Class 7: Clustering and PCA

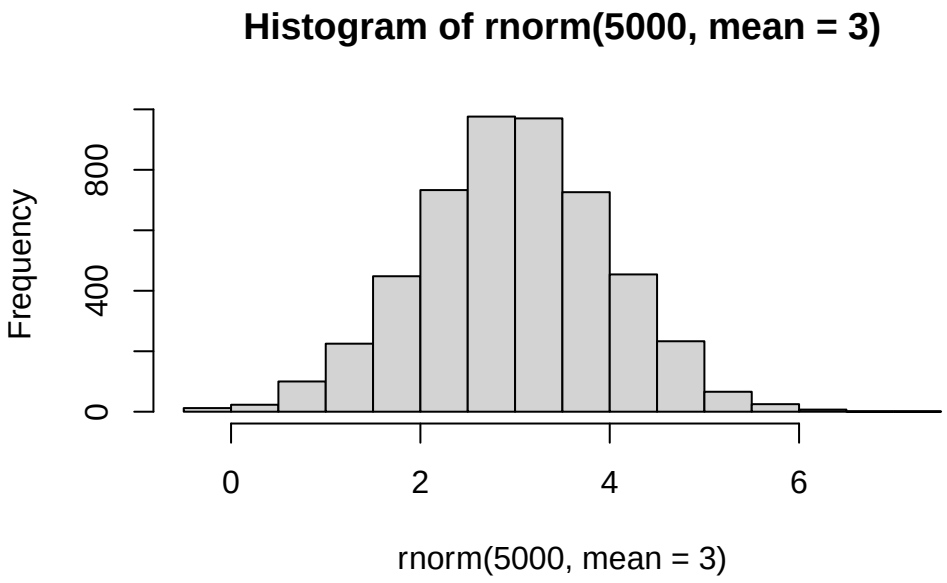
Yipeng Li

Clustering

First let's make up some data to cluster so we can get a feel for these methods and how to work with them.

We can use the `rnorm()` function to get random numbers from a normal distribution around a given `mean`.

```
hist(rnorm(5000, mean=3))
```



Let's get 30 points with a mean of 3.

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

```
 [1]  3.3459531  4.2045087  3.1341435  1.8372251  2.1796406  4.2981072
 [7]  2.1690620  2.7713347  2.0705610  3.2159724  3.2992264  2.9396690
[13]  0.2578273  3.8299785  3.4613033  2.3603114  2.8834384  2.8693134
[19]  2.0474787  4.2295527  3.3856791  3.2494202  2.1095010  2.8882633
[25]  2.8291360  2.5175056  2.5918473  1.9272089  2.4822498  4.6972139
[31] -3.8099443 -3.2410577 -0.7098685 -3.9126381 -2.3862436 -4.3465207
[37] -2.4639012 -3.8578406 -4.8457321 -3.2987615 -3.4062311 -2.9717293
[43] -1.1360644 -3.5682644 -3.8690216 -3.0097373 -3.6284080 -4.4581690
[49] -2.9037750 -4.5628956 -1.6076396 -2.8450474 -3.8666789 -2.7110671
[55] -2.8959718 -3.3818962 -2.8458470 -3.3922494 -4.2598809 -2.6776929
```

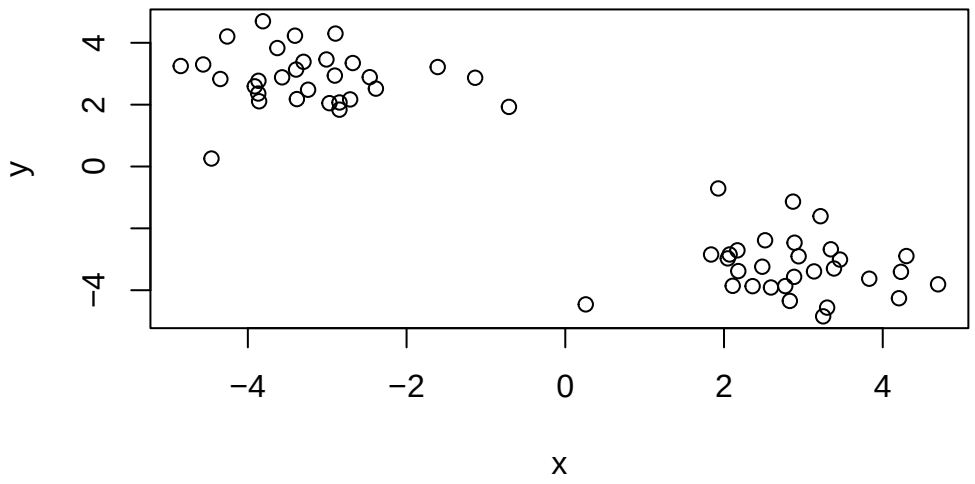
put two of these together:

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

```
              x          y
 [1,]  3.3459531 -2.6776929
 [2,]  4.2045087 -4.2598809
 [3,]  3.1341435 -3.3922494
 [4,]  1.8372251 -2.8458470
 [5,]  2.1796406 -3.3818962
 [6,]  4.2981072 -2.8959718
 [7,]  2.1690620 -2.7110671
 [8,]  2.7713347 -3.8666789
 [9,]  2.0705610 -2.8450474
[10,]  3.2159724 -1.6076396
[11,]  3.2992264 -4.5628956
[12,]  2.9396690 -2.9037750
[13,]  0.2578273 -4.4581690
[14,]  3.8299785 -3.6284080
[15,]  3.4613033 -3.0097373
[16,]  2.3603114 -3.8690216
[17,]  2.8834384 -3.5682644
[18,]  2.8693134 -1.1360644
[19,]  2.0474787 -2.9717293
[20,]  4.2295527 -3.4062311
[21,]  3.3856791 -3.2987615
```

```
[22,]  3.2494202 -4.8457321
[23,]  2.1095010 -3.8578406
[24,]  2.8882633 -2.4639012
[25,]  2.8291360 -4.3465207
[26,]  2.5175056 -2.3862436
[27,]  2.5918473 -3.9126381
[28,]  1.9272089 -0.7098685
[29,]  2.4822498 -3.2410577
[30,]  4.6972139 -3.8099443
[31,] -3.8099443  4.6972139
[32,] -3.2410577  2.4822498
[33,] -0.7098685  1.9272089
[34,] -3.9126381  2.5918473
[35,] -2.3862436  2.5175056
[36,] -4.3465207  2.8291360
[37,] -2.4639012  2.8882633
[38,] -3.8578406  2.1095010
[39,] -4.8457321  3.2494202
[40,] -3.2987615  3.3856791
[41,] -3.4062311  4.2295527
[42,] -2.9717293  2.0474787
[43,] -1.1360644  2.8693134
[44,] -3.5682644  2.8834384
[45,] -3.8690216  2.3603114
[46,] -3.0097373  3.4613033
[47,] -3.6284080  3.8299785
[48,] -4.4581690  0.2578273
[49,] -2.9037750  2.9396690
[50,] -4.5628956  3.2992264
[51,] -1.6076396  3.2159724
[52,] -2.8450474  2.0705610
[53,] -3.8666789  2.7713347
[54,] -2.7110671  2.1690620
[55,] -2.8959718  4.2981072
[56,] -3.3818962  2.1796406
[57,] -2.8458470  1.8372251
[58,] -3.3922494  3.1341435
[59,] -4.2598809  4.2045087
[60,] -2.6776929  3.3459531
```

```r
plot(x)
```



k-means clustering.

Very popular clusterring method that we can use with the `kmeans()` function in base R.

```
km <- kmeans(x, centers=2)
km
```

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

Cluster means:
          x         y
1 -3.229026  2.869421
2  2.869421 -3.229026

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

Within cluster sum of squares by cluster:
[1] 49.99952 49.99952
 (between_SS / total_SS =  91.8 %)
```

```
Available components:

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

Q. Cluster exercise

-cluster size

```
km$size
```

```
[1] 30 30
```

-cluster assignment

```
km$cluster
```

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

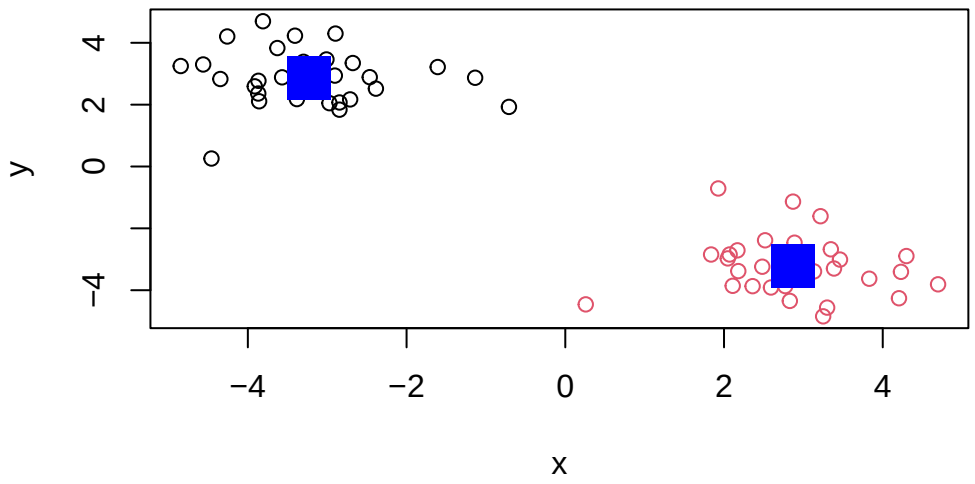
- cluster center

```
km$centers
```

```
          x         y
1 -3.229026  2.869421
2  2.869421 -3.229026
```

Q. plot x colored by the means cluster assignment and add cluster centers as blue points

```
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=15, cex=3)
```



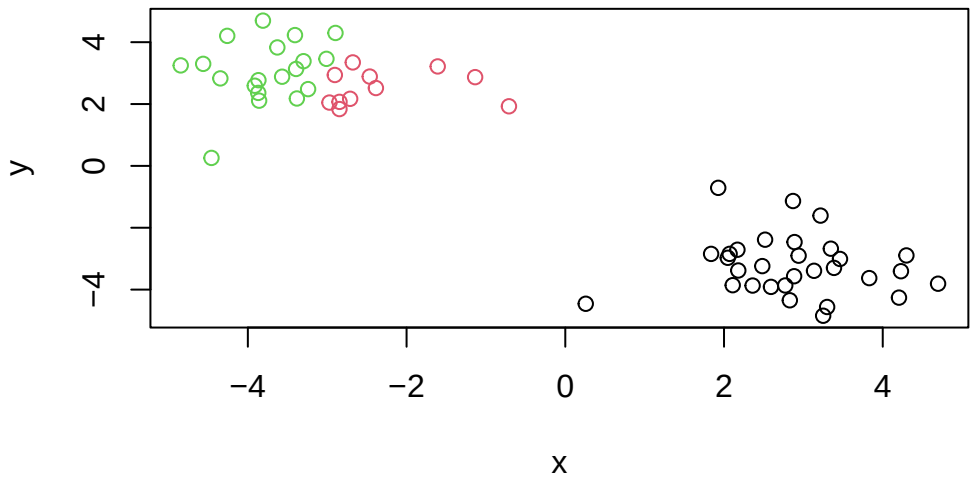
Q. let's cluster into 3 groups for same x data and make a plot.

```
km <- kmeans(x, centers=3)
km
```

```
K-means clustering with 3 clusters of sizes 30, 11, 19

Cluster means:
          x         y
1  2.869421 -3.229026
2 -2.296261  2.529837
3 -3.769047  3.066022

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 3 3 2 3 2 3 2 3
[39] 3 3 3 2 2 3 3 3 3 3 2 3 2 2 3 2 3 3 2 3 3 2

Within cluster sum of squares by cluster:
[1] 49.999524  9.084483 23.800782
 (between_SS / total_SS =  93.2 %)

Available components:
```

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

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



Hierarchical Clustering

We can use the `hclust()` function for hierarchical Clustering. Unlike `kmeans()`, where we could just pass in our data as input, we need to give `hclust()` a "distance matrix"

We will use the `dist()` function to start with

```
d <- dist(x)
d
```

```
           1         2         3         4         5         6         7
2  1.8001212
3  0.7452880 1.3778483
4  1.5180698 2.7574487 1.4073212
```

5	1.3624196	2.2070224	0.9545591	0.6360795				
6	0.9768537	1.3671169	1.2653469	2.4613925	2.1734819			
7	1.1773642	2.5577074	1.1812670	0.3581638	0.6709126	2.1370595		
8	1.3205582	1.4861344	0.5972551	1.3837119	0.7649289	1.8092281	1.3031389	
9	1.2863252	2.5603685	1.1960928	0.2333372	0.5478184	2.2281283	0.1662925	
10	1.0779189	2.8304748	1.7864848	1.8531329	2.0547434	1.6825028	1.5210436	
11	1.8857817	0.9546486	1.1822288	2.2551505	1.6273390	1.9432956	2.1694563	
12	0.4649514	1.8544116	0.5257638	1.1039648	0.8979105	1.3584606	0.7943371	
13	3.5646341	3.9516595	3.0674713	2.2570068	2.2026642	4.3317805	2.5894369	
14	1.0668363	0.7341872	0.7348177	2.1409034	1.6686471	0.8692567	1.8974082	
15	0.3515098	1.4543773	0.5033378	1.6323265	1.3346017	0.8445019	1.3263074	
16	1.5462062	1.8851617	0.9089158	1.1491325	0.5195508	2.1683816	1.1736418	
17	1.0035125	1.4911607	0.3063240	1.2713965	0.7280552	1.5662903	1.1158498	
18	1.6136306	3.3972011	2.2716746	1.9971386	2.3493423	2.2668758	1.7236546	
19	1.3313502	2.5123919	1.1651943	0.2450570	0.4309335	2.2519032	0.2876236	
20	1.1452145	0.8540171	1.0954984	2.4570840	2.0500566	0.5148439	2.1745977	
21	0.6223378	1.2626291	0.2683470	1.6133323	1.2089005	0.9973788	1.3511261	
22	2.1701872	1.1204533	1.4580469	2.4482311	1.8130758	2.2138901	2.3924817	
23	1.7092578	2.1332355	1.1254632	1.0479815	0.4810848	2.3906461	1.1483192	
24	0.5051602	2.2266668	0.9603579	1.1182860	1.1596814	1.4745660	0.7604876	
25	1.7470221	1.3780989	1.0018301	1.7988632	1.1629036	2.0644536	1.7636344	
26	0.8782186	2.5212093	1.1799534	0.8209853	1.0514167	1.8521245	0.4763646	
27	1.4469848	1.6496225	0.7515914	1.3067126	0.6720129	1.9861856	1.2737819	
28	2.4259366	4.2176632	2.9414042	2.1378731	2.6839251	3.2249352	2.0157601	
29	1.0311950	2.0010438	0.6691968	0.7564710	0.3337781	1.8483567	0.6156108	
30	1.7629234	0.6672341	1.6179178	3.0181152	2.5537033	0.9973123	2.7566435	
31	10.2759972	12.0191931	10.6611337	9.4227538	10.0571938	11.1084188	9.5200391	
32	8.3674201	10.0445400	8.6690790	7.3605415	7.9857484	9.2609003	7.4993290	
33	6.1363515	7.9013405	6.5630073	5.4101523	6.0444900	6.9529053	5.4591066	
34	8.9696822	10.6223467	9.2448117	7.9138768	8.5323778	9.8758542	8.0689514	
35	7.7361596	9.4536227	8.0870191	6.8266565	7.4599088	8.6015279	6.9346075	
36	9.4604079	11.1073968	9.7296441	8.3931010	9.0093120	10.3685318	8.5525826	
37	8.0457613	9.7756665	8.4132600	7.1679640	7.8023958	8.8984343	7.2675201	
38	8.6493855	10.2747507	8.8970276	7.5491223	8.1612881	9.5694432	7.7176053	
39	10.1111017	11.7599516	10.3822055	9.0451202	9.6607567	11.0170614	9.2051478	
40	8.9953717	10.7123130	9.3446552	8.0752880	8.7070641	9.8575632	8.1894693	
41	9.6592978	11.4014842	10.0433245	8.8065381	9.4411926	10.4942808	8.9025893	
42	7.8892558	9.5541183	8.1775500	6.8608074	7.4842985	8.7913722	7.0051048	
43	7.1314627	8.9077007	7.5790398	6.4423217	7.0761233	7.9226722	6.4857156	
44	8.8731384	10.5566572	9.1818586	7.8768031	8.5025190	9.7612184	8.0134515	
45	8.7998493	10.4407297	9.0629066	7.7243340	8.3402195	9.7123893	7.8852605	
46	8.8364062	10.5670256	9.2042629	7.9544446	8.5883073	9.6860486	8.0571780	
47	9.5389464	11.2605687	9.8940728	8.6278497	9.2598361	10.3955784	8.7404768	

48	8.3379614	9.7699371	8.4241479	7.0188874	7.5702115	9.3069233	7.2618540
49	8.4032051	10.1173720	8.7492657	7.4799250	8.1119234	9.2694019	7.5937141
50	9.9132968	11.5761600	10.1990323	8.8726249	9.4920595	10.8119310	9.0245710
51	7.6989202	9.4693954	8.1334557	6.9722842	7.6075858	8.4990416	7.0280347
52	7.8022049	9.4747419	8.0989518	6.7893109	7.4146327	8.7000635	6.9285828
53	9.0395775	10.7043011	9.3274478	8.0054513	8.6267160	9.9389182	8.1539498
54	7.7574820	9.4422717	8.0681269	6.7702492	7.3981186	8.6477218	6.9015448
55	9.3607379	11.1200712	9.7726084	8.5696695	9.2056664	10.1739641	8.6477218
56	8.2980507	9.9509284	8.5734900	7.2453263	7.8652008	9.2056664	7.3981186
57	7.6630851	9.3210630	7.9440349	6.6228641	7.2453263	8.5696695	6.7702492
58	8.8983603	10.6010532	9.2297133	7.9440349	8.5734900	9.7726084	8.0681269
59	10.2573588	11.9704545	10.6010532	9.3210630	9.9509284	11.1200712	9.4422717
60	8.5187219	10.2573588	8.8983603	7.6630851	8.2980507	9.3607379	7.7574820

	8	9	10	11	12	13	14
2							
3							
4							
5							
6							
7							
8							
9	1.2388764						
10	2.3023817	1.6861630					
11	0.8737205	2.1120183	2.9564285				
12	0.9775072	0.8710900	1.3252587	1.6976345			
13	2.5821658	2.4265541	4.1080580	3.0432017	3.0997445		
14	1.0851266	1.9259293	2.1119915	1.0746929	1.1479302	3.6672561	
15	1.1001843	1.4004596	1.4233992	1.5615920	0.5322878	3.5157094	0.7201908
16	0.4110300	1.0641797	2.4178512	1.1674856	1.1257692	2.1834684	1.4892335
17	0.3187765	1.0880315	1.9886248	1.0780404	0.6668643	2.7723210	0.9484489
18	2.7323717	1.8864327	0.5852825	3.4536933	1.7691101	4.2256643	2.6710772
19	1.1510440	0.1287676	1.7961399	2.0245204	0.8947745	2.3264469	1.8996137
20	1.5291867	2.2307337	2.0645281	1.4843786	1.3842912	4.1086709	0.4571894
21	0.8366297	1.3911837	1.6996157	1.2670869	0.5957678	3.3358182	0.5532348
22	1.0895462	2.3221646	3.2382653	0.2871883	1.9665053	3.0165931	1.3486756
23	0.6618927	1.0135415	2.5075254	1.3829494	1.2646818	1.9465585	1.7357079
24	1.4076425	0.9021693	0.9168300	2.1388474	0.4428673	3.3009540	1.4976326
25	0.4833107	1.6822183	2.7660644	0.5174970	1.4469737	2.5737315	1.2318164
26	1.5020379	0.6405158	1.0459828	2.3127693	0.6678778	3.0657823	1.8070853
27	0.1852781	1.1880611	2.3880013	0.9608434	1.0671385	2.3969259	1.2703368
28	3.2677210	2.1399857	1.5706382	4.0900184	2.4162577	4.1032416	3.4840214
29	0.6891821	0.5712371	1.7906433	1.5539325	0.5683238	2.5356292	1.4022884
30	1.9267147	2.7982730	2.6540954	1.5878617	1.9773990	4.4864629	0.8860320

31	10.8006247	9.5637884	9.4400574	11.6743281	10.1652502	10.0183732	11.2997522
32	8.7440127	7.5228577	7.6433260	9.6129803	8.1982099	7.7724905	9.3455706
33	6.7592833	5.5231529	5.2827436	7.6285188	6.0545462	6.4582881	7.1746142
34	9.2945173	8.0844601	8.2736194	10.1588058	8.7838475	8.1911849	9.9317514
35	8.2072179	6.9728101	6.9571292	9.0805644	7.5997124	7.4599696	8.7414913
36	9.7722977	8.5659381	8.7679120	10.6348287	9.2711828	8.6200252	10.4189737
37	8.5461651	7.3097331	7.2439014	9.4198412	7.9212549	7.8344031	9.0597972
38	8.9252838	7.7261566	7.9909927	9.7849111	8.4462461	7.7506781	9.5930267
39	10.4239423	9.2183320	9.4118069	11.2859483	9.9234209	9.2440927	11.0712454
40	9.4574185	8.2250578	8.2082269	10.3302122	8.8586256	8.6125070	10.0008175
41	10.1838738	8.9467993	8.8275927	11.0575906	9.5475030	9.4287768	10.6822411
42	8.2437882	7.0257742	7.1866225	9.1116373	7.7110014	7.2631596	8.8588328
43	7.7872563	6.5525847	6.2436634	8.6550295	7.0668347	7.4588828	8.1781395
44	9.2603779	8.0381528	8.1360709	10.1296752	8.7089055	8.2787786	9.8558683
45	9.1032818	7.8977466	8.1204541	9.9656698	8.6063275	7.9701039	9.7539408
46	9.3338157	8.0981164	8.0283026	10.2073890	8.7126148	8.5670893	9.8511785
47	10.0097573	8.7769139	8.7414663	10.8826676	9.4065440	9.1540271	10.5477512
48	8.3232971	7.2285646	7.8976207	9.1332662	8.0451063	6.6694258	9.1540271
49	8.8618983	7.6293487	7.6242589	9.7347677	8.2638778	8.0451063	9.4065440
50	10.2538351	9.0418387	9.1971801	11.1187196	9.7347677	9.1332662	10.8826676
51	8.3270262	7.0897899	6.8216175	9.1971801	7.6242589	7.8976207	8.7414663
52	8.1727942	6.9517200	7.0897899	9.0418387	7.6293487	7.2285646	8.7769139
53	9.3875688	8.1727942	8.3270262	10.2538351	8.8618983	8.3232971	10.0097573
54	8.1539498	6.9285828	7.0280347	9.0245710	7.5937141	7.2618540	8.7404768
55	9.9389182	8.7000635	8.4990416	10.8119310	9.2694019	9.3069233	10.3955784
56	8.6267160	7.4146327	7.6075858	9.4920595	8.1119234	7.5702115	9.2598361
57	8.0054513	6.7893109	6.9722842	8.8726249	7.4799250	7.0188874	8.6278497
58	9.3274478	8.0989518	8.1334557	10.1990323	8.7492657	8.4241479	9.8940728
59	10.7043011	9.4747419	9.4693954	11.5761600	10.1173720	9.7699371	11.2605687
60	9.0395775	7.8022049	7.6989202	9.9132968	8.4032051	8.3379614	9.5389464

	15	16	17	18	19	20	21
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							

```
14
15
16  1.3966219
17  0.8036668  0.6034209
18  1.9649687  2.7799529  2.4322409
19  1.4143354  0.9502619  1.0269775  2.0112379
20  0.8645314  1.9256787  1.3558314  2.6464897  2.2249133
21  0.2987541  1.1732756  0.5699803  2.2234865  1.3775814  0.8506894
22  1.8481806  1.3207869  1.3288591  3.7290904  2.2263311  1.7415002  1.5529599
23  1.5958222  0.2510594  0.8263373  2.8258415  0.8882792  2.1676186  1.3932695
24  0.7913987  1.5010318  1.1043737  1.3279720  0.9822464  1.6392203  0.9718097
25  1.4787243  0.6691801  0.7801485  3.2107077  1.5814676  1.6868052  1.1863978
26  1.1311492  1.4910871  1.2373682  1.2987365  0.7508121  1.9928573  1.2595294
27  1.2534686  0.2356083  0.4512412  2.7904029  1.0870356  1.7142133  1.0035006
28  2.7645691  3.1887029  3.0141005  1.0340232  2.2650562  3.5455830  2.9714479
29  1.0060094  0.6396934  0.5177031  2.1402838  0.5114330  1.7550925  0.9052703
30  1.4723472  2.3376492  1.8298063  3.2389587  2.7791548  0.6178117  1.4076333
31 10.5956660 10.5571040 10.6357653  8.8678983  9.6499790 11.4148733 10.7569800
32  8.6650773  8.4684109  8.6091912  7.1013261  7.5970063  9.5123198  8.7939600
33  6.4631348  6.5591381  6.5659790  4.7110706  5.6216153  7.2693510  6.6395991
34  9.2602787  9.0051497  9.1724386  7.7390046  8.1533046 10.1129726  9.3789501
35  8.0463792  7.9572302  8.0502263  6.4007385  7.0561741  8.8802825  8.1941475
36  9.7495927  9.4787611  9.6539650  8.2335337  8.6332651 10.6032466  9.8660044
37  8.3602906  8.3026457  8.3833639  6.6811969  7.3954081  9.1881982  8.5145091
38  8.9317674  8.6260157  8.8137315  7.4691563  7.7904876  9.7892406  9.0397943
39 10.4011485 10.1291301 10.3063524  8.8743678  9.2854217 11.2542654 10.5183085
40  9.3059028  9.2008580  9.3046722  7.6479613  8.3065592 10.1393076  9.4532265
41  9.9784943  9.9418267 10.0182780  8.2566523  9.0333501 10.7986291 10.1393076
42  8.1828688  7.9646490  8.1129254  6.6522723  7.0982320  9.0333501  8.3065592
43  7.4631781  7.5914295  7.5893880  5.6644597  6.6522723  8.2566523  7.6479613
44  9.1730224  8.9857513  9.1240855  7.5893880  8.1129254 10.0182780  9.3046722
45  9.0868633  8.8096072  8.9857513  7.5914295  7.9646490  9.9418267  9.2008580
46  9.1514333  9.0868633  9.1730224  7.4631781  8.1828688  9.9784943  9.3059028
47  9.8511785  9.7539408  9.8558683  8.1781395  8.8588328 10.6822411 10.0008175
48  8.5670893  7.9701039  8.2787786  7.4588828  7.2631596  9.4287768  8.6125070
49  8.7126148  8.6063275  8.7089055  7.0668347  7.7110014  9.5475030  8.8586256
50 10.2073890  9.9656698 10.1296752  8.6550295  9.1116373 11.0575906 10.3302122
51  8.0283026  8.1204541  8.1360709  6.2436634  7.1866225  8.8275927  8.2082269
52  8.0981164  7.8977466  8.0381528  6.5525847  7.0257742  8.9467993  8.2250578
53  9.3338157  9.1032818  9.2603779  7.7872563  8.2437882 10.1838738  9.4574185
54  8.0571780  7.8852605  8.0134515  6.4857156  7.0051048  8.9025893  8.1894693
55  9.6860486  9.7123893  9.7612184  7.9226722  8.7913722 10.4942808  9.8575632
56  8.5883073  8.3402195  8.5025190  7.0761233  7.4842985  9.4411926  8.7070641
```

57	7.9544446	7.7243340	7.8768031	6.4423217	6.8608074	8.8065381	8.0752880
58	9.2042629	9.0629066	9.1818586	7.5790398	8.1775500	10.0433245	9.3446552
59	10.5670256	10.4407297	10.5566572	8.9077007	9.5541183	11.4014842	10.7123130
60	8.8364062	8.7998493	8.8731384	7.1314627	7.8892558	9.6592978	8.9953717

	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	1.5084248						
24	2.4090564	1.5967271					
25	0.6525725	0.8698751	1.8835478				
26	2.5660832	1.5271102	0.3788033	1.9848930			
27	1.1415195	0.4854490	1.4787497	0.4945302	1.5282038		
28	4.3420745	3.1532458	2.0000641	3.7468270	1.7772686	3.2710056	
29	1.7786316	0.7206683	0.8768233	1.1586106	0.8555408	0.6804644	2.5913296
30	1.7801581	2.5881561	2.2548025	1.9436125	2.6034693	2.1078697	4.1573306
31	11.8702336	10.4033067	9.8054859	11.2190251	9.4979993	10.7290487	7.8836201
32	9.7890563	8.2960970	7.8761022	9.1367039	7.5407744	8.6554821	6.0745863
33	7.8452977	6.4354986	5.6770063	7.2030713	5.3871899	6.7085870	3.7293906
34	10.3253410	8.8240939	8.4742464	9.6743200	8.1319208	9.1987314	6.7085870
35	9.2724310	7.8010742	7.2549870	8.6206172	6.9349486	8.1319208	5.3871899
36	10.7982369	9.2949380	8.9642815	10.1479111	8.6206172	9.6743200	7.2030713
37	9.6154421	8.1502102	7.5691036	8.9642815	7.2549870	8.4742464	5.6770063
38	9.9442659	8.4390955	8.1502102	9.2949380	7.8010742	8.8240939	6.4354986
39	11.4482742	9.9442659	9.6154421	10.7982369	9.2724310	10.3253410	7.8452977

```
40 10.5183085  9.0397943  8.5145091  9.8660044  8.1941475  9.3789501  6.6395991
41 11.2542654  9.7892406  9.1881982 10.6032466  8.8802825 10.1129726  7.2693510
42  9.2854217  7.7904876  7.3954081  8.6332651  7.0561741  8.1533046  5.6216153
43  8.8743678  7.4691563  6.6811969  8.2335337  6.4007385  7.7390046  4.7110706
44 10.3063524  8.8137315  8.3833639  9.6539650  8.0502263  9.1724386  6.5659790
45 10.1291301  8.6260157  8.3026457  9.4787611  7.9572302  9.0051497  6.5591381
46 10.4011485  8.9317674  8.3602906  9.7495927  8.0463792  9.2602787  6.4631348
47 11.0712454  9.5930267  9.0597972 10.4189737  8.7414913  9.9317514  7.1746142
48  9.2440927  7.7506781  7.8344031  8.6200252  7.4599696  8.1911849  6.4582881
49  9.9234209  8.4462461  7.9212549  9.2711828  7.5997124  8.7838475  6.0545462
50 11.2859483  9.7849111  9.4198412 10.6348287  9.0805644 10.1588058  7.6285188
51  9.4118069  7.9909927  7.2439014  8.7679120  6.9571292  8.2736194  5.2827436
52  9.2183320  7.7261566  7.3097331  8.5659381  6.9728101  8.0844601  5.5231529
53 10.4239423  8.9252838  8.5461651  9.7722977  8.2072179  9.2945173  6.7592833
54  9.2051478  7.7176053  7.2675201  8.5525826  6.9346075  8.0689514  5.4591066
55 11.0170614  9.5694432  8.8984343 10.3685318  8.6015279  9.8758542  6.9529053
56  9.6607567  8.1612881  7.8023958  9.0093120  7.4599088  8.5323778  6.0444900
57  9.0451202  7.5491223  7.1679640  8.3931010  6.8266565  7.9138768  5.4101523
58 10.3822055  8.8970276  8.4132600  9.7296441  8.0870191  9.2448117  6.5630073
59 11.7599516 10.2747507  9.7756665 11.1073968  9.4536227 10.6223467  7.9013405
60 10.1111017  8.6493855  8.0457613  9.4604079  7.7361596  8.9696822  6.1363515
           29         30         31         32         33         34         35
```

```
23
24
25
26
27
28
29
30  2.2868533
31 10.1295539 12.0309385
32  8.0939791 10.1295539  2.2868533
33  6.0745863  7.8836201  4.1573306  2.5913296
34  8.6554821 10.7290487  2.1078697  0.6804644  3.2710056
35  7.5407744  9.4979993  2.6034693  0.8555408  1.7772686  1.5282038
36  9.1367039 11.2190251  1.9436125  1.1586106  3.7468270  0.4945302  1.9848930
37  7.8761022  9.8054859  2.2548025  0.8768233  2.0000641  1.4787497  0.3788033
38  8.2960970 10.4033067  2.5881561  0.7206683  3.1532458  0.4854490  1.5271102
39  9.7890563 11.8702336  1.7801581  1.7786316  4.3420745  1.1415195  2.5660832
40  8.7939600 10.7569800  1.4076333  0.9052703  2.9714479  1.0035006  1.2595294
41  9.5123198 11.4148733  0.6178117  1.7550925  3.5455830  1.7142133  1.9928573
42  7.5970063  9.6499790  2.7791548  0.5114330  2.2650562  1.0870356  0.7508121
43  7.1013261  8.8678983  3.2389587  2.1402838  1.0340232  2.7904029  1.2987365
44  8.6091912 10.6357653  1.8298063  0.5177031  3.0141005  0.4512412  1.2373682
45  8.4684109 10.5571040  2.3376492  0.6396934  3.1887029  0.2356083  1.4910871
46  8.6650773 10.5956660  1.4723472  1.0060094  2.7645691  1.2534686  1.1311492
47  9.3455706 11.2997522  0.8860320  1.4022884  3.4840214  1.2703368  1.8070853
48  7.7724905 10.0183732  4.4864629  2.5356292  4.1032416  2.3969259  3.0657823
49  8.1982099 10.1652502  1.9773990  0.5683238  2.4162577  1.0671385  0.6678778
50  9.6129803 11.6743281  1.5878617  1.5539325  4.0900184  0.9608434  2.3127693
51  7.6433260  9.4400574  2.6540954  1.7906433  1.5706382  2.3880013  1.0459828
52  7.5228577  9.5637884  2.7982730  0.5712371  2.1399857  1.1880611  0.6405158
53  8.7440127 10.8006247  1.9267147  0.6891821  3.2677210  0.1852781  1.5020379
54  7.4993290  9.5200391  2.7566435  0.6156108  2.0157601  1.2737819  0.4763646
55  9.2609003 11.1084188  0.9973123  1.8483567  3.2249352  1.9861856  1.8521245
56  7.9857484 10.0571938  2.5537033  0.3337781  2.6839251  0.6720129  1.0514167
57  7.3605415  9.4227538  3.0181152  0.7564710  2.1378731  1.3067126  0.8209853
58  8.6690790 10.6611337  1.6179178  0.6691968  2.9414042  0.7515914  1.1799534
59 10.0445400 12.0191931  0.6672341  2.0010438  4.2176632  1.6496225  2.5212093
60  8.3674201 10.2759972  1.7629234  1.0311950  2.4259366  1.4469848  0.8782186
          36         37         38         39         40         41         42
2
3
4
5
```

1.8835478							
0.8698751	1.5967271						
0.6525725	2.4090564	1.5084248					
1.1863978	0.9718097	1.3932695	1.5529599				
1.6868052	1.6392203	2.1676186	1.7415002	0.8506894			
1.5814676	0.9822464	0.8882792	2.2263311	1.3775814	2.2249133		
3.2107077	1.3279720	2.8258415	3.7290904	2.2234865	2.6464897	2.0112379	
0.7801485	1.1043737	0.8263373	1.3288591	0.5699803	1.3558314	1.0269775	
0.6691801	1.5010318	0.2510594	1.3207869	1.1732756	1.9256787	0.9502619	
1.4787243	0.7913987	1.5958222	1.8481806	0.2987541	0.8645314	1.4143354	
1.2318164	1.4976326	1.7357079	1.3486756	0.5532348	0.4571894	1.8996137	
2.5737315	3.3009540	1.9465585	3.0165931	3.3358182	4.1086709	2.3264469	

49	1.4469737	0.4428673	1.2646818	1.9665053	0.5957678	1.3842912	0.8947745
50	0.5174970	2.1388474	1.3829494	0.2871883	1.2670869	1.4843786	2.0245204
51	2.7660644	0.9168300	2.5075254	3.2382653	1.6996157	2.0645281	1.7961399
52	1.6822183	0.9021693	1.0135415	2.3221646	1.3911837	2.2307337	0.1287676
53	0.4833107	1.4076425	0.6618927	1.0895462	0.8366297	1.5291867	1.1510440
54	1.7636344	0.7604876	1.1483192	2.3924817	1.3511261	2.1745977	0.2876236
55	2.0644536	1.4745660	2.3906461	2.2138901	0.9973788	0.5148439	2.2519032
56	1.1629036	1.1596814	0.4810848	1.8130758	1.2089005	2.0500566	0.4309335
57	1.7988632	1.1182860	1.0479815	2.4482311	1.6133323	2.4570840	0.2450570
58	1.0018301	0.9603579	1.1254632	1.4580469	0.2683470	1.0954984	1.1651943
59	1.3780989	2.2266668	2.1332355	1.1204533	1.2626291	0.8540171	2.5123919
60	1.7470221	0.5051602	1.7092578	2.1701872	0.6223378	1.1452145	1.3313502
	43	44	45	46	47	48	49

32							
33							
34							
35							
36							
37							
38							
39							
40							
41							
42							
43							
44	2.4322409						
45	2.7799529	0.6034209					
46	1.9649687	0.8036668	1.3966219				
47	2.6710772	0.9484489	1.4892335	0.7201908			
48	4.2256643	2.7723210	2.1834684	3.5157094	3.6672561		
49	1.7691101	0.6668643	1.1257692	0.5322878	1.1479302	3.0997445	
50	3.4536933	1.0780404	1.1674856	1.5615920	1.0746929	3.0432017	1.6976345
51	0.5852825	1.9886248	2.4178512	1.4233992	2.1119915	4.1080580	1.3252587
52	1.8864327	1.0880315	1.0641797	1.4004596	1.9259293	2.4265541	0.8710900
53	2.7323717	0.3187765	0.4110300	1.1001843	1.0851266	2.5821658	0.9775072
54	1.7236546	1.1158498	1.1736418	1.3263074	1.8974082	2.5894369	0.7943371
55	2.2668758	1.5662903	2.1683816	0.8445019	0.8692567	4.3317805	1.3584606
56	2.3493423	0.7280552	0.5195508	1.3346017	1.6686471	2.2026642	0.8979105
57	1.9971386	1.2713965	1.1491325	1.6323265	2.1409034	2.2570068	1.1039648
58	2.2716746	0.3063240	0.9089158	0.5033378	0.7348177	3.0674713	0.5257638
59	3.3972011	1.4911607	1.8851617	1.4543773	0.7341872	3.9516595	1.8544116
60	1.6136306	1.0035125	1.5462062	0.3515098	1.0668363	3.5646341	0.4649514
	50	51	52	53	54	55	56
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  2.9564285
52  2.1120183  1.6861630
53  0.8737205  2.3023817  1.2388764
54  2.1694563  1.5210436  0.1662925  1.3031389
55  1.9432956  1.6825028  2.2281283  1.8092281  2.1370595
56  1.6273390  2.0547434  0.5478184  0.7649289  0.6709126  2.1734819
57  2.2551505  1.8531329  0.2333372  1.3837119  0.3581638  2.4613925  0.6360795
```

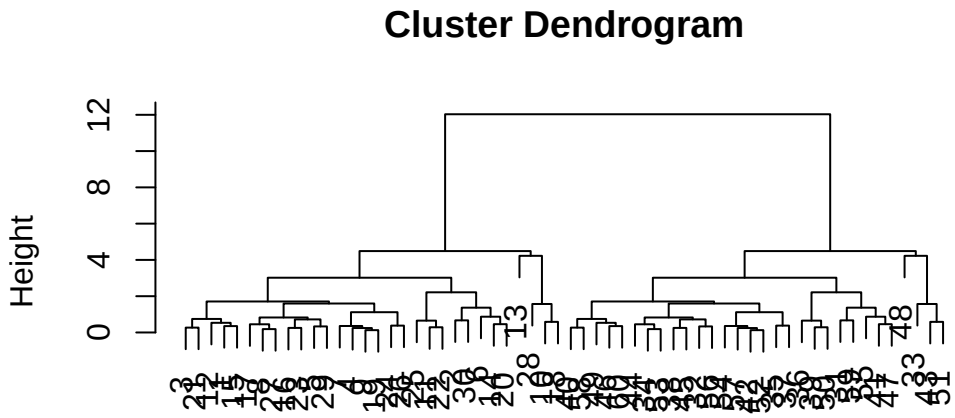
```
58  1.1822288  1.7864848  1.1960928  0.5972551  1.1812670  1.2653469  0.9545591
59  0.9546486  2.8304748  2.5603685  1.4861344  2.5577074  1.3671169  2.2070224
60  1.8857817  1.0779189  1.2863252  1.3205582  1.1773642  0.9768537  1.3624196
           57         58         59
2
3
4
5
6
7
8
9
10
11
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  1.4073212
59  2.7574487  1.3778483
60  1.5180698  0.7452880  1.8001212
```

```
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

```
plot(hc)
```



I can now "cut" my tree with the `cutres()` to yield a cluster membership vector

```
grps <- cutree(hc, h=8)
grps
```

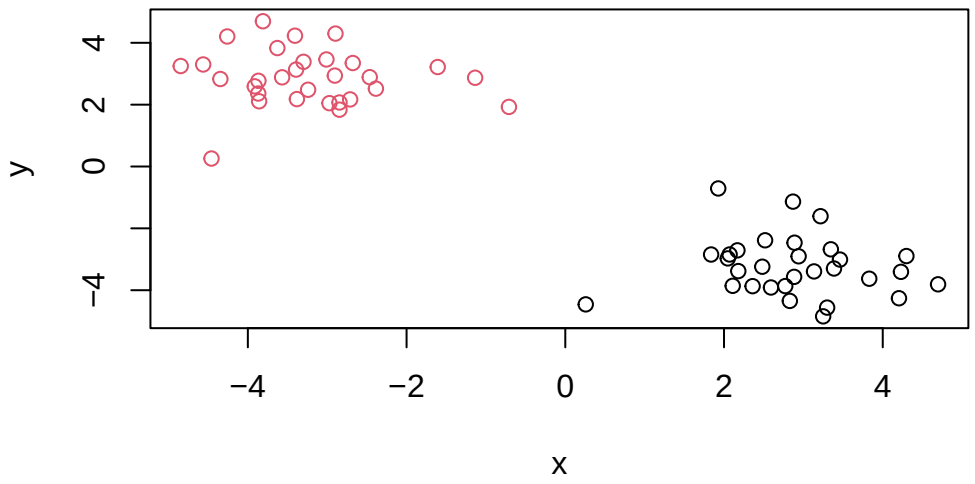
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also tell `cutree()` to cut where it yields "k" groups.

```
cutree(hc, 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 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
```

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



Principal Component Analysis (PCA)

1. PCA of UK food data Suppose that we are examining the following data, from the UK's 'Department for Environment, Food and Rural Affairs' (DEFRA), showing the consumption in grams (per person, per week) of 17 different types of food-stuff measured and averaged in the four countries of the United Kingdom in 1997.

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

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

```
#rows
nrow(x)
```

```
[1] 17
```

```
#columns
ncol(x)
```

```
[1] 5
```

```
#both row and columns
dim(x)
```

```
[1] 17  5
```

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? #Check my data

```
View(x)
```

#To fix the wrong number of columns we can drop column x from the dataframe

```
rownames(x) <- x[,1]
x <- x[,-1]
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
```

#But everytime you run it, you will drop one of the column, so the better way is to read `row.names`.

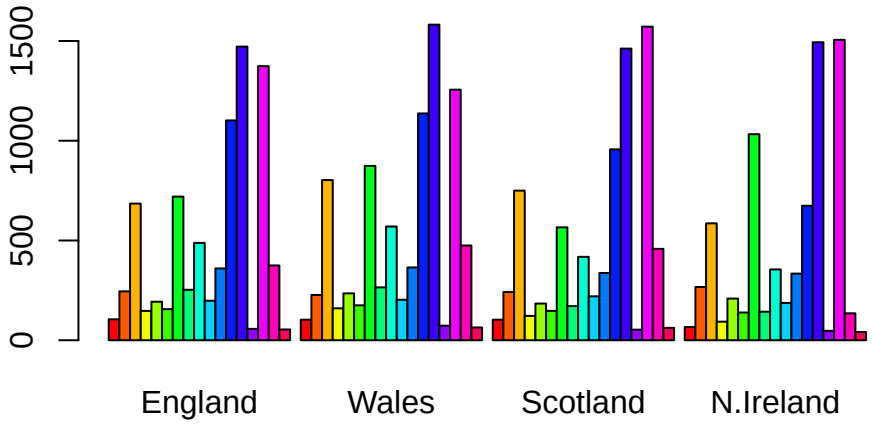
```
x <- read.csv(url, row.names=1)
head(x)
```

```
              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
```

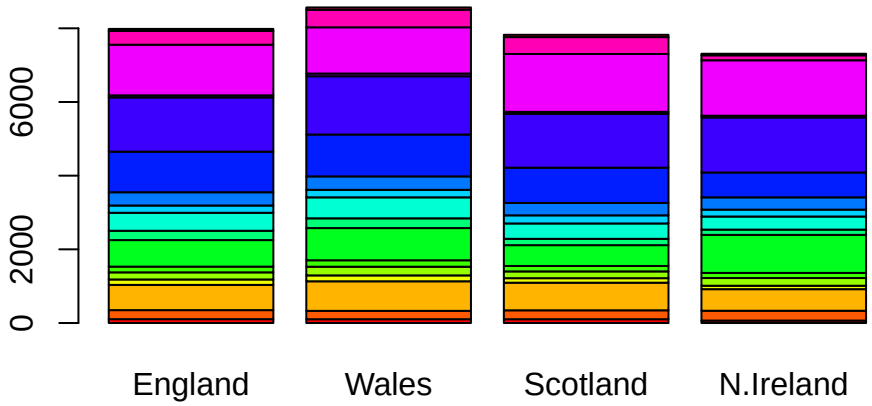
##So use `row.names` during read.csv is better, because -1 will detele one column everytime you run it. Answer: x <- read.csv(url, row.names=1)

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

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




```
# To change it, simply change default the setting
fix_plot <- barplot(as.matrix(x), beside=FALSE, col=rainbow(nrow(x)))
```

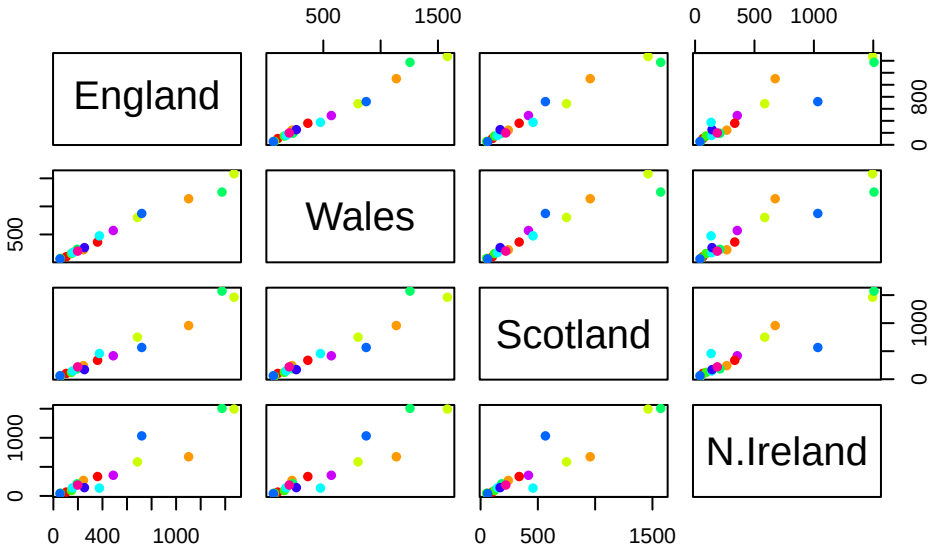



```
fix_plot
```

```
[1] 0.7 1.9 3.1 4.3
```

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(x, col=rainbow(10), pch=16)
```



#the axies are changed, based on the plot format, the row gives the y-axis country and the column gives x-axis country. And England and wales are pretty similar, based on the points (points are aways from each other), we can see Scotland is less similar to England and Wales. And N.Ireland is even more different from England and Wales, and also different from Scotland.

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

#N.Ireland is different from England, Wales, and Scotland. Based on the graph, the point is more far away from each other.

```
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
```

the N. Ireland have way less Alcoholic_drinks, other meat and lots more fresh potatoes.

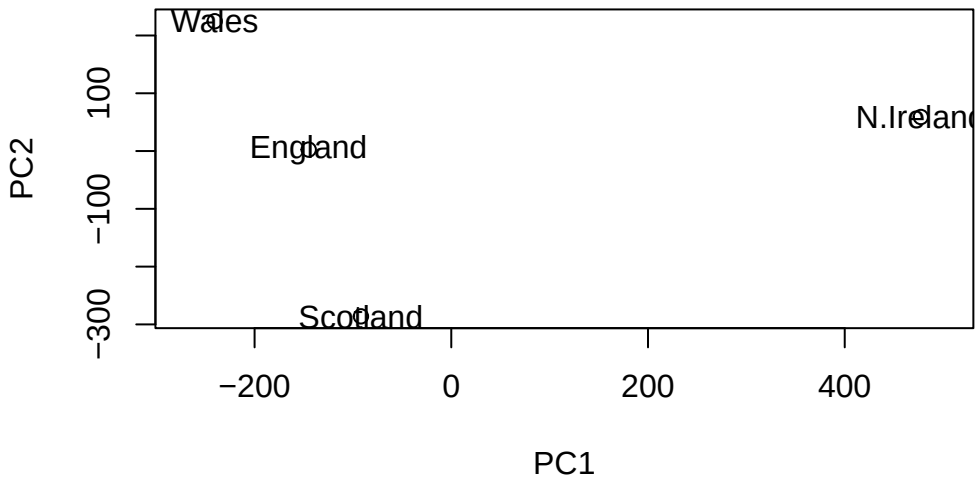
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

The main PCA function in base R is called `prcomp()` it expects the transpose of our data

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

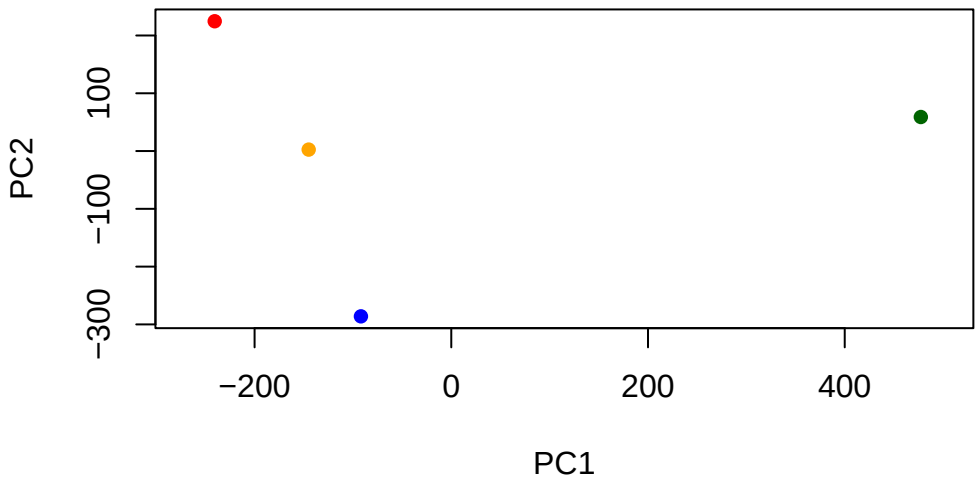
```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

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

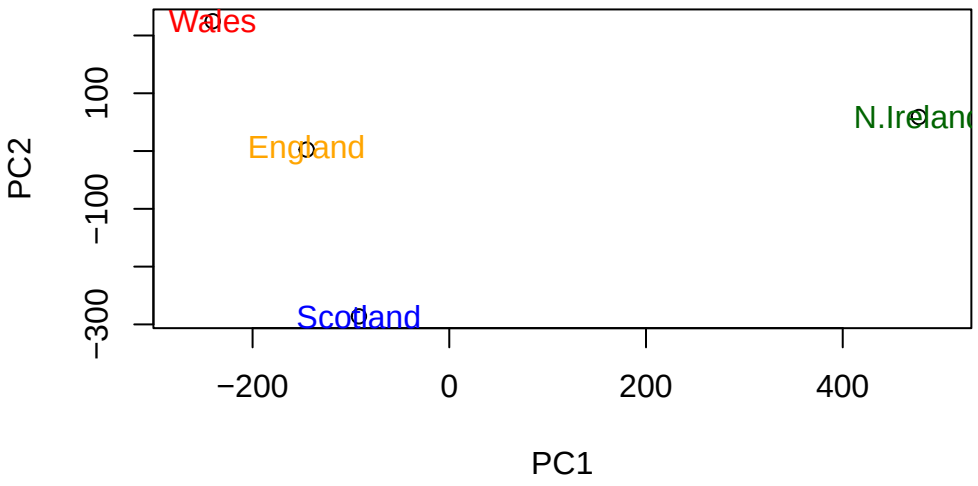


Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
#change the color of the dots
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500), col = c("orange", "re
```




```
#change color of the text
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], labels=colnames(x), col = c("orange", "red", "blue", "darkgreen
```


Wales
England
N.Ireland
Scotland
PC2
100
−100
−300
−200
0
200
400
PC1