LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Statistische Software

Micha Schneider

Institut für Statistik
Ludwig-Maximilians-Universität München

WS 2015/16, R Teil 2

Teil II: Datenanalyse mit R

Import von Daten

Am einfachsten können Textdateien in R eingelesen werden. Folgendes ist zu beachten:

- Struktur: Eine Zeile pro Beobachtung, Merkmale durch eindeutiges Trennzeichen separieren, z. B. Tabulator oder Semikolon.
- Normalerweise eignet sich das Format „Comma Separated Values“ (CSV) am Besten. Auch Excel-Tabellen lassen sich in diesem Format abspeichern.
- Achtung: Je nach Ursprung der Daten kann Punkt oder Komma das Dezimaltrennzeichen sein. Beim Lesen/Schreiben von Daten kann das Dezimaltrennzeichen entsprechend bestimmt werden. In der R-Console wird immer der Punkt verwendet.
- Kontrolle des Formats in einem Editor vor dem Import kann helfen, Fehler zu vermeiden.

Trennzeichen in Zeichenkette

Kommt das Trennzeichen in einer Zeichenkette vor, so muss diese durch einfaches oder doppeltes Hochkomma geschützt werden.

Leerzeichen als Trennzeichen:

```
Name Geburtsjahr Geschlecht
Sting 1951 m
"John Lennon" 1940 m
```

Semikolon: Hochkomma hier nicht notwendig (schadet aber auch nicht):

```
Name;Geburtsjahr;Geschlecht
Sting;1951;m
John Lennon;1940;m
```

Soll auch in der Zeichenkette ein Hochkomma vorkommen, dann am einfachsten jeweils das andere (einfach, doppelt) zum Schützen nehmen. Beides geht auch (durch Backslash schützen).

Münchner Mietspiegel 2003

Zahlreiche deutsche Städte erstellen sogenannte Mietspiegel, um Mietern, Vermietern, Mietberatungsstellen und Sachverständigen eine möglichst objektive Entscheidungshilfe in Mietfragen zur Verfügung zu stellen. Die Mietspiegel werden dabei insbesondere zur Ermittlung der „ortsüblichen Vergleichsmiete" (Nettomiete in Abhängigkeit von Wohnungsgröße, -ausstattung, -alter, etc.) herangezogen. Bei der Erstellung von Mietspiegeln wird aus der Gesamtheit aller in Frage kommenden Wohnungen eine repräsentative Zufallsstichprobe gezogen (im Fall der Stadt München durch Infratest), und die interessierenden Daten werden von Interviewern anhand von Fragebögen ermittelt. Der vorliegende Datensatz stellt einen Ausschnitt aus dem Mietspiegel München des Jahres 2003 dar und enthält die Daten von 2053 Wohnungen.

http://www.stat.uni-muenchen.de/service/datenarchiv/miete/miete03.html ⇒ `miete03.asc`

Einlesen nach R

Vor dem Einlesen sollte das Arbeitsverzeichnis passend gewählt werden:

```
> #setwd("Z:\\REinfuehrung")
> miete <- read.table("Daten/miete03.asc", header=TRUE)
```

Die wichtigsten Argumente:

- **file:** Name der einzulesenden Datei (gegebenenfalls mit Verzeichnis)
- **header:** enthält die erste Zeile die Namen der Variablen?
- **na.strings:** Kodierung der fehlende Werte (z. B. "NA")
- **sep:** Trennzeichen (z. B. "" oder ";" oder "," oder "\t")
- **dec:** Dezimaltrennzeichen (z. B. "." oder ",")
- **skip:** Anzahl der Zeilen, die übersprungen werden bevor die Daten eingelesen werden

Alternativ empfiehlt sich häufig die Verwendung von `read.csv()` oder `read.csv2()`.

Einlesen nach R

Danach unbedingt prüfen, ob der Import erfolgreich war:

```
> names(miete)
 [1] "nm"       "nmqm"     "wfl"      "rooms"    "bj"       "bez"      "wohngut"
 [8] "wohnbest" "ww0"      "zh0"      "badkach0" "badextra" "kueche"
> head(miete)
     nm  nmqm wfl rooms   bj bez wohngut wohnbest ww0 zh0 badkach0 badextra kueche
1 741.4 10.90  68     2 1918   2       1        0   0   0        0        0      0
2 715.8 11.01  65     2 1995   2       1        0   0   0        0        0      0
3 528.2  8.38  63     3 1918   2       1        0   0   0        0        0      0
4 554.0  8.52  65     3 1983  16       0        0   0   0        0        1      0
5 698.2  6.98 100     4 1995  16       1        0   0   0        0        1      1
6 935.6 11.55  81     4 1980  16       0        0   0   0        0        0      0
```

Sind die Variablennamen korrekt erkannt worden?

Einlesen nach R

```
> summary(miete)
       nm              nmqm             wfl             rooms            bj
 Min.   :  77.3   Min.   : 1.47   Min.   : 17.0   Min.   :1.0   Min.   :1918
 1st Qu.: 389.9   1st Qu.: 6.80   1st Qu.: 53.0   1st Qu.:2.0   1st Qu.:1948
 Median : 534.3   Median : 8.47   Median : 67.0   Median :3.0   Median :1960
 Mean   : 570.1   Mean   : 8.39   Mean   : 69.6   Mean   :2.6   Mean   :1958
 3rd Qu.: 700.5   3rd Qu.:10.09   3rd Qu.: 83.0   3rd Qu.:3.0   3rd Qu.:1973
 Max.   :1789.5   Max.   :20.09   Max.   :185.0   Max.   :6.0   Max.   :2001
      bez            wohngut          wohnbest           ww0              zh0
 Min.   : 1.0   Min.   :0.000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.: 5.0   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
 Median :10.0   Median :0.000   Median :0.0000   Median :0.0000   Median :0.0000
 Mean   :11.3   Mean   :0.391   Mean   :0.0219   Mean   :0.0351   Mean   :0.0852
 3rd Qu.:17.0   3rd Qu.:1.000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000
 Max.   :25.0   Max.   :1.000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
    badkach0        badextra         kueche
 Min.   :0.000   Min.   :0.000   Min.   :0.0000
 1st Qu.:0.000   1st Qu.:0.000   1st Qu.:0.0000
 Median :0.000   Median :0.000   Median :0.0000
 Mean   :0.185   Mean   :0.093   Mean   :0.0731
 3rd Qu.:0.000   3rd Qu.:0.000   3rd Qu.:0.0000
 Max.   :1.000   Max.   :1.000   Max.   :1.0000
```

Sind Minimum und Maximum jeder Variablen plausibel?

Einlesen nach R

```
> str(miete)
'data.frame':        2053 obs. of  13 variables:
 $ nm       : num  741 716 528 554 698 ...
 $ nmqm     : num  10.9 11.01 8.38 8.52 6.98 ...
 $ wfl      : int  68 65 63 65 100 81 55 79 52 77 ...
 $ rooms    : int  2 2 3 3 4 4 2 3 1 3 ...
 $ bj       : num  1918 1995 1918 1983 1995 ...
 $ bez      : int  2 2 2 16 16 16 6 6 6 6 ...
 $ wohngut  : int  1 1 1 0 1 0 0 0 0 0 ...
 $ wohnbest: int  0 0 0 0 0 0 0 0 0 0 ...
 $ ww0      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ zh0      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ badkach0: int  0 0 0 0 0 0 0 0 0 0 ...
 $ badextra: int  0 0 0 1 1 0 1 0 0 0 ...
 $ kueche   : int  0 0 0 0 1 0 0 0 0 0 ...
```

Sind kategorische Variablen entsprechend kodiert?

Wir werden später noch sehen, wie wir weiter vorgehen, wenn kategorische Variablen (noch) nicht als solche aufgefasst werden.

Datenmanagement

Auswählen von Fällen oder Variablen:

- Auswahl der Wohnungen, die nach 1970 gebaut wurden und mindestens 70 m^2 Wohnfläche aufweisen:

```
> mieteJungGross <- miete[miete$bj > 1970 & miete$wfl >=70, ]
> dim(miete)
[1] 2053   13
> dim(mieteJungGross)
[1] 313  13
> mieteJungGross2 <- subset(miete, miete$bj > 1970 & miete$wfl >=70)
> dim(mieteJungGross2)
[1] 313  13
```

- Datensatz, der nur Angaben zu Nettomiete, Wohnfläche und Zimmeranzahl enthält:

```
> mieteNmWflRooms <- subset(miete, select=c(nm, wfl, rooms))
> dim(mieteNmWflRooms)
[1] 2053    3
```


Datenmanagement

- Datensatz, der keine Angaben zum Bezirk enthält:

```
> mieteOhneBez <- subset(miete, select=-bez)
> dim(mieteOhneBez)
[1] 2053   12
```

Reproduzierbarkeit: Immer Rohdaten zusammen mit Skript aller notwendigen Transformationen (aus Skriptfenster) speichern! Für weitere Bearbeitung ist zusätzliches Speichern der fertigen Datendatei als .RData-Datei praktisch.

Variablen bearbeiten

Neben dem gesamten Datensatz können auch einzelne Variablen modifiziert werden.

- Erzeugen einer neuen Variable aus bestehenden (oder Umbenennung): Es stehen alle R-Funktionen zur Verfügung, häufig werden jedoch die Grundrechnungsarten und einfache Funktionen wie `log()` oder `exp()` verwendet. Es können auch mehrere Variablen verknüpft werden.

```
> miete$wflProRaum <- miete$wfl / miete$rooms
> summary(miete$wflProRaum)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   12.0    23.5    26.7    27.8    31.0   146.0
```

Variablen bearbeiten

- Konvertiere “numerisch“ zu “Faktor“: Speichermodus auf Faktor (=kategorisch) setzen, Stufen können übernommen oder vorgegeben werden.

```
> miete$badextra <- as.factor(miete$badextra)
> miete$badkach0 <- as.factor(miete$badkach0)
> miete$bez <- as.factor(miete$bez)
> miete$kueche <- as.factor(miete$kueche)
> miete$wohnbest <- as.factor(miete$wohnbest)
> miete$wohngut <- as.factor(miete$wohngut)
> miete$ww0 <- as.factor(miete$ww0)
> miete$zh0 <- as.factor(miete$zh0)
> miete$roomsFaktor <- as.factor(miete$rooms)
```


Variablen bearbeiten

```
> summary(miete)
       nm              nmqm             wfl             rooms            bj
 Min.   :  77.3   Min.   : 1.47   Min.   : 17.0   Min.   :1.0   Min.   :1918
 1st Qu.: 389.9   1st Qu.: 6.80   1st Qu.: 53.0   1st Qu.:2.0   1st Qu.:1948
 Median : 534.3   Median : 8.47   Median : 67.0   Median :3.0   Median :1960
 Mean   : 570.1   Mean   : 8.39   Mean   : 69.6   Mean   :2.6   Mean   :1958
 3rd Qu.: 700.5   3rd Qu.:10.09   3rd Qu.: 83.0   3rd Qu.:3.0   3rd Qu.:1973
 Max.   :1789.5   Max.   :20.09   Max.   :185.0   Max.   :6.0   Max.   :2001

     bez       wohngut   wohnbest ww0       zh0       badkach0 badextra kueche
 9      : 177   0:1250   0:2008   0:1981   0:1878   0:1673   0:1862   0:1903
 2      : 161   1: 803   1:  45   1:  72   1: 175   1: 380   1: 191   1: 150
 5      : 139
 4      : 137
 3      : 132
 25     : 117
 (Other):1190
   wflProRaum    roomsFaktor
 Min.   : 12.0   1:255
 1st Qu.: 23.5   2:715
 Median : 26.7   3:759
 Mean   : 27.8   4:263
 3rd Qu.: 31.0   5: 47
 Max.   :146.0   6: 14
```


Variablen bearbeiten

```
> str(miete)
'data.frame':        2053 obs. of  15 variables:
 $ nm         : num  741 716 528 554 698 ...
 $ nmqm       : num  10.9 11.01 8.38 8.52 6.98 ...
 $ wfl        : int  68 65 63 65 100 81 55 79 52 77 ...
 $ rooms      : int  2 2 3 3 4 4 2 3 1 3 ...
 $ bj         : num  1918 1995 1918 1983 1995 ...
 $ bez        : Factor w/ 25 levels "1","2","3","4",..: 2 2 2 16 16 16 6 6 6 6 ...
 $ wohngut    : Factor w/ 2 levels "0","1": 2 2 2 1 2 1 1 1 1 1 ...
 $ wohnbest   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ ww0        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ zh0        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ badkach0   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ badextra   : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 2 1 1 1 ...
 $ kueche     : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 1 1 ...
 $ wflProRaum : num  34 32.5 21 21.7 25 ...
 $ roomsFaktor: Factor w/ 6 levels "1","2","3","4",..: 2 2 3 3 4 4 2 3 1 3 ...
```


Variablen bearbeiten

- Klassierung einer numerischen Variablen: Gruppenbildung erfolgt durch Anlegen einer neuen kategorischen Variablen (=Faktor), die die Gruppenzuordnung enthält. Die meisten Analysen und Grafiken können getrennt nach den Stufen eines Faktors gerechnet werden.

  Vorgabe der Gruppen:

```
> miete$wflKat1 <- cut(miete$wfl, breaks=c(15, 30, 60, 90, 200))
> summary(miete$wflKat1)
 (15,30]  (30,60]  (60,90] (90,200]
      72      716      918      347
```

  vier Gruppen mit gleicher Länge:

```
> miete$wflKat2 <- cut(miete$wfl, breaks=4)
> summary(miete$wflKat2)
(16.8,59]  (59,101] (101,143] (143,185]
      743      1112       172        26
```

Variablen bearbeiten

vier Gruppen mit annähernd gleicher Häufigkeit (Vorsicht!):

```
> miete$wflKat3 <- cut(miete$wfl, breaks=quantile(miete$wfl, probs=seq(0, 1, 1/4)),
+                  include.lowest=TRUE)
> summary(miete$wflKat3)
 [17,53]  (53,67]  (67,83] (83,185]
     541      489      517      506
```

- Standardisieren: Auf Mittel null und Varianz eins transformieren.

```
> summary(miete$nm)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   77.3   390.0   534.0   570.0   700.0  1790.0
> miete$nmStand <- scale(miete$nm)
> summary(miete$nmStand)
       V1
 Min.   :-2.008
 1st Qu.:-0.734
 Median :-0.146
 Mean   : 0.000
 3rd Qu.: 0.531
 Max.   : 4.969
```


Univariate Deskription

Häufigkeiten

Verteilung der Anzahl der Zimmer:

```
> TableZimmer <- table(miete$rooms)
> TableZimmer
  1   2   3   4   5   6
255 715 759 263  47  14
```

und ihre relativen Häufigkeiten:

```
> TableZimmer/sum(TableZimmer)
       1        2        3        4        5        6
0.124208 0.348271 0.369703 0.128105 0.022893 0.006819
> prop.table(TableZimmer) # Alternative
       1        2        3        4        5        6
0.124208 0.348271 0.369703 0.128105 0.022893 0.006819
```


Lage, Streuung, Quantile

Kennzahlen des numerischen Merkmals Nettomiete:

```
> summary(miete$nm)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   77.3   390.0   534.0   570.0   700.0  1790.0
> sd(miete$nm)
[1] 245.4
```

Kennzahlen sind auch einzeln verfügbar:

```
> min(miete$nm)
[1] 77.31
> max(miete$nm)
[1] 1790
> mean(miete$nm)
[1] 570.1
> median(miete$nm)
[1] 534.3
> quantile(miete$nm, prob=0.75)
  75%
700.5
```

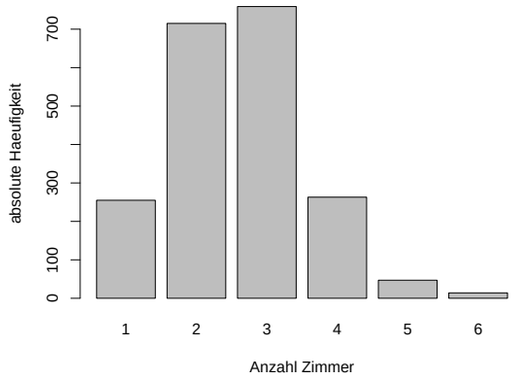
Schichtung

Kennzahlen eines numerischen Merkmals geschichtet nach einem kategorischen Merkmal:

```
> aggregate(x=miete$nm, by=list(miete$roomsFaktor), FUN=summary)
  Group.1 x.Min. x.1st Qu. x.Median x.Mean x.3rd Qu. x.Max.
1       1  106.0     293.0    344.0  347.0     393.0 1290.0
2       2   77.3     374.0    487.0  487.0     590.0 1470.0
3       3  145.0     473.0    618.0  634.0     770.0 1790.0
4       4  193.0     551.0    723.0  748.0     920.0 1750.0
5       5  328.0     625.0    917.0  901.0    1150.0 1660.0
6       6  622.0     781.0    857.0  999.0    1300.0 1540.0
```

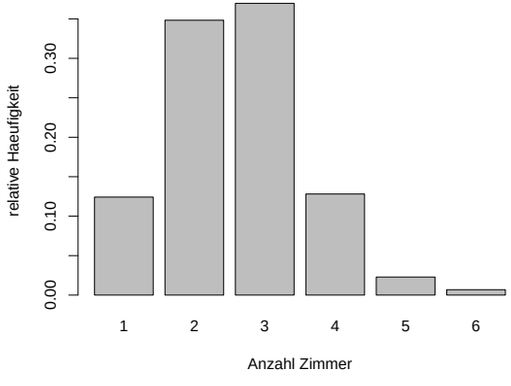

Balkendiagramm: Anzahl Zimmer

```
> TableZimmer <- table(miete$rooms)
> barplot(TableZimmer, xlab="Anzahl Zimmer", ylab="absolute Haeufigkeit")
```



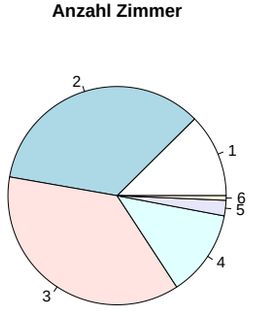
Balkendiagramm: Relative Häufigkeiten

```
> barplot(TableZimmer/sum(TableZimmer), xlab="Anzahl Zimmer", ylab="relative Haeufigkeit")
```



Kreisdiagramm: Anzahl Zimmer

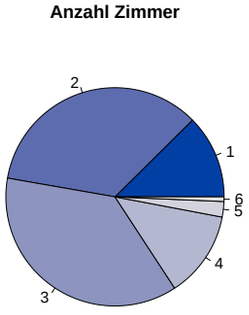
```
> pie(TableZimmer, main="Anzahl Zimmer")
```



Schönere Farben erhält man mit Hilfe des Packages colorspace, zum Beispiel mit den Funktionen `sequential_hcl()`, `diverge_hcl()`, `rainbow_hcl()`, `heat_hcl()` oder `terrain_hcl()`.

Kreisdiagramm: Anzahl Zimmer

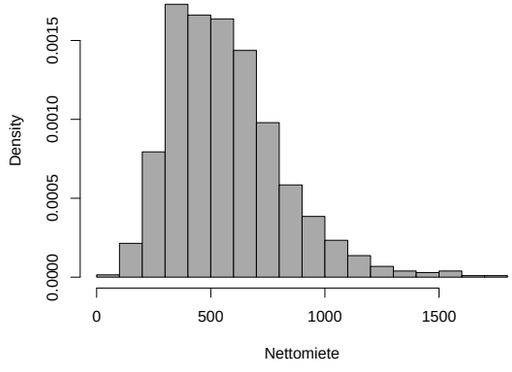
```
> library("colorspace")
> pie(TableZimmer, main="Anzahl Zimmer", col=sequential_hcl(6))
```



Histogramm: Nettomiete

Automatische Klassenanzahl:

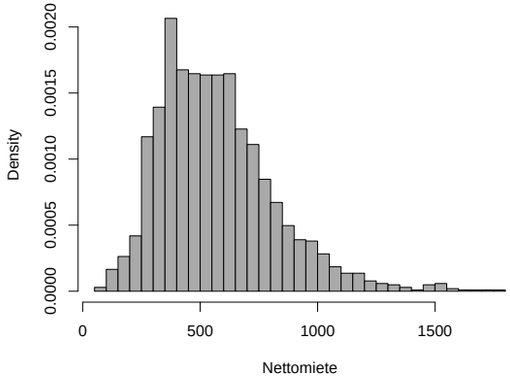
```
> hist(miete$nm, col="darkgray", freq=FALSE, main="", xlab="Nettomiete")
```



Histogramm: Nettomiete

Vorgegebene Klassenanzahl von 50 (wird nur ungefähr eingehalten):

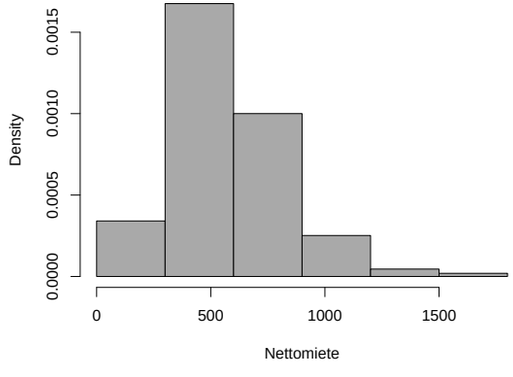
```
> hist(miete$nm, breaks=50, col="darkgray", freq=FALSE, main="", xlab="Nettomiete")
```



Histogramm: Nettomiete

Direkte Vorgabe der Klassen:

```
> hist(miete$nm, breaks=c(0, 300, 600, 900, 1200, 1500, 1800), col="darkgray",
+ freq=FALSE, main="", xlab="Nettomiete")
```



Multivariate Darstellungen

Balkendiagramme für Kontingenztafeln

Anzahl der Zimmer je nachdem, ob die Wohnfläche mehr als 80 m^2 beträgt:

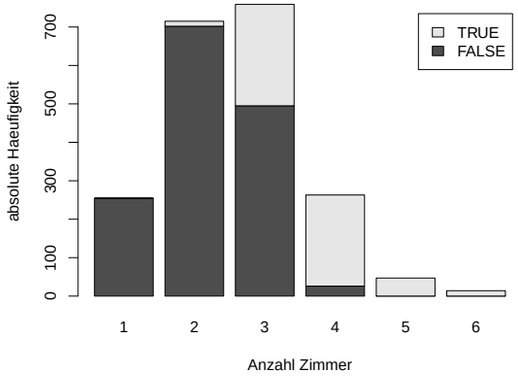
```
> TAB <- table(miete$wfl>80, miete$rooms)
> TAB
         1   2   3   4   5   6
  FALSE 254 702 495  26   0   0
  TRUE    1  13 264 237  47  14
```

Die Anteile je Zeile bzw. Spalte erhhält man durch:

```
> prop.table(TAB, margin=1)
              1        2        3        4        5        6
  FALSE 0.171970 0.475288 0.335139 0.017603 0.000000 0.000000
  TRUE  0.001736 0.022569 0.458333 0.411458 0.081597 0.024306
> TAB2 <- prop.table(TAB, margin=2)
> TAB2
              1        2        3        4        5        6
  FALSE 0.996078 0.981818 0.652174 0.098859 0.000000 0.000000
  TRUE  0.003922 0.018182 0.347826 0.901141 1.000000 1.000000
```

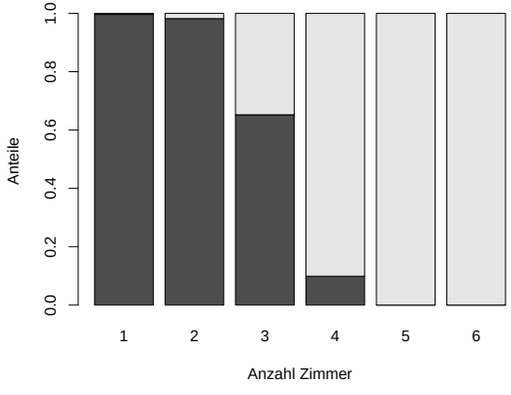

Balkendiagramme für Kontingenztafeln

```
> barplot(TAB, legend=TRUE, xlab="Anzahl Zimmer", ylab="absolute Haeufigkeit")
```



Balkendiagramme für Kontingenztafeln

```
> barplot(TAB2, xlab="Anzahl Zimmer", ylab="Anteile")
```



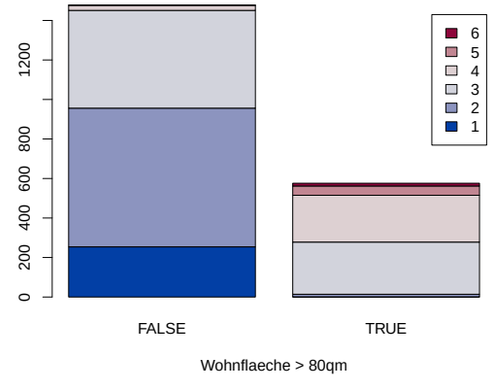
Balkendiagramme für Kontingenztafeln

Die Funktion `t()` transponiert eine Matrix:

```
> TAB
          1   2   3   4   5   6
  FALSE 254 702 495  26   0   0
  TRUE    1  13 264 237  47  14
> t(TAB)
     FALSE TRUE
  1    254    1
  2    702   13
  3    495  264
  4     26  237
  5      0   47
  6      0   14
```

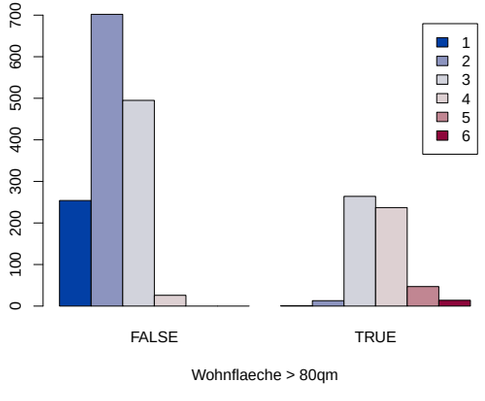

Balkendiagramme für Kontingenztafeln

```
> barplot(t(TAB), legend=TRUE, col=diverge_hcl(6), xlab="Wohnflaeche > 80qm")
```



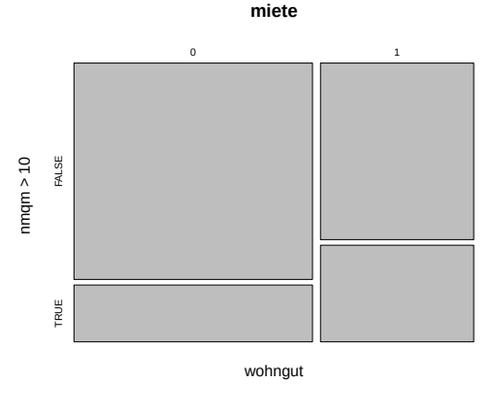
Balkendiagramme für Kontingenztafeln

```
> barplot(t(TAB), beside=TRUE, legend=TRUE, col=diverge_hcl(6), xlab="Wohnflaeche > 80qm")
```



Mosaicplot: Miete, Lage

```
> mosaicplot(~wohngut + (nmqm > 10), data = miete)
```



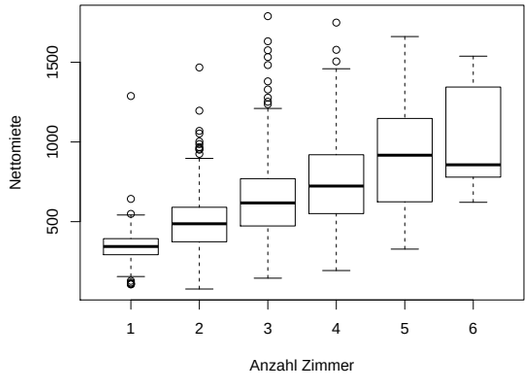
Mosaicplot: Miete, Lage

Welche Zahlen stehen hinter den Flächen des Mosaicplots?

```
> table(miete$nmqm>10, miete$wohngut)
         0   1
  FALSE 990 519
  TRUE  260 284
```

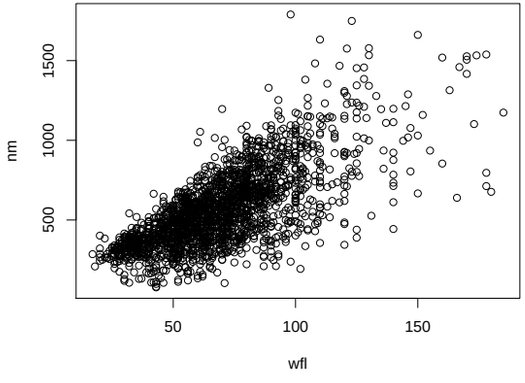

Boxplot: Nettomiete nach Anzahl Zimmer

```
> boxplot(nm ~ rooms, xlab="Anzahl Zimmer", ylab="Nettomiete", data=miete)
```



Streudiagramm: Nettomiete und Fläche

```
> plot(nm ~ wfl, data=miete)
> # plot(x=miete$wfl, y=miete$nm) # Alternative
```



Argumente für Grafikfunktionen

Eine ausführliche Hilfe zu den Graphikparametern findet man mit `?par()`. Hier eine Zusammenfassung von wichtigen Argumenten (Auswahl):

Argument	Beschreibung
`main`	Überschrift der Grafik
`xlab, ylab`	Name(„Label“) der x-/y-Achsen
`xlim, ylim`	Vektor mit Minimum/Maximum für zu plottenden Bereich in x-/y-Richtung
`cex`	Größe
`cex.main, cex.axis, cex.lab`	Größe der Überschrift, Achsenbeschriftung und -namen
`col`	Farbe der Objekte in der Plot-Region Zahlen: 1(Schwarz), 2(rot),... Übersicht mit `palette()` Namen: “black“, “red“,... Übersicht mit `colours()`
`lty`	Linientyp (z.B. gestrichelt,...)
`lwd`	Linienbreite
`type`	“p“: Punkte, “l“: Linien, “n“: Nichts einzeichen

Elemente hinzufügen

Manchmal möchte man Elemente zu einer bestehenden Graphik hinzufügen. Z. B. könnte man den Mittelwert mit `abline()` in ein Histogramm einzeichnen.

Funktion	Beschreibung
`points()`	Punkte an Stellen (x, y)
`lines()`	Linien zwischen den Stellen (x, y)
`segments()`	Liniensegmente
`arrows()`	ähnlich wie `segements`, aber mit Pfeilspitzen
`text()`	Text
`title()`	Beschriftung
`axis()`	Achsen hinzufügen; x-Achse: `side="1"`, y-Achse: `side="2"`
`abline()`	Eine oder mehrere Geraden
`grid()`	Gitternetz

Erweiterungspakete

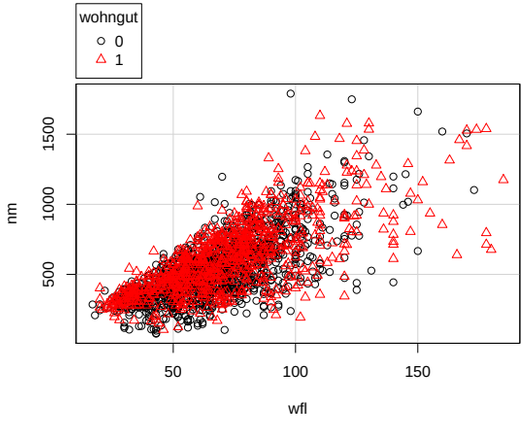
Es existieren noch weitere Pakete, die erweiterte Funktionalität bieten und ansprechende Graphiken erzeugen können, z.B.:

- car
- lattice
- ggplot2

Um diese Funktionen nutzen zu können, müssen sie mit `library()` geladen werden.

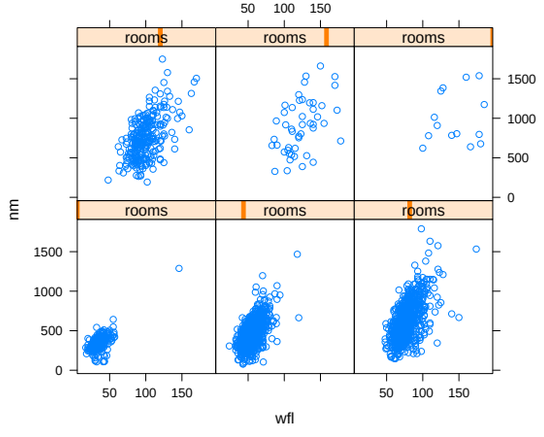
Streudiagramm: Miete, Fläche, Lage

```
> library("car")
> scatterplot(nm ~ wfl | wohngut, reg.line=FALSE, smooth=FALSE, data=miete)
```



Streudiagramm: Miete, Fläche, Zimmer

```
> library("lattice")
> xyplot(nm ~ wfl | rooms, data=miete)
```



Korrelation

Korrelation nach Pearson:

```
> cor(miete[,c("nm","nmqm","wfl")], use="complete.obs", method="pearson")
         nm    nmqm     wfl
nm   1.0000  0.4748  0.7075
nmqm 0.4748  1.0000 -0.2268
wfl  0.7075 -0.2268  1.0000
```

Korrelation nach Spearman:

```
> cor(miete[,c("nm","nmqm","wfl")], use="complete.obs", method="spearman")
         nm    nmqm     wfl
nm   1.0000  0.4611  0.6971
nmqm 0.4611  1.0000 -0.2304
wfl  0.6971 -0.2304  1.0000
```

Ein Test auf die Korrelation zwischen zwei Variablen kann mit `cor.test` durchgeführt werden.

Mittlere Miete nach 2 kat. Merkmalen

```
> tapply(miete$nm, list(rooms=miete$rooms, wohngut=miete$wohngut), mean, na.rm=TRUE)
     wohngut
rooms      0     1
    1  332.1 366.5
    2  459.8 528.9
    3  592.9 701.4
    4  714.0 809.1
    5  854.4 944.8
    6 1266.0 954.4
```

6-Zimmer-Wohnungen in schlechterer Lage teurer?

Kreuztabelle bei 2 Merkmalen

```
> (TabelleZimmerWohngut <- xtabs(~ rooms + wohngut, data=miete))
     wohngut
rooms   0   1
    1 145 110
    2 436 279
    3 475 284
    4 169  94
    5  23  24
    6   2  12
```


Lineare Regression

```
> RegModel1 <- lm(nm ~ wfl, data=miete)
> summary(RegModel1)
Call:
lm(formula = nm ~ wfl, data = miete)

Residuals:
   Min     1Q Median     3Q    Max
-655.2  -97.4    7.4   98.7 1023.4

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   89.847     11.264    7.98  2.5e-15 ***
wfl            6.901      0.152   45.33  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 174 on 2051 degrees of freedom
Multiple R-squared:  0.501,        Adjusted R-squared:   0.5
F-statistic: 2.06e+03 on 1 and 2051 DF,  p-value: <2e-16
```

Dokumente erstellen

Grafiken

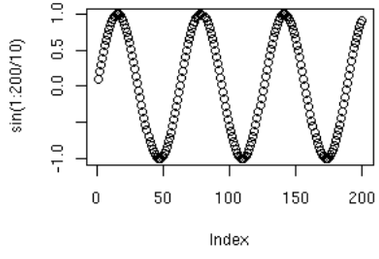
Es gibt zwei grundsätzlich verschiedene Arten, Grafiken zu speichern:

Bitmap: In einem rechteckigen Gitter wird für jeden Gitterpunkt gespeichert, welche Farbe er hat. Reskalierungen reduzieren Qualität.
Bekannte Formate: JPEG, BMP, PNG, GIF

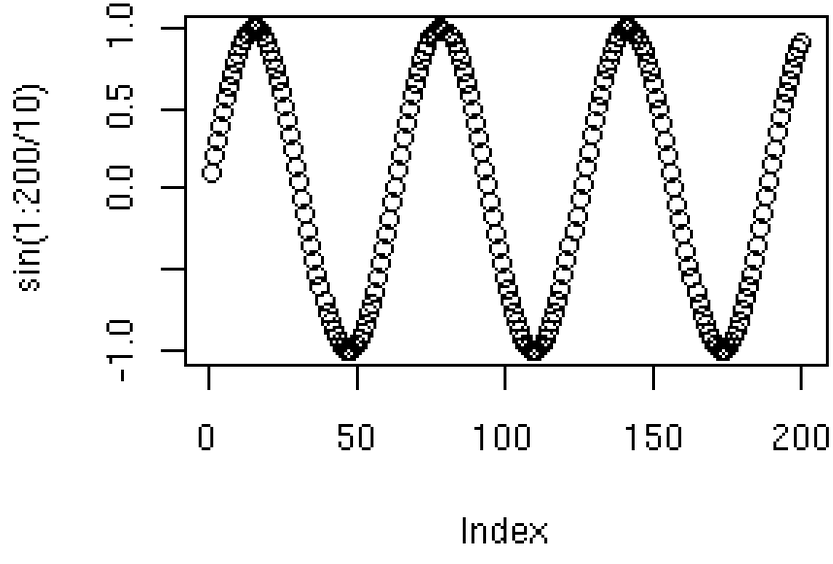
Vektor-Grafik: Eine logische Beschreibung der Grafik, ohne Qualitätsverlust reskalierbar.
Bekannte Formate: EMF, WMF, EPS, PDF

Der große Vorteil der Vektor-Grafiken gegenüber Bitmap-Grafiken liegt in der Skalierbarkeit ohne Qualitätsverslust (d. h. die Bilder werden nicht pixelig). Bitmap-Grafiken lassen sich dafür oft einfacher in andere Software einbinden und betrachten. Ein PDF-Bild kann man in Word nur als „Objekt“, aber nicht als „Grafik“ einbetten.

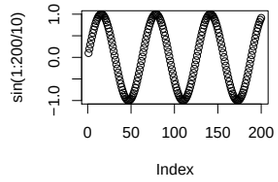
Bitmap vs. Vektorgrafik



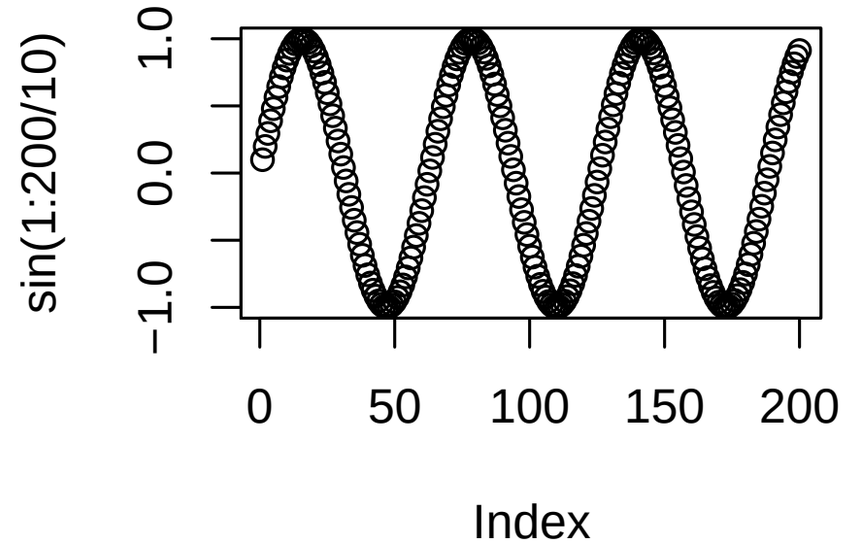
Bitmap vs. Vektorgrafik



Bitmap vs. Vektorgrafik



Bitmap vs. Vektorgrafik



Grafiken

Die in R erzeugten Grafiken können nicht nur über das Menü, sondern direkt durch eine Anweisung im Code im gewünschten Format abgespeichert werden, was gerade bei einer größeren Anzahl von Grafiken deutlich effizienter ist. Beispielsweise lässt sich durch

```
> pdf("graphics/Grafik1.pdf", width=7, height=7)
> boxplot(nm ~ rooms, ylab = "Nettomiete", xlab = "Anzahl Zimmer",
+     data = miete)
> dev.off() # Erst hier wird die Grafik gespeichert!
```

der erzeugte Boxplot als .pdf-Datei namens „Grafik1.pdf“ im Unterordner „graphics“ abspeichern. Weitere Funktionen neben `pdf()` sind beispielsweise `postscript()`, `win.metafile()`, `jpeg()`, `png()`, `tiff()` und `bmp()`.

Neben der Größe der Grafiken können meist noch etliche weitere Parameter variiert werden.

Textverarbeitung

Wenn Ausgaben aus der R-Konsole in ein Textverarbeitungsprogramm (z. B. Word) kopiert werden, ist es meist sinnvoll, diese in einer Schriftart mit fixen Zeichenabständen (Nicht-Proportionalschrift) zu formatieren (z.B. Courier, Typewriter, . . .):

Times New Roman (Proportionalschrift):

```
TAB <- table(miete$wfl>80, miete$rooms)
> TAB

      1  2  3  4  5  6
FALSE 254 702 495  26  0  0
TRUE   1  13 264 237  47 14
```

Courier New (Nicht-Proportionalschrift):

```
TAB <- table(miete$wfl>80, miete$rooms)
> TAB

          1   2   3   4   5   6
FALSE 254 702 495  26   0   0
TRUE    1  13 264 237  47  14
```

Struktur einer R-Sitzung

- Starten von R und eines Editors
- Arbeitsverzeichnis mit `getwd()` überprüfen und mit `setwd()` ändern
- Daten z. B. mit `load()` oder `read.table()` in den workspace laden
- Analyse durchführen (ggf. Pakete mit `library()` laden)
- wichtige Ergebnisse speichern - z. B. mit `save()` oder `write.table()`
- R-Sitzung z. B. mit `q()` beenden
- Ergebnis: Quellcode in einer Textdatei(.R), Daten in einer .RData-Datei und Graphiken im entsprechendem Verzeichnis

Hausübung

In der Hausübung arbeitet jede/r mit einem eigenen Datensatz. Dieser muss durch eine Zufallsstichprobe aus einem Datensatz (data_all) erzeugt werden. Dazu verwendet jede/r die eigene (!) Matrikelnummer als Seed. Der übrige R-Code kann so übernommen werden:

```
> matrikelnummer <-  11111111
> set.seed(matrikelnummer)
> lines_sample <- sample(x=1:nrow(data_all), size=200, replace = FALSE, prob = NULL)
> data <- data_all[lines_sample,]
```

Mit `sample()` werden 200 verschiedene Zahlen (zwischen 1 und 456) zufällig gezogen. Anschließend werden die den Zahlen entsprechenden Beobachtungszeilen im Datensatz ausgewählt. Der so erstellte individuelle (Teil-)Datensatz muss bei der Bearbeitung der Hausübung bei allen drei Software-Paketen verwendet werden. Die Anforderungen an die Daten können sich je nach Software/Plattform etc. unterscheiden (z. B. Dezimaltrennzeichen). Deshalb immer den gespeicherten Datensatz verwenden, der richtig funktioniert.