PCA and K-Means Clustering

Milica Cudina

We consider the **seeds** data set. This data set contains measurements of seeds. First, we import the data set.

```
seeds<-read.csv("seeds_dataset.csv")
seeds
```

```
##    X15.26 X14.84 X0.871 X5.763 X3.312 X2.221 X5.22
## 1   14.88  14.57 0.8811  5.554  3.333  1.018 4.956
## 2   14.29  14.09 0.9050  5.291  3.337  2.699 4.825
## 3   13.84  13.94 0.8955  5.324  3.379  2.259 4.805
## 4   16.14  14.99 0.9034  5.658  3.562  1.355 5.175
## 5   14.38  14.21 0.8951  5.386  3.312  2.462 4.956
## 6   14.69  14.49 0.8799  5.563  3.259  3.586 5.219
## 7   14.11  14.10 0.8911  5.420  3.302  2.700    NA
## 8   16.63  15.46 0.8747  6.053  3.465  2.040 5.877
## 9   16.44  15.25 0.8880  5.884  3.505  1.969 5.533
## 10  15.26  14.85 0.8696  5.714  3.242  4.543 5.314
## 11  14.03  14.16 0.8796  5.438  3.201  1.717 5.001
## 12  13.89  14.02 0.8880  5.439  3.199  3.986 4.738
## 13  13.78  14.06 0.8759  5.479  3.156  3.136 4.872
## 14  13.74  14.05 0.8744  5.482  3.114  2.932 4.825
##  [ reached 'max' / getOption("max.print") -- omitted 195 rows ]
```

```
dim(seeds)
```

```
## [1] 209   7
```

We immediately see that there are missing data points. I will choose to omit those rows in the future analysis.

```
seeds=na.omit(seeds)
dim(seeds)
```

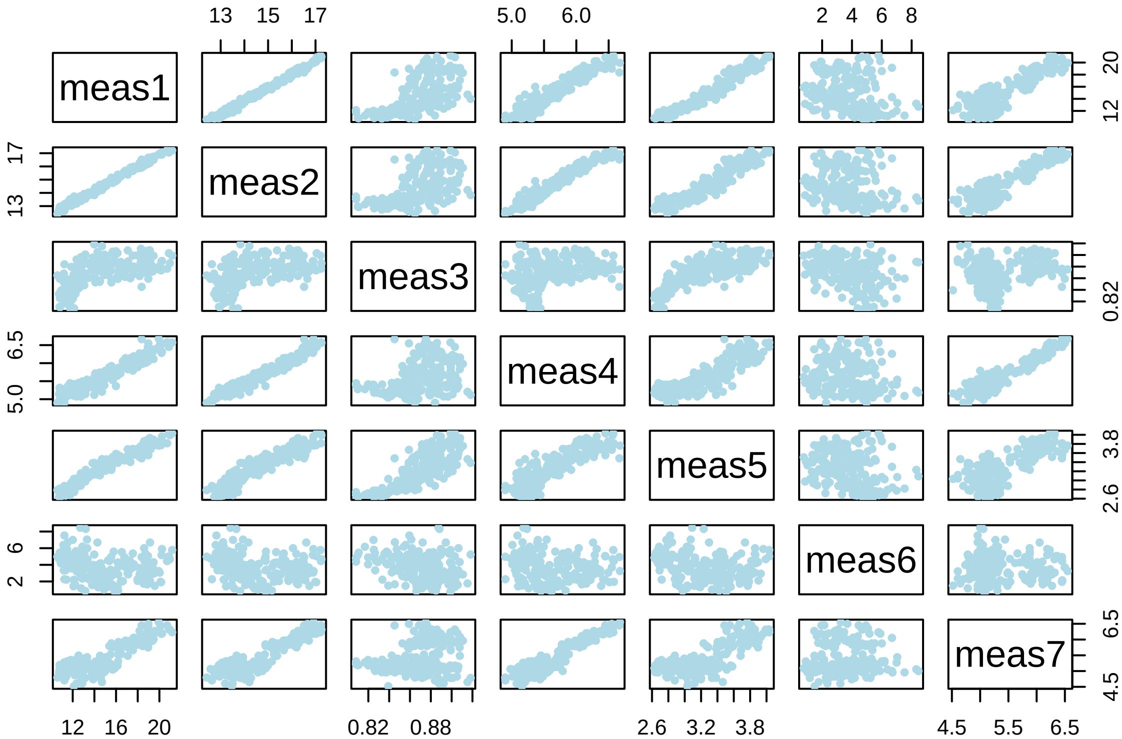
```
## [1] 202   7
```

The columns are not labeled, so I will add names for aesthetic reasons.

```
colnames(seeds)=c("meas1", "meas2", "meas3", "meas4", "meas5", "meas6", "meas7")
attach(seeds)
```

Here is the usual EDA.

```
plot(seeds, col="lightblue", pch=20)
```



We do recognize the strong relationship between some measurements. We could perform K-means clustering on the entire data set.

```r
km.out <- kmeans(seeds, 2, nstart = 20)
km.out
```

```
## K-means clustering with 2 clusters of sizes 127, 75
##
## Cluster means:
##      meas1    meas2     meas3    meas4    meas5    meas6    meas7
## 1 12.94039 13.70441 0.8629654 5.352441 3.023173 3.893568 5.099126
## 2 18.23107 16.08280 0.8844160 6.126080 3.671987 3.420161 5.965347
##
## Clustering vector:
##   1   2   3   4   5   6   8   9  10  11  12  13  14  15  16  17  18  19  20  21
##   1   1   1   2   1   1   2   2   1   1   1   1   1   1   1   2   1   1   1   1
##  22  23  24  25  26  27  28  29  30  31  32  33  34  36  37  38  39  40  41  42
##   2   1   1   2   1   1   1   1   1   1   1   1   1   2   2   1   1   1   1   1
##  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  61  62  63
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83
##   1   1   1   1   1   1   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  [ reached getOption("max.print") -- omitted 102 entries ]
##
## Within cluster sum of squares by cluster:
## [1] 641.9963 318.6342
##  (between_SS / total_SS =  63.6 %)
##
## Available components:
```

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

```
pre.clustering=km.out$cluster
```

However, visualization is challenging in a 7−dimensional space.

So, we want to start with PCA for this data set to reduce the dimension. Let's import the requisite library.

```
library(stats)
```

Let's look at the principal components analysis.

```
pr.out=prcomp(seeds,scale=TRUE)
pr.out$center
```

```
##      meas1      meas2      meas3      meas4      meas5      meas6      meas7
## 14.9047525 14.5874752  0.8709297  5.6396832  3.2640693  3.7177980  5.4207426
```

```
pr.out$scale
```

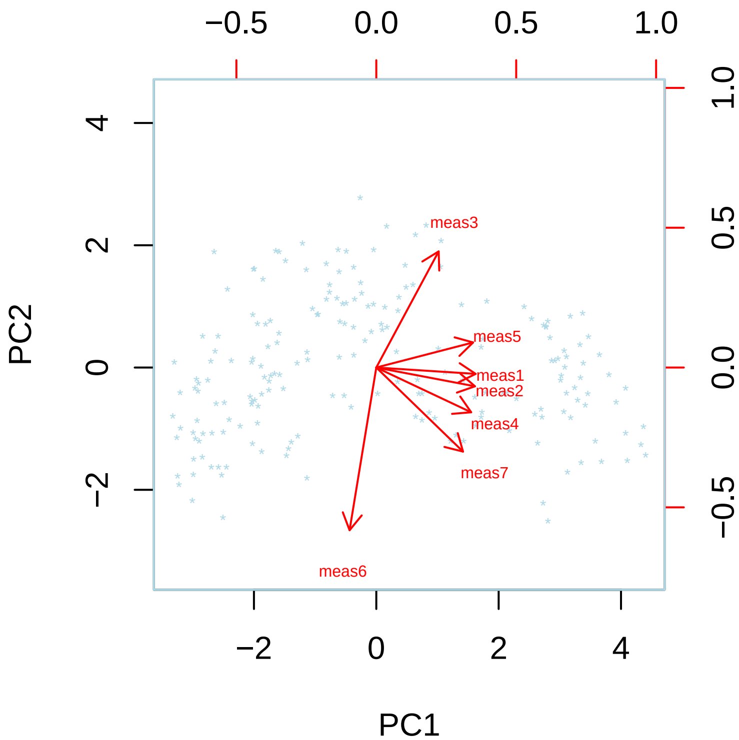
```
##     meas1     meas2     meas3     meas4     meas5     meas6     meas7
## 2.9270018 1.3129940 0.0233935 0.4445863 0.3793446 1.5045096 0.4937631
```

```
pr.out$rotation
```

```
##              PC1         PC2         PC3        PC4        PC5         PC6
## meas1  0.4438209 -0.02837450  0.02522182 -0.1997502  0.1959546  0.42707585
## meas2  0.4409472 -0.08383445 -0.06056566 -0.3028393  0.1611726  0.47786506
## meas3  0.2785985  0.51914024  0.63899850  0.3405375 -0.3227615  0.13865938
## meas4  0.4238398 -0.19999038 -0.21246198 -0.2464757 -0.7690757 -0.28425740
## meas5  0.4324202  0.11208856  0.21551095 -0.2064352  0.4714107 -0.69824260
## meas6 -0.1193594 -0.72725351  0.66831944 -0.0918059 -0.0383849  0.01704236
## meas7  0.3874240 -0.37576910 -0.22167690  0.8003926  0.1275587 -0.03727380
##               PC7
## meas1  0.735451989
## meas2 -0.669944665
## meas3 -0.072041702
## meas4  0.046835025
## meas5 -0.040220518
## meas6 -0.003552126
## meas7 -0.035646645
```

What does the biplot tell us?

```
biplot(pr.out,scale=0, cex=0.5, xlabs=rep("*", length(meas1)),
       col=c("lightblue", "red"))
```

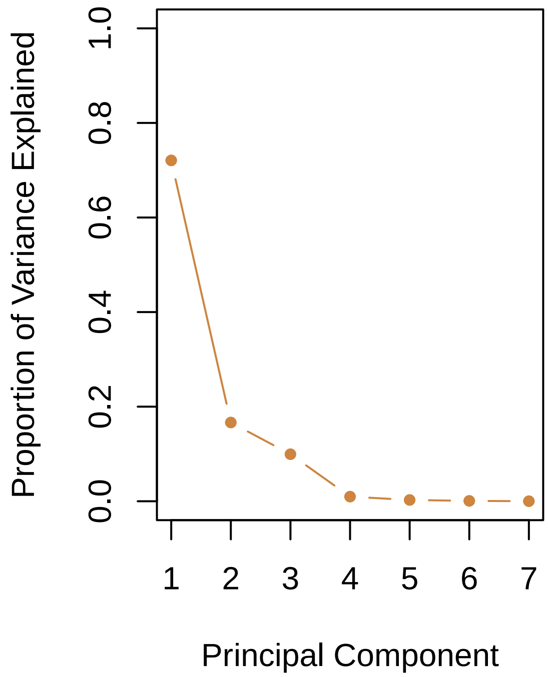


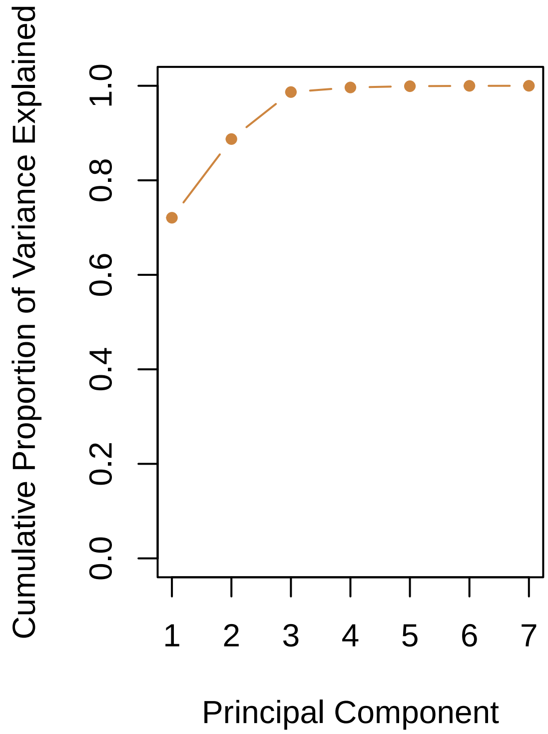
Let's look at the variance explained.

```
pr.var=pr.out$sdev^2
pve=pr.var/sum(pr.var)
pve
```

```
## [1] 0.7207429722 0.1665285024 0.0994654055 0.0097229829 0.0026723103
## [6] 0.0007529615 0.0001148653
```

```
par(mfrow = c(1, 2))
plot(pve,xlab="Principal Component",
     ylab="Proportion of Variance Explained", col="peru",
     pch=20, ylim=c(0,1),type='b')
plot(cumsum(pve),xlab="Principal Component",
     ylab="Cumulative Proportion of Variance Explained",
     col="peru", pch=20, ylim=c(0,1),type='b')
```





The values corresponding to our rows in terms of principal components are available through `pr.out$x`.

```
pr.out$x
```

```
##             PC1         PC2          PC3          PC4           PC5
## 1   -0.04207784  1.94468075 -0.632127428 -0.424848174  0.0385772091
## 2   -0.49042490  1.91804621  0.971251764 -0.097261002 -0.0063964294
## 3   -0.62362012  1.94663985  0.536415581 -0.216966866  0.0773032846
## 4    1.06088821  2.09183935  0.100261549 -0.130928172  0.0194619872
## 5   -0.37072106  1.65455973  0.472378239 -0.087443426 -0.0045658223
## 6   -0.18519061  0.45760937  0.313464401 -0.105994075 -0.0725094802
## 8    1.71369945  0.34847250 -0.955896688  0.239391431 -0.1340264789
## 9    1.39295855  1.04282786 -0.358166002  0.013032605 -0.1010270808
## 10   0.02266578 -0.40731929  0.321063526 -0.356726069 -0.1302589989
## 11  -0.60772047  1.58685848 -0.390767934 -0.127676023 -0.0175404493
## 12  -0.96362326  0.88588011  0.968306548 -0.527798816 -0.2900418275
## 13  -0.94927834  0.89407142  0.153716027 -0.435350237 -0.1918784475
## 14  -1.04227573  0.98242904  0.018047184 -0.494695701 -0.2394084500
## 15  -0.54878468  1.06352121  1.458281375 -0.437619311 -0.0423886542
##                PC6            PC7
## 1   -0.01728263413 -0.00503191898
## 2    0.05331998378 -0.00460471997
## 3   -0.22512186236 -0.01037567716
## 4   -0.04911991102 -0.00134456779
## 5    0.02414801380 -0.00897167289
## 6    0.05845540204 -0.02451329546
## 8   -0.09592381419 -0.03005666416
## 9   -0.06165135197 -0.00864637487
## 10   0.15000942325 -0.02466840104
## 11   0.02223897884 -0.00791191004
## 12   0.04924536861  0.01642390353
```

```
## 13   0.00986276075  0.00674361030
## 14   0.06812257211  0.01505867892
## 15   0.12161224997 -0.00220489260
##  [ reached getOption("max.print") -- omitted 188 rows ]
```

We can isolate the first two principal components as follows:

```
pc1=pr.out$x[,1]
pc2=pr.out$x[,2]
```

Let's look at the scatterplot of these two vectors.

```
plot(pc1, pc2, col="grey",pch=20)
```