# 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
```

```
##       V1    V2     V3    V4    V5    V6    V7
## 1  15.26 14.84 0.8710 5.763 3.312 2.221 5.220
## 2  14.88 14.57 0.8811 5.554 3.333 1.018 4.956
## 3  14.29 14.09 0.9050 5.291 3.337 2.699 4.825
## 4  13.84 13.94 0.8955 5.324 3.379 2.259 4.805
## 5  16.14 14.99 0.9034 5.658 3.562 1.355 5.175
## 6  14.38 14.21 0.8951 5.386 3.312 2.462 4.956
## 7  14.69 14.49 0.8799 5.563 3.259 3.586 5.219
## 8  14.11 14.10 0.8911 5.420 3.302 2.700    NA
## 9  16.63 15.46 0.8747 6.053 3.465 2.040 5.877
## 10 16.44 15.25 0.8880 5.884 3.505 1.969 5.533
## 11 15.26 14.85 0.8696 5.714 3.242 4.543 5.314
## 12 14.03 14.16 0.8796 5.438 3.201 1.717 5.001
## 13 13.89 14.02 0.8880 5.439 3.199 3.986 4.738
## 14 13.78 14.06 0.8759 5.479 3.156 3.136 4.872
##  [ reached 'max' / getOption("max.print") -- omitted 196 rows ]
```

```
dim(seeds)
```

```
## [1] 210   7
```

```
#View(seeds)
```

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)
```

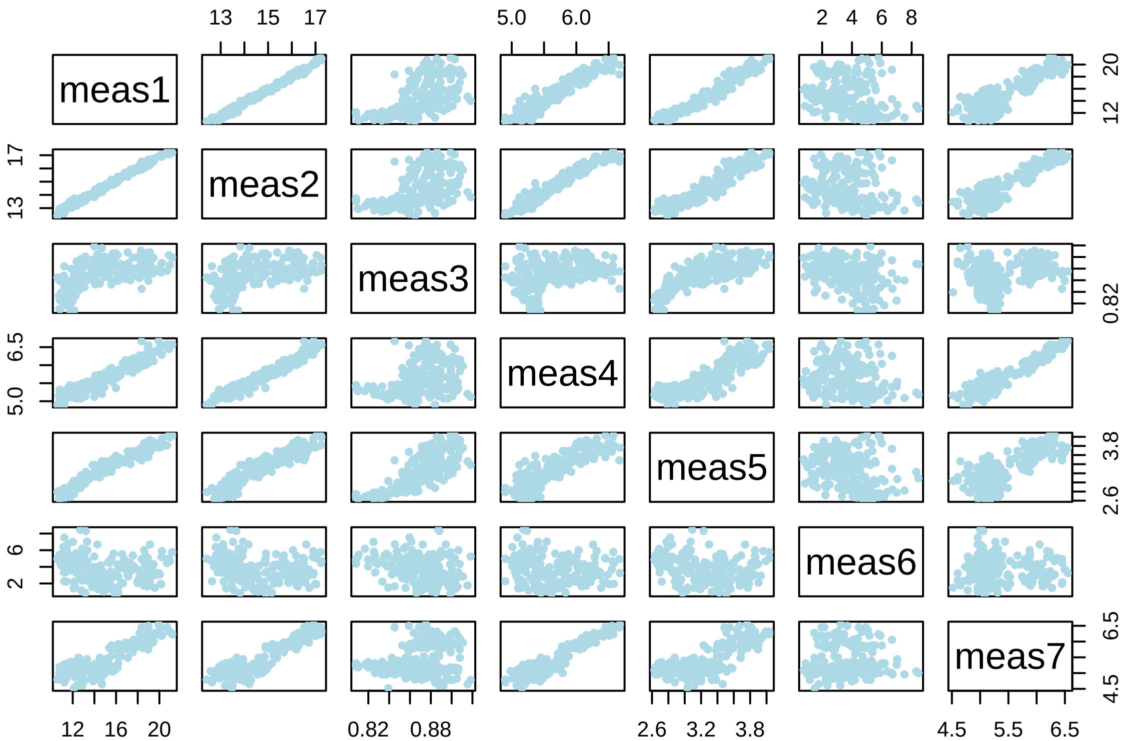
```
## [1] 203   7
```

I don't like how the columns are labeled, so I will change names for aesthetic reasons.

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

Here is the usual EDA.

```r
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 75, 128
##
## Cluster means:
##      meas1    meas2     meas3    meas4    meas5    meas6    meas7
## 1 18.23107 16.08280 0.8844160 6.126080 3.671987 3.420161 5.965347
## 2 12.95852 13.71328 0.8630281 5.355648 3.025430 3.880501 5.100070
##
## Clustering vector:
##   1   2   3   4   5   6   7   9  10  11  12  13  14  15  16  17  18  19  20  21
##   2   2   2   2   1   2   2   1   1   2   2   2   2   2   2   2   1   2   2   2
##  22  23  24  25  26  27  28  29  30  31  32  33  34  35  37  38  39  40  41  42
##   2   1   2   2   1   2   2   2   2   2   2   2   2   2   1   1   2   2   2   2
##  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  62  63
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83
##   2   2   2   2   2   2   2   1   1   1   1   1   1   1   1   1   1   1   1   1
##  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  [ reached 'max' / getOption("max.print") -- omitted 103 entries ]
##
## Within cluster sum of squares by cluster:
## [1] 318.6342 651.6545
##  (between_SS / total_SS =  63.3 %)
```

```
##
## 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
## 14.906502 14.588719  0.870930  5.640291  3.264305  3.710425  5.419754
```

```
pr.out$scale
```

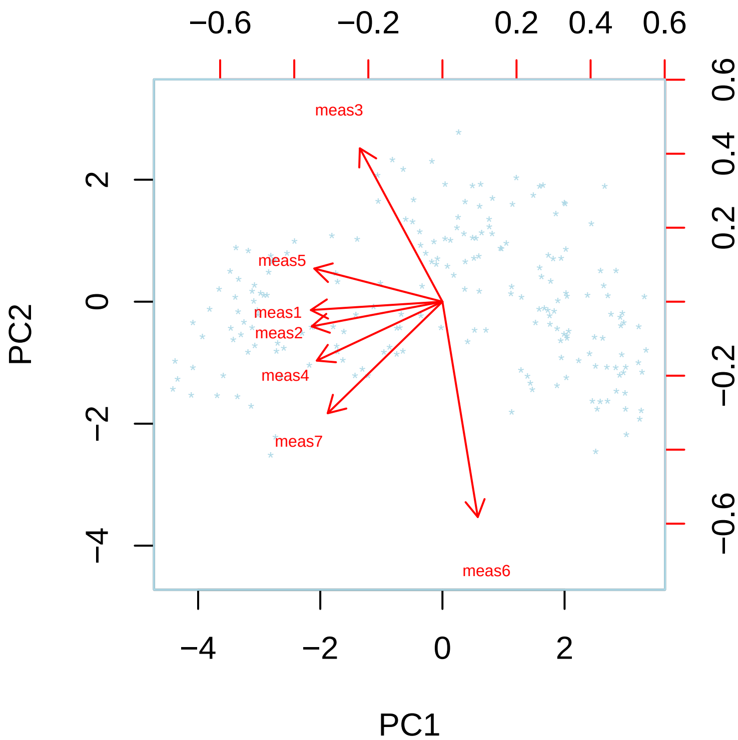
```
##      meas1      meas2      meas3      meas4      meas5      meas6      meas7
## 2.91985421 1.30985992 0.02333552 0.44356887 0.37841943 1.50445337 0.49274090
```

```
pr.out$rotation
```

```
##              PC1         PC2         PC3         PC4         PC5         PC6
## meas1 -0.4438797 -0.02827866  0.02487893  0.19823234 -0.19754734 -0.42733099
## meas2 -0.4409974 -0.08341264 -0.06128012  0.30190710 -0.16267355 -0.47768044
## meas3 -0.2786483  0.51826746  0.63993258 -0.33862828  0.32426621 -0.13864936
## meas4 -0.4238644 -0.19909129 -0.21365717  0.24983603  0.76819351  0.28331567
## meas5 -0.4324862  0.11204425  0.21527972  0.20512206 -0.47158921  0.69855470
## meas6  0.1192982 -0.72757429  0.66757050  0.09462566  0.03866974 -0.01705868
## meas7 -0.3871819 -0.37694281 -0.22015213 -0.80089937 -0.12389829  0.03805960
##                PC7
## meas1 -0.735268248
## meas2  0.670088948
## meas3  0.072103502
## meas4 -0.047383128
## meas5  0.040078258
## meas6  0.003587394
## meas7  0.036030449
```

What does the biplot tell us?

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

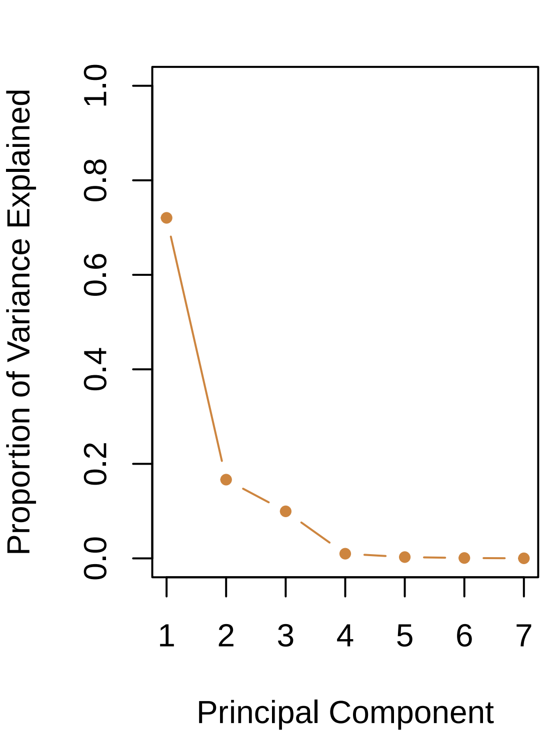


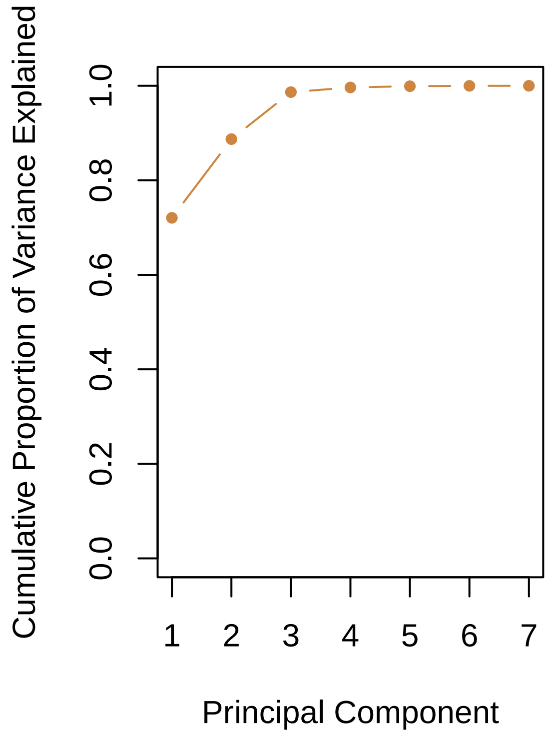

Let's look at the variance explained.

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

```
## [1] 0.7205610767 0.1665472057 0.0995076775 0.0098352503 0.0026805176
## [6] 0.0007532880 0.0001149841
```

```
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   -0.27208727  0.81428921 -0.61045254  0.4068678200  0.1108691927
## 2    0.04374462  1.94324843 -0.62732194  0.4193787646 -0.0422075100
## 3    0.49263170  1.91682186  0.97891526  0.0945618791  0.0051184569
## 4    0.62627162  1.94545680  0.54327189  0.2134810496 -0.0792880095
## 5   -1.06194841  2.09018176  0.10717166  0.1260659263 -0.0214127618
## 6    0.37282678  1.65279631  0.47890409  0.0841153996  0.0032071271
## 7    0.18684029  0.45446902  0.31745265  0.1044517240  0.0715895980
## 9   -1.71585861  0.34378543 -0.95282241 -0.2439069377  0.1343082487
## 10  -1.39463443  1.03957434 -0.35353894 -0.0170437368  0.1002000159
## 11  -0.02164777 -0.41103381  0.32280885  0.3572058770  0.1285946808
## 12   0.61068607  1.58478719 -0.38567551  0.1228737973  0.0156321674
## 13   0.96671193  0.88444950  0.97274446  0.5288777755  0.2878919266
## 14   0.95267769  0.89211163  0.15734662  0.4341716260  0.1895459811
## 15   1.04592550  0.98065151  0.02147881  0.4934043039  0.2367611943
##               PC6            PC7
## 1    0.0240907211  0.01353314270
## 2    0.0166815256  0.00468348379
## 3   -0.0537045502  0.00449527985
## 4    0.2253545574  0.01013462950
## 5    0.0488641611  0.00112007051
## 6   -0.0244820342  0.00884895351
## 7   -0.0588647805  0.02445094078
## 9    0.0959633857  0.02998744531
## 10   0.0614608024  0.00846338822
## 11  -0.1508217703  0.02455405863
## 12  -0.0226973071  0.00771664945
```

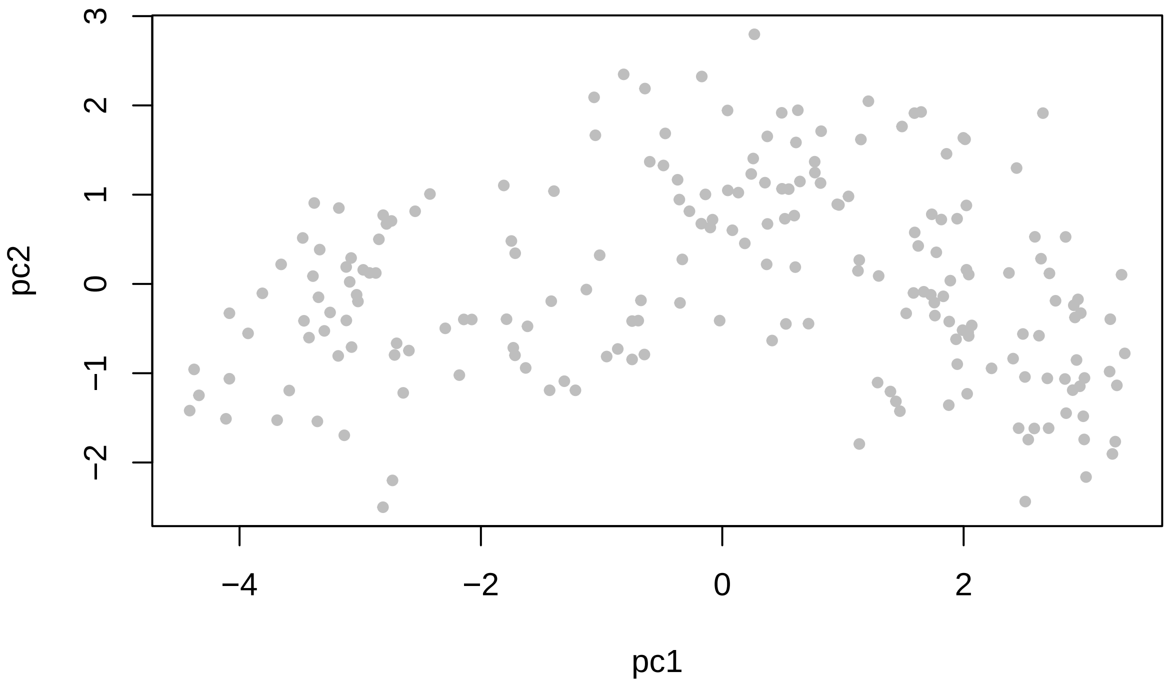
```
## 13  -0.0501568583 -0.01683378752
## 14  -0.0105925230 -0.00711376945
## 15  -0.0691114386 -0.01548348958
##  [ reached 'max' / getOption("max.print") -- omitted 189 rows ]
```

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)
```




Now, let's perform $K-$means clustering to these data based on the first and second principal components.

```
pcs=cbind(pc1, pc2)
km.out=kmeans(pcs, 3, nstart=25)
```

Let's compare the clusters based on the entire data set and the first two principal components.

```
sum(km.out$cluster==pre.clustering)/length(pre.clustering)
```

```
## [1] 0.6305419
```

```
km.out$cluster
```

```
##   1   2   3   4   5   6   7   9  10  11  12  13  14  15  16  17  18  19  20  21
##   3   3   3   3   3   3   3   1   3   3   3   3   3   3   3   3   3   3   2   3
##  22  23  24  25  26  27  28  29  30  31  32  33  34  35  37  38  39  40  41  42
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   1   3   3   3   3
##  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  62  63
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
##  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83
##   3   3   3   3   3   3   2   1   1   1   1   1   1   1   1   1   1   1   1   1
##  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

```
##   [ reached 'max' / getOption("max.print") -- omitted 103 entries ]
```

```r
pre.clustering
```

```
##    1   2   3   4   5   6   7   9  10  11  12  13  14  15  16  17  18  19  20  21
##    2   2   2   2   1   2   2   1   1   2   2   2   2   2   2   2   1   2   2   2
##   22  23  24  25  26  27  28  29  30  31  32  33  34  35  37  38  39  40  41  42
##    2   1   2   2   1   2   2   2   2   2   2   2   2   2   1   1   2   2   2   2
##   43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  62  63
##    2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
##   64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83
##    2   2   2   2   2   2   2   1   1   1   1   1   1   1   1   1   1   1   1   1
##   84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102 103
##    1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##   [ reached 'max' / getOption("max.print") -- omitted 103 entries ]
```

What about the scatterplot?

```r
plot(pcs, col = (km.out$cluster + 1),
    main = "K-Means Clustering Results with K = 2",
    xlab = "", ylab = "", pch = 20, cex = 1)
```