pca_pcaregression.R

vikaskamath

2021-03-21

```
#### This is an example of how to use PCA & perform PCA regression analysis ####

#install.packages("faraway")
library(faraway)

### Let's first look at the data we will use:

?meatspec

dim(meatspec)
```

```
## [1] 215 101
```

```
head(meatspec)
```

```
##        V1      V2      V3      V4      V5      V6      V7      V8      V9
## 1 2.61776 2.61814 2.61859 2.61912 2.61981 2.62071 2.62186 2.62334 2.62511
## 2 2.83454 2.83871 2.84283 2.84705 2.85138 2.85587 2.86060 2.86566 2.87093
## 3 2.58284 2.58458 2.58629 2.58808 2.58996 2.59192 2.59401 2.59627 2.59873
## 4 2.82286 2.82460 2.82630 2.82814 2.83001 2.83192 2.83392 2.83606 2.83842
## 5 2.78813 2.78989 2.79167 2.79350 2.79538 2.79746 2.79984 2.80254 2.80553
## 6 3.00993 3.01540 3.02086 3.02634 3.03190 3.03756 3.04341 3.04955 3.05599
##       V10     V11     V12     V13     V14     V15     V16     V17     V18
## 1 2.62722 2.62964 2.63245 2.63565 2.63933 2.64353 2.64825 2.65350 2.65937
## 2 2.87661 2.88264 2.88898 2.89577 2.90308 2.91097 2.91953 2.92873 2.93863
## 3 2.60131 2.60414 2.60714 2.61029 2.61361 2.61714 2.62089 2.62486 2.62909
## 4 2.84097 2.84374 2.84664 2.84975 2.85307 2.85661 2.86038 2.86437 2.86860
## 5 2.80890 2.81272 2.81704 2.82184 2.82710 2.83294 2.83945 2.84664 2.85458
## 6 3.06274 3.06982 3.07724 3.08511 3.09343 3.10231 3.11185 3.12205 3.13294
##       V19     V20     V21     V22     V23     V24     V25     V26     V27
## 1 2.66585 2.67281 2.68008 2.68733 2.69427 2.70073 2.70684 2.71281 2.71914
## 2 2.94929 2.96072 2.97272 2.98493 2.99690 3.00833 3.01920 3.02990 3.04101
## 3 2.63361 2.63835 2.64330 2.64838 2.65354 2.65870 2.66375 2.66880 2.67383
## 4 2.87308 2.87789 2.88301 2.88832 2.89374 2.89917 2.90457 2.90991 2.91521
## 5 2.86331 2.87280 2.88291 2.89335 2.90374 2.91371 2.92305 2.93187 2.94060
## 6 3.14457 3.15703 3.17038 3.18429 3.19840 3.21225 3.22552 3.23827 3.25084
##       V28     V29     V30     V31     V32     V33     V34     V35     V36
## 1 2.72628 2.73462 2.74416 2.75466 2.76568 2.77679 2.78790 2.79949 2.81225
## 2 3.05345 3.06777 3.08416 3.10221 3.12106 3.13983 3.15810 3.17623 3.19519
## 3 2.67892 2.68411 2.68937 2.69470 2.70012 2.70563 2.71141 2.71775 2.72490
## 4 2.92043 2.92565 2.93082 2.93604 2.94128 2.94658 2.95202 2.95777 2.96419
```

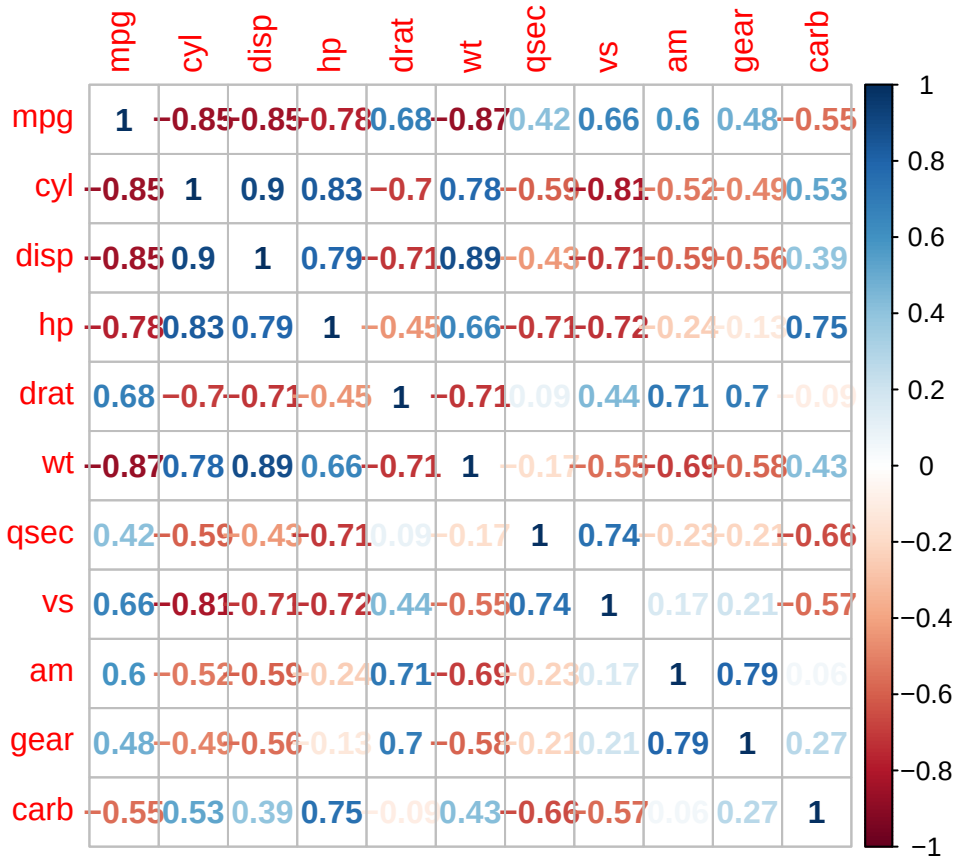
```
## 5 2.94986 2.96035 2.97241 2.98606 3.00097 3.01652 3.03220 3.04793 3.06413
## 6 3.26393 3.27851 3.29514 3.31401 3.33458 3.35591 3.37709 3.39772 3.41828
##       V37     V38     V39     V40     V41     V42     V43     V44     V45
## 1 2.82706 2.84356 2.86106 2.87857 2.89497 2.90924 2.92085 2.93015 2.93846
## 2 3.21584 3.23747 3.25889 3.27835 3.29384 3.30362 3.30681 3.30393 3.29700
## 3 2.73344 2.74327 2.75433 2.76642 2.77931 2.79272 2.80649 2.82064 2.83541
## 4 2.97159 2.98045 2.99090 3.00284 3.01611 3.03048 3.04579 3.06194 3.07889
## 5 3.08153 3.10078 3.12185 3.14371 3.16510 3.18470 3.20140 3.21477 3.22544
## 6 3.43974 3.46266 3.48663 3.51002 3.53087 3.54711 3.55699 3.55986 3.55656
##       V46     V47     V48     V49     V50     V51     V52     V53     V54
## 1 2.94771 2.96019 2.97831 3.00306 3.03506 3.07428 3.11963 3.16868 3.21771
## 2 3.28925 3.28409 3.28505 3.29326 3.30923 3.33267 3.36251 3.39661 3.43188
## 3 2.85121 2.86872 2.88905 2.91289 2.94088 2.97325 3.00946 3.04780 3.08554
## 4 3.09686 3.11629 3.13775 3.16217 3.19068 3.22376 3.26172 3.30379 3.34793
## 5 3.23505 3.24586 3.26027 3.28063 3.30889 3.34543 3.39019 3.44198 3.49800
## 6 3.54937 3.54169 3.53692 3.53823 3.54760 3.56512 3.59043 3.62229 3.65830
##       V55     V56     V57     V58     V59     V60     V61     V62     V63
## 1 3.26254 3.29988 3.32847 3.34899 3.36342 3.37379 3.38152 3.38741 3.39164
## 2 3.46492 3.49295 3.51458 3.53004 3.54067 3.54797 3.55306 3.55675 3.55921
## 3 3.11947 3.14696 3.16677 3.17938 3.18631 3.18924 3.18950 3.18801 3.18498
## 4 3.39093 3.42920 3.45998 3.48227 3.49687 3.50558 3.51026 3.51221 3.51215
## 5 3.55407 3.60534 3.64789 3.68011 3.70272 3.71815 3.72863 3.73574 3.74059
## 6 3.69515 3.72932 3.75803 3.78003 3.79560 3.80614 3.81313 3.81774 3.82079
##       V64     V65     V66     V67     V68     V69     V70     V71     V72
## 1 3.39418 3.39490 3.39366 3.39045 3.38541 3.37869 3.37041 3.36073 3.34979
## 2 3.56045 3.56034 3.55876 3.55571 3.55132 3.54585 3.53950 3.53235 3.52442
## 3 3.18039 3.17411 3.16611 3.15641 3.14512 3.13241 3.11843 3.10329 3.08714
## 4 3.51036 3.50682 3.50140 3.49398 3.48457 3.47333 3.46041 3.44595 3.43005
## 5 3.74357 3.74453 3.74336 3.73991 3.73418 3.72638 3.71676 3.70553 3.69289
## 6 3.82258 3.82301 3.82206 3.81959 3.81557 3.81021 3.80375 3.79642 3.78835
##       V73     V74     V75     V76     V77     V78     V79     V80     V81
## 1 3.33769 3.32443 3.31013 3.29487 3.27891 3.26232 3.24542 3.22828 3.21080
## 2 3.51583 3.50668 3.49700 3.48683 3.47626 3.46552 3.45501 3.44481 3.43477
## 3 3.07014 3.05237 3.03393 3.01504 2.99569 2.97612 2.95642 2.93660 2.91667
## 4 3.41285 3.39450 3.37511 3.35482 3.33376 3.31204 3.28986 3.26730 3.24442
## 5 3.67900 3.66396 3.64785 3.63085 3.61305 3.59463 3.57582 3.55695 3.53796
## 6 3.77958 3.77024 3.76040 3.75005 3.73929 3.72831 3.71738 3.70681 3.69664
##       V82     V83     V84     V85     V86     V87     V88     V89     V90
## 1 3.19287 3.17433 3.15503 3.13475 3.11339 3.09116 3.06850 3.04596 3.02393
## 2 3.42465 3.41419 3.40303 3.39082 3.37731 3.36265 3.34745 3.33245 3.31818
## 3 2.89655 2.87622 2.85563 2.83474 2.81361 2.79235 2.77113 2.75015 2.72956
## 4 3.22117 3.19757 3.17357 3.14915 3.12429 3.09908 3.07366 3.04825 3.02308
## 5 3.51880 3.49936 3.47938 3.45869 3.43711 3.41458 3.39129 3.36772 3.34450
## 6 3.68659 3.67649 3.66611 3.65503 3.64283 3.62938 3.61483 3.59990 3.58535
##       V91     V92     V93     V94     V95     V96     V97     V98     V99
## 1 3.00247 2.98145 2.96072 2.94013 2.91978 2.89966 2.87964 2.85960 2.83940
## 2 3.30473 3.29186 3.27921 3.26655 3.25369 3.24045 3.22659 3.21181 3.19600
## 3 2.70934 2.68951 2.67009 2.65112 2.63262 2.61461 2.59718 2.58034 2.56404
## 4 2.99820 2.97367 2.94951 2.92576 2.90251 2.87988 2.85794 2.83672 2.81617
## 5 3.32201 3.30025 3.27907 3.25831 3.23784 3.21765 3.19766 3.17770 3.15770
## 6 3.57163 3.55877 3.54651 3.53442 3.52221 3.50972 3.49682 3.48325 3.46870
##      V100  fat
## 1 2.81920 22.5
## 2 3.17942 40.1
```

```
## 3 2.54816  8.4
## 4 2.79622  5.9
## 5 3.13753 25.5
## 6 3.45307 42.7
```

```
### Our response variable of interest is fat.

### To get an idea about the structure of the data, let's look at a heat map of the correlation matrix.
### But let's look a data that has no correlation.

### Base R can produce "heatmaps" of the data correlation but the following package is nicer.

# install.packages("corrplot")
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
?corrplot
M <- cor(mtcars)
corrplot(M, method = "number")
```




```
### Explain what a correlation matrix and plot are and run a few examples.

### To calibrate our eyes, let's look at the correlation plot of
###  independently distributed data.
```

```
n_row = nrow(meatspec)
n_row
```

```
## [1] 215
```

```
n_col = ncol(meatspec)
n_col
```

```
## [1] 101
```

```
set.seed(200)
simulated_data = matrix(rnorm(n_row*n_col), n_row, n_col)

head(simulated_data)
```

```
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
## [1,]  0.08475635 -1.0763623  1.7506883 0.93301901 -0.4946630  0.3336762
## [2,]  0.22646034  0.3490071 -0.4719880 0.23241083 -0.1930041  0.2273990
## [3,]  0.43255650 -2.3235292 -0.3977303 0.27924198 -0.9850286 -0.6623948
## [4,]  0.55806524 -1.0222636  1.0392466 0.25846959 -1.1900316  0.8516505
## [5,]  0.05975527 -0.1360361 -0.8744312 0.06565935  0.4929227 -1.0725508
## [6,] -0.11464087 -1.7776987  0.7930322 1.28813712  0.7138606 -1.2832865
##             [,7]       [,8]       [,9]      [,10]      [,11]      [,12]
## [1,] -1.47839885  0.7499767 -0.3011213  0.3821269 -1.1279329  1.9073408
## [2,]  0.55060052  2.0879472 -1.0794214  1.7811801 -0.6338171  1.0651173
## [3,] -0.05280507 -1.6441117  0.6200541 -0.1600355  0.1843089 -1.1278765
## [4,]  1.55808898 -0.1616304  1.3201224 -0.9189571 -0.5521750  0.1102494
## [5,]  1.16046502  2.0851765  1.0067192 -1.5028328 -0.6015227  0.6306605
## [6,]  1.13511743  0.5313593 -0.8058822 -0.8127559 -0.0953380 -1.5887308
##            [,13]      [,14]      [,15]      [,16]      [,17]      [,18]
## [1,] -0.21075577  0.5064882 -0.1924463 -0.3978337 -0.8110883 -1.8955005
## [2,]  0.09820844  1.2199017 -1.0048939 -0.6978304 -0.1761291  0.7521310
## [3,] -0.96576436  0.7480931  0.2215737 -1.9669452 -0.6677863 -2.0406634
## [4,]  0.46681137 -0.0528901 -0.2946896 -0.3333304  1.3054953 -0.6104707
## [5,]  2.13240370 -1.1531076 -0.6401159 -0.4469270 -1.4960778  2.0643218
## [6,]  0.58798772  1.3818172  0.1586785  0.1652997  1.1494378  0.1134043
##            [,19]      [,20]      [,21]      [,22]      [,23]      [,24]
## [1,]  1.7783402 -0.1183706  0.28921974  0.03926498 -1.39076505  1.46808528
## [2,] -0.1072387  0.1227483 -0.54327064  0.17924944 -0.32644187  1.06735961
## [3,] -0.1291278 -0.8156673 -0.06385795  0.10829607  0.04343573 -0.01093337
## [4,] -1.1728815  0.4446182  0.98449893 -0.19452364  0.09734343 -1.95623305
## [5,]  2.7517956  0.4254017 -0.29764163  1.73193303  0.83844768  2.02372985
## [6,] -0.7534270  1.7865080 -0.92523245 -0.29514591  1.11904157  1.74473236
##            [,25]      [,26]      [,27]      [,28]      [,29]      [,30]
## [1,]  0.4730400  1.66210249 -0.75205548 -2.4459645 -1.8012462 -0.5783345
## [2,] -0.8112116 -0.04256802 -0.04163787  1.0488655 -0.6766788 -1.3715817
## [3,] -0.2386180 -0.49331659 -0.15223197 -0.2345446  1.9355891 -0.9264550
## [4,] -0.5950161 -1.54358040 -0.04181483  1.3273845 -0.9337986 -2.5644767
## [5,]  1.3719123 -0.45648329  0.77606499 -0.5338766 -0.7358347 -2.1043653
## [6,] -1.5619458  0.74603041  0.10203716  1.2751531  1.3362554 -1.0469841
##            [,31]      [,32]      [,33]      [,34]      [,35]      [,36]
```

```
## [1,]  0.59136126  1.0641536 -0.8241392 1.85769436  1.7608956  1.8537965
## [2,]  0.53629567 -0.7453631  0.6138689 1.94121695  0.0911907 -0.1164805
## [3,]  0.02030354 -0.6926436 -0.2248578 0.77970240  0.4938296  0.2980373
## [4,] -1.03737387 -1.3520001  0.0567294 1.47800117 -0.1127259  0.1324849
## [5,]  1.48530042 -0.4246535  0.5138586 0.05703903  0.2105586 -0.1272938
## [6,] -0.10231426  0.5942297 -0.4595613 0.70412086 -0.4445429  0.3843618
##           [,37]       [,38]      [,39]      [,40]       [,41]       [,42]
## [1,]  0.47809471  0.20818951 -0.8338248  0.3190540  0.09341359  0.69132615
## [2,] -0.12906658 -0.76633589  0.3813244 -0.5834146 -0.71731529  0.10440450
## [3,] -0.64387651 -0.41476371  0.2136321  1.7247260  1.04733334  0.62453846
## [4,] -0.02845342 -0.34340310 -1.3831728  1.3923395 -0.48247465  0.27655158
## [5,] -0.21822825 -0.04502729  0.1225914 -0.6347982 -0.21653402 -1.27027365
## [6,]  1.21554709 -1.48820828 -1.5704774 -0.4352409 -0.10146586  0.08473175
##           [,43]       [,44]       [,45]      [,46]      [,47]       [,48]
## [1,]  1.12858828 -0.76064555  1.03978710 -0.7468408  2.0075598 -0.03987355
## [2,]  0.11322179 -0.12171003  0.07421448 -0.1145998 -1.5963027  1.36443058
## [3,] -0.08756282  0.04105905 -1.01517106  0.5869118 -0.1353499  1.53623941
## [4,] -2.14477970 -1.02695429 -0.18680178  0.8801537 -2.0504866  0.68620593
## [5,]  0.88333212 -0.67924409 -0.73996253 -0.5995428  1.4882244 -1.63023191
## [6,] -0.89453943  1.09559847 -0.14509676 -0.2074777 -0.1567990  0.47642585
##           [,49]      [,50]       [,51]      [,52]      [,53]      [,54]
## [1,]  0.6592557 -1.3475996 -0.77252080  0.6243319 -0.9117706 -1.1281377
## [2,] -0.6151712 -0.0904501 -0.10564129 -0.4802607 -0.1727344 -0.4612336
## [3,]  1.2206610  0.2748789  0.10696852 -0.7733043 -1.0581801  1.4114389
## [4,] -0.8992801  0.7900278  0.44250279  1.2183554  1.0555947  0.3104291
## [5,] -0.4650668  0.5064733 -0.02565186 -0.6983616  1.2074284 -0.1673483
## [6,]  1.0760452 -0.3108458 -0.62049971  0.1875323 -0.1268406  0.8073165
##           [,55]      [,56]       [,57]      [,58]      [,59]      [,60]
## [1,] -1.03630446 -1.0675422  0.56154007 -1.1997472 -0.8931153 -0.9898096
## [2,]  0.91153435 -1.4375523  0.98825455 -0.3136146 -0.5905109  0.7721553
## [3,]  0.54208959  0.9404785  0.96046081 -0.2267042  0.3534325  0.3842003
## [4,]  0.91111623  1.1756638 -1.59239165  0.8171347 -0.6521539 -0.9640944
## [5,]  0.01387067  1.8807286 -1.56468676 -0.8699376 -0.2571731 -0.4255546
## [6,] -0.92122855 -1.1081753 -0.02131359 -0.4360406  0.5813202 -0.2137216
##           [,61]       [,62]      [,63]       [,64]      [,65]      [,66]
## [1,]  0.03439218  0.25252101  0.1502535 -0.49498165 -1.8514113 -0.8331526
## [2,] -2.00088667 -0.33841771 -0.2182518 -0.62449616 -1.3334970 -2.1358617
## [3,]  2.11016283  1.10335784  0.1519920 -1.36964946 -0.8142130  0.9642059
## [4,] -0.51810178 -2.87303736 -0.5187963  0.06953224 -0.3382068  0.1661545
## [5,]  0.50277399  0.04657395 -0.1433363 -0.27962366 -2.8187410 -0.1519848
## [6,]  1.05237270 -1.55125313  0.4264461  0.47155597 -0.4263986 -0.9796080
##           [,67]      [,68]      [,69]       [,70]      [,71]       [,72]
## [1,] -1.39499765  0.3178009 -1.2033610 -0.38284708 -0.6858962 -0.07490661
## [2,]  0.44377577  0.5955138 -0.0897471 -0.43861178 -2.6303328  1.58351408
## [3,]  0.81474529  1.3404674 -0.4588421 -0.17626281 -0.5993032  0.98007132
## [4,]  0.72366470 -0.1073231  1.3385357 -0.03598364  0.6406015 -1.56624601
## [5,] -0.64462204  0.2692260  1.2610061 -0.49059163 -0.7835260 -1.39466739
## [6,]  0.02128675  0.2962441 -2.0727821 -0.14370079 -0.2653598 -0.74652730
##           [,73]      [,74]      [,75]       [,76]      [,77]      [,78]
## [1,]  0.6966902  0.4405067 -0.1733955  1.31449245 -1.0920495  1.9922502
## [2,]  0.9112216 -0.1243570 -1.9307901 -0.74276345 -1.8055216 -0.3221581
## [3,]  1.2144399  1.4058574 -0.2623772 -1.36290430  0.3322608  0.6769349
## [4,] -0.2047515  0.3361619 -1.6113855  0.04852357 -0.4680480 -0.5393737
## [5,] -0.1576023 -2.2720586 -1.5716906 -1.85141146  0.3866745  0.3510765
```

```
## [6,]  1.6542346 -0.5703144  0.8202650 -1.73918572  0.2205564  1.2689394
##            [,79]      [,80]      [,81]       [,82]       [,83]      [,84]
## [1,] -0.89596399  0.2570740  0.2906488 -0.99827756 -0.96590424  1.7045708
## [2,] -0.60481972  0.1079606  0.1990808  1.25875884  0.01357255 -1.4988646
## [3,]  0.03169231 -2.0221791 -0.2112921 -0.01709169 -0.26890814 -0.4822556
## [4,]  0.69226231  1.8329484  0.3192508  0.22655428  1.03381188  0.2336506
## [5,]  1.65927233  1.2389812 -1.2065993  0.42957638  1.80548113 -1.3167757
## [6,] -1.06938721  0.4621096 -2.0186602 -0.12170938  0.24923886  0.4996826
##            [,85]      [,86]       [,87]       [,88]       [,89]      [,90]
## [1,] -0.06239699  1.2944951 -0.03159267 -0.17777763 -0.01771276 -1.1465045
## [2,]  0.19302425 -2.0255581 -0.40458940  0.22748682 -0.85104703 -1.3493923
## [3,]  0.43889264 -0.3058154  0.01872497  1.19120252  1.10160043  0.5111628
## [4,]  0.69313574  0.6620237  0.72897361  0.64931281 -0.05954708 -0.1493409
## [5,] -1.79667622 -1.1488236  0.26345929 -0.02183333 -1.13124815  0.7312853
## [6,] -0.79715916  0.1205694  0.96380077 -0.24647915  0.67326001  0.3498722
##            [,91]      [,92]       [,93]       [,94]      [,95]       [,96]
## [1,]  0.5837884 -0.1678869 -0.71523205 -1.09935446  0.9401923  1.5923956
## [2,]  0.4471050  0.5511786 -0.03067367 -0.40568091 -1.0369599  0.1126286
## [3,]  0.1797283 -1.8151210  0.09482088 -0.03882875 -0.6292236  0.0204352
## [4,] -1.3422349 -0.6174411 -0.20139420 -1.20666679 -0.7825186  1.5635045
## [5,]  0.4887615  2.0358111  1.61162135 -0.54447328 -0.4541385 -0.7361508
## [6,] -1.6744004 -0.4964060 -1.72290388 -0.49198047 -0.5874646  1.9634258
##            [,97]      [,98]      [,99]     [,100]      [,101]
## [1,] -1.4134896  0.8717244  1.2808729 -0.1692346 -1.12126145
## [2,] -1.3253098 -0.3196441  1.0989019  0.2470156 -0.65045703
## [3,]  1.5533905  0.6533753 -0.9994334  2.9766726  1.12080952
## [4,]  0.6901032  0.4546428  0.6615051  1.5918794  0.17009129
## [5,]  1.8545839 -0.3911771  0.4400238  0.5390242  0.02772295
## [6,] -0.8135587 -1.8662842  1.4565178  0.2984280  1.16507742
```

```
cor_sim = cor(simulated_data)

dim(cor_sim)
```

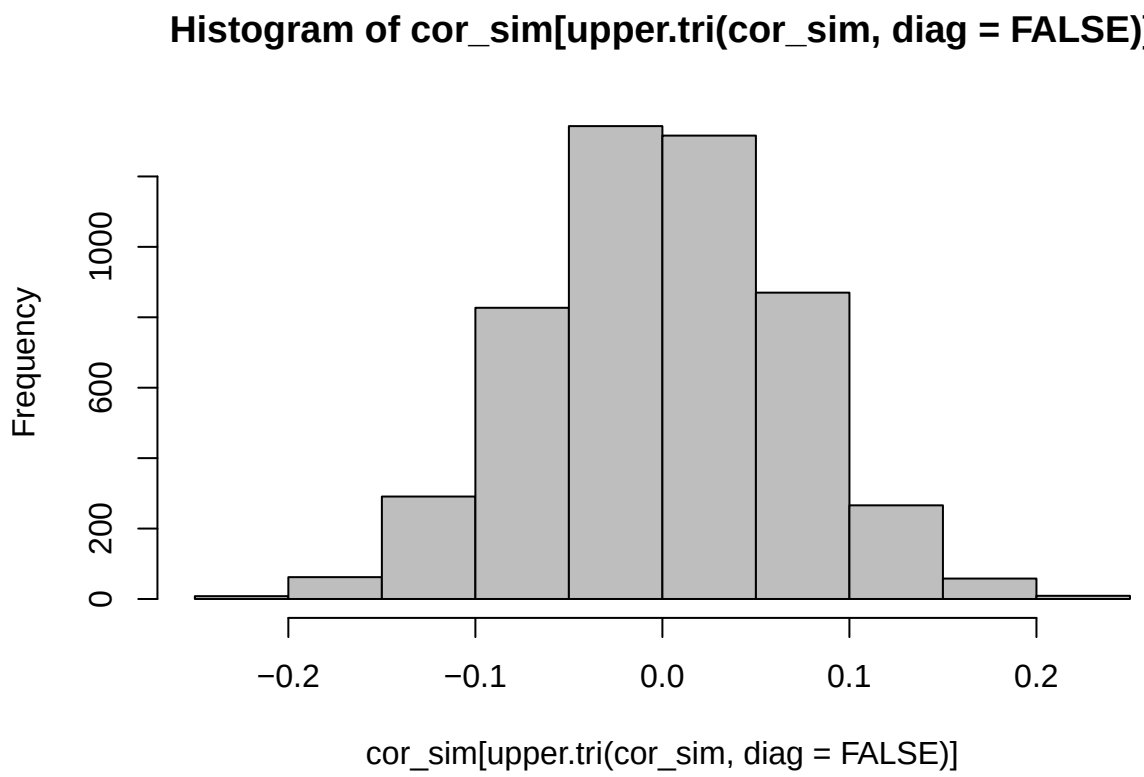
```
## [1] 101 101
```

```
cor_sim[1:10,1:10]
```

```
##                [,1]        [,2]         [,3]        [,4]         [,5]
##  [1,]  1.000000000  0.08439662  0.062169100 -0.11811690 -0.007955169
##  [2,]  0.084396618  1.00000000  0.060195047 -0.02965581  0.077814517
##  [3,]  0.062169100  0.06019505  1.000000000  0.08651927  0.011282514
##  [4,] -0.118116896 -0.02965581  0.086519275  1.00000000 -0.080460462
##  [5,] -0.007955169  0.07781452  0.011282514 -0.08046046  1.000000000
##  [6,]  0.097326830 -0.03896546 -0.121902996 -0.09377456  0.118252645
##  [7,]  0.085990961 -0.02843405  0.009203473  0.05048555 -0.014957915
##  [8,] -0.091117320 -0.05117835  0.015364491  0.08358424  0.008398079
##  [9,] -0.002761098 -0.03042461  0.025882579 -0.03586588 -0.071362699
## [10,]  0.052729956 -0.10563777  0.050582377  0.06293523 -0.075330699
##               [,6]         [,7]         [,8]         [,9]       [,10]
##  [1,]  0.09732683  0.085990961 -0.091117320 -0.002761098  0.05272996
##  [2,] -0.03896546 -0.028434054 -0.051178346 -0.030424606 -0.10563777
##  [3,] -0.12190300  0.009203473  0.015364491  0.025882579  0.05058238
```

```
##  [4,] -0.09377456  0.050485547  0.083584237 -0.035865880  0.06293523
##  [5,]  0.11825265 -0.014957915  0.008398079 -0.071362699 -0.07533070
##  [6,]  1.00000000 -0.141565295 -0.107051547  0.083418399 -0.01829624
##  [7,] -0.14156529  1.000000000  0.061859128 -0.066915488  0.04584342
##  [8,] -0.10705155  0.061859128  1.000000000 -0.035789814  0.01712245
##  [9,]  0.08341840 -0.066915488 -0.035789814  1.000000000 -0.06365527
## [10,] -0.01829624  0.045843418  0.017122454 -0.063655268  1.00000000
```

```
### Describe the correlation matrix.

hist(cor_sim[ upper.tri(cor_sim, diag = FALSE) ], col = "grey")
```



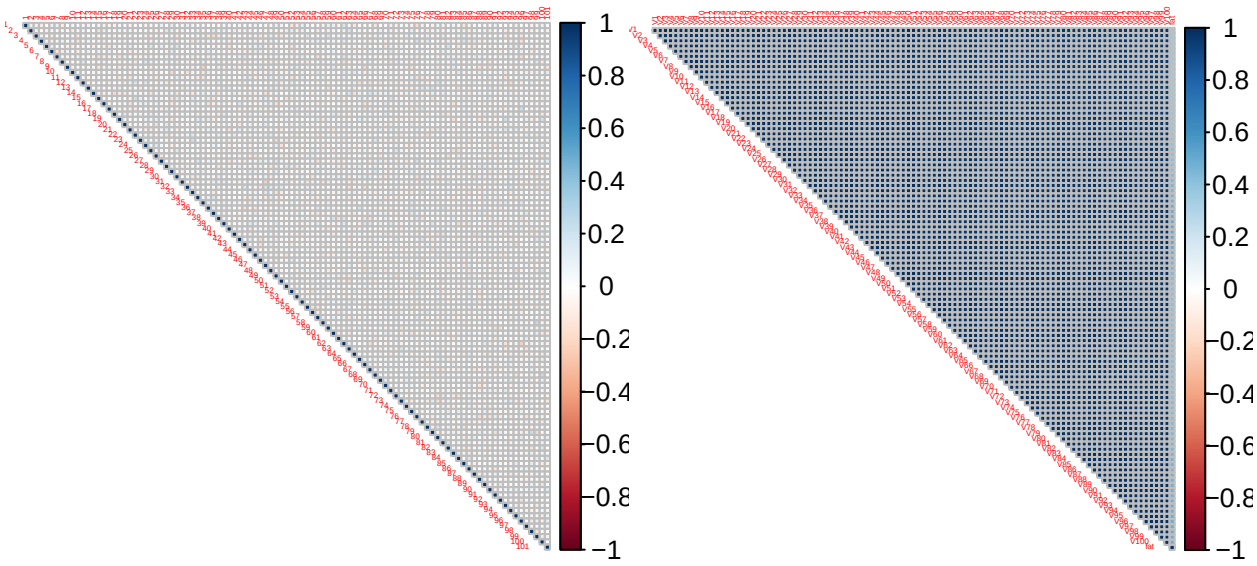

```
### Just by chance, we some larger correlations (near +0.2 and -0.2).

### Why do you think the output of the following is?

identical(t(cor_sim), cor_sim)
```

```
## [1] TRUE
```

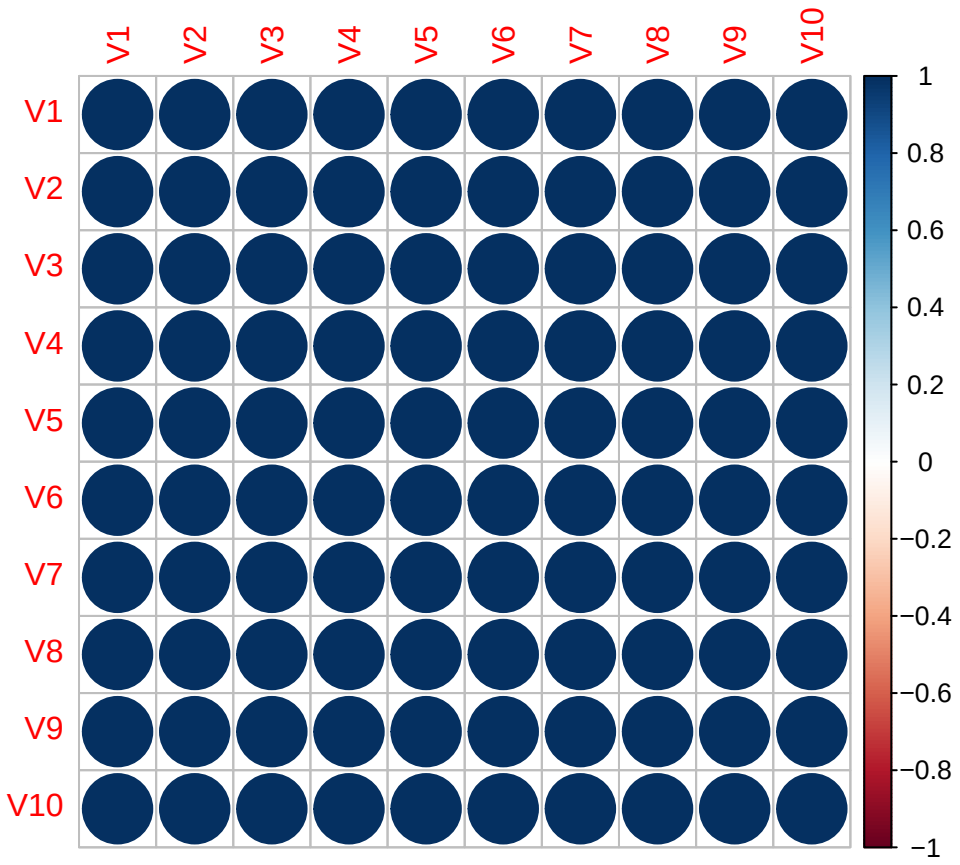
```
### Here come the plots:

cor_meatspec = cor(meatspec)
cor_meatspec[1:10,1:10]
```

```
##            V1        V2        V3        V4        V5        V6        V7
## V1  1.0000000 0.9999908 0.9999649 0.9999243 0.9998715 0.9998088 0.9997385
## V2  0.9999908 1.0000000 0.9999916 0.9999678 0.9999309 0.9998832 0.9998269
## V3  0.9999649 0.9999916 1.0000000 0.9999923 0.9999707 0.9999373 0.9998945
## V4  0.9999243 0.9999678 0.9999923 1.0000000 0.9999930 0.9999735 0.9999436
## V5  0.9998715 0.9999309 0.9999707 0.9999930 1.0000000 0.9999937 0.9999763
## V6  0.9998088 0.9998832 0.9999373 0.9999735 0.9999937 1.0000000 0.9999944
## V7  0.9997385 0.9998269 0.9998945 0.9999436 0.9999763 0.9999944 1.0000000
## V8  0.9996629 0.9997642 0.9998442 0.9999054 0.9999496 0.9999788 0.9999950
## V9  0.9995831 0.9996963 0.9997878 0.9998600 0.9999150 0.9999546 0.9999807
## V10 0.9995015 0.9996254 0.9997272 0.9998095 0.9998742 0.9999233 0.9999587
##            V8        V9       V10
## V1  0.9996629 0.9995831 0.9995015
## V2  0.9997642 0.9996963 0.9996254
## V3  0.9998442 0.9997878 0.9997272
## V4  0.9999054 0.9998600 0.9998095
## V5  0.9999496 0.9999150 0.9998742
## V6  0.9999788 0.9999546 0.9999233
## V7  0.9999950 0.9999807 0.9999587
## V8  1.0000000 0.9999954 0.9999824
## V9  0.9999954 1.0000000 0.9999958
## V10 0.9999824 0.9999958 1.0000000
```

```
par(mfrow=c(1,2))

corrplot(cor_sim,      tl.cex = 0.25, type = "upper")
corrplot(cor_meatspec, tl.cex = 0.25, type = "upper")
```




```
par(mfrow=c(1,1))

### Let's look some smaller parts:

corrplot(cor_meatspec[1:10,1:10])
```

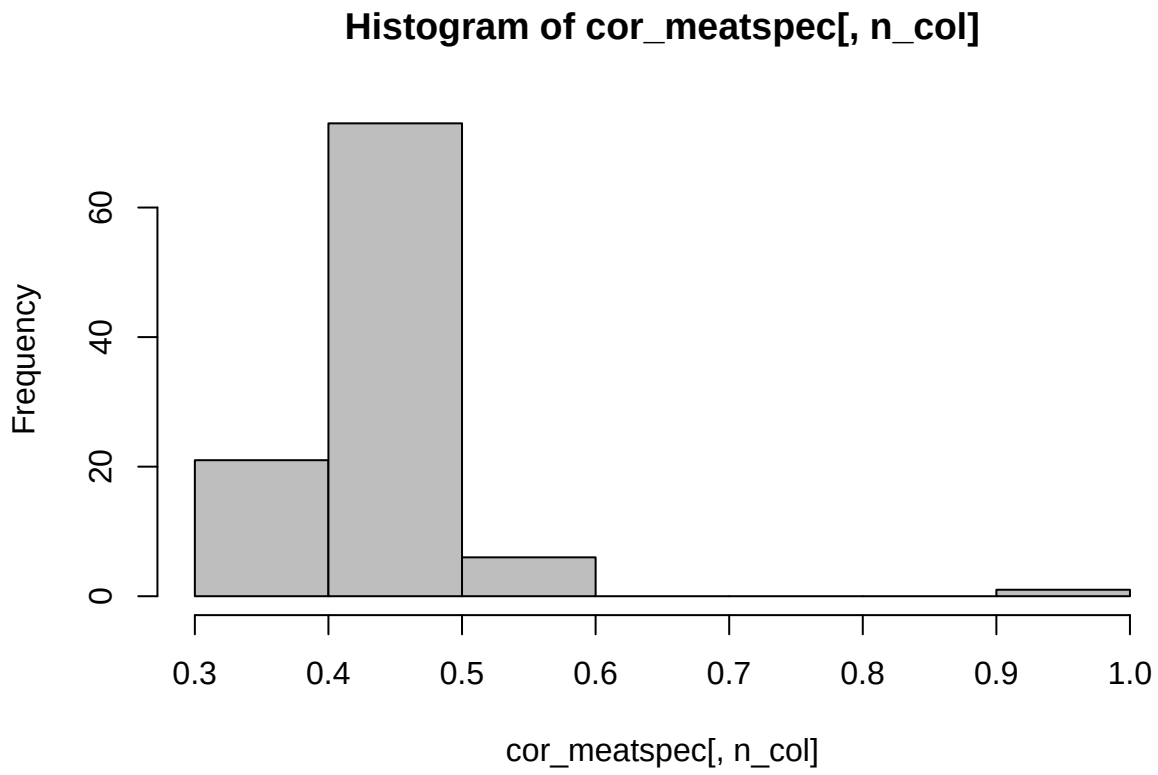



```
cor_meatspec[,n_col]
```

```
##        V1        V2        V3        V4        V5        V6        V7        V8
## 0.3692128 0.3682434 0.3673780 0.3666246 0.3660194 0.3656044 0.3654182 0.3654829
##        V9       V10       V11       V12       V13       V14       V15       V16
## 0.3658032 0.3664009 0.3672576 0.3684259 0.3699811 0.3720024 0.3745331 0.3775638
##       V17       V18       V19       V20       V21       V22       V23       V24
## 0.3810669 0.3850746 0.3895866 0.3945111 0.3996250 0.4045761 0.4090352 0.4128595
##       V25       V26       V27       V28       V29       V30       V31       V32
## 0.4161760 0.4193564 0.4230144 0.4277806 0.4340004 0.4417220 0.4505601 0.4598266
##       V33       V34       V35       V36       V37       V38       V39       V40
## 0.4688135 0.4771119 0.4847793 0.4921960 0.4996134 0.5067275 0.5127979 0.5168768
##       V41       V42       V43       V44       V45       V46       V47       V48
## 0.5179639 0.5152153 0.5082469 0.4973548 0.4835660 0.4684806 0.4538471 0.4412607
##       V49       V50       V51       V52       V53       V54       V55       V56
## 0.4313514 0.4241971 0.4195589 0.4170115 0.4160567 0.4161754 0.4168775 0.4177710
##       V57       V58       V59       V60       V61       V62       V63       V64
## 0.4185752 0.4191938 0.4195832 0.4198345 0.4200706 0.4203303 0.4206748 0.4211071
##       V65       V66       V67       V68       V69       V70       V71       V72
## 0.4216116 0.4221807 0.4227946 0.4235342 0.4244604 0.4256225 0.4269973 0.4285851
##       V73       V74       V75       V76       V77       V78       V79       V80
## 0.4304236 0.4324907 0.4347014 0.4370064 0.4394687 0.4421501 0.4451316 0.4484778
##       V81       V82       V83       V84       V85       V86       V87       V88
## 0.4521013 0.4558695 0.4596247 0.4631830 0.4663355 0.4689114 0.4709441 0.4726796
##       V89       V90       V91       V92       V93       V94       V95       V96
```

```
## 0.4744531 0.4765180 0.4789412 0.4815959 0.4842482 0.4866933 0.4887670 0.4903475
##       V97       V98       V99      V100       fat
## 0.4912600 0.4913482 0.4905760 0.4891052 1.0000000
```

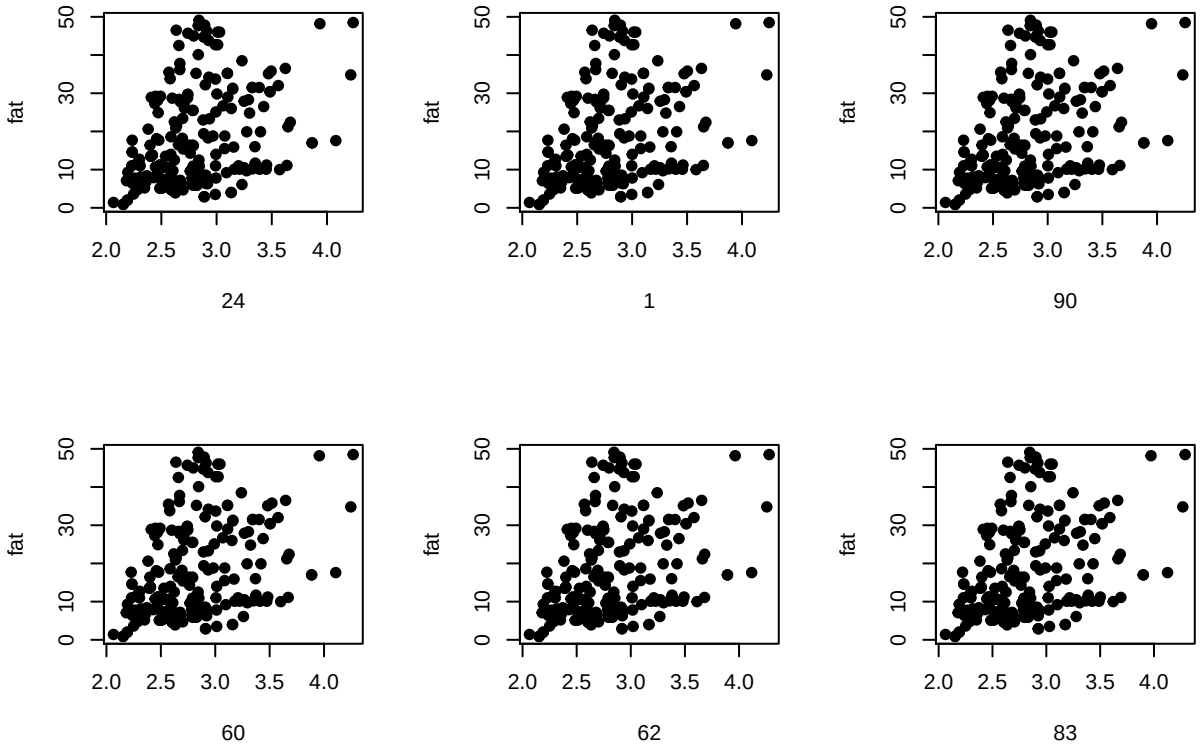
```
hist(cor_meatspec[,n_col], col = "grey")
```



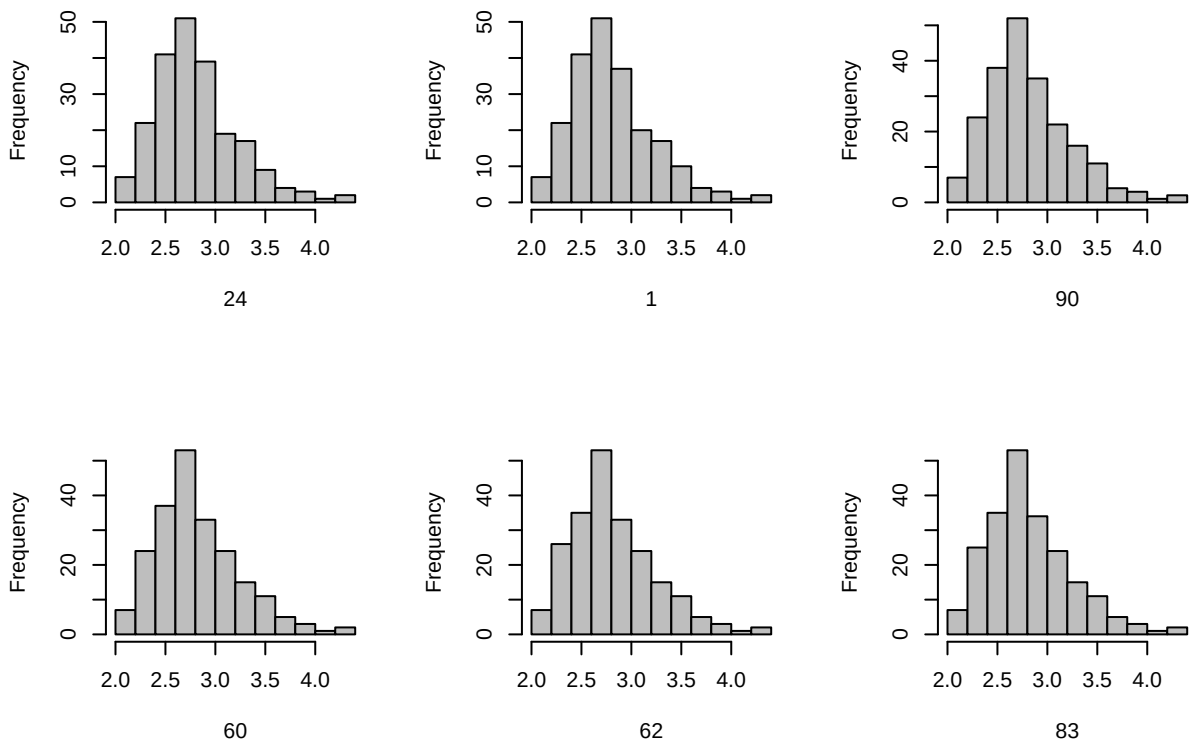

```
### Let's look at a 6  plots chosen at random.

set.seed(777)
k = sample(1:(n_col-1), size = 6, replace = F)
k
```

```
## [1] 24  1 90 60 62 83
```

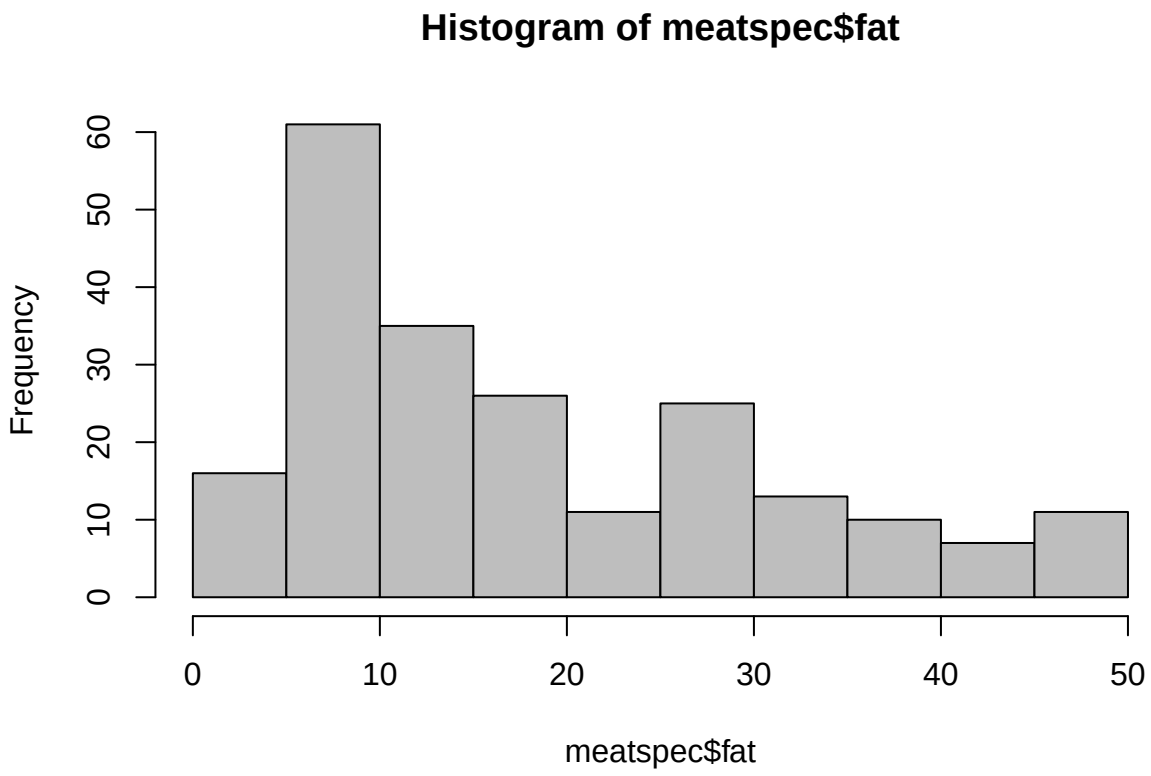
```
par(mfrow=c(2,3))
for (i in 1:6)
  plot(fat ~ meatspec[,i], data = meatspec, pch = 19, xlab = k[i], ylab = "fat")
```




```
par(mfrow=c(1,1))

par(mfrow=c(2,3))
for (i in 1:6)
  hist(meatspec[,i], col = "grey", main ="", xlab = k[i])
```




```
par(mfrow=c(1,1))

hist(meatspec$fat, col = "grey")
```



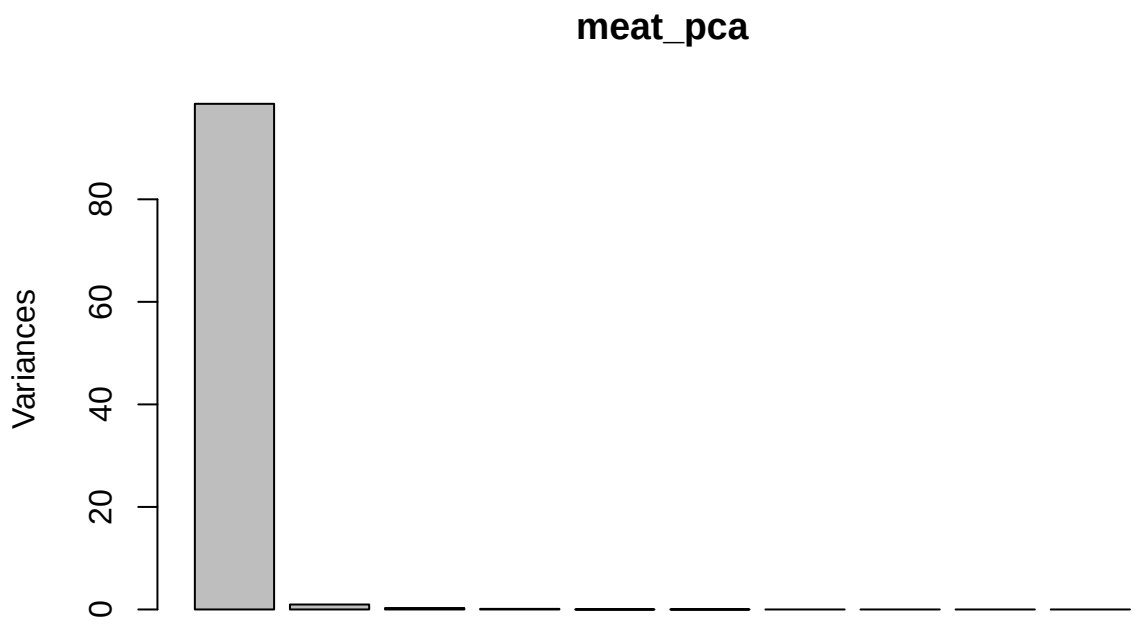

```
### This data is a great candidate for PCA: highly correlated data.
### To start, let's just see if dimension reduction is possible.  We'll ignore
###  the fat variable: I take out the y variable which is fat in the 101 column.

meat_pca = prcomp(meatspec[,-101], scale = TRUE)
summary(meat_pca)
```

```
## Importance of components:
##                            PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     9.9311 0.9847 0.52851 0.33827 0.08038 0.05123 0.02681
## Proportion of Variance 0.9863 0.0097 0.00279 0.00114 0.00006 0.00003 0.00001
## Cumulative Proportion  0.9863 0.9960 0.99875 0.99990 0.99996 0.99999 0.99999
##                            PC8      PC9     PC10     PC11     PC12     PC13
## Standard deviation     0.01961 0.008564 0.006739 0.004442 0.003361 0.001867
## Proportion of Variance 0.00000 0.000000 0.000000 0.000000 0.000000 0.000000
## Cumulative Proportion  1.00000 1.000000 1.000000 1.000000 1.000000 1.000000
##                           PC14      PC15      PC16      PC17      PC18
## Standard deviation     0.001377 0.0009449 0.0008641 0.0007558 0.0006977
## Proportion of Variance 0.000000 0.0000000 0.0000000 0.0000000 0.0000000
## Cumulative Proportion  1.000000 1.0000000 1.0000000 1.0000000 1.0000000
##                           PC19      PC20      PC21      PC22      PC23
## Standard deviation     0.0005884 0.0004628 0.0003897 0.0003341 0.0003123
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## Cumulative Proportion  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##                           PC24      PC25     PC26      PC27      PC28
## Standard deviation     0.0002721 0.0002616 0.000211 0.0001954 0.0001857
```

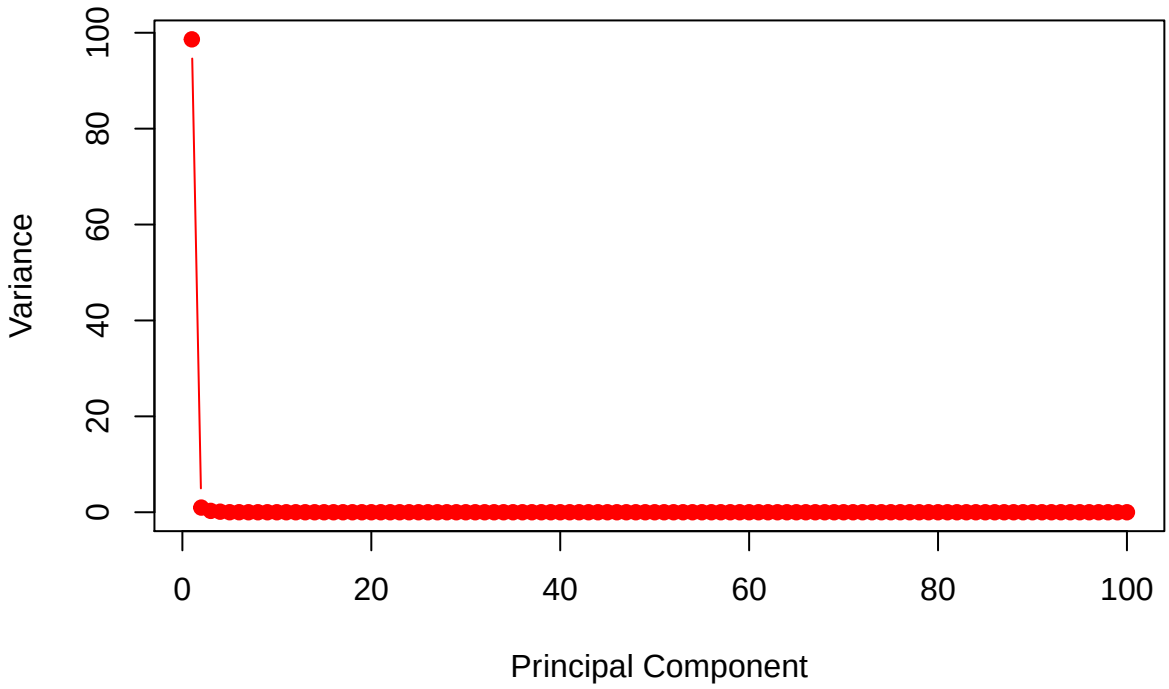
```
## Proportion of Variance 0.0000000 0.0000000 0.000000 0.0000000 0.0000000
## Cumulative Proportion  1.0000000 1.0000000 1.000000 1.0000000 1.0000000
##                            PC29      PC30      PC31      PC32      PC33
## Standard deviation     0.0001729 0.0001656 0.0001539 0.0001473 0.0001392
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
## Cumulative Proportion  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
##                            PC34      PC35      PC36     PC37     PC38
## Standard deviation     0.0001339 0.0001269 0.0001082 0.000104 9.98e-05
## Proportion of Variance 0.0000000 0.0000000 0.0000000 0.000000 0.00e+00
## Cumulative Proportion  1.0000000 1.0000000 1.0000000 1.000000 1.00e+00
##                            PC39      PC40      PC41      PC42     PC43
## Standard deviation     9.081e-05 8.668e-05 8.026e-05 7.762e-05 7.36e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.00e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.00e+00
##                            PC44      PC45     PC46      PC47      PC48
## Standard deviation     6.808e-05 6.541e-05 6.44e-05 5.897e-05 5.422e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.00e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.00e+00 1.000e+00 1.000e+00
##                            PC49      PC50      PC51      PC52      PC53
## Standard deviation     5.027e-05 4.893e-05 4.608e-05 4.419e-05 4.037e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                            PC54    PC55     PC56      PC57      PC58      PC59
## Standard deviation     3.854e-05 3.8e-05 3.64e-05 3.497e-05 3.443e-05 3.264e-05
## Proportion of Variance 0.000e+00 0.0e+00 0.00e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.0e+00 1.00e+00 1.000e+00 1.000e+00 1.000e+00
##                            PC60     PC61      PC62      PC63      PC64
## Standard deviation     3.104e-05 3.04e-05 2.959e-05 2.844e-05 2.699e-05
## Proportion of Variance 0.000e+00 0.00e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.00e+00 1.000e+00 1.000e+00 1.000e+00
##                            PC65      PC66      PC67      PC68      PC69
## Standard deviation     2.586e-05 2.388e-05 2.364e-05 2.284e-05 2.173e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                            PC70      PC71     PC72      PC73      PC74
## Standard deviation     2.058e-05 1.997e-05 1.93e-05 1.854e-05 1.807e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.00e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.00e+00 1.000e+00 1.000e+00
##                            PC75      PC76      PC77      PC78      PC79
## Standard deviation     1.728e-05 1.693e-05 1.612e-05 1.569e-05 1.516e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                            PC80      PC81      PC82      PC83      PC84
## Standard deviation     1.445e-05 1.408e-05 1.356e-05 1.275e-05 1.224e-05
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                            PC85     PC86      PC87      PC88      PC89
## Standard deviation     1.178e-05 1.09e-05 1.045e-05 1.009e-05 9.396e-06
## Proportion of Variance 0.000e+00 0.00e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.00e+00 1.000e+00 1.000e+00 1.000e+00
##                            PC90     PC91      PC92     PC93      PC94
## Standard deviation     8.728e-06 8.27e-06 7.613e-06 6.83e-06 6.383e-06
## Proportion of Variance 0.000e+00 0.00e+00 0.000e+00 0.00e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.00e+00 1.000e+00 1.00e+00 1.000e+00
```

```
##                              PC95      PC96      PC97      PC98      PC99
## Standard deviation     5.946e-06 5.478e-06 4.826e-06 4.521e-06 4.164e-06
## Proportion of Variance 0.000e+00 0.000e+00 0.000e+00 0.000e+00 0.000e+00
## Cumulative Proportion  1.000e+00 1.000e+00 1.000e+00 1.000e+00 1.000e+00
##                             PC100
## Standard deviation     4.122e-06
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

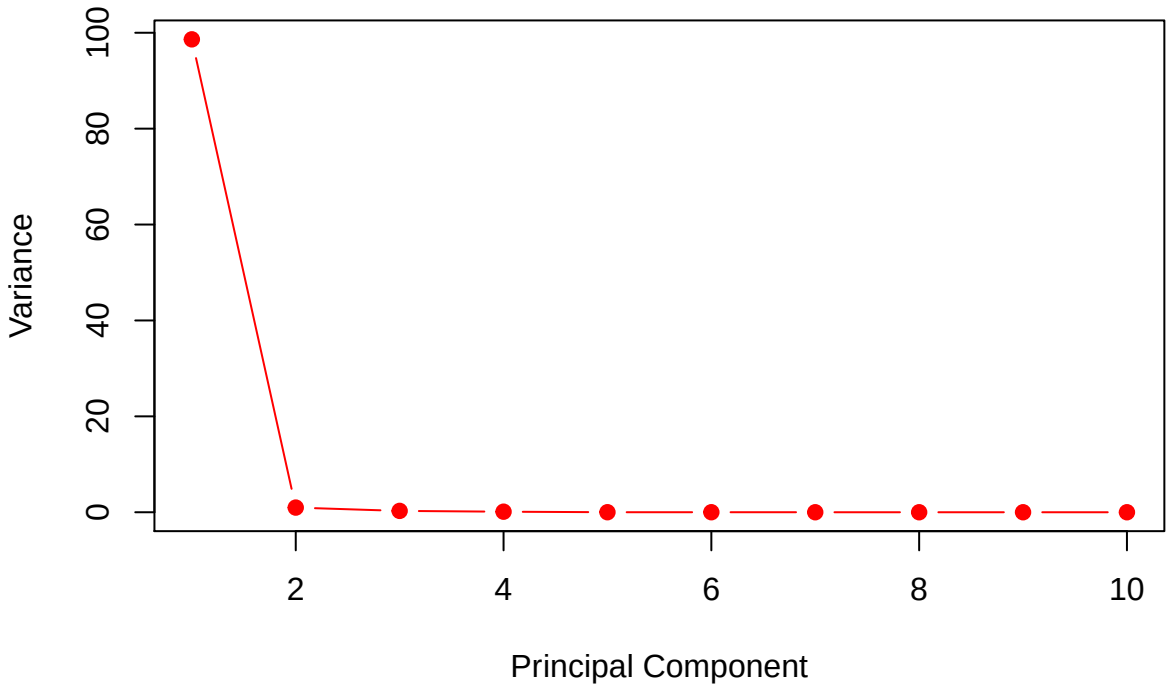
```
### The first few components contain nearly all of the variation in the data!

### Let's look at the "screeplot":

screeplot(meat_pca)
```




```
component = 1:(n_col-1)
variance  = (meat_pca$sdev)^2
plot(variance ~ component, xlab = "Principal Component", ylab = "Variance", type = "b", pch = 19, col =
```




```
plot(variance[1:10] ~ component[1:10], xlab = "Principal Component", ylab = "Variance", type = "b", pch
```




```
### Seems like just 2 PC comps are needed
### Now, let's do PCR = Principal Component Regression.
### We'll use another package that does PCR much easier.

# install.packages("pls")
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:corrplot':
##
##     corrplot

## The following object is masked from 'package:stats':
##
##     loadings
```

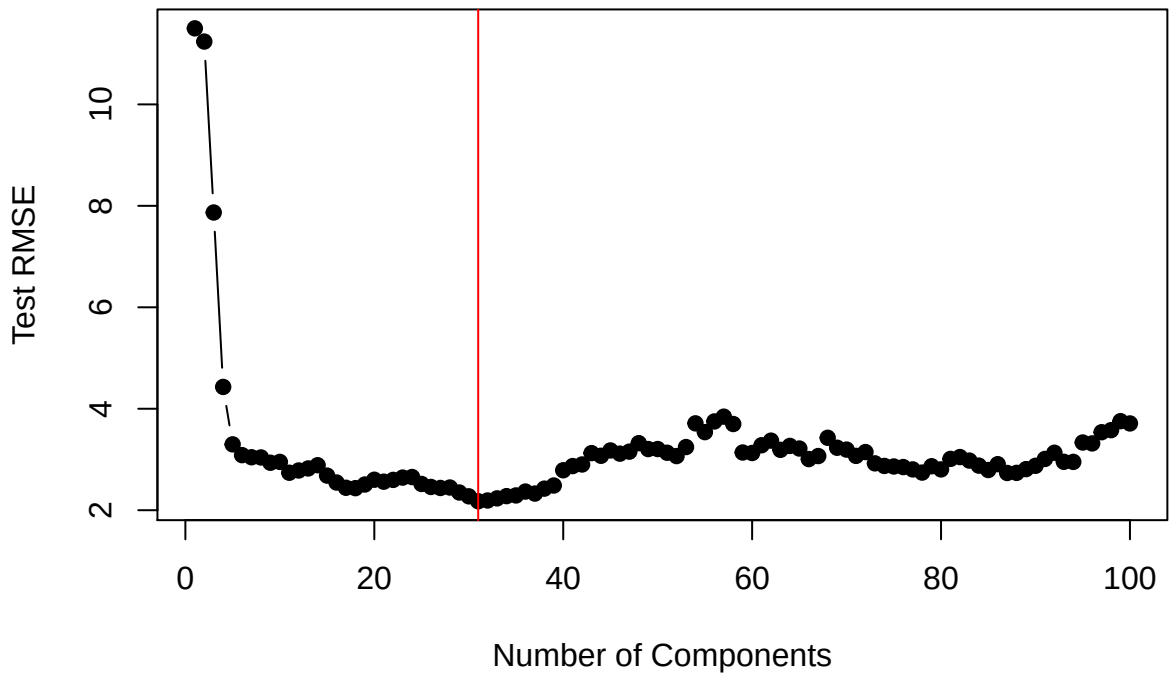
```
### The default for scaling is FALSE so let's change it to TRUE.
### The default number of folds is 10 and we'll leave that alone.

set.seed(123)
pcr_model = pcr(fat ~ ., data = meatspec, scale = TRUE, validation = "CV", ncomp = n_col - 1)
pcr_cv = RMSEP(pcr_model, estimate = "CV")

pcr_cv
```

```
## (Intercept)      1 comps      2 comps      3 comps      4 comps      5 comps
##      12.770       11.499       11.238        7.869        4.430        3.298
##     6 comps      7 comps      8 comps      9 comps     10 comps     11 comps
##       3.086        3.044        3.040        2.936        2.952        2.738
##    12 comps     13 comps     14 comps     15 comps     16 comps     17 comps
##       2.783        2.823        2.887        2.682        2.546        2.442
##    18 comps     19 comps     20 comps     21 comps     22 comps     23 comps
##       2.435        2.507        2.601        2.562        2.597        2.643
##    24 comps     25 comps     26 comps     27 comps     28 comps     29 comps
##       2.654        2.517        2.458        2.438        2.447        2.349
##    30 comps     31 comps     32 comps     33 comps     34 comps     35 comps
##       2.274        2.177        2.192        2.233        2.277        2.291
##    36 comps     37 comps     38 comps     39 comps     40 comps     41 comps
##       2.367        2.328        2.425        2.487        2.787        2.870
##    42 comps     43 comps     44 comps     45 comps     46 comps     47 comps
##       2.902        3.125        3.073        3.178        3.115        3.154
##    48 comps     49 comps     50 comps     51 comps     52 comps     53 comps
##       3.321        3.205        3.206        3.133        3.067        3.245
##    54 comps     55 comps     56 comps     57 comps     58 comps     59 comps
##       3.712        3.539        3.749        3.844        3.699        3.137
##    60 comps     61 comps     62 comps     63 comps     64 comps     65 comps
##       3.128        3.281        3.371        3.190        3.268        3.216
##    66 comps     67 comps     68 comps     69 comps     70 comps     71 comps
##       3.008        3.067        3.428        3.231        3.195        3.071
##    72 comps     73 comps     74 comps     75 comps     76 comps     77 comps
##       3.146        2.923        2.873        2.862        2.850        2.804
##    78 comps     79 comps     80 comps     81 comps     82 comps     83 comps
##       2.742        2.867        2.804        3.011        3.048        2.978
##    84 comps     85 comps     86 comps     87 comps     88 comps     89 comps
##       2.879        2.792        2.907        2.734        2.736        2.810
##    90 comps     91 comps     92 comps     93 comps     94 comps     95 comps
##       2.875        3.010        3.132        2.954        2.952        3.334
##    96 comps     97 comps     98 comps     99 comps    100 comps
##       3.317        3.538        3.578        3.755        3.710
```

```
### Let's plot it.  The [-1] leaves out the intercept only model.

plot(pcr_cv$val[-1], pch = 19, type = "b", ylab = "Test RMSE", xlab = "Number of Components")

best_comp = which.min(pcr_cv$val[-1])
best_comp
```

```
## [1] 31
```

```
abline(v = best_comp, col = "red")
```




```
pcr_cv$val[ best_comp ]
```

```
## [1] 2.273886
```

```
### Is this a good test rmse?
### Just a rule of thumb: compute coefficient of variation.
### This measures the % error relative to the mean of fat:

mean(meatspec[,n_col])
```

```
## [1] 18.14233
```

```
sd(meatspec[,n_col])
```

```
## [1] 12.7403
```

```
pcr_cv$val[ best_comp ] / mean(meatspec[,n_col])
```

```
## [1] 0.125336
```

```r
### Suppose you get a new set (5) of observations.  How would you predict the fat content on these?
### We can use the predict function but the variable names need to match.
### For the sake of easiness, let's take the first 5 rows to be "new" observation:
###
### newdata = meatspec[1:5,-n_col]
###
###
### What would be the predicted fat content?

predict(pcr_model, newdata = meatspec[1:5,-n_col],ncomp = best_comp)
```

```
## , , 31 comps
##
##         fat
## 1 20.089688
## 2 38.148254
## 3  9.708717
## 4  4.914735
## 5 27.684892
```

```r
meatspec[1:5, n_col]
```

```
## [1] 22.5 40.1  8.4  5.9 25.5
```