Explorative multivariate Analyse in R

Achim Zeileis

2009-02-20

1 Datenaufbereitung

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

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

Dann wird der Teildatensatz ausgewählt, der nur die Variable `country` sowie die 8 interessierenden Sommeraktivitäten enthält, und dann werden die fehlenden Werte weggelassen:

```
R> gsa <- GSA[, c(5, 10:12, 14, 25:27, 29)]
R> gsa <- na.omit(gsa)
```

Diese Daten sind jetzt immer noch auf Ebene des Individuums, jedoch möchten wir die Daten über alle Touristen eines Landes aggregieren. Dafür verwenden wir die Funktion `aggregate`. Als Aggregationsfunktion soll dabei die 'Erfolgsrate' für jede Aktivität berechnet werden. Dies haben wir bisher über

```
R> prop.table(table(gsa$SA01.tennis))
```

```
        no        yes
0.97766889 0.02233111
```

gemacht. Um jetzt nur die Erfolgsrate, d.h. den zweiten Wert obiger Tabelle, zu extrahieren, definieren wir uns eine neue Hilfsfunktion `sucrate`:

```
R> sucrate <- function(x) prop.table(table(x))[2]
```

Diese nimmt einen Faktor `x` als Argument, berechnet dann die relative Häufigkeitstabelle und extrahiert schliesslich das zweite Element. Beispiel:

```
R> sucrate(gsa$SA01.tennis)
```

```
       yes
0.02233111
```

Diese möchten wir jetzt für jedes Land anwenden:

```
R> gsa <- aggregate(gsa, list(gsa$country), sucrate)
```

Die Syntax ist dabei ähnlich wie bei `tapply`, genauere Informationen können auf `help(aggregate)` bezogen werden.

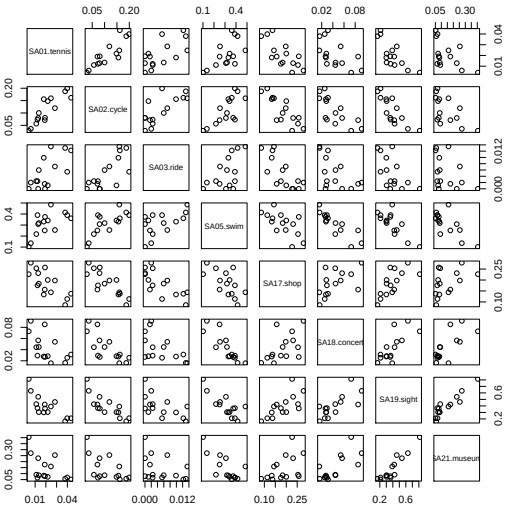
Die so aggregierten Daten werden jetzt noch mit 'schönen' Zeilennamen versehen und die beiden unnötigen ersten Spalten werden weggelassen:

```
R> rownames(gsa) <- gsa[, 1]
R> gsa <- gsa[, -(1:2)]
```

2 Paarweise Streudiagramme

Eine Streudiagramm-Matrix kann in R sehr leicht mit Hilfe des Befehls `pairs` erzeugt werden, dem man einfach einen `"data.frame"` oder eine `"matrix"` übergibt.

```
R> pairs(gsa)
```



Um nur bestimmte Variablen herauszugreifen, können natürlich auch nur einzelne Spalten der Daten selektiert werden, bspw. `pairs(x[,1:4])` oder `pairs(x[,5:8])`.

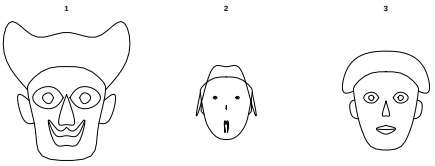
3 Chernoff-Gesichter

Funktionalität zur Darstellung von Chernoff-Gesichtern ist in R nicht direkt vorhanden, aber wenn wir die Datei

```
R> source("faces.R")
```

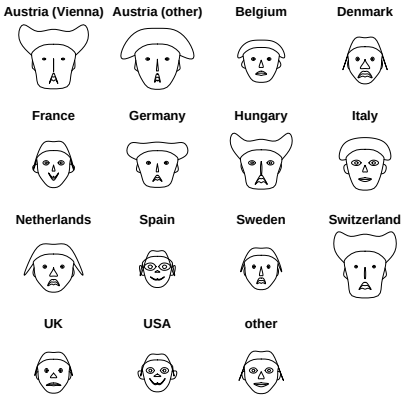
laden, wird eine Funktion `faces` zur Verfügung gestellt. Dieser übergeben wir auch wieder einfach die Datenmatrix als Argument. In einem ersten Schritt wollen wir zwei Extremgesichter (nur 1 bzw. nur 0) und ein Durchschnittsgesicht (mit nur 0.5) zeichnen:

```
R> x <- rbind(rep(1, 11), 0, 0.5)
R> faces(x, nrow = 1, ncol = 3)
```



Völlig analog können dann die Touristendaten visualisiert werden

```
R> faces(gsa)
```



Um zu veranschaulichen, daß die Belegung der Attribute mit verschiedenen Variablen einen deutlichen Einfluß auf die Gesichter hat, kann man dieselbe Grafik für verschiedene Permutationen der Spalten wiederholen, bspw. für `faces(gsa[, c(7, 8, 5, 6, 4, 1, 2, 3)])`.

Trotz dieser etwas willkürlichen Einteilungen, läßt sich eine grobe Einteilung der Länder in Gruppen vornehmen, die wir im folgenden zur Einfärbung verschiedener Grafiken verwenden wollen:

```
R> group <- c(1, 1, 2, 2, 2, 1, 1, 2, 2, 4, 2, 1, 2, 4, 4)
R> names(group) <- rownames(gsa)
R> group
```

```
Austria (Vienna)  Austria (other)          Belgium          Denmark
               1                1                2                2
          France          Germany          Hungary            Italy
               2                1                1                2
     Netherlands            Spain           Sweden      Switzerland
               2                4                2                1
              UK              USA            other
               2                4                4
```

Daß dabei auf die Gruppe `3` verzichtet wird, hat allein den Grund, daß die Farbe `4` (blau) in Grafiken besser abzulesen ist als `3` (grün).

4 Andrews-Kurven

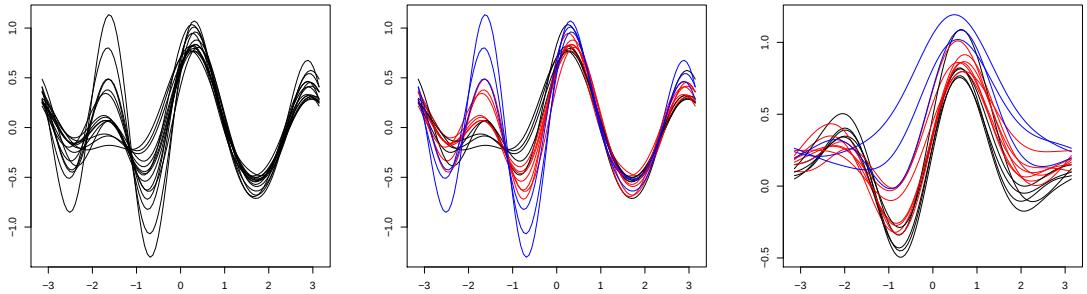
R enthält ebenfalls keine Funktion zur Visualisierung von Andrews-Kurven, eine solche wird allerdings durch

```
R> source("andrews.R")
```

verfügbar gemacht.

Auch hier ist die Syntax wieder komplett analog zu den vorigen Beispielen. Zusätzlich färben wir hier die verschiedenen Kurven nach den oben definierten Gruppen ein, um Unterschiede zwischen den Ländern hervorzuheben.

```
R> andrews(gsa)
R> andrews(gsa, col = group)
R> andrews(gsa[, c(7, 8, 5, 6, 4, 1, 2, 3)], col = group)
```



5 Parallele Koordinaten

Eine Funktion zur Visualisierung von parallelen Koordinaten ist zwar nicht in R selbst verfügbar, jedoch in dem Paket **MASS**, das in jeder R-Installation mitenthalten ist und mit dem Befehl `library` geladen werden kann:

```
R> library(MASS)
```

Nun ist (unter anderem) der Befehl `parcoord` verfügbar, der wie schon die vorangegangenen Befehle aufgerufen werden kann:

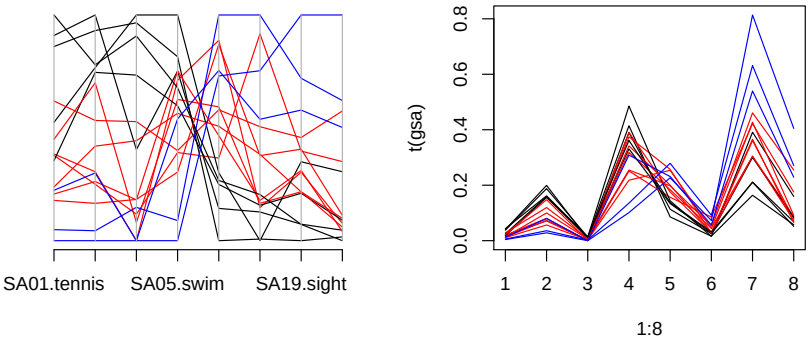
```
R> parcoord(gsa, col = group)
```

Hieraus ist ersichtlich, daß die Touristen aus der ersten (schwarzen) Gruppe eher sportliche Aktivitäten bevorzugen und weniger kulturelle Aktivitäten verfolgen. Für die Touristen der dritten (blauen) Gruppe verhält es sich dagegen genau umgekehrt. Die übrigen Länder liegen irgendwo dazwischen.

Anstatt die Daten auf das Intervall $[0, 1]$ zu reskalieren, kann man auch die ursprünglichen Daten direkt visualisieren, da diese ja bereits Anteilen entsprechen. Dies kann allerdings nicht mit `parcoord` durchgeführt werden, aber mit `matplot`. Der entsprechende Aufruf ist allerdings ein bisschen komplzierter, sollte aber dennoch selbsterklärend sein (falls nicht: siehe `help(matplot)`).

```
R> matplot(1:8, t(gsa), type = "l", lty = 1, col = group)
```

Hier ist dieselbe Information etwas schwerer ablesbar, da einfach die verschiedenen Aktivitäten sich sehr unterschiedlicher Popularität erfreuen und der Gesamteindruck von den populären Aktivitäten dominiert wird.

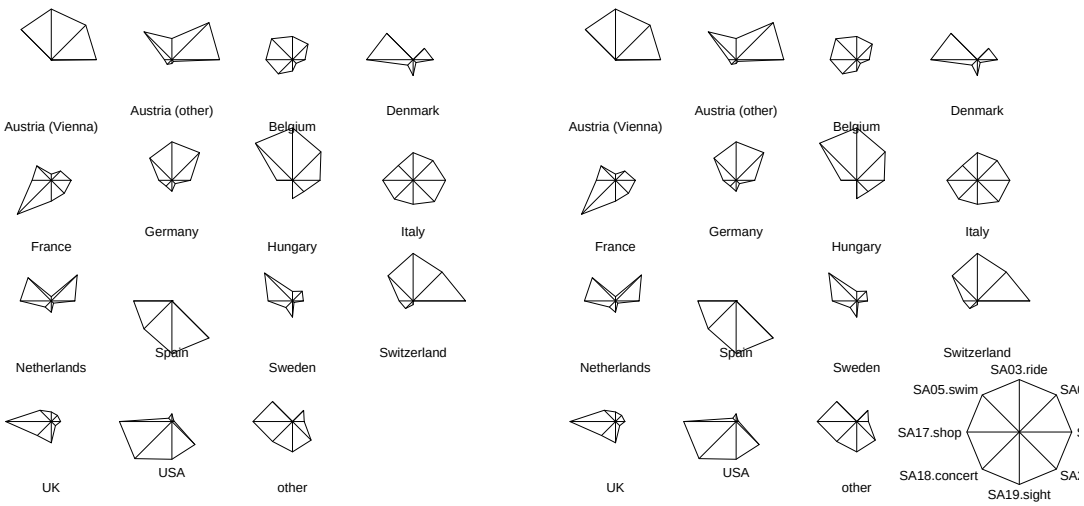


6 Sterne

Sterne können in R mit Hilfe der Funktion `stars` gemalt werden. Im einfachsten Fall kann diese wieder einfach als `stars(x)` aufgerufen werden, jedoch ermöglichen zahlreiche Argumente eine Verbesserung der Darstellung, bspw. durch Verwendung von Farben oder Hinzufügen einer Legende. Die Hilfeseite `help(stars)` gibt einen Überblick über die Argumente und die meisten werden auch in `example(stars)` illustriert. (Anmerkung: damit nicht alle Grafiken des Beispiels auf einmal an einem 'vorbeirauschen', kann man vorher den Parameter `par(ask = TRUE)` setzen, dann wird vor jeder neuen Grafik auf eine Bestätigung per Return gewartet.)

Einige der Argumente sollen hier kurz für den Touristen-Datensatz vorgeführt werden. Ein einfache Darstellung von Sternen ohne bzw. mit Legen kann folgendermaßen generiert werden:

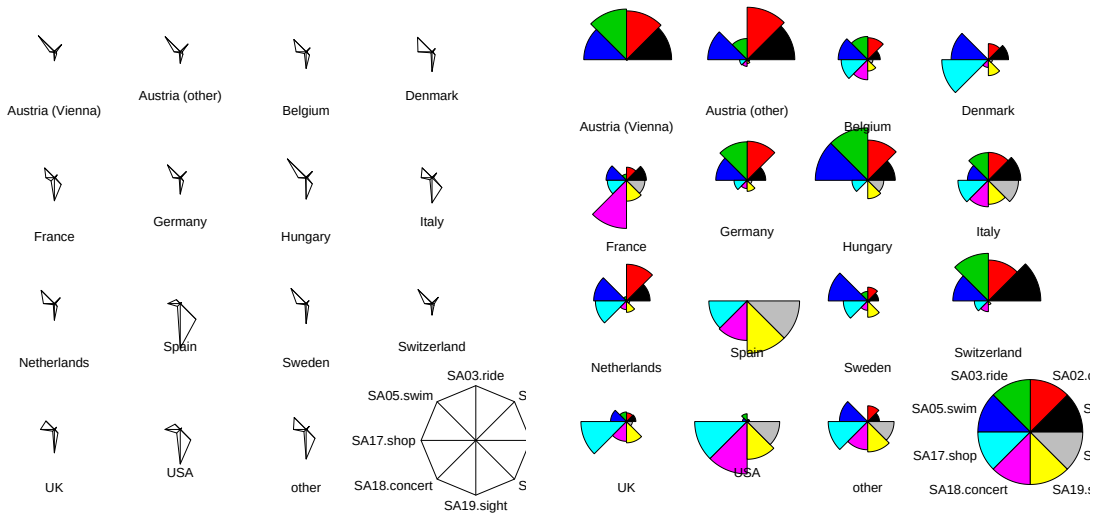
```
R> stars(gsa)
R> stars(gsa, key.loc = c(10, 2.1))
```



Dabei fallen schon deutliche Unterschiede in den Interessen der Touristen verschiedener Länder auf. Beispielsweise bevorzugen die österreichischen und schweizerischen Touristen eher sportliche Aktivitäten, während u.a. die Spanier und US-Amerikaner fas ausschließlich an kulturellem Programm interessiert sind.

Um nicht jede einzelne Aktivität auf das Intervall $[0, 1]$ zu skalieren (d.h. also $\tilde{X}$ zu verwenden), kann man auch das Argument `scale` setzen, dann werden die Originalbeobachtungen `X` visualisiert. Dabei ist abzulesen, daß es nur einige wenige populäre Aktivitäten gibt (wie bspw. Schwimmen oder Sightseeing). Eine andere Modifikation der Darstellung wäre es farbige Segmente zu malen, dies kann durch das Argument `draw.segments` gesteuert werden.

```
R> stars(gsa, key.loc = c(10, 2.1), scale = FALSE)
R> stars(gsa, key.loc = c(10, 2.1), draw.segments = TRUE)
```

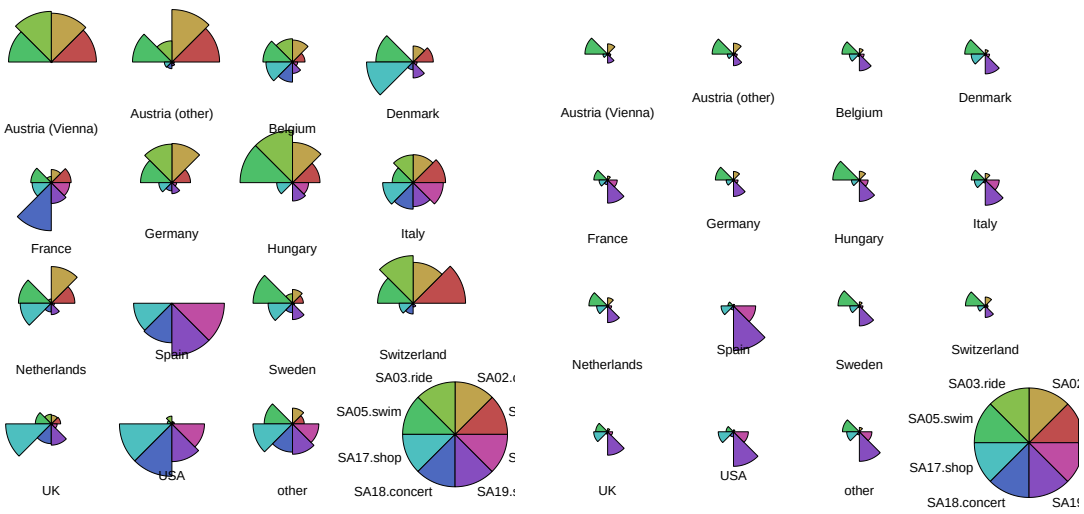


Die dabei gewählten Farben sind allerdings sehr grell, was bei längerer Betrachtung sehr unangenehm ist. Deshalb erzeugen wir uns mit dem Befehl `rainbow` eine Sequenz von 8 Farben mit niedrigerer Saturierung (`s`) und höherem Grau-Anteil (`v`).

```
R> myc <- rainbow(8, s = 0.6, v = 0.75)
```

Diese kann dann sowohl in dem skalierten als auch unskalierten Sterne-Diagramm eingesetzt werden:

```
R> stars(gsa, key.loc = c(10, 2.1), draw.segments = TRUE, col.segments = myc)
R> stars(gsa, key.loc = c(10, 2.1), draw.segments = TRUE, col.segments = myc,
+      scale = FALSE)
```



Alternativ könnte man sich auch noch das transponierte Problem anschauen: wo also die Aktivitäten bezüglich der verschiedenen Länder unterschieden werden:

```
R> myc2 <- rainbow(15, s = 0.6, v = 0.75)
R> stars(t(gsa), key.loc = c(7, 2.5), draw.segments = TRUE, col.segments = myc2,
+       scale = FALSE)
```