Hauptkomponentenanalyse in R

Achim Zeileis

2009-02-20

Um die Ergebnisse aus der Vorlesung zu reproduzieren, wird zunächst der `GSA` Datensatz geladen

```
R> load("GSA.rda")
```

und wie schon in *MVA.pdf* aggregiert. Die nach `country` aggregierten "Erfolgsanteile" für 8 verschiedene Sommeraktivitäten werden berechnet:

```
R> gsa <- GSA[, c(5, 10:12, 14, 25:27, 29)]
R> gsa <- na.omit(gsa)
R> sucrate <- function(x) prop.table(table(x))[2]
R> gsa <- aggregate(gsa, list(gsa$country), sucrate)
R> rownames(gsa) <- gsa[, 1]
R> gsa <- gsa[, -(1:2)]
```

Die Hauptkomponentenanalyse (Principal Component Analysis) wird am besten mit dem Befehl `prcomp` durchgeführt, dem man einfach die Datenmatrix übergibt. Zusätzlich kann man das Argument `scale = FALSE` setzen (Voreinstellung) oder `scale = TRUE`. In ersterem Fall wird mit den Daten X direkt gerechnet, d.h. auf Basis der Kovarianzen. In letzterem Fall wird mit den skalierten Daten $\hat{X}$ gerechnet, dies entspricht den Korrelationen. In den meisten Situationen ist eine Skalierung angebracht:

```
R> gsa.pca <- prcomp(gsa, scale = TRUE)
R> gsa.pca

Standard deviations:
[1] 2.41548164 0.98906079 0.66908664 0.53615901 0.48939671 0.36629424 0.26459726
[8] 0.09149568

Rotation:
                   PC1        PC2         PC3        PC4         PC5
SA01.tennis  -0.3510511 -0.2870180 -0.41758509  0.5263352  0.30026958
SA02.cycle   -0.3867540 -0.2022680 -0.03744281  0.3259185 -0.01554934
SA03.ride    -0.3151780 -0.4769682  0.34952662 -0.5287251  0.50209261
SA05.swim    -0.3629343  0.2486504  0.40518694  0.1724966 -0.29008759
SA17.shop     0.3440531  0.3867074  0.09456128  0.2605538  0.73765974
SA18.concert  0.3511512 -0.3030234 -0.53164265 -0.2376023 -0.11006148
SA19.sight    0.3844260 -0.2319453  0.38371626  0.2006746 -0.06894624
SA21.museum   0.3265930 -0.5405722  0.31472449  0.3779282 -0.11108374
                   PC6         PC7         PC8
SA01.tennis   0.25446444 -0.41903259  0.11169380
SA02.cycle   -0.38098760  0.71937451  0.19746184
SA03.ride     0.13295112  0.02701752  0.03069495
```

```
SA05.swim      0.70460259  0.17549561  0.03325372
SA17.shop      0.15583618  0.29128063 -0.04453692
SA18.concert   0.49850346  0.40633166  0.14158478
SA19.sight    -0.01707316 -0.14968541  0.76384472
SA21.museum    0.05360391  0.05502547 -0.58394930
```

Die Print-Ausgabe gibt uns die zu den Hauptkomponenten gehörigen Standardabweichungen an, sowie die zugehörigen Rotationen (auch genannt Ladungen). Hier kontrastiert die erste Hauptkomponente die sportlichen Aktivitäten mit den kulturellen. Die zweite Hauptkomponente vergleicht vor allem die Aktivitäten "Schwimmen & Shopping" mit allen übrigen.

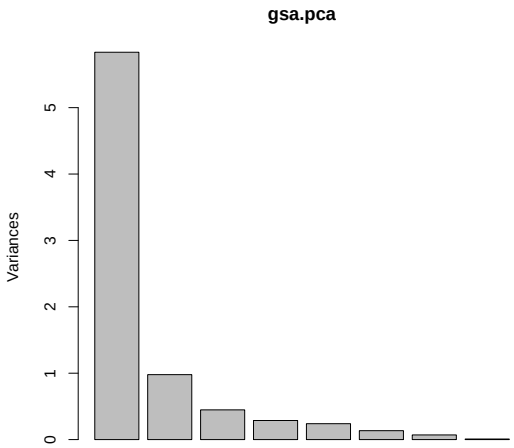
Die zu den Hauptkomponenten gehörenden Standardabweichungen kann man sich extrahieren und daraus die Varianzen berechnen:

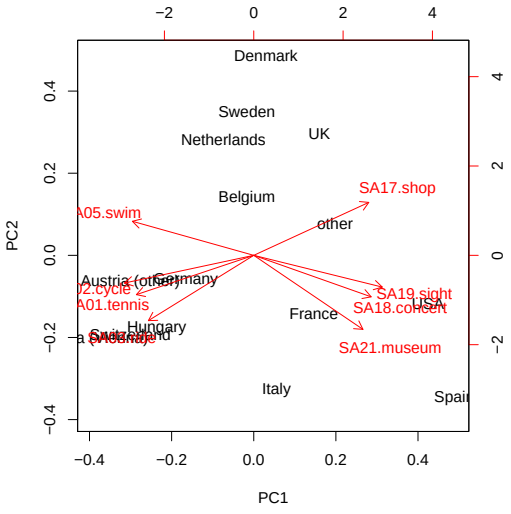
```
R> gsa.var <- gsa.pca$sdev^2
R> gsa.var

[1] 5.834551560 0.978241243 0.447676928 0.287466488 0.239509144 0.134171467
[7] 0.070011711 0.008371459

R> gsa.var/sum(gsa.var)

[1] 0.729318945 0.122280155 0.055959616 0.035933311 0.029938643 0.016771433
[7] 0.008751464 0.001046432

R> cumsum(gsa.var)/sum(gsa.var)

[1] 0.7293189 0.8515991 0.9075587 0.9434920 0.9734307 0.9902021 0.9989536
[8] 1.0000000
```

Der obige Code hat außerdem die Anteile der Varianzen berechnet, sowie die kumulativen Anteile. Also erklärt die erste Hauptkomponente rund 73% der Varianz, die zweite 12%. Demnach erklären sie zusammen 85%.

Den zugehörigen Screeplot kann man sich entweder "von Hand" per `barplot(gsa.var)` generieren oder direkt als `plot(gsa.pca)`. (Bemerkung: Die Plot-Methode für `"prcomp"` Objekte zeichnet also Screeplots.) Analog kann man sich den zugehörigen Biplot generieren als `biplot(gsa.pca)`.

```
R> plot(gsa.pca)
R> biplot(gsa.pca)
```





Aus dem Screeplot kann man ablesen, daß die ersten beiden Hauptkomponenten den größten Teil der Variation in den Daten einfangen, man also mit den ersten zwei Komponenten gut auskommt. Der Biplot zeigt, daß die Daten in x-Richtung nach kulturellen versus sportlichen Aktivitäten unterschieden werden. In y-Richtung werden vor allem die Länder hervorgehoben, die nach Österreich zum Shoppen und/oder Schwimmen kommen.

Bemerkung: Um den Biplot etwas zu verschönern, kann man einige Grafik-Parameter ändern. Mit `xlim` und `ylim` kann man generell die Spannbreite der x- bzw. y-Skala setzen. Und beim Biplot kann man mit `expand` die Länge der Variablenachsen steuern. Ein deutlich lesbarerer Biplot wird deshalb durch `biplot(gsa.pca, xlim = c(-0.5, 0.5), expand = 0.8)` erzeugt.

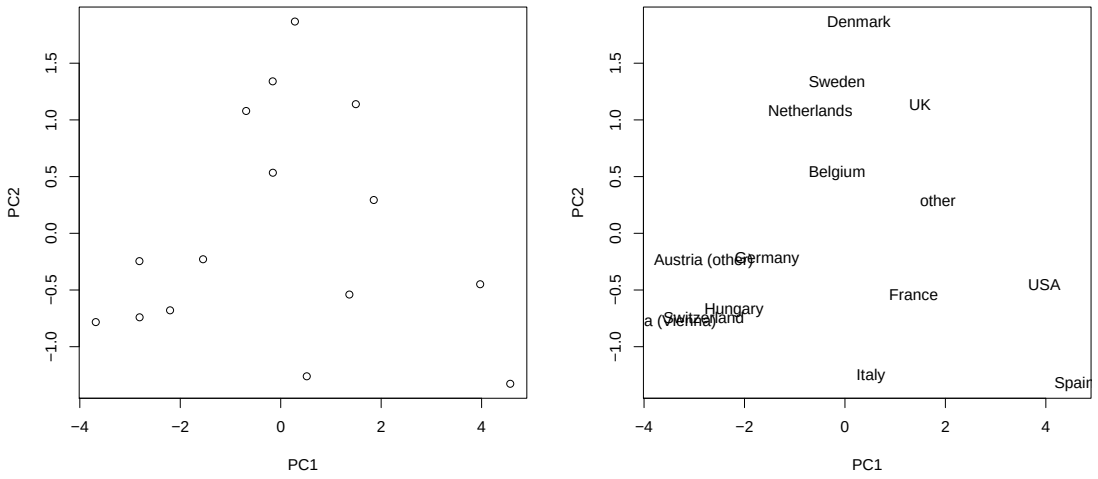
Um zu illustrieren, wie die Hauptkomponenten auch "von Hand" ausgerechnet werden können, schauen wir uns noch ein paar Befehle mehr an. Zunächst wollen wir uns die konkreten Linearkombinationen XA ausrechnen, dies tut die entsprechende `predict`-Methode:

```
R> gsa.pred <- predict(gsa.pca)
R> gsa.pred
```

```
                      PC1        PC2         PC3          PC4         PC5
Austria (Vienna) -3.6822297 -0.7828332 -0.03216091 -0.242384898 -0.05575787
Austria (other)  -2.8133293 -0.2453209 -0.75417806  0.812150078 -0.42712998
Belgium          -0.1565185  0.5342101 -0.06000080 -0.795772731 -0.03879853
Denmark           0.2826928  1.8675474  0.17065021  0.622818994  0.36539598
France            1.3669924 -0.5399365 -1.20632326 -0.429653107 -0.83219333
Germany          -1.5462097 -0.2287684  0.34663736 -0.737649700 -0.13798341
Hungary          -2.2030790 -0.6790603  1.68203514 -0.008114349  0.02656072
Italy             0.5187196 -1.2608745 -0.07803747  0.513258573  0.51901410
Netherlands      -0.6880971  1.0788038 -0.28337583  0.600982560 -0.10300886
Spain             4.5702924 -1.3269237  0.60826366  0.327412430 -0.37002484
Sweden           -0.1595130  1.3401868  0.43641079 -0.397328850 -0.65746791
Switzerland      -2.8097011 -0.7407420 -0.62324910 -0.037170849  0.74173069
UK                1.4957117  1.1389414 -0.17016541 -0.458704297  0.69152076
USA               3.9714954 -0.4490180 -0.33084574 -0.347674263  0.58854482
other             1.8527730  0.2937880  0.29433943  0.577830410 -0.31040231
                       PC6         PC7           PC8
Austria (Vienna) -0.11636848 -0.17791696 -0.1139368430
Austria (other)  -0.26248745  0.08522520  0.0769156738
Belgium          -0.01202968  0.19900717 -0.1709009896
Denmark           0.28585273 -0.17006536 -0.0719815443
France            0.53063642 -0.01898972  0.0145414009
Germany          -0.59969658  0.27227363  0.0371712902
Hungary           0.47025199  0.17911637  0.0858836589
Italy             0.07269465  0.01239793 -0.1739980503
Netherlands      -0.44075494  0.35005418  0.0230283920
Spain            -0.53073339 -0.39453318  0.0071682970
Sweden            0.18312484 -0.37201850  0.0292067964
Switzerland       0.24221150 -0.32999361  0.1124813150
UK               -0.36006541 -0.22759334  0.0659580674
USA               0.22231754  0.37669780  0.0789452498
other             0.31504628  0.21633838 -0.0004827141
```

Um jetzt die beiden ersten Hauptkomponenten gegeneinander abzutragen, extrahieren wir einfach die ersten beiden Spalten und visualisieren diese in einem Streudiagramm. Da dieses nicht besonders gut lesbar ist, erzeugen wir noch einmal denselben Plot, aber lassen die Punkte weg (`type = "n"`, no plot), fügen aber Achsenbeschriftungen (`xlab` und `ylab`) hinzu. Danach fügen wir die Zeilennamen (d.h., Ländernamen) mit dem Befehl `text` hinzu.

```
R> plot(gsa.pred[, 1:2])
R> plot(gsa.pred[, 1:2], type = "n", xlab = "PC1", ylab = "PC2")
R> text(gsa.pred[, 1:2], rownames(gsa))
```



Bemerkung: Der Befehl `plot(gsa.pred[, 1:2])` macht dasselbe wie der Befehl `plot(gsa.pred[, 1], gsa.pred[, 2])`, er plottet also die erste Spalte von `gsa.pred` gegen die zweite.