Class 07: Machine Learning

Jessica Diaz-Vigil

4/26/23

Example of K-means clustering

First step is to make up some data with a known structure, so we know the answer it should be

```
tmp <- c(rnorm(30, mean = -3), rnorm(30, mean = 3))
tmp
```

```
 [1] -3.587131 -1.603602 -2.402392 -1.660325 -3.247032 -3.285262 -2.796644
 [8] -3.377857 -3.971861 -2.107170 -3.697691 -1.938304 -3.475901 -2.161622
[15] -3.280608 -3.037849 -2.952950 -1.363480 -3.901889 -3.337814 -4.524870
[22] -3.010901 -3.537338 -2.814610 -3.001174 -4.817944 -1.383292 -3.849647
[29] -3.230579 -4.388052  3.986258  1.903911  2.738712  4.064392  5.420767
[36]  1.241818  3.003695  4.062444  3.683812  4.471315  3.480921  2.668516
[43]  3.495446  1.331608  2.902785  1.836697  3.618256  2.925723  4.196351
[50]  2.986549  1.709973  0.949041  4.294535  3.987710  4.027427  1.853112
[57]  3.123365  5.147928  3.472910  2.922214
```

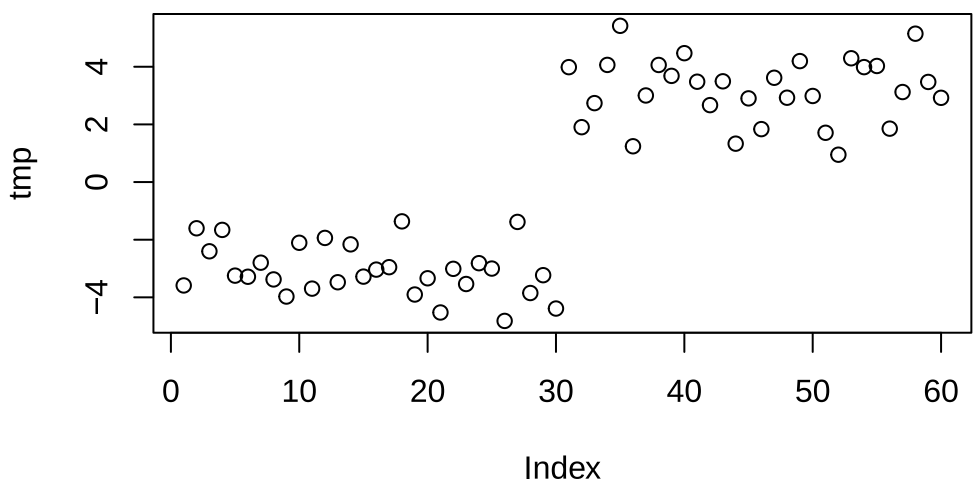
```
x <- cbind( x = tmp, y= rev(tmp) )
x
```

```
             x        y
 [1,] -3.587131  2.922214
 [2,] -1.603602  3.472910
 [3,] -2.402392  5.147928
 [4,] -1.660325  3.123365
 [5,] -3.247032  1.853112
 [6,] -3.285262  4.027427
```

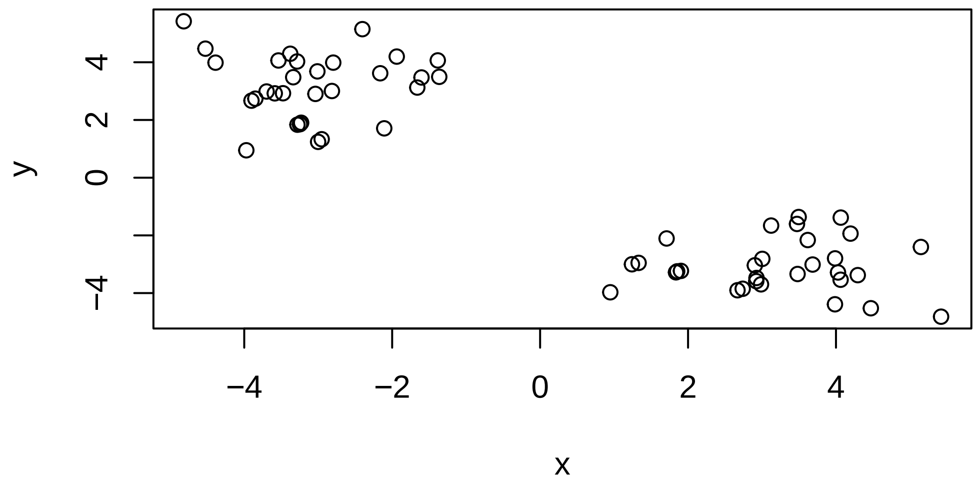
```
 [7,] -2.796644  3.987710
 [8,] -3.377857  4.294535
 [9,] -3.971861  0.949041
[10,] -2.107170  1.709973
[11,] -3.697691  2.986549
[12,] -1.938304  4.196351
[13,] -3.475901  2.925723
[14,] -2.161622  3.618256
[15,] -3.280608  1.836697
[16,] -3.037849  2.902785
[17,] -2.952950  1.331608
[18,] -1.363480  3.495446
[19,] -3.901889  2.668516
[20,] -3.337814  3.480921
[21,] -4.524870  4.471315
[22,] -3.010901  3.683812
[23,] -3.537338  4.062444
[24,] -2.814610  3.003695
[25,] -3.001174  1.241818
[26,] -4.817944  5.420767
[27,] -1.383292  4.064392
[28,] -3.849647  2.738712
[29,] -3.230579  1.903911
[30,] -4.388052  3.986258
[31,]  3.986258 -4.388052
[32,]  1.903911 -3.230579
[33,]  2.738712 -3.849647
[34,]  4.064392 -1.383292
[35,]  5.420767 -4.817944
[36,]  1.241818 -3.001174
[37,]  3.003695 -2.814610
[38,]  4.062444 -3.537338
[39,]  3.683812 -3.010901
[40,]  4.471315 -4.524870
[41,]  3.480921 -3.337814
[42,]  2.668516 -3.901889
[43,]  3.495446 -1.363480
[44,]  1.331608 -2.952950
[45,]  2.902785 -3.037849
[46,]  1.836697 -3.280608
[47,]  3.618256 -2.161622
[48,]  2.925723 -3.475901
[49,]  4.196351 -1.938304
```

```
[50,]  2.986549 -3.697691
[51,]  1.709973 -2.107170
[52,]  0.949041 -3.971861
[53,]  4.294535 -3.377857
[54,]  3.987710 -2.796644
[55,]  4.027427 -3.285262
[56,]  1.853112 -3.247032
[57,]  3.123365 -1.660325
[58,]  5.147928 -2.402392
[59,]  3.472910 -1.603602
[60,]  2.922214 -3.587131
```

```r
plot(tmp)
```




```r
plot(x)
```



Now we have some structured data in `x`. Let's see if the k-means is able to identify the two groups.

```
k <- kmeans(x, centers = 2, nstart = 20)
k
```

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

Cluster means:
          x         y
1 -3.058193  3.183606
2  3.183606 -3.058193

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] 61.29342 61.29342
 (between_SS / total_SS =  90.5 %)

Available components:
```

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

Let's explore `k`:

```r
k$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
```

```r
k$centers
```

```
          x         y
1 -3.058193  3.183606
2  3.183606 -3.058193
```

```r
k$totss
```

```
[1] 1291.389
```

```r
k$withinss
```

```
[1] 61.29342 61.29342
```

```r
k$tot.withinss
```

```
[1] 122.5868
```

```r
k$betweenss
```

```
[1] 1168.802
```

```r
k$size
```

```
[1] 30 30
```

```
k$iter
```

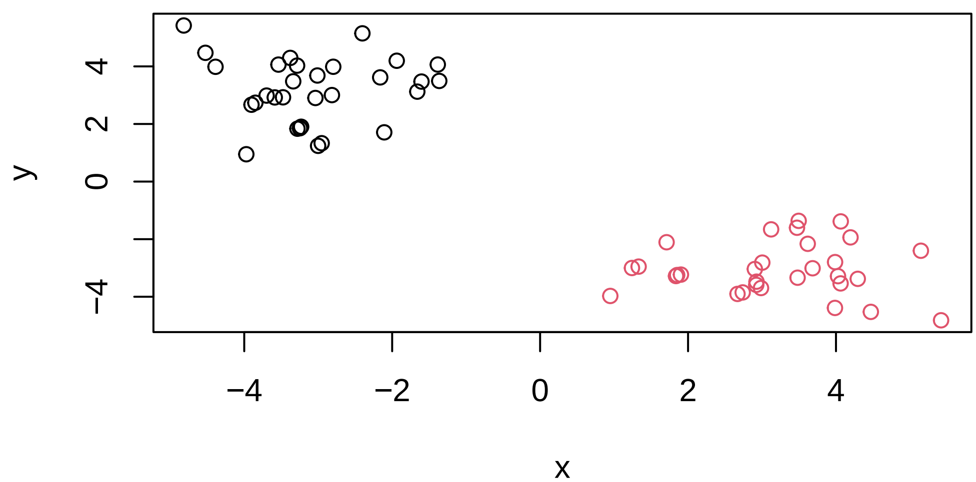
```
[1] 1
```

```
k$ifault
```

```
[1] 0
```

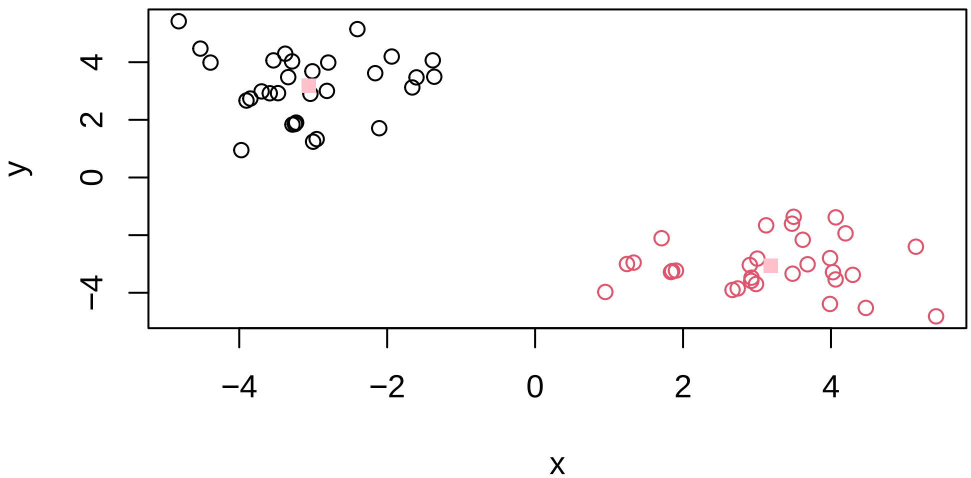
We can color the data points of what `k` gave us:

```
plot(x, col = k$cluster)
```



Now we can add the cluster centers:

```
plot(x, col = k$cluster)
points(k$centers, col = 'pink', pch = 15)
```



Example of Hierarchical Clustering

Let's use the same data as before, which we stored in **`a`**. We will use the **`hclust()`** function.

First we must look at **`dist()`** since it is required for **`hclust()`**.

```
dist(x)
```

```
            1          2          3          4          5          6
2  2.05855580
3  2.52139040 1.85573461
4  1.93727728 0.35411715 2.15627420
5  1.12189415 2.30751125 3.40135729 2.03253144
6  1.14569688 1.77072625 1.42652801 1.85950297 2.17465173
7  1.32670747 1.29937233 1.22537306 1.42769541 2.18159615 0.49023015
8  1.38818570 1.95526187 1.29607607 2.07883530 2.44492554 0.28270138
9  2.01033045 3.46100619 4.48262034 3.17346577 1.15875825 3.15402602
10 1.91306319 1.83344646 3.45060696 1.48234518 1.14881421 2.59971060
11 0.12791651 2.14982660 2.51979294 2.04195483 1.21974369 1.11960881
12 2.08375927 0.79711524 1.05871432 1.10840931 2.68394064 1.35750901
13 0.11128471 1.95061987 2.46791751 1.82630253 1.09675728 1.11807661
```

14	1.58636362	0.57663836	1.54850466	0.70442596	2.07216049	1.19582151
15	1.12796417	2.34297675	3.42571356	2.06901706	0.03737325	2.19073528
16	0.54962468	1.54340813	2.33333995	1.39507365	1.07031328	1.15153572
17	1.71236996	2.53099030	3.85582801	2.20936021	0.59870690	2.71622376
18	2.29634904	0.24117748	1.95193053	0.47598476	2.49900638	1.99405414
19	0.40427156	2.43498943	2.89758098	2.28724716	1.04581190	1.49226950
20	0.61181046	1.73423062	1.91152513	1.71517256	1.63033843	0.54902756
21	1.81082012	3.08717052	2.22771658	3.16584758	2.91339291	1.31668679
22	0.95502497	1.42301436	1.58553415	1.46224381	1.84586618	0.43971093
23	1.14131655	2.02160426	1.57047070	2.09882024	2.22832342	0.25449571
24	0.77680569	1.29873144	2.18349703	1.16047232	1.22915844	1.12673937
25	1.77962757	2.63267462	3.95173781	2.31043163	0.65888220	2.80005777
26	2.78525891	3.75847589	2.43091217	3.90494737	3.89819518	2.07135396
27	2.48223182	0.63117948	1.48748590	0.98095794	2.89193474	1.90232939
28	0.32029308	2.36299891	2.81049258	2.22285621	1.07118283	1.40688198
29	1.07892064	2.26026801	3.34806514	1.98815672	0.05339729	2.12422033
30	1.33179074	2.83137632	2.30050545	2.86095867	2.41914084	1.10355838
31	10.52597798	9.64578988	11.47823018	9.39708943	9.55367042	11.12183939
32	8.24672035	7.56567290	9.42038315	7.28535370	7.23713507	8.92222922
33	9.26684310	8.51325574	10.36278419	8.24465984	8.26744131	9.91647936
34	8.77970244	7.46383594	9.19108917	7.28576243	7.99570051	9.12651622
35	11.87654209	10.86646237	12.66966466	10.63984198	10.93771978	12.41110508
36	7.64233397	7.07178710	8.92682047	6.77734523	6.61164615	8.36036437
37	8.73785605	7.79487573	9.62433275	7.55067047	7.80123148	9.29324794
38	10.01208216	9.01374713	10.82718472	8.78151657	9.08214604	10.54582614
39	9.38447947	8.36639367	10.17881910	8.13566294	8.46730275	9.90485024
40	10.97258415	10.04336080	11.86637530	9.80267836	10.01256863	11.54583394
41	9.44167866	8.49931325	10.32575310	8.25707184	8.49770884	10.00140041
42	9.25751040	8.52283118	10.37368279	8.25185182	8.25310520	9.91572178
43	8.27828882	7.02786991	8.78538111	6.83474573	7.47044010	8.66255670
44	7.66234575	7.06449796	8.92003238	6.77298058	6.63793462	8.36905978
45	8.81143302	7.91817543	9.75458062	7.66697639	7.85759147	9.39202099
46	8.23971471	7.57929094	9.43451542	7.29657617	7.22491339	8.92422772
47	8.81833193	7.68217046	9.46983167	7.46957135	7.95300332	9.27161751
48	9.12979405	8.29462221	10.13702295	8.03630185	8.15483196	9.74044550
49	9.17644909	7.93225612	9.68287595	7.74087472	8.35336948	9.56893343
50	9.32933043	8.51392976	10.35787886	8.25349862	8.34679255	9.95053124
51	7.30438289	6.48976643	8.33954365	6.22233094	6.34474041	7.91110996
52	8.25258251	7.87023467	9.71610244	7.55982923	7.17895071	9.05085274
53	10.09017038	9.03996781	10.84148643	8.81624834	9.17814033	10.59677104
54	9.49123528	8.40059950	10.19556864	8.18210248	8.60010015	9.97316781
55	9.82416669	8.79666854	10.60477564	8.56860689	8.90621384	10.34170477
56	8.22531638	7.55688325	9.41194236	7.27503893	7.21269234	8.90621384

57	8.12590968	6.97813076	8.76848254	6.76515858	7.27503893	8.56860689
58	10.22998868	8.94998985	10.67776413	8.76848254	9.41194236	10.60477564
59	8.38612997	7.17927147	8.94998985	6.97813076	7.55688325	8.79666854
60	9.20560312	8.38612997	10.22998868	8.12590968	8.22531638	9.82416669

	7	8	9	10	11	12
2						
3						
4						
5						
6						
7						
8	0.65722911					
9	3.25801287	3.39781808				
10	2.37980275	2.88003528	2.01397366			
11	1.34692584	1.34652086	2.05587214	2.03946157		
12	0.88333292	1.44289712	3.83149745	2.49210533	2.13519542	
13	1.26063791	1.37231821	2.03795201	1.83070293	0.22997923	1.99466752
14	0.73467638	1.39161096	3.22516249	1.90905936	1.66089148	0.61972932
15	2.20478573	2.45976075	1.12506191	1.18026044	1.22315952	2.71472718
16	1.11141554	1.43268046	2.16552333	1.51293198	0.66513695	1.69773756
17	2.66069734	2.99323868	1.08836441	0.92655482	1.81479175	3.03912074
18	1.51534905	2.16708517	3.64524803	1.93416392	2.38904111	0.90647130
19	1.72100010	1.70837505	1.72089833	2.03465523	0.37794470	2.48795977
20	0.74141845	0.81459871	2.61006321	2.15655735	0.61148514	1.57177207
21	1.79461431	1.16055612	3.56542174	3.67018831	1.69963346	2.60113984
22	0.37183377	0.71248784	2.89869234	2.17089135	0.97870098	1.18876358
23	0.74445448	0.28160286	3.14357868	2.75308856	1.08777821	1.60463019
24	0.98417960	1.40837291	2.35814165	1.47451258	0.88324711	1.47997962
25	2.75349887	3.07586839	1.01387968	1.00916388	1.87862254	3.13989695
26	2.47776200	1.82818176	4.55106430	4.59546329	2.67962306	3.12914013
27	1.41543023	2.00779866	4.05044434	2.46318622	2.55307373	0.57048387
28	1.63365011	1.62578261	1.79383915	2.02349429	0.29071297	2.40373466
29	2.12850189	2.39515593	1.20883229	1.14002618	1.17910967	2.63158770
30	1.59140950	1.05618601	3.06560022	3.22240579	1.21491441	2.45874044
31	10.77780883	11.38496871	9.58207845	8.62066035	10.65025012	10.43036053
32	8.61387926	9.19371564	7.21068080	6.36379021	8.36843047	8.36193180
33	9.59501595	10.18528878	8.24979959	7.37508960	9.38940152	9.30658683
34	8.71329297	9.36081100	8.36786316	6.90336614	8.90760575	8.19541177
35	12.04430896	12.66700600	11.02177683	9.96411207	12.00234897	11.63672771
36	8.07178284	8.63532056	6.54115034	5.78019277	7.76218915	7.86876988
37	8.93954647	9.55322680	7.92612434	6.82589136	8.86352197	8.57769943
38	10.18201478	10.80260592	9.20204549	8.09928402	10.13808600	9.78871341
39	9.53817929	10.16053912	8.61919173	7.47142042	9.51083555	9.14071475

40	11.19317872	11.80641351	10.06235158	9.06364906	11.09748073	10.82325796
41	9.64733752	10.26136324	8.59773676	7.53039917	9.56713305	9.28071318
42	9.59759109	10.18528311	8.22351088	7.36886543	9.37972168	9.31688175
43	8.25988094	8.90255166	7.81718812	6.39026001	8.40618680	7.77414700
44	8.07559483	8.64321102	6.58424780	5.79379398	7.78276967	7.86160479
45	9.04665533	9.65454885	7.94707772	6.90227958	8.93642301	8.70458460
46	8.61952969	9.19643148	7.18535114	6.36081619	8.36102298	8.37589042
47	8.88623781	9.51985036	8.20281015	6.91156627	8.94576673	8.44388394
48	9.40483814	10.00573812	8.19492395	7.22656946	9.25380359	9.08417377
49	9.16622176	9.80902158	8.66351220	7.28315193	9.30430418	8.67571240
50	9.61824903	10.21671847	8.36729291	7.42891696	9.45294322	9.30430418
51	7.58005028	8.17727566	6.45164036	5.39825559	7.42891696	7.28315193
52	8.79687040	9.33034510	6.95920612	6.45164036	8.36729291	8.66351220
53	10.22430402	10.85040028	9.33034510	8.17727566	10.21671847	9.80902158
54	9.59452555	10.22430402	8.79687040	7.58005028	9.61824903	9.16622176
55	9.97316781	10.59677104	9.05085274	7.91110996	9.95053124	9.56893343
56	8.60010015	9.17814033	7.17895071	6.34474041	8.34679255	8.35336948
57	8.18210248	8.81624834	7.55982923	6.22233094	8.25349862	7.74087472
58	10.19556864	10.84148643	9.71610244	8.33954365	10.35787886	9.68287595
59	8.40059950	9.03996781	7.87023467	6.48976643	8.51392976	7.93225612
60	9.49123528	10.09017038	8.25258251	7.30438289	9.32933043	9.17644909

	13	14	15	16	17	18
2						
3						
4						
5						
6						
7						
8						
9						
10						
11						
12						
13						
14	1.48557454					
15	1.10639848	2.10382530				
16	0.43865198	1.13122671	1.09337743			
17	1.67770082	2.41970184	0.60205816	1.57346834		
18	2.18790067	0.80753518	2.53511915	1.77616501	2.68488573	
19	0.49761551	1.98255790	1.03822641	0.89523541	1.63945365	2.66970721
20	0.57211215	1.18418280	1.64521851	0.65132180	2.18349803	1.97438775
21	1.86793726	2.51249929	2.91365761	2.16136951	3.51122343	3.30858119
22	0.88933922	0.85180527	1.86670187	0.78149230	2.35291749	1.65815491

23	1.13837954	1.44564739	2.24050409	1.26265500	2.79266369	2.24658489
24	0.66587203	0.89670438	1.25659769	0.24498714	1.67779957	1.53218766
25	1.74954321	2.52037743	0.65723954	1.66137104	0.10192040	2.78583587
26	2.83307610	3.21015447	3.89986637	3.08366176	4.49437619	3.95476681
27	2.38234747	0.89712576	2.92616309	2.02160618	3.15149674	0.56929021
28	0.41792212	1.90342464	1.06650684	0.82821177	1.66853410	2.59878313
29	1.05084877	2.02030880	0.08378888	1.01729696	0.63608817	2.45337421
30	1.39884051	2.25663888	2.41806703	1.73117409	3.01772857	3.06413720
31	10.44869075	10.09442433	9.56842963	10.12395148	8.99261540	9.52728983
32	8.17572229	7.96461519	7.24958745	7.87649311	6.66353146	7.47765068
33	9.19385968	8.93212413	8.28050205	8.88616853	7.69677995	8.41298826
34	8.68467789	7.98615843	8.01980952	8.29531689	7.52421224	7.29821057
35	11.79470544	11.34293080	10.95436702	11.45241597	10.38923104	10.73025888
36	7.57528779	7.44313465	6.62248668	7.29193253	6.03067844	6.99954657
37	8.65659233	8.24998547	7.81838311	8.31798403	7.25759914	7.67391860
38	9.92964230	9.48374963	9.09948614	9.58589242	8.53946028	8.88260626
39	9.30080625	8.83825853	8.48541950	8.95278762	7.93120391	8.23454353
40	10.89355700	10.50268302	10.02805255	10.56208326	9.45611174	9.91818060
41	9.36105055	8.95685171	8.51433070	9.02437990	7.94972931	8.37625586
42	9.18532284	8.93771865	8.26579972	8.88066401	7.68051943	8.42481842
43	8.18516639	7.53791151	7.49373328	7.80288183	6.98894242	6.87155921
44	7.59413892	7.44200262	6.64930475	7.30628414	6.05928050	6.98894242
45	8.73222935	8.36372816	7.87372468	8.40132509	7.30628414	7.80288183
46	8.16959243	7.97376130	7.23696150	7.87372468	6.64930475	7.49373328
47	8.72972753	8.17398137	7.97376130	8.36372816	7.44200262	7.53791151
48	9.05326398	8.72972753	8.16959243	8.73222935	7.59413892	8.18516639
49	9.08417377	8.44388394	8.37589042	8.70458460	7.86160479	7.77414700
50	9.25380359	8.94576673	8.36102298	8.93642301	7.78276967	8.40618680
51	7.22656946	6.91156627	6.36081619	6.90227958	5.79379398	6.39026001
52	8.19492395	8.20281015	7.18535114	7.94707772	6.58424780	7.81718812
53	10.00573812	9.51985036	9.19643148	9.65454885	8.64321102	8.90255166
54	9.40483814	8.88623781	8.61952969	9.04665533	8.07559483	8.25988094
55	9.74044550	9.27161751	8.92422772	9.39202099	8.36905978	8.66255670
56	8.15483196	7.95300332	7.22491339	7.85759147	6.63793462	7.47044010
57	8.03630185	7.46957135	7.29657617	7.66697639	6.77298058	6.83474573
58	10.13702295	9.46983167	9.43451542	9.75458062	8.92003238	8.78538111
59	8.29462221	7.68217046	7.57929094	7.91817543	7.06449796	7.02786991
60	9.12979405	8.81833193	8.23971471	8.81143302	7.66234575	8.27828882

	19	20	21	22	23	24
2						
3						
4						
5						

6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						
17						
18						
19						
20	0.98903074					
21	1.90740351	1.54595705				
22	1.35080945	0.38475606	1.70653577			
23	1.44080942	0.61479972	1.06882954	0.64845784		
24	1.13777005	0.70815730	2.25364109	0.70787669	1.28190637	
25	1.68723294	2.26426711	3.57089598	2.44201307	2.87113195	1.77172652
26	2.90069617	2.44003827	0.99365529	2.50647481	1.86681381	3.13936011
27	2.87954865	2.03975359	3.16782263	1.67151159	2.15404634	1.78150198
28	0.08750275	0.90157991	1.85952654	1.26360949	1.36007444	1.06841793
29	1.01748640	1.58065139	2.87519605	1.79340639	2.18022115	1.17582091
30	1.40456365	1.16549004	0.50398301	1.40997186	0.85411952	1.85503420
31	10.58385718	10.75001268	12.28526344	10.68247309	11.31438813	10.04438846
32	8.27687379	8.51586211	10.03236741	8.48317025	9.09919580	7.81860674
33	9.30505421	9.52162735	11.04527182	9.47686917	10.09900853	8.82086609
34	8.93749309	8.85738195	10.39479887	8.70260350	9.35106011	8.15881857
35	11.95654612	12.06581474	13.60904179	11.97384095	12.61383809	11.35779296
36	7.65526695	7.93666053	9.43889735	7.92304583	8.52848300	7.24658949
37	8.81769614	8.93579550	10.47683130	8.85465133	9.49099428	8.22832576
38	10.09669343	10.19900781	11.74225157	10.10827392	10.74771359	9.49099428
39	9.47621445	9.56279126	11.10702545	9.46775360	10.10827392	8.85465133
40	11.03881126	11.18370153	12.72252691	11.10702545	11.74225157	10.47683130
41	9.51745151	9.64314696	11.18370153	9.56279126	10.19900781	8.93579550
42	9.29195644	9.51745151	11.03881126	9.47621445	10.09669343	8.81769614
43	8.42481842	8.37625586	9.91818060	8.23454353	8.88260626	7.67391860
44	7.68051943	7.94972931	9.45611174	7.93120391	8.53946028	7.25759914
45	8.88066401	9.02437990	10.56208326	8.95278762	9.58589242	8.31798403
46	8.26579972	8.51433070	10.02805255	8.48541950	9.09948614	7.81838311
47	8.93771865	8.95685171	10.50268302	8.83825853	9.48374963	8.24998547
48	9.18532284	9.36105055	10.89355700	9.30080625	9.92964230	8.65659233

49	9.31688175	9.28071318	10.82325796	9.14071475	9.78871341	8.57769943
50	9.37972168	9.56713305	11.09748073	9.51083555	10.13808600	8.86352197
51	7.36886543	7.53039917	9.06364906	7.47142042	8.09928402	6.82589136
52	8.22351088	8.59773676	10.06235158	8.61919173	9.20204549	7.92612434
53	10.18528311	10.26136324	11.80641351	10.16053912	10.80260592	9.55322680
54	9.59759109	9.64733752	11.19317872	9.53817929	10.18201478	8.93954647
55	9.91572178	10.00140041	11.54583394	9.90485024	10.54582614	9.29324794
56	8.25310520	8.49770884	10.01256863	8.46730275	9.08214604	7.80123148
57	8.25185182	8.25707184	9.80267836	8.13566294	8.78151657	7.55067047
58	10.37368279	10.32575310	11.86637530	10.17881910	10.82718472	9.62433275
59	8.52283118	8.49931325	10.04336080	8.36639367	9.01374713	7.79487573
60	9.25751040	9.44167866	10.97258415	9.38447947	10.01208216	8.73785605

	25	26	27	28	29	30
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	4.55678194					
27	3.25337703	3.69277482				
28	1.72063860	2.85149358	2.80005937			
29	0.70070918	3.85849720	2.84255986	1.03929679		
30	3.07496068	1.49753815	3.00577611	1.35876869	2.38241797	
31	8.97327437	13.18055033	10.01378473	10.59208107	9.57452581	11.84306389

32	6.63793601	10.95577986	8.00139376	8.29070940	7.26126533	9.57452581
33	7.67263363	11.96008414	8.92316783	9.31734643	8.29070940	10.59208107
34	7.53746783	11.18888288	7.70418832	8.92316783	8.00139376	10.01378473
35	10.37544205	14.47972317	11.18888288	11.96008414	10.95577986	13.18055033
36	6.00049704	10.37544205	7.53746783	7.67263363	6.63793601	8.97327437
37	7.24658949	11.35779296	8.15881857	8.82086609	7.81860674	10.04438846
38	8.52848300	12.61383809	9.35106011	10.09900853	9.09919580	11.31438813
39	7.92304583	11.97384095	8.70260350	9.47686917	8.48317025	10.68247309
40	9.43889735	13.60904179	10.39479887	11.04527182	10.03236741	12.28526344
41	7.93666053	12.06581474	8.85738195	9.52162735	8.51586211	10.75001268
42	7.65526695	11.95654612	8.93749309	9.30505421	8.27687379	10.58385718
43	6.99954657	10.73025888	7.29821057	8.41298826	7.47765068	9.52728983
44	6.03067844	10.38923104	7.52421224	7.69677995	6.66353146	8.99261540
45	7.29193253	11.45241597	8.29531689	8.88616853	7.87649311	10.12395148
46	6.62248668	10.95436702	8.01980952	8.28050205	7.24958745	9.56842963
47	7.44313465	11.34293080	7.98615843	8.93212413	7.96461519	10.09442433
48	7.57528779	11.79470544	8.68467789	9.19385968	8.17572229	10.44869075
49	7.86876988	11.63672771	8.19541177	9.30658683	8.36193180	10.43036053
50	7.76218915	12.00234897	8.90760575	9.38940152	8.36843047	10.65025012
51	5.78019277	9.96411207	6.90336614	7.37508960	6.36379021	8.62066035
52	6.54115034	11.02177683	8.36786316	8.24979959	7.21068080	9.58207845
53	8.63532056	12.66700600	9.36081100	10.18528878	9.19371564	11.38496871
54	8.07178284	12.04430896	8.71329297	9.59501595	8.61387926	10.77780883
55	8.36036437	12.41110508	9.12651622	9.91647936	8.92222922	11.12183939
56	6.61164615	10.93771978	7.99570051	8.26744131	7.23713507	9.55367042
57	6.77734523	10.63984198	7.28576243	8.24465984	7.28535370	9.39708943
58	8.92682047	12.66966466	9.19108917	10.36278419	9.42038315	11.47823018
59	7.07178710	10.86646237	7.46383594	8.51325574	7.56567290	9.64578988
60	7.64233397	11.87654209	8.77970244	9.26684310	8.24672035	10.52597798

	31	32	33	34	35	36
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	2.38241797					
33	1.35876869	1.03929679				
34	3.00577611	2.84255986	2.80005937			
35	1.49753815	3.85849720	2.85149358	3.69277482		
36	3.07496068	0.70070918	1.72063860	3.25337703	4.55678194	
37	1.85503420	1.17582091	1.06841793	1.78150198	3.13936011	1.77172652
38	0.85411952	2.18022115	1.36007444	2.15404634	1.86681381	2.87113195
39	1.40997186	1.79340639	1.26360949	1.67151159	2.50647481	2.44201307
40	0.50398301	2.87519605	1.85952654	3.16782263	0.99365529	3.57089598
41	1.16549004	1.58065139	0.90157991	2.03975359	2.44003827	2.26426711
42	1.40456365	1.01748640	0.08750275	2.87954865	2.90069617	1.68723294
43	3.06413720	2.45337421	2.59878313	0.56929021	3.95476681	2.78583587
44	3.01772857	0.63608817	1.66853410	3.15149674	4.49437619	0.10192040
45	1.73117409	1.01729696	0.82821177	2.02160618	3.08366176	1.66137104
46	2.41806703	0.08378888	1.06650684	2.92616309	3.89986637	0.65723954
47	2.25663888	2.02030880	1.90342464	0.89712576	3.21015447	2.52037743
48	1.39884051	1.05084877	0.41792212	2.38234747	2.83307610	1.74954321
49	2.45874044	2.63158770	2.40373466	0.57048387	3.12914013	3.13989695
50	1.21491441	1.17910967	0.29071297	2.55307373	2.67962306	1.87862254
51	3.22240579	1.14002618	2.02349429	2.46318622	4.59546329	1.00916388
52	3.06560022	1.20883229	1.79383915	4.05044434	4.55106430	1.01387968
53	1.05618601	2.39515593	1.62578261	2.00779866	1.82818176	3.07586839
54	1.59140950	2.12850189	1.63365011	1.41543023	2.47776200	2.75349887
55	1.10355838	2.12422033	1.40688198	1.90232939	2.07135396	2.80005777
56	2.41914084	0.05339729	1.07118283	2.89193474	3.89819518	0.65888220
57	2.86095867	1.98815672	2.22285621	0.98095794	3.90494737	2.31043163

58	2.30050545	3.34806514	2.81049258	1.48748590	2.43091217	3.95173781
59	2.83137632	2.26026801	2.36299891	0.63117948	3.75847589	2.63267462
60	1.33179074	1.07892064	0.32029308	2.48223182	2.78525891	1.77962757

	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						
38	1.28190637					
39	0.70787669	0.64845784				
40	2.25364109	1.06882954	1.70653577			

41	0.70815730	0.61479972	0.38475606	1.54595705		
42	1.13777005	1.44080942	1.35080945	1.90740351	0.98903074	
43	1.53218766	2.24658489	1.65815491	3.30858119	1.97438775	2.66970721
44	1.67779957	2.79266369	2.35291749	3.51122343	2.18349803	1.63945365
45	0.24498714	1.26265500	0.78149230	2.16136951	0.65132180	0.89523541
46	1.25659769	2.24050409	1.86670187	2.91365761	1.64521851	1.03822641
47	0.89670438	1.44564739	0.85180527	2.51249929	1.18418280	1.98255790
48	0.66587203	1.13837954	0.88933922	1.86793726	0.57211215	0.49761551
49	1.47997962	1.60463019	1.18876358	2.60113984	1.57177207	2.48795977
50	0.88324711	1.08777821	0.97870098	1.69963346	0.61148514	0.37794470
51	1.47451258	2.75308856	2.17089135	3.67018831	2.15655735	2.03465523
52	2.35814165	3.14357868	2.89869234	3.56542174	2.61006321	1.72089833
53	1.40837291	0.28160286	0.71248784	1.16055612	0.81459871	1.70837505
54	0.98417960	0.74445448	0.37183377	1.79461431	0.74141845	1.72100010
55	1.12673937	0.25449571	0.43971093	1.31668679	0.54902756	1.49226950
56	1.22915844	2.22832342	1.84586618	2.91339291	1.63033843	1.04581190
57	1.16047232	2.09882024	1.46224381	3.16584758	1.71517256	2.28724716
58	2.18349703	1.57047070	1.58553415	2.22771658	1.91152513	2.89758098
59	1.29873144	2.02160426	1.42301436	3.08717052	1.73423062	2.43498943
60	0.77680569	1.14131655	0.95502497	1.81082012	0.61181046	0.40427156

	43	44	45	46	47	48
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  2.68488573
45  1.77616501  1.57346834
46  2.53511915  0.60205816  1.09337743
47  0.80753518  2.41970184  1.13122671  2.10382530
48  2.18790067  1.67770082  0.43865198  1.10639848  1.48557454
49  0.90647130  3.03912074  1.69773756  2.71472718  0.61972932  1.99466752
50  2.38904111  1.81479175  0.66513695  1.22315952  1.66089148  0.22997923
51  1.93416392  0.92655482  1.51293198  1.18026044  1.90905936  1.83070293
52  3.64524803  1.08836441  2.16552333  1.12506191  3.22516249  2.03795201
53  2.16708517  2.99323868  1.43268046  2.45976075  1.39161096  1.37231821
54  1.51534905  2.66069734  1.11141554  2.20478573  0.73467638  1.26063791
55  1.99405414  2.71622376  1.15153572  2.19073528  1.19582151  1.11807661
56  2.49900638  0.59870690  1.07031328  0.03737325  2.07216049  1.09675728
57  0.47598476  2.20936021  1.39507365  2.06901706  0.70442596  1.82630253
58  1.95193053  3.85582801  2.33333995  3.42571356  1.54850466  2.46791751
59  0.24117748  2.53099030  1.54340813  2.34297675  0.57663836  1.95061987
60  2.29634904  1.71236996  0.54962468  1.12796417  1.58636362  0.11128471
            49          50          51          52          53          54
2
3
4
5
6
```

```
50  2.13519542
51  2.49210533  2.03946157
52  3.83149745  2.05587214  2.01397366
53  1.44289712  1.34652086  2.88003528  3.39781808
54  0.88333292  1.34692584  2.37980275  3.25801287  0.65722911
55  1.35750901  1.11960881  2.59971060  3.15402602  0.28270138  0.49023015
56  2.68394064  1.21974369  1.14881421  1.15875825  2.44492554  2.18159615
57  1.10840931  2.04195483  1.48234518  3.17346577  2.07883530  1.42769541
58  1.05871432  2.51979294  3.45060696  4.48262034  1.29607607  1.22537306
59  0.79711524  2.14982660  1.83344646  3.46100619  1.95526187  1.29937233
60  2.08375927  0.12791651  1.91306319  2.01033045  1.38818570  1.32670747
            55          56          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  2.17465173
57  1.85950297  2.03253144
58  1.42652801  3.40135729  2.15627420
59  1.77072625  2.30751125  0.35411715  1.85573461
60  1.14569688  1.12189415  1.93727728  2.52139040  2.05855580
```

Now we can put the distance into `hclust()`.

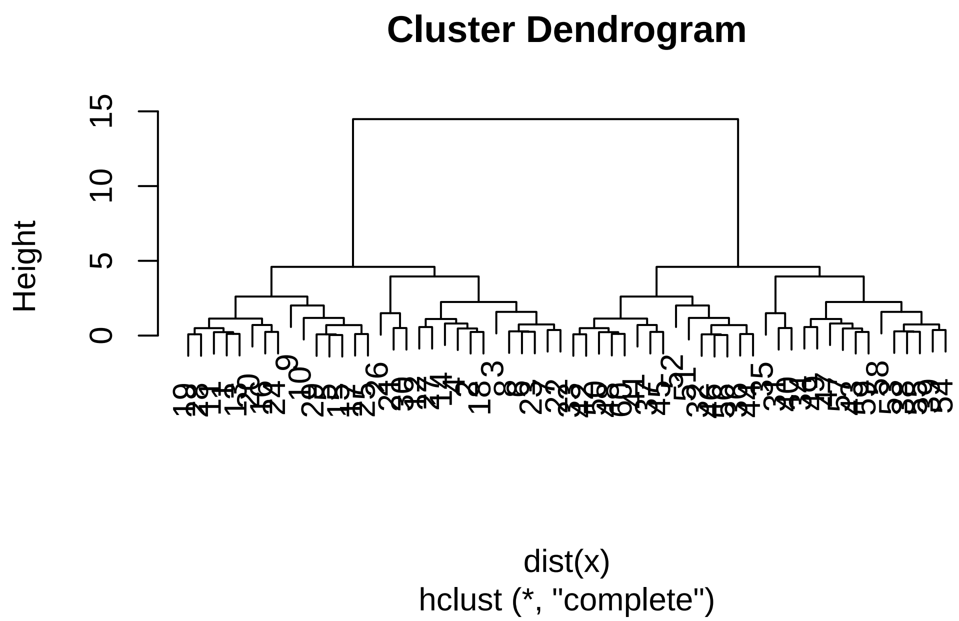
```r
clustering <- hclust(dist(x))
clustering
```

```
Call:
hclust(d = dist(x))

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

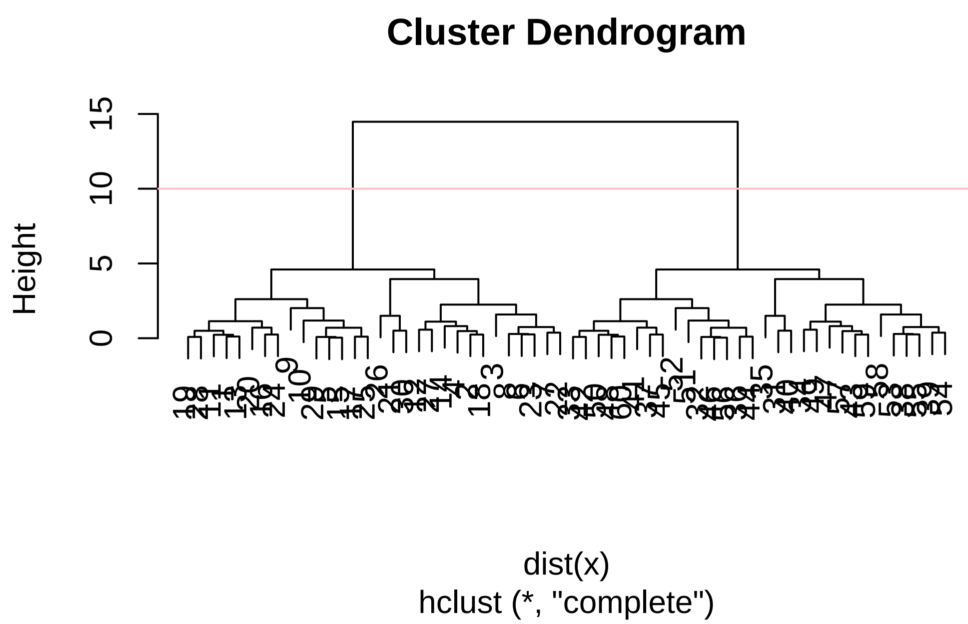
We can now plot `clustering` to get a dendrogram:

```
plot(clustering)
```



Let's add an horizontal line:

```
plot(clustering)
abline(h = 10, col = 'pink')
```



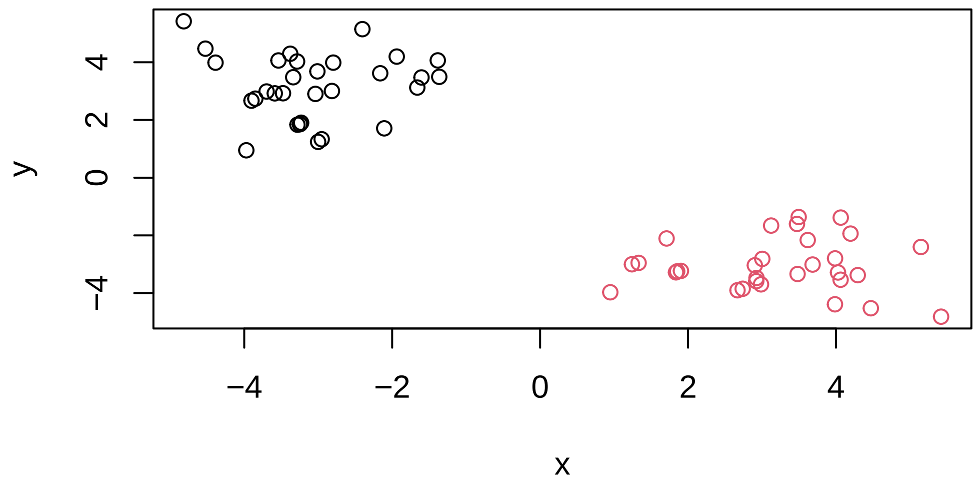
To get our results (i.e., membership vector), we need to 'cut' the tree. The function for doing that is `cutree`.

```
subgroups <- cutree(clustering, h = 10)
subgroups
```

```
 [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 2 2
```

Plotting this:

```
plot(x, col = subgroups)
```



You can also 'cut' your tree with the number of clusters you want:

```
cutree(clustering, k = 2)
```

```
 [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
```

PCA of UK Food Data

First we will look at the PCA of the UK Food Data

```
url <- "https://tinyurl.com/UK-foods"
j <- read.csv(url)
j
```

```
                    X England Wales Scotland N.Ireland
1              Cheese     105   103      103        66
2       Carcass_meat      245   227      242       267
3         Other_meat      685   803      750       586
4                Fish     147   160      122        93
```

```
5        Fats_and_oils      193   235       184       209
6               Sugars      156   175       147       139
7      Fresh_potatoes       720   874       566      1033
8           Fresh_Veg       253   265       171       143
9           Other_Veg       488   570       418       355
10 Processed_potatoes       198   203       220       187
11       Processed_Veg      360   365       337       334
12         Fresh_fruit     1102  1137       957       674
13             Cereals     1472  1582      1462      1494
14           Beverages       57    73        53        47
15         Soft_drinks     1374  1256      1572      1506
16     Alcoholic_drinks     375   475       458       135
17       Confectionery       54    64        62        41
```

```r
url <- "https://tinyurl.com/UK-foods"
j <- read.csv(url, row.names = 1)
j
```

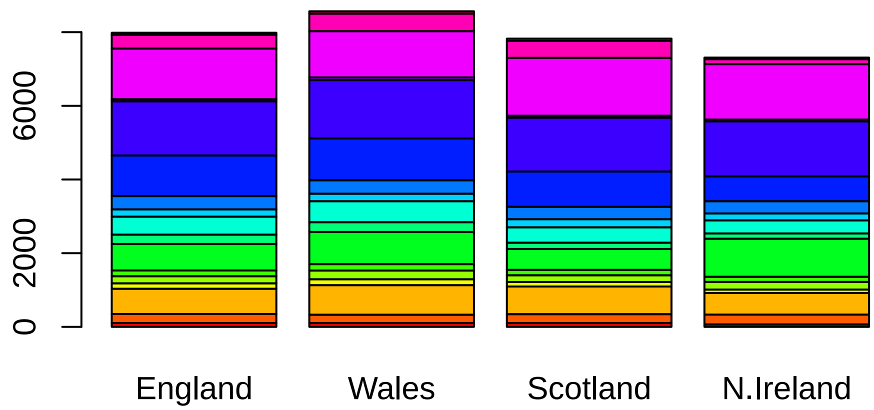
```
                    England Wales Scotland N.Ireland
Cheese                  105   103      103        66
Carcass_meat            245   227      242       267
Other_meat              685   803      750       586
Fish                    147   160      122        93
Fats_and_oils           193   235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

```r
dim(j)
```

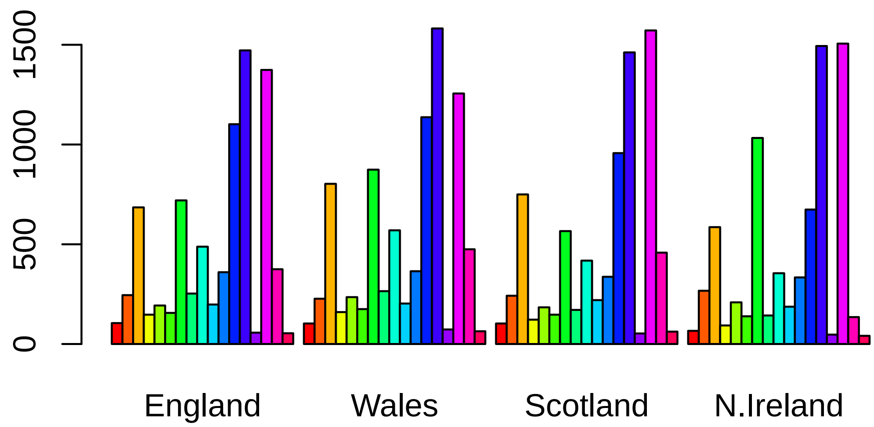
```
[1] 17  4
```

Now we can generate some basic visualizations.

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

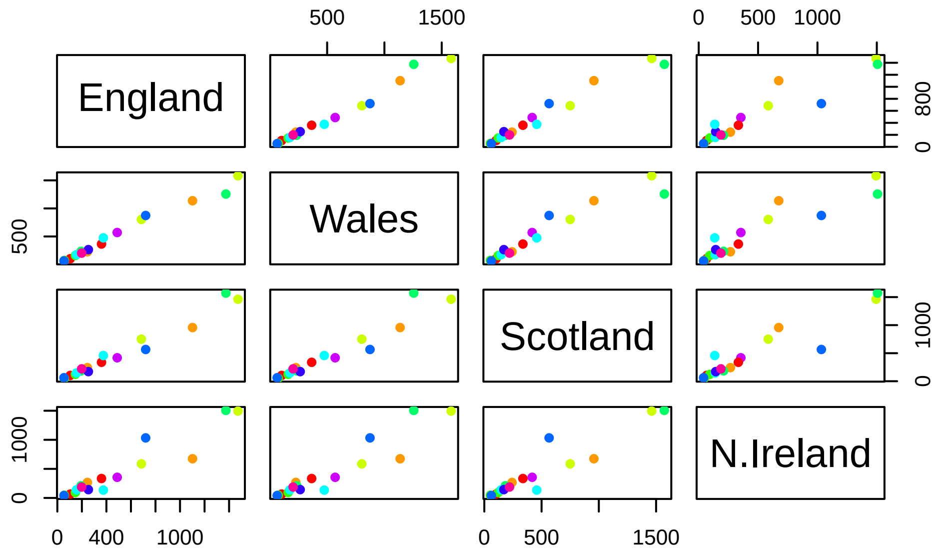



```
barplot(as.matrix(j), beside=T, col=rainbow(nrow(j)))
```



Other visualizations we can do:

```
pairs(j, col=rainbow(10), pch=16)
```



Let's apply PCA (principal components analysis). For that, we need to use the command `prcomp()`. This function expects the transpose of our data.

```
t(j)
```

```
          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
```

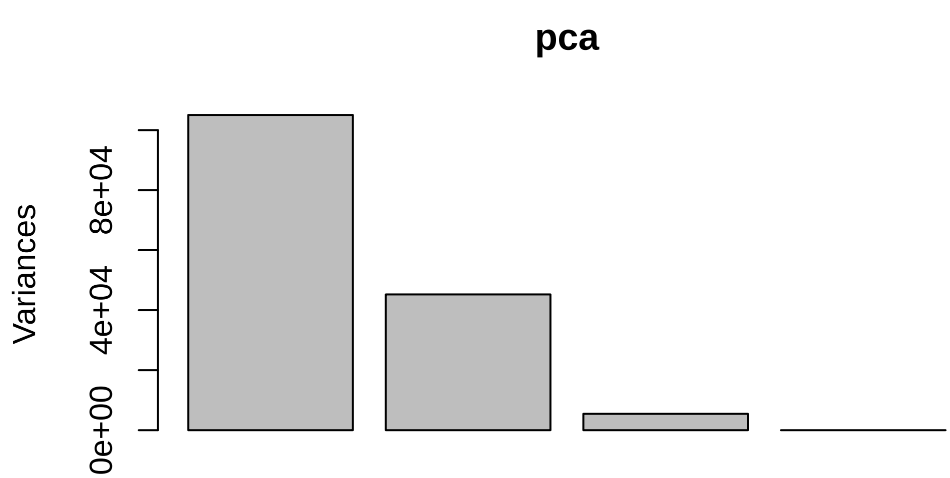
```
pca <- prcomp(t(j))
```

```
pca <- prcomp(t(j))
summary(pca)
```

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

Let's plot the PCA results.

```
plot(pca)
```



We need to access the results of the PCA analysis.

```r
pca$sdev
```

```
[1] 3.241502e+02 2.127478e+02 7.387622e+01 4.188568e-14
```

```r
pca$center
```

```
           Cheese       Carcass_meat         Other_meat               Fish
            94.25             245.25             706.00             130.50
    Fats_and_oils             Sugars     Fresh_potatoes          Fresh_Veg
           205.25             154.25             798.25             208.00
        Other_Veg  Processed_potatoes      Processed_Veg        Fresh_fruit
           457.75             202.00             349.00             967.50
          Cereals          Beverages        Soft_drinks    Alcoholic_drinks
          1502.50              57.50            1427.00             360.75
    Confectionery
            55.25
```

```r
pca$scale
```

```
[1] FALSE
```

```r
pca$rotation
```

```
                            PC1          PC2         PC3          PC4
Cheese             -0.056955380 -0.016012850 -0.02394295 -0.691718038
Carcass_meat        0.047927628 -0.013915823 -0.06367111  0.635384915
Other_meat         -0.258916658  0.015331138  0.55384854  0.198175921
Fish               -0.084414983  0.050754947 -0.03906481 -0.015824630
Fats_and_oils      -0.005193623  0.095388656  0.12522257  0.052347444
Sugars             -0.037620983  0.043021699  0.03605745  0.014481347
Fresh_potatoes      0.401402060  0.715017078  0.20668248 -0.151706089
Fresh_Veg          -0.151849942  0.144900268 -0.21382237  0.056182433
Other_Veg          -0.243593729  0.225450923  0.05332841 -0.080722623
Processed_potatoes -0.026886233 -0.042850761  0.07364902 -0.022618707
Processed_Veg      -0.036488269  0.045451802 -0.05289191  0.009235001
Fresh_fruit        -0.632640898  0.177740743 -0.40012865 -0.021899087
Cereals            -0.047702858  0.212599678  0.35884921  0.084667257
```

```
Beverages           -0.026187756  0.030560542  0.04135860 -0.011880823
Soft_drinks          0.232244140 -0.555124311  0.16942648 -0.144367046
Alcoholic_drinks    -0.463968168 -0.113536523  0.49858320 -0.115797605
Confectionery       -0.029650201 -0.005949921  0.05232164 -0.003695024
```

```r
pca$x
```

```
                PC1        PC2         PC3           PC4
England   -144.99315   2.532999 -105.768945  2.842865e-14
Wales     -240.52915 224.646925   56.475555  7.804382e-13
Scotland   -91.86934 -286.081786  44.415495 -9.614462e-13
N.Ireland  477.39164  58.901862    4.877895  1.448078e-13
```

```r
attributes(pca)
```

```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

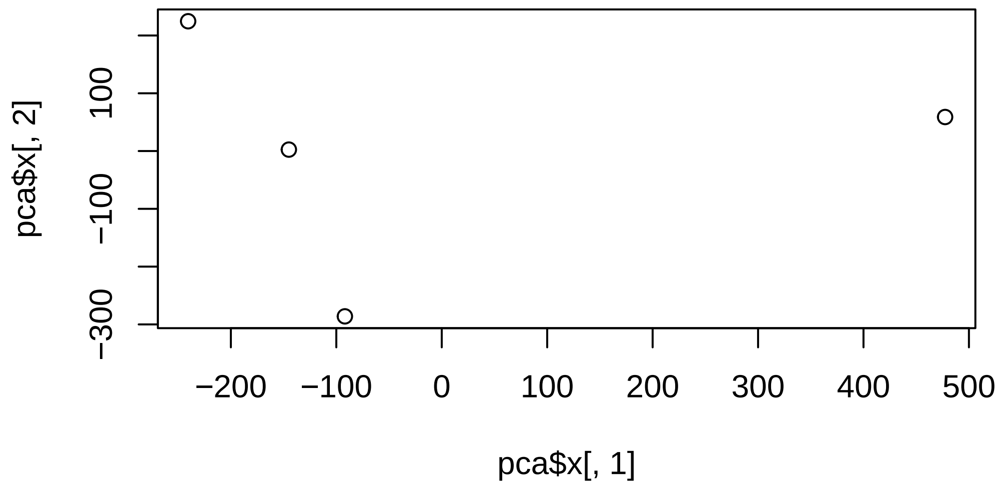
We can explore the `pca$x` dataframe:

```r
pca$x
```

```
                PC1        PC2         PC3           PC4
England   -144.99315   2.532999 -105.768945  2.842865e-14
Wales     -240.52915 224.646925   56.475555  7.804382e-13
Scotland   -91.86934 -286.081786  44.415495 -9.614462e-13
N.Ireland  477.39164  58.901862    4.877895  1.448078e-13
```

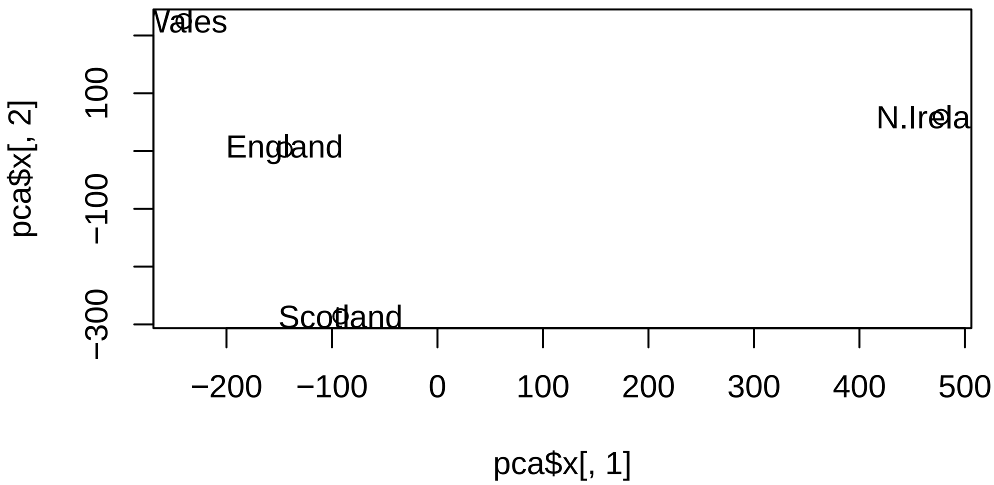
Plotting:

```r
plot(x = pca$x[,1], y = pca$x[,2])
```



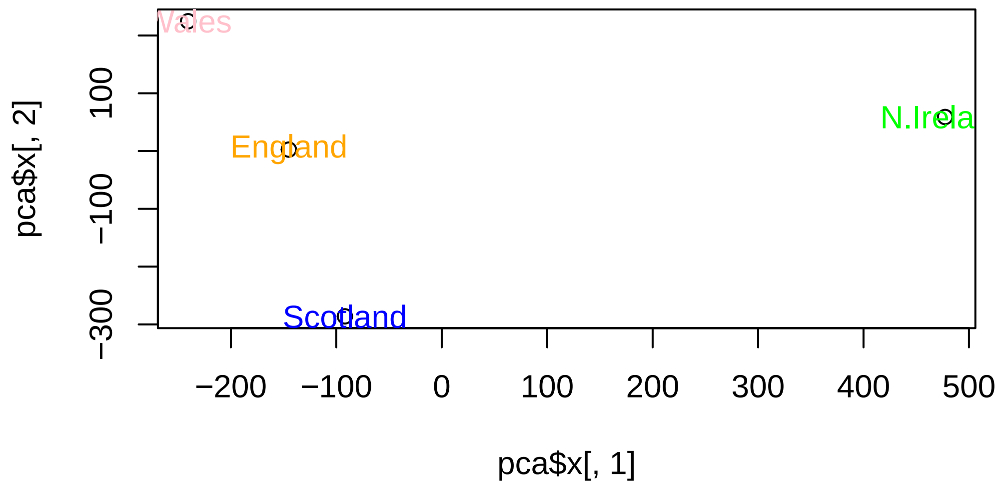
We don't know which country is which, so we need to add labels:

```
plot(x = pca$x[,1], y = pca$x[,2])
text(x = pca$x[,1], y = pca$x[,2], colnames(j))
```



Let's add some color!

```
plot(x = pca$x[,1], y = pca$x[,2])
colors_countries <- c('orange', 'pink', 'blue', 'green')
text(x = pca$x[,1], y = pca$x[,2], colnames(j), col = colors_countries)
```



PCA of RNA Seq Data

Now, we will look at the PCA of RNA seq data

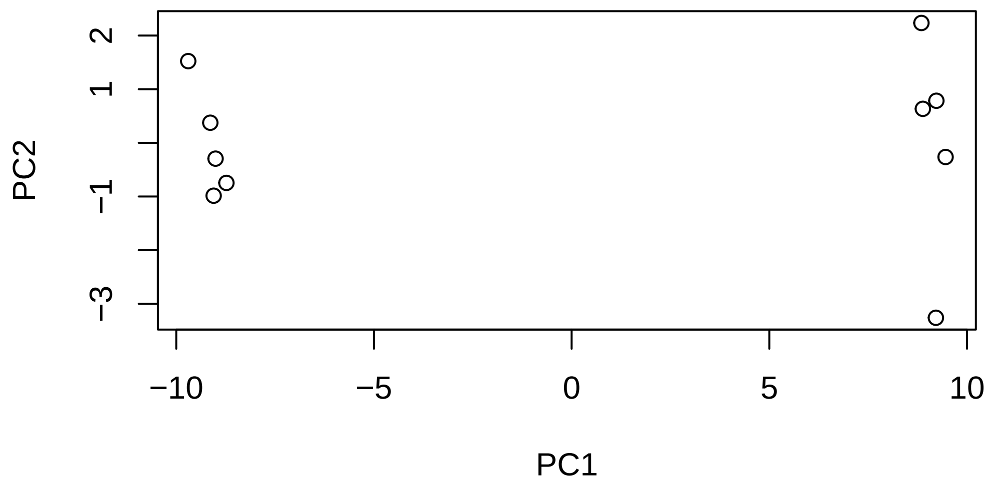
```
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

```
       wt1 wt2  wt3  wt4 wt5 ko1 ko2 ko3 ko4 ko5
gene1  439 458  408  429 420  90  88  86  90  93
gene2  219 200  204  210 187 427 423 434 433 426
gene3 1006 989 1030 1017 973 252 237 238 226 210
gene4  783 792  829  856 760 849 856 835 885 894
gene5  181 249  204  244 225 277 305 272 270 279
gene6  460 502  491  491 493 612 594 577 618 638
```

blah

```
## Again we have to take the transpose of our data
pca <- prcomp(t(rna.data), scale=TRUE)
```

```r
## Simple un polished plot of pc1 and pc2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```



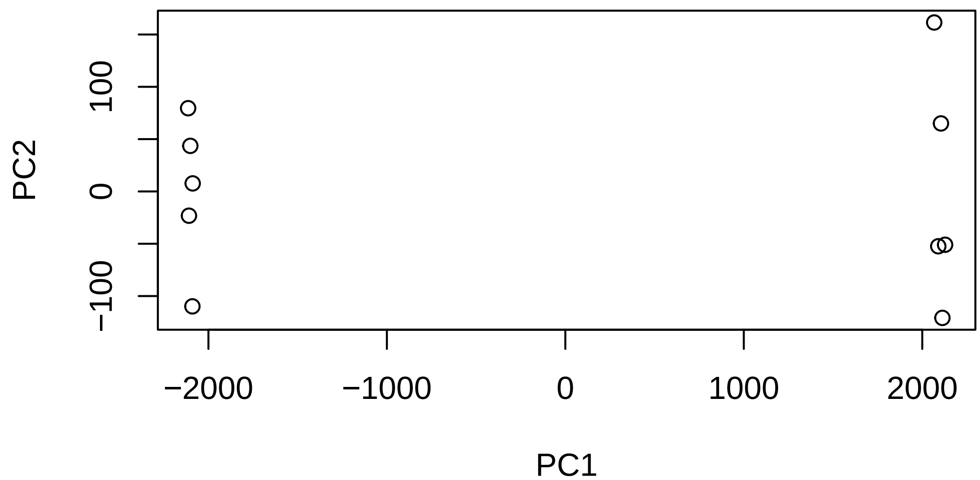
Let's apply PCA:

```r
pca_rna <- prcomp (t(rna.data))
summary (pca_rna)
```

```
Importance of components:
                             PC1     PC2      PC3      PC4      PC5      PC6
Standard deviation     2214.2633 88.9209 84.33908 77.74094 69.66341 67.78516
Proportion of Variance    0.9917  0.0016  0.00144  0.00122  0.00098  0.00093
Cumulative Proportion     0.9917  0.9933  0.99471  0.99593  0.99691  0.99784
                            PC7      PC8      PC9      PC10
Standard deviation     65.29428 59.90981 53.20803 3.142e-13
Proportion of Variance  0.00086  0.00073  0.00057 0.000e+00
Cumulative Proportion   0.99870  0.99943  1.00000 1.000e+00
```

Let's plot

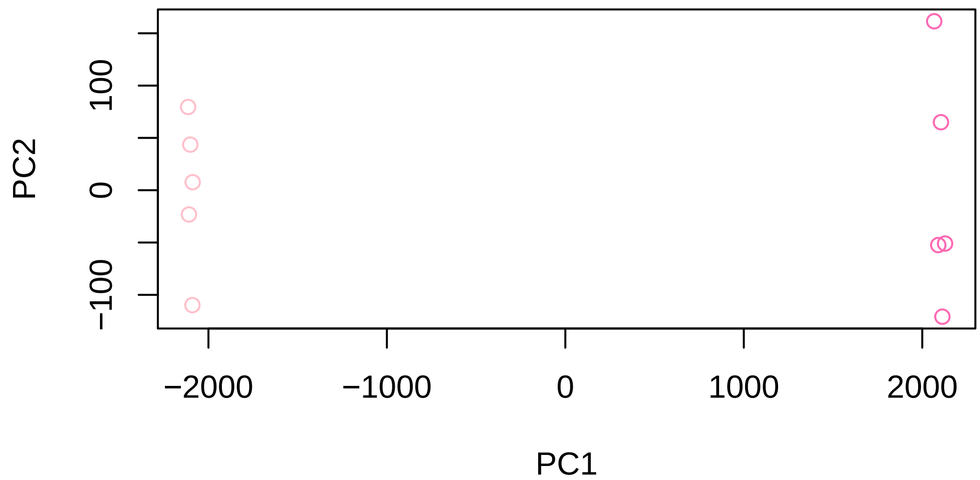
```r
plot(pca_rna$x[,1], pca_rna$x[,2], xlab = "PC1", ylab = "PC2")
```




```r
cols_samples <- c(rep('pink', 5), rep('hotpink', 5))
cols_samples
```

```
[1] "pink"    "pink"    "pink"    "pink"    "pink"    "hotpink" "hotpink"
[8] "hotpink" "hotpink" "hotpink"
```

```r
plot(pca_rna$x[,1], pca_rna$x[,2], xlab = "PC1", ylab = "PC2", col = cols_samples)
```



Questions

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

```
dim(j)
```

```
[1] 17  4
```

Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

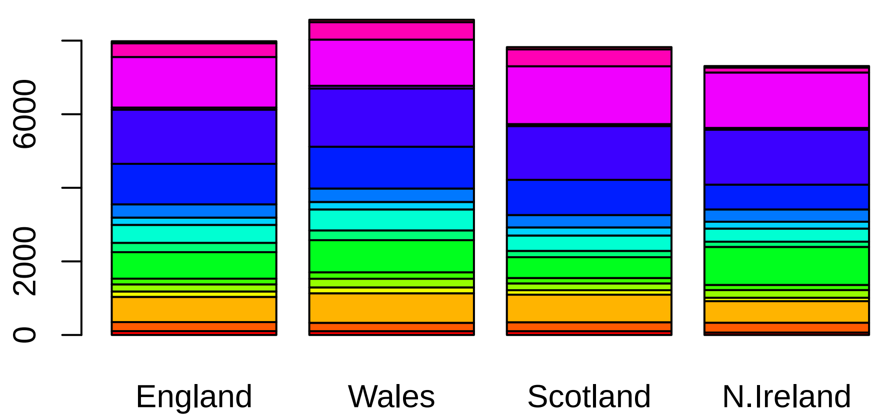
```
url <- "https://tinyurl.com/UK-foods"
j <- read.csv(url, row.names = 1)
j
```

```
                    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
```

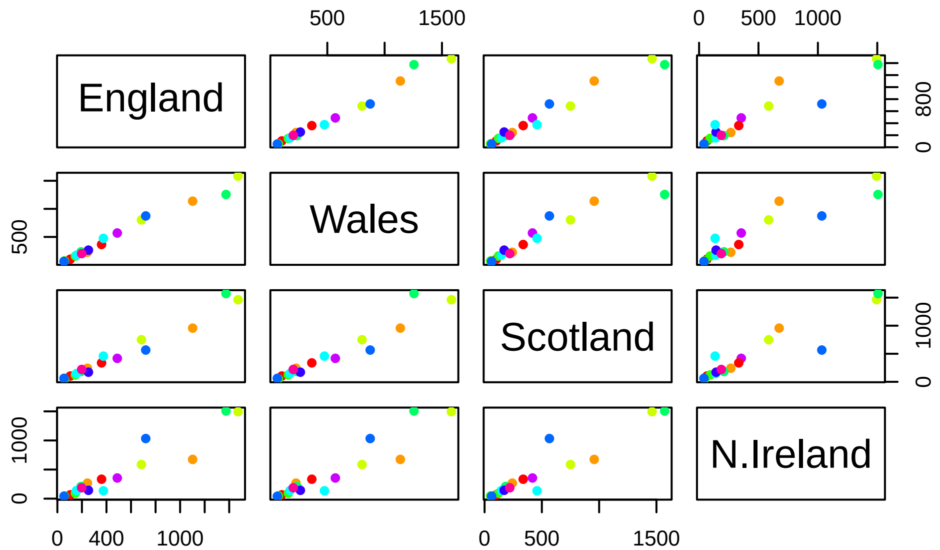
Q3: Changing what optional argument in the above **barplot()**function results in the following plot?

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



Q5: Generating all pairwise plots may help somewhat. Can you make sense of the following code and resulting figure? What does it mean if a given point lies on the diagonal for a given plot?

```
pairs(j, col=rainbow(10), pch=16)
```



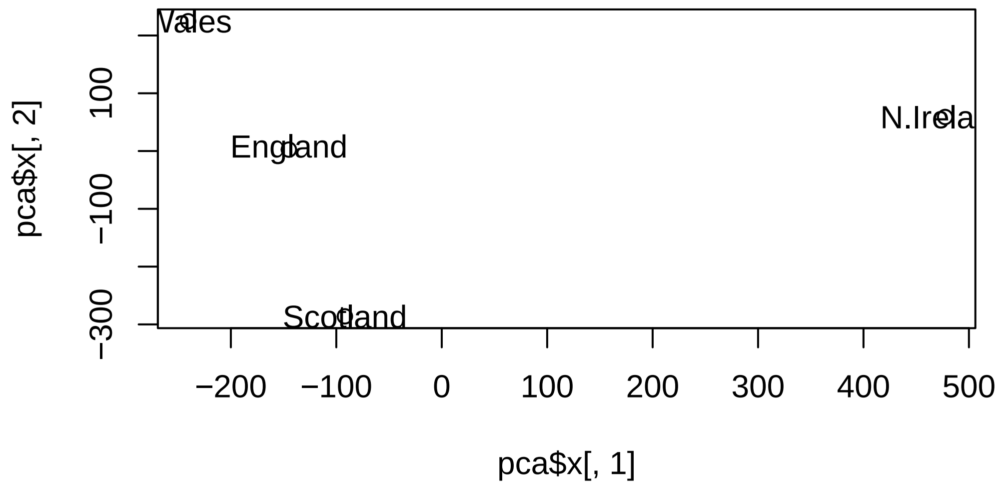
Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

```
pca <- prcomp( t(j) )
summary(pca)
```

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

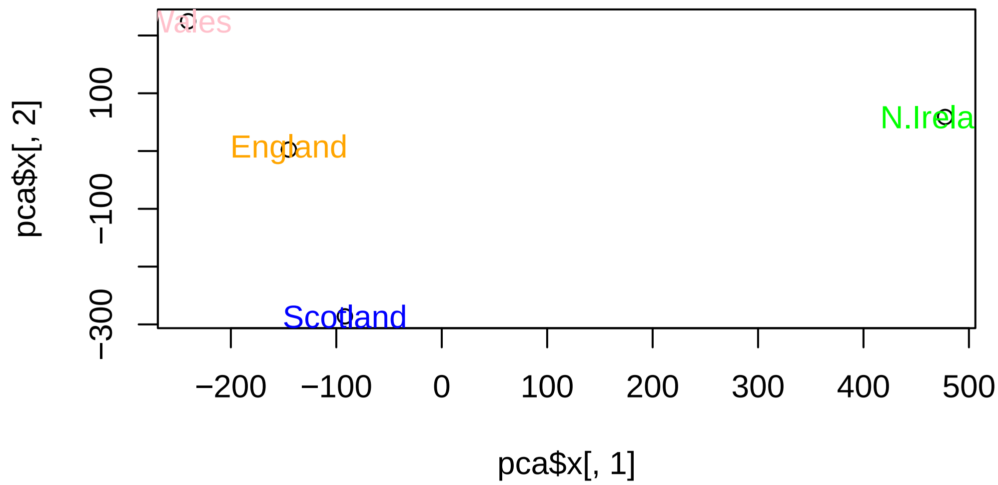
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
plot(x = pca$x[,1], y = pca$x[,2])
text(x = pca$x[,1], y = pca$x[,2], colnames(j))
```



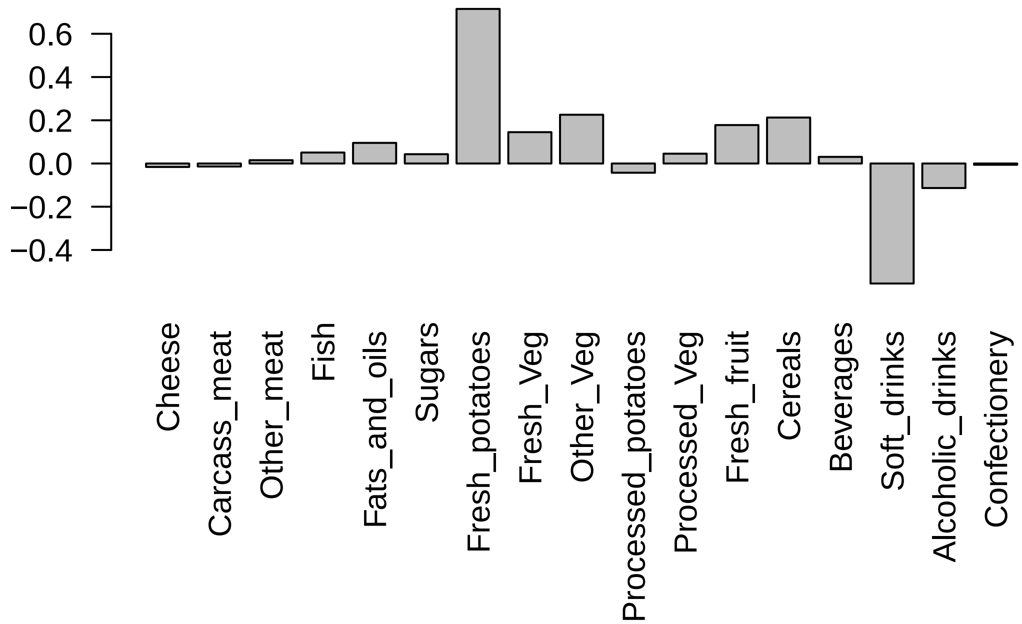
Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
plot(x = pca$x[,1], y = pca$x[,2])
colors_countries <- c('orange', 'pink', 'blue', 'green')
text(x = pca$x[,1], y = pca$x[,2], colnames(j), col = colors_countries)
```



Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2)
```



Q10: How many genes and samples are in this data set?

```
dim(rna.data)
```

```
[1] 100  10
```

I have 100 genes, and 10 samples.