Principal Component Analysis using R

November 25, 2009

This tutorial is designed to give the reader a short overview of *Principal Component Analysis* (PCA) using R. PCA is a useful statistical method that has found application in a variety of fields and is a common technique for finding patterns in data of high dimension. Consider we are confronted with the following situation:
The data, we want to work with, are in form of a matrix $(x_{ij})_{i=1...N,j=1...M}$, where $x_{i,j}$represents the value of the i-th observation of the j-th variable. Thus the N members of the population can be identified with the N rows of the data matrix, each correspondending to a M-dimensional vector denoting the values of the different variables. If M is very large it is often desireable to reduce the number of considered variables and to replace M by a smaller number of variables, while losing as little information as possible. Mathematically spoken, PCA is a linear orthogonal transformation, that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on .... The (Principal) Components are linear combinations of the original data and a formally description can be found in the book of Carmona (p. 84ff).

The data for this analysis are in the package **Rsafd**:

```
R> library("Rsafd")
```

In a first application we want to reproduce the example from Carmona (p. 87ff.). It concerns the market of fixed income securities. In particular, we use the data of the US yield curve provided by the bank of international settlement (BIS).

With

```
R> data(us.bis.yield)
```

we load the data into our workspace.

`us.bis.yield` gives for each trading day, starting at the 3rd of January 1995, the yields on the US Treasury bonds for different times to maturity (in months: $x = 0, 1, 2, 3, 4, 5, 5.5, 6.5, 7.5, 8.5, 9.5$).

The first six rows of our data set are

```
R> head(us.bis.yield)
```

```
     Col2    Col3    Col4    Col5    Col6    Col7    Col8    Col9   Col10
1 5.59957 7.22989 7.60820 7.72886 7.75856 7.74944 7.73191 7.71845 7.75699
2 5.63497 7.17471 7.51762 7.65834 7.69625 7.68582 7.67083 7.66179 7.69745
3 5.61633 7.20652 7.55039 7.71713 7.75815 7.74095 7.72610 7.71934 7.75459
4 5.71211 7.20071 7.54165 7.71308 7.76433 7.74835 7.73323 7.72261 7.74459
5 5.78938 7.19130 7.56918 7.73931 7.78977 7.77982 7.76339 7.73900 7.77044
6 5.77390 7.14409 7.53450 7.68514 7.73473 7.71907 7.69908 7.68651 7.72154
    Col11   Col12
```

```
1 7.79656 7.80871
2 7.74334 7.75632
3 7.79603 7.80092
4 7.78738 7.79996
5 7.80845 7.81773
6 7.76073 7.77132
```

and the dimension of our data set is

```
R> dim(us.bis.yield)
```

```
[1] 1352   11
```

Further, the different maturities should be the `colnames` of our data matrix:

```
R> colnames(us.bis.yield) <- c("m0", "m1", "m2", "m3", "m4", "m5",
+     "m5.5", "m6.5", "m7.5", "m8.5", "m9.5")
```

PCA by "hand"

In a first step we calculate the covariance matrix of the data set

```
R> cov(us.bis.yield)
```

which gives (first six rows)

```
R> head(cov(us.bis.yield))
```

```
          m0        m1        m2        m3        m4        m5      m5.5
m0 0.1436181 0.1623816 0.1658599 0.1596904 0.1546596 0.1485336 0.1450543
m1 0.1623816 0.2547775 0.2917109 0.2950897 0.2940413 0.2883980 0.2843243
m2 0.1658599 0.2917109 0.3490359 0.3597939 0.3630219 0.3588354 0.3550393
m3 0.1596904 0.2950897 0.3597939 0.3746697 0.3804396 0.3773456 0.3740542
m4 0.1546596 0.2940413 0.3630219 0.3804396 0.3882237 0.3861293 0.3832672
m5 0.1485336 0.2883980 0.3588354 0.3773456 0.3861293 0.3850190 0.3825816
        m6.5      m7.5      m8.5      m9.5
m0 0.1420191 0.1376879 0.1371878 0.1398011
m1 0.2828456 0.2775029 0.2736390 0.2771187
m2 0.3566147 0.3529677 0.3484491 0.3531910
m3 0.3777618 0.3762414 0.3725839 0.3779079
m4 0.3886225 0.3886781 0.3856650 0.3916271
m5 0.3885547 0.3889885 0.3861106 0.3924130
```

By using the function `eigen` the eigenvalues and eigenvectors of the covariance matrix are computed

```
R> Eigenvalues <- eigen(cov(us.bis.yield))$values
R> Eigenvectors <- eigen(cov(us.bis.yield))$vectors
```

Now, the Principal Components can be estimated via a matrix multiplication

```
R> PC <- as.matrix(us.bis.yield) %*% Eigenvectors
```

As a check of the result, we compute the covariance matrix of `PC`. The variances of `cov(PC)` should be equal to the `Eigenvalues` and the covariances should be 0 (aside from rounding errors) since the Principal Components have to be uncorrelated.

```
R> cov(PC)
```

We do this for the first three Eigenvalues

```
R> Eigenvalues[1:3]
```

```
[1] 3.71791147 0.14822423 0.03232023
```

```
R> cov(PC)[1:3, 1:3]
```

```
              [,1]          [,2]          [,3]
[1,]  3.717911e+00  4.852262e-16 -2.081010e-16
[2,]  4.852262e-16  1.482242e-01 -7.016016e-17
[3,] -2.081010e-16 -7.016016e-17  3.232023e-02
```

In a next step we calculate the proportions of the variation explained by the various components:

```
R> print(round(Eigenvalues/sum(Eigenvalues) * 100, digits = 2))
```

```
R> round(cumsum(Eigenvalues)/sum(Eigenvalues) * 100, digits = 2)
```

```
 [1]  95.26  99.05  99.88  99.95  99.97  99.99  99.99 100.00 100.00 100.00
[11] 100.00
```

Thus, the first component `round(Eigenvalues[1]/sum(Eigenvalues)*100, digits=2)` explains 95.26, and the first three eigenvectors of the covariance matrix explain 99.88 of the total variation in the data. This suggest that the effective dimension of the space of yield curves could be three and any of the yield curves from our data set can be described by a linear combination of the first three loadings, while the relative error being very small.

PCA using `prcomp`

The best way to do PCA with R is to use the function `prcomp` from the package **stats**. `prcomp` uses as arguments simply a data matrix. Furthermore, with the argument `scale = TRUE` (default: `scale = FALSE`) the variables can be scaled to a unit variance before the analysis takes place.

To run PCA on this data we use

```
R> us.bis.yield.pca <- prcomp(us.bis.yield)
```

Note: To reproduce our previous calculation we use the default case (`scale = FALSE`). The Print-Output of `us.bis.yield.pca` gives us the estimated standard deviations as well as the rotations (loadings).

```
R> us.bis.yield.pca
```

```
Standard deviations:
 [1] 1.928188651 0.384999002 0.179778286 0.052338514 0.027326914 0.024617071
 [7] 0.016790628 0.010343001 0.008574908 0.005125979 0.002547559

Rotation:
           PC1         PC2         PC3         PC4         PC5         PC6
m0  -0.1301106 -0.67887236 -0.61014315 -0.33654289  0.13727233 -0.13029652
m1  -0.2432087 -0.47517336  0.13122421  0.45003331 -0.39767404  0.49493488
m2  -0.3003251 -0.26575103  0.30397937  0.20265271  0.13853751 -0.08577518
```

```
m3   -0.3153850 -0.12425768  0.26781888  0.16774144  0.32738294 -0.31042743
m4   -0.3225106 -0.02475883  0.19802262 -0.01434977  0.24703047 -0.31506722
m5   -0.3211199  0.03397813  0.18586080 -0.31143290 -0.21798846 -0.18336528
m5.5 -0.3190031  0.06308141  0.16473170 -0.40297136 -0.37989315 -0.02242068
m6.5 -0.3244514  0.14581976  0.03042842 -0.40901415 -0.01803397  0.40812177
m7.5 -0.3257698  0.23154538 -0.17028398 -0.03727271  0.44826363  0.48204013
m8.5 -0.3243321  0.26272145 -0.35610015  0.33596518  0.17588162  0.01146841
m9.5 -0.3298148  0.27862703 -0.43445547  0.27178461 -0.45997583 -0.31920096
             PC7         PC8          PC9          PC10         PC11
m0    0.009155617 -0.02288999 -0.006855907 -0.012994144 -0.004823424
m1    0.187014331  0.23682836 -0.026960532  0.000368626  0.020392310
m2   -0.507611913 -0.54043227  0.329862783  0.133076850 -0.053887246
m3    0.074000757  0.01601809 -0.633381685 -0.393361603  0.141580327
m4    0.110710697  0.57324382  0.200065544  0.451496299 -0.333051260
m5    0.190505362  0.08134537  0.398539545 -0.155467004  0.676119727
m5.5  0.289017291 -0.30075917 -0.046599651 -0.214356525 -0.581211231
m6.5 -0.297955976  0.00311431 -0.446854774  0.458536593  0.199093779
m7.5 -0.120221491  0.18029974  0.289156264 -0.478698703 -0.129795826
m8.5  0.529150858 -0.40209500  0.012512048  0.317891383  0.110217189
m9.5 -0.431838050  0.17659292 -0.069034106 -0.113830637 -0.045950235
```

We can extract the variances of the components with

```
R> us.bis.yield.pca.var <- us.bis.yield.pca$sdev^2
```

```
R> us.bis.yield.pca.var[1:3]
```

```
[1] 3.71791147 0.14822423 0.03232023
```

which are identical to the `Eigenvalues`

```
R> Eigenvalues[1:3]
```

```
[1] 3.71791147 0.14822423 0.03232023
```

estimated above.

Further on in Figure 1, we generate a screenplot with `barplot(us.bis.yield)` or directly with:

(**Note:** The plot method for `prcomp` objects creates screenplots by default.)

In order to better understand the implications of our results, we plot the first four loadings (see Figure 2):

The first loading is essentially flat, so a component on this loading represents the average yield over the maturities. The monotone nature of the second loading might indicate the trend in the yield curve. The shape of the third loading suggests that the third component captures the curvature nature of the yield curve. Finally, the shape of the fourth loading does not seem to have an obvious interpretation. (Remember, that most of the variation is explained by the first three components.)

```
R> plot(us.bis.yield.pca)
```

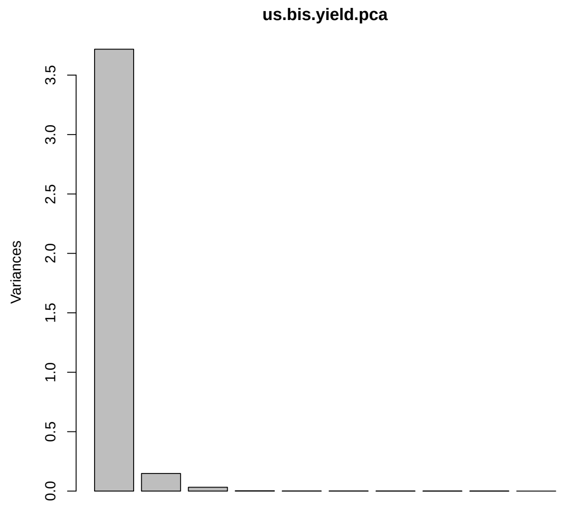

Figure 1: Proportions of the variances explained by the components of the daily changes in the US yield curve.

```
R> par(mfrow = c(2, 2))
R> plot(us.bis.yield.pca$rotation[, 1], ylim = c(-0.7, 0.7))
R> plot(us.bis.yield.pca$rotation[, 2], ylim = c(-0.7, 0.7))
R> plot(us.bis.yield.pca$rotation[, 3], ylim = c(-0.7, 0.7))
R> plot(us.bis.yield.pca$rotation[, 4], ylim = c(-0.7, 0.7))
```

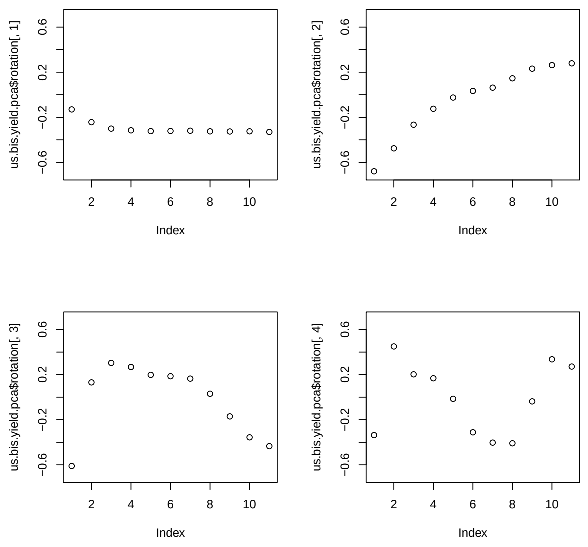

Figure 2: From top left to right and top down: Sequential plots of the first four US yield loadings.