

Patchwork 3

1. Download the image 'leaf.jpg' from Moodle. Calculate the area of the leaf in image pixels in ImageJ, using three different **manual** selection tools (your choice which you use). Write down briefly but clearly the steps you took, and the numerical answer for each. Conclude which of the three you think is best to use in this case, and provide evidence for this. For extra marks, also calculate the area in square centimetres for each. [20 marks]

2. There are a number of **automatic segmentation** algorithms built into ImageJ – we looked at Otsu as an example.
 - a. Research and briefly explain these two types of automated thresholding: intermodes and percentile
 - b. What assumptions does each approach make?[30 marks]

3. In the slides there is some simple pseudocode to make a **histogram**. Your task is to write a python program which will create a histogram from the image. This **MUST** be done **WITHOUT** the use of the built-in histogram function. In other words, you must read pixel values and maintain a list of frequencies yourself. More specifically you must:
 - a. Write code to count the frequencies of intensities of the pixels in an 8-bit greyscale image. You should store frequencies of intensities for each intensity value, 0-255.
 - b. Print out a list of the frequencies for each intensity
 - c. For bonus marks, display this histogram in a new image, similar to the built-in function. You do not need to label axes, just display the plot. You might want to consider scaling the values.

Run this on the 'crater_lake_crop greyscale' image, and submit 1) a screenshot of your code
2) your output list of intensity frequencies 3) your output histogram image (if attempted)

Think about the *quality* of the code you are writing too - make sure you use good style and formatting. [50 marks]

Patch 3

- 1) To get the area in centimeter squared, you use the straight-line tool to draw over the '1cm distance' black box on the bottom right and then used the set scale command. Imagej automatically sets the distance in pixels to '92', and I put '1' in known distance and 'cm' in unit of length. For this question, I first used one of the tools to cover the leaf and get a measurement in image pixels, and then manually put in the numbers in set scale mentioned above to get the area in centimetre squared for the same tool selection on the leaf.

Firstly, I used the oval selection tool and drew a circle that covered the leaf with the rim of the circle touching the edges of the leaf. I used the white squares to help get the entire leaf inside the oval. The area in image pixels was 713380 and was 83.179 in centimeter squared. Secondly, I used the polygon selection and I connected multiple straight lines that went from the top of the leaf, across the border, down the stem and back across the border to the top in a clockwise direction. Each new line was created through clicking. My area in image pixels was 474996 and was 56.120 centimeter squared. Finally, I used the freehand selection, and without letting go off the touchpad, I drew my line across the border of the leaf and stem. My area was 479657 image pixels and was 51.887 in centimeter squared.

I would conclude that the polygon selection tool was the best manual selection tool that you can use for this image, as with oval, the shape of the leaf fits in a large oval but so does a major part of the background (Figure 1). With freehand selection, this process can be difficult with a touchpad and any error results in an immediate redo that can end up being bothersome and time-consuming. With polygon selection, you can match the lines to the edges of the leaves and this works well with a spiky border such as the leaf (Figure 2).

Figure 1 – White areas part of the oval but not part of the leaf shown by arrows

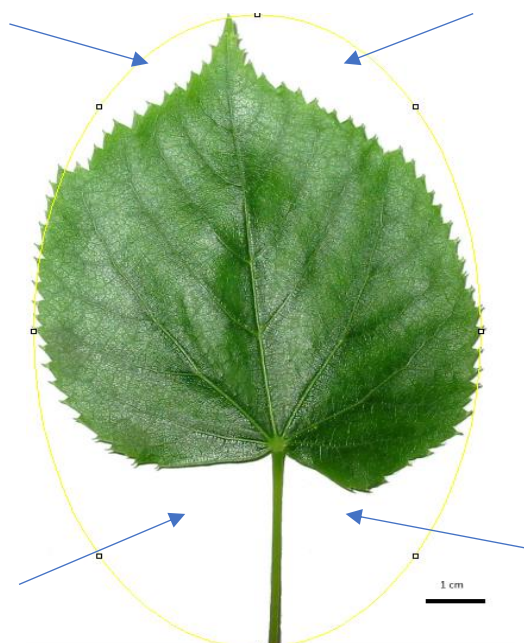


Figure 2 – The yellow lines touches multiple spikes found on the border of the leaf.



2) Intermodes

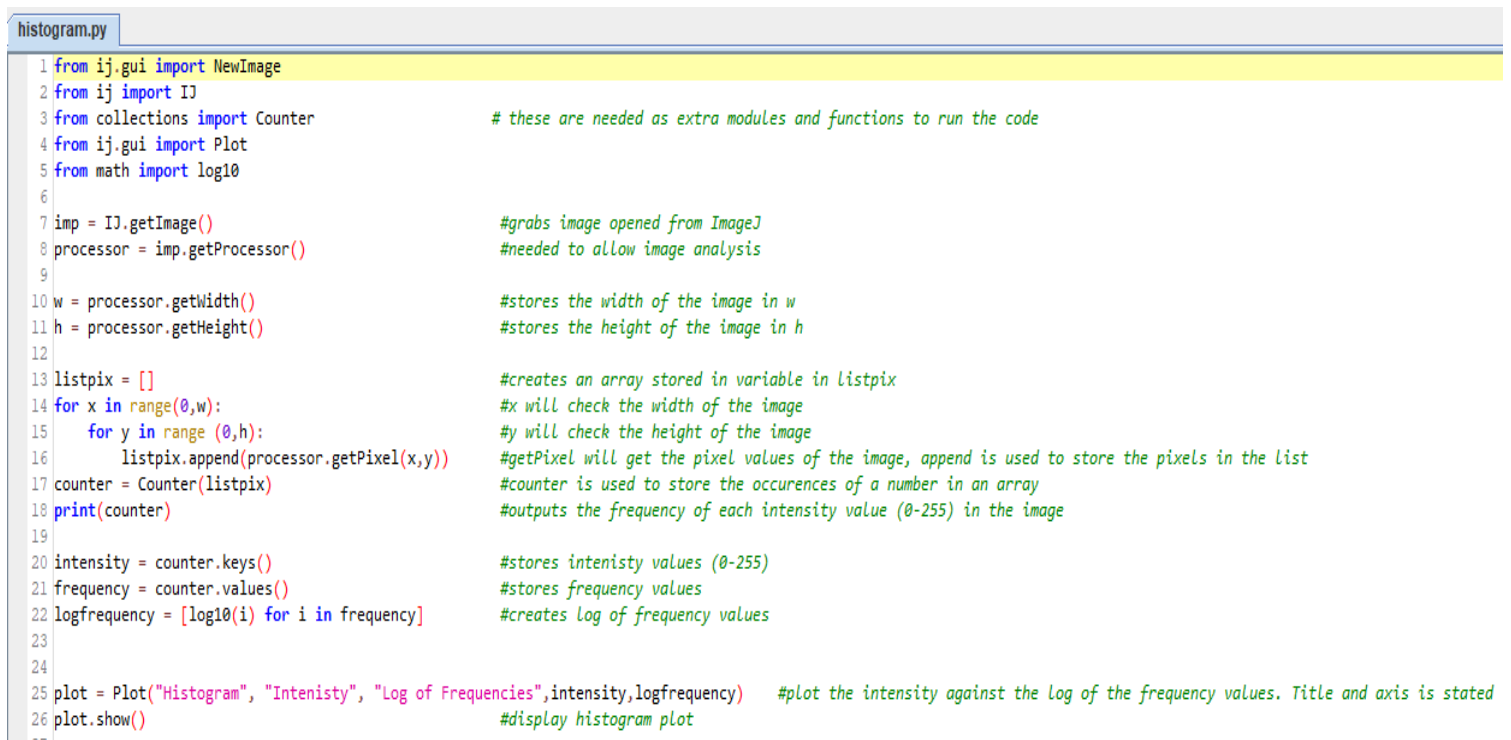
Intermodes assumes a bimodal histogram. Each object to be segmented creates a clear peak around the most frequent grey level value. The bimodal histogram is smoothed iteratively using a mean filter (e.g. three-point mean filter of size 3), until there are only two local maxima: j and k. The threshold, t, is then worked out by averaging the first and second peaks of the histogram i.e. $(j+k)/2$. One limitation of this method is that images that have extremely unequal peaks or a broad histogram will not work well with this method.

Percentile

A percentile does not have a strict definition but in general it could be thought of as all the numbers lower than your target number divided by the total set of numbers, multiplied by 100. So, the highest number in a set will be the 99th percentile. The 25th percentile is called the first quartile, the 50th percentile is the median and the 75th percentile is the third quartile. Percentile thresholding is known as the P-Tile method. This method needs prior knowledge on the area or size of the objects present in the image. The threshold is chosen to be the intensity value where the cumulative sum of pixel intensities is closest to the percentile. Percentile assumes a fraction of foreground pixels to be 0.5.

3)

a) Screenshot of the code



```
1 from ij.gui import NewImage
2 from ij import IJ
3 from collections import Counter          # these are needed as extra modules and functions to run the code
4 from ij.gui import Plot
5 from math import log10
6
7 imp = IJ.getImage()                    #grabs image opened from ImageJ
8 processor = imp.getProcessor()          #needed to allow image analysis
9
10 w = processor.getWidth()                #stores the width of the image in w
11 h = processor.getHeight()               #stores the height of the image in h
12
13 listpix = []                            #creates an array stored in variable in listpix
14 for x in range(0,w):                    #x will check the width of the image
15     for y in range(0,h):                 #y will check the height of the image
16         listpix.append(processor.getPixel(x,y)) #getPixel will get the pixel values of the image, append is used to store the pixels in the list
17 counter = Counter(listpix)              #counter is used to store the occurrences of a number in an array
18 print(counter)                          #outputs the frequency of each intensity value (0-255) in the image
19
20 intensity = counter.keys()               #stores intensity values (0-255)
21 frequency = counter.values()             #stores frequency values
22 logfrequency = [log10(i) for i in frequency] #creates log of frequency values
23
24
25 plot = Plot("Histogram", "Intensity", "Log of Frequencies",intensity,logfrequency) #plot the intensity against the log of the frequency values. Title and axis is stated
26 plot.show()                             #display histogram plot
27
```

Figure 3 – Code used for Patch 3

b) Output list of frequencies for each intensity

{0: 2, 1: 15, 2: 39, 3: 83, 4: 202, 5: 485, 6: 9647, 7: 564, 8: 468, 9: 220, 10: 123, 11: 101, 12: 81, 13: 68, 14: 52, 15: 42, 16: 37, 17: 31, 18: 29, 19: 25, 20: 23, 21: 19, 22: 23, 23: 13, 24: 10, 25: 14, 26: 12, 27: 14, 28: 16, 29: 11, 30: 11, 31: 11, 32: 20, 33: 10, 34: 12, 35: 15, 36: 14, 37: 12, 38: 15, 39: 26, 40: 25, 41: 28, 42: 35, 43: 46, 44: 40, 45: 56, 46: 57, 47: 56, 48: 54, 49: 58, 50: 73, 51: 61, 52: 71, 53: 80, 54: 70, 55: 98, 56: 83, 57: 91, 58: 108, 59: 92, 60: 108, 61: 122, 62: 119, 63: 120, 64: 114, 65: 107, 66: 154, 67: 118, 68: 131, 69: 133, 70: 136, 71: 147, 72: 125, 73: 141, 74: 127, 75: 116, 76: 117, 77: 127, 78: 124, 79: 134, 80: 121, 81: 142, 82: 143, 83: 150, 84: 138, 85: 103, 86: 130, 87: 119, 88: 111, 89: 110, 90: 99, 91: 94, 92: 96, 93: 98, 94: 102, 95: 93, 96: 101, 97: 106, 98: 83, 99: 96, 100: 82, 101: 103, 102: 81, 103: 96, 104: 89, 105: 82, 106: 90, 107: 87, 108: 111, 109: 103, 110: 102, 111: 78, 112: 102, 113: 96, 114: 103, 115: 108, 116: 119, 117: 93, 118: 86, 119: 94, 120: 106, 121: 119, 122: 97, 123: 105, 124: 115, 125: 94, 126: 121, 127: 130, 128: 114, 129: 101, 130: 83, 131: 109, 132: 109, 133: 129, 134: 87, 135: 106, 136: 79, 137: 127, 138: 105, 139: 107, 140: 118, 141: 122, 142: 105, 143: 121, 144: 101, 145: 116, 146: 116, 147: 108, 148: 97, 149: 102, 150: 128, 151: 114, 152: 119, 153: 116, 154: 99, 155: 111, 156: 121, 157: 123, 158: 119, 159: 113, 160: 120, 161: 114, 162: 123, 163: 107, 164: 116, 165: 113, 166: 107, 167: 121, 168: 104, 169: 118, 170: 112, 171: 114, 172: 116, 173: 98, 174: 99, 175: 104, 176: 112, 177: 110, 178: 95, 179: 114, 180: 125, 181: 100, 182: 91, 183: 117, 184: 99, 185: 99, 186: 93, 187: 113, 188: 86, 189: 99, 190: 103, 191: 95, 192: 93, 193: 100, 194: 83, 195: 111, 196: 90, 197: 102, 198: 95, 199: 86, 200: 113, 201: 96, 202: 101, 203: 91, 204: 93, 205: 94, 206: 93, 207: 100, 208: 93, 209: 100, 210: 111, 211: 87, 212: 100, 213: 105, 214: 99, 215: 98, 216: 102, 217: 93, 218: 87, 219: 108, 220: 118, 221: 96, 222: 102, 223: 103, 224: 103, 225: 98, 226: 101, 227: 128, 228: 121, 229: 117, 230: 119, 231: 118, 232: 140, 233: 148, 234: 137, 235: 129, 236: 173, 237: 140, 238: 176, 239: 164, 240: 186, 241: 167, 242: 190, 243: 230, 244: 270, 245: 293, 246: 337, 247: 427, 248: 402, 249: 416, 250: 314, 251: 241, 252: 171, 253: 70, 254: 32, 255: 5}

Figure 4 – Output (a: b), a = Pixel intensity values (0-256) , b = Frequency of pixel value

c) Histogram

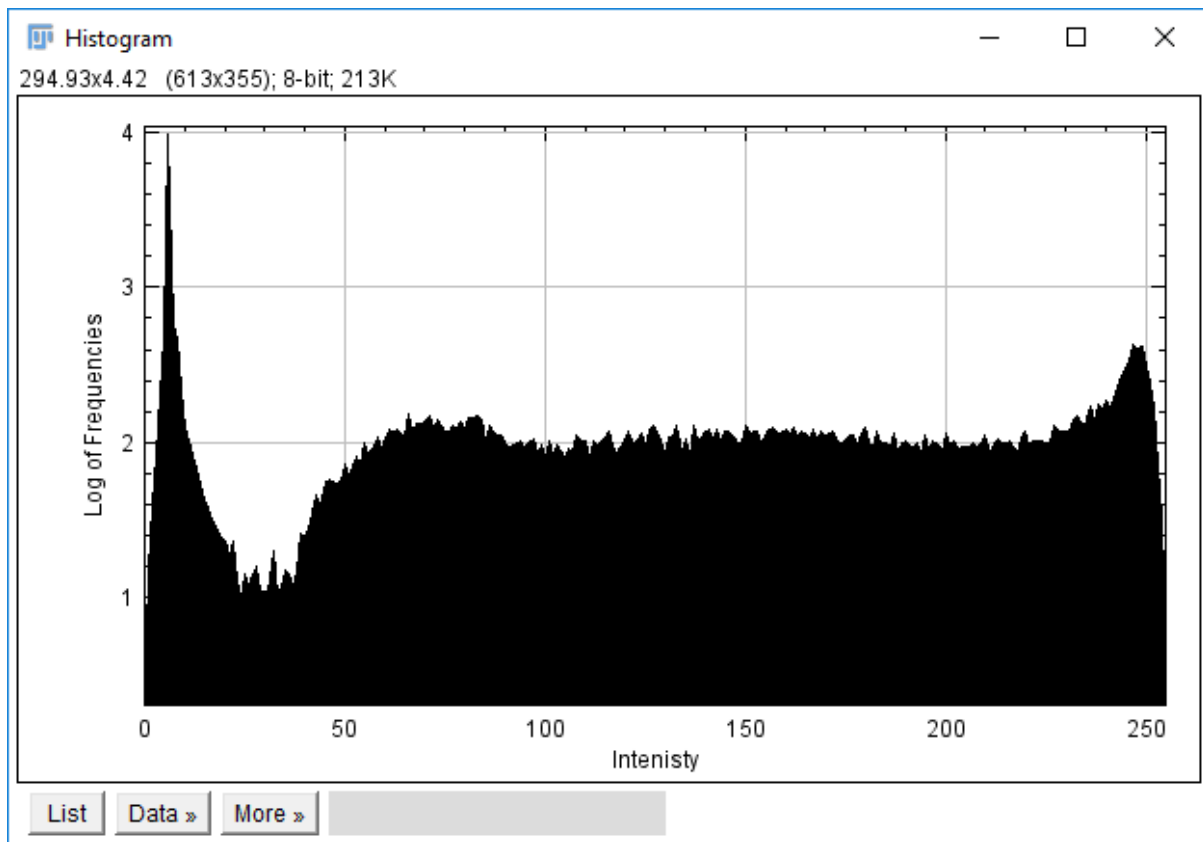


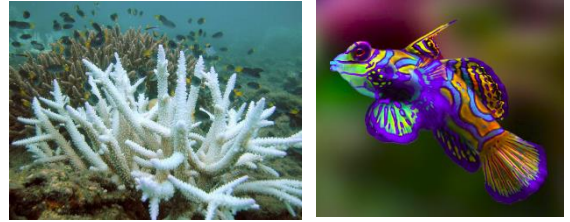
Figure 5 – Histogram created by code

In terms of scaling the values, I did \log_{10} to the frequency values just so that the data is easier to see, as the frequencies have high and low values.

Patchwork 4:

PART A

Download the **patchword_4_images.zip** file from Moodle.



For each of the two images in the file:

1. Choose one of the two **model based** approaches we used in the lab that you think will work best, and justify why you chose it.
2. Segment the main foreground (ie. the pale coral or the fish) object using the approach. Perform any prior processing on the image as you deem necessary, and justify.
3. Record in the patchwork for each image:
 - a. Any pre-processing of the image you performed and why
 - b. Which segmentation approach you chose and why
 - c. A screen shot of the output – ie. a selection around the original object
 - d. The area, in pixels, of the foreground object.
 - i. Fish
 - ii. Coral

Tips

- The output might not create a selection directly – you might need to figure out or look up how to create a selection over the original image, to take a measurement from.

[50 marks]

PART B

1. K-means is a clustering algorithm. With the following numbers, MANUALLY work through the k-means algorithm until the means of the clusters do not change. For each iteration, show the means and clusters you are working with.

We can assume $k=2$, and the initial means to use are 8 and 15.

9 4 11 15 8 7 13 14 6 [15 marks]

2. Write Python code to perform k-means clustering with 3 clusters. The code must:
 - a. Load in a data file with the name “data.txt” in the same folder as the program.
 - b. The format of the file it can read **must match the supplied data.txt**. file. i.e.
 - i. First line the initial cluster centres: mean1, mean2, mean3
 - ii. Subsequent lines: the data
 - c. Each iteration you must print out the clusters, and the means.
 - d. The software should stop after the means do not change, OR until 10 iterations are reached.

You do not need to use the Fiji editor for this, use your Python editor of choice (e.g Spyder, etc.)

We will test the code you submit on a NEW data.txt file.

[35 marks]

Partial marks will be awarded for partial answers. Patchwork 4 should take MAX 3 pages of A4. Submit on Moodle as a single PDF comprising Patchworks 3, 4 and 5. **Also submit your .py file for PART B (2) as an additional file.**

Patch 4

Part A

1) Model based approaches to segmentation – Trainable Weka Segmentation and SplineSnake

Trainable Weka Segmentation combines multiple machine learning algorithms with various image features to produce pixel-based segmentations. Weka has a collection of visualisation tools; algorithms for data analysis and predictive modelling; and is easy to use due to graphical user interfaces. There are two classes producing a binary pixel classification and by drawing traces you can segment the image into foreground for class 1 and background for class 2.

SplineSnake uses the snake algorithm for the segmentation of greylevel images. The snake is a spline curve that is attracted by image information to the object boundaries. The formulation ensures that the extracted boundary is smooth. In terms of pre-processing, for SplineSnake, the images had to be converted to a greylevel.

Weka Segmentation and SplineSnake – Fish

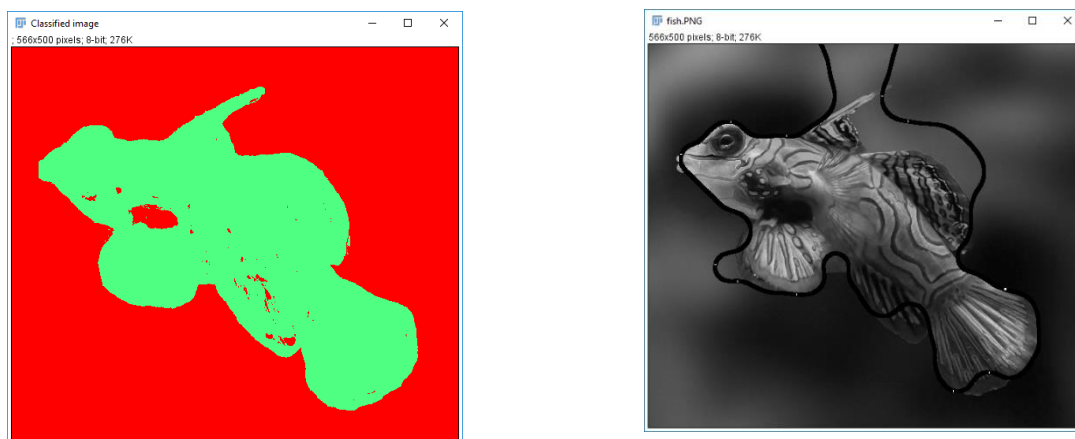


Figure 7 – Output of two model-based segmentation to the Fish Image, left = Weka Segmentation, right = SplineSnake

From Figure 7, we can see that the Trainable Weka Segmentation approach is better for the image than SplineSnake as the active contours do not go around the fish. Too much of the background is taken into the foreground and the active contour cuts into the fins of the fish at certain places. The one issue with the Trainable Weka Segmentation is that the darker spot above the left fin is wrongly classified as background, as well other areas (the red marks found in the green foreground image). From Trainable Weka Segmentation, to find the area I used polygon interactions and highlighted around the foreground, giving the area of 95736 pixels squared.

Weka Segmentation and SplineSnake – Coral

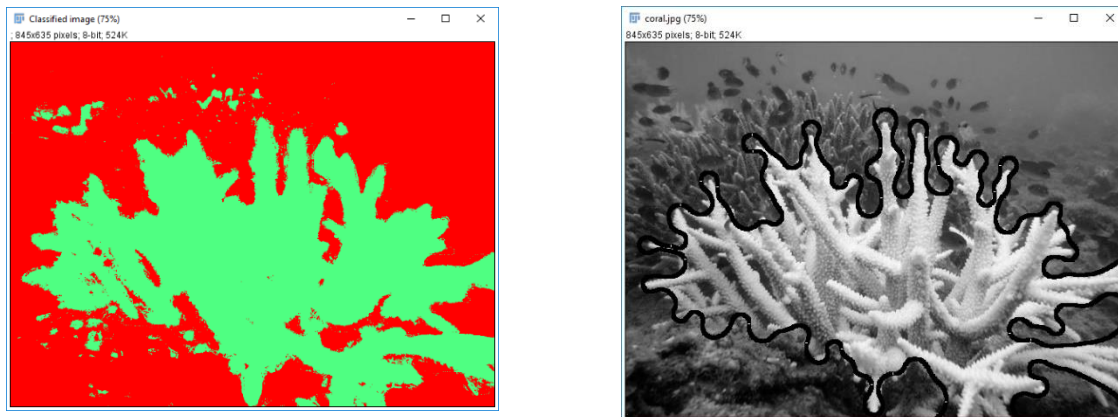


Figure 8 – Output of two model-based segmentation to the Coral Image, left = Weka Segmentation, right = SplineSnake

From Figure 8, we can see that the SplineSnake has worked better than the Trainable Weka Segmentation for the Coral image. The SplineSnake has covered the boundary of the coral fairly well except for the bottom right of the image. The Trainable Weka Segmentation has done well but it has included some of the background into the foreground class, which SplineSnake has not done. Also, SplineSnake looks rounder around the spikes of the coral. The coral's boundary has more edges and is rougher than the fish due to all the spikes, which could explain why it worked better. The area of the coral calculated using SplineSnake is 214817 pixels squared.

From these two images we can see that the Trainable Weka Segmentation, works fairly well for both images, so if you needed a general model-based approach for a lot of images, I would choose Weka Segmentation over SplineSnake. In terms of the process, both require a lot of human judgment, the Trainable Weka Segmentation took longer but I felt that the SplineSnake was tough to work with it.

Part B

1) Iteration 1

Cluster 1 – Mean 8: 9,4,11,8,7,6 new mean is 7.5

Cluster 2 – Mean 15: 15,13,14 new mean is 14

Iteration 2

Cluster 1 – Mean 7.5: 9,4,8,7,6 new mean is 6.8

Cluster 2 – Mean 14: 11,15,13,14 new mean is 13.25

Iteration 3

Cluster 1 – Mean 6.8: 9,4,8,7,6

Cluster 2 – Mean 13.25: 11,15,13,14

New means are the same as the old ones (6.8 and 13.25) so no need for more iterations.

$ 9-8 =1$	$ 9-15 =6$	so 9 grouped with 8
$ 4-8 =4$	$ 4-15 =11$	so 4 grouped with 8
$ 11-8 =3$	$ 11-15 =4$	so 3 grouped with 8
$ 15-8 =7$	$ 15-15 =0$	so 15 grouped with 15
$ 8-8 =0$	$ 8-15 =7$	so 8 grouped with 8
$ 7-8 =1$	$ 7-15 =8$	so 7 grouped with 8
$ 13-8 =5$	$ 13-15 =2$	so 13 grouped with 15
$ 14-8 =6$	$ 14-15 =1$	so 14 grouped with 15
$ 6-8 =2$	$ 6-15 =9$	so 6 grouped with 8

$$(9+4+11+8+7+6)/6 = 7.5$$

$$(15+13+14)/3 = 14$$

$ 9-6.8 =2.2$	$ 7-13.25 =4.25$	9 grouped with 6.8
$ 4-6.8 =2.8$	$ 4-13.25 =9.25$	4 grouped with 6.8
$ 11-6.8 =4.25$	$ 11-13.25 =2.25$	11 grouped with 13.25
$ 15-6.8 =8.2$	$ 15-13.25 =1.75$	15 grouped with 13.25
$ 8-6.8 =1.2$	$ 8-13.25 =5.25$	8 grouped with 6.8
$ 7-6.8 =0.2$	$ 7-13.25 =6.25$	7 grouped with 6.8
$ 13-6.8 =6.2$	$ 13-13.25 =0.25$	13 grouped with 13.25
$ 14-6.8 =7.2$	$ 14-13.25 =0.75$	14 grouped with 13.25
$ 6-6.8 =0.8$	$ 6-13.25 =7.25$	6 grouped with 6.8

$$(9+4+8+7+6)/5 = 6.8$$

$$(11+15+13+14)/5 = 13.25$$

$ 9-7.5 =1.5$	$ 9-14 =5$	so 9 grouped with 7.5
$ 4-7.5 =3.5$	$ 4-14 =10$	so 4 grouped with 7.5
$ 11-7.5 =3.5$	$ 11-14 =3$	so 11 grouped with 14
$ 15-7.5 =7.5$	$ 15-14 =1$	so 15 grouped with 14
$ 8-7.5 =0.5$	$ 8-14 =6$	so 8 grouped with 7.5
$ 7-7.5 =0.5$	$ 7-14 =7$	so 7 grouped with 7.5
$ 13-7.5 =5.5$	$ 13-14 =1$	so 13 grouped with 14
$ 14-7.5 =6.5$	$ 14-14 =0$	so 14 grouped with 14
$ 6-7.5 =1.5$	$ 6-14 =8$	so 6 grouped with 7.5

$$(9+4+8+7+6)/5 = 6.8$$

$$(11+15+13+14)/4 = 13.25$$

Figure 9 – Working out of Part B Q1. The absolute difference of each value against the means was calculated. Grouping was based on how close the absolute difference values were to the means of each cluster. Then a new mean was created based on the value inside that cluster.