R und R-Commander

sowie

R-Befehle zur Vorlesung (107.238) und Übung (107.248 und 107.249)

Leonhard Seyfang

seyfang@utanet.at

(Tutor für die EDV-Übungen)

Institut für Statistik und Wahrscheinlichkeitstheorie

Technische Universität Wien

Wien, am 5. Oktober 2005

Inhaltsverzeichnis

1 Grundlegende Befehle und Hinweise 3
1.1 R installieren . . . 3
1.2 RGui . . . 3
1.3 Daten laden, einlesen und ausgeben . . . 3
1.4 Hilfe und Suche . . . 4
1.5 Zuweisung . . . 5
1.6 Operatoren und Vergleiche . . . 5
1.7 Konstante . . . 5
1.8 Elementare Funktionen . . . 5
1.9 Datentypen und -strukturen . . . 6
1.10 Grafiken . . . 8
1.11 Indizieren . . . 9
1.12 Formelnotation . . . 10
1.13 Sweave . . . 10

2 Der R-Commander 11
2.1 Allgemeines . . . 11
2.2 Beispielsitzung . . . 11

3 R-Befehle zur Vorlesung (107.238) 15
3.1 Histogramm . . . 15
3.2 Ortsparameter . . . 15
3.3 Streuungsmaße . . . 16
3.4 Höhere Momente . . . 16
3.5 Boxplots . . . 17
3.6 Empirische Verteilungsfunktion . . . 17
3.7 Verteilungen . . . 18
3.8 Quantile-Quantile Plots . . . 19
3.9 Abhängigkeitsmaße . . . 20
3.10 Tests . . . 20
3.11 Varianzanalyse . . . 21
3.12 Regression . . . 21
3.13 Zweifache Klassifizierung (Kontingenztafel) . . . 22

4 R-Befehle zur Übung (107.248 und 107.249) 23

Literatur 24

1 Grundlegende Befehle und Hinweise

1.1 R installieren

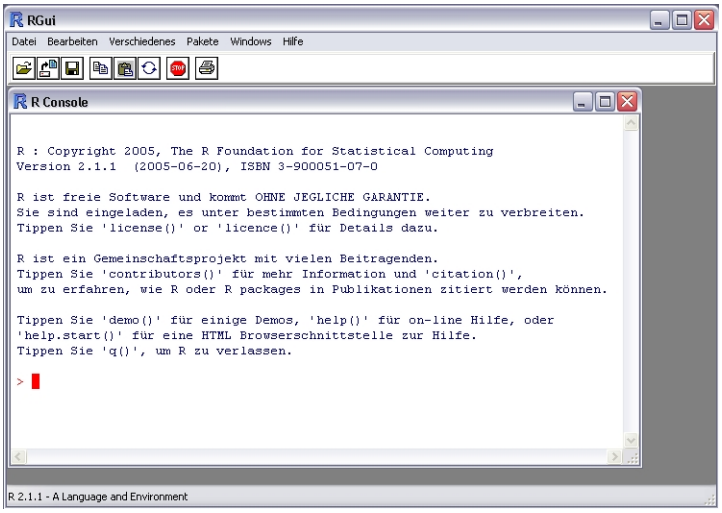
R ist *Open Source* Software (unter *GPL*[1] lizensiert) und kann von

`http://www.r-project.org/`

geladen werden. Dort stehen außerdem zahlreiche Zusatzpakete und Dokumentationen, vor allem die Einführung in R ([1]) zur Verfügung.

1.2 RGui

Unter Windows steht ein sehr einfaches *GUI* zur Verfügung.



1.3 Daten laden, einlesen und ausgeben

`data(Prestige)`

lädt den Datensatz *Prestige*. (Eine Liste von vorhandenen Daten, z.B. im package *datasets*, bekommt man mit `help(package = datasets)`.) Der Datensatz enthält ein Objekt der Klasse *data.frame* mit mehreren Spalten, z.B.: `education`, `income` und `prestige`. Diese können mit:
`objektname$spaltenname` angesprochen werden, also z.B.:
`Prestige$education`.

`attach(Prestige)`

hängt das Objekt an den Suchpfad an. Die einzelnen Spalten können jetzt direkt angesprochen werden, z.B.: `education`.

[1] General Public License `http://www.gnu.org/copyleft/gpl.html`

`read.table()`

`a <- read.table("Filename.txt", header = TRUE)` liest die angegebene Datei und speichert den Inhalt in `a` ab. Das Argument `header = TRUE` bewirkt, dass der Inhalt der ersten Zeile als Spaltenbeschriftung verwendet wird. Dabei wird davon ausgegangen, dass sich die Datei im *Working Directory* befindet. Mit `getwd()` kann man den Namen des aktuellen Working Directories abfragen und mit `setwd("Pfad")` ändern. (Im Windows-Gui kann man dafür das Menü DATEI → VERZEICHNIS WECHSELN... verwenden.

`read.csv2()`

`a <- read.csv2("Filename.csv")` liest die angegebene Datei und speichert den Inhalt in `a` ab.

Eine *csv*-Datei kann man z.B. mit *Excel* erzeugen: DATEI → SPEICHERN UNTER... → Dateityp: csv.

Für Dateien, die mit der englischen *Excel*-Version erzeugt wurden, verwendet man `read.csv()` statt `read.csv2()`.

`write.table()`

`write.table(a, file = "Filename.txt", quote = FALSE, row.names = FALSE)` schreibt den Inhalt von `a` in die Datei `Filename.txt`.

`sink("Filename")`

leitet die Ausgabe auf eine Datei mit dem angegebenen Namen um. Mit `sink()` wird die Ausgabe wieder zurückgesetzt.

1.4 Hilfe und Suche

`help.start()`

öffnet eine umfangreiche *HTML*-Hilfe.

`help(foo)`

oder `?foo` sucht nach einem Befehl namens *foo*. Gefunden werden aber nur Befehle von Paketen, die zur Zeit geladen sind!

`library()`

gibt eine Liste der zur Zeit installierten (nicht unbedingt geladenen) Pakete.

`help(package = car)`

öffnet die Hilfeseite zum Paket *car*. Hier werden alle Befehle und Datensätze, die im Paket enthalten sind, aufgelistet.

```
library(car)
```

lädt das Paket *car*.

```
help.search("fooo")
```

durchsucht die Hilfeseiten auf ähnliche Textstellen.

1.5 Zuweisung

```
x <- a
```

weist dem Objekt `x` den Wert `a` zu. (Statt „`<-`“ ist auch „=“ möglich.)

Mit der Eingabe des Variablennamen und Betätigung der Eingabetaste kann man den Wert der Variablen anzeigen lassen.

Die Zeichen: >␣ vor den R-Befehlen entsprechen dem *Prompt* (der *Eingabeaufforderung*).

```
> x <- 15.75
> x
```

```
[1] 15.75
```

1.6 Operatoren und Vergleiche

`+`	Addition	`!=`	ungleich
`-`	Subtraktion	`>`	größer
`*`	Multiplikation	`>=`	größer gleich
`/`	Division	`<`	kleiner
`^`	Potenz	`<=`	kleiner gleich
`==`	gleich	`!`	Negation

1.7 Konstante

`pi`	die Zahl π
`Inf`	(*Infinity*) unendlich
`-Inf`	-unendlich
`NaN`	(*Not a Number*) nicht definiert
`Na`	(*Not available*) fehlender Wert
`NULL`	leere Menge
`TRUE`	wahr
`FALSE`	falsch

1.8 Elementare Funktionen

`sin(x)`, `cos(x)`, `tan(x)`, `sqrt(x)`, `log(x)`, `exp(x)`, `abs(x)`, `min(x)` und `max(x)` funktionieren in gewohnter Weise.

1.9 Datentypen und -strukturen

R arbeitet mit benannten Datenstrukturen. Die wichtigsten werden in der nachfolgenden Tabelle vorgestellt:

`vector`	Ein *Vektor* kann aus beliebig vielen Elementen bestehen, diese müssen aber alle vom selben Datentyp sein (z.B.: *logical*, *numeric* oder *character*).
`factor`	Dieser Datentyp dient zum Speichern von *kategoriellen* Daten.
`matrix`	Genau wie beim Vektor müssen auch die Elemente von Matrizen vom selben Datentyp sein.
`data.frame`	Ein *Datensatz* ist eine Matrix-artige Datenstruktur, wobei aber die *Spalten* verschiedenen Datentypen angehören können. Die Spalten können auch wie Listenelemente angesprochen werden.
`list`	Eine *Liste* ist ein Vektor, wobei die einzelnen Elemente verschiedenen Datentypen angehören können. Ein Element einer Liste kann z.B. selbst auch wieder eine Liste sein.

`c()`

(*combine* oder *concatenate*) erzeugt einen Vektor, indem die übergebenen Argumente miteinander verknüpft werden. Die Argumente können selbst auch Vektoren sein.

`a:b`

erzeugt einen Vektor mit den Werten von `a` bis `b` in Stufen von 1.
(Mehr Möglichkeiten bieten die Befehle `seq` und `rep`.)

`factor()`

wandelt einen als Argument übergebenen Vektor in ein Objekt vom Datentyp *factor* um.

`matrix(Inhalt, Zeilenanzahl, Spaltenanzahl)`

erzeugt eine Matrix.

`data.frame()`

verbindet die als Argumente übergebenen Objekte, die alle dieselbe Länge haben müssen, zu einem Datensatz.

`list()`

verbindet die übergebenen Objekte zu einer Liste.

Beispiele:

```
> v <- c(6, 9, 13)
> v

[1]  6  9 13

> v2 <- v > 10
> v2

[1] FALSE FALSE  TRUE

> f <- factor(c("grün", "rot", "grün"))
> f

[1] grün rot  grün
Levels: grün rot

> m <- matrix(1:6, 2, 3)
> m

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

> df <- data.frame(v, v2, f)
> df

   v    v2    f
1  6 FALSE grün
2  9 FALSE  rot
3 13  TRUE grün
```

`length()`

gibt die *Länge* eines Vektors zurück.

`class()`

informiert über die *Klasse* des Objekts.

`summary()`

erstellt eine Zusammenfassung des übergebenen Objekts (abhängig von der Klasse des Objekts).

`str()`

gibt einen Überblick über die *Struktur* des Objekts.

1.10 Grafiken

`curve(foo, a, b)`

zeichnet den Grafen der gegebenen Funktion *foo* im Intervall $[a, b]$.

`lines(x, y)`

verbindet die durch die Koordinaten `x` und `y` gegebenen Punkte mit einem Polygon.

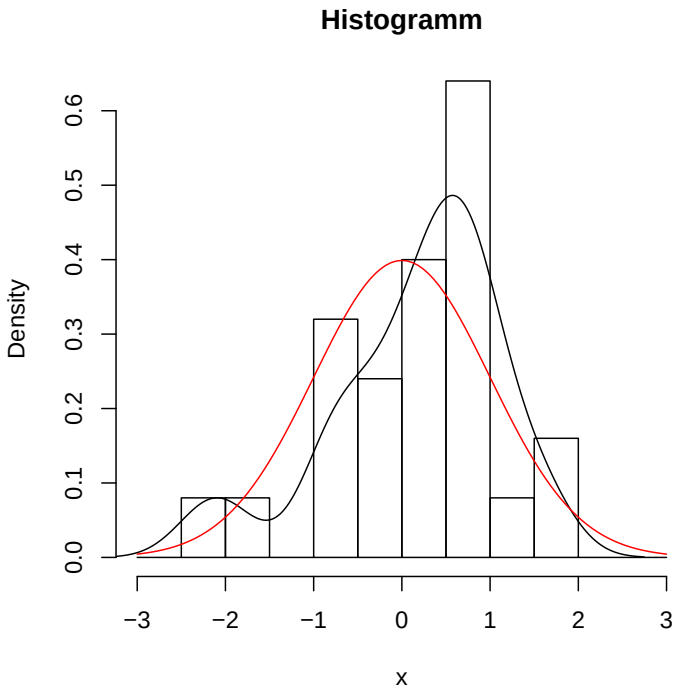
`lines(density())`

berechnet und zeichnet eine Dichteschätzung für die übergebenen Daten.

`par()`

setzt Parameter für die Grafikausgabe.

```
> set.seed(1)
> x <- rnorm(25)
> par(mar = c(4, 4, 3, 1))
> hist(x, freq = FALSE, breaks = (-6:6)/2, main = "Histogramm")
> lines(density(x))
> curve(dnorm, -3, 3, add = TRUE, col = "red")
```




```
pdf("Filename.pdf")
```

leitet die Grafikausgabe auf eine Datei (im Format *pdf*) mit dem angegebenen Namen um. Mit `dev.off()` wird die Ausgabe in die Datei beendet. Analog dazu funktionieren `jpeg()`, `postscript()` und andere Dateiformate (siehe *Hilfe*).

1.11 Indizieren

Um einen Teil eines Vektors, einer Matrix oder eines beliebig-dimensionalen Arrays anzusprechen, wird nach dem Variablennamen ein bzw. mehrere *Index-Vektoren* in eckiger Klammer geschrieben.

Index-Vektoren können folgenden Typen angehören:

1. *Logischer Vektor*: Die Elemente bei denen der Index-Vektor *TRUE* ergibt, werden genommen. Z.B. `x[x>5]`: alle Elemente aus `x` deren Wert größer als 5 ist.

2. *Vektor aus positiven ganzen Zahlen*: Die Elemente mit den entsprechenden Indizes werden genommen. Z.B. `x[3:5]`.

3. *Vektor aus negativen ganzen Zahlen*: Alle Elemente außer den angegebenen werden genommen. Z.B. `x[c(-5,-7)]`: Alle Elemente außer auf Position 5 und 7.

4. *Vektor aus Zeichenketten (Strings)*. Elemente mit entsprechenden Namen werden ausgewählt. Z.B. `x[c("foo1","foo2")]`.

Beispiel:

```
> m

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

> m[2, -2]

[1] 2 6
```

1.12 Formelnotation

Die Formelnotation dient zur einfachen Spezifikation von statistischen Modellen. Die allgemeine Form ist:

$$\texttt{y} \sim \texttt{model}.$$

Dabei ist y die *abhängige* Variable. In *model* werden eine oder mehrere unabhängige Variable spezifiziert. Diese können mittels mathematischer Symbole verknüpft werden. Die wichtigsten Operationen sind:

+	Hinzunahme einer Variablen
:	Wechselwirkung von Variablen
*	Hinzunahme von Variablen und deren Wechselwirkung
.	alle Variablen aus dem Datensatz aufnehmen

Beispiel:

$$\texttt{y} \sim \texttt{x1 + x2 + x1:x2}$$

ist äquivalent zu

$$\texttt{y} \sim \texttt{x1 * x2}.$$

Für detailiertere Informationen siehe *Hilfe*.

1.13 Sweave

```
Sweave("Filename.Rnw")
```

Mit Sweave können R- und LaTeX-Code kombiniert werden[2]. Durch den Aufruf von Sweave wird aus einer *.Rnw*-Datei eine *.tex*-Datei generiert, wobei der R-Code durch die Ausgabe von R (Text, Grafiken, Tabellen) ersetzt wird. Siehe [2].

[2]Dieses Dokument wurde z.B. so erstellt.

2 Der R-Commander

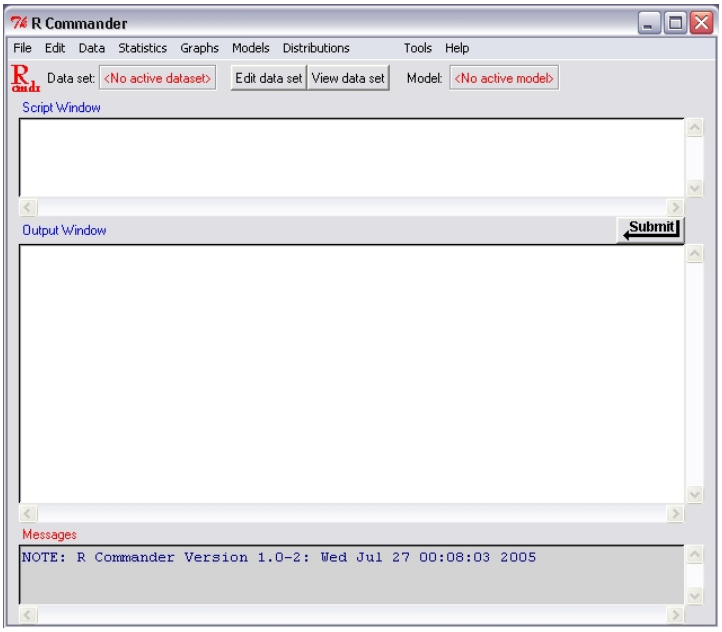
2.1 Allgemeines

Für eine einfache Funktionalität steht mit dem R-*Commander* eine grafische Benutzeroberfläche zur Verfügung. Das dazugehörende Paket muss wie jedes andere Zusatzpaket geladen werden (`library(Rcmdr)`). Falls der Commander während einer R-Sitzung geschlossen wurde, kann er mit `Commander()` wieder geöffnet werden.

Der Commander besteht aus einer Menüleiste, einer Werkzeugleiste, einem *Script*-Fenster, einem Ausgabe-Fenster und einem Nachrichten-Fenster. Grafiken werden jedoch nach wie vor in einem eigenen Fenster des RGui ausgegeben. Man muss also je nach Bedarf zwischen den verschiedenen Fenstern wechseln.)

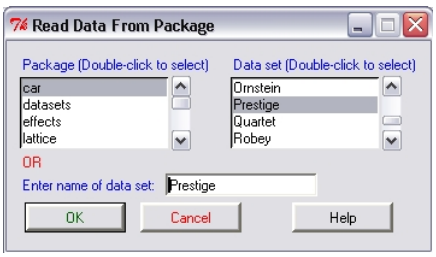
Die R-Befehle im Script-Fenster werden im wesentlichen automatisch generiert, indem man sich durch das Menü „klickt“. Diese Kommandos können aber auch nachträglich geändert oder überhaupt manuell eingegeben werden.

Mit der Schaltfläche Submit wird der vorher mit der Maus markierte Text, oder falls nichts markiert ist, die Zeile in der sich der Cursor befindet, ausgeführt.



2.2 Beispielsitzung

Mit: Data → Data in packages → Read data set from an attached package wird ein Dialogfenster zum Auswählen eines Datensatzes geöffnet:



Hier wird zuerst das Package *car* und dann der Datensatz *Prestige* ausgewählt und mit OK bestätigt.[3]

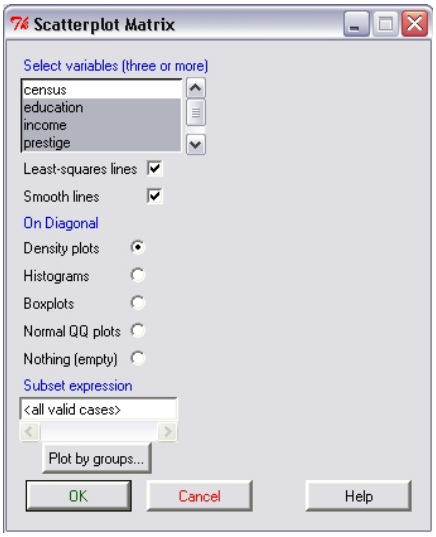
In der Werkzeugleiste scheint jetzt neben *Data set:* der Name des aktiven Datensatzes auf.

Mit Edit data set kann man die Daten direkt manipulieren und mit View data set betrachten.

Mit: Statistics → Summaries → Active data set kann man sich einen Überblick über die verschiedenen Variablen verschaffen.

Angenommen, es soll die Abhängigkeit zwischen *education*, *income* und *prestige* grafisch dargestellt werden, und zwar abhängig von der kategoriellen Variablen *type*.

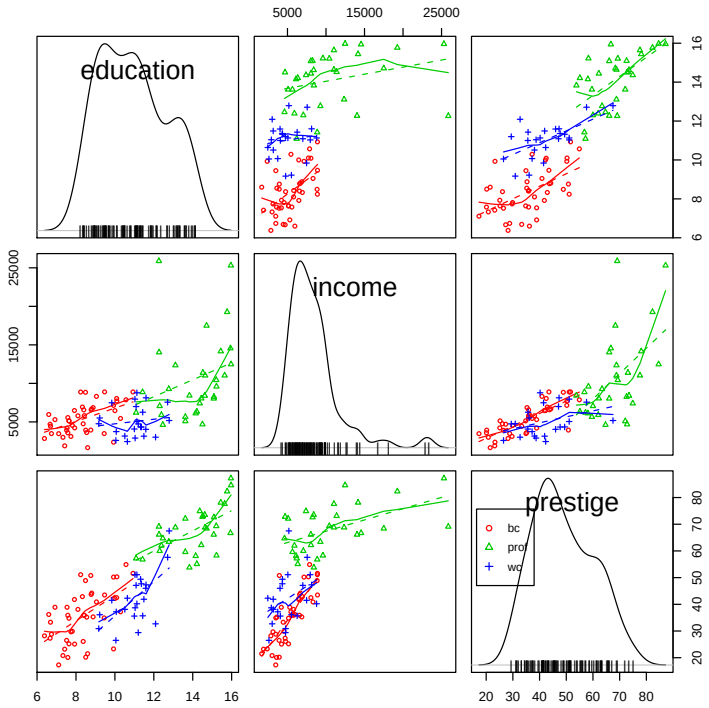
Mit: Graphs → Scatterplot matrix... bekommt man folgendes Fenster:



Hier wählt man die drei Variablen aus, mit Plot by groups... kann man die Gruppierungsvariable angeben. In diesem Fall steht nur *type* zur Auswahl, die Zeile muß aber trotzdem im Auswahlfenster markiert werden.

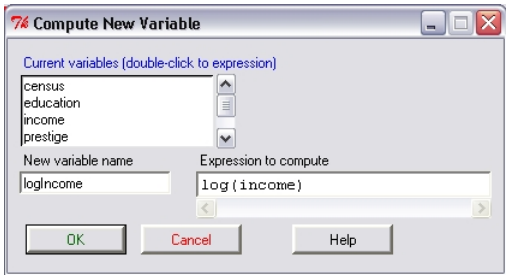
[3]Der Datensatz beschreibt das Einkommen und verschiedene soziale Einflussfaktoren in Kanada 1971.

Mit OK beendet man zuerst das Fenster *Groups* und dann das Fenster *Scatterplot Matrix*. Im aktiven Grafikausgabefenster erscheint folgende Grafik:



In den Diagonalfeldern befinden sich *Dichteschätzungen* für die jeweilige Variable. Die Farben bzw. unterschiedlichen Symbole der Datenpunkte entsprechen den drei Gruppen der Gruppierungsvariablen *type*.

Wie man an der Dichte der Variable *income* sieht, handelt es sich hierbei um eine rechtsschiefe (asymmetrische) Verteilung. Dies soll durch eine *Log-Transformation* ausgeglichen (bzw. vermindert) werden. Dazu wird eine neue Variable *logIncome* eingeführt: Data → Manage variables in active data set → Compute new variable... → New variable name: `logIncome` → Expression to compute: `log(income)` → OK



Wiederholt man jetzt die *Scatterplot-Matrix*, so kann man erkennen, dass der Zusammenhang zwischen *logIncome* und *education* bzw. *prestige* deutlich *linearer* zu sehen ist.

Jetzt soll dieser lineare Zusammenhang[4] in einem Modell erklärt werden: STATISTICS→ FIT MODELS → LINEAR REGRESSION... → Response variable `logIncome` → Explanatory variables `education` → (OK)

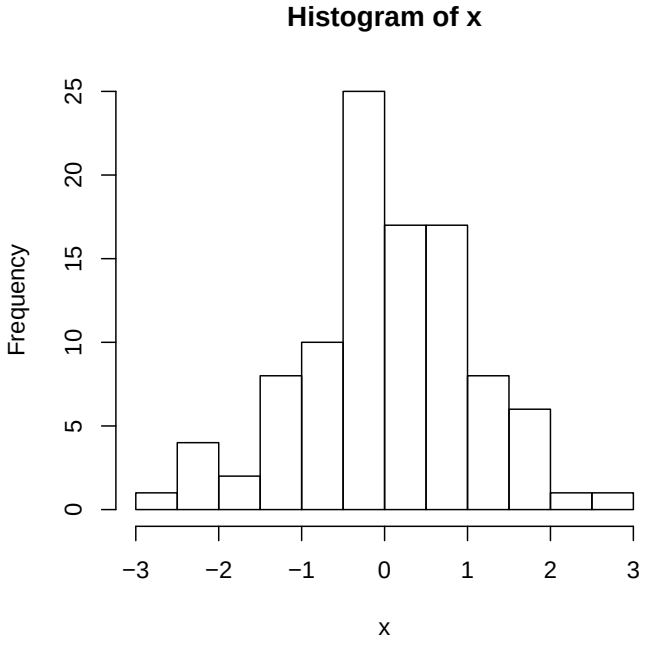
Im Ausgabefenster erscheint nun die wesentlichen Informationen zu dem Modell.

[4]Siehe auch 3.12

3 R-Befehle zur Vorlesung (107.238)

3.1 Histogramm

```
> x <- rnorm(100)
> hist(x)
```



Das Argument `x` ist ein Vektor mit den jeweiligen Werten. Mit dem zusätzlichen Argument `freq = FALSE` wird auf der y-Achse die relative (statt der absoluten) Häufigkeit angegeben, was z.B. dann sinnvoll ist, wenn man noch eine Dichteschätzung einzeichnen will (siehe Kapitel 1.10).

Die Anzahl bzw. Breite der Balken wird aufgrund der Anzahl der Datenpunkte und deren Wertebereich automatisch berechnet. Falls man diese aber selbst festlegen möchte, kann dies mit dem Argument `breaks = a` geschehen, wobei `a` ein Vektor mit den gewünschten Intervallgrenzen ist. Viele weitere Optionen sind auf der Hilfeseite (aufrufen der Hilfe siehe Kapitel 1.4) beschrieben.

3.2 Ortsparameter

`mean(x)`

berechnet den *Mittelwert* $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

`median(x)`

berechnet den *Median* eines numerischen Vektors.

`quantile(x, probs = 0.25)`

berechnet das *Quantil* Q_{probs} für die gegebene Wahrscheinlichkeit `probs`. Fehlt das Argument `probs`, werden die Quantile für die Wahrscheinlichkeiten 0, 0.25, 0.5, 0.75 und 1 ausgegeben.

```
> quantile(x)
```

```
          0%          25%          50%          75%         100%
-2.888920672 -0.455148476 -0.001606084  0.698490772  2.649166881
```

3.3 Streuungsmaße

`var(x)`

Varianz $s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$

`sd(x)`

Standardabweichung (Standard Deviation) $s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$

`IQR(x)`

Interquartiler Abstand (Interquartile Range) $Q_{0.75} - Q_{0.25}$

`mad(x)`

Median der absoluten Abweichung vom Median (Median Absolute Deviation, *Medmed*).

3.4 Höhere Momente

`skewness(x)`[5]

berechnet die *Schiefe* $\frac{1}{n}\sum(x_i - \bar{x})^3/s^3$

`kurtosis(x)`

berechnet die *Kurtosis* $\frac{1}{n}\sum(x_i - \bar{x})^4/s^4 - 3$

[5]Für die Befehle `skewness` und `kurtosis` muss das *package e1071* installiert und geladen sein!

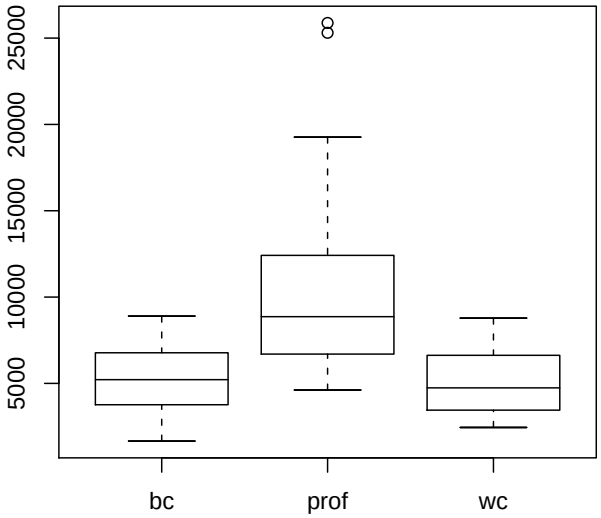
3.5 Boxplots

`boxplot(x)`

Als Argument können ein oder mehrere Vektoren übergeben werden. Mit Hilfe der *Formelnotation* (siehe Kapitel 1.12) kann auch eine Gruppierungsvariable angegeben werden: Z.B. `y ~ grp`. Dabei ist y ein numerischer Vektor mit den darzustellenden Daten und grp ist die *Gruppierungsvariable*.

Falls das Argument vom Typ *data.frame* oder *list* ist, werden die einzelnen Komponenten nebeneinander dargestellt.

```
> library(car)
> data(Prestige)
> attach(Prestige)
> boxplot(income ~ type)
```

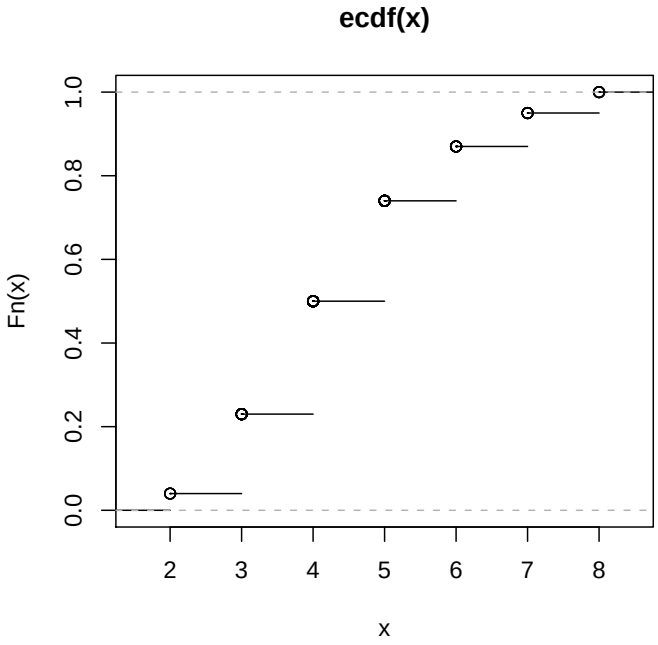


3.6 Empirische Verteilungsfunktion

`plot(ecdf(x))`

Empirische Verteilungsfunktion (Empirical Cumulative Distribution Function).

```
> x <- rbinom(100, 10, 0.5)
> plot(ecdf(x))
```



3.7 Verteilungen

Für die Berechnung von Werten der Dichte- bzw. Verteilungsfunktion, Pseudo-Zufallszahlen und Quantilen von verschiedenen Verteilungen sind in R bereits Funktionen implementiert. Diese beginnen jeweils mit den folgenden Buchstaben:

- `d` (*density*) für Dichtefunktion
- `p` (*probability*) für Verteilungsfunktion
- `q` (*quantiles*) für Berechnung von Quantilen
- `r` (*random*) für das Erzeugen von Zufallszahlen[6]

Diesem Buchstaben folgt die Bezeichnung der Verteilung gemäß nachfolgenden Tabellen, und dann in runder Klammer die Parameter. Diese können je nach Verteilung verschieden sein – siehe *Hilfe*.

Diskrete Verteilungen

Befehl	Name
`binom`	Binomialverteilung
`nbinom`	Negative Binomialverteilung
`hyper`	Hypergeometrische Verteilung
`pois`	Poissonverteilung
`geom`	Geometrische Verteilung
`multinom`	Multinomialverteilung

[6] Mit dem Befehl `set.seed()` wird der *Pseudozufallszahlengenerator* initialisiert. Dies ist aber nicht erforderlich.

Stetige Verteilungen

Befehl	Name
`norm`	Normalverteilung
`lnorm`	Log-Normalverteilung
`exp`	Exponentialverteilung
`t`	T-Verteilung
`f`	F-Verteilung
`chisq`	χ^2-Verteilung
`gamma`	Gammaverteilung
`weibull`	Weibullverteilung

Beispiele:

```
> pnorm(c(-2, -1, 0, 1, 2))

[1] 0.02275013 0.15865525 0.50000000 0.84134475 0.97724987

> rnorm(10)

 [1]  0.45018710 -0.01855983 -0.31806837 -0.92936215 -1.48746031 -1.07519230
 [7]  1.00002880 -0.62126669 -1.38442685  1.86929062
```

Zum Zeichnen der Dichte- oder Verteilungsfunktion einer bestimmten Verteilung siehe auch `curve(funktion, a, b)` im Kapitel 1.10.

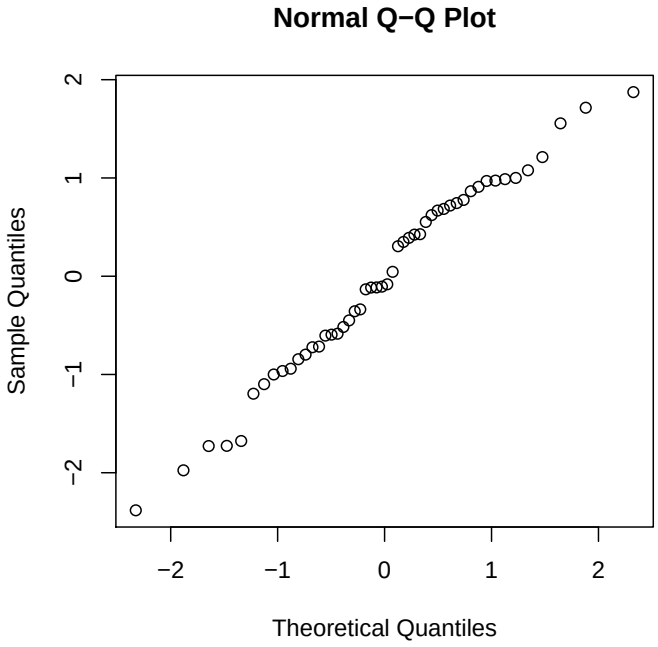
Die (geschätzte) Dichte einer bestimmten Stichprobe `x` kann man mit `plot(density(x))` zeichnen.

3.8 Quantile-Quantile Plots

`qqnorm(x)`

zeichnet einen Quantile-Quantile-Plot zur Prüfung auf Normalverteilung. Dabei werden die empirischen Quantile der Stichprobe gegen die theoretischen Quantile der Normalverteilung geplottet. (Siehe auch `qqplot()`.)

```
> x <- rnorm(50)
> qqnorm(x)
```



3.9 Abhängigkeitsmaße

`cov(x, y)`

Kovarianz: $\text{cov}(X,Y) = s_{XY} = \frac{1}{n-1}\sum(x_i - \bar{x})(y_i - \bar{y})$

`cor(x, y)`

Korrelation: $\text{cor}(X,Y) = r_{XY} = \frac{s_{XY}}{s_X s_Y}$

3.10 Tests

Befehl[7]	Name
`t.test(x)`	Einstichproben-t-Test
`t.test(x, y)`	2-Stichproben-t-Test
`var.test(x, y)`	F-Test
`binom.test(x, n)`	Binomialtest
`wilcox.test(x)`	Wilcoxon Vorzeichen-Rangtest
`wilcox.test(x, y)`	2-Stichproben-Wilcoxon-Test
`chisq.test(x)`	χ^2-Test
`ks.test(x, y)`	Kolmogorov-Smirnov-Test

[7] x und y stehen für numerische Vektoren.

3.11 Varianzanalyse

`aov()`

Varianzanalyse für ein durch eine *Formel* (siehe Kapitel 1.12) bezeichnetes Modell.

`anova()`

Varianzanalyse für ein oder mehrere Objekte vom Typ *lm* (siehe Kapitel 3.12).

3.12 Regression

`lm(y ~ x)`

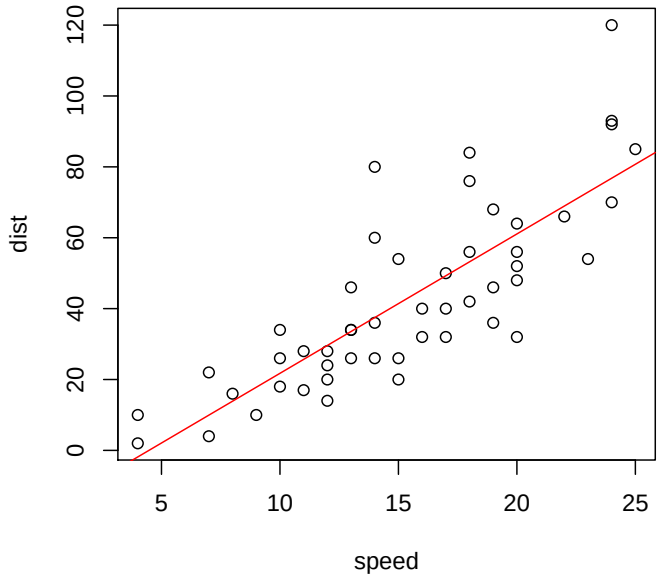
`lm` (Linear Model) berechnet nach der Methode der *kleinsten Quadrate* ein *lineares Regressionsmodell* der Form

$$\mu_{y.x} = a + bx$$

Das Ergebnis ist ein Objekt der Klasse *lm* und enthält (unter anderem) die Koeffizienten a und b. Auf diese kann durch `objektname$coef[1]` bzw. `objektname$coef[2]` direkt zugegriffen werden. Zum Zeichnen der Regressionsgerade ist das aber nicht notwendig, da der Befehl *abline*, wenn ein Objekt der Klasse *lm* übergeben wird, automatisch eine Gerade anhand der Koeffizienten zeichnet (siehe nachfolgendes Beispiel).

Die Argumente werden in Form der *Formelnotation* (siehe auch Kapitel 1.12) übergeben: $y \sim x$ bedeutet *y wird durch x erklärt*.

```
> data(cars)
> attach(cars)
> plot(speed, dist)
> a <- lm(dist ~ speed)
> abline(a, col = "red")
```



3.13 Zweifache Klassifizierung (Kontingenztafel)

`table(a, b)`

Sind `Augenfarbe` und `Haarfarbe` jeweils von der Klasse *factor*, dann kann mit `table` eine entsprechende Kontingenztafel erzeugt werden:

```
> table(Augenfarbe, Haarfarbe)

          Haarfarbe
Augenfarbe dunkel hell
    anders  9      6
    blau   12     32
    braun  22     14
```

Der entsprechenden χ^2-Test wird mit `chisq.test(table(Augenfarbe, Haarfarbe))` durchgeführt.

4 R-Befehle zur Übung (107.248 und 107.249)

Die bei den „Rechen“-Übungen verwendete Beispielsammlung enthält Datensätze, die auch in elektronischer Form zur Verfügung stehen. Bei jedem Beispiel in dieser Sammlung gibt es eine Kodierung in einem Kästchen rechts oben (z.B. *a0148* bei Beispiel 1). Falls bei dieser Kodierung ein *R* dabeisteht, sind diese Daten elektronisch verfügbar (z.B. *a0001 R* bei Beispiel 13). Bei den entsprechenden Dateien handelt es sich um *csv*-Dateien, die insbesondere in R einfach eingelesen werden können (aber auch z.B. in *Excel*), siehe Kapitel 1.3.

Laden Sie hierfür das Paket *bspsam107*, das auf der Übungs-Homepage verfügbar ist. Aufgerufen wird das Paket wie üblich mit `library(bspsam107)`. In der Hilfe sieht man alle zur Verfügung stehenden Datensätze, sprich die oben erwähnten Kodierungen. Möchte man also z.B. die Daten *a0001 R* zu Beispiel 13 laden, dann lautet der Befehl einfach `data(a0001)`.

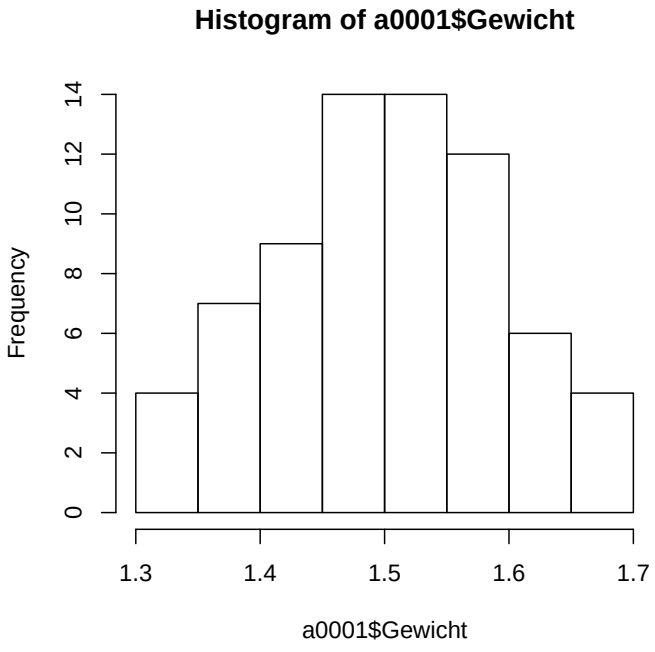
Z.B. kann dann mit

```
> library(bspsam107)
> data(a0001)
> mean(a0001)

Gewicht
  1.507
```

sofort das arithmetische Mittel berechnet werden oder mit

```
> hist(a0001$Gewicht)
```



das Histogramm dargestellt werden.

Literatur

[1] W. N. Venables, D. M. Smith and the R Development Core Team: *An Introduction to R*. http://www.r-project.org/

[2] Leisch, F.: *Sweave User Manual*. Institut für Statistik und Wahrscheinlichkeitstheorie, Technische Universität Wien, Vienna, Austria. URL http://www.ci.tuwien.ac.at/~leisch/Sweave/.