Week 6: Solutions of the exercises

Anikó Lovik

2024-03-15

Use the following libraries: ISLR2, nFactors, GPArotation, psych, EFA.dimensions

Exercise 1. College dataset

In this exercise you will perform dimension reduction on the College dataset from the ISLR2 package. This dataset contains the number of applications, type of institution (private/public) and other characteristics of colleges in the US (for a description of the variables see ISLR2, pp. 54-55).

a) Check the sample size and variables of the dataset. Check for missing values, for this exercise, remove them if necessary. Remove the variables 'Private' and 'Apps'.

```r
str(College)
```

```
## 'data.frame':    777 obs. of  18 variables:
##  $ Private    : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Apps       : num  1660 2186 1428 417 193 ...
##  $ Accept     : num  1232 1924 1097 349 146 ...
##  $ Enroll     : num  721 512 336 137 55 158 103 489 227 172 ...
##  $ Top10perc  : num  23 16 22 60 16 38 17 37 30 21 ...
##  $ Top25perc  : num  52 29 50 89 44 62 45 68 63 44 ...
##  $ F.Undergrad: num  2885 2683 1036 510 249 ...
##  $ P.Undergrad: num  537 1227 99 63 869 ...
##  $ Outstate   : num  7440 12280 11250 12960 7560 ...
##  $ Room.Board : num  3300 6450 3750 5450 4120 ...
##  $ Books      : num  450 750 400 450 800 500 500 450 300 660 ...
##  $ Personal   : num  2200 1500 1165 875 1500 ...
##  $ PhD        : num  70 29 53 92 76 67 90 89 79 40 ...
##  $ Terminal   : num  78 30 66 97 72 73 93 100 84 41 ...
##  $ S.F.Ratio  : num  18.1 12.2 12.9 7.7 11.9 9.4 11.5 13.7 11.3 11.5 ...
##  $ perc.alumni: num  12 16 30 37 2 11 26 37 23 15 ...
##  $ Expend     : num  7041 10527 8735 19016 10922 ...
##  $ Grad.Rate  : num  60 56 54 59 15 55 63 73 80 52 ...
```

```r
data_College = College[,-c(1,2)]
```

b) Run a PCA with and without scaling (normalisation) and check the explained variance per component. Do you expect any differences? Why yes/no?

```r
# centered (default) and normalised variables
pca_result_sc = prcomp(data_College, scale=TRUE)
# not normalised/scaled
pca_result_unsc = prcomp(data_College, scale=FALSE)
```

```r
# Explained variance per component - normalised
summary(pca_result_sc) # gives output for the variance explained
```

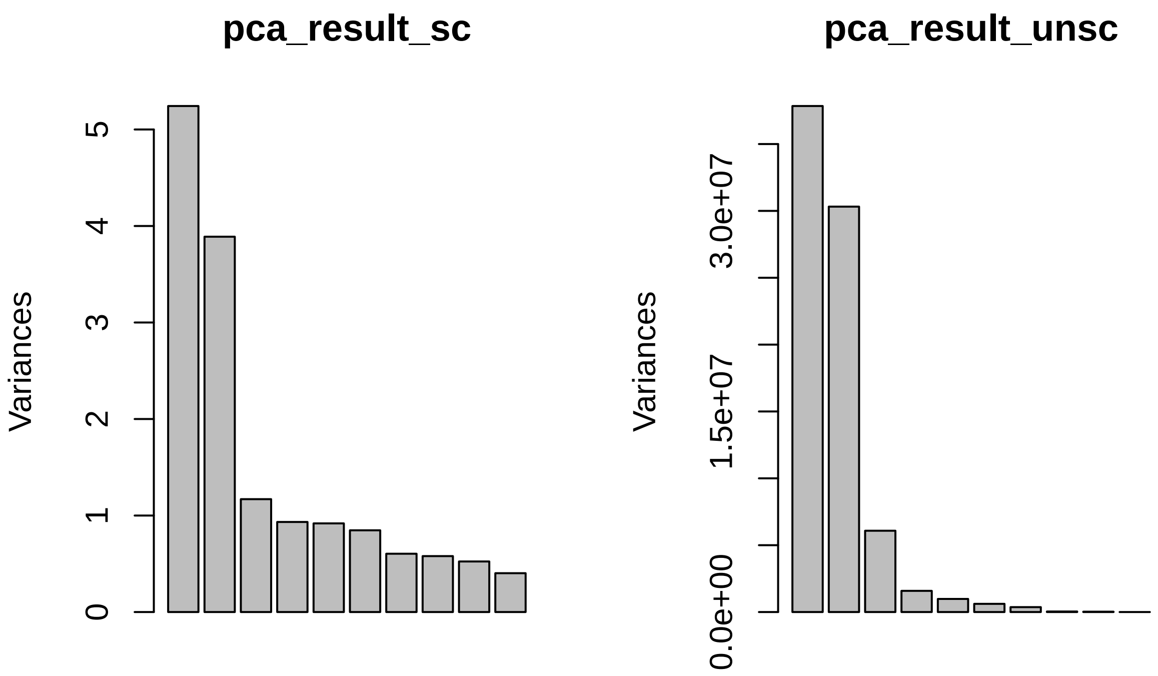
```
## Importance of components:
##                           PC1    PC2     PC3     PC4    PC5     PC6     PC7
## Standard deviation     2.2898 1.9721 1.08153 0.96602 0.9584 0.92026 0.77723
## Proportion of Variance 0.3277 0.2431 0.07311 0.05832 0.0574 0.05293 0.03776
## Cumulative Proportion  0.3277 0.5707 0.64386 0.70219 0.7596 0.81252 0.85028
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     0.76132 0.72389 0.63446 0.55903 0.46932 0.37991 0.32171
## Proportion of Variance 0.03623 0.03275 0.02516 0.01953 0.01377 0.00902 0.00647
## Cumulative Proportion  0.88650 0.91925 0.94441 0.96394 0.97771 0.98673 0.99320
##                           PC15    PC16
## Standard deviation     0.28463 0.16674
## Proportion of Variance 0.00506 0.00174
## Cumulative Proportion  0.99826 1.00000
```

```r
# Explained variance per component - not normalised
summary(pca_result_unsc)
```

```
## Importance of components:
##                              PC1       PC2       PC3       PC4      PC5
## Standard deviation     6151.7492 5506.3030 2465.8686 1.259e+03 990.3943
## Proportion of Variance    0.4861    0.3894    0.0781 2.035e-02   0.0126
## Cumulative Proportion     0.4861    0.8755    0.9536 9.739e-01   0.9865
##                              PC6       PC7       PC8       PC9     PC10  PC11
## Standard deviation     783.31965 606.58942 200.45559 159.16467 21.27253 14.84
## Proportion of Variance   0.00788   0.00473   0.00052   0.00033  0.00001  0.00
## Cumulative Proportion    0.99442   0.99915   0.99966   0.99999  0.99999  1.00
##                         PC12  PC13  PC14  PC15  PC16
## Standard deviation     12.56 9.004 6.037 5.326 2.911
## Proportion of Variance  0.00 0.000 0.000 0.000 0.000
## Cumulative Proportion   1.00 1.000 1.000 1.000 1.000
```

There is quite some difference, which is expected given the different measurement units (and ranges) of the variables. To make it clearer, we can also plot it:

```r
# Explained variance per component plotted
par(mfrow=c(1,2))
plot(pca_result_sc)
plot(pca_result_unsc)
```



c) Decide on how many components to keep using Kaiser's rule, the cumulative PVE and a scree plot for both (you can also add the average eigenvalue-rule, the MAP test and parallel analysis, if you want). Do you see a difference between the methods? Do you see a difference between scaled and unscaled PCA results?

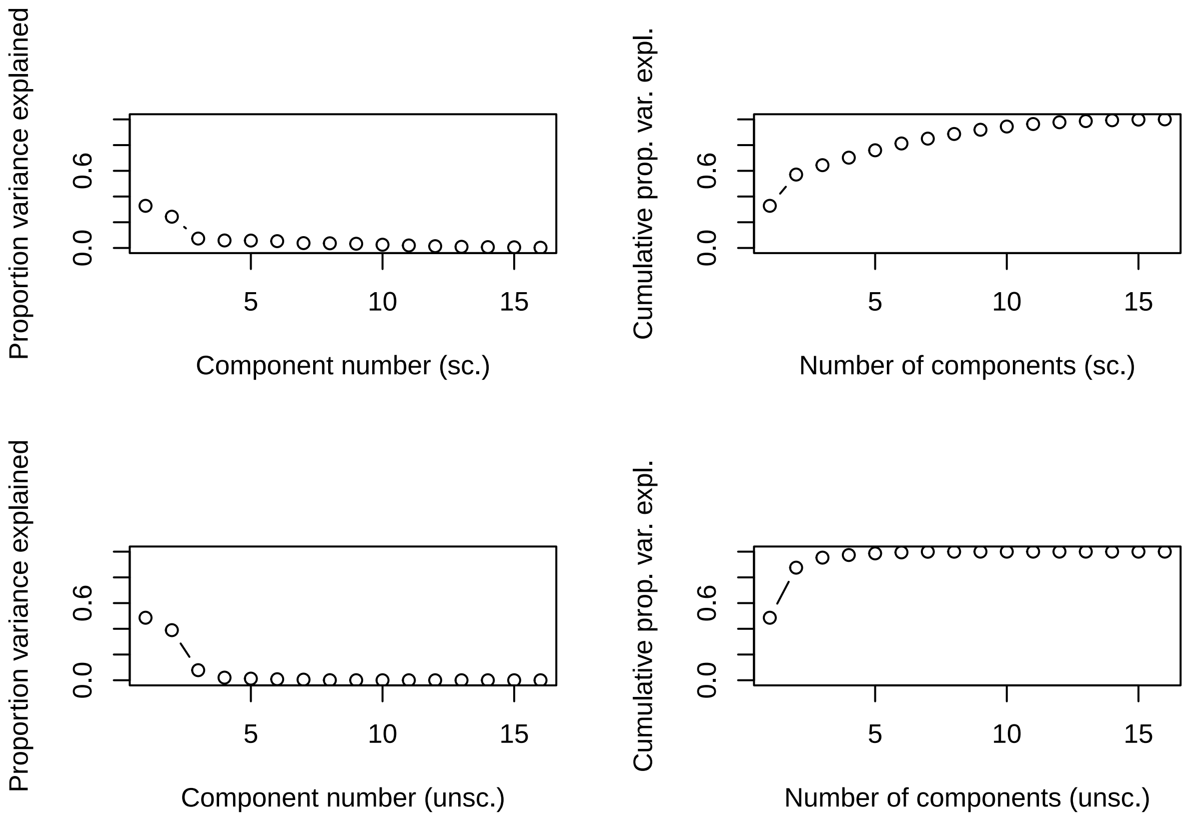
For Kaiser's rule, we need the eigenvalues and use the cut-off at 1. Therefore, we would keep three components. We can use Kaiser's rule as an upper-bound, so we should extract up to three components (at least based on the scaled data).

```
ev <- eigen(cor(data_College))
ev$values
```

```
##  [1] 5.24305902 3.88900943 1.16971040 0.93319643 0.91847416 0.84687786
##  [7] 0.60409041 0.57960844 0.52401052 0.40254377 0.31251497 0.22025905
## [13] 0.14432852 0.10349737 0.08101683 0.02780282
```

Scree plot and cumulative proportion of variance explained for both sclaed and unscaled results:

```
# We first save PVE and cumulative PVE from the summary
prop_var_expl_sc = summary(pca_result_sc)$importance[2,]
cumul_prop_var_expl_sc = summary(pca_result_sc)$importance[3,]
prop_var_expl_unsc = summary(pca_result_unsc)$importance[2,]
cumul_prop_var_expl_unsc = summary(pca_result_unsc)$importance[3,]
# scree plot
par(mfrow=c(2,2))
plot(prop_var_expl_sc, type="b", xlab="Component number (sc.)",
     ylab="Proportion variance explained", ylim=c(0,1) )
plot(cumul_prop_var_expl_sc, type="b", xlab="Number of components (sc.)",
     ylab="Cumulative prop. var. expl.", ylim=c(0,1) )
plot(prop_var_expl_unsc, type="b", xlab="Component number (unsc.)",
     ylab="Proportion variance explained", ylim=c(0,1) )
plot(cumul_prop_var_expl_unsc, type="b", xlab="Number of components (unsc.)",
     ylab="Cumulative prop. var. expl.", ylim=c(0,1) )
```



To get an idea about the “elbow” in the scree plot

```
cumul_prop_var_expl <- cumul_prop_var_expl_sc
nElem=length(cumul_prop_var_expl_sc)
ratios=(cumul_prop_var_expl[2:(nElem-1)]-cumul_prop_var_expl[1:(nElem-2)])/
  (cumul_prop_var_expl[3:nElem]-cumul_prop_var_expl[2:(nElem-1)])
ratios
```

```
##      PC2      PC3      PC4      PC5      PC6      PC7      PC8      PC9
## 3.324579 1.253386 1.016202 1.084451 1.401748 1.042518 1.105954 1.301669
##     PC10     PC11     PC12     PC13     PC14     PC15
## 1.288274 1.418301 1.526608 1.394127 1.278656 2.908046
```

```
nComponents_selected = which.max( ratios ) + 1
nComponents_selected
```

```
## PC2
##   2
```

So, based on the scree plot we would probably select 2 components. However, running this for the unscaled results, we get something different:

```
cumul_prop_var_expl <- cumul_prop_var_expl_unsc
nElem=length(cumul_prop_var_expl_sc)
ratios=(cumul_prop_var_expl[2:(nElem-1)]-cumul_prop_var_expl[1:(nElem-2)])/
  (cumul_prop_var_expl[3:nElem]-cumul_prop_var_expl[2:(nElem-1)])
ratios
```

```
##      PC2      PC3      PC4      PC5      PC6      PC7      PC8      PC9
## 4.986172 3.837838 1.615079 1.598985 1.665962 9.274510 1.545455      Inf
##     PC10     PC11     PC12     PC13     PC14     PC15
## 0.000000      Inf      NaN      NaN      NaN      NaN
```

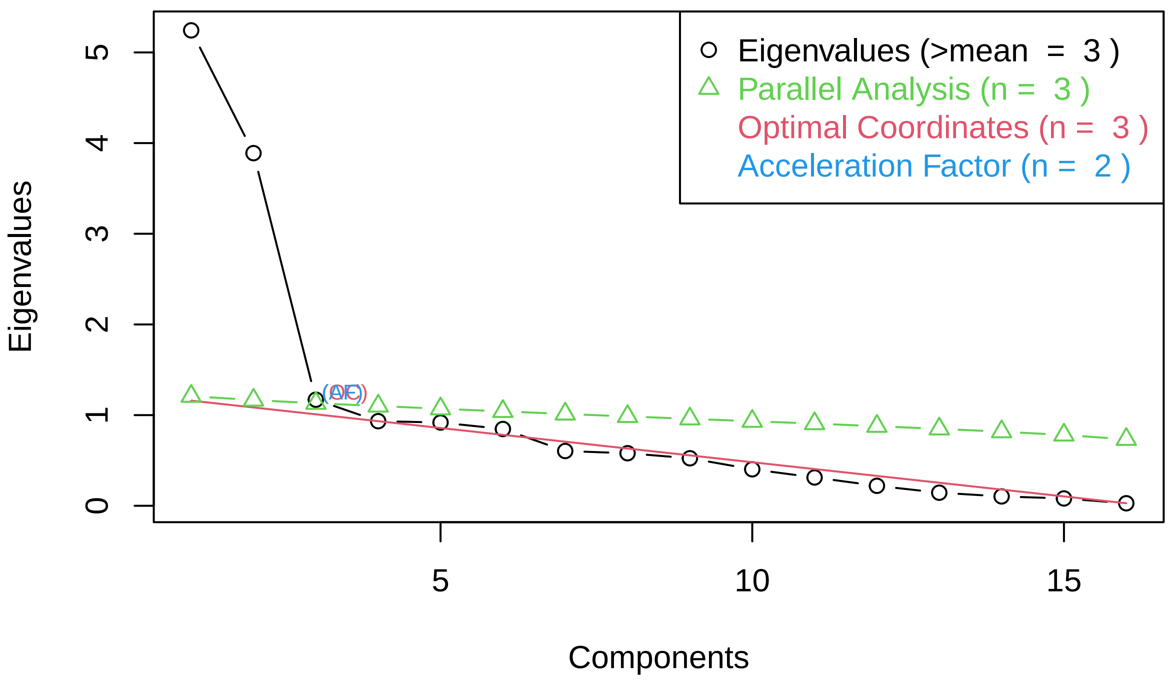
```
nComponents_selected = which.max( ratios ) + 1
nComponents_selected
```

```
## PC9
##   9
```

We can conclude that the scaled and unscaled results can indeed be very different.

Let's keep the scaled results and also look at the other methods to decide on the number of components. First, here is Horn's parallel analysis for the scaled version (the one with the random samples):

```
#parallel analysis
ev <- eigen(cor(data_College))
ap <- parallel(subject=nrow(data_College), var=ncol(data_College), rep=1000)
nS <- nScree(x=ev$values, aparallel=ap$eigen$qevpea)
plotnScree(nS,main="")
```



```
#cbind(ev$values, ap$eigen[,3])
```

Velicer's MAP test based on partial correlations gives different results for the original (2) and revised (5) test:

```
#Velicer's MAP
MAP(data_College, corkind='pearson', verbose=TRUE)
```

```
##
##
## MINIMUM AVERAGE PARTIAL (MAP) TEST
##
## Number of cases = 777
##
## Number of variables = 16
##
## Specified kind of correlations for this analysis: Pearson
```

```
##
##
## Total Variance Explained (Initial Eigenvalues):
##             Eigenvalues   Proportion of Variance   Cumulative Prop. Variance
## Factor 1           5.24                     0.33                        0.33
## Factor 2           3.89                     0.24                        0.57
## Factor 3           1.17                     0.07                        0.64
## Factor 4           0.93                     0.06                        0.70
## Factor 5           0.92                     0.06                        0.76
## Factor 6           0.85                     0.05                        0.81
## Factor 7           0.60                     0.04                        0.85
## Factor 8           0.58                     0.04                        0.89
## Factor 9           0.52                     0.03                        0.92
## Factor 10          0.40                     0.03                        0.94
## Factor 11          0.31                     0.02                        0.96
## Factor 12          0.22                     0.01                        0.98
## Factor 13          0.14                     0.01                        0.99
## Factor 14          0.10                     0.01                        0.99
## Factor 15          0.08                     0.01                        1.00
## Factor 16          0.03                     0.00                        1.00
##
## Velicer's Average Squared Correlations
##    root   Avg.Corr.Sq.   Avg.Corr.power4
##       0        0.13224           0.04920
##       1        0.08619           0.02959
##       2        0.05245           0.00969
##       3        0.06080           0.01264
##       4        0.07008           0.01275
##       5        0.05547           0.00786
##       6        0.07176           0.01359
##       7        0.09492           0.02476
##       8        0.13004           0.04370
##       9        0.15490           0.08708
##      10        0.16713           0.08862
##      11        0.22729           0.14243
##      12        0.28754           0.18299
##      13        0.41485           0.31210
##      14        0.65933           0.55287
##      15        1.00000           1.00000
##
## The smallest average squared correlation is 0.05245
##
## The smallest average 4rth power correlation is 0.00786
##
## The number of components according to the original (1976) MAP Test is = 2
##
## The number of components according to the revised (2000) MAP Test is = 5
```

Finally, we could have the Very Simple Structure method where the main focus is interpretability:

```r
#Revelle and and Rocklin's very simple structure
fit = vss(data_College, n = 15, rotate="oblimin", diagonal = FALSE, fm = "pc")
fit
```

In sum, the different methods lead to different number of components, but 2 and 3 components are supported by more than one method each. Let's have a look at both.

d) Which variables load high on each component? Can you give an interpretation of the components?

```r
# Loadings
round(pca_result_sc$rotation[,1:2], 3)
```

```
##                PC1    PC2
## Accept       0.094  0.437
## Enroll       0.058  0.471
## Top10perc    0.368  0.043
## Top25perc    0.348  0.082
## F.Undergrad  0.034  0.480
## P.Undergrad -0.059  0.344
## Outstate     0.355 -0.154
## Room.Board   0.279 -0.057
## Books        0.049  0.086
## Personal    -0.099  0.222
## PhD          0.297  0.189
## Terminal     0.299  0.177
## S.F.Ratio   -0.240  0.193
## perc.alumni  0.268 -0.179
## Expend       0.346 -0.025
## Grad.Rate    0.290 -0.092
```

```r
round(pca_result_sc$rotation[,1:3], 3)
```

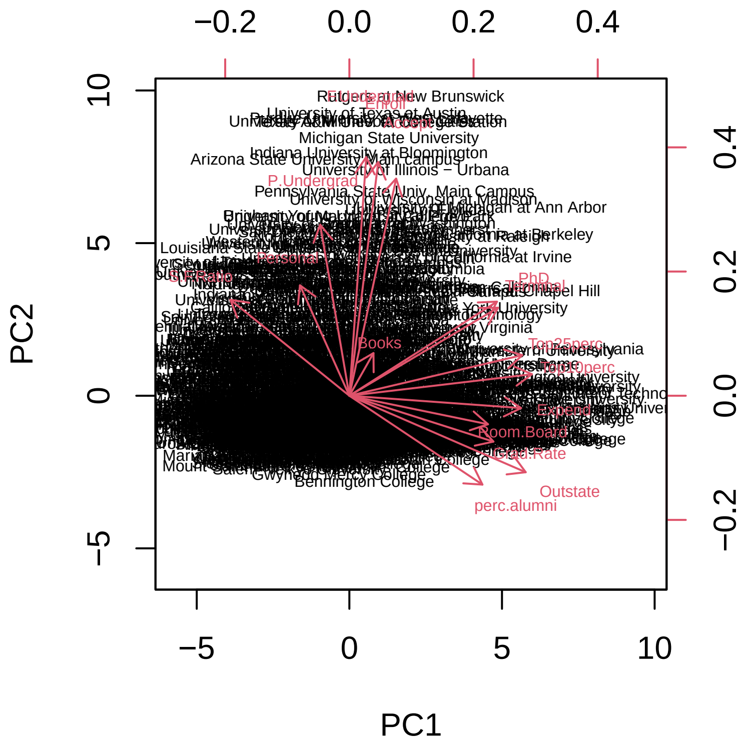
```
##                PC1    PC2    PC3
## Accept       0.094  0.437 -0.088
## Enroll       0.058  0.471 -0.083
## Top10perc    0.368  0.043  0.027
## Top25perc    0.348  0.082 -0.039
## F.Undergrad  0.034  0.480 -0.066
## P.Undergrad -0.059  0.344  0.116
## Outstate     0.355 -0.154  0.058
## Room.Board   0.279 -0.057  0.163
## Books        0.049  0.086  0.679
## Personal    -0.099  0.222  0.477
## PhD          0.297  0.189 -0.175
## Terminal     0.299  0.177 -0.113
## S.F.Ratio   -0.240  0.193 -0.306
## perc.alumni  0.268 -0.179 -0.145
## Expend       0.346 -0.025  0.232
## Grad.Rate    0.290 -0.092 -0.189
```

Note, that if we extract three components, the first two will not change (unless we rotate the components, but we will come back to this later).

e) Create a bi-plot for the first two components.

```r
# bi-plot (first two principal components)
par(mfrow=c(1,1))
```

```r
biplot(pca_result_sc, scale=0 , cex=.5) #cex to change fond of the figure
```



f) Rotate the selected components using an orthogonal rotation. (There are many ways to do this, you could for example use the 'varimax' function in the EFA.dimensions package). Do you obtain a simple structure?

Here we rotate both the 2- and the 3-component versions using varimax:

```r
unrot2 = round(pca_result_sc$rotation[,1:2], 6)
varimax2 <- varimax(pca_result_sc$rotation[,1:2])
varimax2
```

```
## $loadings
##
## Loadings:
##             PC1    PC2
## Accept              0.446
## Enroll              0.474
## Top10perc    0.359
## Top25perc    0.334  0.126
## F.Undergrad         0.481
## P.Undergrad -0.103  0.334
## Outstate     0.372 -0.106
## Room.Board   0.284
## Books
## Personal    -0.128  0.207
## PhD          0.270  0.227
## Terminal     0.274  0.215
## S.F.Ratio   -0.263  0.160
## perc.alumni  0.289 -0.142
## Expend       0.346
## Grad.Rate    0.300
```

```
##
##                  PC1   PC2
## SS loadings    1.000 1.000
## Proportion Var 0.062 0.062
## Cumulative Var 0.062 0.125
##
## $rotmat
##            [,1]      [,2]
## [1,]  0.9914483 0.1304998
## [2,] -0.1304998 0.9914483
```

```
unrot3 = round(pca_result_sc$rotation[,1:3], 6)
varimax3 <- varimax(pca_result_sc$rotation[,1:3])
varimax3
```

```
## $loadings
##
## Loadings:
##             PC1    PC2    PC3
## Accept              0.456
## Enroll              0.478
## Top10perc    0.342  0.118
## Top25perc    0.294  0.165 -0.125
## F.Undergrad         0.478
## P.Undergrad         0.287  0.223
## Outstate     0.367        -0.109
## Room.Board   0.325
## Books        0.289         0.619
## Personal                   0.526
## PhD          0.182  0.286 -0.200
## Terminal     0.209  0.261 -0.148
## S.F.Ratio   -0.360  0.197 -0.141
## perc.alumni  0.214        -0.270
## Expend       0.408
## Grad.Rate    0.207        -0.292
##
##                  PC1   PC2   PC3
## SS loadings    1.000 1.000 1.000
## Proportion Var 0.062 0.062 0.062
## Cumulative Var 0.062 0.125 0.187
##
## $rotmat
##            [,1]       [,2]       [,3]
## [1,]  0.9172453  0.2263970 -0.3277278
## [2,] -0.1304920  0.9481577  0.2897736
## [3,]  0.3763415 -0.2230276  0.8992362
```

Note that the rotated component loadings do differ for the two PCAs. Moreover, the loadings are generally low and that several variables load on all components. In such cases, one can opt to drop variables that do not contribute to any of the components.

Exercise 2. Perform principal component analysis on the heptathlon dataset.

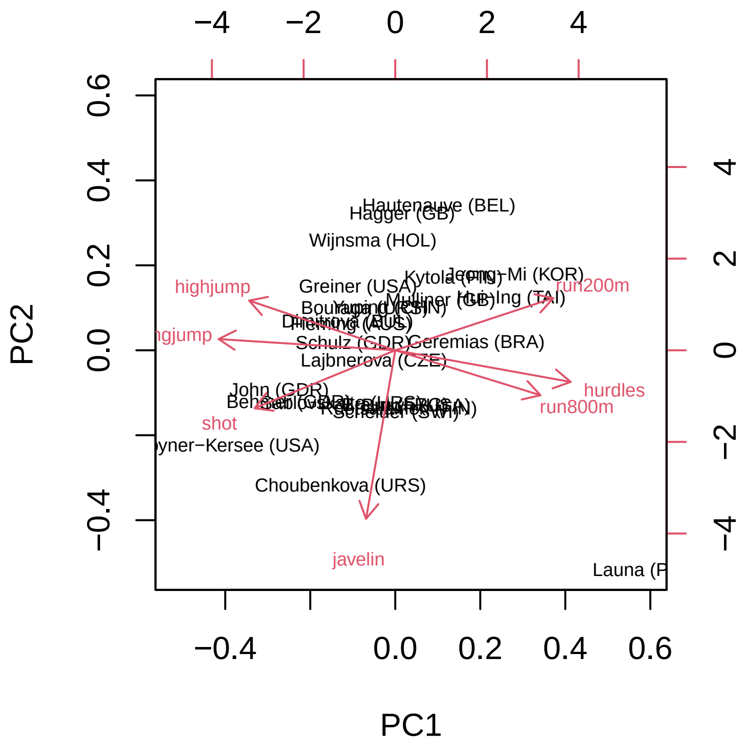
The heptathlon dataset contains data from 25 individuals on the seven disciplines that form the sport. Run a PCA with the normalised data. Decide on how many components to keep using at least three different methods. How much variance do the extracted components explain? Which variables load high on each component, can you give an interpretation of the components? Do the components become easier to interpret if you rotate the components?

```
library(HSAUR)
```

```
## Loading required package: tools
```

```
data_hepthathlon <- heptathlon[,-8]

hep.PC = prcomp(heptathlon[,-8], center = TRUE, scale=TRUE)
biplot(hep.PC, cex=0.55)
```



```
ev <- eigen(cor(data_hepthathlon))
ev$values
```

```
## [1] 4.46027516 1.19432056 0.52101413 0.45716683 0.24526674 0.07295558 0.04900101
```

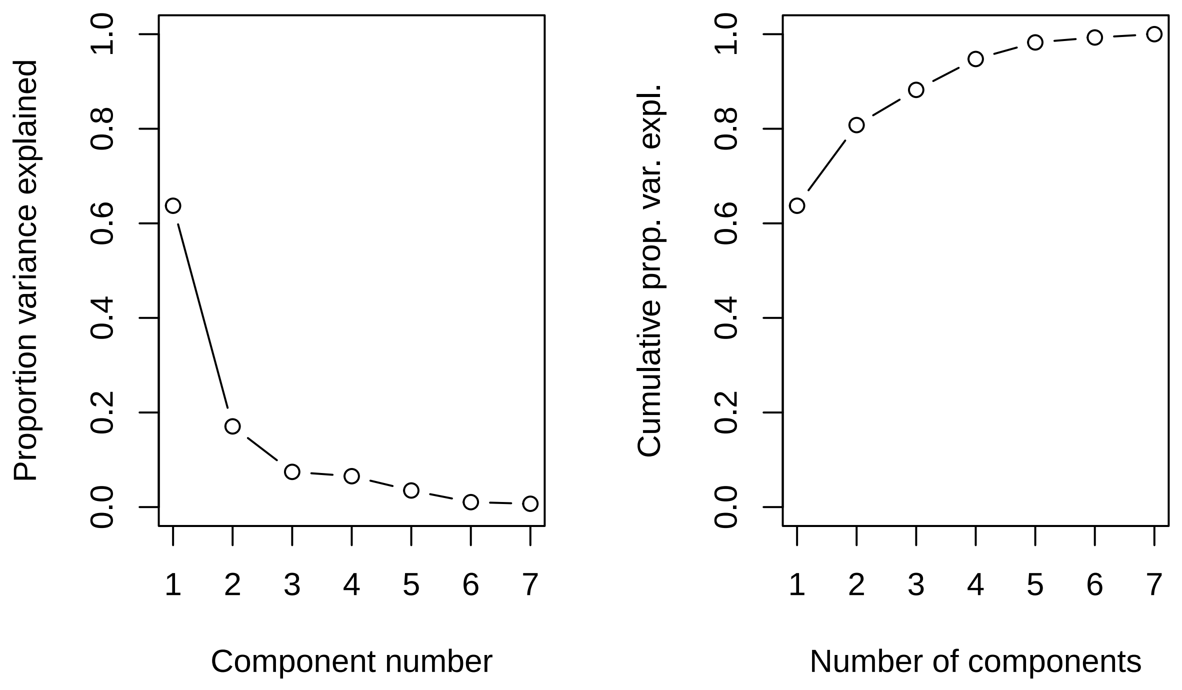
```
hep.prop.var = summary(hep.PC)$importance[2,]
hep.prop.var
```

```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7
## 0.63718 0.17062 0.07443 0.06531 0.03504 0.01042 0.00700
```

```
hep.cum.prop.var = summary(hep.PC)$importance[3,]
hep.cum.prop.var
```

```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7
## 0.63718 0.80780 0.88223 0.94754 0.98258 0.99300 1.00000
```

```
par(mfrow=c(1,2))
plot(hep.prop.var, type="b", xlab="Component number", ylab="Proportion variance explained", ylim=c(0,1)
plot(hep.cum.prop.var, type="b", xlab="Number of components", ylab="Cumulative prop. var. expl.", ylim=
```



```
nElem=length(hep.cum.prop.var)
ratios=(hep.cum.prop.var[2:(nElem-1)]-hep.cum.prop.var[1:(nElem-2)])/(hep.cum.prop.var[3:nElem]-hep.cum
ratios
```

```
##      PC2      PC3      PC4      PC5      PC6
## 2.292355 1.139642 1.863870 3.362764 1.488571
```

```
nComponents_selected=which.max(ratios)+1
nComponents_selected
```

```
## PC5
##   5
```

```
MAP(data_hepthathlon, corkind='pearson', verbose=TRUE)
```

```
##
##
## MINIMUM AVERAGE PARTIAL (MAP) TEST

##

## Number of cases = 25

##

## Number of variables = 7

##

## Specified kind of correlations for this analysis: Pearson

##

##

## Total Variance Explained (Initial Eigenvalues):

##            Eigenvalues   Proportion of Variance   Cumulative Prop. Variance
## Factor 1          4.46                     0.64                        0.64
## Factor 2          1.19                     0.17                        0.81
## Factor 3          0.52                     0.07                        0.88
## Factor 4          0.46                     0.07                        0.95
## Factor 5           0.25                      0.04                        0.98
## Factor 6           0.07                      0.01                        0.99
## Factor 7           0.05                      0.01                        1.00
```

```
##
## Velicer's Average Squared Correlations
```

```
##    root   Avg.Corr.Sq.   Avg.Corr.power4
##       0        0.35402           0.19166
##       1        0.10242           0.02154
##       2        0.12381           0.03505
##       3        0.21293           0.11828
##       4        0.30273           0.19741
##       5        0.48438           0.34550
##       6        1.00000           1.00000
```

```
##
## The smallest average squared correlation is 0.10242
```

```
##
## The smallest average 4rth power correlation is 0.02154
```

```
##
## The number of components according to the original (1976) MAP Test is = 1
```

```
##
## The number of components according to the revised (2000) MAP Test is = 1
```

Despite having only seven variables, the different methods give rather different solutions for the number of components. Kaiser's rule selects 2, the cut-off at the average of the eigenvalues 1, the MAP-test 1, the scree plot 2 or 5, etc. Since 2 components explain 81% and 3 components 88% of the variance and there are only seven variables, keeping more than three would not make much sense.

Let's look at the results with two and three components. Since performance in these disciplines is likely correlated, we employ the oblique oblimin rotation:

```
unrot2 = round(hep.PC$rotation[,1:2], 3)
unrot2
```

```
##             PC1    PC2
## hurdles   0.453 -0.158
## highjump -0.377  0.248
## shot     -0.363 -0.289
## run200m   0.408  0.260
## longjump -0.456  0.056
## javelin  -0.075 -0.842
## run800m   0.375 -0.224
```

```
oblimin2 <- oblimin(hep.PC$rotation[,1:2])
oblimin2
```

```
## Oblique rotation method Oblimin Quartimin converged.
## Loadings:
##              PC1     PC2
## hurdles   0.4702  0.0807
## highjump -0.4073 -0.1822
## shot     -0.3212  0.3458
## run200m   0.3695 -0.3246
## longjump -0.4598  0.0205
## javelin   0.0381  0.8430
## run800m   0.4019  0.1593
##
##                  PC1   PC2
## SS loadings    1.000 1.000
## Proportion Var 0.143 0.143
## Cumulative Var 0.143 0.286
##
## Phi:
##        PC1    PC2
## PC1 1.0000 0.0321
## PC2 0.0321 1.0000
```

```
unrot3 = round(hep.PC$rotation[,1:3], 3)
unrot3
```

```
##             PC1    PC2    PC3
## hurdles   0.453 -0.158  0.045
## highjump -0.377  0.248 -0.368
## shot     -0.363 -0.289  0.676
## run200m   0.408  0.260 -0.084
## longjump -0.456  0.056  0.139
## javelin  -0.075 -0.842 -0.472
## run800m   0.375 -0.224  0.396
```

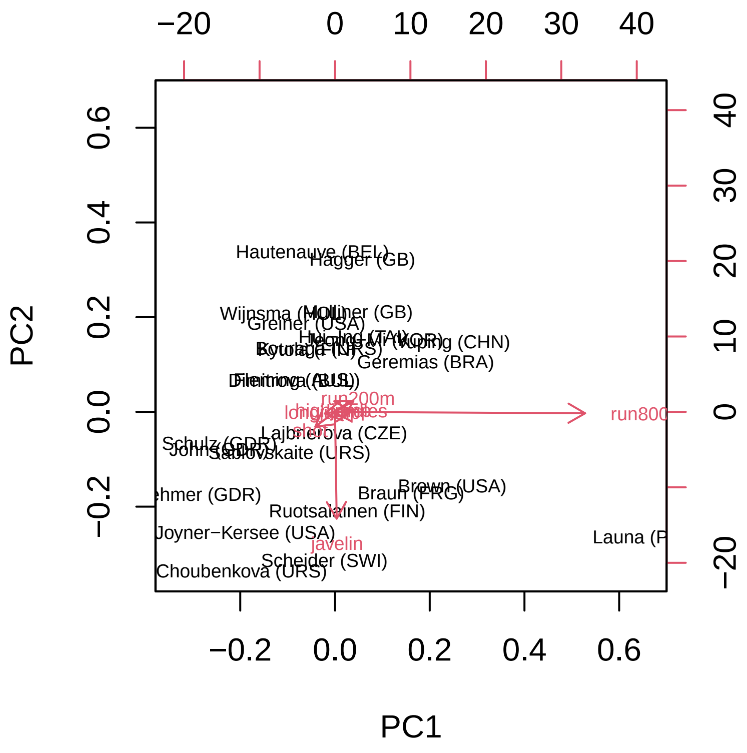
```
oblimin3 <- oblimin(hep.PC$rotation[,1:3])
oblimin3
```

```
## Oblique rotation method Oblimin Quartimin converged.
## Loadings:
##               PC1     PC2      PC3
## hurdles   0.45452  0.0729 -0.18521
## highjump -0.55795 -0.0110 -0.12547
## shot      0.07627 -0.0208  0.80920
## run200m   0.22494 -0.2319 -0.39670
## longjump -0.34511 -0.0678  0.35634
## javelin   0.00851  0.9676 -0.00392
## run800m   0.56039 -0.0226  0.13898
##
##                  PC1   PC2   PC3
## SS loadings    1.000 1.000 1.000
## Proportion Var 0.143 0.143 0.143
## Cumulative Var 0.143 0.286 0.429
##
## Phi:
##          PC1      PC2     PC3
## PC1  1.00000 -0.00727  0.0863
## PC2 -0.00727  1.00000 -0.0323
## PC3  0.08632 -0.03233  1.0000
```

Extra: We indeed obtain very different results (that do not make much sense). If you look carefully, the first component, which explains over 80% of the variance, consists only of the 800m run, the second component of javelin throwing, etc. This is because the variables are not scaled and the largest variance is observed in the variable(s) with the largest values and variances).

```
library(HSAUR)
data_hepthathlon <- heptathlon[,-8]
```

```r
hep.PC2 = prcomp(heptathlon[,-8], center = TRUE, scale=FALSE)
biplot(hep.PC2, cex=0.55)
```



```r
ev2 <- eigen(cor(data_hepthathlon))
ev2$values
```

```
## [1] 4.46027516 1.19432056 0.52101413 0.45716683 0.24526674 0.07295558 0.04900101
```

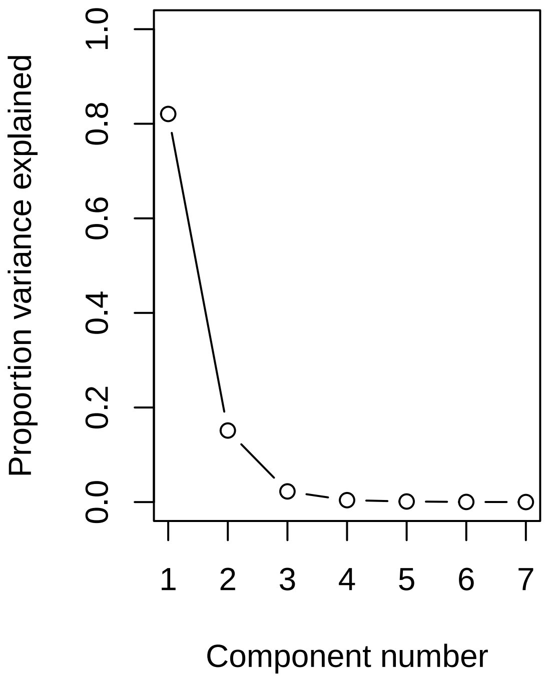
```r
hep.prop.var2 = summary(hep.PC2)$importance[2,]
hep.prop.var2
```

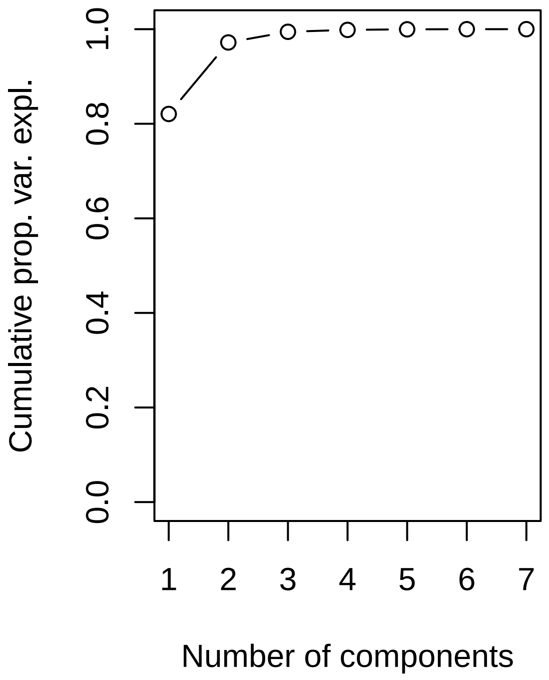
```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7
## 0.82070 0.15126 0.02252 0.00402 0.00123 0.00025 0.00001
```

```r
hep.cum.prop.var2 = summary(hep.PC2)$importance[3,]
hep.cum.prop.var2
```

```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7
## 0.82070 0.97196 0.99448 0.99850 0.99973 0.99999 1.00000
```

```r
par(mfrow=c(1,2))
plot(hep.prop.var2, type="b", xlab="Component number", ylab="Proportion variance explained", ylim=c(0,1
plot(hep.cum.prop.var2, type="b", xlab="Number of components", ylab="Cumulative prop. var. expl.", ylim
```





```r
unrot2 = round(hep.PC2$rotation[,1:7], 3)
unrot2
```

```
##             PC1    PC2    PC3    PC4    PC5    PC6    PC7
## hurdles   0.070  0.009  0.222  0.327 -0.807  0.425 -0.083
## highjump -0.006 -0.001 -0.015 -0.021  0.140  0.098 -0.985
## shot     -0.078 -0.136 -0.884  0.425 -0.104 -0.052 -0.016
## run200m   0.073  0.101  0.310  0.816  0.462  0.082  0.051
## longjump -0.040 -0.015 -0.185 -0.204  0.319  0.895  0.142
## javelin   0.007 -0.985  0.160  0.032  0.049  0.006  0.005
## run800m   0.991 -0.013 -0.117 -0.058  0.028 -0.003  0.001
```