Class 7

Courtney Cameron PID:A69028599

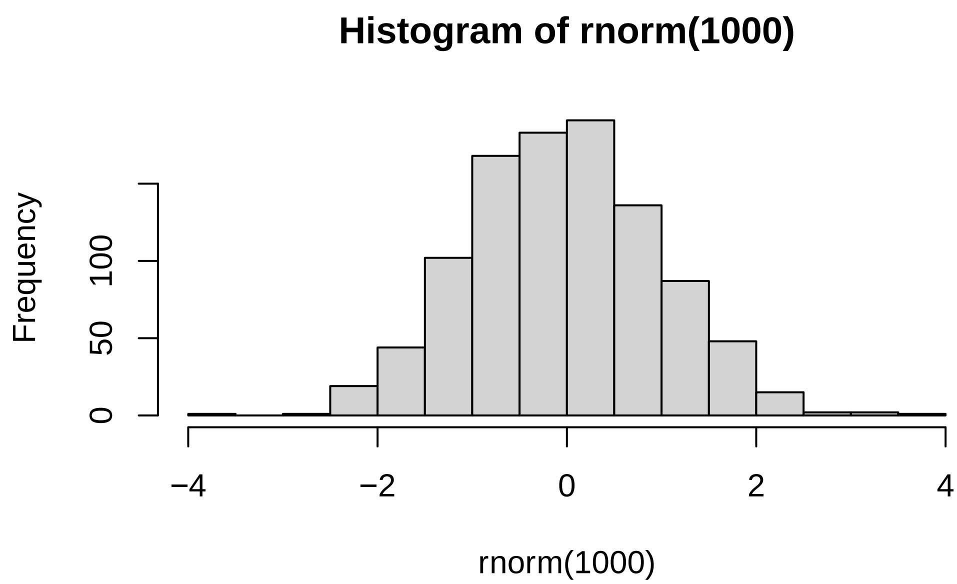
#Machine Learning methods using clustering and dimensionality reduction approaches

#Kmeans clustering

the main function for k-means in base R is kmeans()

using made up data to determine how kmeans works and to look at how the results are given

```
hist(rnorm(1000))
```



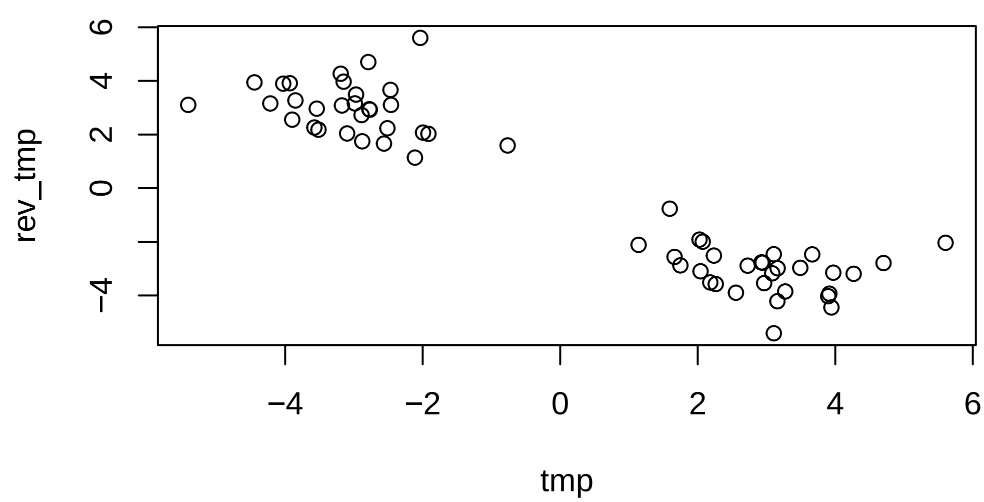
make a vector with 60 points, half centered at +3 and half centered at -3

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

```
 [1]  1.6632450  3.0839056  2.9261319  5.6040907  3.1583641  4.7017970
 [7]  3.8972452  3.6647712  2.2621675  3.2721998  3.1062810  2.1821310
[13]  2.0405044  2.7262137  2.0726826  4.2674208  1.1397543  1.7477060
[19]  3.1056991  3.1617587  2.2348899  2.5553554  2.9657435  3.9443030
[25]  3.9711806  3.4912037  2.9442367  3.9141066  2.0259699  1.5928234
[31] -0.7645645 -1.9166444 -3.9329610 -2.7761096 -2.9690468 -3.1497545
[37] -4.4465216 -3.5404328 -3.8971947 -2.5131877 -2.9861231 -2.4611006
[43] -2.8787827 -2.1124925 -3.1919195 -1.9951467 -2.8884250 -3.0986156
[49] -3.5154646 -5.4095989 -3.8504689 -3.5749708 -2.4686420 -4.0279586
[55] -2.7909383 -4.2159344 -2.0352577 -2.7681295 -3.1741551 -2.5627707
```

make a scatter plot of tmp

```r
rev_tmp <- rev(tmp)
x <- cbind(tmp, rev_tmp)


plot(x)
```



find kmeans of tmp_df

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

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

Cluster means:
        tmp   rev_tmp
1 -3.063777  2.980796
2  2.980796 -3.063777

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] 52.96636 52.96636
 (between_SS / total_SS =  91.2 %)

Available components:

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

What is in the result object

```r
attributes(k)
```

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

$class
[1] "kmeans"
```

Whar are the cluster centers

```r
k$centers
```

```
        tmp   rev_tmp
1 -3.063777  2.980796
2  2.980796 -3.063777
```

what is the clustering results

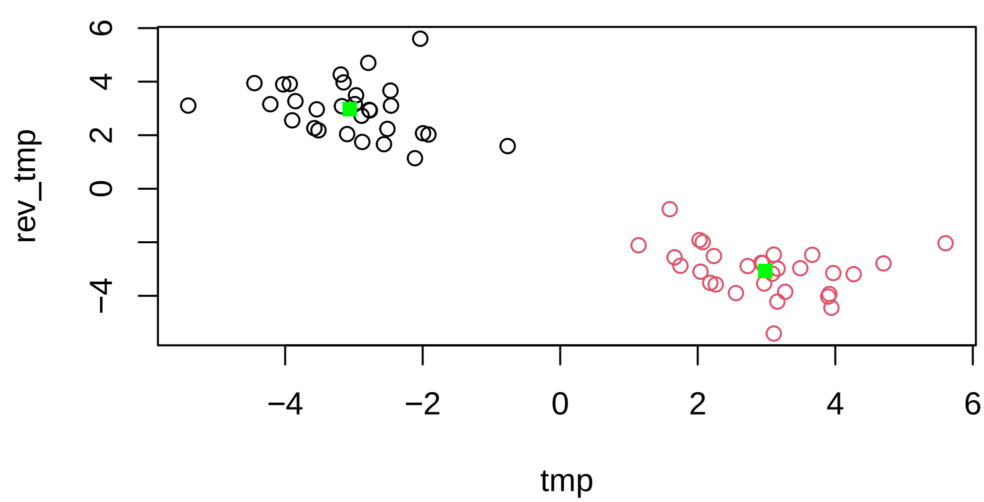
```
k$cluster
```

```
 [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 Plot your data 'x' showing your clustering results and the center point for each cluster?<

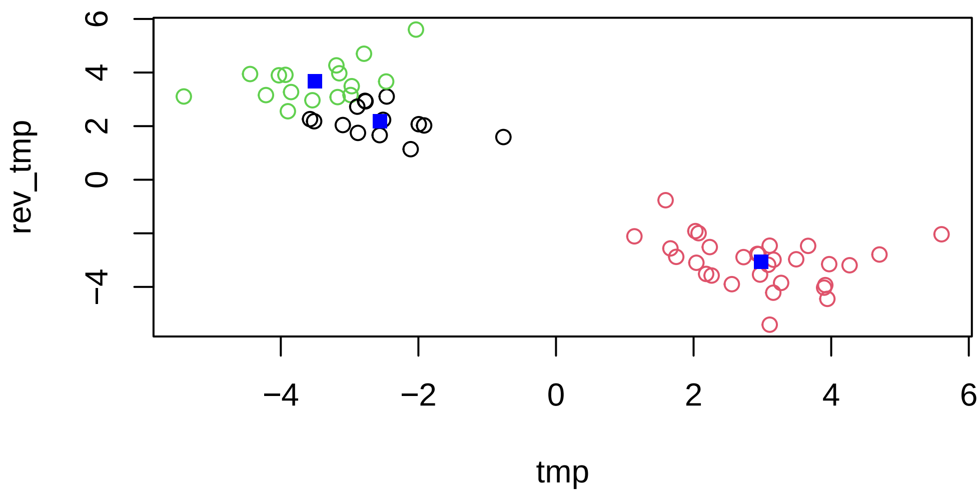
'points' can be used to add additional chuncks - will add points to the previosly exsisting grph with no '+' like in ggplot

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



> Q. run kmeans and cluster in 3 groups and plot the results?

```r
k2<- kmeans(x, centers=3, nstart=20 )

plot(x, col=k2$cluster)
points(k2$centers, pch=15, col='blue')
```



a greater number of centers yeilds a lower sum of squares

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

```
[1] 105.9327
```

```r
k2$tot.withinss
```

```
[1] 82.84164
```

The major limitation of kmeans is that it imposes structure on the data, it will cluster based on the number of groups specified regardless if that is the best structure

#Hierarchical Clusterin hclust() doesn't want the data, it was a distance matrix found by dist() in base R dist() measures euclidian distance by default - symetrical matrix the determines distance between all values

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

```
             1           2           3           4           5           6
2   1.54663100
3   1.27947471  0.43560228
4   3.97599483  2.76557772  2.77643016
5   2.22897536  1.04443673  1.46631195  3.27672537
6   3.04710657  1.66265675  1.77581155  1.17693975  2.10066624
7   2.67161606  1.17919534  1.59066978  2.62377174  0.76241733  1.47564311
8   2.00373829  0.91386731  0.79704498  1.98715432  1.81919723  1.08595461
9   1.17611948  0.91427936  1.04491222  3.67956071  1.10181793  2.56251806
10  2.06080129  0.70203633  1.13631929  2.95511539  0.38278397  1.77942496
11  3.19167411  2.23555582  2.64760543  4.19824157  1.19480032  3.06644009
12  1.08483562  0.96420416  1.05453645  3.72838044  1.20153608  2.62176561
13  0.65532772  1.04613208  0.94528151  3.71885431  1.58050988  2.67901911
14  1.11173426  0.45780480  0.23332018  3.00167798  1.39607848  1.97798711
15  0.69988293  1.55326523  1.15146783  3.53163596  2.47196323  2.74691224
16  2.67909681  1.18364847  1.40664633  1.76764053  1.50950755  0.59115869
17  0.69050205  2.21514156  1.90289385  4.46500451  2.91534794  3.62607737
18  0.32710430  1.36845689  1.18360960  3.94756092  1.94369002  2.95539674
19  1.44603270  0.71338750  0.35568403  2.53442358  1.75562389  1.62982251
20  1.55716747  0.20351202  0.32100031  2.62090262  1.22981596  1.55235786
21  0.57379116  1.07596737  0.73675704  3.40292985  1.93704701  2.48249395
22  1.60516296  0.89562914  1.18838688  3.57233765  0.68206638  2.41474938
23  1.62859618  0.38486576  0.77331850  3.03750359  0.70242796  1.89093195
24  2.95833447  1.53596878  1.96307727  2.92730058  0.81906677  1.82064633
25  2.38141060  0.88761038  1.11254858  1.97699233  1.34067522  0.81397137
26  1.87256321  0.45602751  0.59972813  2.31003329  1.29054669  1.22362528
27  1.29863506  0.42183844  0.01978543  2.76110217  1.45565998  1.75762288
28  2.63510898  1.12473110  1.52739204  2.54112651  0.80698242  1.38732544
29  0.74097814  1.64333849  1.23907977  3.58008632  2.56301606  2.81503831
30  1.79958468  2.83362902  2.40665422  4.20772233  3.78983801  3.71105213
31  4.81281843  6.12656559  5.71306833  7.32957989  7.00935341  7.00703032
32  5.81997837  7.21434682  6.81438704  8.54722328  8.04469264  8.18574188
33  8.55964148  9.97395969  9.57598221 11.24056457 10.78816291 10.93236813
34  7.07354223  8.47198301  8.07134965  9.74798038  9.29978642  9.42397572
35  7.62290846  9.00362375  8.59839424 10.20002265  9.84607237  9.91499631
36  8.11526242  9.48231728  9.07384682 10.61636036 10.33543477 10.36209982
37  8.92587569 10.36242155  9.97059526 11.69486872 11.15453993 11.36024684
38  7.59228115  9.03217667  8.64255514 10.42268317  9.82091534 10.05354355
39  7.55736096  9.03122650  8.65432129 10.55216342  9.77912446 10.12548832
```

40	6.36082833	7.78365087	7.39029094	9.17193375	8.58950714	8.79289214
41	7.37474483	8.77434060	8.37366906	10.03995783	9.60130609	9.72310223
42	7.01011960	8.37756902	7.97020302	9.56434774	9.22955580	9.27779252
43	6.26180691	7.73164726	7.35457710	9.28816216	8.48599722	8.83541062
44	5.28818354	6.75369289	6.37646386	8.34424098	7.51434164	7.86669467
45	8.37998445	9.73463065	9.32360025	10.82097746	10.59685886	10.58920176
46	5.90518903	7.30247072	6.90307503	8.63849427	8.13052134	8.27670605
47	6.97789769	8.39542049	7.99980494	9.73624329	9.20636076	9.38351937
48	6.62310042	8.08803088	7.70847218	9.60983534	8.84832314	9.17547143
49	7.02375443	8.49949935	8.12399195	10.04751998	9.24528156	9.60493659
50	9.06439569	10.56330903	10.19770109	12.15470199	11.27052923	11.70544843
51	8.02794632	9.46789531	9.07787951	10.84240761	10.25650494	10.48345816
52	7.12172266	8.59617581	8.21998930	10.13523722	9.34363970	9.69735854
53	7.47360481	8.80918262	8.39558200	9.88227422	9.68342493	9.64838746
54	8.60939050	10.02912336	9.63257987	11.31242525	10.83821560	10.99729199
55	8.52136687	9.82570165	9.40663030	10.76403357	10.72008079	10.59632787
56	8.20342213	9.66377071	9.28075631	11.10885237	10.42883297	10.72008079
57	8.96529691	10.16186172	9.73187848	10.80367018	11.10885237	10.76403357
58	7.05444062	8.45339087	8.05290172	9.73187848	9.28075631	9.40663030
59	7.43541486	8.85023439	8.45339087	10.16186172	9.66377071	9.82570165
60	5.97648880	7.43541486	7.05444062	8.96529691	8.20342213	8.52136687

	7	8	9	10	11	12
2						
3						
4						
5						
6						
7						
8	1.57655074					
9	1.69666648	1.78640989				
10	0.64975719	1.43650887	1.04693095			
11	1.59202846	2.99351614	2.01950195	1.56793354		
12	1.79004660	1.81495437	0.09973374	1.14038492	2.10755734	
13	2.07633445	1.74215648	0.52540339	1.44303734	2.54490147	0.44025126
14	1.63396807	1.02815753	0.82866394	1.10617777	2.54966061	0.83018349
15	2.73154772	1.66100691	1.59114692	2.20931254	3.56746544	1.52425243
16	0.91432555	0.94144406	2.04151146	1.19337845	2.50326740	2.11024049
17	3.35749412	2.55001037	1.84354393	2.75097908	3.83902830	1.74782154
18	2.43744210	1.96044742	0.86564914	1.80783164	2.87241306	0.77077166
19	1.75544560	0.55912294	1.39723023	1.39930940	2.94849843	1.40166384
20	1.27528882	0.72167040	1.07517716	0.87137299	2.42411077	1.11349580
21	2.24899007	1.43057502	1.06213343	1.69243404	3.02465218	1.00366459
22	1.34824610	1.80874707	0.43564593	0.71836563	1.60962285	0.53386726

23	1.05136901	1.27959961	0.70442322	0.43593325	1.87444199	0.78401018
24	0.42120000	1.99753487	1.89451331	0.89833266	1.27663580	1.99301711
25	0.88131084	0.74686071	1.76111741	0.98973471	2.41970003	1.82604551
26	1.13409167	0.52965139	1.37028235	0.90822223	2.47072061	1.41853573
27	1.57332490	0.78339408	1.05042729	1.12330216	2.63847013	1.06181490
28	0.09648235	1.48539496	1.69028396	0.64718571	1.68316426	1.78158433
29	2.82122654	1.72926897	1.67506292	2.30060130	3.65620074	1.60642851
30	3.99500954	2.68269418	2.88901456	3.51327643	4.88537607	2.81331392
31	7.30244201	6.00953537	5.98892333	6.77679100	8.00108510	5.89725534
32	8.39353094	7.16615207	6.98806157	7.83942829	8.97315231	6.89256526
33	11.15296056	9.92295393	9.71935684	10.59259057	11.68256878	9.62251939
34	9.65117458	8.41333469	8.23919300	9.09670652	10.21709782	8.14321131
35	10.18252261	8.91780810	8.79183848	9.63604247	10.77654371	8.69643796
36	10.66050807	9.37598398	9.28618866	10.12026172	11.27550462	9.19123820
37	11.54016462	10.34016124	10.07700290	10.96982811	12.02249178	9.97933709
38	10.20935473	9.02482748	8.74363277	9.63704907	10.69229470	8.64602776
39	10.20261320	9.07875969	8.69014621	9.61430193	10.60609112	8.59145253
40	8.96197078	7.76468851	7.52053837	8.39656471	9.48770978	7.42378571
41	9.95353175	8.71411540	8.53979381	9.39882659	10.51596689	8.44370548
42	9.55602601	8.28248657	8.18172421	9.01437774	10.17379172	8.08699009
43	8.90353055	7.78432324	7.40001739	8.31709974	9.32995187	7.30163085
44	7.92604589	6.81155630	6.43166247	7.34147383	8.37434319	6.33362882
45	10.91190067	9.61190668	9.55249554	10.37713697	11.54608336	9.45799794
46	8.48163341	7.25660387	7.07253226	7.92645333	9.05588766	6.97689318
47	9.57414035	8.36247009	8.13839841	9.01144003	10.10583727	8.04170918
48	9.26111832	8.12870242	7.76347587	8.67703036	9.69560634	7.66518607
49	9.67023683	8.55485103	8.15630150	9.08100489	10.07382146	8.05761704
50	11.72666716	10.65006853	10.17325540	11.12519654	12.04327288	10.07382146
51	10.64515788	9.45706604	9.17867465	10.07297464	11.12519654	9.08100489
52	9.76727690	8.64837686	8.25496011	9.17867465	10.17325540	8.15630150
53	9.98511951	8.67395615	8.64837686	9.45706604	10.65006853	8.55485103
54	11.20793071	9.98511951	9.76727690	10.64515788	11.72666716	9.67023683
55	10.99729199	9.64838746	9.69735854	10.48345816	11.70544843	9.60493659
56	10.83821560	9.68342493	9.34363970	10.25650494	11.27052923	9.24528156
57	11.31242525	9.88227422	10.13523722	10.84240761	12.15470199	10.04751998
58	9.63257987	8.39558200	8.21998930	9.07787951	10.19770109	8.12399195
59	10.02912336	8.80918262	8.59617581	9.46789531	10.56330903	8.49949935
60	8.60939050	7.47360481	7.12172266	8.02794632	9.06439569	7.02375443

	13	14	15	16	17	18
2						
3						
4						
5						

6						
7						
8						
9						
10						
11						
12						
13						
14	0.71720101					
15	1.10393798	1.10681932				
16	2.22887016	1.57080500	2.49982822			
17	1.33558591	1.76604778	0.94027934	3.30869472		
18	0.36613852	0.97855515	0.94150001	2.53909771	0.97816463	
19	1.24139645	0.57150271	1.13324143	1.37247721	1.99661380	1.42077566
20	1.12688321	0.44636801	1.47245408	1.12465142	2.20266486	1.41812092
21	0.61685624	0.61822501	0.54284216	2.14286235	1.16613832	0.60910410
22	0.95015787	1.02313674	1.96233519	1.85164278	2.27795707	1.29979245
23	1.02531450	0.69461410	1.78478766	1.34752561	2.31802726	1.38614437
24	2.33265932	1.97772763	3.08418576	1.29554290	3.64872382	2.69867440
25	1.93135330	1.27209893	2.22202921	0.29922594	3.01544150	2.23992516
26	1.45647394	0.76922658	1.72066365	0.80757992	2.50259855	1.74583262
27	0.95955307	0.24525246	1.17026044	1.38698025	1.92263994	1.20092767
28	2.05097962	1.58181703	2.67319411	0.82095880	3.31830328	2.40927031
29	1.18206064	1.19778922	0.09134932	2.57884259	0.90759825	1.00156923
30	2.37659699	2.40735481	1.32083215	3.61185882	1.42203431	2.11988381
31	5.46607825	5.68041539	4.57422126	6.94367629	4.16602873	5.12901198
32	6.47459675	6.76072538	5.66428353	8.09129358	5.14474909	6.12242285
33	9.21197931	9.51940355	8.42538010	10.85089257	7.87733058	8.85512978
34	7.72760211	8.01844768	6.92158171	9.34150593	6.39566253	7.37376851
35	8.27776067	8.55193859	7.45111236	9.85041110	6.94864422	7.92618971
36	8.77045075	9.03223067	7.92911788	10.31137686	7.44385407	8.42063635
37	9.57518719	9.90627596	8.81913182	11.26314629	8.23961465	9.21536513
38	8.24156009	8.57567184	7.49116122	9.94381178	6.90598524	7.88180813
39	8.19900354	8.57346389	7.50643082	9.98459388	6.86728901	7.83548089
40	7.01301593	7.32800665	6.23886329	8.68486680	5.67835521	6.65618976
41	8.02865315	8.32078809	7.22393165	9.64277587	6.69631165	7.67445626
42	7.66537467	7.92702673	6.82450127	9.21591018	6.34000622	7.31628836
43	6.90552095	7.27390510	6.20692799	8.68723922	5.57223222	6.54284311
44	5.93389946	6.29597535	5.22908160	7.71146494	4.59937146	5.57223222
45	9.03530594	9.28603414	8.18142942	10.54910028	7.71146494	8.68723922
46	6.55963454	6.84863726	5.75277935	8.18142942	5.22908160	6.20692799
47	7.63039854	7.94029817	6.84863726	9.28603414	6.29597535	7.27390510
48	7.26781330	7.63039854	6.55963454	9.03530594	5.93389946	6.90552095

49	7.66518607	8.04170918	6.97689318	9.45799794	6.33362882	7.30163085
50	9.69560634	10.10583727	9.05588766	11.54608336	8.37434319	9.32995187
51	8.67703036	9.01144003	7.92645333	10.37713697	7.34147383	8.31709974
52	7.76347587	8.13839841	7.07253226	9.55249554	6.43166247	7.40001739
53	8.12870242	8.36247009	7.25660387	9.61190668	6.81155630	7.78432324
54	9.26111832	9.57414035	8.48163341	10.91190067	7.92604589	8.90353055
55	9.17547143	9.38351937	8.27670605	10.58920176	7.86669467	8.83541062
56	8.84832314	9.20636076	8.13052134	10.59685886	7.51434164	8.48599722
57	9.60983534	9.73624329	8.63849427	10.82097746	8.34424098	9.28816216
58	7.70847218	7.99980494	6.90307503	9.32360025	6.37646386	7.35457710
59	8.08803088	8.39542049	7.30247072	9.73463065	6.75369289	7.73164726
60	6.62310042	6.97789769	5.90518903	8.37998445	5.28818354	6.26180691

	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	0.52800693					
21	0.87236563	1.04055457				
22	1.53793511	1.09442972	1.42062435			
23	1.08836835	0.58794663	1.26070604	0.54378068		
24	2.15526173	1.65684620	2.58067306	1.49363167	1.33363245	
25	1.10603003	0.82579596	1.84930333	1.60100834	1.07867197	1.29704559
26	0.63767007	0.32988720	1.33646246	1.31805555	0.77626694	1.54539018
27	0.35397852	0.30235993	0.75650566	1.18661720	0.76462577	1.94689727
28	1.67925455	1.20935082	2.19898279	1.35922189	1.02638733	0.51444757
29	1.20923438	1.56006445	0.63206920	2.05008013	1.87612993	3.17494580
30	2.27310971	2.71972073	1.86277547	3.27716949	3.09682982	4.36878299
31	5.60475150	6.03181270	5.08488474	6.41577477	6.34551155	7.65814079

32	6.73481513	7.13521223	6.15135655	7.42176932	7.40422517	8.73177242
33	9.49663113	9.89687921	8.90800180	10.15453975	10.15688351	11.48700973
34	7.98832547	8.39210621	7.40902122	8.67350571	8.66139800	9.98935698
35	8.50484939	8.91866234	7.94566999	9.22536987	9.20139379	10.52624755
36	8.97245454	9.39357618	8.42860761	9.71903328	9.68626099	11.00833147
37	9.90278912	10.29158565	9.29195591	10.51262894	10.53389504	11.86641795
38	8.58030922	8.96348636	7.96071850	9.17924586	9.20112283	10.53389504
39	8.61425281	8.97423015	7.95566397	9.12528381	9.17924586	10.51262894
40	7.32285562	7.71128994	6.71479562	7.95566397	7.96071850	9.29195591
41	8.29016551	8.69441784	7.71128994	8.97423015	8.96348636	10.29158565
42	7.87264359	8.29016551	7.32285562	8.61425281	8.58030922	9.90278912
43	7.31628836	7.67445626	6.65618976	7.83548089	7.88180813	9.21536513
44	6.34000622	6.69631165	5.67835521	6.86728901	6.90598524	8.23961465
45	9.21591018	9.64277587	8.68486680	9.98459388	9.94381178	11.26314629
46	6.82450127	7.22393165	6.23886329	7.50643082	7.49116122	8.81913182
47	7.92702673	8.32078809	7.32800665	8.57346389	8.57567184	9.90627596
48	7.66537467	8.02865315	7.01301593	8.19900354	8.24156009	9.57518719
49	8.08699009	8.44370548	7.42378571	8.59145253	8.64602776	9.97933709
50	10.17379172	10.51596689	9.48770978	10.60609112	10.69229470	12.02249178
51	9.01437774	9.39882659	8.39656471	9.61430193	9.63704907	10.96982811
52	8.18172421	8.53979381	7.52053837	8.69014621	8.74363277	10.07700290
53	8.28248657	8.71411540	7.76468851	9.07875969	9.02482748	10.34016124
54	9.55602601	9.95353175	8.96197078	10.20261320	10.20935473	11.54016462
55	9.27779252	9.72310223	8.79289214	10.12548832	10.05354355	11.36024684
56	9.22955580	9.60130609	8.58950714	9.77912446	9.82091534	11.15453993
57	9.56434774	10.03995783	9.17193375	10.55216342	10.42268317	11.69486872
58	7.97020302	8.37366906	7.39029094	8.65432129	8.64255514	9.97059526
59	8.37756902	8.77434060	7.78365087	9.03122650	9.03217667	10.36242155
60	7.01011960	7.37474483	6.36082833	7.55736096	7.59228115	8.92587569
	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	0.51286755					
27	1.09280572	0.57999799				
28	0.78528328	1.05260512	1.50962014			
29	2.30312947	1.80401245	1.25773382	2.76235284		
30	3.36834003	2.90922499	2.42335132	3.92773377	1.23081435	
31	6.70218815	6.23876763	5.73086214	7.24045966	4.48368660	3.33384984
32	7.83929877	7.36172612	6.83287137	8.33705668	5.57569850	4.48368660
33	10.60064093	10.12403189	9.59452138	11.09742946	8.33705668	7.24045966
34	9.09190039	8.61661977	8.08979122	9.59452138	6.83287137	5.73086214
35	9.60567963	9.13617379	8.61661977	10.12403189	7.36172612	6.23876763
36	10.07052297	9.60567963	9.09190039	10.60064093	7.83929877	6.70218815
37	11.00833147	10.52624755	9.98935698	11.48700973	8.73177242	7.65814079
38	9.68626099	9.20139379	8.66139800	10.15688351	7.40422517	6.34551155
39	9.71903328	9.22536987	8.67350571	10.15453975	7.42176932	6.41577477
40	8.42860761	7.94566999	7.40902122	8.90800180	6.15135655	5.08488474
41	9.39357618	8.91866234	8.39210621	9.89687921	7.13521223	6.03181270
42	8.97245454	8.50484939	7.98832547	9.49663113	6.73481513	5.60475150
43	8.42063635	7.92618971	7.37376851	8.85512978	6.12242285	5.12901198
44	7.44385407	6.94864422	6.39566253	7.87733058	5.14474909	4.16602873
45	10.31137686	9.85041110	9.34150593	10.85089257	8.09129358	6.94367629
46	7.92911788	7.45111236	6.92158171	8.42538010	5.66428353	4.57422126
47	9.03223067	8.55193859	8.01844768	9.51940355	6.76072538	5.68041539
48	8.77045075	8.27776067	7.72760211	9.21197931	6.47459675	5.46607825
49	9.19123820	8.69643796	8.14321131	9.62251939	6.89256526	5.89725534
50	11.27550462	10.77654371	10.21709782	11.68256878	8.97315231	8.00108510
51	10.12026172	9.63604247	9.09670652	10.59259057	7.83942829	6.77679100
52	9.28618866	8.79183848	8.23919300	9.71935684	6.98806157	5.98892333
53	9.37598398	8.91780810	8.41333469	9.92295393	7.16615207	6.00953537
54	10.66050807	10.18252261	9.65117458	11.15296056	8.39353094	7.30244201
55	10.36209982	9.91499631	9.42397572	10.93236813	8.18574188	7.00703032
56	10.33543477	9.84607237	9.29978642	10.78816291	8.04469264	7.00935341
57	10.61636036	10.20002265	9.74798038	11.24056457	8.54722328	7.32957989

58	9.07384682	8.59839424	8.07134965	9.57598221	6.81438704	5.71306833
59	9.48231728	9.00362375	8.47198301	9.97395969	7.21434682	6.12656559
60	8.11526242	7.62290846	7.07354223	8.55964148	5.81997837	4.81281843
	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	1.23081435					
33	3.92773377	2.76235284				
34	2.42335132	1.25773382	1.50962014			
35	2.90922499	1.80401245	1.05260512	0.57999799		
36	3.36834003	2.30312947	0.78528328	1.09280572	0.51286755	
37	4.36878299	3.17494580	0.51444757	1.94689727	1.54539018	1.29704559
38	3.09682982	1.87612993	1.02638733	0.76462577	0.77626694	1.07867197
39	3.27716949	2.05008013	1.35922189	1.18661720	1.31805555	1.60100834
40	1.86277547	0.63206920	2.19898279	0.75650566	1.33646246	1.84930333

41	2.71972073	1.56006445	1.20935082	0.30235993	0.32988720	0.82579596
42	2.27310971	1.20923438	1.67925455	0.35397852	0.63767007	1.10603003
43	2.11988381	1.00156923	2.40927031	1.20092767	1.74583262	2.23992516
44	1.42203431	0.90759825	3.31830328	1.92263994	2.50259855	3.01544150
45	3.61185882	2.57884259	0.82095880	1.38698025	0.80757992	0.29922594
46	1.32083215	0.09134932	2.67319411	1.17026044	1.72066365	2.22202921
47	2.40735481	1.19778922	1.58181703	0.24525246	0.76922658	1.27209893
48	2.37659699	1.18206064	2.05097962	0.95955307	1.45647394	1.93135330
49	2.81331392	1.60642851	1.78158433	1.06181490	1.41853573	1.82604551
50	4.88537607	3.65620074	1.68316426	2.63847013	2.47072061	2.41970003
51	3.51327643	2.30060130	0.64718571	1.12330216	0.90822223	0.98973471
52	2.88901456	1.67506292	1.69028396	1.05042729	1.37028235	1.76111741
53	2.68269418	1.72926897	1.48539496	0.78339408	0.52965139	0.74686071
54	3.99500954	2.82122654	0.09648235	1.57332490	1.13409167	0.88131084
55	3.71105213	2.81503831	1.38732544	1.75762288	1.22362528	0.81397137
56	3.78983801	2.56301606	0.80698242	1.45565998	1.29054669	1.34067522
57	4.20772233	3.58008632	2.54112651	2.76110217	2.31003329	1.97699233
58	2.40665422	1.23907977	1.52739204	0.01978543	0.59972813	1.11254858
59	2.83362902	1.64333849	1.12473110	0.42183844	0.45602751	0.88761038
60	1.79958468	0.74097814	2.63510898	1.29863506	1.87256321	2.38141060

	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	1.33363245					
39	1.49363167	0.54378068				
40	2.58067306	1.26070604	1.42062435			
41	1.65684620	0.58794663	1.09442972	1.04055457		
42	2.15526173	1.08836835	1.53793511	0.87236563	0.52800693	
43	2.69867440	1.38614437	1.29979245	0.60910410	1.41812092	1.42077566
44	3.64872382	2.31802726	2.27795707	1.16613832	2.20266486	1.99661380
45	1.29554290	1.34752561	1.85164278	2.14286235	1.12465142	1.37247721
46	3.08418576	1.78478766	1.96233519	0.54284216	1.47245408	1.13324143
47	1.97772763	0.69461410	1.02313674	0.61822501	0.44636801	0.57150271
48	2.33265932	1.02531450	0.95015787	0.61685624	1.12688321	1.24139645
49	1.99301711	0.78401018	0.53386726	1.00366459	1.11349580	1.40166384
50	1.27663580	1.87444199	1.60962285	3.02465218	2.42411077	2.94849843
51	0.89833266	0.43593325	0.71836563	1.69243404	0.87137299	1.39930940
52	1.89451331	0.70442322	0.43564593	1.06213343	1.07517716	1.39723023
53	1.99753487	1.27959961	1.80874707	1.43057502	0.72167040	0.55912294
54	0.42120000	1.05136901	1.34824610	2.24899007	1.27528882	1.75544560
55	1.82064633	1.89093195	2.41474938	2.48249395	1.55235786	1.62982251
56	0.81906677	0.70242796	0.68206638	1.93704701	1.22981596	1.75562389
57	2.92730058	3.03750359	3.57233765	3.40292985	2.62090262	2.53442358
58	1.96307727	0.77331850	1.18838688	0.73675704	0.32100031	0.35568403
59	1.53596878	0.38486576	0.89562914	1.07596737	0.20351202	0.71338750
60	2.95833447	1.62859618	1.60516296	0.57379116	1.55716747	1.44603270

	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	0.97816463					
45	2.53909771	3.30869472				
46	0.94150001	0.94027934	2.49982822			
47	0.97855515	1.76604778	1.57080500	1.10681932		
48	0.36613852	1.33558591	2.22887016	1.10393798	0.71720101	
49	0.77077166	1.74782154	2.11024049	1.52425243	0.83018349	0.44025126

50	2.87241306	3.83902830	2.50326740	3.56746544	2.54966061	2.54490147
51	1.80783164	2.75097908	1.19337845	2.20931254	1.10617777	1.44303734
52	0.86564914	1.84354393	2.04151146	1.59114692	0.82866394	0.52540339
53	1.96044742	2.55001037	0.94144406	1.66100691	1.02815753	1.74215648
54	2.43744210	3.35749412	0.91432555	2.73154772	1.63396807	2.07633445
55	2.95539674	3.62607737	0.59115869	2.74691224	1.97798711	2.67901911
56	1.94369002	2.91534794	1.50950755	2.47196323	1.39607848	1.58050988
57	3.94756092	4.46500451	1.76764053	3.53163596	3.00167798	3.71885431
58	1.18360960	1.90289385	1.40664633	1.15146783	0.23332018	0.94528151
59	1.36845689	2.21514156	1.18364847	1.55326523	0.45780480	1.04613208
60	0.32710430	0.69050205	2.67909681	0.69988293	1.11173426	0.65532772

	49	50	51	52	53	54
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  2.10755734
51  1.14038492  1.56793354
52  0.09973374  2.01950195  1.04693095
53  1.81495437  2.99351614  1.43650887  1.78640989
54  1.79004660  1.59202846  0.64975719  1.69666648  1.57655074
55  2.62176561  3.06644009  1.77942496  2.56251806  1.08595461  1.47564311
56  1.20153608  1.19480032  0.38278397  1.10181793  1.81919723  0.76241733
57  3.72838044  4.19824157  2.95511539  3.67956071  1.98715432  2.62377174
58  1.05453645  2.64760543  1.13631929  1.04491222  0.79704498  1.59066978
59  0.96420416  2.23555582  0.70203633  0.91427936  0.91386731  1.17919534
60  1.08483562  3.19167411  2.06080129  1.17611948  2.00373829  2.67161606
            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	2.10066624		
57	1.17693975	3.27672537	
58	1.77581155	1.46631195	2.77643016

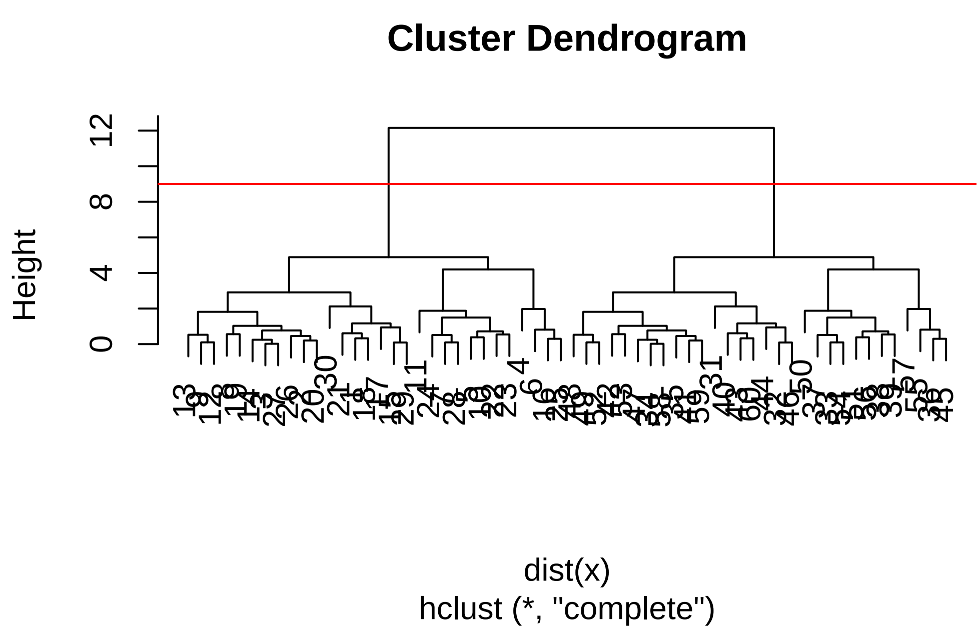
```
59  1.66265675  1.04443673  2.76557772  0.43560228
60  3.04710657  2.22897536  3.97599483  1.27947471  1.54663100
```

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

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

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

plotting hc

```r
plot(hc)
abline(h=9, col='red')
```



to get the cluster membership vector, we need to cut the tree at a given height of our choosing.

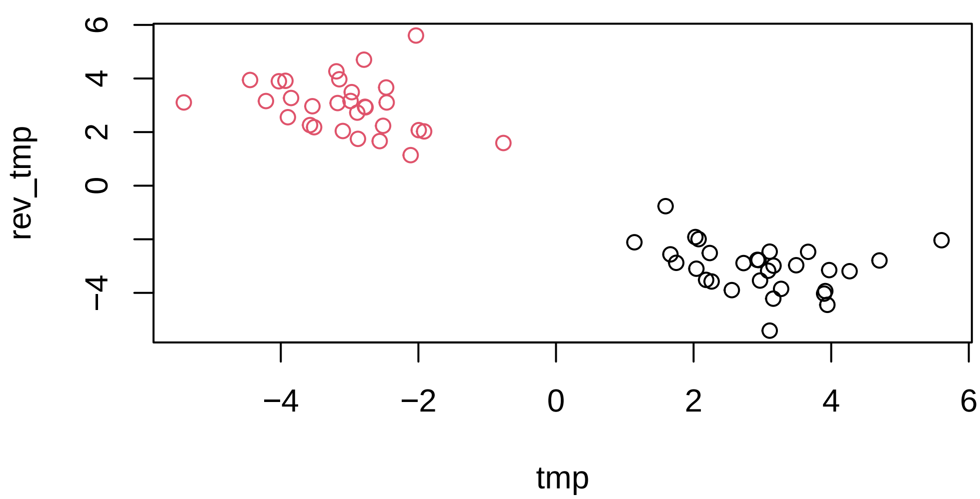
The function to do this is 'cutree()' h cuts at a height, k cuts into that number of clusters

```
cutree(hc, h=9)
```

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

> Q Plot x, colored with the hclust results

```
grps <- cutree(hc, k=2)

plot(x, col=grps)
```



#Principal Component Analysis (PCA)

PCA of food UK data

data import

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

#PCA

function to do PCA in base R is prcomp() - foods and columns and countries as rows the table needs to be transposed

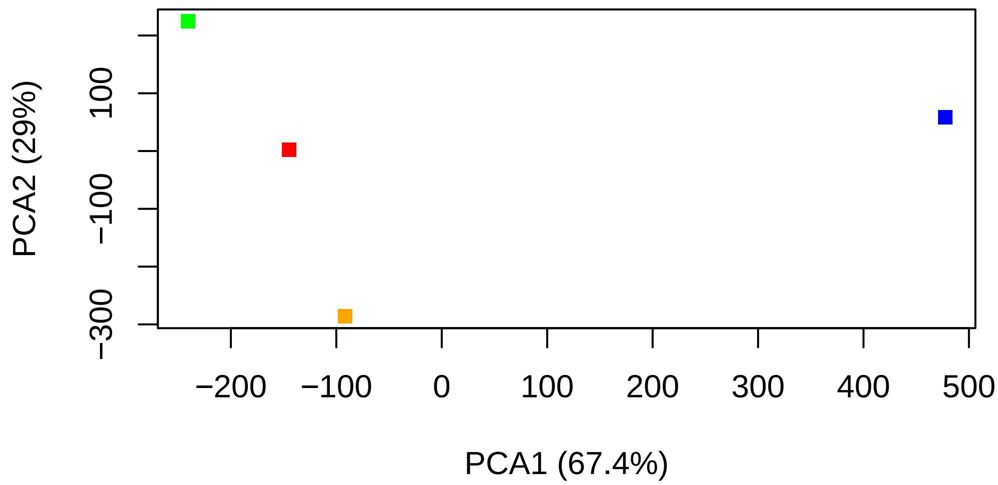
```
pca <-prcomp(t(x))
summary(pca)
```

```
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
```

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

```
plot(pca$x[,1],pca$x[,2], xlab='PCA1 (67.4%)', ylab='PCA2 (29%)', col=c('red','green','ora
abline()
```



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

```r
nrow(x)
```

```
[1] 17
```

```r
ncol(x)
```

```
[1] 4
```

Checking the data - view the first section of the data

```r
head(x)
```

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

```
Fats_and_oils       193    235       184        209
Sugars              156    175       147        139
```

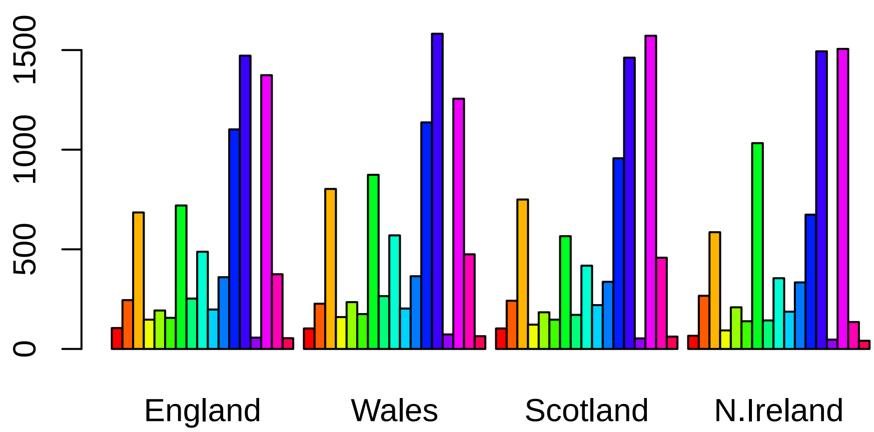
> Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

The method done during the read.csv step is simplier and allows for the merging of two steps rather than having to do a second step to make the data frame four rows

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

Changing beside from true to false will yeild a stacked bar plot based on country

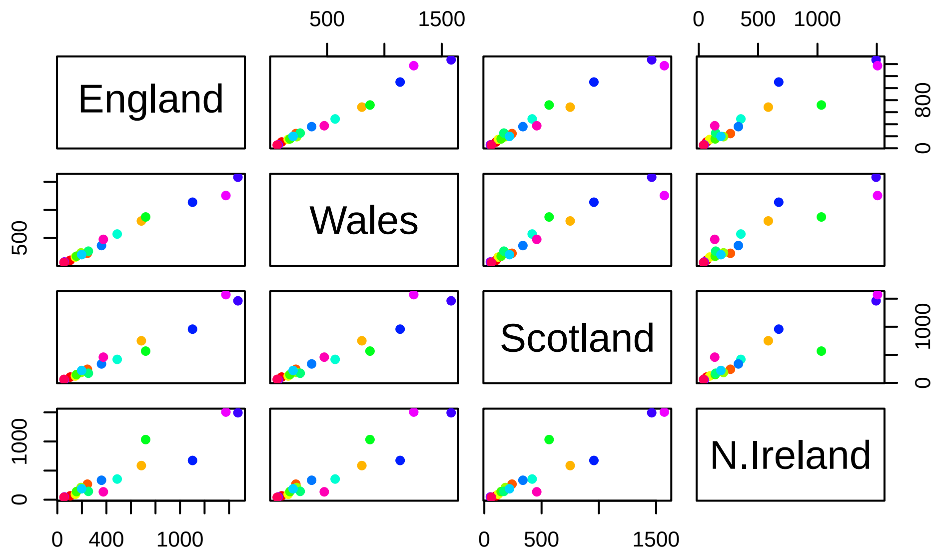
```r
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



> 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?

The pairs plot shows similarity between the two countries where if the frequency a food is eaten is the same between two countries, it will lie on the diagonal one useful plot is a pairs plot

```
pairs(x, col=rainbow(17), pch=16)
```



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

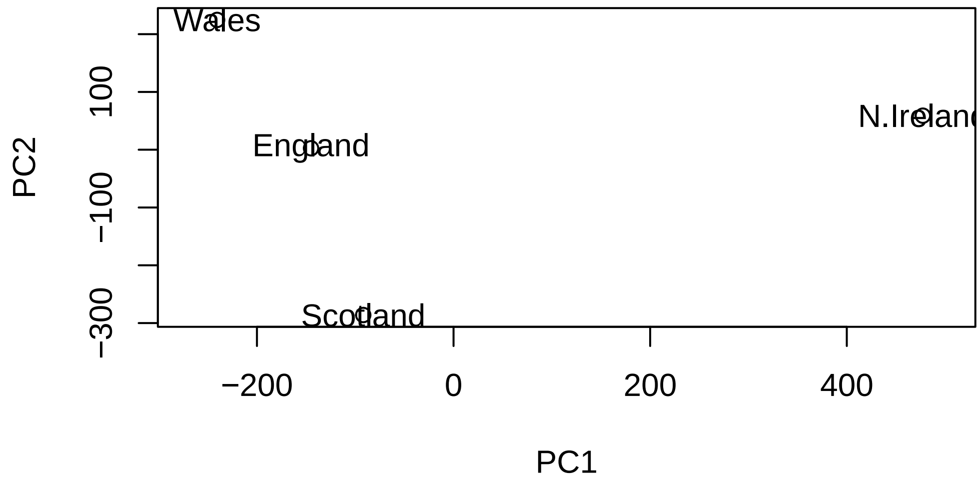
The greatest variation in the data when comparing N.Ireland to the other countries comes from the ammount of Fresh potatoes, fresh fruit and acoholic drinks being either noticably higher or lower in frequency.

```
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
```

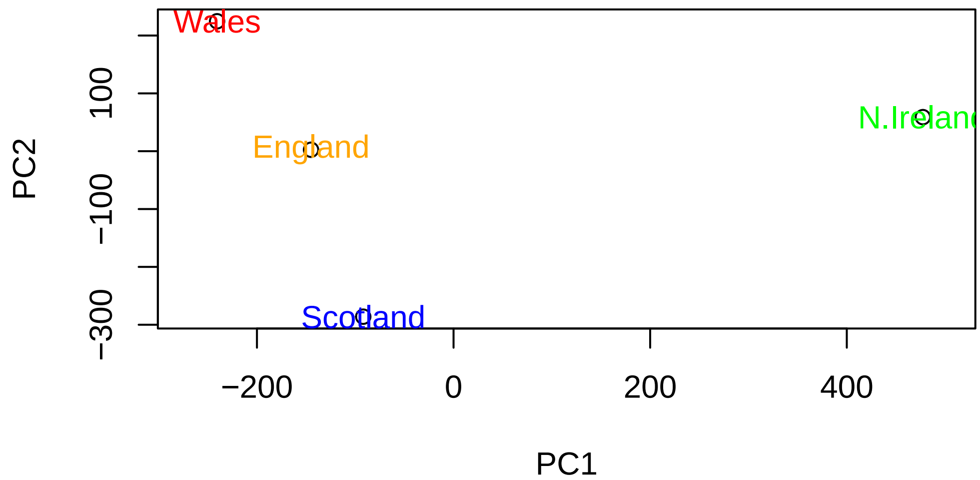
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

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

```r
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x),col=c('orange','red','blue','green'))
```



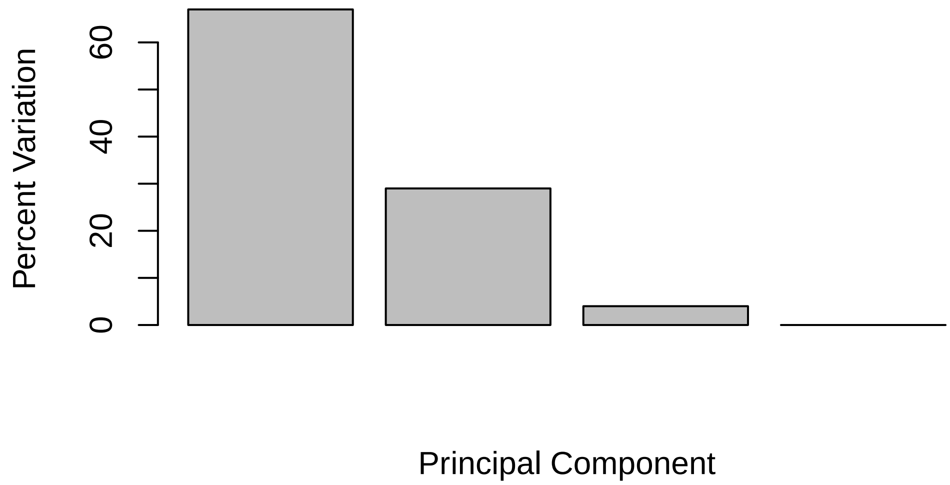

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

```
[1] 67 29  4  0
```

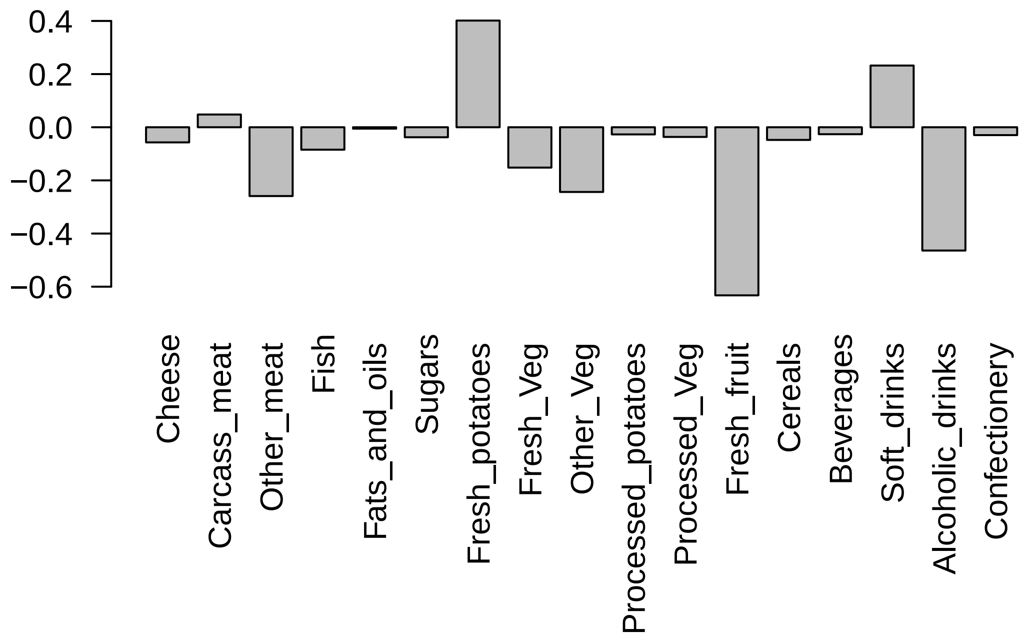
```
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 4.188568e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



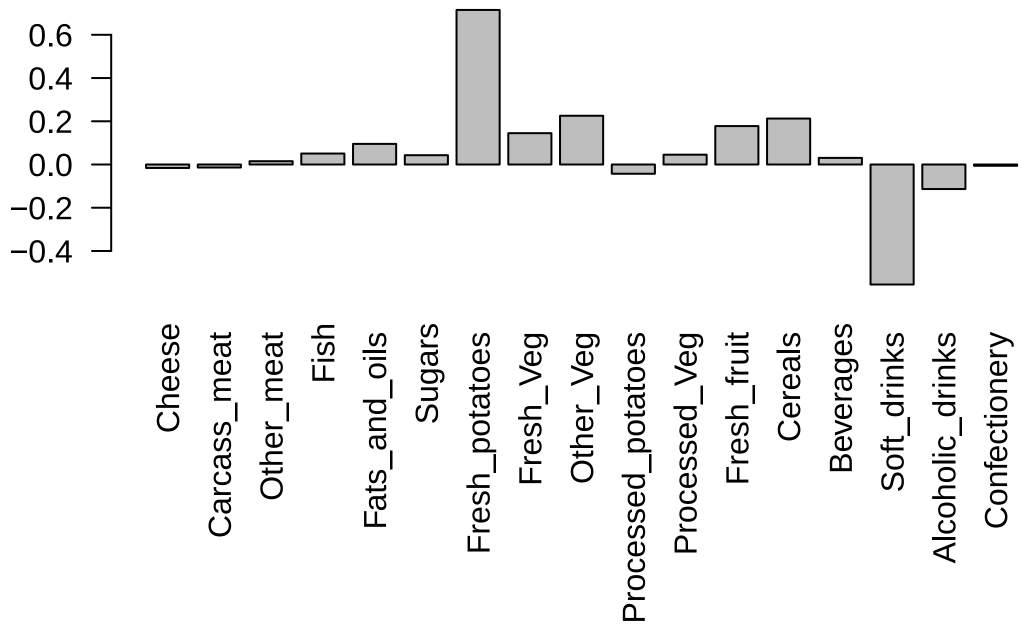

```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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

PC2 is mostly affected by the data for fresh potatoes and soft drinks with fresh potatoes hacing a positive loading score that pushes N.Ierland to the right while soft drinks has a high negative score that pushes it to the left of the plot.

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



#making the plots in ggplot

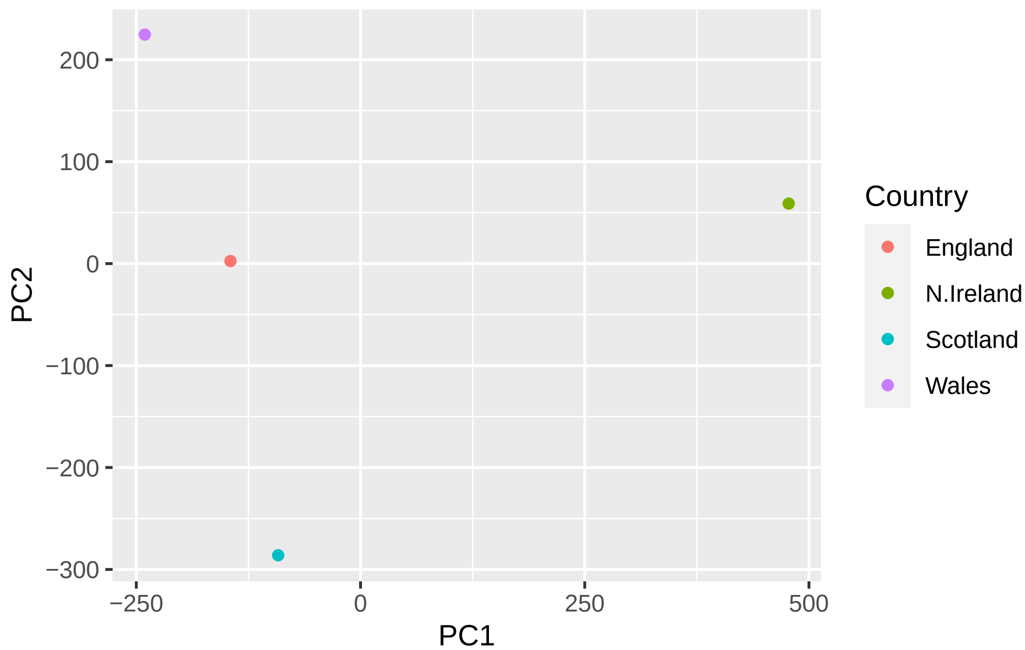
Basic PCA

```
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.2.3
```

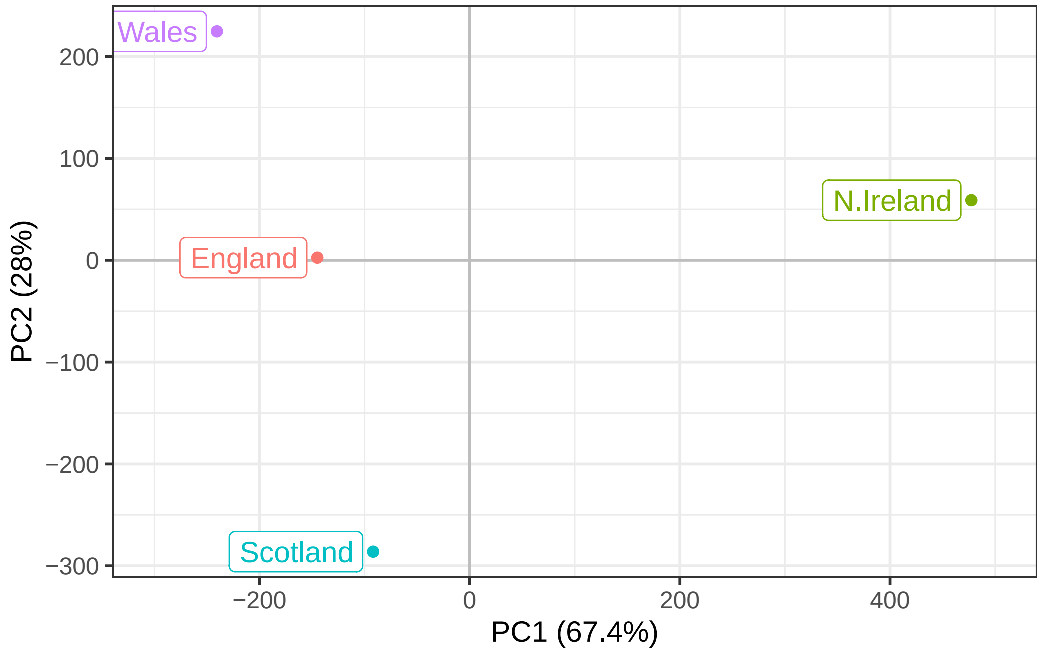
```
df <- as.data.frame(pca$x)
df_lab <- tibble::rownames_to_column(df, "Country")
```

```r
ggplot(df_lab) +
  aes(PC1, PC2, col=Country) +
  geom_point()
```



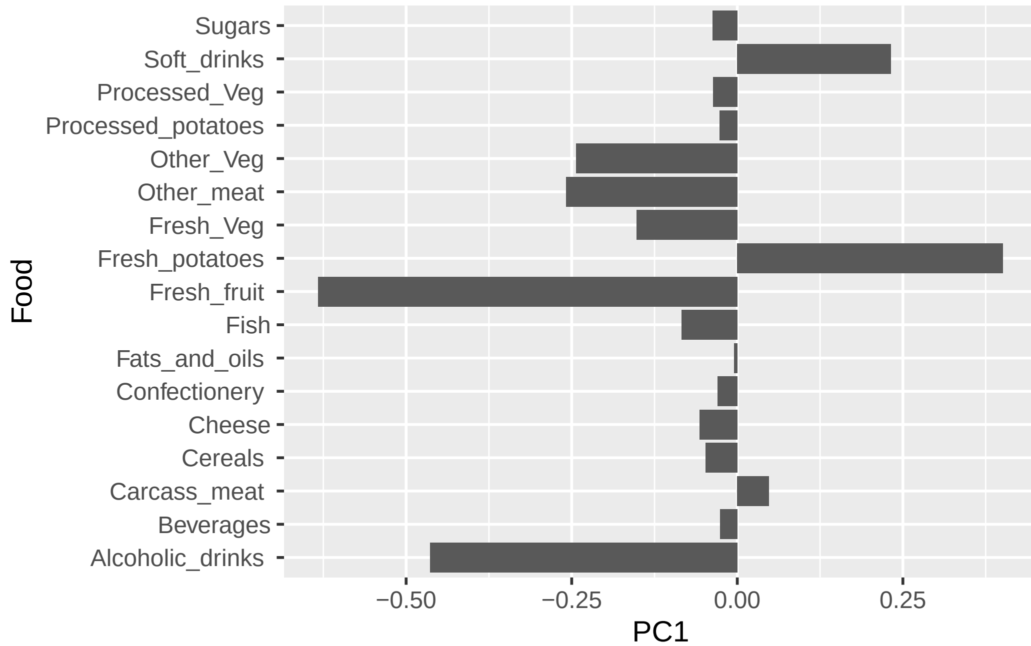
Making the PCA look nicer

```r
ggplot(df_lab) +
  aes(PC1, PC2, col=Country, label=Country) +
  geom_hline(yintercept = 0, col="gray") +
  geom_vline(xintercept = 0, col="gray") +
  geom_point(show.legend = FALSE) +
  geom_label(hjust=1, nudge_x = -10, show.legend = FALSE) +
  expand_limits(x = c(-300,500)) +
  xlab("PC1 (67.4%)") +
  ylab("PC2 (28%)") +
  theme_bw()
```



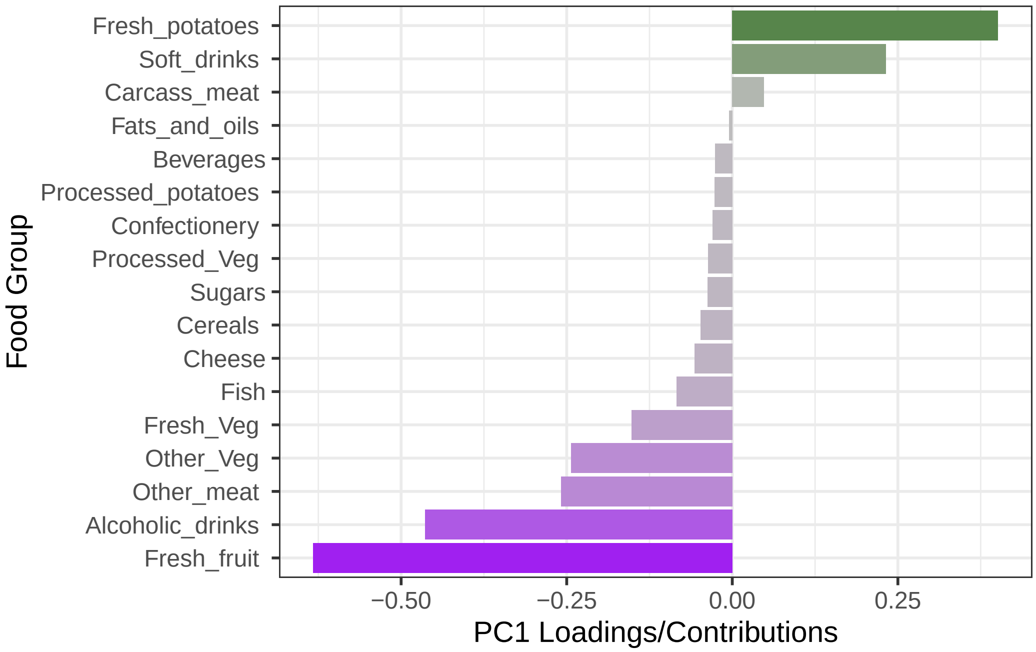
Basic loadings graph

```
ld <- as.data.frame(pca$rotation)
ld_lab <- tibble::rownames_to_column(ld, "Food")

ggplot(ld_lab) +
  aes(PC1, Food) +
  geom_col()
```



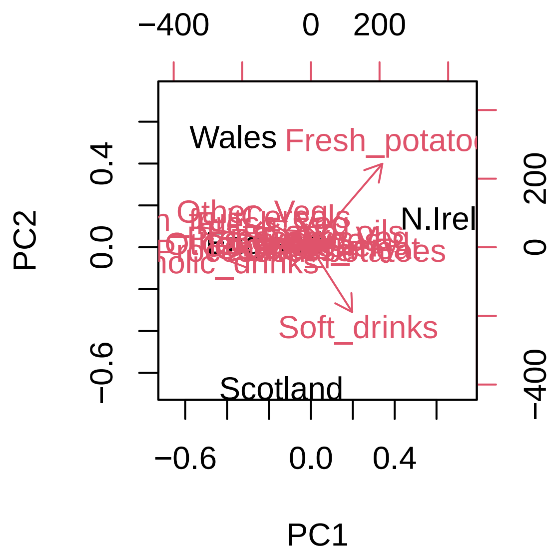
Loadings graph that looks nicer

```
ggplot(ld_lab) +
  aes(PC1, reorder(Food, PC1), bg=PC1) +
  geom_col() +
  xlab("PC1 Loadings/Contributions") +
  ylab("Food Group") +
  scale_fill_gradient2(low="purple", mid="gray", high="darkgreen", guide=NULL) +
  theme_bw()
```



biplot

```
biplot(pca)
```



#PCA of RNA-sqe Data

```r
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

```r
nrow(rna.data)
```

```
[1] 100
```

```r
ncol(rna.data)
```

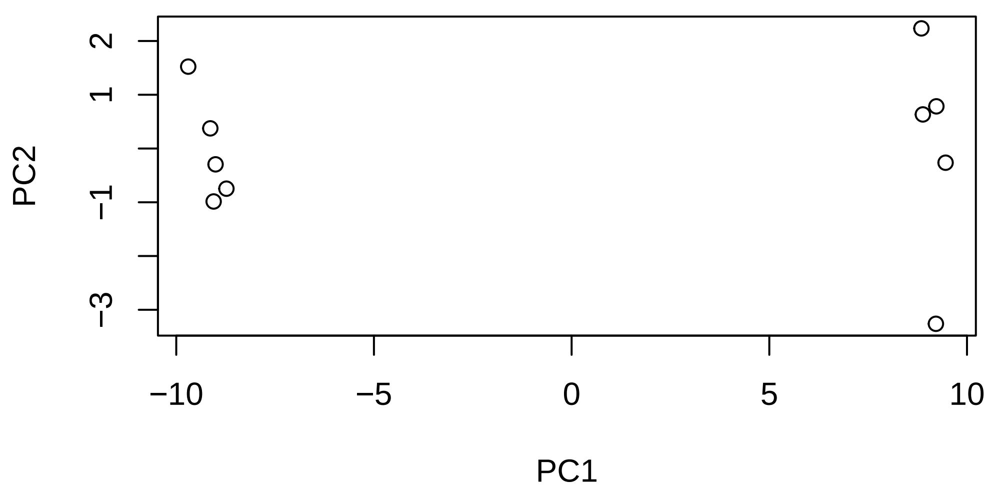
```
[1] 10
```

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

there are 100 genes and 10 samples

RNA seq PCA

```
pca <- prcomp(t(rna.data), scale=TRUE)

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



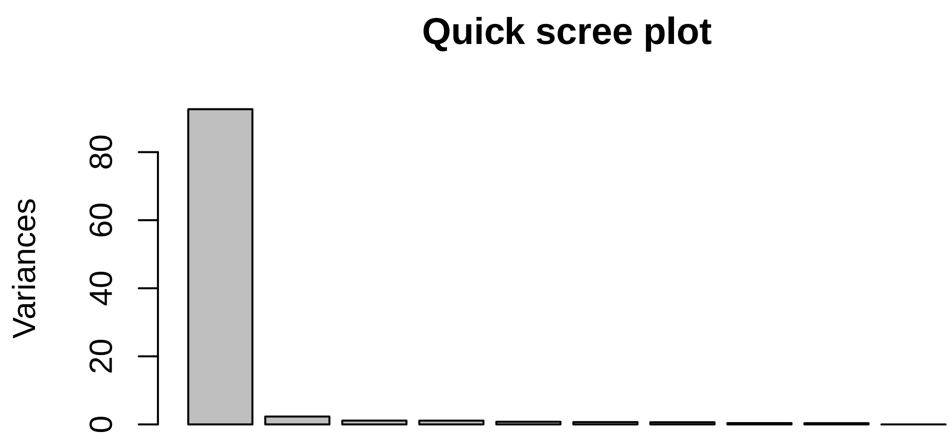

```
summary(pca)
```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     9.6237 1.5198 1.05787 1.05203 0.88062 0.82545 0.80111
Proportion of Variance 0.9262 0.0231 0.01119 0.01107 0.00775 0.00681 0.00642
Cumulative Proportion  0.9262 0.9493 0.96045 0.97152 0.97928 0.98609 0.99251
                           PC8     PC9      PC10
Standard deviation     0.62065 0.60342 3.348e-15
```

```
Proportion of Variance 0.00385 0.00364 0.000e+00
Cumulative Proportion  0.99636 1.00000 1.000e+00
```

scree plot

```r
plot(pca, main="Quick scree plot")
```

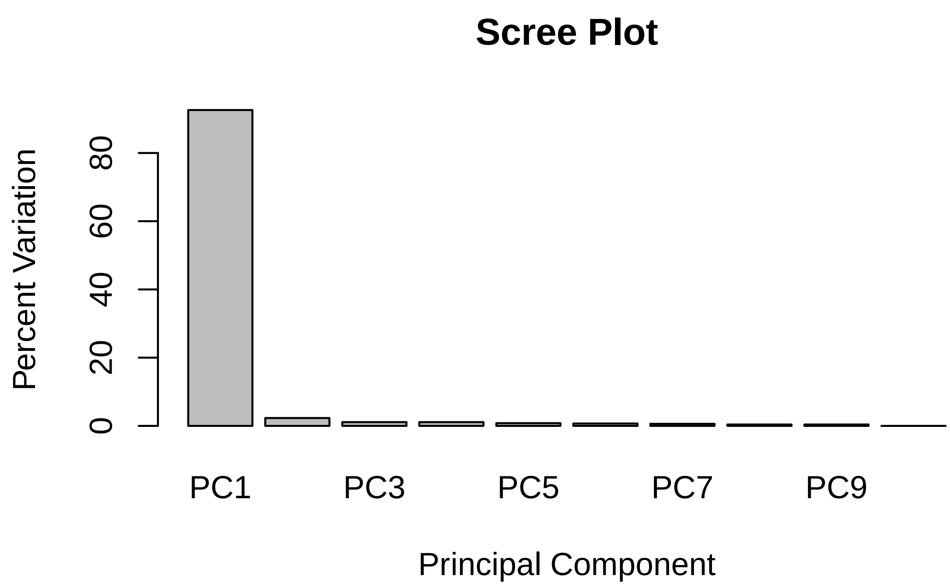


scree plot made with info from prcomp function

```r
pca.var <- pca$sdev^2

pca.var.per <- round(pca.var/sum(pca.var)*100, 1)
pca.var.per
```

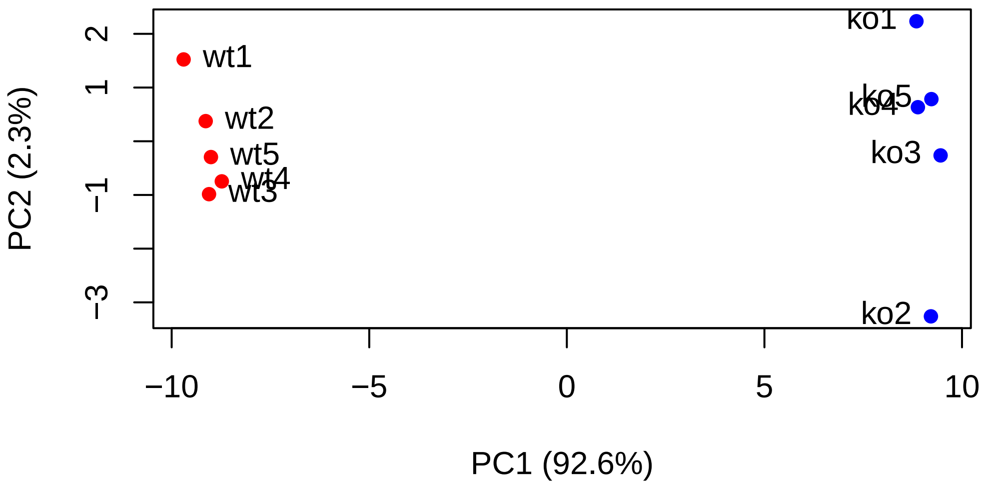
```
[1] 92.6  2.3  1.1  1.1  0.8  0.7  0.6  0.4  0.4  0.0
```

```r
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```



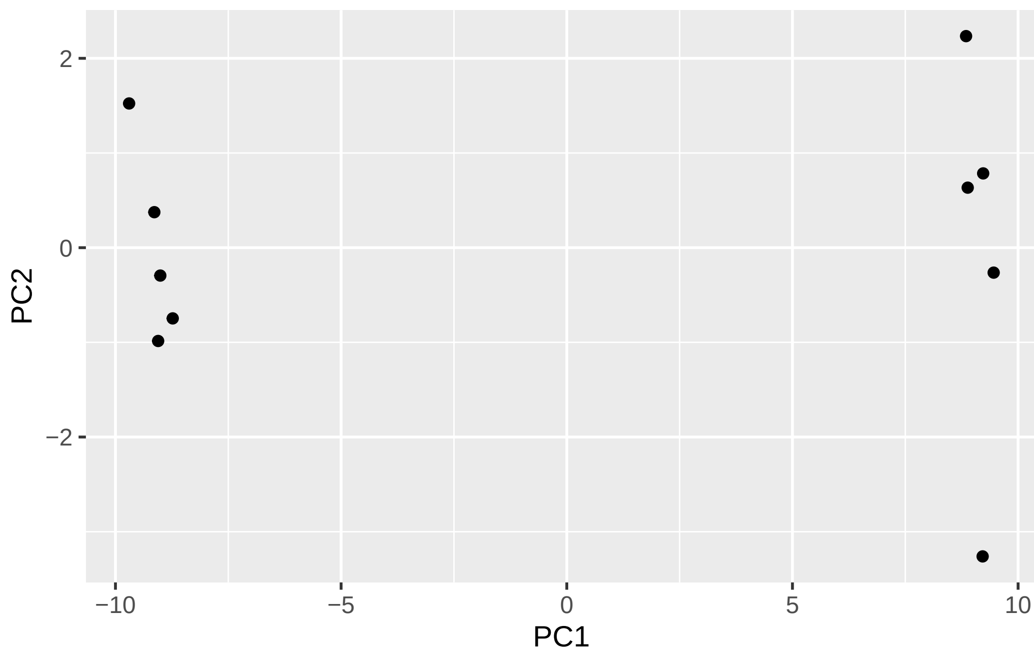
A better looking pca using base R

```
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

plot(pca$x[,1], pca$x[,2], col=colvec, pch=16,
     xlab=paste0("PC1 (", pca.var.per[1], "%)"),
     ylab=paste0("PC2 (", pca.var.per[2], "%)"))

text(pca$x[,1], pca$x[,2], labels = colnames(rna.data), pos=c(rep(4,5), rep(2,5)))
```

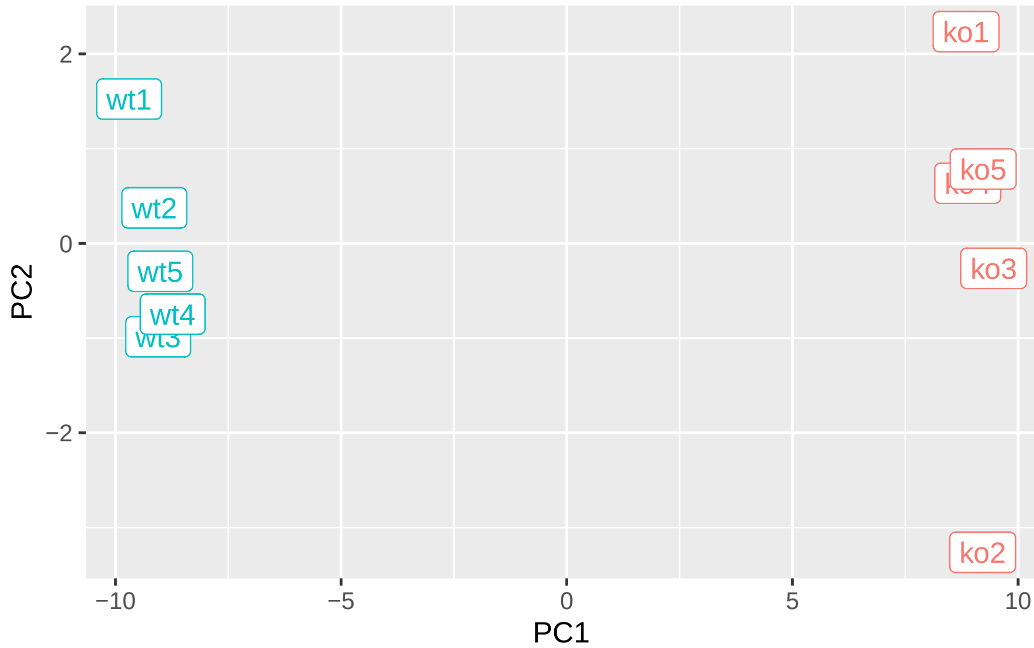


Using ggplot to make better looking graphs

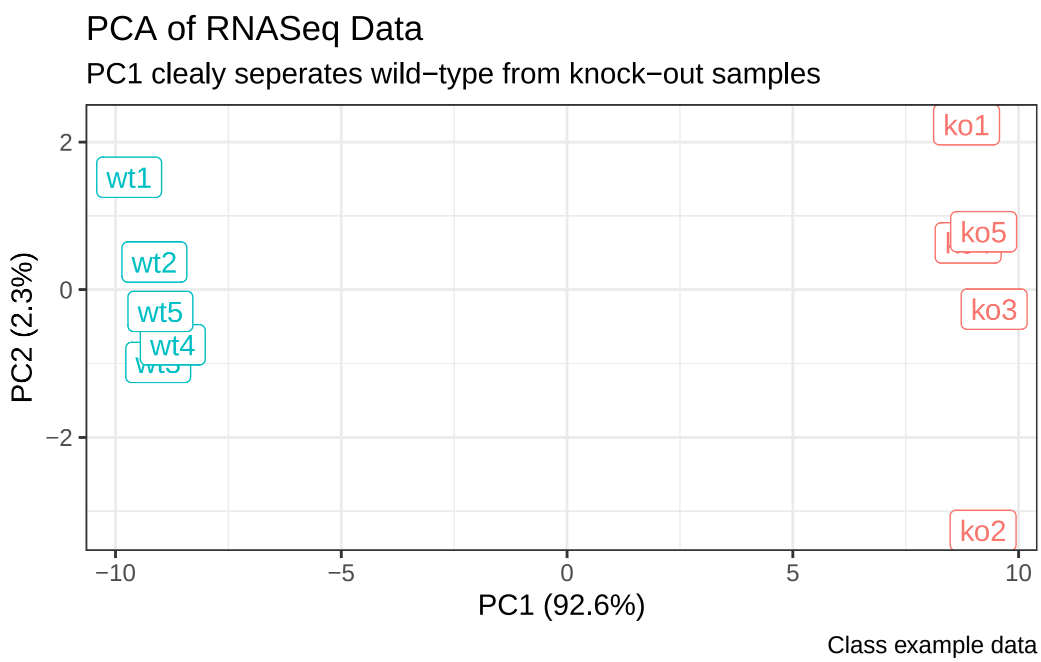
```
library(ggplot2)

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

ggplot(df) +
  aes(PC1, PC2) +
  geom_point()
```




```
df$samples <- colnames(rna.data)
df$condition <- substr(colnames(rna.data),1,2)

p <- ggplot(df) +
        aes(PC1, PC2, label=samples, col=condition) +
        geom_label(show.legend = FALSE)
p
```




```
p + labs(title="PCA of RNASeq Data",
        subtitle = "PC1 clealy seperates wild-type from knock-out samples",
        x=paste0("PC1 (", pca.var.per[1], "%)"),
        y=paste0("PC2 (", pca.var.per[2], "%)"),
        caption="Class example data") +
      theme_bw()
```


PCA of RNASeq Data
PC1 clealy seperates wild−type from knock−out samples
2
0
−2
PC2 (2.3%)
wt1
wt2
wt5
wt4
ko1
ko5
ko3
ko2
−10
−5
0
5
10
PC1 (92.6%)
Class example data