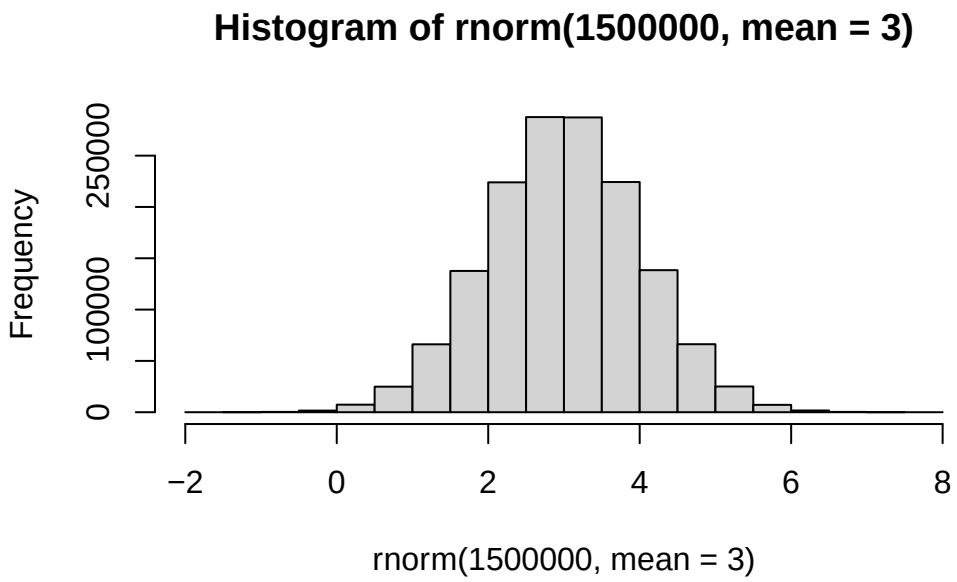
Class7: Machine Learning 1

Mudit (PID: 911)

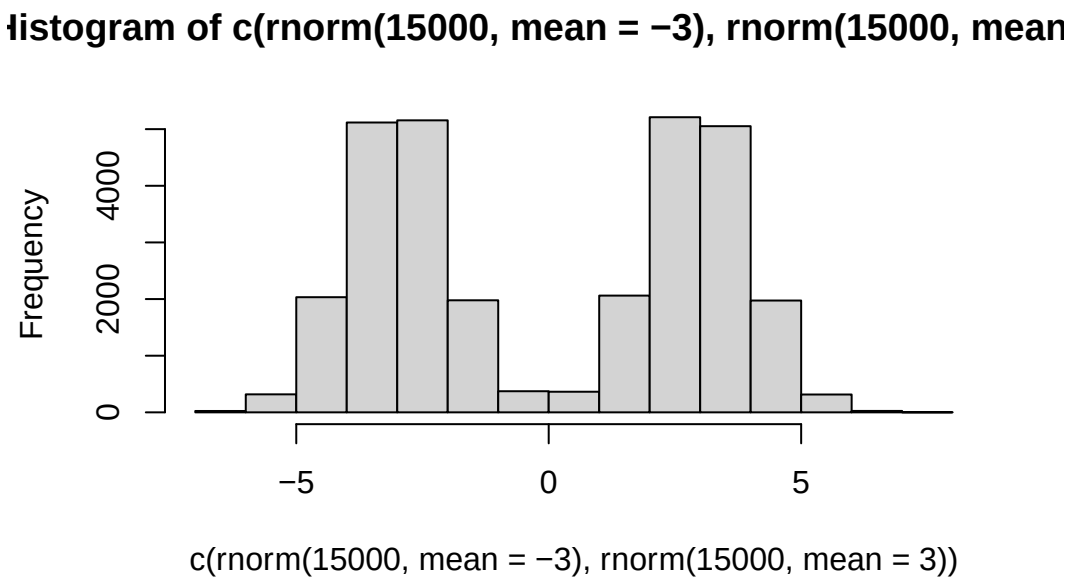
Before we get into clustering methods, let's make some sample to cluster where we know what the answer should be.

To help whith this I will use the `rnorm()` function

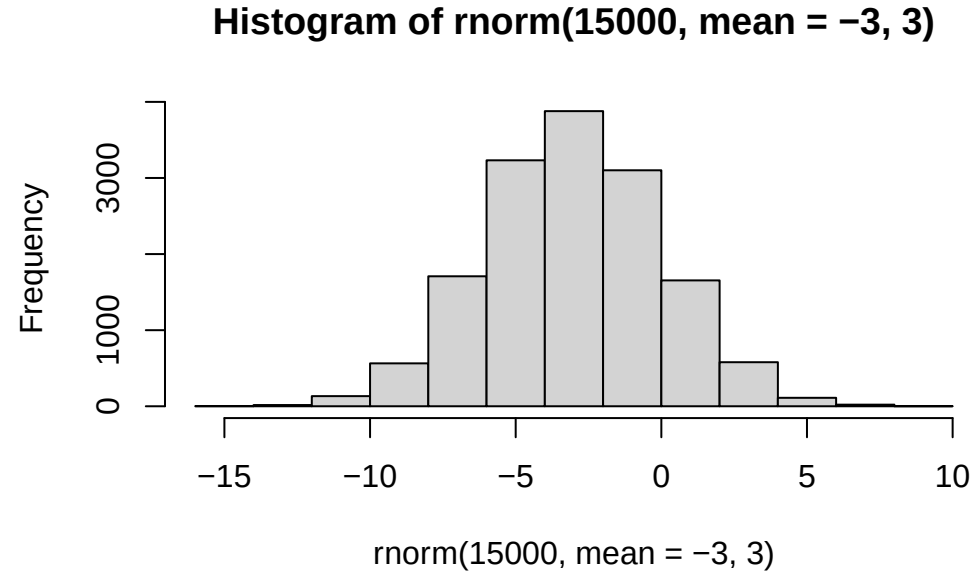
```
hist(rnorm(1500000, mean = 3))
```




```
hist(c(rnorm(15000, mean = -3), rnorm(15000, mean = 3)))
```




```
hist(rnorm(15000, mean = -3, 3))
```



```r
c(rnorm(1000, mean = -3), rnorm(1000, mean = 3))
```

```
   [1] -3.78204638 -3.08157852 -2.06583139 -2.64280907 -2.94330385 -4.21376137
   [7] -2.00916278 -2.67534316 -2.72003814 -3.71549026 -2.61309941 -1.90360620
  [13] -1.80446856 -4.62831871 -1.68098416 -5.38701517 -3.92231378 -2.65323104
  [19] -2.69450785 -4.50404602 -4.54168297 -1.82826307 -0.90022851 -2.97141136
  [25] -4.82681145 -1.13467322 -2.46038805 -4.92019470 -2.84494826 -3.72669639
  [31] -3.55306647 -2.07102036 -3.44942272 -1.85139116 -2.60528576 -2.74475383
  [37] -3.29163665 -0.34983234 -2.71957338 -2.93851714 -3.04209694 -3.28201081
  [43] -3.01752553 -2.01261286 -3.23295627 -5.08785595 -2.89736464 -3.91474015
  [49] -2.72009801 -3.50957884 -1.93282643 -2.70623587 -2.24202864 -2.97899318
  [55] -2.58283815 -2.86487574 -1.61527354 -2.21065529 -3.59769537 -4.46991649
  [61] -3.66736879 -2.17053439 -2.70540766 -4.96115357 -3.97691691 -2.86360684
  [67] -3.49202512 -3.71155948 -3.70183264 -2.50885320 -2.15658589 -2.28596358
  [73] -4.47462377 -3.90396322 -3.19057683 -4.27789615 -1.31323529 -2.25119915
  [79] -2.67958768 -1.87598271 -3.22553343 -3.88241241 -3.55181855 -3.98763221
  [85] -3.52275918 -3.68260803 -1.74211595 -3.36212269 -2.00182664 -3.91317614
  [91] -3.17178062 -1.97032193 -1.90646541 -3.58737581 -4.11113750 -3.21453792
  [97] -2.72219107 -2.64742117 -4.89721918 -3.08083534 -5.57206962 -4.08221842
 [103] -1.87192718 -2.30731607 -1.94122664 -3.49235888 -2.17601100 -1.91328071
 [109] -2.89101157 -2.72339778 -1.61118823 -2.55631415 -1.70198680 -2.62872342
 [115] -2.05556668 -1.79236140 -1.83924931 -1.19329774 -3.82987077 -2.54400277
 [121] -2.88003215 -3.25985379 -1.45411523 -3.52527736 -3.98818372 -1.38791615
 [127] -1.82925201 -2.80087585 -2.62728650 -4.38187140 -0.58364620 -2.61279166
 [133] -2.93235314 -2.59680001 -1.15024226 -3.11913624 -3.49565356 -5.13014761
 [139] -3.33668519 -3.31229422 -2.70250551 -3.10703217 -2.53592034 -4.28302791
 [145] -2.02139487 -3.67210563 -2.62817479 -4.01521753 -2.84198411 -2.45968852
 [151] -3.12625138 -4.15360316 -3.81091291 -2.65560579 -3.65746506 -3.37541340
 [157] -2.19618582 -2.64135798 -2.38961939 -2.52388497 -0.75902054 -2.58568550
 [163] -2.90670980 -2.71624288 -3.31030936 -2.19448338 -4.05717376 -3.01449628
 [169] -2.99306847 -2.45511551 -2.01287651 -3.34487265 -1.65197012 -1.67928887
 [175] -3.03581609 -2.66151380 -4.07908368 -4.26537490 -2.19903714 -3.15803681
 [181] -4.11465179 -2.43043648 -1.00812582 -1.88787566 -3.30063074 -2.57611701
 [187] -2.41140197 -2.31740537 -2.20779310 -3.87229550 -2.78837266 -4.06275952
 [193] -2.37805856 -3.53327440 -3.84781562 -2.35254526 -4.75759964 -5.64354484
 [199] -3.26566533 -2.12523675 -2.23037111 -2.66920841 -3.64551531 -4.89411640
 [205] -2.24053261 -3.96028251 -2.93285798 -3.93686708 -1.31156708 -2.05009147
 [211] -3.45044482 -3.31016288 -2.90160172 -0.84991055 -2.15760538 -3.45420762
 [217] -2.48556016 -3.29767478 -2.52848238 -4.70726521 -2.31939615 -0.87345823
 [223] -3.72417082 -4.42748549 -3.91008434 -4.75878674 -2.01497902 -3.75611700
 [229] -3.52805059 -2.61166760 -4.69921114 -3.02724274 -3.00183916 -3.38515456
 [235] -2.50437470 -4.45800898 -3.43969838 -2.36761680 -1.80669746 -3.15301979
```

```
[241] -2.08929813 -2.33910966 -3.00158276 -2.83908001 -3.52362270 -2.92341499
[247] -2.36096908 -2.11680858 -0.92665875 -2.27760012 -3.47472933 -2.68420388
[253] -3.18694498 -3.25304375 -1.07627928 -4.06294355 -3.42026395 -2.08524111
[259] -3.89660597 -3.76402412 -2.82362381 -2.14986190 -2.78106887 -2.51756299
[265] -4.24495121 -0.77415931 -5.33280431 -5.14362669 -3.60383655 -3.92317806
[271] -3.47787089 -3.75291302 -4.00845622 -2.77889058 -2.22904024 -4.12493793
[277] -3.24970579 -2.74895153 -2.98641575 -3.12047769 -2.30543611 -3.89669295
[283] -3.69046116 -3.92208744 -3.60913337 -2.64897792 -2.05759280 -2.56973175
[289] -1.54225745 -2.99422461 -1.31447851 -4.67392701 -3.67703394  0.06391750
[295] -1.22510266 -3.31441874 -3.09321083 -3.82549010 -3.34638839 -2.10926421
[301] -1.31938506 -4.79139924 -4.63788173 -2.59844340 -1.20474996 -4.16596464
[307] -3.33180004 -2.95714154 -2.00501809 -3.33465164 -1.39345081 -3.62675508
[313] -4.04286134 -2.39080850 -2.65879350 -2.67006805 -3.21624160 -3.61842331
[319] -2.87119303 -3.04200456 -2.88143015 -1.89529353 -3.47439736 -5.47374935
[325] -3.67369954 -2.72970842 -3.58033403 -4.62562473 -2.16366365 -2.26512034
[331] -2.58495846 -2.68848059 -2.83857289 -2.08990281 -4.39197668 -3.53712843
[337] -1.84653900 -2.53213330 -3.93488497 -3.26784914 -3.21556739 -3.00494959
[343] -3.00083118 -2.44806157 -3.73082923 -5.12513334 -3.34191567 -2.44505764
[349] -3.14852735 -3.72074025 -2.62382371 -3.17769793 -3.73716034 -3.50539695
[355] -2.52011522 -4.02345993 -4.06184667 -4.28906592 -2.82483685 -3.82895779
[361] -1.70685943 -3.05355266 -1.69651237 -3.97458117 -3.32442573 -3.43457079
[367] -1.88310212 -4.02998425 -1.62752967 -2.82775003 -1.97693382 -2.89052183
[373] -1.73155905 -2.34635440 -3.34445572 -3.50501313 -2.22501167 -2.97481446
[379] -4.70474189 -1.16026206 -2.49670577 -2.19420220 -2.19312374 -3.02760959
[385] -2.60936674 -4.34175455 -3.00861919 -3.42354012 -2.77378028 -3.02986473
[391] -3.22581550 -3.20216709 -3.71525929 -2.53248188 -2.84903381 -3.99691182
[397] -2.19027925 -3.94520732 -3.66913069 -2.65055742 -3.67147729 -3.74235939
[403] -3.12674728 -3.24823486 -3.13764885 -1.73849090 -2.43329006 -4.01567989
[409] -2.37273627 -2.16744856 -1.19043147 -3.13900143 -3.29664812 -2.28416224
[415] -3.48115884 -3.32947736 -2.39035590 -3.54446076 -2.20164341 -3.16696706
[421] -3.76913718 -2.70040093 -1.92630283 -2.62779879 -2.77438187 -2.92617701
[427] -2.60441236 -3.76050041 -2.24624414 -1.51368639 -4.01684306 -3.46228247
[433]  0.49135867 -2.98992149 -2.17876369 -3.07257536 -2.80207840 -3.14657541
[439] -3.46619633 -1.07634292 -1.93964271 -2.98389364 -3.15160803 -2.63545105
[445] -1.99573211 -1.94640551 -3.92112146 -2.60850048 -2.15657895 -1.62929248
[451] -1.87428189 -3.26281359 -3.84455265 -3.68247943 -1.70996002 -4.21621187
[457] -2.64258042 -2.77316328 -4.40306938 -2.71975836 -3.42793224 -2.16798571
[463] -4.04674915 -2.64149226 -2.84464832 -5.40930257 -2.80012192 -3.23371118
[469] -3.90916014 -2.48180327 -3.64517918 -4.42152587 -2.63446738 -1.86760449
[475] -3.74153026 -4.97417900 -4.29227081 -2.88269119 -3.87622952 -3.23845128
[481] -3.44513169 -3.33903044 -2.83220577 -2.86381792 -2.24230139 -4.54724807
[487] -3.21430297 -2.84388838 -3.16657376 -3.05999389 -2.91611747 -2.11207047
[493] -3.31725533 -1.42167461 -2.77686773 -2.92591237 -2.86000861 -2.49297223
```

```
[499] -2.55007765 -0.65077401 -3.48071600 -1.38210706 -3.42158989 -3.55803657
[505] -3.54224430 -1.93832916 -3.78733749 -4.48754382 -2.89827845 -3.20156516
[511] -3.47851366 -2.46532839 -3.01399360 -1.82971427 -1.86881703 -2.81860366
[517] -3.33598748 -5.61807799 -1.29115195 -2.17274249 -4.14391455 -3.64431147
[523] -2.73638042 -2.14741879 -2.07060953 -4.34568959 -4.39497147 -3.96857707
[529] -3.57615270 -4.06017096 -1.32469565 -4.14527566 -1.97859592 -3.18337886
[535] -1.34146410 -4.39103855 -3.21768572 -4.50181217 -3.97550317 -2.43125357
[541] -3.36204377 -1.57718686 -4.25663058 -3.18163939 -4.43663294 -2.04634987
[547] -2.62764521 -3.97307393 -2.99377137 -2.92446995 -2.71149687 -4.21698281
[553] -2.54303862 -4.04461001 -1.96098137 -4.68961124 -3.62303233 -2.84300692
[559] -2.43894838 -2.32221150 -3.12850012 -2.73934128 -2.06318243 -2.60830618
[565] -3.05778062 -2.56624053 -1.78987440 -2.37983030 -5.00237638 -2.70490561
[571] -1.02754760 -2.36074242 -2.55909771 -3.11221031 -2.59328869 -2.90649876
[577] -4.68777384 -1.70026836 -5.91429326 -5.58681816 -2.86259824 -3.49277406
[583] -2.91913642 -0.13320984 -2.19032984 -4.88087873 -3.93927020 -2.18868913
[589] -0.69399398 -2.98727846 -1.46272836 -2.66131066 -4.27692075 -2.52385307
[595] -2.71515789 -1.93219812 -3.97686085 -3.53443261 -3.86145250 -3.10963423
[601] -2.93789931 -3.07689098 -3.10783285 -2.77862130 -1.79645940 -4.72759150
[607] -1.28088902 -1.52918900 -2.26969805 -1.32011541 -2.33783076 -4.63804536
[613] -2.53035876 -2.92767891 -4.37508673 -3.95977724 -1.30077494 -3.23861655
[619] -2.28544758 -0.87077964 -1.02112112 -2.15573238 -3.79050104 -2.61245405
[625] -3.17037231 -3.80310350 -2.87388846 -0.74225607 -2.79735249 -3.22785887
[631] -3.79328926 -2.09886263 -1.32666616 -2.52995878 -2.57300093 -2.20674658
[637] -2.78667726 -4.85606965 -3.58856419 -2.76310136 -3.56528113 -3.08850178
[643] -3.29539910 -4.36817878 -5.11396286 -1.11101095 -2.63998333 -2.40884353
[649] -1.72671474 -3.49027810 -3.03347365 -3.59703670 -3.58944734 -3.01953130
[655] -3.83007271 -1.89454161 -2.11227975 -2.57299582 -3.68882275 -2.66077268
[661] -1.96299067 -2.33205530 -2.22206204 -2.31450009 -2.03907797 -3.46277311
[667] -4.40142591 -4.42650935 -4.15567349 -2.80471983 -1.88871370 -4.55094409
[673] -1.82751488 -2.42914675 -3.46288298 -2.09361698 -2.81345995 -2.22035785
[679] -1.76093172 -3.29005569 -1.61197816 -4.07721414 -3.47509500 -3.45485946
[685] -2.69506870 -2.02397935 -4.62196309 -3.80803708 -2.20262575 -1.61219921
[691] -3.00868455 -2.38352388 -2.82413492 -1.97585576 -2.11370630 -2.54573265
[697] -2.15556113 -0.57719128 -4.12708678 -1.37797433 -2.11762699 -3.44851989
[703] -2.75498073 -4.46084829 -2.15852334 -2.62744409 -2.13724517 -2.73958810
[709] -1.50178909 -2.93265295 -3.83888847 -2.74721326 -4.38259464 -3.10624503
[715] -3.58671557 -2.49844442 -3.38711834 -2.44326175 -3.44110475 -3.78462907
[721] -2.61164653 -4.97213256 -4.54531660 -2.85742147 -3.37394941 -3.34850679
[727] -4.15377589 -2.60573941 -3.28837234 -1.88934664 -3.79612412 -2.35155468
[733] -1.45270883 -2.91866669 -4.04384915 -4.02901887 -4.23294207 -4.09507372
[739] -2.35166971 -3.19785191 -2.60455105 -4.22131971 -3.14383627 -2.58101602
[745] -2.00893262 -1.70046884 -1.84098931 -5.51827560 -2.63440545 -3.37401610
[751] -4.20684946 -3.82125010 -3.81188339 -4.41384884 -2.90156736 -4.15689982
```

```
 [757] -2.89822996 -2.59512585 -2.86587433 -4.84336011 -3.68003760 -2.87864488
 [763] -0.92310939 -1.17430017 -2.52744591 -3.04580354 -1.12146481 -3.36410967
 [769] -2.22531124 -3.31633129 -3.93380253 -3.85613722 -3.08662653 -3.10598617
 [775] -3.09253526 -2.80963651 -3.82010926 -3.47777746 -3.23389261 -3.17371851
 [781] -3.87687412 -2.95674160 -3.70914063 -3.46839194 -2.88420865 -3.79075036
 [787] -4.12394379 -2.84501339 -3.41802065 -3.40723989 -1.53921313 -3.01967308
 [793] -2.67165333 -3.60362657 -2.78474740 -2.74758025 -0.86564305 -2.23315003
 [799] -3.66132301 -4.48888408 -2.43722161 -4.23146347 -3.48460112 -3.48249548
 [805] -3.29148933 -4.46524388 -2.05004298 -3.13533105 -1.28628765 -3.35701320
 [811] -2.26097458 -2.61863213 -2.77815668 -3.41243968 -1.43008208 -2.61593195
 [817] -3.68574835 -2.93736811 -4.16563124 -2.28826604 -3.55917107 -2.94747139
 [823] -3.65304729 -4.10345510 -4.63393622 -1.77943103 -2.53659773 -4.87189622
 [829] -4.27546918 -3.96478449 -2.22887871 -2.67303512 -3.36624746 -2.50528049
 [835] -3.68136344 -2.44828323 -3.41275361 -1.37150564 -4.07939721 -2.15118917
 [841] -4.93884712 -3.37030847 -3.79029637 -4.11605964 -3.45448102 -3.26311170
 [847] -2.77495368 -1.93343903 -2.75959642 -3.43167591 -1.54208277 -2.60059735
 [853] -2.43127902 -2.40969982 -3.48715781 -2.94280639 -2.88824568 -2.49813287
 [859] -2.04815567 -3.51515620 -2.19201621 -3.04400427 -3.48732236 -3.52989818
 [865] -2.56261602 -2.13110008 -1.95648987 -1.14262150 -2.43102182 -2.19668649
 [871] -3.52515358 -1.45022633 -2.25134710 -2.23711019 -3.97939170 -1.36770519
 [877] -1.23966960 -0.87720928 -4.10990986 -3.13887670 -3.42953657 -3.12887120
 [883] -2.93171946 -3.34346472 -2.71488771 -3.74772863 -3.38637592 -3.11586848
 [889] -0.67944410 -3.34725785 -4.02593899 -1.92312023 -3.05181244 -2.66779160
 [895] -2.28999500 -1.26317291 -5.58461826 -2.70383721 -2.25817219 -2.76095871
 [901] -3.68758610 -2.43682287 -3.38547122 -2.68759235 -4.85358175 -2.08097178
 [907] -3.91212997 -2.35330601 -3.18168119 -1.60143680 -2.01605060 -2.83477561
 [913] -3.23949184 -5.65170003 -1.78287284 -0.88982870 -3.00481986 -1.35358024
 [919] -1.54401151 -2.95666974 -1.47704147 -2.52498789 -3.33034815 -3.31828117
 [925] -1.39054201 -3.96404567 -4.46605982 -1.93391410 -2.05636497 -1.67774496
 [931] -2.22190600 -1.27894295 -2.98383899 -0.20846611 -0.80390309 -4.44925738
 [937] -2.95550195 -3.12439473 -5.39490720 -4.19967401 -4.22016742 -3.38712003
 [943] -3.95394694 -1.66974704 -3.91662693 -2.09468869 -1.54160568 -4.58889582
 [949] -2.36622589 -3.67880443 -3.81171910 -2.21803629 -4.55156723 -4.97069359
 [955] -2.59771003 -2.84502817 -2.85560004 -1.48729229 -2.96268424 -4.35802304
 [961] -2.75920398 -3.83687517 -2.24624800 -2.81311381 -3.03898822 -2.46132606
 [967] -2.28806861 -2.73677665 -3.28588395 -4.31679017 -3.67595388 -3.86120493
 [973] -2.88355406 -5.04316428 -3.11173757 -3.26934897 -2.03625308 -3.91654220
 [979] -2.75880346 -3.54763623 -2.40932330 -4.51957010 -2.58508258 -3.31344305
 [985] -2.65607240 -4.12843593 -3.86616730 -4.12505556 -1.48551044 -3.25899493
 [991] -5.13443150 -2.02556766 -3.85316531 -2.64134144 -1.12425764 -3.05636555
 [997] -3.76555240 -2.38627890 -3.22358784 -3.17125817  4.97096909  5.07501372
[1003]  4.12882688  2.80447051  1.64157388  3.05078958  3.83234679  2.73259302
[1009]  2.00370531  2.27875562  4.77719241  2.93528613  3.48364767  4.28654614
```

```
[1015]  2.81617570  3.36354703  3.86584558  3.16461199  3.77251982  3.55113853
[1021]  2.64904226  2.50738238  4.53386440  3.55445908  3.33364090  2.62542404
[1027]  2.23309637  2.87522779  3.06633467  2.16220413 -0.06694611  1.79352195
[1033]  2.60166470  2.36068054  2.96044379  2.08062163  1.79889262  2.08659603
[1039]  2.95291269  3.09869974  3.42087508  3.37292780  4.28034966  2.77559989
[1045]  3.00896862  3.53767420  3.70572305  3.07954114  2.36137015  2.94127580
[1051]  3.80155563  3.87561288  3.02061956  2.88233562  3.80379430  2.20118055
[1057]  2.59434527  3.30095149  4.12984421  2.62836206  1.53716471  2.25296219
[1063]  2.32923823  2.44943229  2.58891220  3.43456794  2.93645566  2.33445266
[1069]  2.39460526  3.49759306  1.11018601  2.74781573  3.57436973  2.27856912
[1075]  3.19206174  1.64776395  1.84908714  3.85506762  3.28805590  2.56478117
[1081]  2.70969272  3.53524908  0.82733021  4.58216323  2.47073633  3.50168259
[1087]  3.60154525  1.01651278  1.68080781  1.89853901  5.12957515  3.36445375
[1093]  2.06748144  2.97202449  2.80235583  2.70606184  2.72854970  3.34052708
[1099]  3.22276198  3.78941097  3.39405060  2.39965782  2.52406500  3.67846177
[1105]  1.53603186  1.51362316  2.59331343  4.08251903  3.87811634  3.11360124
[1111]  2.25711841  2.37931688  2.07792937  1.70482876  4.56233249  4.65177768
[1117]  4.29213534  2.76526874  3.81657450  2.32182798  3.31208780  4.76898054
[1123]  4.84522594  2.55443374  2.91088548  3.21933887  2.07302318  1.23223726
[1129]  2.56006415  3.84163202  3.42794339  2.65178984  3.71774432  2.14993194
[1135]  2.17086458  1.86890643  3.99067569  3.26570222  3.63141635  2.37108922
[1141]  1.32668308  3.41568463  2.13768980  3.01247053  0.73158350  2.77318386
[1147]  3.59463634  3.36251143  4.27074484  4.14018673  0.77460390  5.66718238
[1153]  2.01319855  2.60855031  1.61107093 -0.07974076  3.40881722  2.90078443
[1159]  3.76977924  3.08456385  3.90747838  3.63293544  2.65397277  4.35650580
[1165]  3.37227160  4.86886417  2.64277006  3.99845156  2.50770516  3.82486360
[1171]  2.69198819  1.84384613  5.12355657  2.87990515  2.66036773  3.97868628
[1177]  2.92363612  2.48653484  3.81566547  5.81138902  3.12423035  3.92457562
[1183]  1.63673239  4.38639402  2.45069361  2.61332289  3.70131350  1.95843853
[1189]  4.36719942  4.83408200  3.15606363  4.26083465  2.49103270  4.57796326
[1195]  3.60455580  4.35542642  1.92972717  2.65797528  2.75197015  2.87611529
[1201]  3.68628975  2.93527127  0.44026695  3.68481444  3.37581870  3.57038105
[1207]  2.63741253  2.44033921  2.75739305  3.58799004  3.64952455  3.25965871
[1213]  3.21633834  2.51121772  4.18762765  1.98283555  2.38036837  1.98984512
[1219]  1.32710364  3.36108611  3.27448283  2.92450964  1.19096105  2.39906111
[1225]  5.37796108  2.74765862  3.31677090  3.77960067  3.13893227  3.57328865
[1231]  3.88485120  4.21674949  3.74670042  3.00144214  3.03120549  3.01310703
[1237]  3.42804142  4.47481103  2.91935992  2.63048229  4.85658610  1.13365808
[1243]  2.20739262  2.98644738  2.38952951  4.70289946  3.84425856  5.31075384
[1249]  3.11977316  3.64058836  4.02360610  1.69778578  2.92668279  2.70658054
[1255]  2.72010501  3.94826780  3.28761504  3.60267233  3.31619502  2.40809769
[1261]  2.82718567  3.66973924  0.85614894  4.16897865  2.73748821  2.47845532
[1267]  3.23361351  1.37590817  2.33045867  3.60912643  1.78777694  3.21559062
```

```
[1273]  3.39987160  2.05365244  4.12838727  3.45462211  2.17706709  2.59039620
[1279]  3.30293061  2.82807875  4.16934081  3.42870179  2.33793738  1.14081698
[1285]  3.55369213  2.92549605  4.88451757  3.41077895  2.52923105  1.84897994
[1291]  1.15658807  4.64363512  3.33749405  4.89371422  2.69913930  2.45936559
[1297]  2.53664168  3.82153519  1.29231879  3.44780631  1.98277061  2.98491013
[1303]  3.01033031  2.04445402  5.02357487  1.00080881  2.33808335  3.93540027
[1309]  2.80485707  2.49354995  3.38036164  2.91673576  1.89326195  2.51460945
[1315]  2.27919612  1.54723116  2.85174475  3.15218526  1.71650722  4.45291106
[1321]  3.07929604  0.91109615  3.78501851  3.89424395  2.41347893  3.06467253
[1327]  2.11254602  0.94660855  1.71742607  3.80110136  2.79195761  3.05187480
[1333]  4.17441981  3.29364249  3.65783833  3.60727548  3.15372891  2.20270962
[1339]  3.25863001  4.81504471  4.01454701  0.91031308  4.25277290  4.01114620
[1345]  1.56595133  1.89234883  3.03795265  3.92219323  2.84129627  2.43620066
[1351]  5.10640635  1.46224865  2.60857193  2.46185368  3.25759041  2.40681496
[1357]  4.26437896  3.99423802  2.19023315  3.05983678  1.32369539  2.91969555
[1363]  1.95407345  1.76765632  2.53046775  3.02643430  3.32867326  3.38660812
[1369]  3.23399329  2.22789960  2.86769618  1.61868341  2.17451229  3.35135593
[1375]  3.44718229  3.11583867  3.70578946  1.98513052  2.39145348  1.84164719
[1381]  2.48696294  3.94838661  2.51460583  4.10062190  3.82161785  2.90934794
[1387]  3.20605075  1.21720089  4.32826440  3.58992595  2.95693966  1.91379183
[1393]  2.81960114  2.83648143  2.22875023  3.23247012  2.36590486  4.22905931
[1399]  2.62546318  1.54913035  4.11290923  2.72431959  3.73180115  3.82904425
[1405]  2.73090606  4.55217370  3.67660509  3.69589396  2.18411926  3.63882005
[1411]  2.00268256  4.49789833  2.93467602  4.43487291  4.23340467  4.56432380
[1417]  2.26336864  2.66688372  3.07983824  3.52542056  2.35729715  3.95806208
[1423]  3.45986318  3.54599542  1.84913753  3.06888298  3.84084873  2.83515711
[1429]  1.64898111  4.09518503  4.11972007  1.47402484  3.74356935  3.35563155
[1435]  2.43396739  3.22067838  1.97517306  2.22431026  1.61799422  2.39138974
[1441]  1.73812497  3.76177090  2.16475017  2.67607068  2.46153869  2.04115298
[1447]  4.47175285  4.66552787  3.65003815  3.25567492  2.58102733  4.49638089
[1453]  3.59912884  4.57523258  1.86051545  4.04328475  3.64689340  0.50508084
[1459]  4.76880209  2.57186957  4.90161278  2.84583781  3.47282290  2.38799543
[1465]  3.34015249  0.75743840  2.14668140  3.66676212  2.39419384  2.35313381
[1471]  4.33254299  3.46728224  2.27366605  3.99309426  3.85576412  3.00963218
[1477]  3.06573576  3.29898958  2.92564231  3.27152659  2.34279382  3.05279480
[1483]  3.83745296  2.54201765  1.20692778  3.89405853  2.37455560  4.59066185
[1489]  4.03474725  3.41077507  1.56937295  2.71096305  3.12665888  3.30667339
[1495]  4.45613604  3.53915125  4.56391525  2.55083155  1.53491217  1.58478525
[1501]  3.30927250  4.99638694  2.96435593  2.97662473  2.68204555  2.91934227
[1507]  2.53161850  3.71004567  3.05241926  2.07349298  3.29995655  3.45874435
[1513]  3.58980671  2.59438778  3.32359150  4.57085157  3.53304696  3.43826296
[1519]  1.28475268  2.98347479  4.47031116  1.77644805  0.31730031  1.63150810
[1525]  3.94279331  1.56641949  2.20313092  1.71282564  1.70705286  2.68255497
```

```
[1531]  1.08755204  1.89091903  4.48698187  5.40996892  2.63575568  0.63608717
[1537]  2.04184895  2.18297938  2.61636441  3.69849803  3.13911810  3.41113958
[1543]  2.87358399  3.13679761  2.89158682  3.13261292  3.45456968  4.40517798
[1549]  1.34650168  4.05000952  2.19482339  3.34153173  3.49889545  2.88995946
[1555]  4.49536676  1.37735507  4.13927475  2.38931465  3.35794858  3.53511832
[1561]  3.53443436  2.00468300  3.17033657  4.90059909  3.20840839  3.74685679
[1567]  3.75691078  3.62834362  2.58836584  1.18738581  2.35205073  4.02116873
[1573]  1.74384281  1.89131478  2.91827883  2.46075444  1.86686983  2.68628737
[1579]  3.17950382  4.22936900  3.91827112  1.99686992  3.00642625  5.27120906
[1585]  3.04027295  3.17631486  2.69395529  1.42259936  2.16423673  1.91554819
[1591]  1.74368552  4.19139701  2.58247004  4.09468269  2.04979482  1.90140988
[1597]  3.17503837  0.64357107  2.04067340  2.98836884  4.66580569  4.63391768
[1603]  2.16199153  4.43516565  2.79168474  3.29645408  2.71543721  4.33927488
[1609]  3.78624452  2.89440028  3.35394593  3.85043733  1.68563692  1.65660327
[1615]  3.10822481  3.13236853  2.14388851  2.33308021  3.64564617  2.16617319
[1621]  2.98196223  4.52496898  1.81663989  2.42171724  1.89164935  3.38113025
[1627]  3.90477186  4.80027938  2.43144679  2.61811877  2.08691228  5.02023309
[1633]  4.34870323  3.10229810  1.78971952  3.01281538  1.49560106  3.02580916
[1639]  4.78492320  5.27484819  2.64815313  3.52690283  1.78601438  3.04967864
[1645]  3.55368609  2.10003777  2.10744004  3.86154337  3.53732073  1.96867219
[1651]  2.49290155  3.06881999  2.82434268  3.31674534  4.18460022  1.95847743
[1657]  3.46472868  2.49806136  2.33609263  2.09352164  3.08327576  4.01959201
[1663]  1.88349812  3.44085527  2.63447978  3.34243700  3.80648659  3.92627094
[1669]  4.15231023  1.85740306  2.05068191  1.51015576  2.53961810  4.22058939
[1675]  2.27780210  3.44359989  2.04257162  2.01793008  0.38858189  3.19110178
[1681]  4.56699535  3.16787511  1.52926927  3.93589629  4.53335414  2.62630194
[1687]  3.33412742  3.20795772  3.11507971  3.38396844  2.40615138  2.19828440
[1693]  2.70981305  4.16186318  4.64940679  2.72517212  2.96451246  4.00459387
[1699]  3.80541186  3.40465136  2.87347630  3.05629870  2.89195750  2.05806373
[1705]  4.87703892  0.69291509  0.53676369  2.39655844  3.06709642  3.46945680
[1711]  3.43671480  5.61920762  3.21351896  3.80170903  3.56617247  5.22206278
[1717]  2.80359898  1.81376367  4.58727704  3.58729141  2.38061215  3.52194992
[1723]  3.90915638  1.60081288  3.59670366  2.09204496 -0.16319030  2.83276042
[1729]  2.92791729  3.24690723  3.51816175  3.95392943  1.20073946  2.97890749
[1735]  2.68592819  4.40740926  1.75601120  3.02364543  3.01522753  1.79729947
[1741]  3.28575631  2.42092919  0.68849651  2.02437772  3.50073860  1.93635478
[1747]  4.05594186  2.51281408  2.61030913  1.72861124  2.67229822  2.47227790
[1753]  3.17990535  1.97645335  4.41696142  5.27809760  3.05149335  0.78787214
[1759]  3.98662152  2.94856145  4.76241503  2.84492770  3.80825798  1.76532889
[1765]  2.28160752  3.38797767  2.98727639  4.16204110  2.91421054  3.53207976
[1771]  2.25582069  4.45123128  2.81466439  3.33866145  3.03582649  1.26801527
[1777]  2.35867675  3.99347340  1.92009059  2.71921014  0.43316633  2.18353915
[1783]  3.71471658  3.04105897  2.61508405  3.54401315  3.70710010  3.98244464
```

```
[1789]  4.42685899  3.23178117  3.54845046  2.88527318  2.30098878  3.46700481
[1795]  2.26416272  4.15795074  1.62063074  2.65408476  3.97692620  4.31510583
[1801]  4.36343775  3.92995354  3.88387201  2.66021036  2.59605023  3.57587202
[1807]  2.59417663  2.95282667  2.78584754  2.75567560  4.33471001  3.36018010
[1813]  3.77289592  4.15393540  0.92637353  3.76309415  4.74642966  5.43393130
[1819]  2.09441030  4.16984464  4.26799202  3.68242290  1.88342237  2.69427983
[1825]  2.35759735  2.06282619  3.07324794  3.53071703  3.46688048  2.58026854
[1831]  2.76005231  2.70479675  1.78892568  3.34451679  2.47223945  3.47171616
[1837]  3.73520759  4.12565301  2.46516492  3.46005379  2.41024881  3.84003419
[1843]  2.93558740  2.17198231  2.52233701  2.50168435  2.80125677  1.94260588
[1849]  2.50164205  1.67283991  3.16019418  4.77588944  3.39501893  1.75279178
[1855]  3.39807922  4.05018830  3.45366441  2.69048543  1.54904902  3.60323224
[1861]  2.63983669  4.09302330  3.01352571  3.02239445  4.11727064  2.48168506
[1867]  2.54904010  1.04581627  3.60312341  1.37349182  0.03790787  4.18225492
[1873]  3.81712406  1.31565404  2.73520304  4.90324271  2.40537420  2.41042016
[1879]  3.30430844  3.45345481  3.52451531  2.48475931  5.35682670  4.85634443
[1885]  3.18514504  5.34353729  3.48576780  4.35762512  4.39455937  4.42567868
[1891]  2.21234840  2.76032996  1.67186412  3.62120259  2.66660081  3.43430536
[1897]  4.82651033  2.92485188  3.51238775  3.35659634  1.96331324  2.06556382
[1903]  2.96730994  2.73660347  4.67994851  2.47444548  3.51859400  1.91286082
[1909]  1.62890461  2.41404268  4.19400923  3.03929682  3.25305096  2.55344246
[1915]  2.60960895  1.75243259  3.91643811  2.88978537  2.01032980  2.25554813
[1921]  4.16893629  1.73408601  4.15491994  4.69627057  2.56824006  4.18160128
[1927]  2.44886417  2.89096554  3.14656854  4.43830526  4.29828528  5.12745121
[1933]  2.79822778  3.93074584  1.81421495  2.32618494  3.83381566  2.12874144
[1939]  3.89300199  3.44919096  2.29805689  1.42714923  2.73549080  1.49374188
[1945]  3.18062812  2.73704467  3.35022243  1.65505154  4.23965236  3.89970540
[1951]  3.67842006  2.78710049  5.05180422  3.85077894  2.26548551  0.51085350
[1957]  2.30353990  3.87319432  2.52811868  2.02767507  4.14630968  3.35082805
[1963]  2.38213784  2.75652322  3.14442929  3.68062581  3.58883089  3.26278188
[1969]  3.24319494  3.28657048  2.74338866  3.96890758  2.33076900  2.07198397
[1975]  3.94349321  2.39559741  4.87921725  1.54806680  2.37326587  2.32966687
[1981]  2.36858804  2.65313725  4.03852614  1.39693980  4.74492360  2.19320097
[1987]  4.49852538  2.14867048  1.37822757  2.61218492  4.55608511  3.70314706
[1993]  0.90399010  4.31141762  3.78824947  3.29779886  5.13347779  2.67663300
[1999]  3.11401842  3.18433915
```

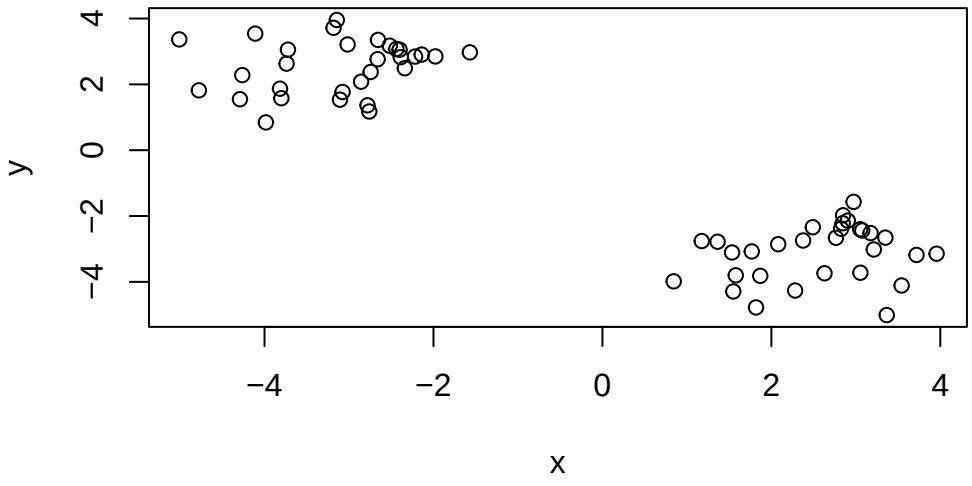
```r
n = 30
x <- c(rnorm(n, mean = -3), rnorm(n, mean = 3))
y <- rev(x)

z <- cbind(x, y)
```

```
z
```

```
               x          y
 [1,] -2.2178387  2.8443467
 [2,] -3.0163480  3.2126141
 [3,] -4.7768598  1.8186997
 [4,] -2.7599391  1.1757071
 [5,] -2.3868128  2.8259324
 [6,] -1.5685791  2.9729457
 [7,] -3.9837980  0.8438542
 [8,] -2.3389878  2.4905582
 [9,] -3.7388772  2.6289534
[10,] -2.7804233  1.3634899
[11,] -3.1442403  3.9563275
[12,] -3.1824550  3.7181997
[13,] -5.0089677  3.3657001
[14,] -3.8009958  1.5787854
[15,] -2.6610994  2.7643014
[16,] -3.7224432  3.0541677
[17,] -2.6564048  3.3497524
[18,] -3.1066609  1.5348625
[19,] -2.8565841  2.0810203
[20,] -2.1389825  2.9043855
[21,] -2.5173890  3.1749413
[22,] -4.1102054  3.5417678
[23,] -3.8167385  1.8684096
[24,] -2.4387514  3.0758480
[25,] -2.7437248  2.3755614
[26,] -4.2632077  2.2812497
[27,] -3.0759826  1.7681601
[28,] -2.4008080  3.0518112
[29,] -4.2908103  1.5495166
[30,] -1.9775986  2.8490604
[31,]  2.8490604 -1.9775986
[32,]  1.5495166 -4.2908103
[33,]  3.0518112 -2.4008080
[34,]  1.7681601 -3.0759826
[35,]  2.2812497 -4.2632077
[36,]  2.3755614 -2.7437248
[37,]  3.0758480 -2.4387514
[38,]  1.8684096 -3.8167385
[39,]  3.5417678 -4.1102054
```

```
[40,]  3.1749413 -2.5173890
[41,]  2.9043855 -2.1389825
[42,]  2.0810203 -2.8565841
[43,]  1.5348625 -3.1066609
[44,]  3.3497524 -2.6564048
[45,]  3.0541677 -3.7224432
[46,]  2.7643014 -2.6610994
[47,]  1.5787854 -3.8009958
[48,]  3.3657001 -5.0089677
[49,]  3.7181997 -3.1824550
[50,]  3.9563275 -3.1442403
[51,]  1.3634899 -2.7804233
[52,]  2.6289534 -3.7388772
[53,]  2.4905582 -2.3389878
[54,]  0.8438542 -3.9837980
[55,]  2.9729457 -1.5685791
[56,]  2.8259324 -2.3868128
[57,]  1.1757071 -2.7599391
[58,]  1.8186997 -4.7768598
[59,]  3.2126141 -3.0163480
[60,]  2.8443467 -2.2178387
```

```
plot(z)
```



K-means clustering

The function in base R for k-means clustering is called `kmeans()`.

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

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

Cluster means:
          x         y
1  2.534031 -3.116084
2 -3.116084  2.534031

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] 40.25479 40.25479
 (between_SS / total_SS =  92.2 %)

Available components:

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

```
km$centers
```

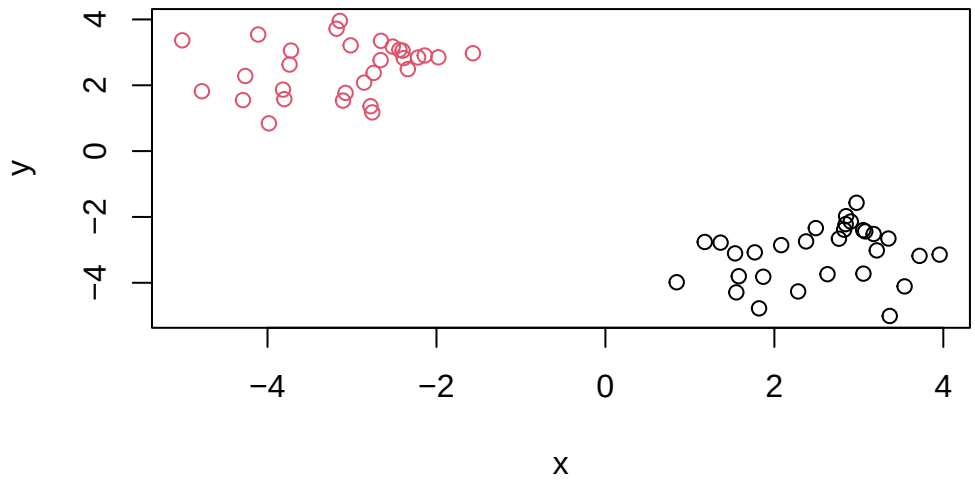
```
          x         y
1  2.534031 -3.116084
2 -3.116084  2.534031
```

Q. Print out the membership factor (i.e. our main answer)

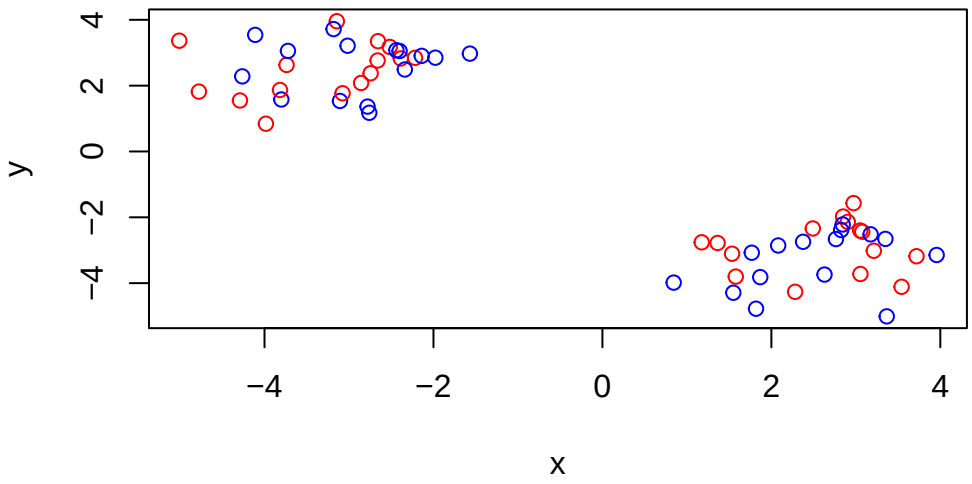
```
km$cluster
```

```
 [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
```

```r
cl <- kmeans(z, centers = 2)
plot(z, col = cl$cluster)
```

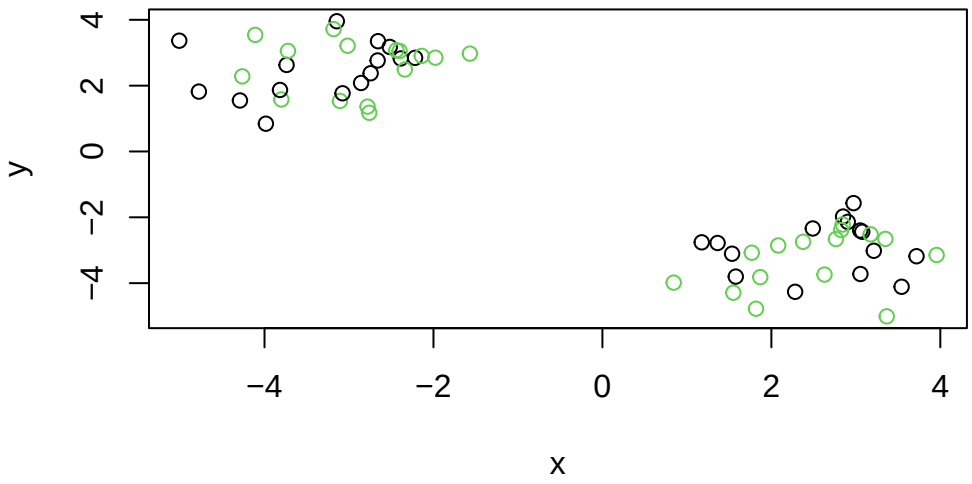



```r
plot(z, col = c("red", "blue"))
```



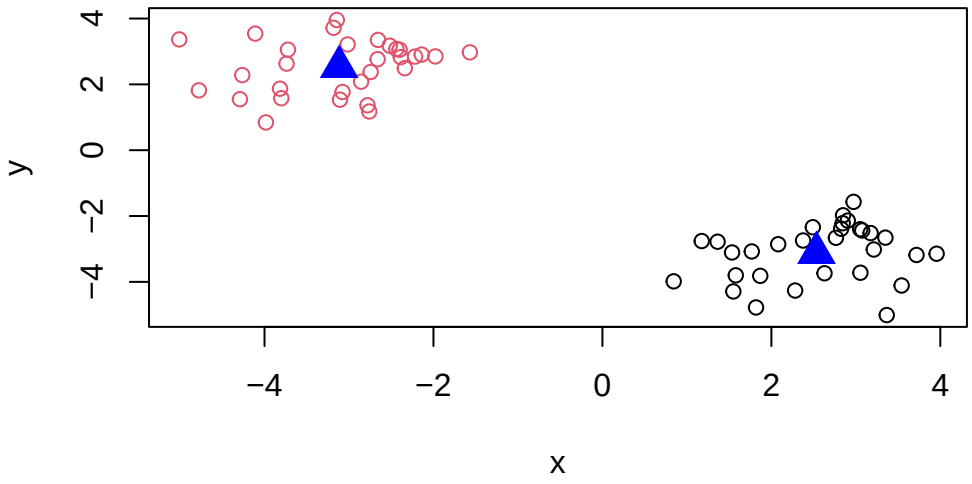

```
#points(cl$centers, col = 1:5, pch = 8)
```

```
#color by number
plot(z, col = c(1,3))
```



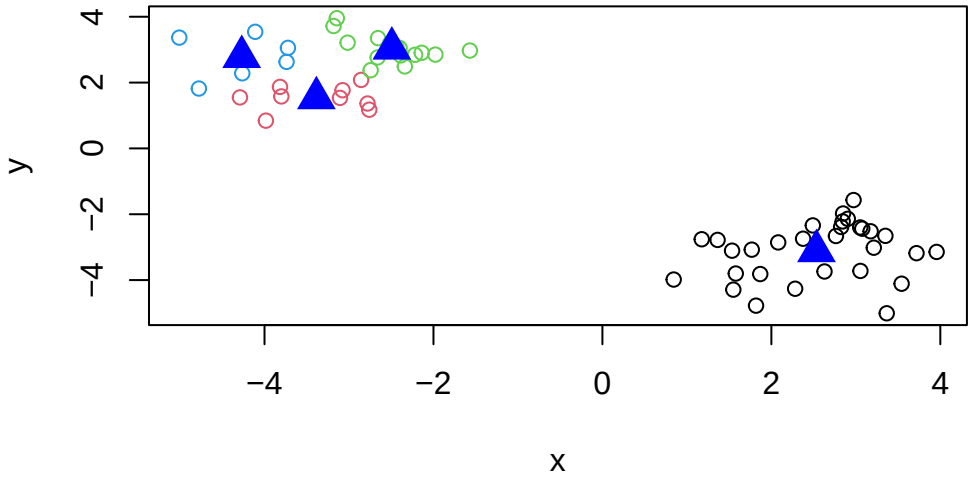
plot with clustering result

```
plot(z, col = cl$cluster)
points(cl$centers, col = "blue", pch = 17, cex = 2)
```



Q. Can you cluster our data i `z` into four clusters

```
cl <- kmeans(z, centers = 4)
plot(z, col = cl$cluster)
points(cl$centers, col = "blue", pch = 17, cex = 2)
```



##Hierarchical Clustering

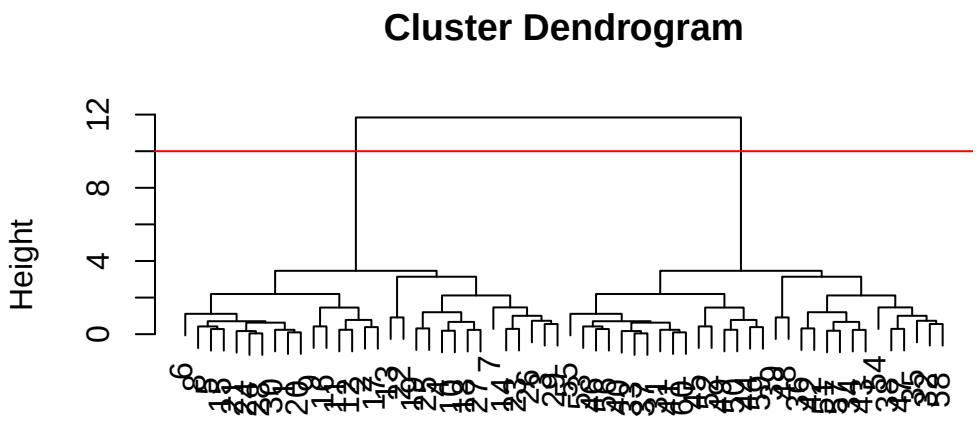
The main funtion for hierarchical clustering in base R is called `hclust()`

Unlike K-means (`kmeans`) I can not just pass mu data as input, first I need a distance matirx from my data.

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

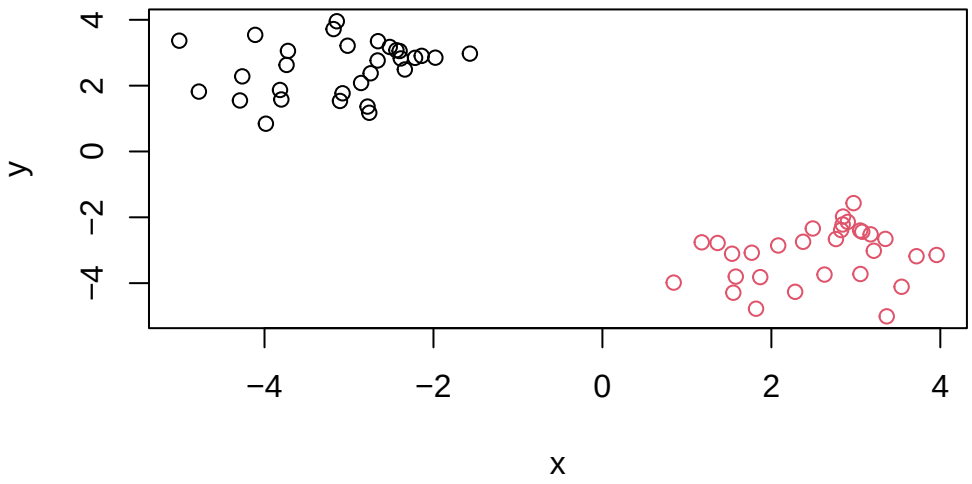
```
Call:
hclust(d = d)

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

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



TO get my clustering results (i.e. the membership vector), I can "cut" my tree/dendrogram at a given eight. To do this I will use the `cutree()`

```
grps <- cutree(hc, h =10)
plot(z, col = grps)
```



Principle Component Analysis

Principal component analysis (PCA) is a well established "multivariate statistical technique" used to reduce the dimensionality of a complex data set to a more manageable number (typically 2D or 3D). This method is particularly useful for highlighting strong paterns and relationships in large datasets (i.e. revealing major similarities and diferences) that are otherwise hard to visualize. As we will see again and again in this course PCA is often used to make all sorts of bioinformatics data easy to explore and visualize.

PCA of UK food data

```
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?
dim(x)
```

```
[1] 17  5
```

```
#gives number of rows & columns
```

```
## Preview the first 6 rows
head(x, 6)
```

```
              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
```

```
# Note how the minus indexing works
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
```

```
dim(x)
```

```
[1] 17  4
```

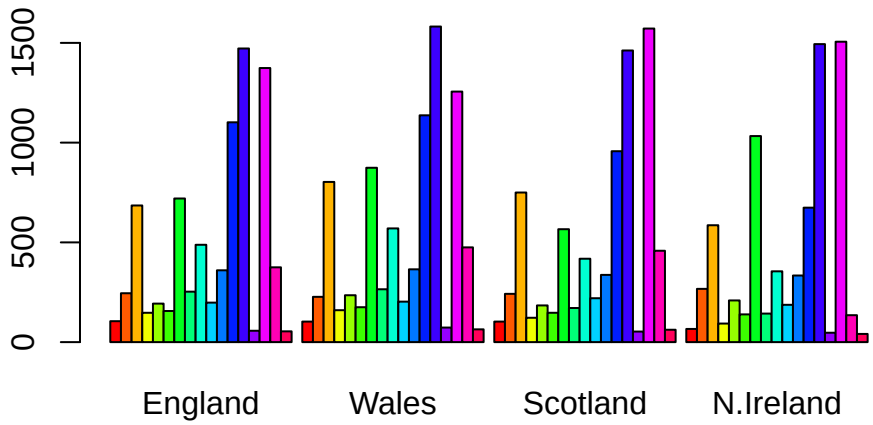
```
x <- read.csv(url, row.names=1)
head(x)
```

```
              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
```

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?

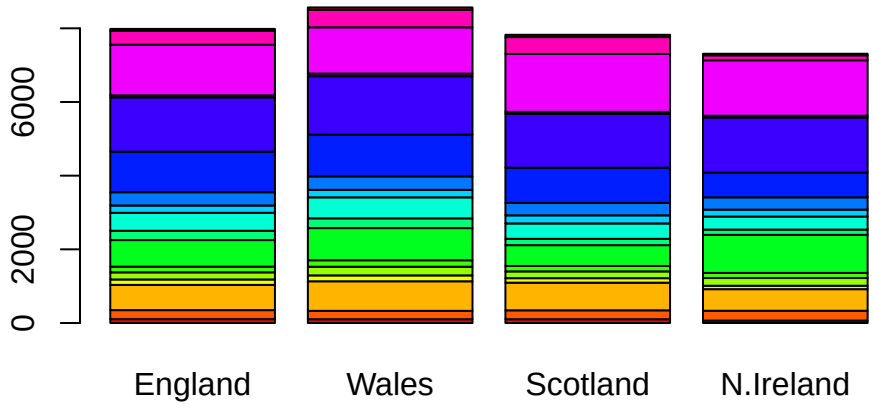
I prefer 'x <- read.csv(url, row.names=1)' as the other methods keeps on overwritting the rownames as the next column on the left and we loose the data.

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



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

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

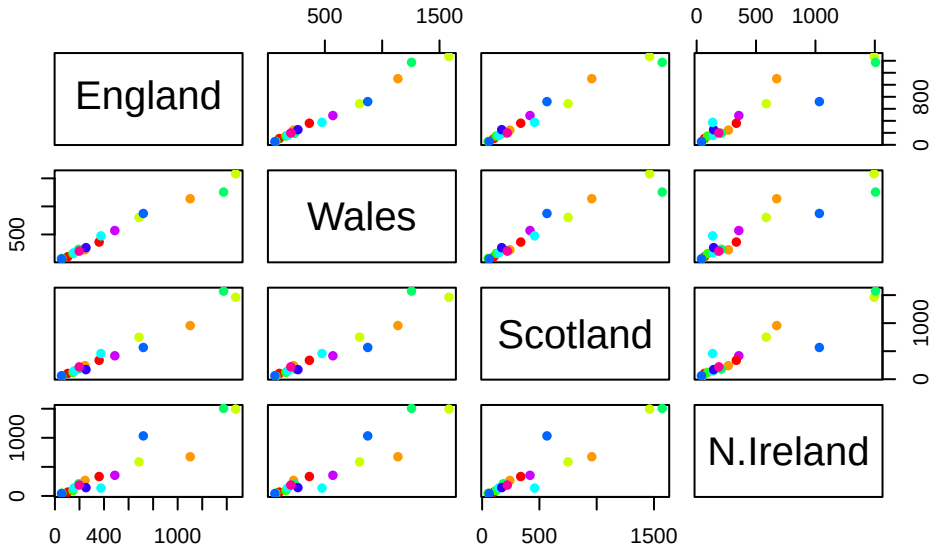


Q5: Generating all pairwise plots may help somewhat. Can you make sense of the following code and resulting figure? What does it mean if a given point lies on the diagonal for a given plot?

The plot gives the scatter plot between all pairs of country for all rows in the data.

if a given point lies on the diagonal for a given plot then that particular food is consumed in equal quantity in both the countries of that corresponding plot.

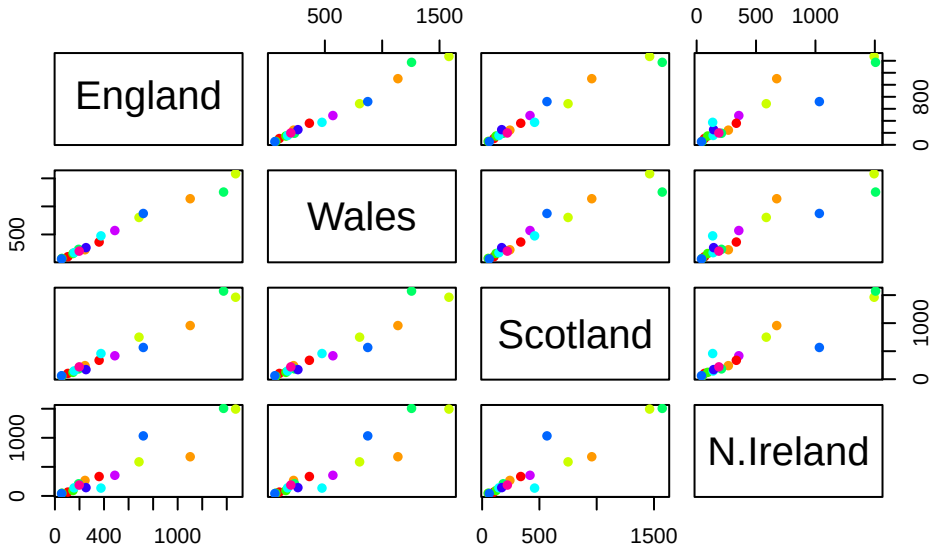
```
pairs(x, col=rainbow(10), pch=16)
```



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

Looking at this plot we can say Northern Ireland is somewhat different from rest of the countries based on the consumption of foods in the data provided

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




```
#The plot here gives matrix kind of plots where each plot is a scatter plot
```

PCA to the rescue

The main function to do PCA in base R is called `prcomp`

Note that I need to take the transpose of this particular data as that is what `prcomp()` help page was asking for

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

Let's see what is inside our result object `pca` that we just calculated:

```
attributes(pca)
```

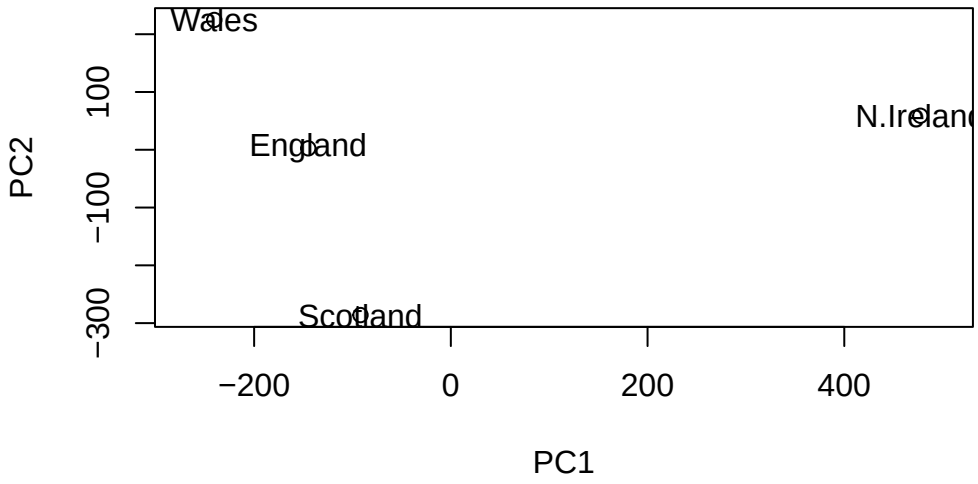
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
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
```

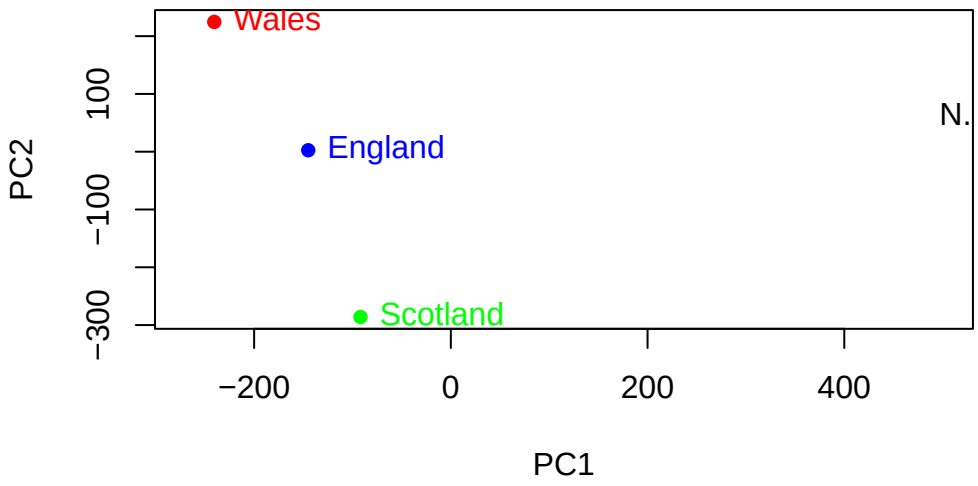
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
# 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=rownames(pca$x))
```



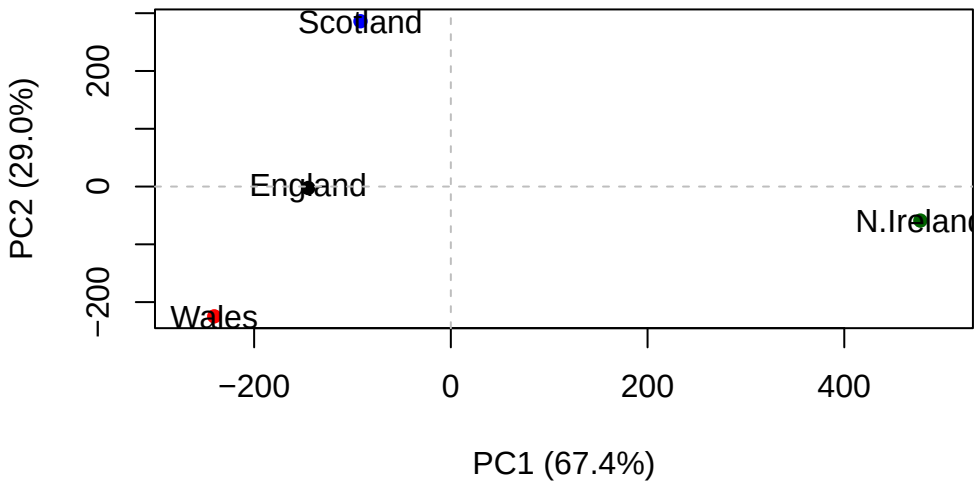
Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
# Define country names and their corresponding colors
country_names <- colnames(x)
country_colors <- c("England" = "blue", "Scotland" = "green", "Wales" = "red", "N. Ireland" =

# Plot PC1 vs PC2 with customized colors for country names
plot(pca$x[,1], -pca$x[,2], xlab = "PC1", ylab = "PC2", xlim = c(-270, 500), col = country_co

# Add country names as labels with their corresponding colors
text(pca$x[,1], -pca$x[,2], labels = country_names, col = country_colors[country_names], pos
```



To make our main result figure, called a "PC plot" (or "score plot", "ordination plot", or "PC1 vs PC2 plot")

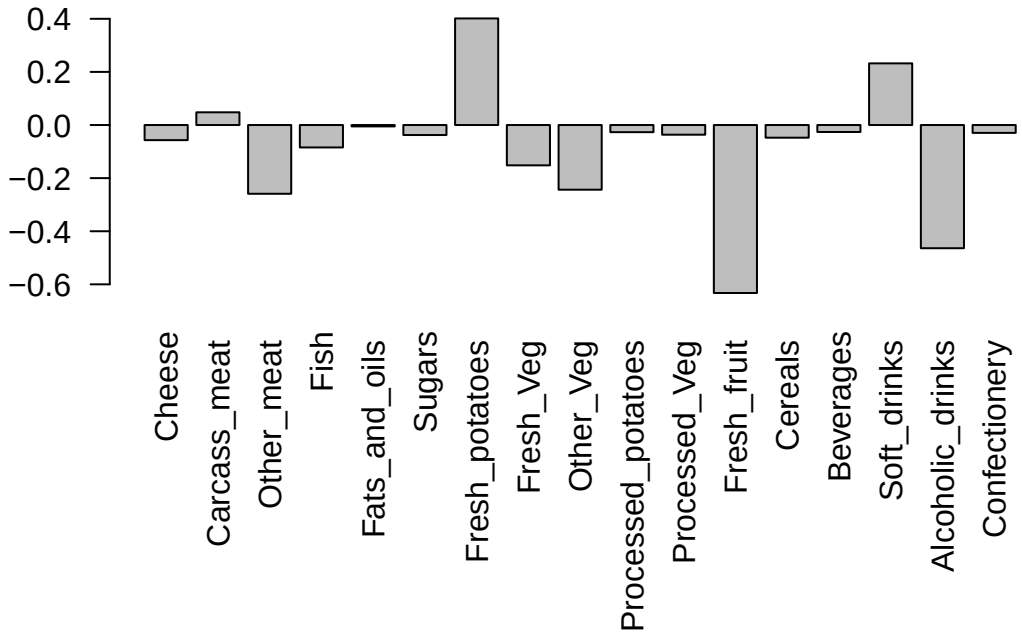
```
plot(pca$x[,1], pca$x[,2], col = c("black", "red", "blue", "darkgreen"), pch = 16, xlab="PC1
text(pca$x[,1], pca$x[,2], colnames(x))
abline(h = 0, col="gray", lty = 2)
abline(v = 0, col="gray", lty = 2)
```



Variable loadings plot

can give us insights on how original varibales in

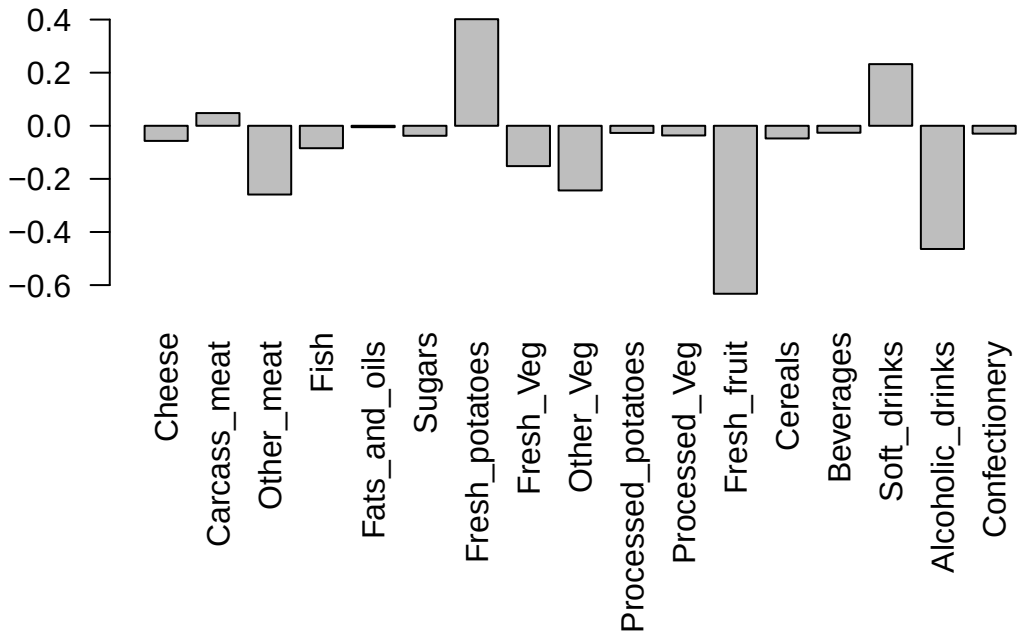
```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```




```
pca$rotation
```

	PC1	PC2	PC3	PC4
Cheese	-0.056955380	0.016012850	0.02394295	-0.694538519
Carcass_meat	0.047927628	0.013915823	0.06367111	0.489884628
Other_meat	-0.258916658	-0.015331138	-0.55384854	0.279023718
Fish	-0.084414983	-0.050754947	0.03906481	-0.008483145
Fats_and_oils	-0.005193623	-0.095388656	-0.12522257	0.076097502
Sugars	-0.037620983	-0.043021699	-0.03605745	0.034101334
Fresh_potatoes	0.401402060	-0.715017078	-0.20668248	-0.090972715
Fresh_Veg	-0.151849942	-0.144900268	0.21382237	-0.039901917
Other_Veg	-0.243593729	-0.225450923	-0.05332841	0.016719075
Processed_potatoes	-0.026886233	0.042850761	-0.07364902	0.030125166
Processed_Veg	-0.036488269	-0.045451802	0.05289191	-0.013969507
Fresh_fruit	-0.632640898	-0.177740743	0.40012865	0.184072217
Cereals	-0.047702858	-0.212599678	-0.35884921	0.191926714
Beverages	-0.026187756	-0.030560542	-0.04135860	0.004831876
Soft_drinks	0.232244140	0.555124311	-0.16942648	0.103508492
Alcoholic_drinks	-0.463968168	0.113536523	-0.49858320	-0.316290619
Confectionery	-0.029650201	0.005949921	-0.05232164	0.001847469

```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



Below we can use the square of pca$sdev , which stands for "standard deviation", to calculate how much variation in the original data each PC accounts for.

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

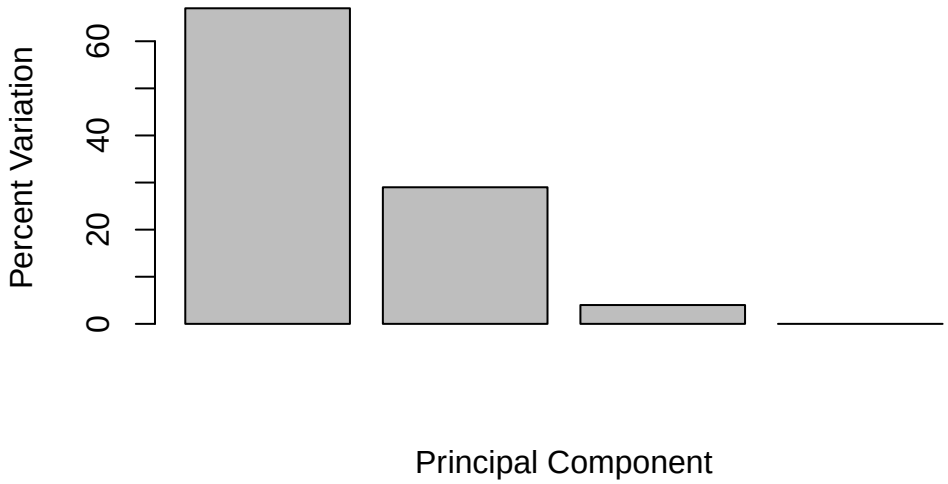
```
[1] 67 29  4  0
```

```
## or the second row here...
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3           PC4
Standard deviation     324.15019 212.74780 73.87622 3.175833e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

This information can be summarized in a plot of the variances (eigenvalues) with respect to the principal component number (eigenvector number), which is given below.

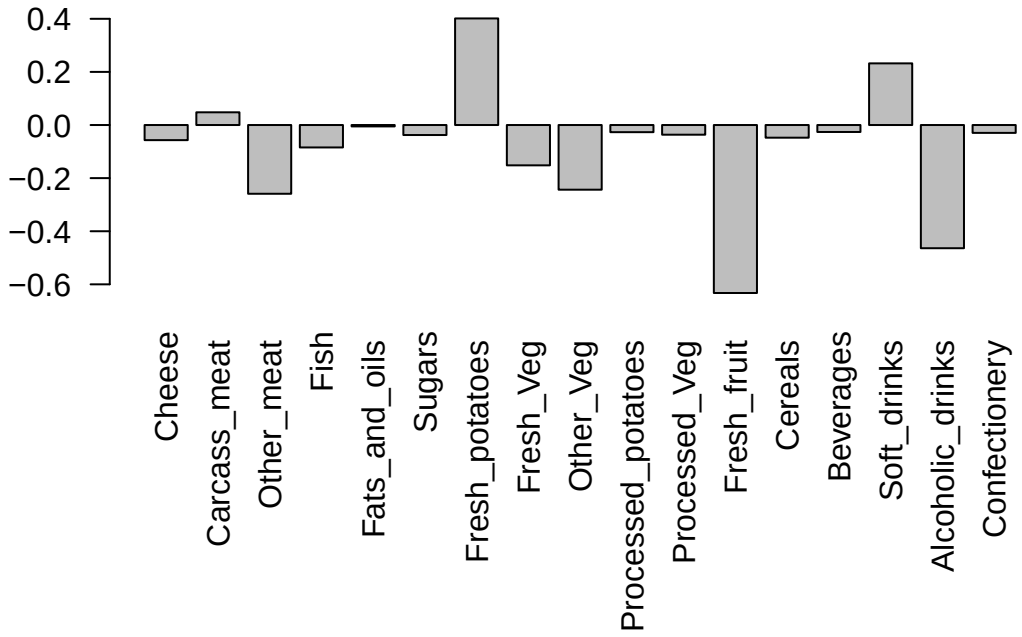
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



Digging deeper (variable loadings)

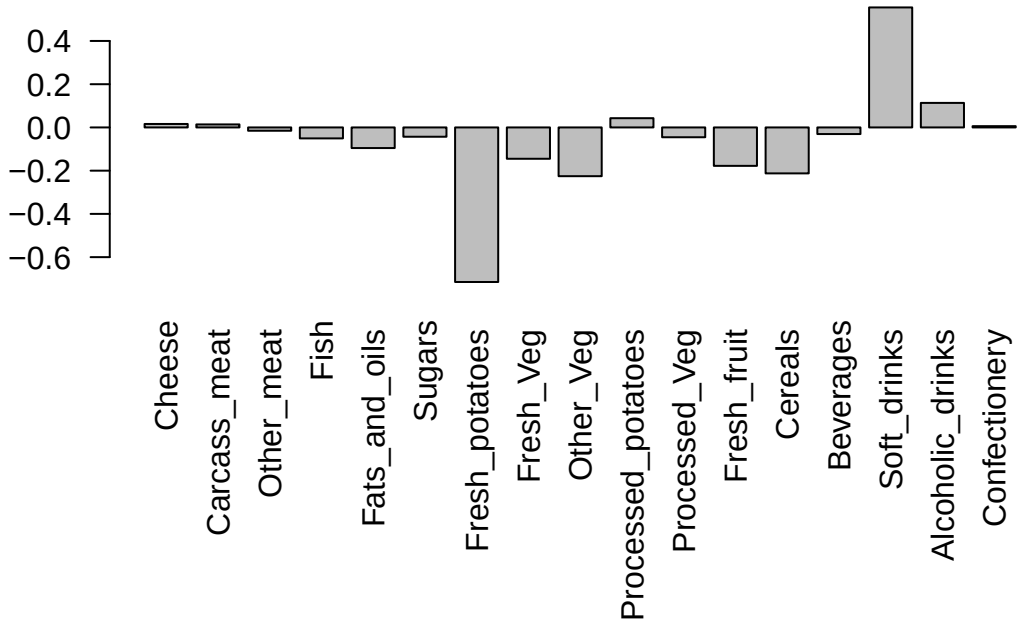
We can also consider the influence of each of the original variables upon the principal components (typically known as loading scores). This information can be obtained from the prcomp() returned $rotation component. It can also be summarized with a call to biplot(), see below:

```
## Lets focus on PC1 as it accounts for > 90% of variance
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?

```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```

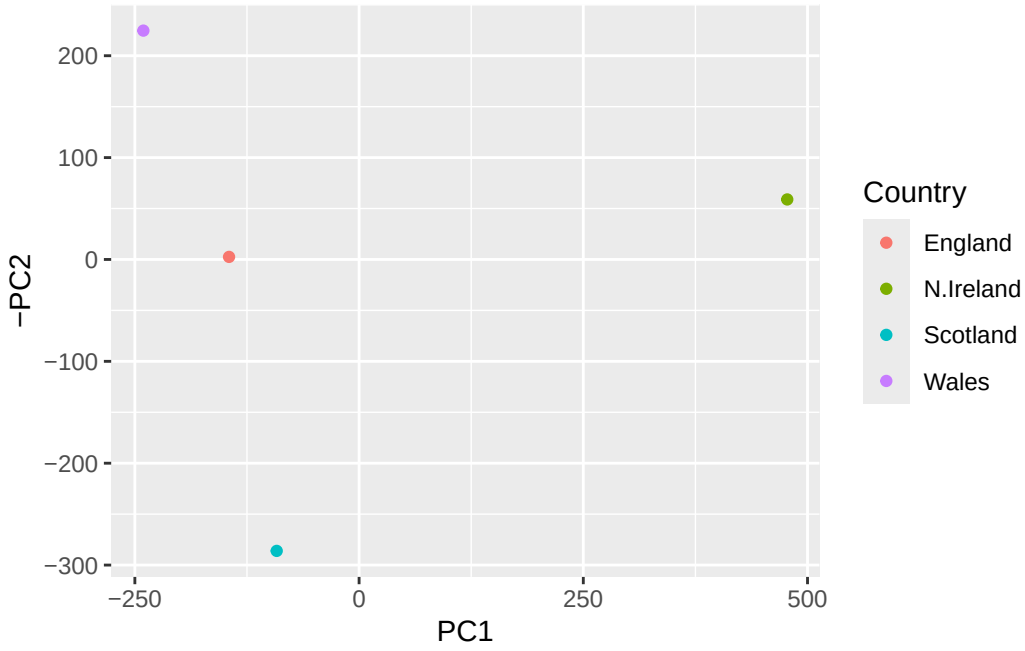


Using ggplot for these figures

We could use the ggplot2 package to make somewhat better figures than all of the above "base" R plots() and barplots(). Recall that ggplot works with data.frames and unfortunately most of the output of these older base R functions like prcomp() are lists of vectors and matrices.

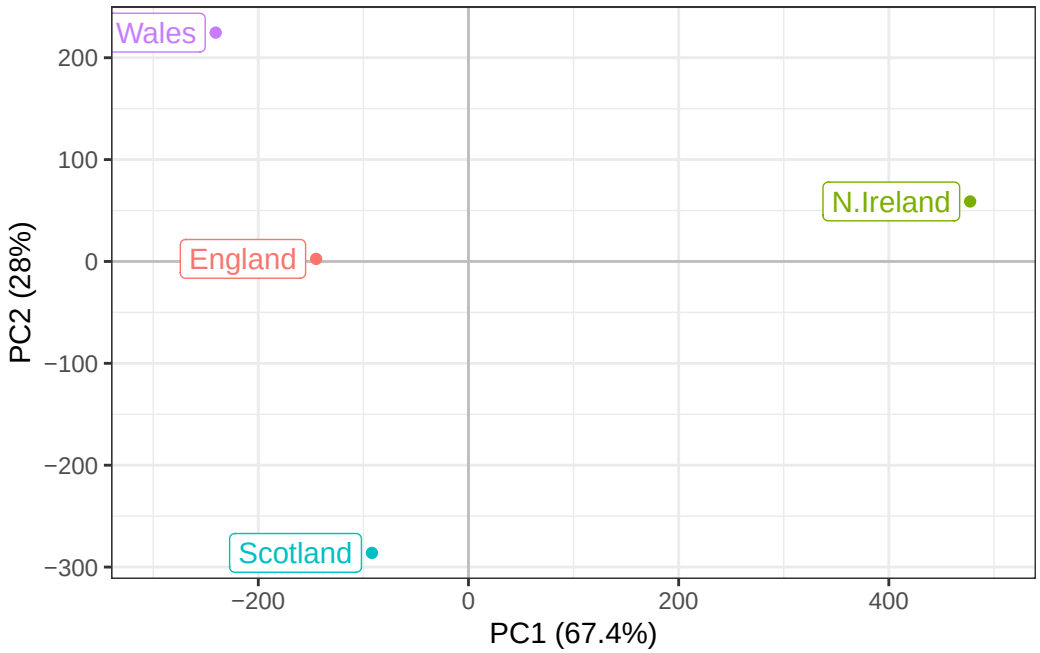
So first we will need to take whatever it is we want to plot and convert it to a data.frame with the as.data.frame() function. Then to make our plotting life easier we will also add the food labels as a column (called "Food") to this data frame with the rownames_to_column() function from the tibble package (you might need to install this):

```
library(ggplot2)

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

# Our first basic plot
ggplot(df_lab) +
  aes(PC1, -PC2, col=Country) +
  geom_point()
```



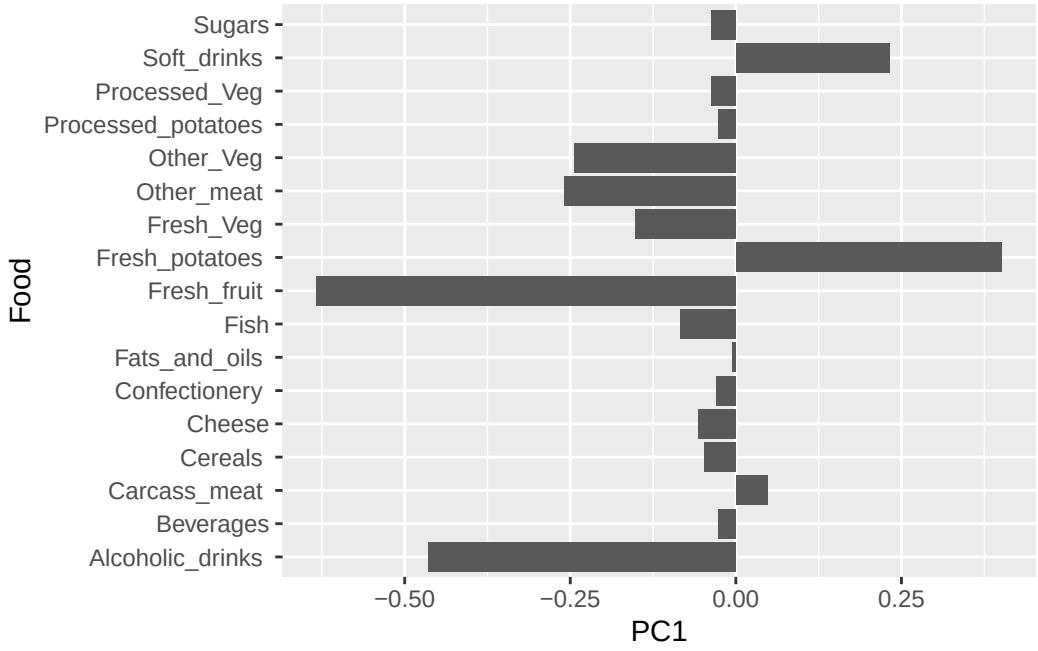
And then we can get carried away and make this look much nicer:

```
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()
```



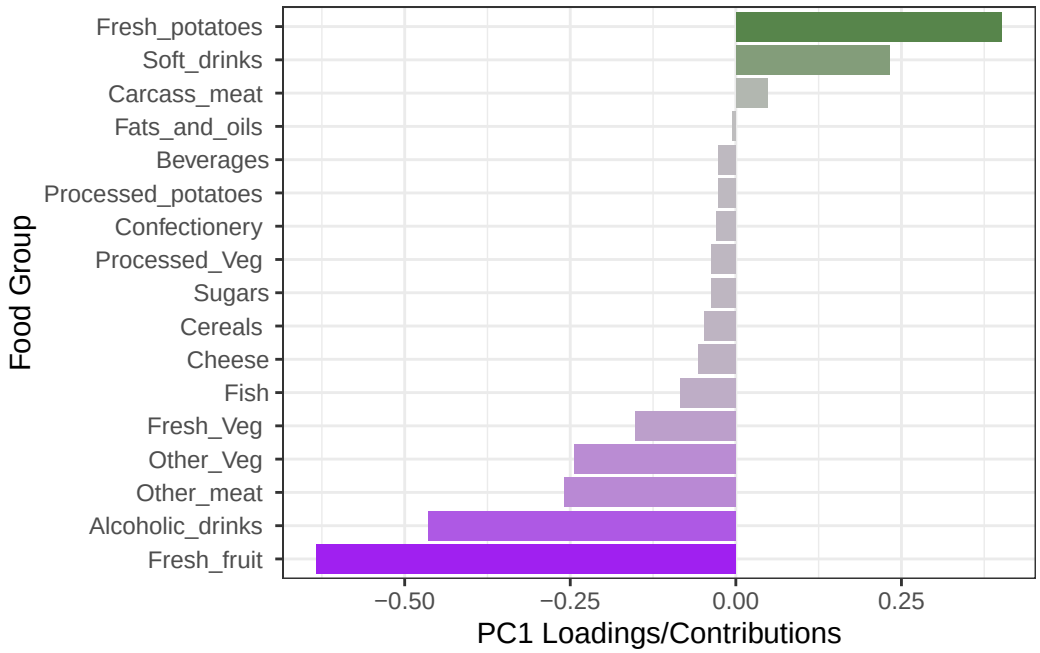
Let's do the same for our loadings/PC contributions figures. This data is stored in the pca$rotation object that we convert to a data frame, add the useful row names as a new column and then plot and customize with additional ggplot layers. Which do you prefer, base graphics or ggplot?

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

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



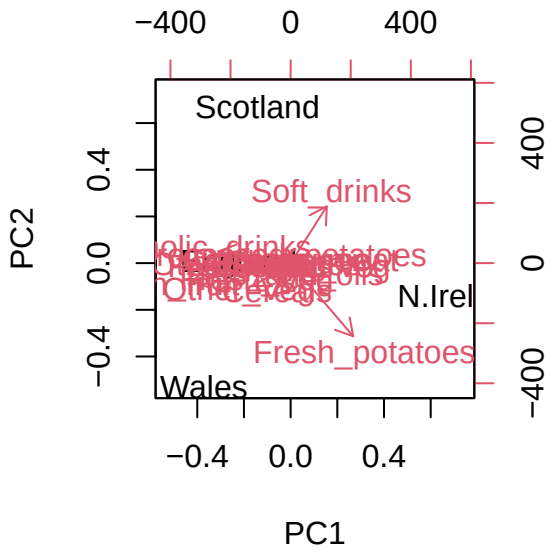
We can now add some additional features to the plot, such as reordering the y axis by the PC1 loadings and selecting a rather ugly color scale (to match our country colors) and our prefered theme layer.

```
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()
```



The inbuilt biplot() can be useful for small datasets

```
biplot(pca)
```



2. PCA of RNA-seq data

RNA-seq results often contain a PCA (or related MDS plot). Usually we use these graphs to verify that the control samples cluster together. However, there's a lot more going on, and if you are willing to dive in, you can extract a lot more information from these plots. The good news is that PCA only sounds complicated. Conceptually, as we have hopefully demonstrated here and in the lecture, it is readily accessible and understandable.

In this example, a small RNA-seq count data set (available from the class website (expression.csv and the tinyurl short link: "https://tinyurl.com/expression-CSV") is read into a data frame called rna.data where the columns are individual samples (i.e. cells) and rows are measurements taken for all the samples (i.e. genes).

```
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
```

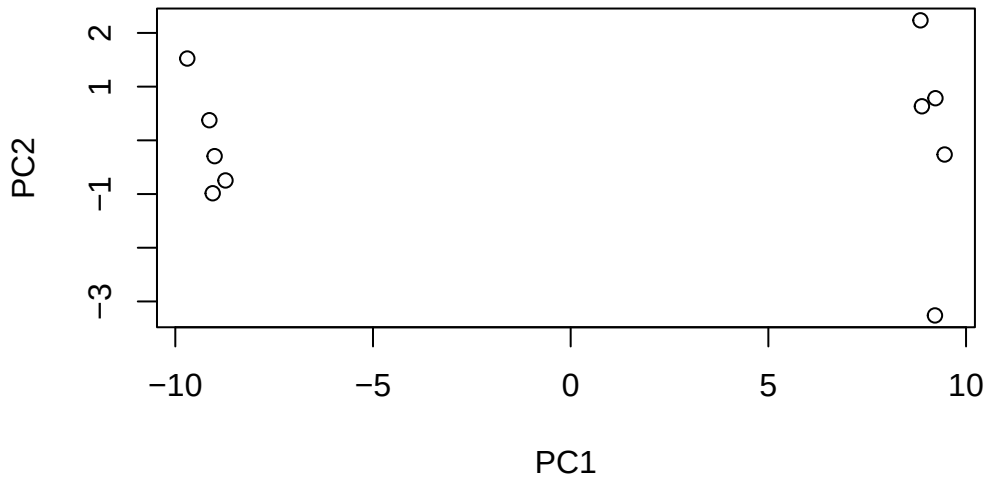
Q10: How many genes and samples are in this data set?

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

```
[1] 100  10
```

The data set has 100 genes and 10 samples

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

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

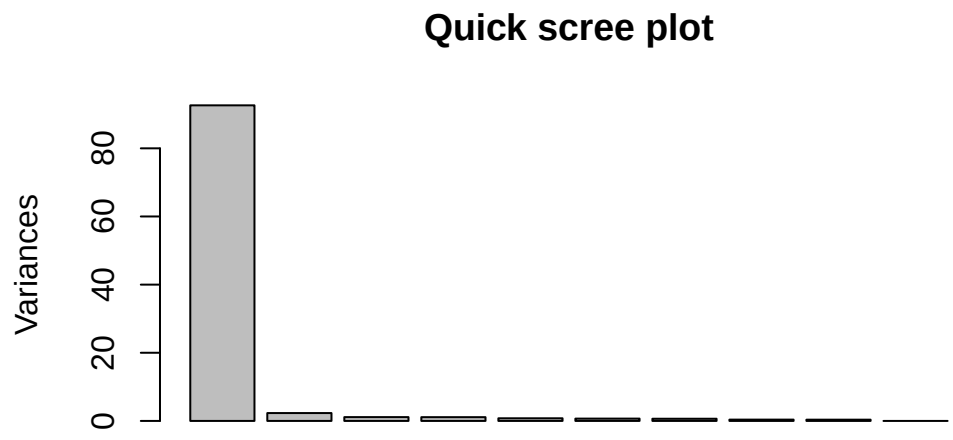



```
summary(pca)
```

```
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 3.457e-15
Proportion of Variance 0.00385 0.00364 0.000e+00
Cumulative Proportion  0.99636 1.00000 1.000e+00
```

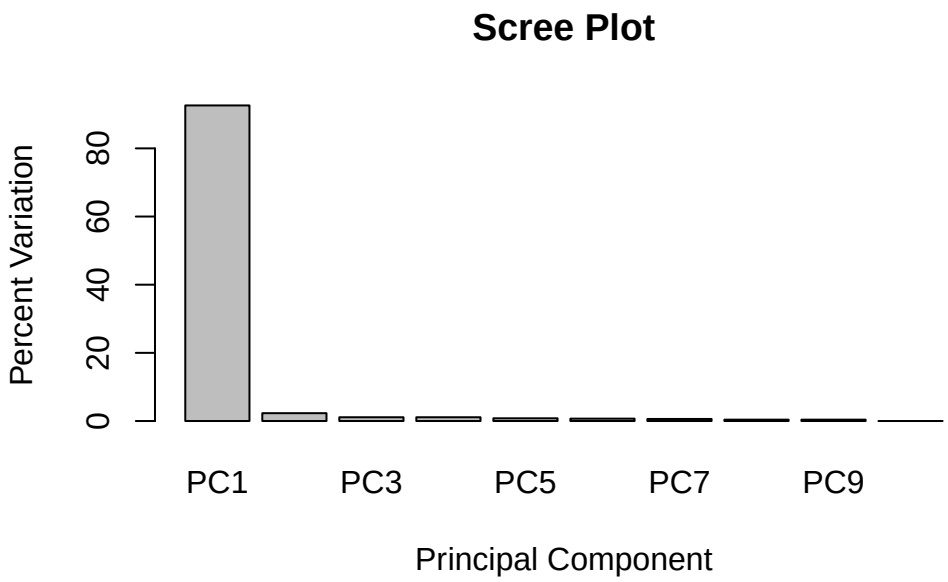
```r
plot(pca, main="Quick scree plot")
```



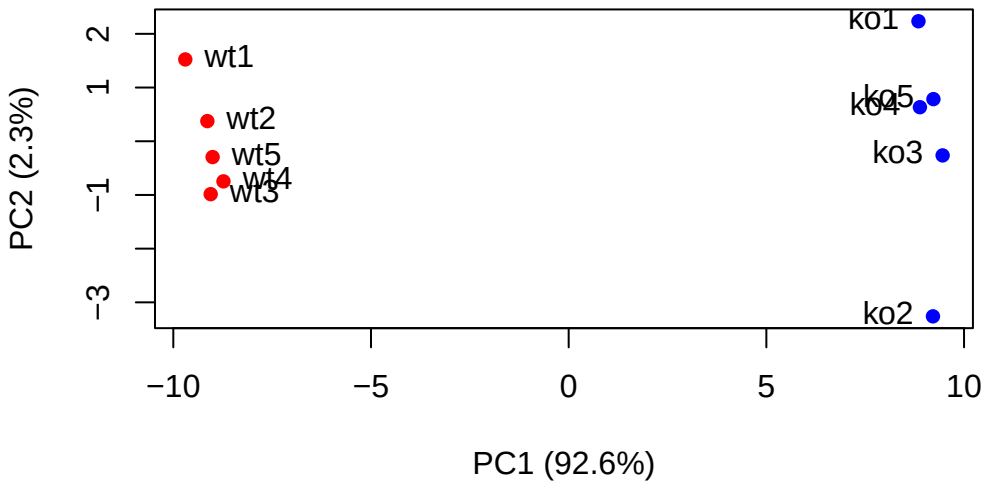

```r
## Variance captured per PC
pca.var <- pca$sdev^2

## Percent variance is often more informative to look at
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
```

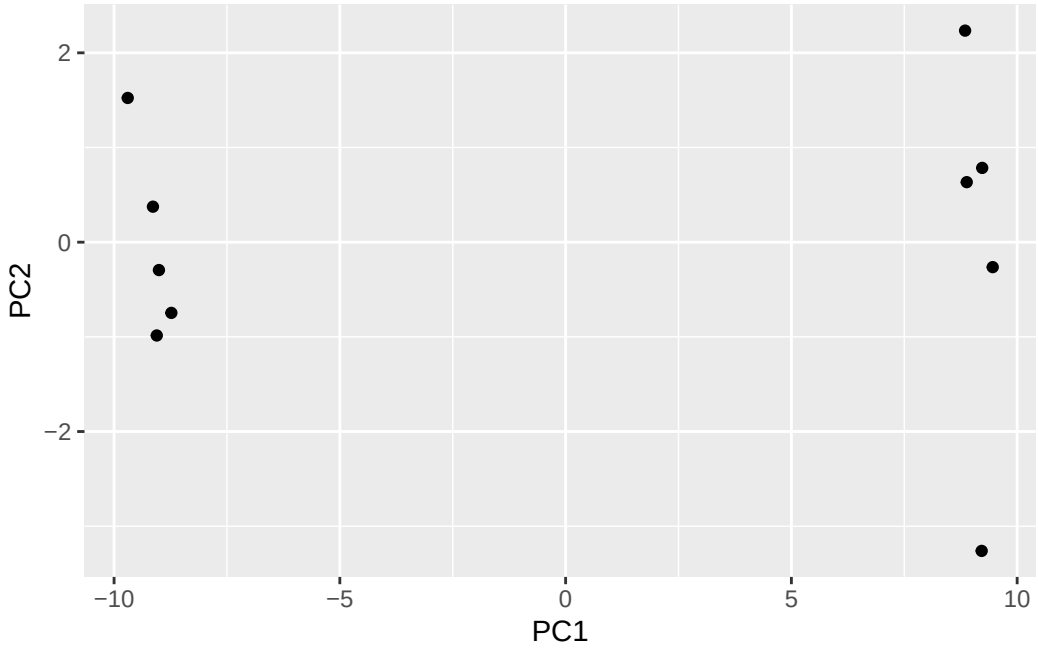
```r
barplot(pca.var.per, main="Scree Plot",
        names.arg = paste0("PC", 1:10),
        xlab="Principal Component", ylab="Percent Variation")
```



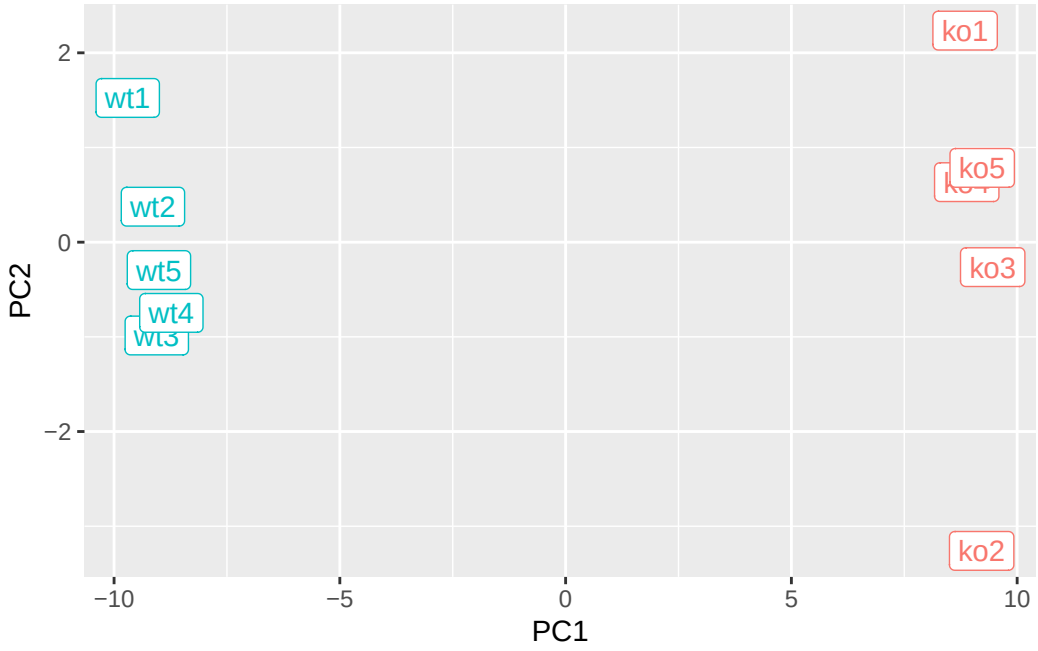

```
## A vector of colors for wt and ko samples
colvec <- colnames(rna.data)
colvec[grep("wt", colvec)] <- "red"
colvec[grep("ko", colvec)] <- "blue"

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)))
```



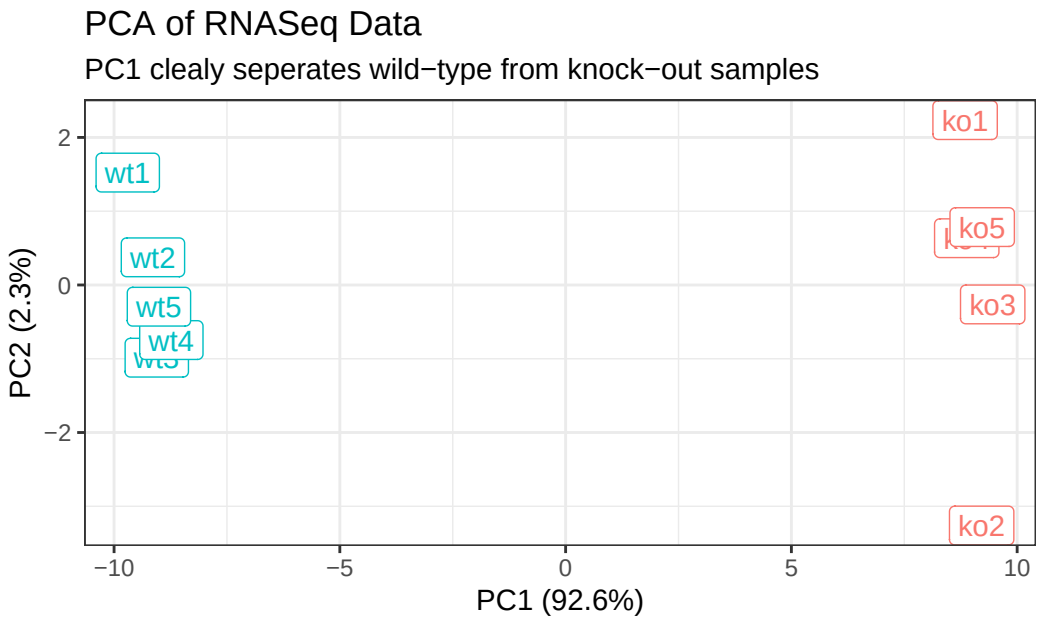

```
library(ggplot2)

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

# Our first basic plot
ggplot(df) +
  aes(PC1, PC2) +
  geom_point()
```




```
# Add a 'wt' and 'ko' "condition" column
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()
```




```
loading_scores <- pca$rotation[,1]

## Find the top 10 measurements (genes) that contribute
## most to PC1 in either direction (+ or -)
gene_scores <- abs(loading_scores)
gene_score_ranked <- sort(gene_scores, decreasing=TRUE)

## show the names of the top 10 genes
top_10_genes <- names(gene_score_ranked[1:10])
top_10_genes
```

```
 [1] "gene100" "gene66"  "gene45"  "gene68"  "gene98"  "gene60"  "gene21"
 [8] "gene56"  "gene10"  "gene90"
```