Class 7: Machine Learning 1

Lilith Sadil, A16470107

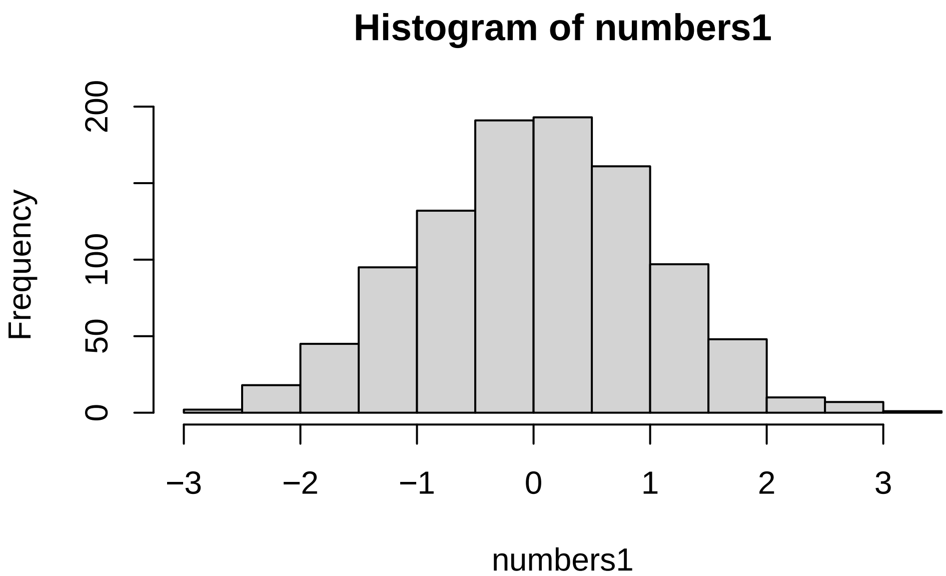
Today, we'll start out multi-part exploration of some key machine learning methods. We'll begin with clustering (a way of bunching/grouping data based off of similarity/patterns and then using dimensional reduction).

Clustering

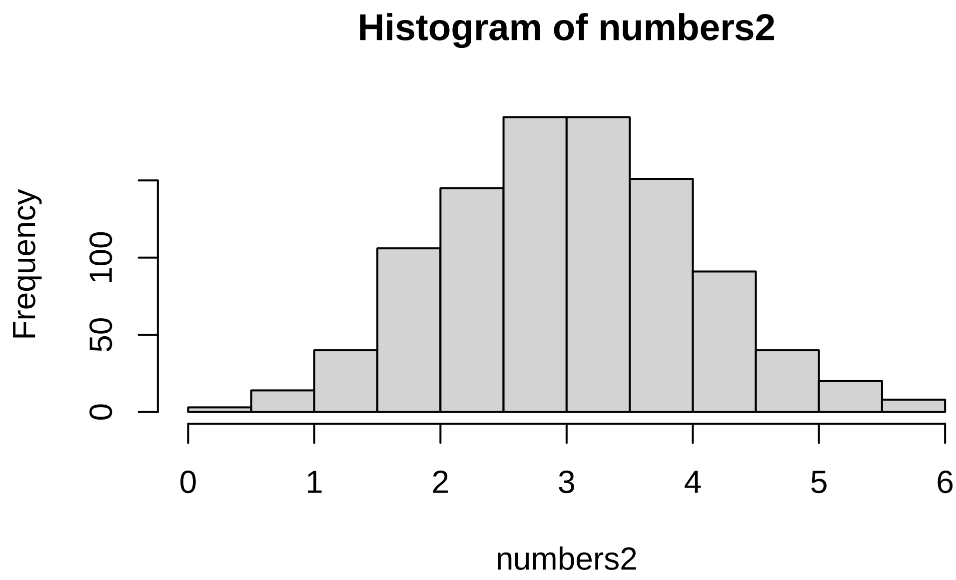
Let's start with "k-means" clustering. In this approach, you define how many groups (# = k) you want and then the computer bunches the data you provide into k groups

The main function in base R for this is `kmeans()`.

```
#Make up some data; here, 1000 points will be randomly generated along a normal distributi
numbers1 = rnorm(1000)
# we can turn this data into a histogram:
hist(numbers1)
```




```
#we can also make a histogram with a shifted mean - at 3 instead of 0
numbers2 = rnorm(1000, mean=3)
hist(numbers2)
```



Next, we'll make two data sets of 30 points - one is centered around -3 and the other around +3

```
rnorm(30,-3)
```

```
 [1] -3.596417 -2.448804 -2.661117 -2.706176 -3.284758 -1.760778 -4.170519
 [8] -3.391516 -2.356209 -3.647777 -3.094302 -3.445408 -3.509427 -2.329083
[15] -4.093018 -4.500561 -1.744900 -1.735718 -2.551377 -2.563225 -2.903172
[22] -2.274950 -3.892316 -2.521947 -2.936546 -1.943961 -3.417303 -2.273850
[29] -3.633772 -1.765607
```

```
rnorm(30,3)
```

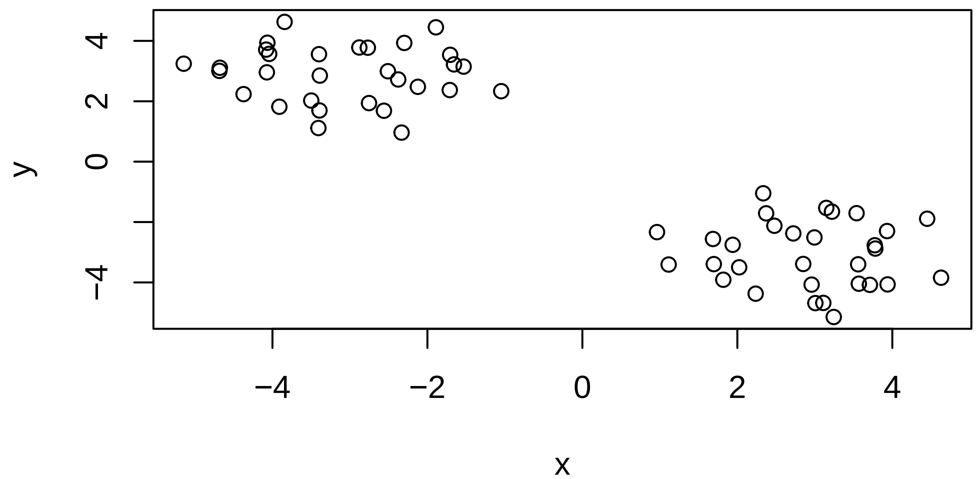
```
 [1] 3.794407 1.730416 3.952541 2.995379 2.210211 1.265681 1.772598 2.439948
 [9] 3.607551 4.460356 2.458461 4.182485 3.192787 2.168058 3.459718 1.367180
[17] 3.798306 1.961188 2.234658 1.113222 3.312811 4.139672 2.863702 3.691212
[25] 3.539662 3.155386 4.072791 3.831392 3.918510 4.849186
```

Next, we'll concatenate these two data sets/vectors into one data set

```
combined_data = c(rnorm(30,-3), rnorm(30,3))
```

If we want to print the values centered around +3 first, we can reverse the order of the vector

```
combined_data = c(rnorm(30,-3), rnorm(30,3))
xy_data = cbind(x=combined_data, y=rev(combined_data))
plot(xy_data)
```



Now, we'll use kmeans to analyse these groups

```
km = kmeans(xy_data, centers=2)
#in the parentheses of kmeans, we input x (numeric data matrix we're analysing) and center
km
```

```
K-means clustering with 2 clusters of sizes 30, 30

Cluster means:
          x         y
1 -3.071113  2.893238
2  2.893238 -3.071113
```

```
Clustering vector:
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1] 59.29021 59.29021
 (between_SS / total_SS =  90.0 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

```
#the center of each cluster group is located at (3.11,-2.97) and (-2.97, 3.11)
```

When we use the k-means operation, we are able to access 9 different pieces of information/components regarding the data set - one is "size" which tells us the size of each cluster

> Q. How many points in each cluster?

```
km$size
```

```
[1] 30 30
```

> Q. What component of your result object details cluster allignment/membership?

```
km$cluster
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

> Q. What are centers/mean values of each cluster?

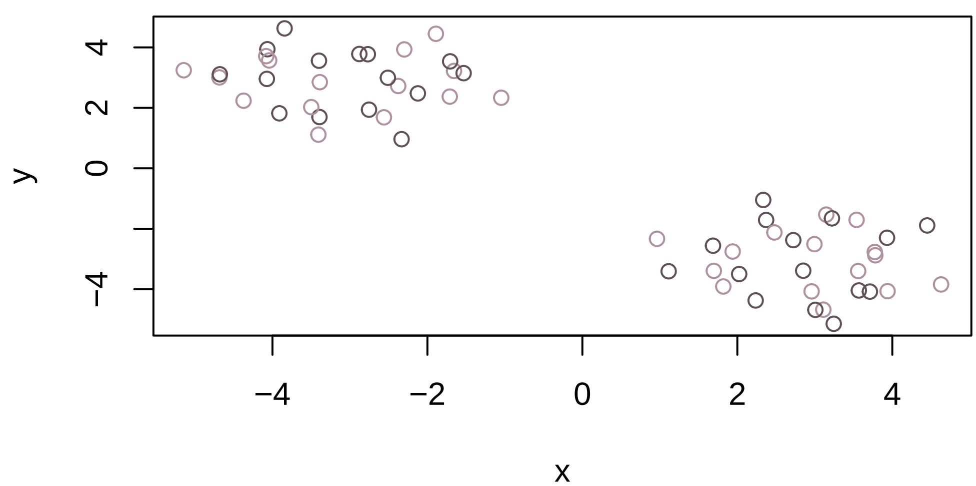
```
km$centers
```

```
          x         y
1 -3.071113  2.893238
2  2.893238 -3.071113
```

> Q. Make a plot of the data showing clustering results.

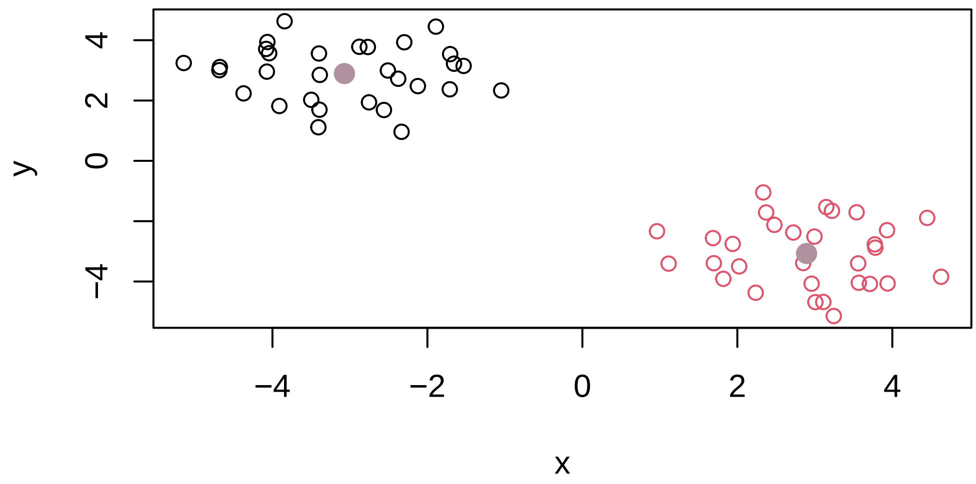
Here, we want to separate out our data clusters based on color. How do we do this? By default, if only 2 color values are assigned, then those colors will be "recycled" to alternate the color of every point.

```
plot(xy_data, col=c("#AF929D", "#615055"))
```



To assign color by clusters, we have to split the data clusters into their respective vectors and color them (items=1 are black, items=2 are red). Next, we'll mark the cluster centers with a pink dot

```
plot(xy_data, col=km$cluster)
points(km$centers, col="#AF929D", pch=20, cex=2)
```



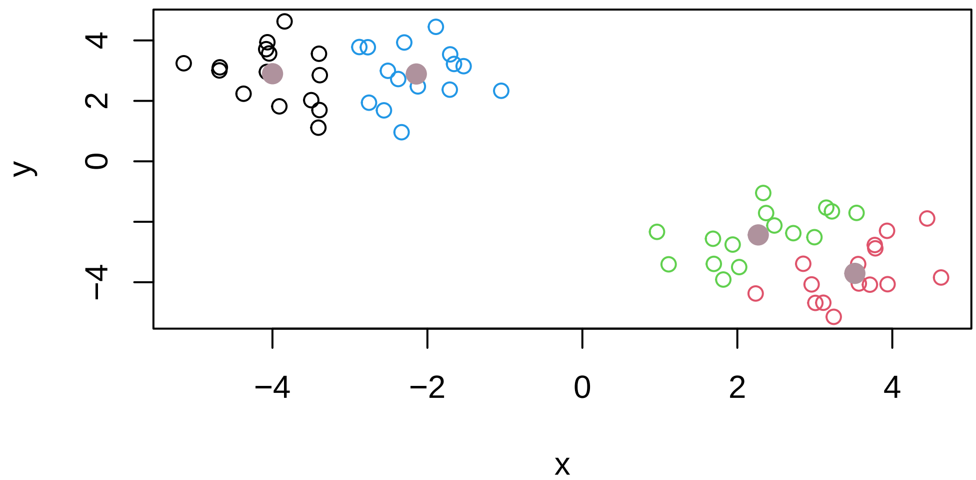
Q. Run `kmeans()` again and cluster into four groups then plot.

```r
kmeans(xy_data, centers=4)
```

```
K-means clustering with 4 clusters of sizes 7, 30, 11, 12

Cluster means:
          x         y
1 -3.122661  1.605793
2  2.893238 -3.071113
3 -4.161219  3.346488
4 -2.041779  3.228769

Clustering vector:
 [1] 3 1 4 3 4 1 3 4 3 1 4 3 3 4 1 3 4 4 4 1 1 4 4 4 1 3 3 3 3 4 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

Within cluster sum of squares by cluster:
[1]  2.993512 59.290206  6.904859  8.373880
 (between_SS / total_SS =  93.5 %)

Available components:
```

```
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

```r
km4 = kmeans(xy_data, centers=4)
plot(xy_data, col=km4$cluster)
points(km4$centers, col="#AF929D", pch=20, cex=2)
```



Here, the clusters aren't as distinct as before - however, 4 clusters were formed since we told the computer it had to.

Hierarchical Clustering

This form of clustering aims to reveal the structure in your data by progressively grouping points into an ever smaller number of clusters.

The main function in base R for this is called `hclust()`. This function doesn't take our input data directly; it requires a "distance matrix" that details how dis/similar all out input points are to each other.

We can use the `dist()` function on our xy_data dataset in order to get the distance between every pair of points in the dataset

```r
dist(xy_data)
```

```
            1          2          3          4          5          6          7
2   3.1151670                                                                  
3   3.0354590  2.3573405                                                       
4   1.1190258  3.4434031  2.5136786                                            
5   3.6986481  1.8801216  1.0756810  3.4181296                                 
6   2.2060413  1.0633575  1.6867137  2.3912492  1.7512147                      
7   0.8527644  3.1155553  2.4109383  0.3717598  3.2390708  2.0765471           
8   2.6152925  1.5297785  0.8771866  2.4304086  1.0852688  0.8290773  2.2070938
9   1.3049768  2.1624060  1.7715459  1.2818552  2.3976192  1.1102340  0.9706492
10  1.4176684  1.7945937  2.6548368  2.1259026  2.9094954  1.1635750  1.7541681
11  2.5589692  2.9695433  0.9582688  1.7668044  2.0295105  2.0437594  1.7813324
12  0.1028124  3.1792256  3.0242135  1.0316446  3.7130698  2.2525755  0.7843407
13  0.8312701  2.4040869  2.8892144  1.7303771  3.3267638  1.6460382  1.3721986
14  2.0635411  2.8445151  1.2421237  1.3071501  2.2439841  1.8342211  1.2901224
15  2.2840189  1.0839130  2.7395937  2.9013124  2.6566485  1.0533764  2.5354641
16  0.6140655  2.6467464  2.4299650  0.9806703  3.0885535  1.6662812  0.6103383
17  2.3249043  1.7601726  0.8760315  2.0810768  1.3850579  0.8686776  1.8677111
18  1.9646732  2.8701389  1.3442448  1.1973153  2.3335866  1.8452779  1.1830121
19  3.1444604  3.5151884  1.2512755  2.2339240  2.2776519  2.6544816  2.3253849
20  1.8396872  1.2887877  2.3110279  2.3413083  2.4310798  0.6844408  1.9810545
21  2.5009148  0.7578453  1.7816366  2.7093865  1.6464793  0.3188015  2.3952927
22  3.0251617  2.6509537  0.3215511  2.3928213  1.3727696  1.9112819  2.3361663
23  3.0401204  1.5401399  0.8509234  2.8280792  0.6649968  1.1282369  2.6200123
24  3.1550185  2.3270913  0.1439333  2.6536007  0.9471566  1.7176551  2.5447287
25  1.5399504  1.5760438  2.1972819  1.9972620  2.4712110  0.7501555  1.6370132
26  1.8275606  3.9651446  2.6011281  0.7251267  3.6172353  2.9019614  1.0798172
27  0.9273043  3.2561284  2.4730852  0.2289425  3.3308601  2.2129053  0.1476574
28  1.3985773  2.8066510  1.7755605  0.7665983  2.6519669  1.7435270  0.6428473
29  0.5180611  3.6204669  3.4885416  1.2839437  4.1973308  2.7239226  1.1492581
30  2.1743655  2.0398313  0.8829398  1.8194018  1.6049877  1.0830471  1.6358735
31  9.4557146  6.3604746  7.3810986  9.5624990  6.3094335  7.2696756  9.2988105
32 11.3715942  8.2713940  9.6955106 11.6596872  8.6229643  9.2825456 11.3581681
33 10.4402666  7.3318241  8.4281324 10.5825274  7.3550843  8.2680060 10.3116577
34 10.9867677  7.8720612  9.0618882 11.1705929  7.9868311  8.8335439 10.8908281
35 11.5616061  8.4605982  9.4559174 11.6689315  8.3897401  9.3782737 11.4072634
36  9.3448950  6.2366355  7.6617339  9.6127143  6.5924089  7.2390939  9.3135212
37  9.0525435  6.0226179  6.7583693  9.0536065  5.7053780  6.8471070  8.8154778
38  8.4882651  5.4121600  6.3681712  8.5652821  5.2969376  6.2928456  8.3073474
39  9.4779225  6.4503097  7.1600128  9.4705054  6.1120637  7.2722290  9.2351676
40  8.4604610  5.3453104  6.6782928  8.6790433  5.6064849  6.3217285  8.3889822
```

41	9.0374691	5.9343423	7.4154496	9.3253067	6.3508284	6.9457583	9.0217700
42	10.3645565	7.3593711	7.9635320	10.3198229	6.9334077	8.1587430	10.0954303
43	10.3100888	7.2209622	8.1711358	10.3943776	7.1052927	8.1188004	10.1370528
44	9.1568434	6.0595757	7.1071343	9.2718855	6.0338826	6.9729453	9.0058710
45	10.4176250	7.3046522	8.6306043	10.6543738	7.5557980	8.2928797	10.3626536
46	8.6459692	5.5657699	7.1834015	8.9880664	6.1344218	6.5986357	8.6737008
47	10.2420838	7.1572441	8.0848296	10.3174181	7.0203563	8.0484988	10.0621925
48	10.1170063	7.0250972	8.5332554	10.4305789	7.4676693	8.0465072	10.1230540
49	10.9458877	7.8392720	9.2259173	11.2135994	8.1523579	8.8419336	10.9158938
50	10.1189835	7.0637997	7.8548175	10.1424933	6.8014638	7.9160561	9.9000169
51	9.4945905	6.4028123	7.9334752	9.8105099	6.8714714	7.4254486	9.5019654
52	9.8831659	6.7683199	7.9999838	10.0759842	6.9243103	7.7329183	9.7931863
53	8.8101179	5.7162602	6.7567143	8.9200663	5.6830805	6.6240387	8.6547722
54	10.8532820	7.7382832	8.9470493	11.0440921	7.8716552	8.7033651	10.7626651
55	8.7785335	5.6633665	6.9733568	8.9918913	5.8999693	6.6374374	8.7033651
56	8.1054686	5.0823100	5.8433212	8.1129872	4.7824007	5.8999693	7.8716552
57	11.1526377	8.0392644	9.1869582	11.3200369	8.1129872	8.9918913	11.0440921
58	9.1782553	6.1411004	6.8976472	9.1869582	5.8433212	6.9733568	8.9470493
59	7.7725197	4.6618882	6.1411004	8.0392644	5.0823100	5.6633665	7.7382832
60	10.8770943	7.7725197	9.1782553	11.1526377	8.1054686	8.7785335	10.8532820

	8	9	10	11	12	13	14
2							
3							
4							
5							
6							
7							
8							
9	1.3193212						
10	1.9054761	1.1560831					
11	1.4646550	1.5359576	2.6581939				
12	2.6323877	1.3155977	1.5017932	2.5182366			
13	2.2625933	1.1598038	0.6225280	2.6792627	0.9253192		
14	1.4477237	1.1124772	2.2642892	0.4960563	2.0222002	2.2220197	
15	1.8736385	1.7369288	0.8668532	3.0288272	2.3667196	1.4809747	2.7359278
16	2.0074433	0.6919008	1.1510005	2.0233780	0.6249459	0.7822362	1.5375881
17	0.3523712	1.0199357	1.7803597	1.2122341	2.3340154	2.0541827	1.1220147
18	1.5058951	1.0612139	2.2169943	0.5994258	1.9211665	2.1489998	0.1100819
19	1.9855947	2.1917284	3.3173340	0.6592036	3.0935133	3.3259030	1.1081995
20	1.4915076	1.1540012	0.5318838	2.4895056	1.9103016	1.1183890	2.1694521
21	0.9051520	1.4288348	1.3565811	2.2616303	2.5518558	1.8936446	2.0983928
22	1.1394108	1.8178435	2.7960066	0.7111613	3.0031339	2.9674608	1.0881462
23	0.4252711	1.7444153	2.2679678	1.6673138	3.0576383	2.6648273	1.7561584

24	0.8928690	1.8799985	2.7244687	1.0964684	3.1463612	2.9829006	1.3857266
25	1.4490021	0.8338208	0.4600384	2.2545810	1.6031938	0.8991398	1.8963853
26	2.7544313	1.8361806	2.8113700	1.6947234	1.7342960	2.4510054	1.3735004
27	2.3132023	1.1039537	1.8994411	1.7957227	0.8479615	1.5030892	1.3134068
28	1.6727316	0.7090012	1.8139929	1.1623904	1.3558871	1.6424044	0.6663356
29	3.1174110	1.7997184	1.8852884	2.9283622	0.4853305	1.2696793	2.4345947
30	0.6457296	0.8906231	1.8291780	0.9607184	2.1716095	2.0113153	0.8206781
31	7.1468516	8.3359328	8.1505435	8.3381994	9.5113381	8.7642333	8.5269754
32	9.3229785	10.3884586	9.9843829	10.6356115	11.4414274	10.6065113	10.7563895
33	8.1752964	9.3453764	9.1122866	9.3844695	10.4990623	9.7302553	9.5656085
34	8.7777796	9.9217103	9.6380023	10.0157910	11.0489047	10.2588330	10.1821126
35	9.2496433	10.4448955	10.2467267	10.4141641	11.6181192	10.8627094	10.6185264
36	7.2748479	8.3433346	7.9687160	8.5969087	9.4125595	8.5912387	8.7100088
37	6.6232251	7.8699579	7.8138277	7.7153024	9.0995116	8.4119721	7.9474955
38	6.1445317	7.3479980	7.2064939	7.3254220	8.5410094	7.8148213	7.5174690
39	7.0401611	8.2916630	8.2414099	8.1157405	9.5244305	8.8392468	8.3555049
40	6.3165807	7.4184266	7.1066798	7.6181038	8.5240764	7.7277690	7.7441712
41	7.0043988	8.0524334	7.6555276	8.3440139	9.1064325	8.2779994	8.4442611
42	7.8925734	9.1610687	9.1468970	8.9134797	10.4084183	9.7402782	9.1759468
43	7.9718362	9.1773553	9.0126349	9.1293880	10.3643416	9.6250424	9.3355009
44	6.8594189	8.0419456	7.8489824	8.0633078	9.2129580	8.4631007	8.2453963
45	8.2903156	9.3920098	9.0494441	9.5764146	10.4834664	9.6717747	9.7140336
46	6.7163776	7.7087316	7.2493600	8.0935710	8.7185749	7.8704788	8.1631201
47	7.8932865	9.1037595	8.9499520	9.0430964	10.2956034	9.5612455	9.2526621
48	8.1202239	9.1549981	8.7248256	9.4623927	10.1882266	9.3465041	9.5612455
49	8.8656464	9.9455549	9.5657393	10.1683165	11.0141444	10.1882266	10.2956034
50	7.7125729	8.9503640	8.8581803	8.8116826	10.1683165	9.4623927	9.0430964
51	7.5072067	8.5342615	8.1030339	8.8581803	9.5657393	8.7248256	8.9499520
52	7.6912969	8.8234965	8.5342615	8.9503640	9.9455549	9.1549981	9.1037595
53	6.5071075	7.6912969	7.5072067	7.7125729	8.8656464	8.1202239	7.8932865
54	8.6547722	9.7931863	9.5019654	9.9000169	10.9158938	10.1230540	10.0621925
55	6.6240387	7.7329183	7.4254486	7.9160561	8.8419336	8.0465072	8.0484988
56	5.6830805	6.9243103	6.8714714	6.8014638	8.1523579	7.4676693	7.0203563
57	8.9200663	10.0759842	9.8105099	10.1424933	11.2135994	10.4305789	10.3174181
58	6.7567143	7.9999838	7.9334752	7.8548175	9.2259173	8.5332554	8.0848296
59	5.7162602	6.7683199	6.4028123	7.0637997	7.8392720	7.0250972	7.1572441
60	8.8101179	9.8831659	9.4945905	10.1189835	10.9458877	10.1170063	10.2420838

	15	16	17	18	19	20	21
2							
3							
4							
5							
6							

7							
8							
9							
10							
11							
12							
13							
14							
15							
16	1.9612493						
17	1.9104876	1.7116345					
18	2.7191537	1.4495146	1.1711503				
19	3.6649545	2.6427459	1.7946703	1.1932513			
20	0.5830018	1.4331505	1.4443247	2.1470904	3.1370291		
21	1.0215341	1.9758144	1.0528595	2.1189127	2.8442912	0.8325280	
22	2.9619612	2.4360001	1.0562455	1.1970222	0.9299553	2.4979500	2.0402835
23	2.1114734	2.4326925	0.7523634	1.8297127	2.0860389	1.8124951	1.0918233
24	2.7661949	2.5467635	0.9454491	1.4874836	1.3506537	2.3599104	1.7873523
25	0.9149742	1.0964027	1.3218634	1.8633324	2.9109359	0.3441369	0.9972231
26	3.5428603	1.6865352	2.4054575	1.2848059	1.9608181	2.9673656	3.2107616
27	2.6831157	0.7522561	1.9694771	1.2040608	2.3113707	2.1283613	2.5317012
28	2.4457664	0.9019434	1.3212662	0.5660989	1.7521853	1.8629238	2.0525495
29	2.7495597	1.1100915	2.8167199	2.3290608	3.4705431	2.3374884	3.0175728
30	2.0844193	1.5626090	0.3033151	0.8678632	1.5813418	1.5706219	1.3102028
31	7.3563629	8.9359562	7.4992166	8.6066503	8.5037352	7.6488481	6.9623038
32	9.1326153	10.9175937	9.6691823	10.8238640	10.8825502	9.5319519	8.9642099
33	8.3003107	9.9329289	8.5274511	9.6437648	9.5559994	8.6200557	7.9571482
34	8.8113046	10.4938850	9.1291379	10.2576508	10.2052322	9.1563678	8.5195968
35	9.4420480	11.0445232	9.6019810	10.7002659	10.5494178	9.7493850	9.0703008
36	7.1252275	8.8803809	7.6206652	8.7766493	8.8606678	7.5058025	6.9211415
37	7.0686434	8.5027673	6.9733460	8.0341706	7.8215634	7.2945766	6.5533134
38	6.4321992	7.9581350	6.4968590	7.5979932	7.4920681	6.6968646	5.9892671
39	7.4961066	8.9266022	7.3896999	8.4432186	8.2083723	7.7219593	6.9792136
40	6.2799244	7.9760981	6.6648433	7.8134709	7.8704603	6.6273102	6.0057240
41	6.8084333	8.5802760	7.3481144	8.5086691	8.6246634	7.1978229	6.6273102
42	8.4117652	9.8052760	8.2395001	9.2668389	8.9671376	8.6246634	7.8704603
43	8.2219555	9.7847346	8.3240592	9.4177717	9.2668389	8.5086691	7.8134709
44	7.0536305	8.6391623	7.2117239	8.3240592	8.2395001	7.3481144	6.6648433
45	8.2103791	9.9428976	8.6391623	9.7847346	9.8052760	8.5802760	7.9760981
46	6.3928876	8.2103791	7.0536305	8.2219555	8.4117652	6.8084333	6.2799244
47	8.1631201	9.7140336	8.2453963	9.3355009	9.1759468	8.4442611	7.7441712
48	7.8704788	9.6717747	8.4631007	9.6250424	9.7402782	8.2779994	7.7277690
49	8.7185749	10.4834664	9.2129580	10.3643416	10.4084183	9.1064325	8.5240764

50	8.0935710	9.5764146	8.0633078	9.1293880	8.9134797	8.3440139	7.6181038
51	7.2493600	9.0494441	7.8489824	9.0126349	9.1468970	7.6555276	7.1066798
52	7.7087316	9.3920098	8.0419456	9.1773553	9.1610687	8.0524334	7.4184266
53	6.7163776	8.2903156	6.8594189	7.9718362	7.8925734	7.0043988	6.3165807
54	8.6737008	10.3626536	9.0058710	10.1370528	10.0954303	9.0217700	8.3889822
55	6.5986357	8.2928797	6.9729453	8.1188004	8.1587430	6.9457583	6.3217285
56	6.1344218	7.5557980	6.0338826	7.1052927	6.9334077	6.3508284	5.6064849
57	8.9880664	10.6543738	9.2718855	10.3943776	10.3198229	9.3253067	8.6790433
58	7.1834015	8.6306043	7.1071343	8.1711358	7.9635320	7.4154496	6.6782928
59	5.5657699	7.3046522	6.0595757	7.2209622	7.3593711	5.9343423	5.3453104
60	8.6459692	10.4176250	9.1568434	10.3100888	10.3645565	9.0374691	8.4604610

	22	23	24	25	26	27	28
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23	1.1668296						
24	0.4277071	0.7963557					
25	2.3469872	1.8212373	2.2650597				
26	2.3992630	3.1051554	2.7449566	2.6282026			
27	2.3805002	2.7213926	2.6095065	1.7842801	0.9488669		
28	1.6933195	2.0639810	1.9117995	1.5386491	1.1584988	0.6977079	
29	3.4512174	3.5426782	3.6137156	2.0489203	1.9008847	1.1621329	1.7736609
30	0.9702490	1.0131420	0.9893915	1.3861601	2.1091321	1.7258767	1.0532898
31	7.6608523	6.7810231	7.2463624	7.9201931	9.8860645	9.4213672	8.8162838
32	9.9954899	9.0031696	9.5700008	9.8418061	12.0734487	11.4922949	10.9499011

33	8.7099244	7.8157806	8.2941296	8.9012262	10.9206270	10.4365443	9.8411529
34	9.3485286	8.4279755	8.9297155	9.4468860	11.5293255	11.0184915	10.4365443
35	9.7279150	8.8786926	9.3189758	10.0246672	11.9821300	11.5293255	10.9206270
36	7.9650329	6.9582771	7.5383989	7.8105064	10.0246672	9.4468860	8.9012262
37	7.0198114	6.2333080	6.6189002	7.5383989	9.3189758	8.9297155	8.2941296
38	6.6477125	5.7743650	6.2333080	6.9582771	8.8786926	8.4279755	7.8157806
39	7.4173680	6.6477125	7.0198114	7.9650329	9.7279150	9.3485286	8.7099244
40	6.9792136	5.9892671	6.5533134	6.9211415	9.0703008	8.5195968	7.9571482
41	7.7219593	6.6968646	7.2945766	7.5058025	9.7493850	9.1563678	8.6200557
42	8.2083723	7.4920681	7.8215634	8.8606678	10.5494178	10.2052322	9.5559994
43	8.4432186	7.5979932	8.0341706	8.7766493	10.7002659	10.2576508	9.6437648
44	7.3896999	6.4968590	6.9733460	7.6206652	9.6019810	9.1291379	8.5274511
45	8.9266022	7.9581350	8.5027673	8.8803809	11.0445232	10.4938850	9.9329289
46	7.4961066	6.4321992	7.0686434	7.1252275	9.4420480	8.8113046	8.3003107
47	8.3555049	7.5174690	7.9474955	8.7100088	10.6185264	10.1821126	9.5656085
48	8.8392468	7.8148213	8.4119721	8.5912387	10.8627094	10.2588330	9.7302553
49	9.5244305	8.5410094	9.0995116	9.4125595	11.6181192	11.0489047	10.4990623
50	8.1157405	7.3254220	7.7153024	8.5969087	10.4141641	10.0157910	9.3844695
51	8.2414099	7.2064939	7.8138277	7.9687160	10.2467267	9.6380023	9.1122866
52	8.2916630	7.3479980	7.8699579	8.3433346	10.4448955	9.9217103	9.3453764
53	7.0401611	6.1445317	6.6232251	7.2748479	9.2496433	8.7777796	8.1752964
54	9.2351676	8.3073474	8.8154778	9.3135212	11.4072634	10.8908281	10.3116577
55	7.2722290	6.2928456	6.8471070	7.2390939	9.3782737	8.8335439	8.2680060
56	6.1120637	5.2969376	5.7053780	6.5924089	8.3897401	7.9868311	7.3550843
57	9.4705054	8.5652821	9.0536065	9.6127143	11.6689315	11.1705929	10.5825274
58	7.1600128	6.3681712	6.7583693	7.6617339	9.4559174	9.0618882	8.4281324
59	6.4503097	5.4121600	6.0226179	6.2366355	8.4605982	7.8720612	7.3318241
60	9.4779225	8.4882651	9.0525435	9.3448950	11.5616061	10.9867677	10.4402666

	29	30	31	32	33	34	35
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							

16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30	2.6466349						
31	9.9684525	7.7854131					
32	11.8639007	9.9684525	2.6466349				
33	10.9499011	8.8162838	1.0532898	1.7736609			
34	11.4922949	9.4213672	1.7258767	1.1621329	0.6977079		
35	12.0734487	9.8860645	2.1091321	1.9008847	1.1584988	0.9488669	
36	9.8418061	7.9201931	1.3861601	2.0489203	1.5386491	1.7842801	2.6282026
37	9.5700008	7.2463624	0.9893915	3.6137156	1.9117995	2.6095065	2.7449566
38	9.0031696	6.7810231	1.0131420	3.5426782	2.0639810	2.7213926	3.1051554
39	9.9954899	7.6608523	0.9702490	3.4512174	1.6933195	2.3805002	2.3992630
40	8.9642099	6.9623038	1.3102028	3.0175728	2.0525495	2.5317012	3.2107616
41	9.5319519	7.6488481	1.5706219	2.3374884	1.8629238	2.1283613	2.9673656
42	10.8825502	8.5037352	1.5813418	3.4705431	1.7521853	2.3113707	1.9608181
43	10.8238640	8.6066503	0.8678632	2.3290608	0.5660989	1.2040608	1.2848059
44	9.6691823	7.4992166	0.3033151	2.8167199	1.3212662	1.9694771	2.4054575
45	10.9175937	8.9359562	1.5626090	1.1100915	0.9019434	0.7522561	1.6865352
46	9.1326153	7.3563629	2.0844193	2.7495597	2.4457664	2.6831157	3.5428603
47	10.7563895	8.5269754	0.8206781	2.4345947	0.6663356	1.3134068	1.3735004
48	10.6065113	8.7642333	2.0113153	1.2696793	1.6424044	1.5030892	2.4510054
49	11.4414274	9.5113381	2.1716095	0.4853305	1.3558871	0.8479615	1.7342960
50	10.6356115	8.3381994	0.9607184	2.9283622	1.1623904	1.7957227	1.6947234
51	9.9843829	8.1505435	1.8291780	1.8852884	1.8139929	1.8994411	2.8113700
52	10.3884586	8.3359328	0.8906231	1.7997184	0.7090012	1.1039537	1.8361806
53	9.3229785	7.1468516	0.6457296	3.1174110	1.6727316	2.3132023	2.7544313
54	11.3581681	9.2988105	1.6358735	1.1492581	0.6428473	0.1476574	1.0798172
55	9.2825456	7.2696756	1.0830471	2.7239226	1.7435270	2.2129053	2.9019614
56	8.6229643	6.3094335	1.6049877	4.1973308	2.6519669	3.3308601	3.6172353
57	11.6596872	9.5624990	1.8194018	1.2839437	0.7665983	0.2289425	0.7251267
58	9.6955106	7.3810986	0.8829398	3.4885416	1.7755605	2.4730852	2.6011281

59	8.2713940	6.3604746	2.0398313	3.6204669	2.8066510	3.2561284	3.9651446
60	11.3715942	9.4557146	2.1743655	0.5180611	1.3985773	0.9273043	1.8275606

	36	37	38	39	40	41	42
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32							
33							
34							
35							
36							
37	2.2650597						
38	1.8212373	0.7963557					
39	2.3469872	0.4277071	1.1668296				
40	0.9972231	1.7873523	1.0918233	2.0402835			
41	0.3441369	2.3599104	1.8124951	2.4979500	0.8325280		

42	2.9109359	1.3506537	2.0860389	0.9299553	2.8442912	3.1370291	
43	1.8633324	1.4874836	1.8297127	1.1970222	2.1189127	2.1470904	1.1932513
44	1.3218634	0.9454491	0.7523634	1.0562455	1.0528595	1.4443247	1.7946703
45	1.0964027	2.5467635	2.4326925	2.4360001	1.9758144	1.4331505	2.6427459
46	0.9149742	2.7661949	2.1114734	2.9619612	1.0215341	0.5830018	3.6649545
47	1.8963853	1.3857266	1.7561584	1.0881462	2.0983928	2.1694521	1.1081995
48	0.8991398	2.9829006	2.6648273	2.9674608	1.8936446	1.1183890	3.3259030
49	1.6031938	3.1463612	3.0576383	3.0031339	2.5518558	1.9103016	3.0935133
50	2.2545810	1.0964684	1.6673138	0.7111613	2.2616303	2.4895056	0.6592036
51	0.4600384	2.7244687	2.2679678	2.7960066	1.3565811	0.5318838	3.3173340
52	0.8338208	1.8799985	1.7444153	1.8178435	1.4288348	1.1540012	2.1917284
53	1.4490021	0.8928690	0.4252711	1.1394108	0.9051520	1.4915076	1.9855947
54	1.6370132	2.5447287	2.6200123	2.3361663	2.3952927	1.9810545	2.3253849
55	0.7501555	1.7176551	1.1282369	1.9112819	0.3188015	0.6844408	2.6544816
56	2.4712110	0.9471566	0.6649968	1.3727696	1.6464793	2.4310798	2.2776519
57	1.9972620	2.6536007	2.8280792	2.3928213	2.7093865	2.3413083	2.2339240
58	2.1972819	0.1439333	0.8509234	0.3215511	1.7816366	2.3110279	1.2512755
59	1.5760438	2.3270913	1.5401399	2.6509537	0.7578453	1.2887877	3.5151884
60	1.5399504	3.1550185	3.0401204	3.0251617	2.5009148	1.8396872	3.1444604

	43	44	45	46	47	48	49
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							

25							
26							
27							
28							
29							
30							
31							
32							
33							
34							
35							
36							
37							
38							
39							
40							
41							
42							
43							
44	1.1711503						
45	1.4495146	1.7116345					
46	2.7191537	1.9104876	1.9612493				
47	0.1100819	1.1220147	1.5375881	2.7359278			
48	2.1489998	2.0541827	0.7822362	1.4809747	2.2220197		
49	1.9211665	2.3340154	0.6249459	2.3667196	2.0222002	0.9253192	
50	0.5994258	1.2122341	2.0233780	3.0288272	0.4960563	2.6792627	2.5182366
51	2.2169943	1.7803597	1.1510005	0.8668532	2.2642892	0.6225280	1.5017932
52	1.0612139	1.0199357	0.6919008	1.7369288	1.1124772	1.1598038	1.3155977
53	1.5058951	0.3523712	2.0074433	1.8736385	1.4477237	2.2625933	2.6323877
54	1.1830121	1.8677111	0.6103383	2.5354641	1.2901224	1.3721986	0.7843407
55	1.8452779	0.8686776	1.6662812	1.0533764	1.8342211	1.6460382	2.2525755
56	2.3335866	1.3850579	3.0885535	2.6566485	2.2439841	3.3267638	3.7130698
57	1.1973153	2.0810768	0.9806703	2.9013124	1.3071501	1.7303771	1.0316446
58	1.3442448	0.8760315	2.4299650	2.7395937	1.2421237	2.8892144	3.0242135
59	2.8701389	1.7601726	2.6467464	1.0839130	2.8445151	2.4040869	3.1792256
60	1.9646732	2.3249043	0.6140655	2.2840189	2.0635411	0.8312701	0.1028124
	50	51	52	53	54	55	56
2							
3							
4							
5							
6							
7							

```
51  2.6581939
52  1.5359576  1.1560831
53  1.4646550  1.9054761  1.3193212
54  1.7813324  1.7541681  0.9706492  2.2070938
55  2.0437594  1.1635750  1.1102340  0.8290773  2.0765471
56  2.0295105  2.9094954  2.3976192  1.0852688  3.2390708  1.7512147
57  1.7668044  2.1259026  1.2818552  2.4304086  0.3717598  2.3912492  3.4181296
58  0.9582688  2.6548368  1.7715459  0.8771866  2.4109383  1.6867137  1.0756810
59  2.9695433  1.7945937  2.1624060  1.5297785  3.1155553  1.0633575  1.8801216
60  2.5589692  1.4176684  1.3049768  2.6152925  0.8527644  2.2060413  3.6986481
           57         58         59
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
```

```
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58  2.5136786
59  3.4434031  2.3573405
60  1.1190258  3.0354590  3.1151670
```

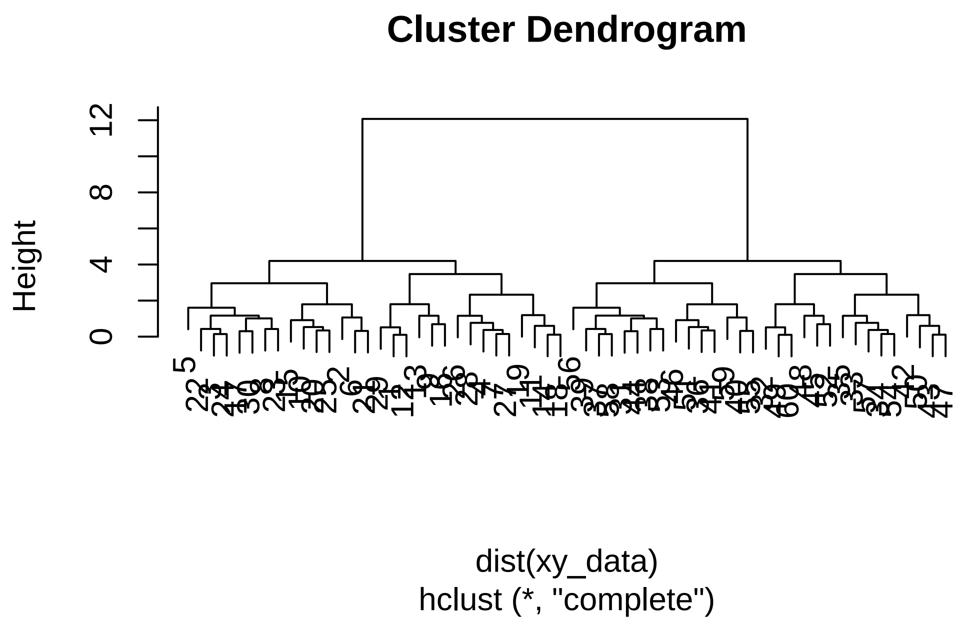
Now, we'll assign the resulting distance matrix to a variable, hc, and apply heirarchical clustering

```r
hc = hclust(dist(xy_data))
hc
```

```

Call:
hclust(d = dist(xy_data))

Cluster method   : complete
Distance         : euclidean
Number of objects: 60
```

The resulting information isn't very useful on its own; let's plot it:

```r
plot(hc)
```

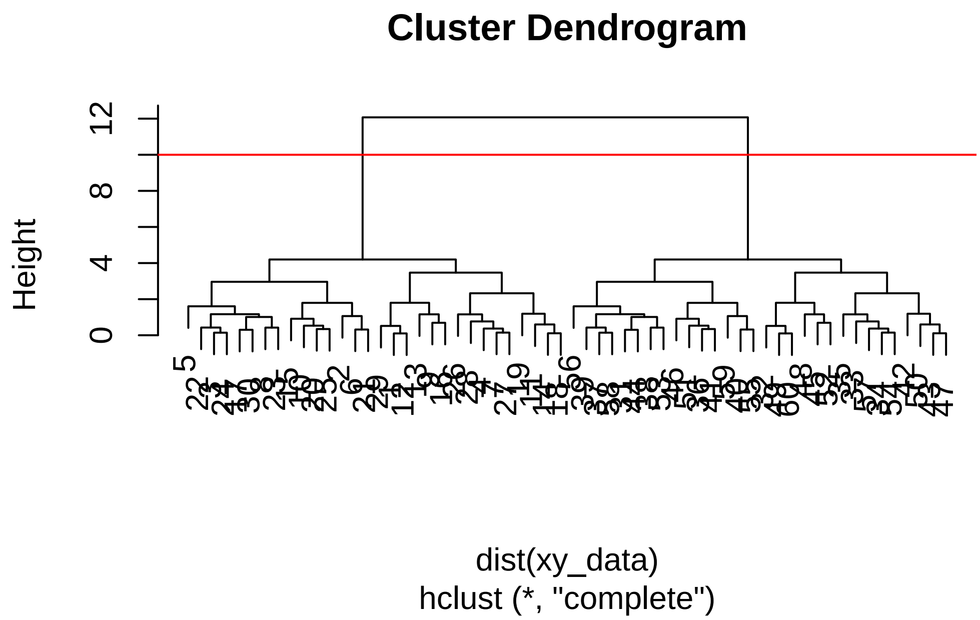


The vertical height of the dendrogram shows the similarity of two clusters through the vertical height that separates the crossbar between them. The two major clusters on the left and right sides are split by the first and last 30 data points; the 1st 30 make up the 1st cluster & points 30-60 create the second cluster.

The biggest "goalpost" indicates the optimal separation of clusters (ex. the top-most crossbar separates two clusters with a very larger vertical distance; the left and right groups are good clusterings).

We can create an upper limit or cutoff line, we can insert an `abline()` at our desired y-value/height.

```r
plot(hc)
abline(h=10,col="red")
```



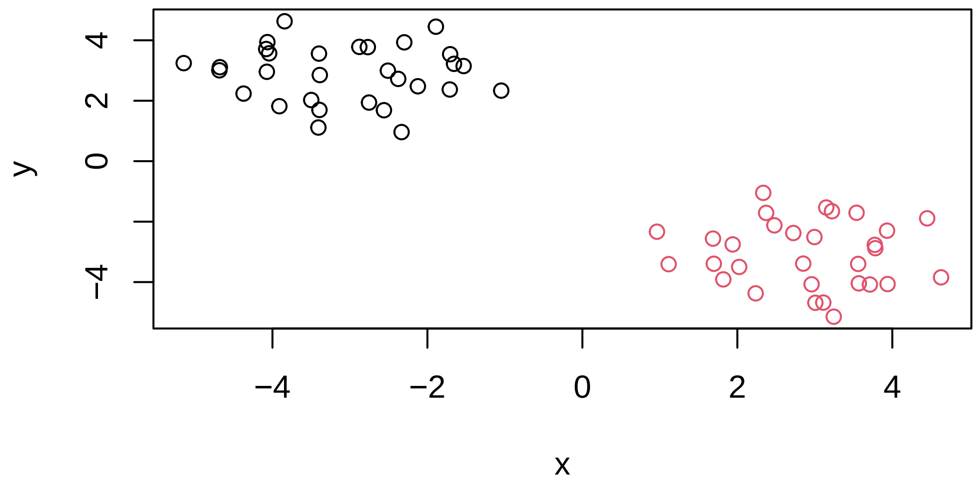
Now, we can cut the tree at a y-value of 10. Everything clustered under that line gets assigned to a cluster (cluster1 assigned a value of 1, cluster2 assigned a value of 2).

```
grps = cutree(hc, h=10)
grps
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Now, we can plot the original data (xy_data) with the colors split based off of the clustering we did above (grps).

```
plot(xy_data, col=grps)
```




```
# alternatively, we can use: plot(xy_data, col=cutree(hc, h=10))
```

Principal Component Analysis

The goal of PCA is to reduce the dimensionality of a dataset down to some smaller subset of new variables (called PCs) which are useful bases for further analysis - like visualization, clustering, etc.

In this part of the module, we'll look at a way of reading the food consumption of individuals across 4 countries in the UK, across 17 different categories (potato, cheese, fruit…).

First, we'll import the dataset:

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names=1) # saying row.names=1 means that we start counting the colu
x
```

```
                    England Wales Scotland N.Ireland
Cheese                  105   103      103        66
Carcass_meat            245   227      242       267
Other_meat              685   803      750       586
```

```
Fish                 147  160     122      93
Fats_and_oils        193  235     184     209
Sugars               156  175     147     139
Fresh_potatoes       720  874     566    1033
Fresh_Veg            253  265     171     143
Other_Veg            488  570     418     355
Processed_potatoes   198  203     220     187
Processed_Veg        360  365     337     334
Fresh_fruit         1102 1137     957     674
Cereals             1472 1582    1462    1494
Beverages             57   73      53      47
Soft_drinks         1374 1256    1572    1506
Alcoholic_drinks     375  475     458     135
Confectionery         54   64      62      41
```

To look at the first few lines of the datset:

```r
head(x)
```

```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars             156   175      147       139
```

> Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

```r
dim(x) # or separately,
```

```
[1] 17  4
```

```r
ncol(x)
```

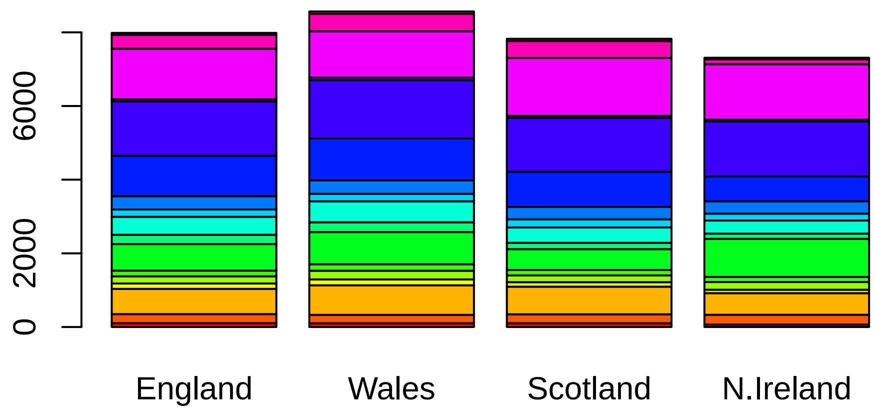
```
[1] 4
```

```r
nrow(x)
```

```
[1] 17
```

We can make a barplot which compares the consumption of each of the 17 food categories across the 4 countries; not very readable without PCA since there are too many dimensions to work with.

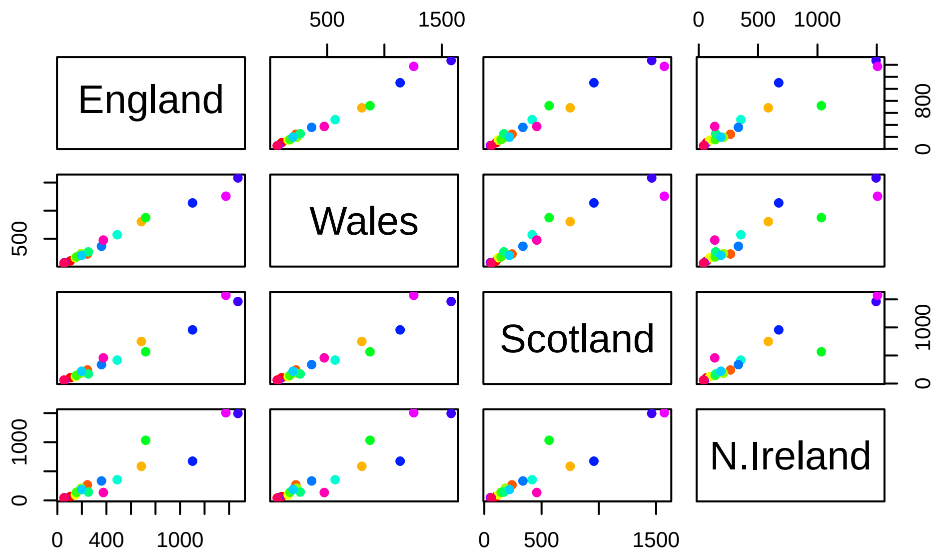
```
barplot(as.matrix(x), col=rainbow(nrow(x)))
```




```
#using rainbow(nrow(x)) gives us one color of the rainbow for each of the 17 rows in x
```

We can also plot each of these relationships pairwise (ex. england is the y-axis for the top, rightmost graph and n.ireland is the x-axis); below and above the diagnoal are the same. For each graph, we can draw a diagonal line across the plot; any departure from the straight line means that the two countries being compared differ for that data point (ex. the scotland v. north ireland plot shows that food consumption associated with the green dot is different). Pairs plots may be useful for small datasets but are less readable for large datasets.

```
pairs(x, col=rainbow(nrow(x)), pch=16)
```



PCA will help make this data comparison more readable:

The main function to do this in base R is `prcomp()`

```
t(x) #this transposes the matrix so that the food act as columns and the countries are the
```

```
         Cheese Carcass_meat  Other_meat  Fish Fats_and_oils  Sugars
England     105           245         685   147           193     156
Wales       103           227         803   160           235     175
Scotland    103           242         750   122           184     147
N.Ireland    66           267         586    93           209     139
         Fresh_potatoes  Fresh_Veg  Other_Veg  Processed_potatoes
England             720        253        488                 198
Wales               874        265        570                 203
Scotland            566        171        418                 220
N.Ireland          1033        143        355                 187
         Processed_Veg  Fresh_fruit  Cereals  Beverages Soft_drinks
England            360         1102     1472         57        1374
Wales              365         1137     1582         73        1256
Scotland           337          957     1462         53        1572
N.Ireland          334          674     1494         47        1506
         Alcoholic_drinks  Confectionery
```

```
England                 375            54
Wales                   475            64
Scotland                458            62
N.Ireland               135            41
```

```
pca = prcomp(t(x)) #apple PCA to the transposed food/country matrix
summary(pca) #gives a summary of the pca analysis
```

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

Note how PC1 always has the most variable (largest standard deviation); proportion of variance tells us that PC1 (a 1 dimensional line across the data) captures the broadest/largest amount of the variance in the data (67.44%). PC2 adds a second dimension and captures an additional 29.05% of variance in the data. Adding a third dimension, PC3, captures an additional 3.5% of variance in the data. Together, all 3 dimensions cover ~100% of the data's variance. PCA1 is the most important since it captures the greatest amount of variance in the data/

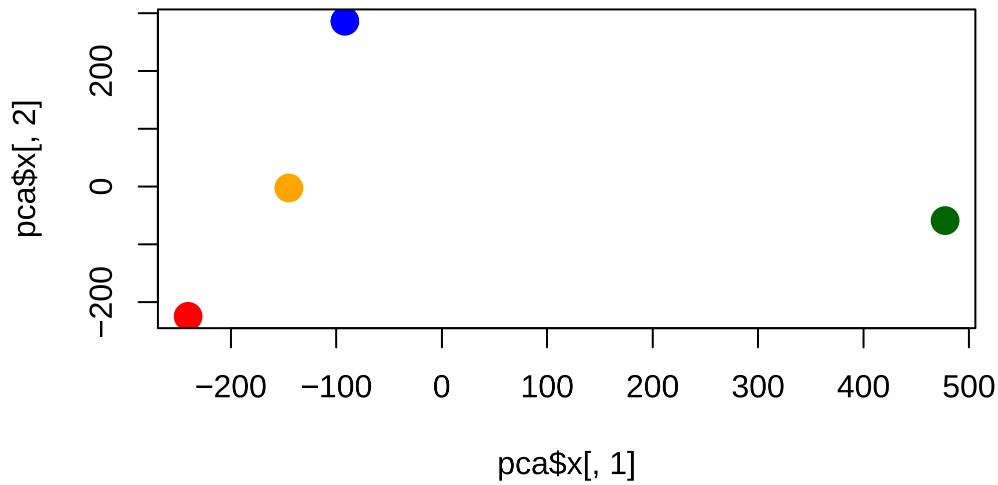
```
pca$x
```

```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

A major PCA result vizualization is called a "PCA plot" (a.k.a. a score plot/biplot/PC1 v PC2 plot/ordination plot - depending on the field of data analysis you're working in).

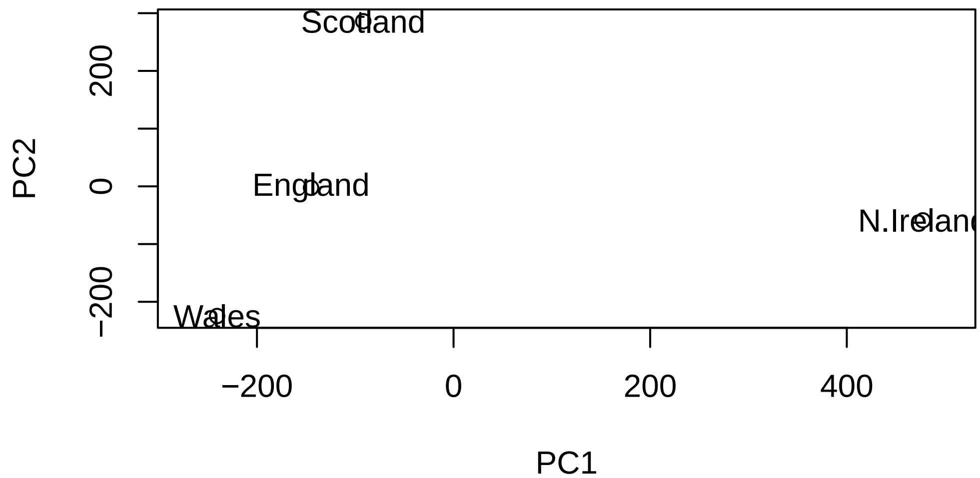
Now, we'll plot PCA1 vs PCA2. By doing so, we see that there's a large outliar in the data set (n.ireland, dark green

```
mycols = c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=mycols, pch=16, cex=2)
```



We can also plot this using the names of each country

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



Now, we see that ireland is very different from the other countries. But how? What categories of food consumption set it aside from the rest of the UK?

Another important output from PCA is called the "loadings" vector or the "rotation" component - it tells us how much the original variables (here, the food categories) contribute to the new PCs.

```
pca$rotation
```

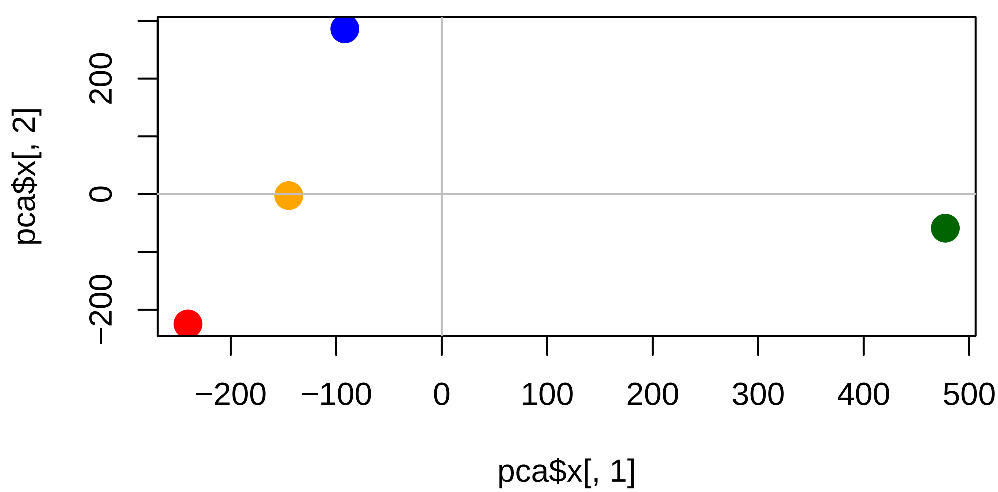
```
                        PC1          PC2         PC3          PC4
Cheese         -0.056955380  0.016012850  0.02394295 -0.694538519
Carcass_meat    0.047927628  0.013915823  0.06367111  0.489884628
Other_meat     -0.258916658 -0.015331138 -0.55384854  0.279023718
Fish           -0.084414983 -0.050754947  0.03906481 -0.008483145
Fats_and_oils  -0.005193623 -0.095388656 -0.12522257  0.076097502
Sugars         -0.037620983 -0.043021699 -0.03605745  0.034101334
Fresh_potatoes  0.401402060 -0.715017078 -0.20668248 -0.090972715
Fresh_Veg      -0.151849942 -0.144900268  0.21382237 -0.039901917
Other_Veg      -0.243593729 -0.225450923 -0.05332841  0.016719075
```

```
Processed_potatoes  -0.026886233  0.042850761 -0.07364902  0.030125166
Processed_Veg       -0.036488269 -0.045451802  0.05289191 -0.013969507
Fresh_fruit         -0.632640898 -0.177740743  0.40012865  0.184072217
Cereals             -0.047702858 -0.212599678 -0.35884921  0.191926714
Beverages           -0.026187756 -0.030560542 -0.04135860  0.004831876
Soft_drinks          0.232244140  0.555124311 -0.16942648  0.103508492
Alcoholic_drinks    -0.463968168  0.113536523 -0.49858320 -0.316290619
Confectionery       -0.029650201  0.005949921 -0.05232164  0.001847469
```

Some values are positive and negative; positive values for a certain category means that that food is consumed more than the other countries; negative values say the opposite (i.e that country eats less of a particular food). Ireland, for example, had a positive 0.40 value for fresh potatoes

Here, we've added in axes which shows PCA1 vs PCA2 ::: {.cell}

```r
mycols = c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=mycols, pch=16, cex=2)
abline(h=0, col="grey")
abline(v=0, col="grey")
```



:::

We'll leave it off here for this class, but the main takeaway for this section is that PCS is a useful method for gaining some insight into data with many dimensions which are difficult to examine in other ways.