class 7: Machine learning 1

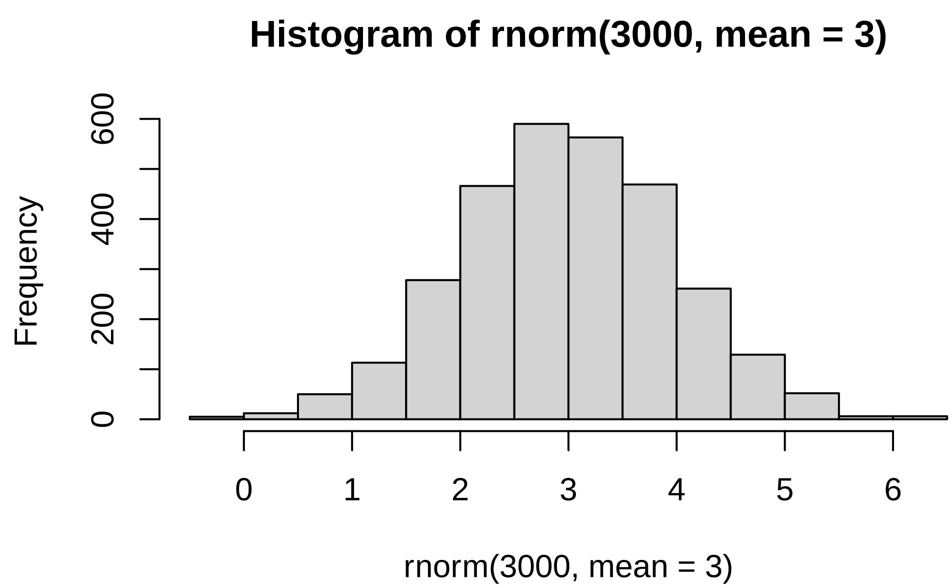
Bryn Baxter (PID: A69038039)

Today we will delve into unsupervised machine learning with an intial focus on clustering and dimensionality reduction.

Let's start by making up some data to cluster:

The `rnorm()` function can help us here.

```
hist( rnorm(3000, mean=3))
```



Lets get some data centered at 3, -3 -3, 3.

```
rnorm(30, mean=3)
```

```
 [1] 2.609279 2.451696 2.912795 4.367789 1.824336 2.687714 2.802145 4.147380
 [9] 3.493030 4.382111 3.618325 4.577126 3.793580 2.453449 2.742733 3.092380
[17] 3.659383 2.236772 3.567003 3.351578 3.459404 2.587295 4.788598 3.442584
[25] 1.653610 3.234214 2.134520 2.513011 2.565129 2.044085
```

```r
rnorm(30, mean=-3)
```

```
 [1] -2.083751 -3.309044 -2.874853 -5.267329 -2.232342 -2.303351 -5.519375
 [8] -3.593790 -3.646662 -1.554001 -3.298173 -3.096396 -2.210474 -4.108582
[15] -2.683395 -3.108678 -4.032385 -4.044795 -1.242615 -3.214500 -3.199939
[22] -3.626888 -3.440742 -4.653416 -4.817898 -2.398877 -2.648846 -2.297548
[29] -3.020069 -5.950576
```

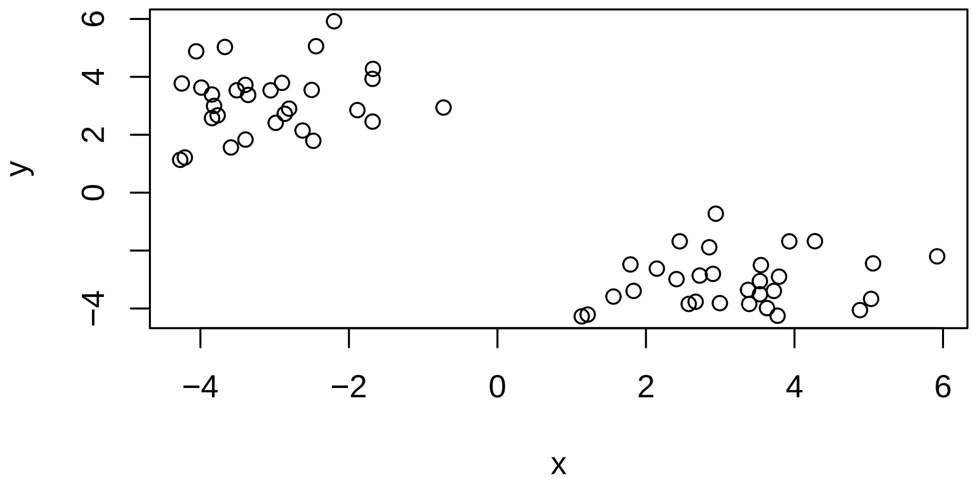
Lets put them together into a vector

```r
#Combine 30+3 values with the 30-3 values
x <- c(rnorm(30, mean=3), rnorm(30, mean=-3))

#Bind these values together
z <- cbind(x=x, y=rev(x))
head(z)
```

```
            x         y
[1,] 1.790954 -2.479328
[2,] 3.535804 -3.510345
[3,] 2.851825 -1.886156
[4,] 5.921131 -2.199676
[5,] 3.390454 -3.844894
[6,] 2.411287 -2.985943
```

now lets try plotting it

```r
plot(z)
```



K-means

Now we can see how K-means clusters this data. The main function of K-means clustering in "base R" is called `kmeans()`.

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

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

Cluster means:
          x         y
1 -3.049135  3.186829
2  3.186829 -3.049135

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 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] 62.67768 62.67768
 (between_SS / total_SS =  90.3 %)
```

```
Available components:

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

```r
attributes(km
           )
```

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

$class
[1] "kmeans"
```

> Q. what size is each cluster?

```r
km$size
```

```
[1] 30 30
```

> Q. the cluster membership vector (i.e. the answer: cluster to wich each point is allocated)

```r
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 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 1 1
```

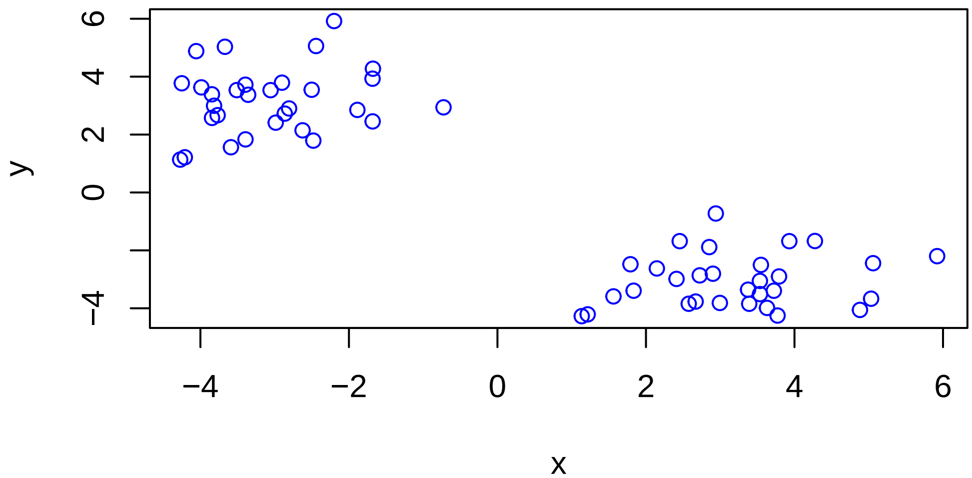
> Q. Cluster centers

```r
km$centers
```

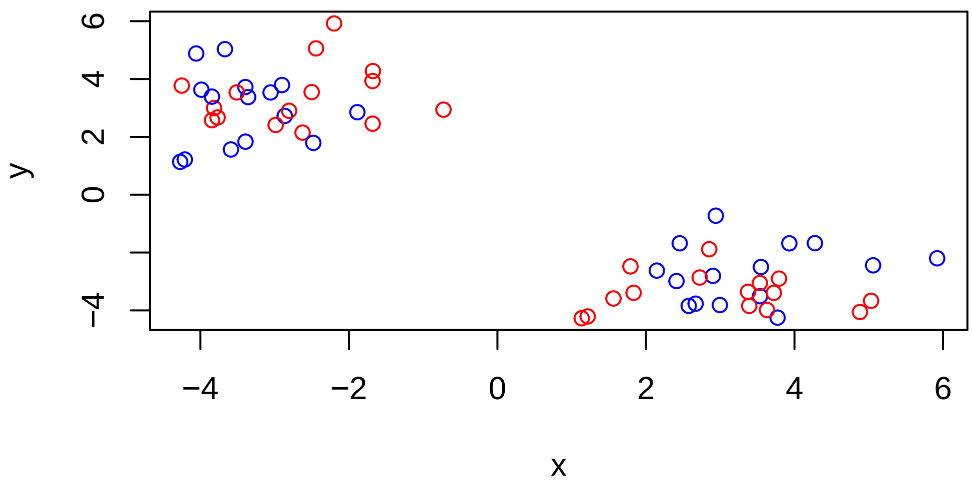
```
          x         y
1 -3.049135  3.186829
2  3.186829 -3.049135
```

> Q. Make a results figure, i.e. plot the data `z` colored by cluster membership and show the cluster centers.

```r
plot(z, col="blue")
```

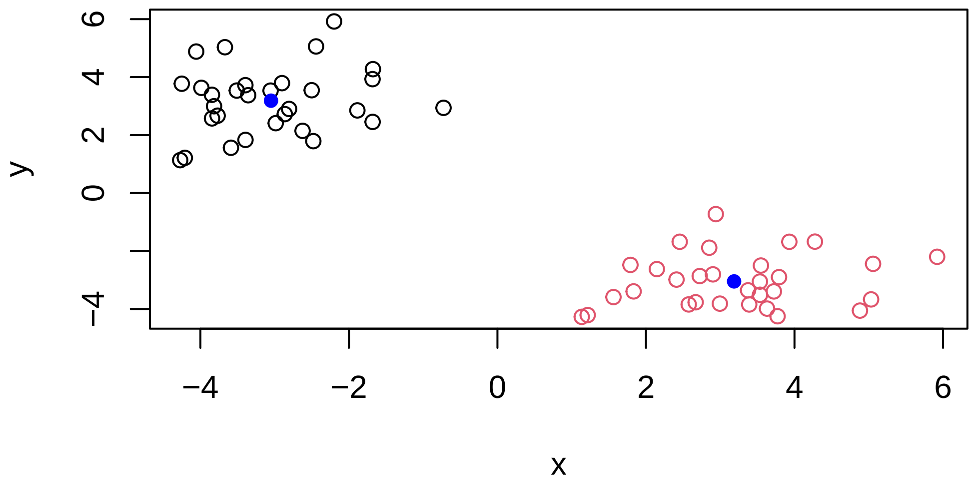



```r
plot(z, col=c("red", "blue"))
```



You can specify color based on a number, where 1 is black and 2 is red

```
plot(z, col=km$cluster )
points(km$centers, col="blue", pch=16)
```



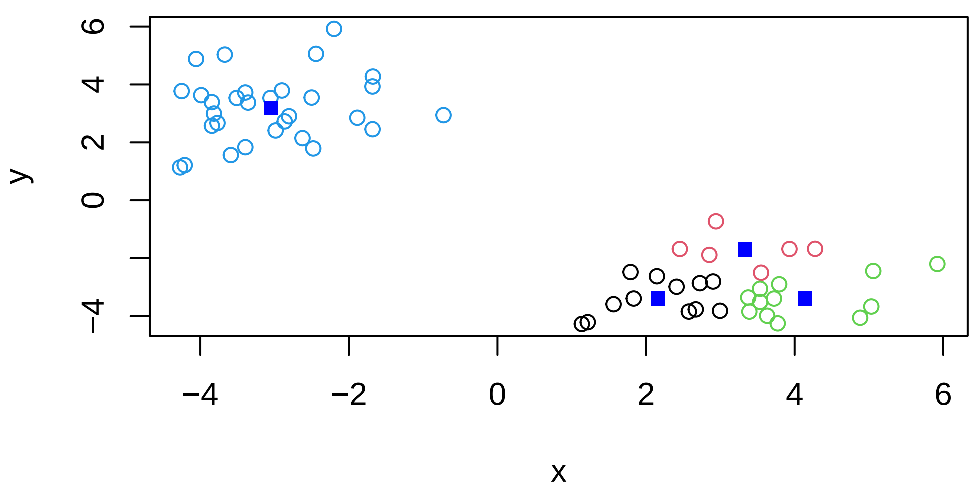
> Q. Re-run your k-means clustering and as for 4 clusters and plot the results as above.

```
km4 <- kmeans(z, centers = 4)
km4
```

```
K-means clustering with 4 clusters of sizes 12, 6, 12, 30

Cluster means:
          x         y
1  2.163351 -3.387570
2  3.333337 -1.692273
3  4.137054 -3.389132
4 -3.049135  3.186829

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

```
[39] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4

Within cluster sum of squares by cluster:
[1]  8.822089  4.074457 12.439027 62.677682
 (between_SS / total_SS =  93.2 %)

Available components:

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

```
plot(z, col=km4$cluster)
points(km4$centers, col="blue", pch=15)
```



##Hierarchical clustering

the main "base R" function for this is `hclust()`. Unlike `kmeans()` you cant just give your dataset as an input, you need to privide a distance matrix.

We can use the `dist()` function for this

```
d <- dist(z)
#hclust()
d
```

	1	2	3	4	5	6	7
2	2.0266960						
3	1.2154423	1.7623331					
4	4.1396330	2.7216976	3.0852764				
5	2.1031329	0.3647590	2.0314462	3.0184541			
6	0.8009187	1.2407808	1.1847385	3.5968347	1.3025225		
7	3.4699093	1.4524915	2.9717130	2.1280680	1.5061615	2.6924093	
8	1.7570826	1.0089759	0.9290284	2.3923412	1.3526458	1.2355671	2.0487687
9	2.0449387	0.6605769	1.3836224	2.2419266	1.0253195	1.3831573	1.5881228
10	1.1578520	0.9477614	0.9210589	3.0795705	1.1482025	0.5225358	2.3418325
11	1.8237548	2.4238141	2.8422014	5.1173474	2.2057436	1.7113284	3.6697594
12	0.3839605	1.6478145	1.0208985	3.7984161	1.7428033	0.4482507	3.0875021
13	2.1370766	0.2196013	1.7413150	2.5024476	0.5595616	1.3730942	1.3352403
14	1.5740482	1.0162566	1.9769611	3.7275002	0.8146599	0.8734413	2.3156703
15	1.8357324	0.4555453	1.3532580	2.5355283	0.8030520	1.1249759	1.6791021
16	1.7990364	0.6210274	1.9350668	3.3426887	0.3963653	1.0147600	1.9018050
17	1.8108564	0.2222776	1.5610670	2.7968804	0.4881671	1.0325929	1.6612527
18	2.6574459	0.7770615	2.5372224	2.9707493	0.5569518	1.8576741	1.1265242
19	1.1330102	1.9750638	2.1360305	4.5748915	1.8459645	1.0413817	3.3521244
20	2.0962493	2.8470374	1.1632512	3.3250647	3.1509329	2.3207717	3.8548448
21	0.9145453	1.7062594	1.8185189	4.2581332	1.6211520	0.7066308	3.1194961
22	1.0379788	2.1251219	0.4473273	3.5053953	2.3577151	1.3056621	3.3963975
23	1.9092313	2.5190401	2.9396909	5.2158420	2.2958310	1.8122164	3.7531143
24	1.5594062	0.9042881	1.8907385	3.6105462	0.7255270	0.8234622	2.2313904
25	2.3783457	0.4861041	2.2405904	2.9057745	0.2788071	1.5774292	1.2533730
26	3.2677027	1.8595495	2.2757774	0.8963058	2.1789534	2.7022955	1.6233687
27	1.0091064	1.0378395	0.9860056	3.2654503	1.1860561	0.3356482	2.4654593
28	2.2834366	1.8704552	1.0979927	2.0566561	2.2292936	2.0021740	2.5579064
29	3.4525718	1.5044546	2.8166821	1.7182235	1.6506372	2.7082638	0.4148856
30	2.6104560	1.9770001	1.4384669	1.7272357	2.3416495	2.2775757	2.4559951
31	7.5924472	9.3691775	7.6463819	9.9824021	9.5712266	8.3326226	10.6031804
32	9.2863838	11.1753783	9.5073569	12.0116813	11.3420879	10.0629953	12.4790519
33	7.2904007	9.0881696	7.3750647	9.7667587	9.2838467	8.0371047	10.3382285
34	6.9814744	8.9342932	7.3431383	10.0706266	9.0700389	7.7736468	10.2940944
35	8.6454380	10.4484021	8.7327899	11.0741724	10.6441299	9.3954965	11.6933029
36	8.4091308	10.3722254	8.7864954	11.4961947	10.5027090	9.2040243	11.7364278
37	7.5767902	9.5671573	8.0356762	10.8436176	9.6785798	8.3763065	10.9567849
38	7.0586086	9.0852930	7.7381047	10.7248929	9.1385893	7.8517332	10.5235058
39	6.0328522	7.9241531	6.2758401	8.9135820	8.0870061	6.8075852	9.2444879
40	6.7433979	8.7501042	7.2686218	10.1497800	8.8464012	7.5442869	10.1568912
41	5.9757166	7.7315326	6.0081462	8.4028406	7.9364214	6.7055993	8.9669453
42	6.7289694	8.7461442	7.3059217	10.2272008	8.8289860	7.5293001	10.1648782
43	8.6936619	10.6611928	9.0807257	11.7951559	10.7890285	9.4895119	12.0285383

44	7.7957268	9.7425475	8.1380638	10.8240157	9.8819763	8.5868140	11.0951621
45	7.8362500	9.8167783	8.2635231	11.0361287	9.9356855	8.6343166	11.1970954
46	7.7229090	9.6468162	8.0167332	10.6510180	9.7975339	8.5082190	10.9821247
47	7.5699529	9.5655085	8.0461024	10.8700534	9.6725891	8.3700920	10.9599981
48	8.0837222	10.0168335	8.3945495	11.0384196	10.1634011	8.8715746	11.3579188
49	6.3947186	8.3633900	6.8007201	9.5871180	8.4896173	7.1901823	9.7374194
50	7.0468190	9.0732554	7.7125942	10.6909362	9.1305613	7.8413861	10.5092931
51	7.0772401	9.0184482	7.4117470	10.1086836	9.1603649	7.8666529	10.3690632
52	7.8323753	9.7344726	8.0832921	10.6646978	9.8948262	8.6118164	11.0532631
53	7.3994676	9.2880105	7.6280020	10.1967808	9.4534547	8.1747159	10.5989935
54	9.4009631	11.3168210	9.6710779	12.2352166	11.4722028	10.1849305	12.6406161
55	6.8364364	8.8090244	7.2489691	10.0298025	8.9329856	7.6328354	10.1849305
56	8.1373906	10.1042432	8.5257644	11.2527679	10.2323268	8.9329856	11.4722028
57	9.3001527	11.0384996	9.2989996	11.4845551	11.2527679	10.0298025	12.2352166
58	6.4762905	8.3591188	6.7005175	9.2989996	8.5257644	7.2489691	9.6710779
59	8.0178298	9.9647601	8.3591188	11.0384996	10.1042432	8.8090244	11.3168210
60	6.0390915	8.0178298	6.4762905	9.3001527	8.1373906	6.8364364	9.4009631

	8	9	10	11	12	13	14
2							
3							
4							
5							
6							
7							
8							
9	0.4685021						
10	0.7142119	0.8951637					
11	2.8914304	2.8898980	2.1945371				
12	1.4067509	1.6685237	0.7768206	1.8388771			
13	0.9097358	0.4977918	1.0096538	2.6364326	1.7538348		
14	1.6573415	1.5384450	1.0878928	1.4090657	1.2928567	1.2314016	
15	0.5535317	0.2999007	0.6795822	2.5907492	1.4528986	0.3881337	1.2413264
16	1.4259559	1.2128660	1.0143538	1.8231051	1.4629556	0.8404838	0.4202785
17	0.8728441	0.6175556	0.7263332	2.3220875	1.4303354	0.3493324	0.9357083
18	1.7634998	1.3493805	1.6865963	2.5572013	2.2994137	0.8577760	1.2634923
19	2.2639367	2.3331757	1.5516508	0.7112595	1.1278772	2.1686990	1.0450072
20	1.8762583	2.3358756	2.0799375	3.8873182	2.0572858	2.7802543	3.1387164
21	1.9322271	2.0189956	1.2188853	1.0246289	0.8298920	1.8886561	0.8684184
22	1.3671316	1.8106456	1.2106080	2.8161075	0.9922057	2.1314587	2.1661705
23	2.9931658	2.9898863	2.2962587	0.1017758	1.9338016	2.7323841	1.5034040
24	1.5416563	1.4183878	0.9899406	1.5195362	1.2574802	1.1175777	0.1200914
25	1.4880301	1.0977466	1.3878491	2.4253472	2.0146915	0.5999804	1.0641296
26	1.5116905	1.3470366	2.1869750	4.2300441	2.9175786	1.6403212	2.8506159

27	0.9001345	1.0685851	0.1870846	2.0219204	0.6251506	1.1305049	0.9910540
28	0.9043951	1.2272436	1.5236549	3.7099621	2.0176652	1.7249069	2.5512121
29	1.8886606	1.4587222	2.2985269	3.8546854	3.0688755	1.3382022	2.4621371
30	1.0994486	1.3165327	1.7777791	3.9722987	2.3298249	1.8043323	2.7537597
31	8.5566552	9.0225157	8.4321412	8.9640162	7.8877277	9.3790581	9.1650133
32	10.4328826	10.8909307	10.2281486	10.4531841	9.6148124	11.2090098	10.8527838
33	8.2900251	8.7544930	8.1481704	8.6406368	7.5911091	9.1028539	8.8640291
34	8.2713269	8.7145149	7.9888193	8.0445100	7.3284803	8.9894067	8.5278401
35	9.6458057	10.1109637	9.5085863	9.9639910	8.9491069	10.4622285	10.2194165
36	9.7146689	10.1575487	9.4276915	9.4090063	8.7601735	10.4297399	9.9465052
37	8.9585965	9.3905067	8.6290792	8.4940825	7.9372837	9.6368175	9.0917682
38	8.6241835	9.0177824	8.1850755	7.6599608	7.4384512	9.1882729	8.4664715
39	7.2042259	7.6574136	6.9764997	7.2661987	6.3593752	7.9625083	7.6017078
40	8.1833312	8.6046930	7.8196936	7.5997284	7.1103603	8.8292724	8.2375551
41	6.9196456	7.3850907	6.7957141	7.4089891	6.2623859	7.7406166	7.5450315
42	8.2127809	8.6256765	7.8237997	7.5097728	7.1009645	8.8330551	8.1994222
43	10.0086505	10.4506477	9.7172806	9.6738247	9.0462895	10.7202047	10.2273961
44	9.0669099	9.5130510	8.7962693	8.8561589	8.1411521	9.7947888	9.3438181
45	9.1890238	9.6253610	8.8757771	8.7876112	8.1931743	9.8817536	9.3607930
46	8.9456173	9.3964514	8.6992514	8.8431120	8.0609921	9.6918569	9.2810086
47	8.9674314	9.3969340	8.6292984	8.4638091	7.9324483	9.6377679	9.0786105
48	9.3235527	9.7731256	9.0695837	9.1740849	8.4248909	10.0643396	9.6377679
49	7.7265041	8.1638218	7.4204644	7.4258419	6.7468759	8.4248909	7.9324483
50	8.6019943	8.9984424	8.1696305	7.6719447	7.4258419	9.1740849	8.4638091
51	8.3405839	8.7867665	8.0718607	8.1696305	7.4204644	9.0695837	8.6292984
52	9.0110629	9.4657167	8.7867665	8.9984424	8.1638218	9.7731256	9.3969340
53	8.5550586	9.0110629	8.3405839	8.6019943	7.7265041	9.3235527	8.9674314
54	10.5989935	11.0532631	10.3690632	10.5092931	9.7374194	11.3579188	10.9599981
55	8.1747159	8.6118164	7.8666529	7.8413861	7.1901823	8.8715746	8.3700920
56	9.4534547	9.8948262	9.1603649	9.1305613	8.4896173	10.1634011	9.6725891
57	10.1967808	10.6646978	10.1086836	10.6909362	9.5871180	11.0384196	10.8700534
58	7.6280020	8.0832921	7.4117470	7.7125942	6.8007201	8.3945495	8.0461024
59	9.2880105	9.7344726	9.0184482	9.0732554	8.3633900	10.0168335	9.5655085
60	7.3994676	7.8323753	7.0772401	7.0468190	6.3947186	8.0837222	7.5699529

	15	16	17	18	19	20	21
2							
3							
4							
5							
6							
7							
8							
9							

10							
11							
12							
13							
14							
15							
16	0.9326332						
17	0.3415599	0.5956876					
18	1.2192637	0.8902006	0.9779102				
19	2.0429315	1.4506932	1.8274030	2.3065933			
20	2.4030513	3.0901170	2.6663512	3.6211625	3.1771522		
21	1.7338399	1.2361911	1.5417428	2.1196924	0.3347682	2.8871511	
22	1.7473971	2.2023549	1.9121758	2.8877933	2.1062515	1.0713006	1.8209523
23	2.6905973	1.9155523	2.4199346	2.6373162	0.8059751	3.9792011	1.1232416
24	1.1212378	0.3296353	0.8168708	1.2039308	1.1210851	3.0538357	0.9157857
25	0.9373258	0.6577561	0.6799651	0.2991737	2.1058881	3.3330932	1.8923667
26	1.6425379	2.4783868	1.9156838	2.2187866	3.6792211	2.7263884	3.3618757
27	0.8324204	0.9899521	0.8166805	1.7382885	1.3694103	2.1484813	1.0356634
28	1.4288783	2.3299310	1.7647244	2.5734990	3.0406843	1.3763291	2.7064221
29	1.6189958	2.0418635	1.6861425	1.3872492	3.4704141	3.6109074	3.2101576
30	1.5645785	2.4927968	1.9061288	2.6226105	3.3189476	1.6384708	2.9842061
31	8.9932598	9.3428374	9.1523523	10.1180226	8.5049217	6.8065872	8.4331571
32	10.8300605	11.0771635	10.9544360	11.8972245	10.0843249	8.7663841	10.0629298
33	8.7181257	9.0490768	8.8703318	9.8326236	8.1897400	6.5615281	8.1236837
34	8.6213706	8.7805020	8.7120290	9.6269804	7.7100073	6.7522075	7.7124065
35	10.0772013	10.4080054	10.2306554	11.1930305	9.5299816	7.9020384	9.4717123
36	10.0628073	10.2077338	10.1499484	11.0595058	9.1057342	8.1834978	9.1216976
37	9.2781459	9.3698896	9.3452797	10.2339351	8.2204374	7.5186561	8.2537708
38	8.8595834	8.7934094	8.8687477	9.6804961	7.5070691	7.4488074	7.6013606
39	7.5860518	7.8218969	7.7028154	8.6424222	6.8586486	5.6100666	6.8221738
40	8.4782185	8.5278233	8.5290709	9.3996344	7.3456580	6.8309968	7.3913473
41	7.3547897	7.7133532	7.5150435	8.4820100	6.9188280	5.1853707	6.8309968
42	8.4886642	8.5015646	8.5261870	9.3794897	7.2850888	6.9188280	7.3456580
43	10.3541313	10.4917854	10.4389256	11.3457030	9.3794897	8.4820100	9.3996344
44	9.4249105	9.5946571	9.5203320	10.4389256	8.5261870	7.5150435	8.5290709
45	9.5197336	9.6322903	9.5946571	10.4917854	8.5015646	7.7133532	8.5278233
46	9.3181913	9.5197336	9.4249105	10.3541313	8.4886642	7.3547897	8.4782185
47	9.2810086	9.3607930	9.3438181	10.2273961	8.1994222	7.5450315	8.2375551
48	9.6918569	9.8817536	9.7947888	10.7202047	8.8330551	7.7406166	8.8292724
49	8.0609921	8.1931743	8.1411521	9.0462895	7.1009645	6.2623859	7.1103603
50	8.8431120	8.7876112	8.8561589	9.6738247	7.5097728	7.4089891	7.5997284
51	8.6992514	8.8757771	8.7962693	9.7172806	7.8237997	6.7957141	7.8196936
52	9.3964514	9.6253610	9.5130510	10.4506477	8.6256765	7.3850907	8.6046930

53	8.9456173	9.1890238	9.0669099	10.0086505	8.2127809	6.9196456	8.1833312
54	10.9821247	11.1970954	11.0951621	12.0285383	10.1648782	8.9669453	10.1568912
55	8.5082190	8.6343166	8.5868140	9.4895119	7.5293001	6.7055993	7.5442869
56	9.7975339	9.9356855	9.8819763	10.7890285	8.8289860	7.9364214	8.8464012
57	10.6510180	11.0361287	10.8240157	11.7951559	10.2272008	8.4028406	10.1497800
58	8.0167332	8.2635231	8.1380638	9.0807257	7.3059217	6.0081462	7.2686218
59	9.6468162	9.8167783	9.7425475	10.6611928	8.7461442	7.7315326	8.7501042
60	7.7229090	7.8362500	7.7957268	8.6936619	6.7289694	5.9757166	6.7433979

	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	2.9079005						
24	2.0980583	1.6149882					
25	2.5886853	2.5116922	0.9858709				
26	2.7132994	4.3292048	2.7322987	2.1035085			
27	1.2131803	2.1234747	0.9058698	1.4433571	2.3721663		
28	1.4763582	3.8113476	2.4378600	2.3246467	1.3604251	1.6890385	
29	3.2555761	3.9433204	2.3648120	1.4370511	1.2272290	2.4445423	2.2723782
30	1.8206473	4.0740277	2.6367935	2.3987454	1.0958411	1.9531825	0.3443256
31	7.2483744	8.9977324	9.1419495	9.8197159	9.5127377	8.3861916	8.1810117
32	9.0866129	10.4713372	10.8451673	11.6031254	11.4914594	10.1597704	10.1411651
33	6.9713981	8.6730814	8.8431845	9.5348215	9.2766776	8.0980289	7.9375944
34	6.9055147	8.0582381	8.5298311	9.3387292	9.4582609	7.9022961	8.0980289
35	8.3308341	9.9930711	10.1999233	10.8952428	10.6085024	9.4582609	9.2766776

36	8.3487724	9.4171730	9.9518397	10.7726390	10.8952428	9.3387292	9.5348215
37	7.5921166	8.4981366	9.1033564	9.9518397	10.1999233	8.5298311	8.8431845
38	7.2921267	7.6466454	8.4981366	9.4171730	9.9930711	8.0582381	8.6730814
39	5.8482150	7.2921267	7.5921166	8.3487724	8.3308341	6.9055147	6.9713981
40	6.8221738	7.6013606	8.2537708	9.1216976	9.4717123	7.7124065	8.1236837
41	5.6100666	7.4488074	7.5186561	8.1834978	7.9020384	6.7522075	6.5615281
42	6.8586486	7.5070691	8.2204374	9.1057342	9.5299816	7.7100073	8.1897400
43	8.6424222	9.6804961	10.2339351	11.0595058	11.1930305	9.6269804	9.8326236
44	7.7028154	8.8687477	9.3452797	10.1499484	10.2306554	8.7120290	8.8703318
45	7.8218969	8.7934094	9.3698896	10.2077338	10.4080054	8.7805020	9.0490768
46	7.5860518	8.8595834	9.2781459	10.0628073	10.0772013	8.6213706	8.7181257
47	7.6017078	8.4664715	9.0917682	9.9465052	10.2194165	8.5278401	8.8640291
48	7.9625083	9.1882729	9.6368175	10.4297399	10.4622285	8.9894067	9.1028539
49	6.3593752	7.4384512	7.9372837	8.7601735	8.9491069	7.3284803	7.5911091
50	7.2661987	7.6599608	8.4940825	9.4090063	9.9639910	8.0445100	8.6406368
51	6.9764997	8.1850755	8.6290792	9.4276915	9.5085863	7.9888193	8.1481704
52	7.6574136	9.0177824	9.3905067	10.1575487	10.1109637	8.7145149	8.7544930
53	7.2042259	8.6241835	8.9585965	9.7146689	9.6458057	8.2713269	8.2900251
54	9.2444879	10.5235058	10.9567849	11.7364278	11.6933029	10.2940944	10.3382285
55	6.8075852	7.8517332	8.3763065	9.2040243	9.3954965	7.7736468	8.0371047
56	8.0870061	9.1385893	9.6785798	10.5027090	10.6441299	9.0700389	9.2838467
57	8.9135820	10.7248929	10.8436176	11.4961947	11.0741724	10.0706266	9.7667587
58	6.2758401	7.7381047	8.0356762	8.7864954	8.7327899	7.3431383	7.3750647
59	7.9241531	9.0852930	9.5671573	10.3722254	10.4484021	8.9342932	9.0881696
60	6.0328522	7.0586086	7.5767902	8.4091308	8.6454380	6.9814744	7.2904007

	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	2.1319577						
31	10.3982583	8.4169571					
32	12.3059356	10.3982583	2.1319577				
33	10.1411651	8.1810117	0.3443256	2.2723782			
34	10.1597704	8.3861916	1.9531825	2.4445423	1.6890385		
35	11.4914594	9.5127377	1.0958411	1.2272290	1.3604251	2.3721663	
36	11.6031254	9.8197159	2.3987454	1.4370511	2.3246467	1.4433571	2.1035085
37	10.8451673	9.1419495	2.6367935	2.3648120	2.4378600	0.9058698	2.7322987
38	10.4713372	8.9977324	4.0740277	3.9433204	3.8113476	2.1234747	4.3292048
39	9.0866129	7.2483744	1.8206473	3.2555761	1.4763582	1.2131803	2.7132994
40	10.0629298	8.4331571	2.9842061	3.2101576	2.7064221	1.0356634	3.3618757
41	8.7663841	6.8065872	1.6384708	3.6109074	1.3763291	2.1484813	2.7263884
42	10.0843249	8.5049217	3.3189476	3.4704141	3.0406843	1.3694103	3.6792211
43	11.8972245	10.1180226	2.6226105	1.3872492	2.5734990	1.7382885	2.2187866
44	10.9544360	9.1523523	1.9061288	1.6861425	1.7647244	0.8166805	1.9156838
45	11.0771635	9.3428374	2.4927968	2.0418635	2.3299310	0.9899521	2.4783868
46	10.8300605	8.9932598	1.5645785	1.6189958	1.4288783	0.8324204	1.6425379
47	10.8527838	9.1650133	2.7537597	2.4621371	2.5512121	0.9910540	2.8506159
48	11.2090098	9.3790581	1.8043323	1.3382022	1.7249069	1.1305049	1.6403212
49	9.6148124	7.8877277	2.3298249	3.0688755	2.0176652	0.6251506	2.9175786
50	10.4531841	8.9640162	3.9722987	3.8546854	3.7099621	2.0219204	4.2300441
51	10.2281486	8.4321412	1.7777791	2.2985269	1.5236549	0.1870846	2.1869750
52	10.8909307	9.0225157	1.3165327	1.4587222	1.2272436	1.0685851	1.3470366
53	10.4328826	8.5566552	1.0994486	1.8886606	0.9043951	0.9001345	1.5116905
54	12.4790519	10.6031804	2.4559951	0.4148856	2.5579064	2.4654593	1.6233687
55	10.0629953	8.3326226	2.2775757	2.7082638	2.0021740	0.3356482	2.7022955
56	11.3420879	9.5712266	2.3416495	1.6506372	2.2292936	1.1860561	2.1789534
57	12.0116813	9.9824021	1.7272357	1.7182235	2.0566561	3.2654503	0.8963058
58	9.5073569	7.6463819	1.4384669	2.8166821	1.0979927	0.9860056	2.2757774
59	11.1753783	9.3691775	1.9770001	1.5044546	1.8704552	1.0378395	1.8595495
60	9.2863838	7.5924472	2.6104560	3.4525718	2.2834366	1.0091064	3.2677027
	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	0.9858709						
38	2.5116922	1.6149882					
39	2.5886853	2.0980583	2.9079005				
40	1.8923667	0.9157857	1.1232416	1.8209523			
41	3.3330932	3.0538357	3.9792011	1.0713006	2.8871511		
42	2.1058881	1.1210851	0.8059751	2.1062515	0.3347682	3.1771522	
43	0.2991737	1.2039308	2.6373162	2.8877933	2.1196924	3.6211625	2.3065933
44	0.6799651	0.8168708	2.4199346	1.9121758	1.5417428	2.6663512	1.8274030

45	0.6577561	0.3296353	1.9155523	2.2023549	1.2361911	3.0901170	1.4506932
46	0.9373258	1.1212378	2.6905973	1.7473971	1.7338399	2.4030513	2.0429315
47	1.0641296	0.1200914	1.5034040	2.1661705	0.8684184	3.1387164	1.0450072
48	0.5999804	1.1175777	2.7323841	2.1314587	1.8886561	2.7802543	2.1686990
49	2.0146915	1.2574802	1.9338016	0.9922057	0.8298920	2.0572858	1.1278772
50	2.4253472	1.5195362	0.1017758	2.8161075	1.0246289	3.8873182	0.7112595
51	1.3878491	0.9899406	2.2962587	1.2106080	1.2188853	2.0799375	1.5516508
52	1.0977466	1.4183878	2.9898863	1.8106456	2.0189956	2.3358756	2.3331757
53	1.4880301	1.5416563	2.9931658	1.3671316	1.9322271	1.8762583	2.2639367
54	1.2533730	2.2313904	3.7531143	3.3963975	3.1194961	3.8548448	3.3521244
55	1.5774292	0.8234622	1.8122164	1.3056621	0.7066308	2.3207717	1.0413817
56	0.2788071	0.7255270	2.2958310	2.3577151	1.6211520	3.1509329	1.8459645
57	2.9057745	3.6105462	5.2158420	3.5053953	4.2581332	3.3250647	4.5748915
58	2.2405904	1.8907385	2.9396909	0.4473273	1.8185189	1.1632512	2.1360305
59	0.4861041	0.9042881	2.5190401	2.1251219	1.7062594	2.8470374	1.9750638
60	2.3783457	1.5594062	1.9092313	1.0379788	0.9145453	2.0962493	1.1330102

	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	0.9779102						
45	0.8902006	0.5956876					
46	1.2192637	0.3415599	0.9326332				
47	1.2634923	0.9357083	0.4202785	1.2413264			
48	0.8577760	0.3493324	0.8404838	0.3881337	1.2314016		
49	2.2994137	1.4303354	1.4629556	1.4528986	1.2928567	1.7538348	
50	2.5572013	2.3220875	1.8231051	2.5907492	1.4090657	2.6364326	1.8388771
51	1.6865963	0.7263332	1.0143538	0.6795822	1.0878928	1.0096538	0.7768206
52	1.3493805	0.6175556	1.2128660	0.2999007	1.5384450	0.4977918	1.6685237
53	1.7634998	0.8728441	1.4259559	0.5535317	1.6573415	0.9097358	1.4067509
54	1.1265242	1.6612527	1.9018050	1.6791021	2.3156703	1.3352403	3.0875021
55	1.8576741	1.0325929	1.0147600	1.1249759	0.8734413	1.3730942	0.4482507
56	0.5569518	0.4881671	0.3963653	0.8030520	0.8146599	0.5595616	1.7428033
57	2.9707493	2.7968804	3.3426887	2.5355283	3.7275002	2.5024476	3.7984161
58	2.5372224	1.5610670	1.9350668	1.3532580	1.9769611	1.7413150	1.0208985
59	0.7770615	0.2222776	0.6210274	0.4555453	1.0162566	0.2196013	1.6478145
60	2.6574459	1.8108564	1.7990364	1.8357324	1.5740482	2.1370766	0.3839605

	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  2.1945371
52  2.8898980  0.8951637
53  2.8914304  0.7142119  0.4685021
```

```
54  3.6697594  2.3418325  1.5881228  2.0487687
55  1.7113284  0.5225358  1.3831573  1.2355671  2.6924093
56  2.2057436  1.1482025  1.0253195  1.3526458  1.5061615  1.3025225
57  5.1173474  3.0795705  2.2419266  2.3923412  2.1280680  3.5968347  3.0184541
58  2.8422014  0.9210589  1.3836224  0.9290284  2.9717130  1.1847385  2.0314462
59  2.4238141  0.9477614  0.6605769  1.0089759  1.4524915  1.2407808  0.3647590
60  1.8237548  1.1578520  2.0449387  1.7570826  3.4699093  0.8009187  2.1031329
           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
```

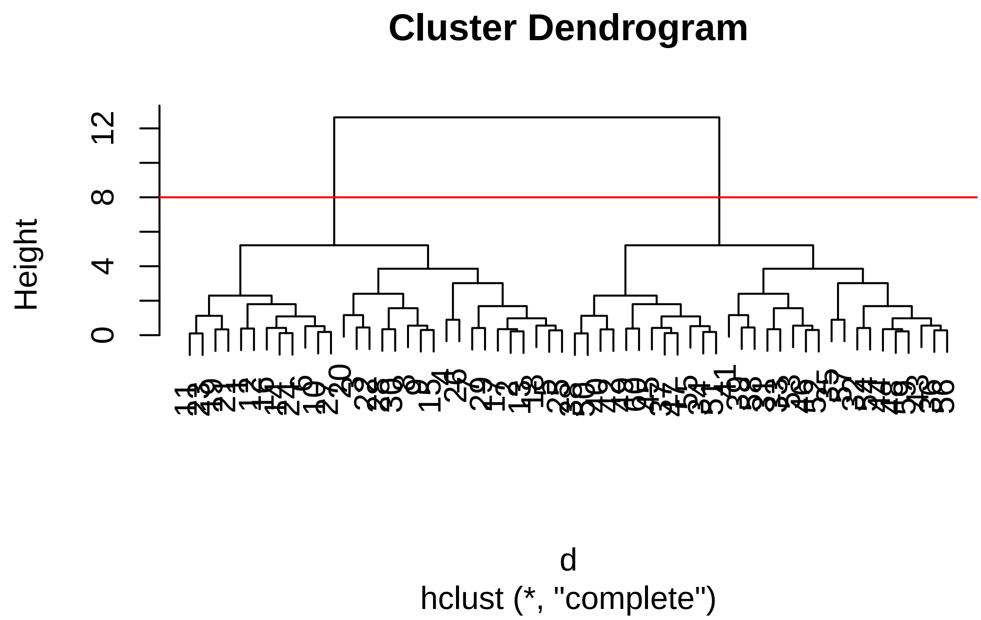
```
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58  3.0852764
59  2.7216976  1.7623331
60  4.1396330  1.2154423  2.0266960
```

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

```
Call:
hclust(d = d)

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

There is a custom plot for hclust objects, lets see it.

```
plot(hc)
abline(h=8, col="red")
```



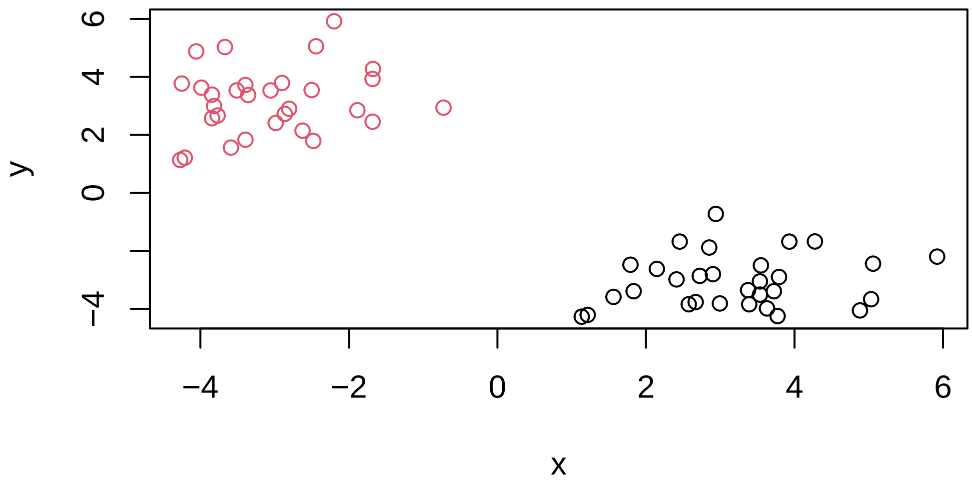
The function to extract clusters/grps from hclust object/tree is called `cutree()`

```
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 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
```

Q plot data with hclust clusters:

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




```r
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 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
```

Principal Component Analysis (PCA)

Method to reduce the number of things you need to look at to something that looks more natural.

In other words, reduce the features dimensionality while only losing a small amount of information.

The first principal component (PC) follows a "best fit" through the data points.

The main function for PCA in base R for PAC is called `prcomp()`. There are many, many add on packages with PCA functions tailored to particular data types (RNAseq, protein, structure, metagenomics, etc….)

PCA of UK food data

Read the data into R, it is a CSV file and we can use `read.csv()` to read it

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

```
                     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
```

I would like the food names as row names no their own column of data (first column currently). I can fix this like so:

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

```
                                     X England Wales Scotland N.Ireland
Cheese                          Cheese     105   103      103        66
Carcass_meat              Carcass_meat     245   227      242       267
Other_meat                  Other_meat     685   803      750       586
Fish                              Fish     147   160      122        93
Fats_and_oils            Fats_and_oils     193   235      184       209
Sugars                          Sugars     156   175      147       139
Fresh_potatoes          Fresh_potatoes     720   874      566      1033
```

```
Fresh_Veg                     Fresh_Veg      253   265       171        143
Other_Veg                     Other_Veg      488   570       418        355
Processed_potatoes   Processed_potatoes      198   203       220        187
Processed_Veg             Processed_Veg      360   365       337        334
Fresh_fruit                 Fresh_fruit     1102  1137       957        674
Cereals                         Cereals     1472  1582      1462       1494
Beverages                      Beverages      57    73        53         47
Soft_drinks                 Soft_drinks     1374  1256      1572       1506
Alcoholic_drinks       Alcoholic_drinks      375   475       458        135
Confectionery             Confectionery       54    64        62         41
```

A better way to do this is to do it at the time of data import with `read.csv()`.

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

```
                   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
```

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

```
dim(food)
```

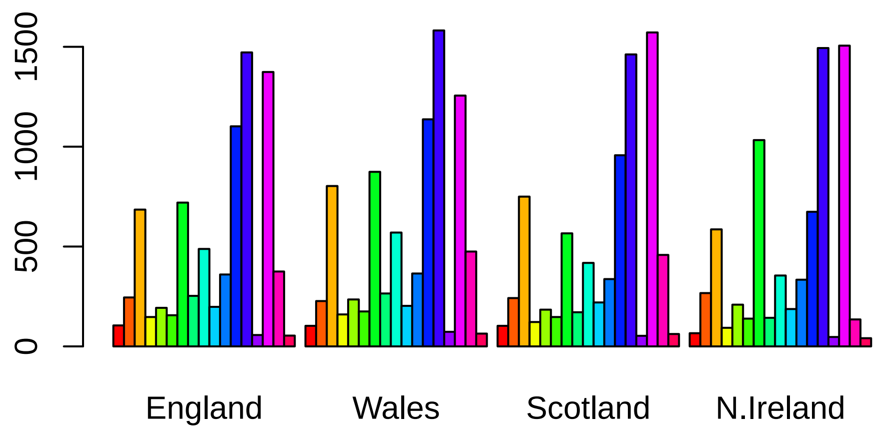
```
[1] 17  4
```

Lets make some plots and dig into the data alittle.

```
rainbow(nrow(food))
```

```
 [1] "#FF0000" "#FF5A00" "#FFB400" "#F0FF00" "#96FF00" "#3CFF00" "#00FF1E"
 [8] "#00FF78" "#00FFD2" "#00D2FF" "#0078FF" "#001EFF" "#3C00FF" "#9600FF"
[15] "#F000FF" "#FF00B4" "#FF005A"
```

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

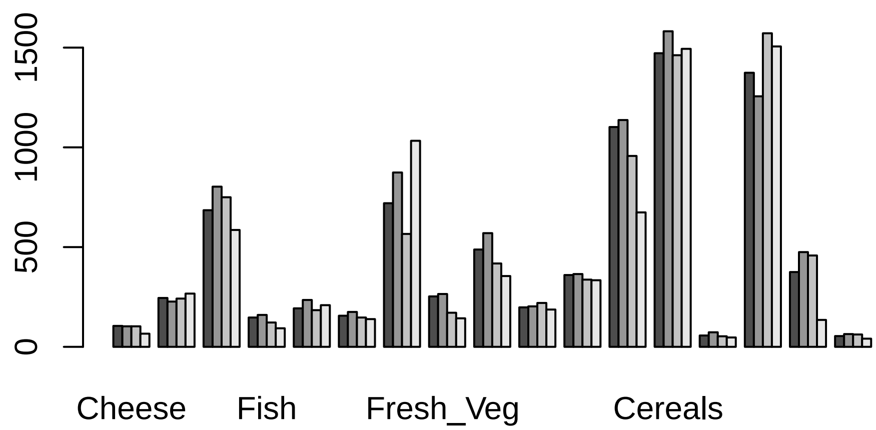



```
t(food)
```

```
          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
```

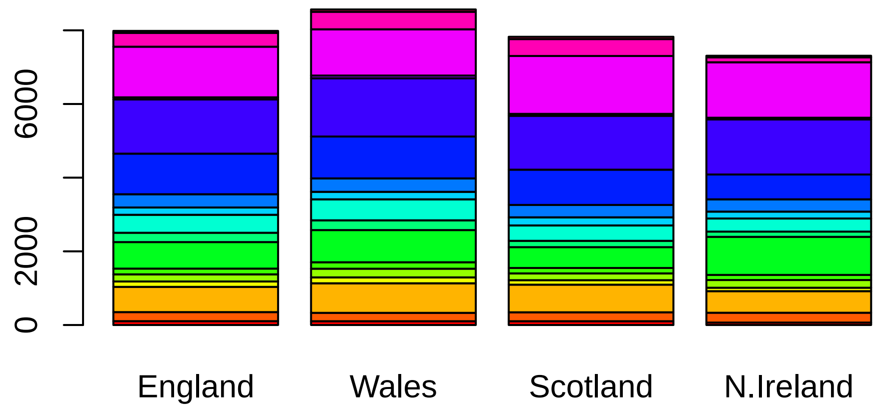
```
N.Ireland            1033        143        355                   187
          Processed_Veg  Fresh_fruit  Cereals  Beverages Soft_drinks
England             360         1102     1472         57        1374
Wales               365         1137     1582         73        1256
Scotland            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
```

```
barplot(as.matrix(t(food)), beside=T)
```



This is not helpful:

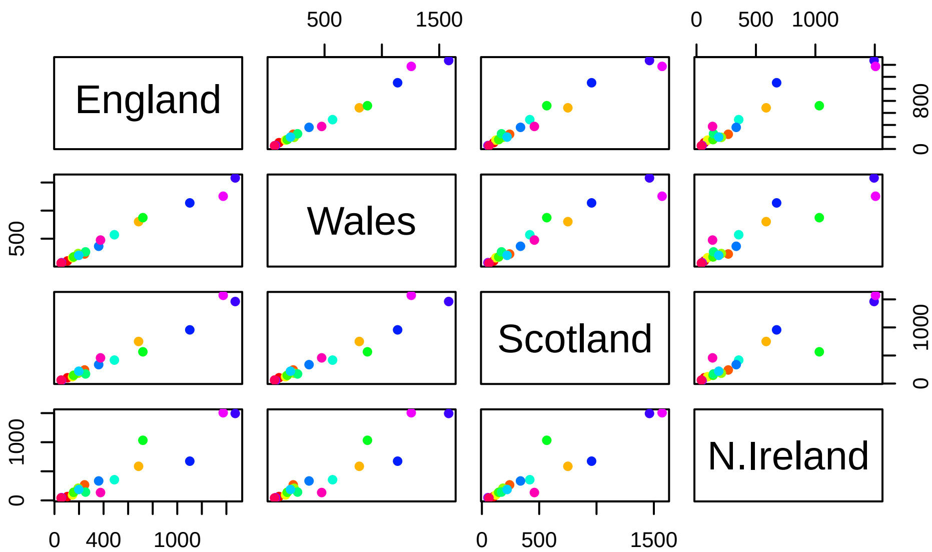
```
barplot(as.matrix(food), beside=F, col=rainbow(nrow(food)))
```



How about something called a "pairs" plot where we plot each country against all other countries.

This pairwise analysis only works really if you have small data sets(ex: this data set only has 4 countries in it, if there was 1000 it would be very difficult to read)

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



Really there is a better way...

PCA to the rescue!

We can run a Principal Component Analysis (PCA) for this data with the `prcomp()` function.

```
head(food)
```

```
               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
```

We need to take a transpose of this data to get the food in the column and the countries in the rows.

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

```
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
```

What is in my pca result object?

```r
attributes(pca)
```

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

$class
[1] "prcomp"
```

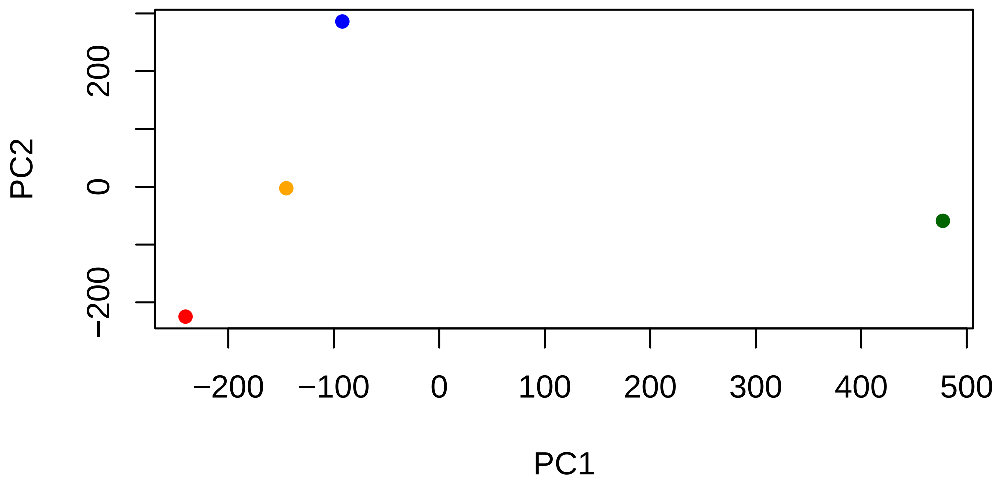
The scores along the new PCs

```r
pca$x
```

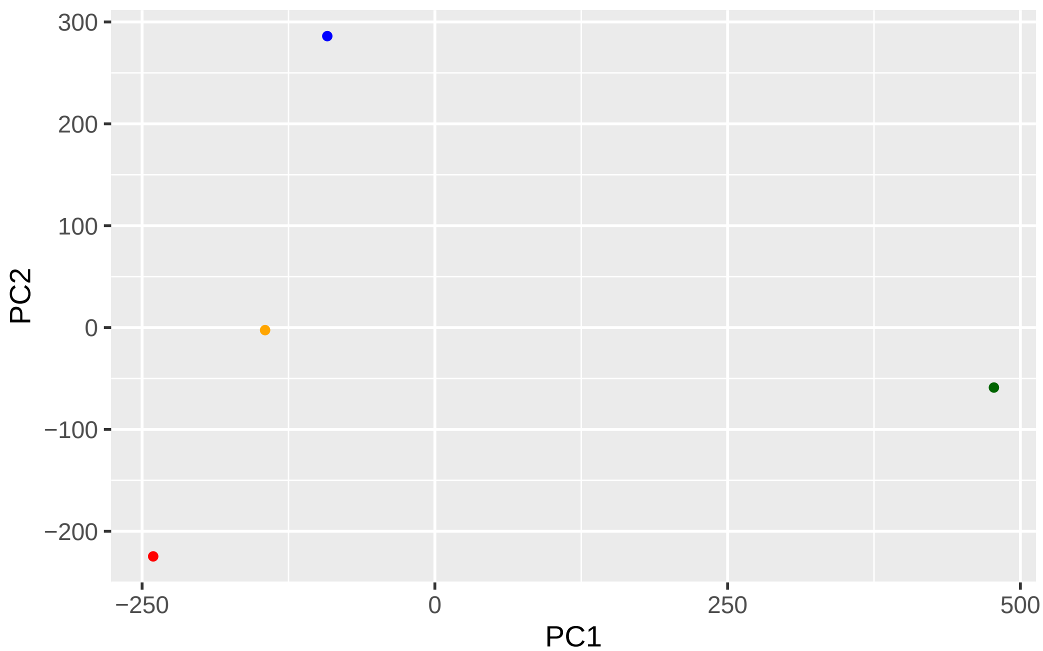
```
                 PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

To make my main result figure, often called a PC plot (or score plot, orientation plot, or PC1 vs. PC2 plot etc.)

```r
plot(pca$x[,1], pca$x[,2], xlab = "PC1", ylab= "PC2",
     col=c("orange","red", "blue", "darkgreen"), pch=16)
```

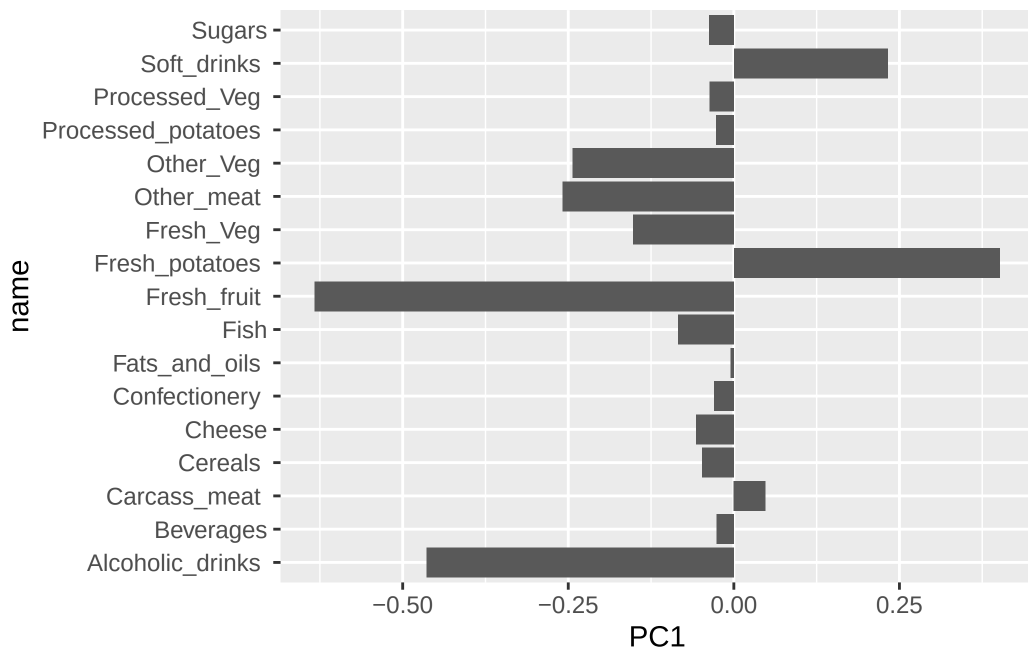



```r
library(ggplot2)

data <-as.data.frame(pca$x)

ggplot(data)+aes(PC1, PC2)+geom_point(col=c("orange","red", "blue", "darkgreen"), pch=16)
```



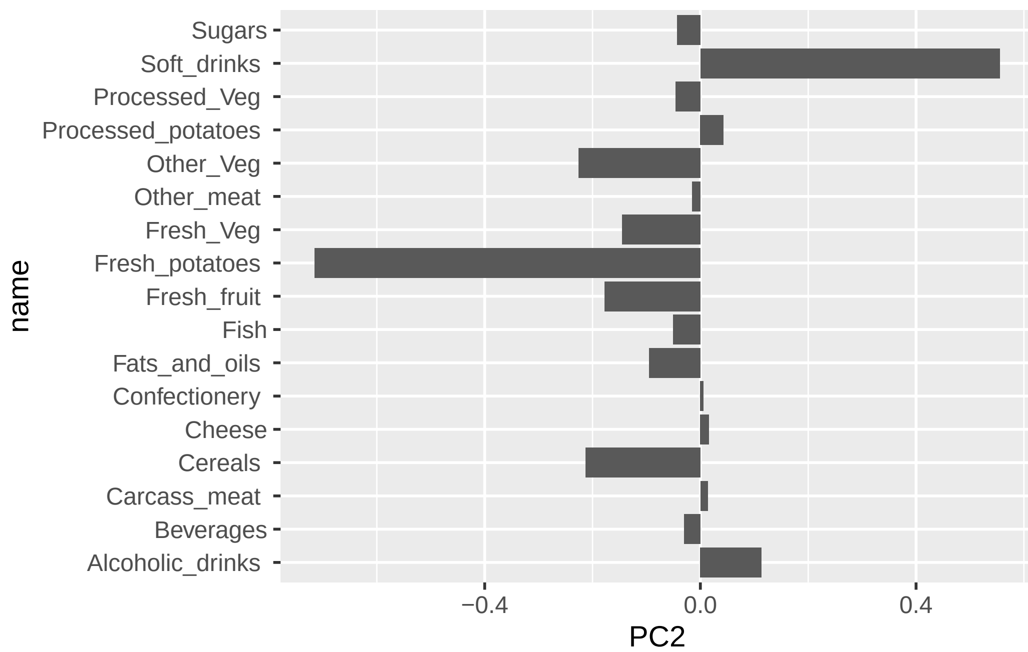
To see the contributions of the original variables (foods) to these new PCs we can look at the `pca$rotation` component of our result objects.

```
loadings <- as.data.frame(pca$rotation)
loadings$name <- rownames(loadings)

ggplot(loadings)+
  aes(PC1, name)+
geom_col()
```



And PC2

```
ggplot(loadings)+
  aes(PC2, name)+
geom_col()
```


Sugars
Soft_drinks
Processed_Veg
Processed_potatoes
Other_Veg
Other_meat
Fresh_Veg
Fresh_potatoes
Fresh_fruit
Fish
Fats_and_oils
Confectionery
Cheese
Cereals
Carcass_meat
Beverages
Alcoholic_drinks
name
−0.4
0.0
0.4
PC2