Bivariate explorative Datenanalyse in R

Achim Zeileis

2009-02-20

1 Zwei metrische Merkmale

Zunächst einmal wird wieder der `BBBClub` Datensatz geladen und der Bequemlichkeit halber auch attached.

```
R> load("BBBClub.rda")
R> attach(BBBClub)
```

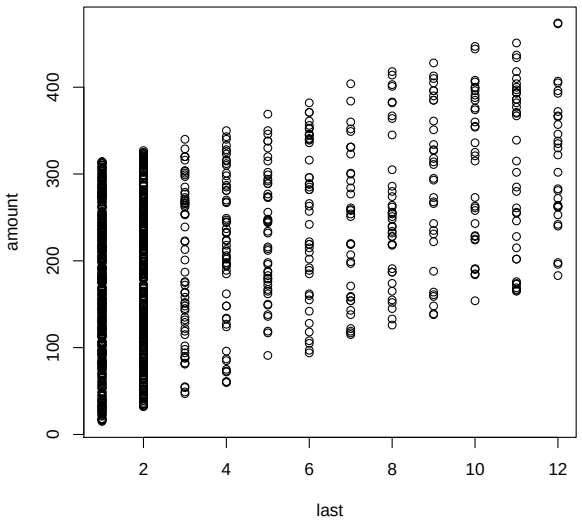
Zur Messung des Zusammenhangs von zwei metrischen Variablen, kann die Korrelation verwendet werden. Der Korrelationskoeffizient von `amount` und `last` wird berechnet durch

```
R> cor(amount, last)
```

```
[1] 0.4521105
```

Aber viel anschaulicher wird der Zusammenhang der beiden Variablen in einem Streudiagramm dargestellt.

```
R> plot(amount ~ last)
```

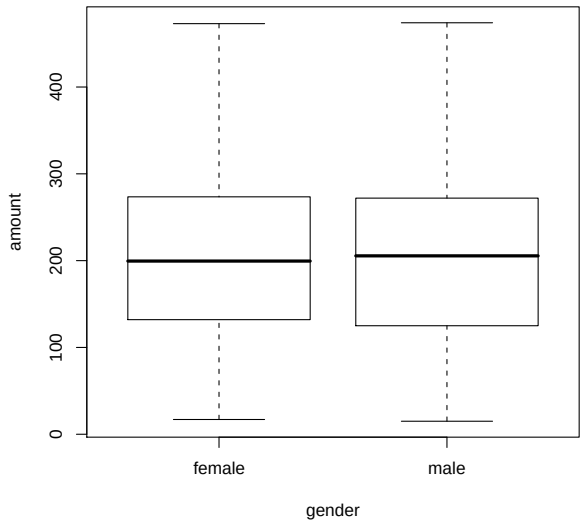


Hier wird die Formelnotation verwendet, in der `amount` erklärt wird (`~`) durch `last`. Äquivalent könnte auch `plot(last, amount)` verwendet werden. Sowohl Streudiagramm als auch Korrelationskoeffizient zeigen an, daß die beiden Merkmale positiv gekoppelt sind, also die Ausgaben um so höher sind je länger der letzte Kauf zurückliegt. Im Streudiagramm ist außerdem nochmal deutlich zu sehen, daß die Variable `last` diskret ist.

2 Ein abhängiges metrisches und ein erklärendes kategoriales Merkmal

Ist die erklärende Variable nicht metrisch wie in obigem Beispiel, wählt die Funktion `plot` automatisch eine passende Grafik aus, nämlich einen parallelen Boxplot. Die Syntax ist aber identisch zum obigen Beispiel:

```
R> plot(amount ~ gender)
```



Aus diesem Boxplot läßt sich sehr klar ablesen, daß die Verteilung der Ausgaben bei Männern und Frauen annähernd identisch sind. Um numerische Zusammenfassungen der beiden Teilstichproben zu bekommen, wird in R der Befehl `tapply` verwendet. Als erstes Argument wird dabei die metrische Variable, als zweites die gruppierende kategoriale Variable und als drittes die zu evaluierende Funktion übergeben. Um die Mittelwerte in beiden Gruppen zu berechnen:

```
R> tapply(amount, gender, mean)
```

```
  female     male
203.4912 200.2227
```

liefert einen Vektor mit beiden Mittelwerten zurück. Liefert die zu evaluierende Funktion nicht einen Skalar (also nur einen Wert, wie `mean`) sondern selbst schon einen Vektor liefert, dann returniert `tapply` eine Liste:

```
R> tapply(amount, gender, summary)
```

```
$female
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   17.0   132.0   199.5   203.5   273.2   473.0

$male
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   15.0   125.0   205.5   200.2   272.0   474.0
```

3 Zwei kategoriale Merkmale

Die einfachste numerische Beschreibung für zwei (oder mehr) kategoriale Merkmale, ist ihre Kontingenztafel, in der immer noch die gesamte Information der Originaldaten enthalten ist. In R gibt es zwei Befehle für die Erzeugung von Kontingenztafeln aus Faktoren (d.h. Objekten der Klasse `"factor"`, die qualitative Variablen kodieren) und zwar `table` und `xtabs`. Das Resultat ist äquivalent, aber `xtabs` hat ein Formelinterface, das manchmal bequemer zu benutzen sein kann. Um also die Kontigenztafel von Geschlecht (`gender`) und Kaufentscheidung (`choice`) zu berechnen können wir völlig äquivalent sagen:

```
R> table(gender, choice)

        choice
gender    no yes
  female 273 183
  male   627 217

R> xtabs(~gender + choice)

        choice
gender    no yes
  female 273 183
  male   627 217
```

Um mit der Kontigenztafel weiter rechnen zu können, speichern wir sie im Objekt `tab`.

```
R> tab <- table(gender, choice)
```

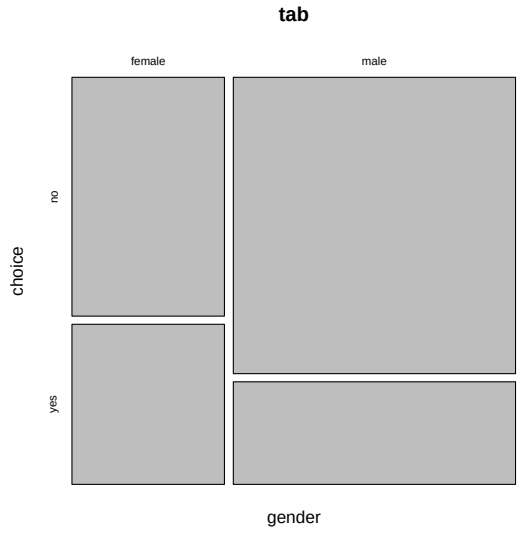
Diese enthält nun die absoluten Häufigkeiten, die in den entsprechenden Kombinationen von Kategorien beobachtet wurden. So haben beispielsweise 273 Frauen das Buch nicht gekauft (`female` und `no`). Wenn wir nun vergleichen wollen, ob Frauen oder Männer das Buch eher gekauft haben, ist dies von den absoluten Häufigkeiten schwer abzulesen, da unterschiedlich viele Frauen und Männer in der Stichprobe sind. Als Ausweg betrachtet man *bedingte* relative Häufigkeiten, d.h. also den Anteil der Frauen, die das Buch gekauft haben, sowie den Anteil der Männer, die das Buch gekauft haben. Diese bedingte Häufigkeitstabelle kann wieder mit dem Befehl `prop.table` berechnet werden. Als Argumente gibt man die Tabelle der absoluten Häufigkeiten an und die Nummer der Variablen auf die bedingt werden soll. Hier ist das die erste Variable `gender`, daher:

```
R> prop.table(tab, 1)

        choice
gender          no       yes
  female 0.5986842 0.4013158
  male   0.7428910 0.2571090
```

Die Zeilensummen in dieser Tabelle sind konsequenterweise genau 1, entsprechen also genau 100%. Aus der Tabelle können wir dann leicht ablesen, daß die empirische Kaufwahrscheinlichkeit bei den Frauen bei 40% liegt, während sie bei den Männern nur 25% beträgt.

Als Visualisierung für die Kontingenztafel ist der sogenannte Mosaikplot sehr gut geeignet, der all die Information sowohl der Kontingenztafel mit den absoluten Häufigkeiten als auch mit den bedingten relativen Häufigkeiten darstellt. Der Mosaikplot wird in R mit dem Befehl `mosaicplot` erzeugt.

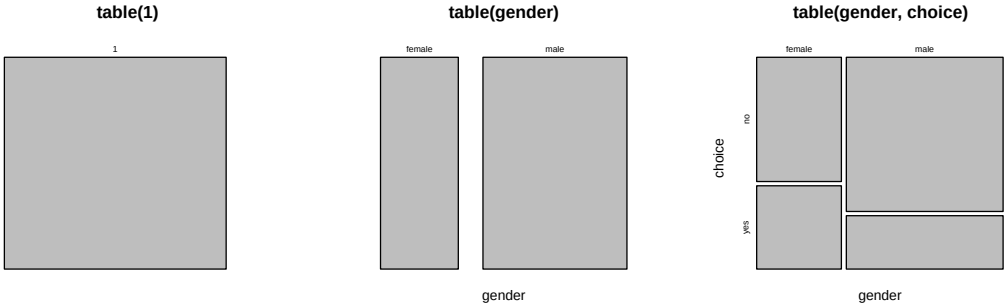
```
R> mosaicplot(tab)
```



Dieser ist in erster Linie eine flächenproportionale Darstellung der Kontingenztafel, d.h. salopp gesprochen: je größer der Eintrag in der Kontingenztafel, desto größer ist die Fläche des entsprechenden Mosaiks im Vergleich zu den übrigen Mosaiks. Hier ist also beispielsweise sehr einfach abzulesen, dass die häufigste Kombination der beiden Merkmale, die Ausprägung `male`/`no` ist, also Männer, die das Produkt nicht gekauft haben.

Um zu verstehen, wie der Mosaikplot konstruiert wird, schauen wir uns folgende Sequenz von Mosaikplots an:

```
R> mosaicplot(table(1))
R> mosaicplot(table(gender))
R> mosaicplot(table(gender, choice))
```



Zuerst haben wir einen Mosaikplot von einer Tabelle, die nur die Zahl 1 enthält. Dementsprechend hat der Mosaikplot nur ein einziges Rechteck, das 100% entspricht. Dann teilen wir dieses Rechteck gemäß der Variablen `gender`. Diese teilt das Rechteck vertikal in zwei Hälften der Anteile

```
R> prop.table(table(gender))
```

```
gender
   female      male
0.3507692 0.6492308
```

d.h. etwa in Anteile von 1/3 und 2/3. Im letzten Schritt wird jede dieser Hälften noch einmal horizontal geteilt und zwar gemäß der bedingten relativen Häufigkeiten

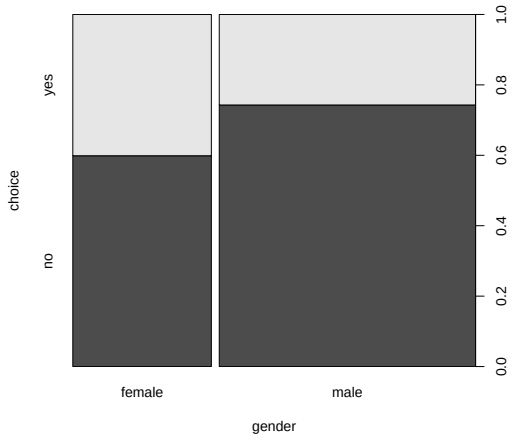
```
R> prop.table(tab, 1)
```

```
        choice
gender          no       yes
  female 0.5986842 0.4013158
  male   0.7428910 0.2571090
```

so daß die bedingte relative Häufigkeit bei den Frauen etwa 40% und bei den Männern nur 25% beträgt.

Der Mosaikplot läßt sich noch etwas verschönern, wenn man die Mosaikflächen bezüglich der bedingten Variable einfärbt und direkt aneinanderlegt. Eine solche Darstellung nennt man auch Spineplot, sie kann einfach über `spineplot(tab)` generiert werden. Diese Funktion hat außerdem ein angenehmes Formelinterface, so daß man die Kontingenztafel nicht vorher von Hand berechnen muß, sondern direkt `spineplot(choice ~ gender)` sagen kann. Dasselbe macht auch die `plot()` Funktion selbst, wenn sie eine Formel mit zwei kategorialen Variablen übergeben bekommt:

```
R> plot(choice ~ gender)
```



Auf der linken Seite steht, wie bei den Befehlen in den vorangegangenen Abschnitten, die abhängige Variable und rechts die erklärende Variable. Die Funktion liefert außerdem die Kontigenztabelle, die sie visualisiert hat unsichtbar zurück:

```
R> tab <- plot(choice ~ gender)
R> tab
```

```
        choice
gender    no yes
  female 273 183
  male   627 217
```

4 Ein abhängiges kategoriales und ein erklärendes metrisches Merkmal

Die einfachste Möglichkeit den Zusammenhang eines abhängigen kategorialen Merkmals mit einem erklärenden metrischem Merkmal numerisch und graphisch zu beschreiben, ist die metrische Variable zu diskretisieren und dann fortzufahren als hätte man zwei kategoriale Variablen. Diese Idee ist sehr ähnlich zu der Idee des Histogramms, wo die metrische Variable auch in verschiedene Intervalle (und damit also Kategorien) eingeteilt wird, für die dann wieder absolute und relative Häufigkeiten berechnet werden können.

Um eine quantitative Variable in verschiedene Kategorien gleichsam zu "zerschneiden" steht in R der Befehl `cut` zur Verfügung. Dieser erwartet als erstes Argument die metrische Variable und als zweites Argument die Stelle, an denen er schneiden soll. Eine einfache und in der Regel recht gute Wahl ist es, die Variable in vier gleichgroße Stücke zu zerschneiden. Man teilt dann also einfach mit den Werten der Fünf-Punkt-Zusammenfassung

```
R> fivenum(amount)

[1]  15 127 204 273 474
```

die uns das Minimum, das untere Quartil, den Median, das obere Quartil und das Maximum liefert. Zwischen jeweils zwei benachbarten Werten liegen also immer 25% der Daten. Die diskretisierte Variable erhalten wir dann mit

```
R> amount2 <- cut(amount, fivenum(amount), include.lowest = TRUE)
R> summary(amount2)

 [15,127] (127,204] (204,273] (273,474]
      326       327       330       317
```

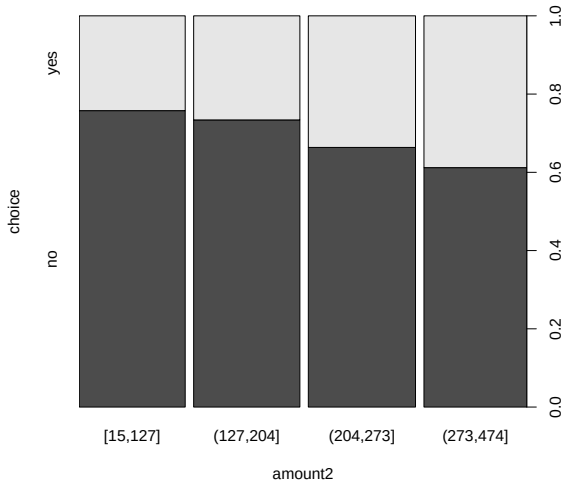
Das Argument `include.lowest` muß man hier auf `TRUE` setzen, damit der kleinste Wert 15 am linken Rand mit in das Intervall aufgenommen wird. Und mit der resultierenden kategorialen Variablen `amount2` verfahren wir jetzt genau wie im vorangegangenen Abschnitt, d.h. wir können Kontingenztafeln absoluter und relativer Häufigkeiten berechnen

```
R> tab <- table(amount2, choice)
R> tab

           choice
amount2      no yes
  [15,127]  247  79
  (127,204] 240  87
  (204,273] 219 111
  (273,474] 194 123

R> prop.table(tab, 1)
```

```
          choice
amount2            no       yes
  [15,127]  0.7576687 0.2423313
  (127,204] 0.7339450 0.2660550
  (204,273] 0.6636364 0.3363636
  (273,474] 0.6119874 0.3880126
```

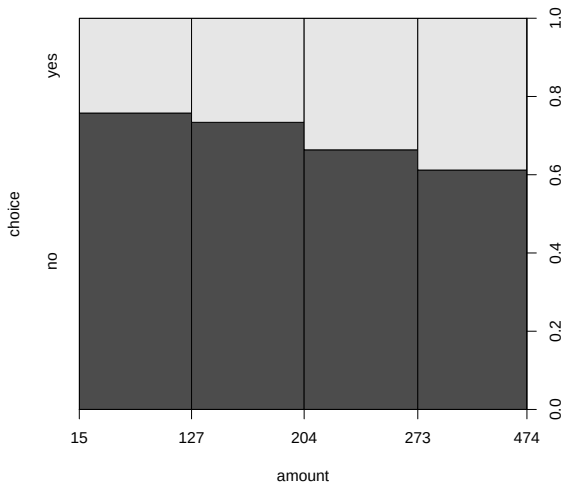
woraus wir ablesen können, dass die empirische Kaufwahrscheinlichkeit mit dem bisherigen Kaufvolumen steigt. Dies kann auch wieder mit einem Mosaikplot oder Spineplot visualisiert werden:

```
R> spineplot(tab)
```



Dabei wird allerdings die x-Achse unterbrochen, eine etwas schönere Darstellung bekommen wir, wenn wir `spinplot()` mit dem Formelinterface aufrufen und dabei die gewünschten Bruchpunkte mit dem Argument `breaks` uebergeben:

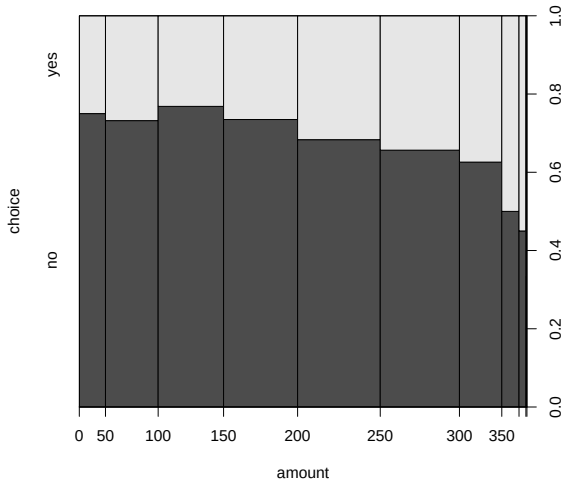
```
R> spineplot(choice ~ amount, breaks = fivenum(amount))
```



Setzt man das Argument `breaks` nicht, so werden die Bruchpunkte automatisch so gewählt wie bei der Funktion `hist()`. Diese versucht 'schöne' Bruchpunkte, z.B. in Zehnerschritten, zu finden wie das folgende Beispiel verdeutlicht. Dabei können allerdings sehr dünn besetzte Zellen in der Kontingenztabelle entstehen. Wie bereits im letzten Abschnitt illustriert liefert die Funktion auch immer die Kontigenztabelle, die sie visualisiert, unsichtbar zurück.

```
R> tab <- plot(choice ~ amount)
R> tab
```

```
           choice
amount       no yes
  [0,50]     57  19
  (50,100]  112  41
  (100,150] 146  44
  (150,200] 158  57
  (200,250] 164  76
  (250,300] 151  79
  (300,350]  77  46
  (350,400]  25  25
  (400,450]   9  11
  (450,500]   1   2
```



Nach Abschluß der Analysen soll der Arbeitsplatz aufgeräumt werden. Als erstes wird dafür der Datensatz, der attached wurde, auch wieder detached

```
R> detach(BBBClub)
```

und die Objekte, die man nicht mehr benötigt

```
R> objects()
```

```
[1] "BBBClub" "amount2" "tab"
```

sollten entfernt werden.

```
R> remove(BBBClub, tab, amount2)
```