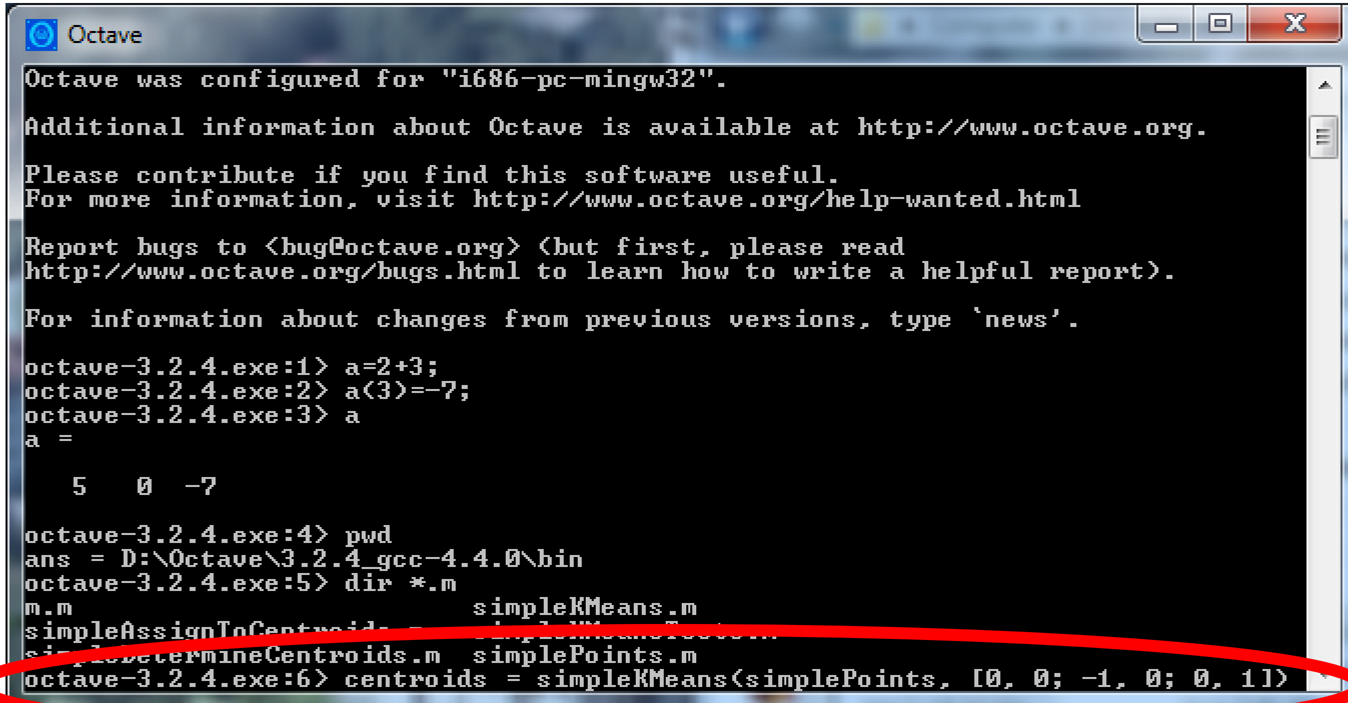
centroids = simpleKMeans(simplePoints, [0, 0; -1, 0; 0, 1])

Octave m-files (7): Paste command into Octave console



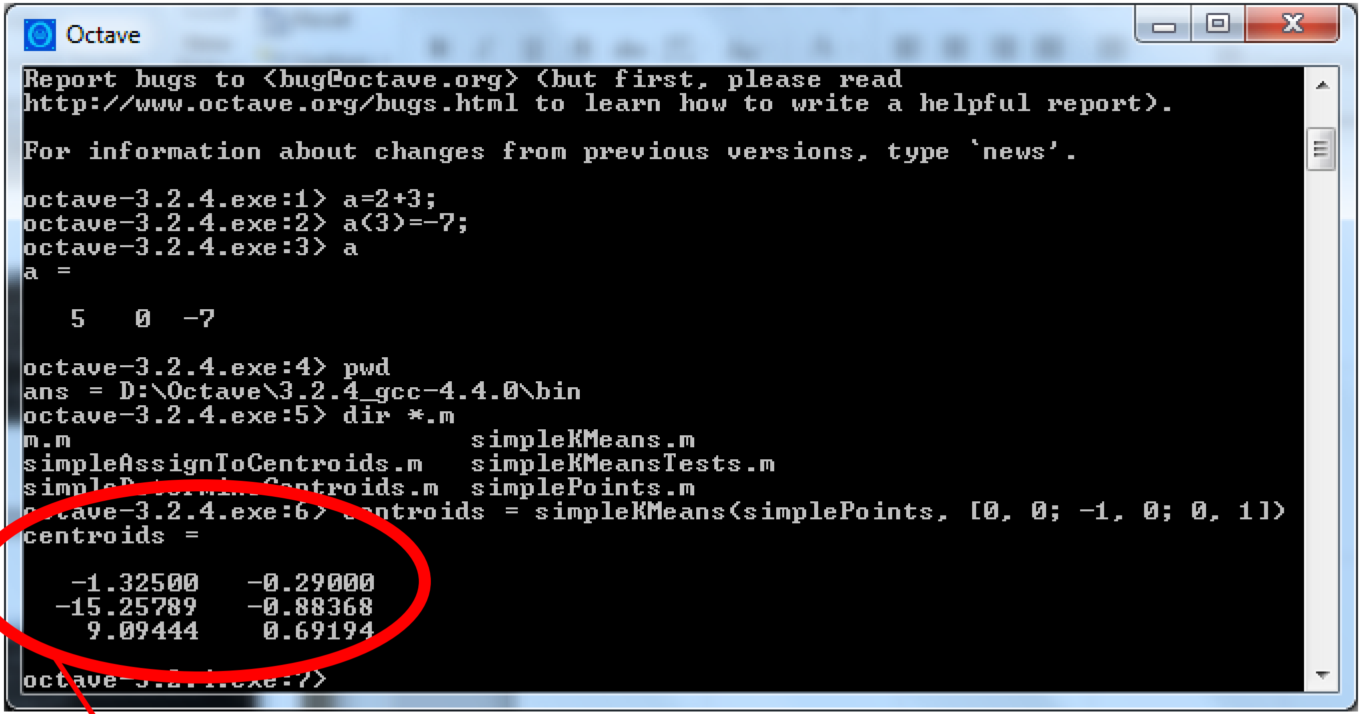
You should see the text appear in the Octave console. If the text is too long for the console, you might not see the first part of the text. In that case, you can see the first part of the pasted text by using the left-arrow on your keyboard.

Octave m-files (8): Paste command into Octave console



You should see the text appear in the Octave console. If the text is too long for the console, you might not see the first part of the text. In that case, you can see the first part of the pasted text by using the left-arrow on your keyboard.

Octave m-files (9): See results in Octave console

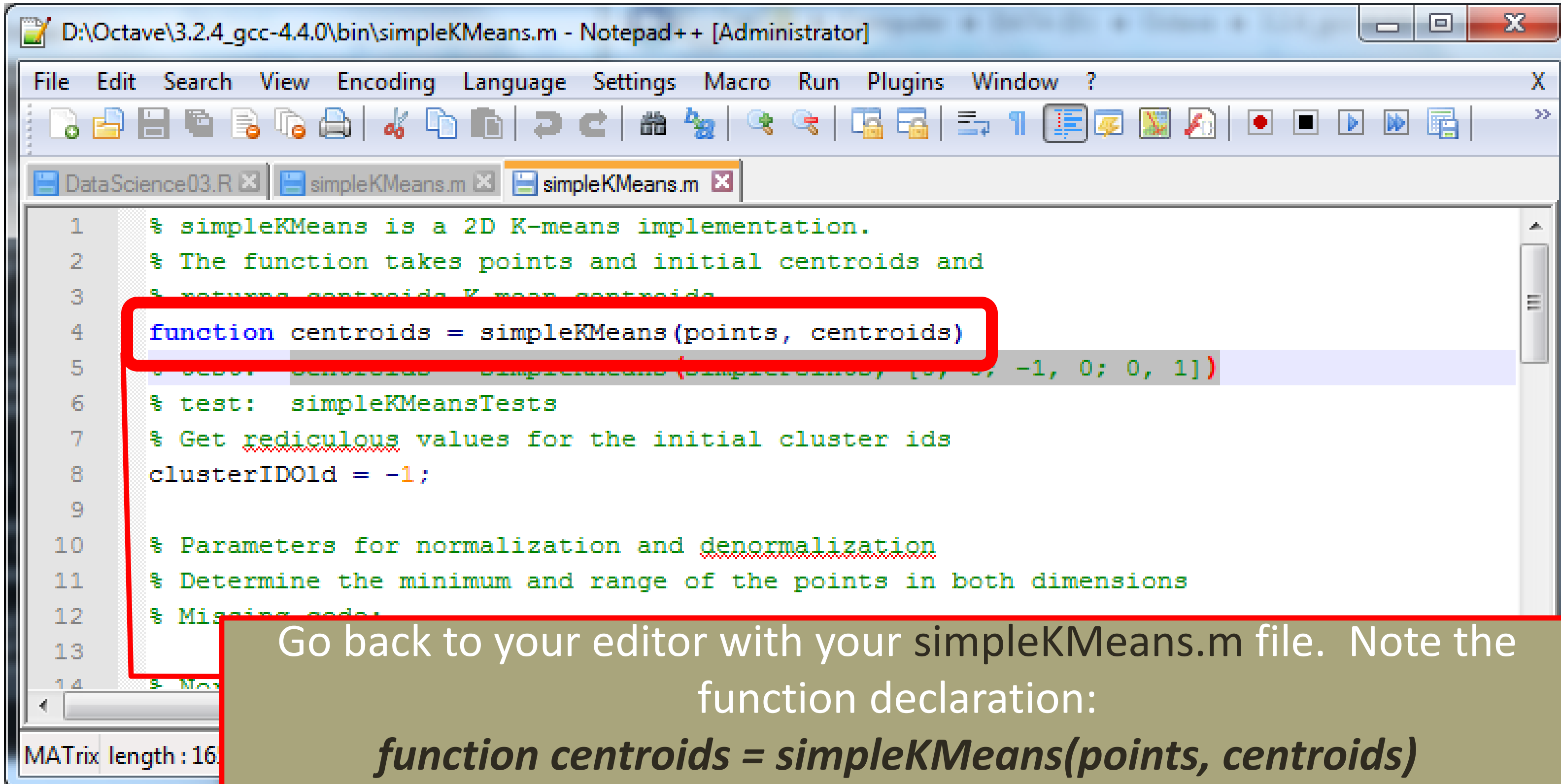


You see the results of the simpleKMeans method. The results are the centroids of the clusters.

For the fun of it, use different starting centroids:

centroids = simpleKMeans(simplePoints, [?, ?; ?, ?; ?, ?])

Octave m-files (10): Function Structure



Go back to your editor with your simpleKMeans.m file. Note the function declaration:

function centroids = simpleKMeans(points, centroids)

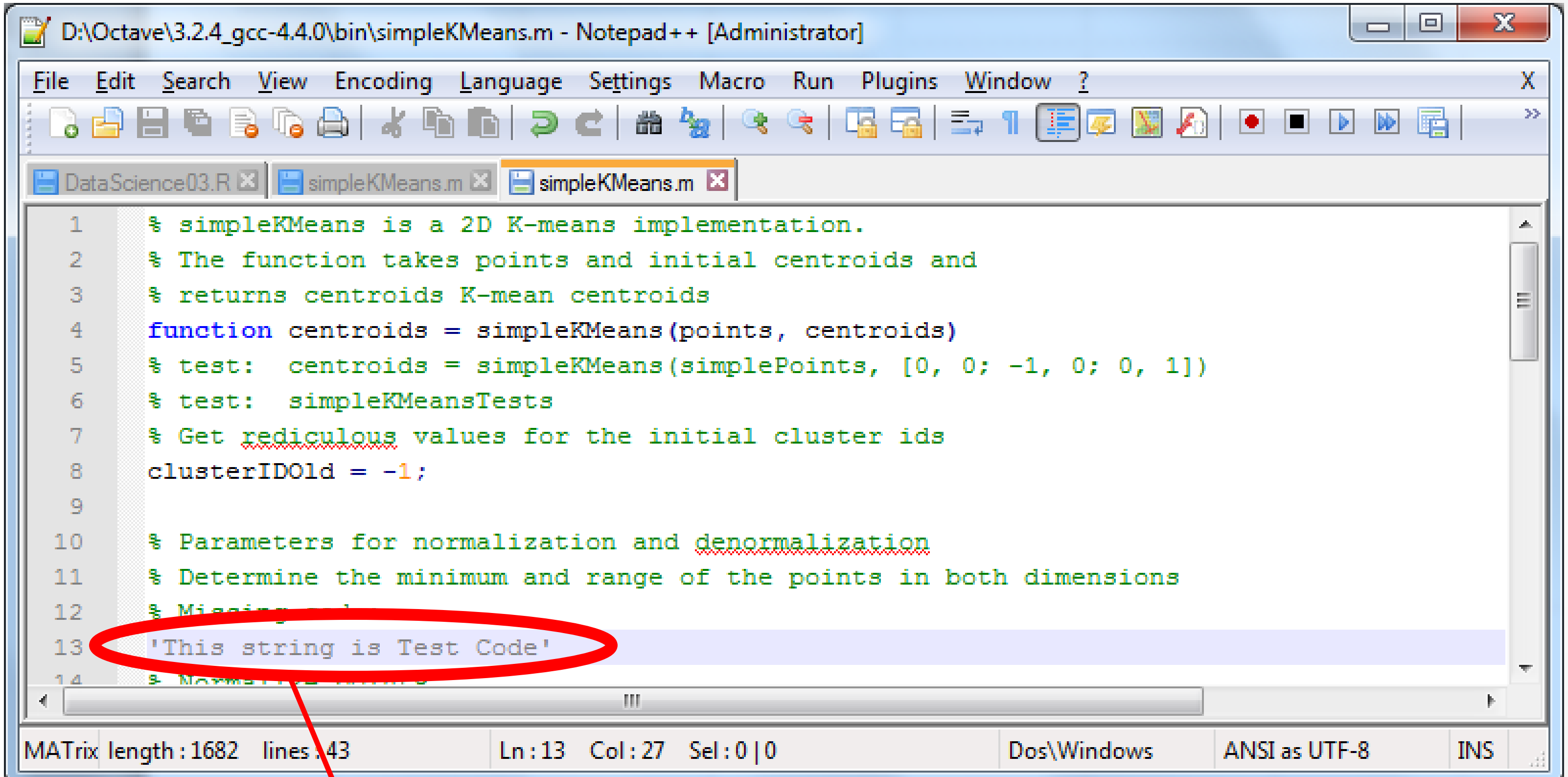
Note the name of the file:

simpleKMeans.M

You can learn how to construct a MATLAB or Octave function here:

http://www.mathworks.com/help/matlab/ref/function.html

Octave m-files (11): Modify Octave Function



Make a change to the code by introducing a simple string. A good place where you can introduce a string is after the first occurrence of ***% Missing code:***. In this example, I introduced ‘This string is Test Code’ on line 13.

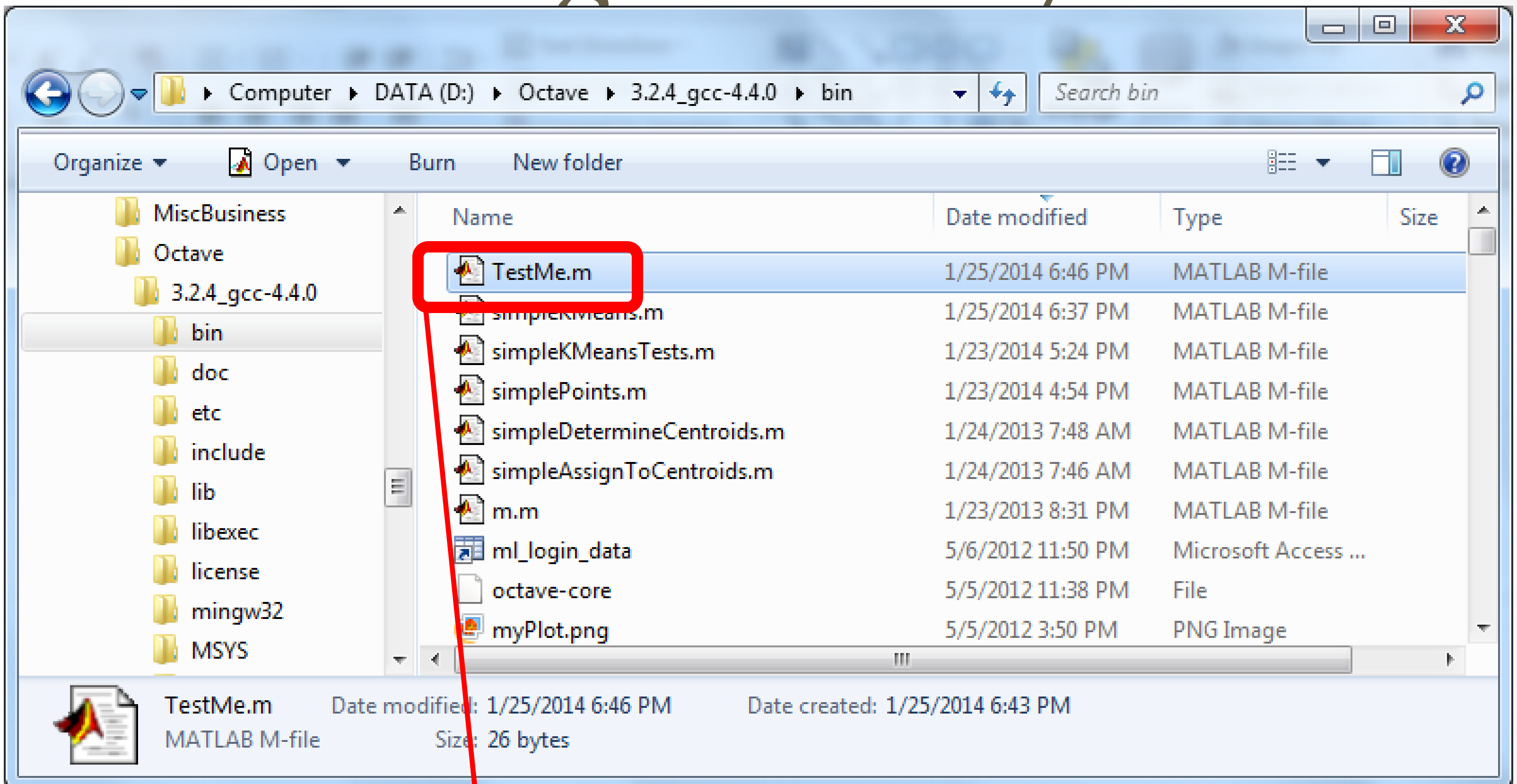
Octave m-files (12): Note the effects of your modification

```
Octave
a =

   5   0  -7

octave-3.2.4.exe:4> pwd
ans = D:\Octave\3.2.4_gcc-4.4.0\bin
octave-3.2.4.exe:5> dir *.m
m.m                          simpleKMeans.m
simpleAssignToCentroids.m    simpleKMeansTests.m
simpleDetermineCentroids.m   simplePoints.m
octave-3.2.4.exe:6> centroids = simpleKMeans(simplePoints, [0, 0; -1, 0; 0, 1])
centroids =

   -1.32500   -0.29000
  -15.25789   -0.88368
    9.09444    0.69194

octave-3.2.4.exe:7> centroids = simpleKMeans(simplePoints, [0, 0; -1, 0; 0, 1])
ans = This string is Test Code
centroids =

   -1.32500   -0.29000
  -15.25789   -0.88368
    9.09444    0.69194

octave-3.2.4.exe:8> _
```

Run simpleKMeans again. You now see the effect of your change to the code.

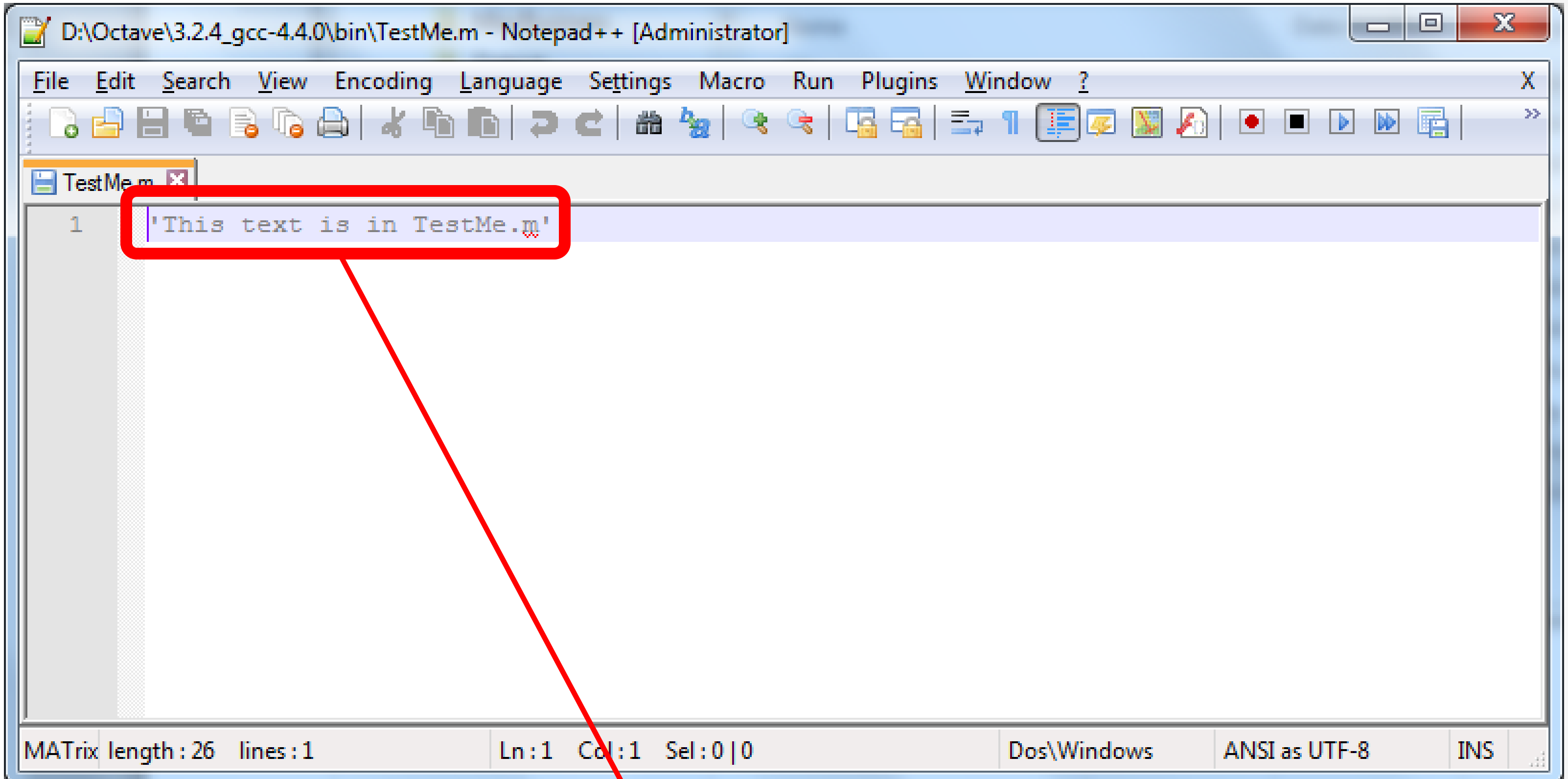
Octave (0) new m-file

Octave (1): Add an m-file to your Working Directory



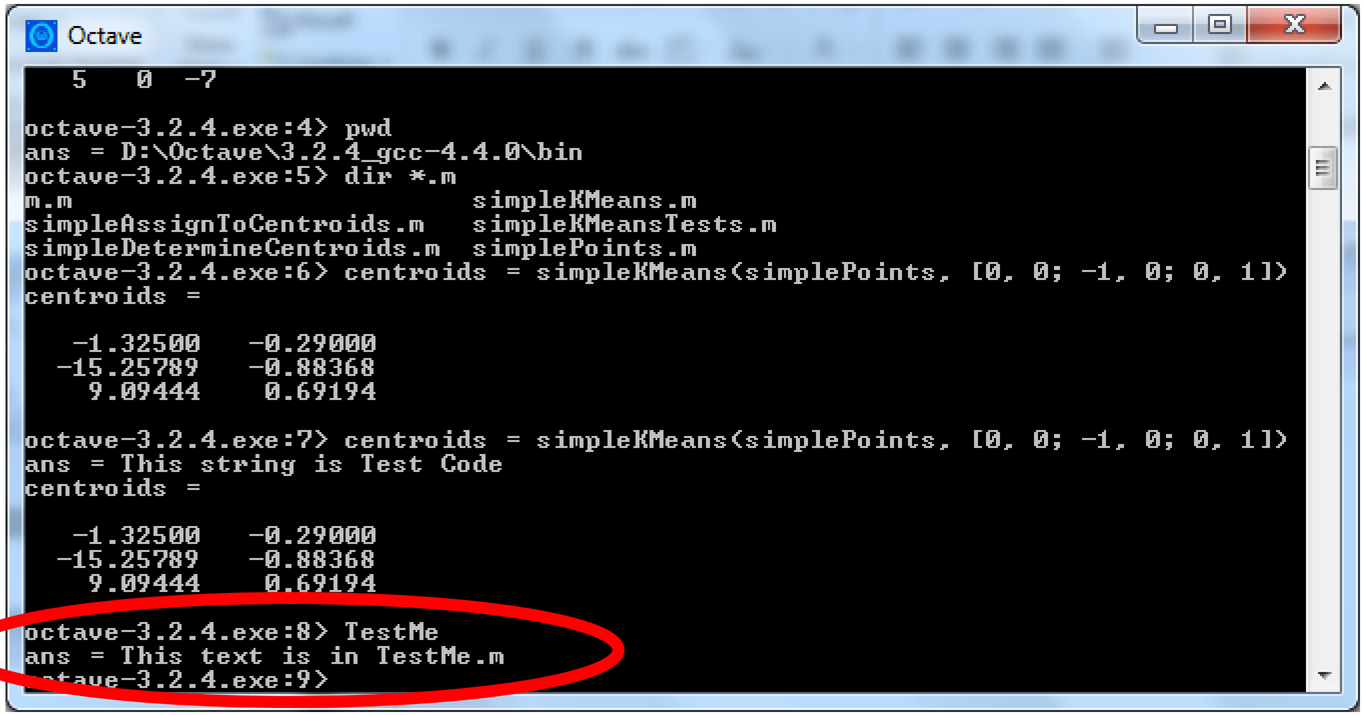
Add an m-file to your working directory. For instance, I created a new text file and renamed it ***TestMe.m***. This m-file has no text, yet.

Octave (2): Add a string to your m-file.



Open the new m-file in your working directory. Add a proper string like: ***'This text is in TestMe.m'*** Save the m-file!

Octave (3): Run your m-file

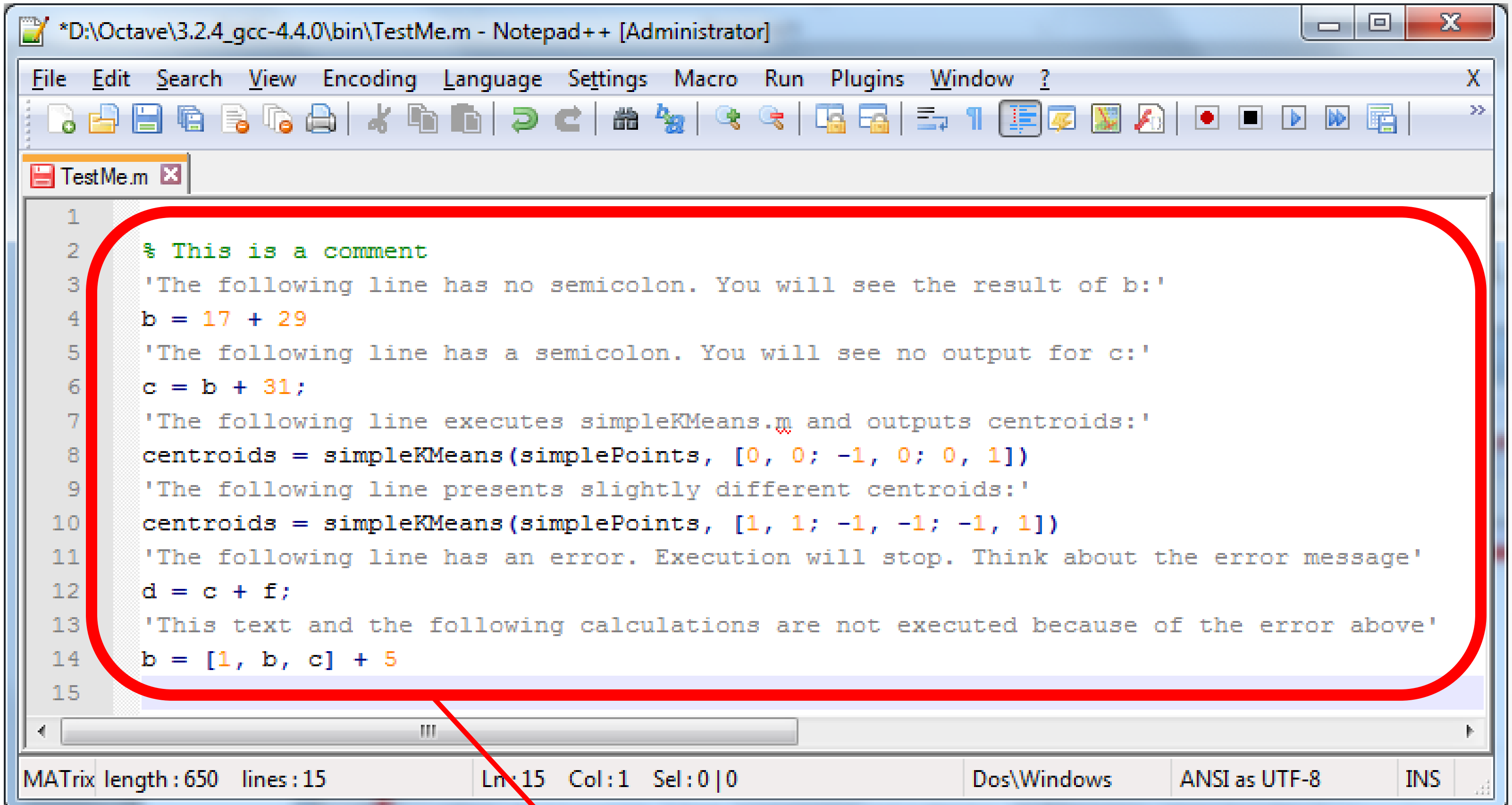


Run TestMe.m by typing TestMe into the Octave console and hitting Enter. Note that console presents the string in TestMe.m.

Octave (4): Paste this Code into TestMe.m

```
% This is a comment
'The following line has no semicolon. You will see the result of b:'
b = 17 + 29
'The following line has a semicolon. You will see no output for c:'
c = b + 31;
'The following line executes simpleKMeans.m and outputs centroids:'
centroids = simpleKMeans(simplePoints, [0, 0; -1, 0; 0, 1])
'The following line presents slightly different centroids:'
centroids = simpleKMeans(simplePoints, [1, 1; -1, -1; -1, 1])
'The following line can be calculated if f is defined.'
d = c + f;
'This text and the following calculations are executed if there is no error'
b = [1, b, c] + 5
```

Octave (5): Edit your m-file



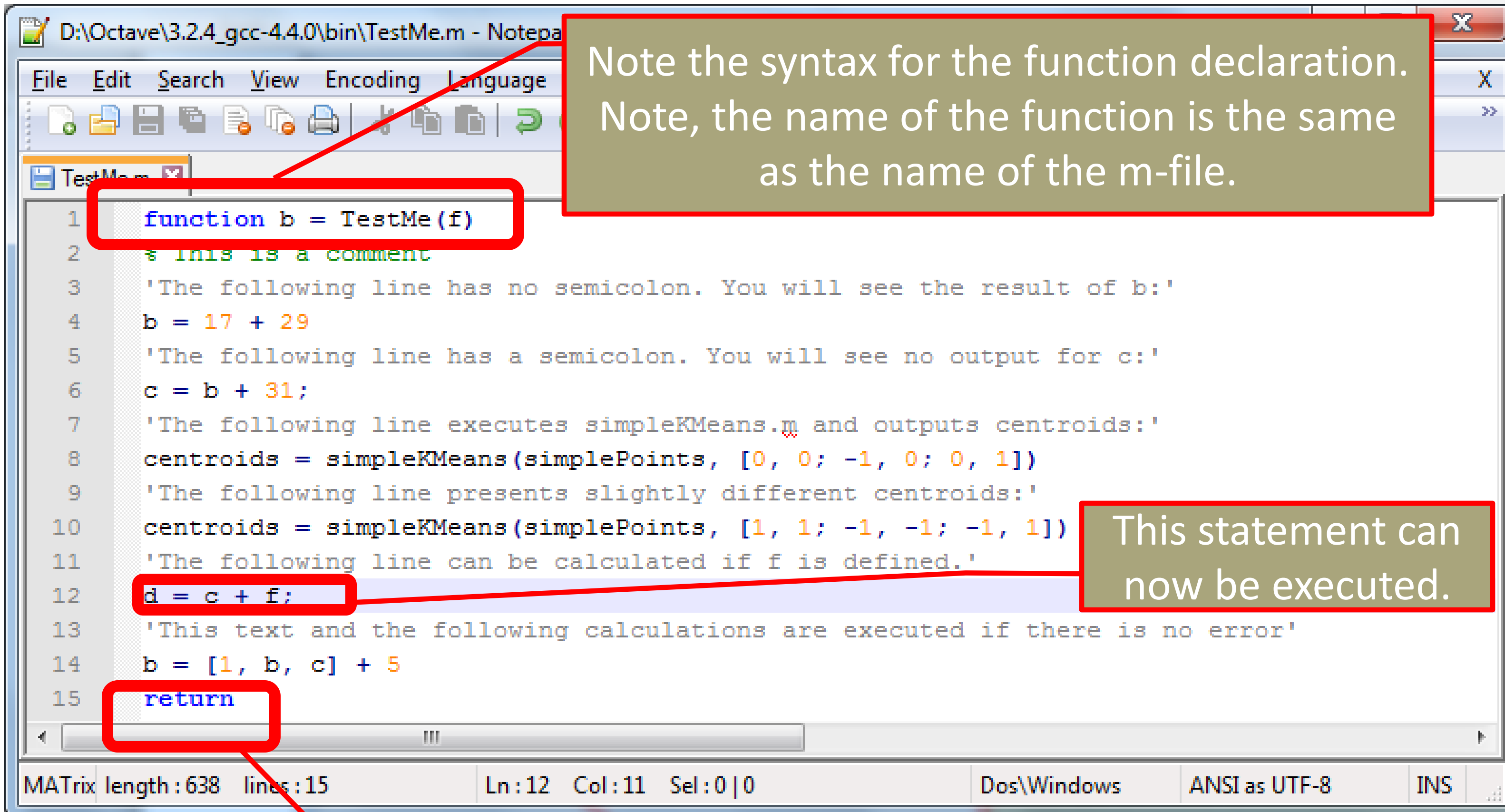
Edit and save your m-file. Pay attention to the text in the m-file and the console.

Octave (6): Run your m-file. Read the error code.

```
octave-3.2.4.exe:1> TestMe
ans = The following line has no semicolon. You will see the result of b:
b =  46
ans = The following line has a semicolon. You will see no output for c:
ans = The following line executes simpleKMeans.m and outputs centroids:
ans = This string is Test Code
centroids =

   -1.32500   -0.29000
  -15.25789   -0.88368
    9.09444    0.69194

ans = The following line presents slightly different centroids:
ans = This string is Test Code
centroids =

    8.94865    0.66568
  -15.63889   -0.93556
   -1.75714   -0.27821

ans = The following line has an error. Execution will stop. Think about the error message
error: `f' undefined near line 12 column 9
error: called from:
error:   D:\Octave\3.2.4_gcc-4.4.0\bin\TestMe.m at line 12, column 3
octave-3.2.4.exe:1>
```

Run your m-file. Relate the console's output to the code. Read the error code.

Octave (7): Create a Function



The return statement is usually not necessary. But, I like it.

Octave (8): Relate Output to Code

Octave

```
octave-3.2.4.exe:2> q = TestMe(-7)
ans = The following line has no semicolon. You will see the result of b:
b =  46
ans = The following line has a semicolon. You will see no output for c:
ans = The following line executes simpleKMeans.m and outputs centroids:
ans = This string is Test Code
centroids =

   -1.32500   -0.29000
  -15.25789   -0.88368
    9.09444    0.69194

ans = The following line presents slightly different centroids:
ans = This string is Test Code
centroids =

    8.94865    0.66568
  -15.63889   -0.93556
   -1.75714   -0.27821

ans = The following line can be calculated if f is defined.
ans = This text and the following calculations are executed if there is no error
b =

    6   51   82

q =

    6   51   82

octave-3.2.4.exe:3> _
```

Execute your function. Note that you can supply an input argument and a return value. Why do you see the return value twice? What happens if you do not supply an input argument? What happens if you do not supply a return value?

Octave (9): Matrix-Oriented Programming

- MatrixOrientedProgrammining.m

```
MA = [1, 2, 3; 8, 9, 0];
numberOfRows = size(MA, 1);
meanMA = mean(MA);

% Loop
for rowNumber = 1:numberOfRows
        MAplusMean(rowNumber, :) = MA(rowNumber, :) + meanMA;
end % end the for loop
MAplusMean

% Vectorized approach
meanMAs = repmat(meanMA, numberOfRows, 1);
MAplusMean = MA + meanMAs

% Broadcasting in Octave 3.2.6
MAplusMean = MA + meanMA
```

Introduction to Octave

Introduction to Data Science