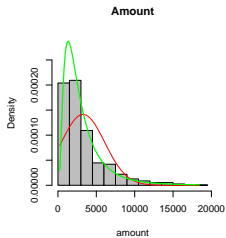
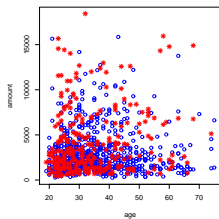


Einführung in R

Marlene Müller

Folienversion: 13. April 2012



Inhaltsverzeichnis

Was ist eigentlich R?

Wie bekomme ich Hilfe?

Etwas Rechnerei zum Anfang

Daten & Dateien

Schöne bunte Welt der Grafik

Etwas Statistik

“Höhere” Mathematik

Einstieg ins Programmieren

Literatur

Was ist eigentlich R?

Was ist eigentlich R?

Wie fange ich an?

Arbeiten unter Unix/Linux

Arbeiten unter Windows

Wie bekomme ich Hilfe?

Etwas Rechnerei zum Anfang

Daten & Dateien

Schöne bunte Welt der Grafik

Etwas Statistik


"Höhere" Mathematik

Einstieg ins Programmieren

Literatur

Was ist eigentlich R?

Programmiersprache S = in den Bell Labs für Statistik, Simulation, Grafik entwickelt (Becker and Chambers, 1984)

- S-PLUS: kommerzielle Implementation
- R: Implementation unter GPL (GNU General Public License), offener Quellcode
 - + interpretierter Programmcode, objektorientiert
 - + leicht erweiterbar durch eigene Routinen, Pakete, DLLs
 - + viele Grafikmöglichkeiten (meist statisch)
 - + standardisiertes, einfach handhabbares Datenformat ( `data.frame`)
 - + gut durchdachtes Format zur Anpassung von (Regressions-)Modellen
 - + aktive Entwicklergruppen, hilfreiche Mailingliste
 - + zunehmend erscheinen Bücher
 - bisher kein „Standard“-GUI
 - verfügbare Routinen/Pakete manchmal unübersichtlich

Wie fange ich an?

Was ist eigentlich R?

Wie fange ich an?

Arbeiten unter Unix/Linux

Arbeiten unter Windows

Wie bekomme ich Hilfe?

Etwas Rechnerei zum Anfang

Daten & Dateien

Schöne bunte Welt der Grafik

Etwas Statistik

"Höhere" Mathematik

Einstieg ins Programmieren

Literatur

Wie fange ich an?

R ist kommandozeilenorientiert, also am einfachsten durch Eingeben von Ausdrücken wie z.B.:

```
> 1+1  
[1] 2
```

```
> 1+2*3^4  
[1] 163
```

```
> x <- 1; y <- 2  
> x+y  
[1] 3
```

Im folgenden ist R-Code so angegeben, dass man ihn direkt kopieren kann:

```
x <- seq(-pi,pi,by=0.1)  
plot(x,sin(x),type="l",col="red",main="Sinuskurve")
```

Arbeiten unter Unix/Linux

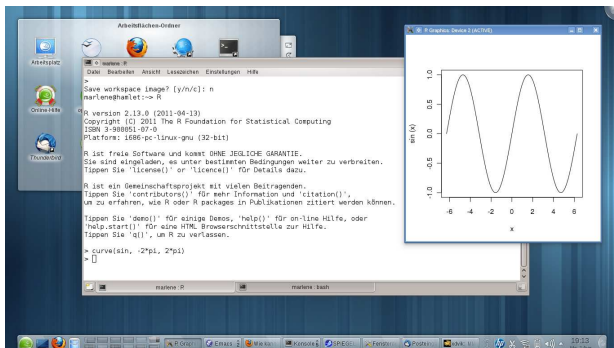


Abbildung: R in einer Unix/Linux-Konsole aufrufen

Arbeiten unter Windows

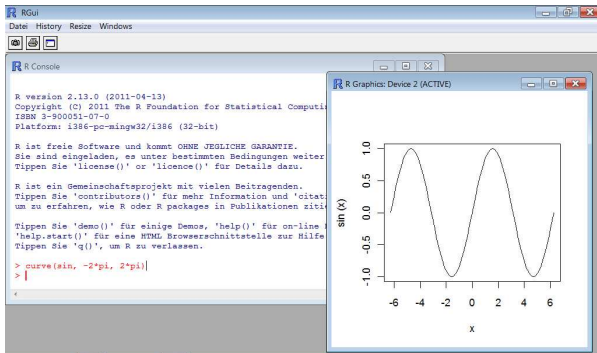


Abbildung: R im Windows-Desktop starten

Wie bekomme ich Hilfe?

Was ist eigentlich R?

Wie bekomme ich Hilfe?

WWW

Mailing-Listen

Bücherauswahl

Etwas Rechnerei zum Anfang

Daten & Dateien

Schöne bunte Welt der Grafik

Etwas Statistik

"Höhere" Mathematik

Einstieg ins Programmieren

Literatur

Wie bekomme ich Hilfe?

Lokale Hilfeseiten:

- ▶ Hilfe zu einer Funktion:
`help(<Funktion>)` oder `?<Funktion>`
- ▶ Hilfe zu einem Package:
`library(help=<Package>)`

Üblicherweise entsprechen die Hilfetexte im lokalen Hilfesystem denen in der Dokumentation zu den Packages.

<http://www.r-project.org>

R-Webseite, dort findet man insbesondere FAQs und eine Google-Site-Search, aber auch:

- ▶ Manuals (<http://cran.r-project.org/manuals.html>)
Einführung, Sprachdefinition, “Writing R Extensions” (DLLs, Packages),
Einführungen in weiteren Sprachen
- ▶ CRAN (<http://cran.r-project.org>)
Comprehensive R Archive Network (→ R Software zum Download)
- ▶ Mailing-Listen (<http://www.r-project.org/mail.html>)
- ▶ Bücher-Liste (<http://www.r-project.org/doc/bib/R-books.html>)
- ▶ Konferenzankündigungen, Verwandte Projekte, ...

Mailing-Listen

- ▶ R-help
wichtigste Liste bei User-Fragen, beim Fragen aber auf alle Fälle <http://www.r-project.org/posting-guide.html> beachten!
→ auch als (Usenet-)NewsGroup [gmane.comp.lang.r.general](http://news.gmane.org) auf <http://news.gmane.org> verfügbar
- ▶ R-announce, R-packages, R-devel
Ankündigungs-, Pakete-, Entwicklerlisten (→ eher für Spezialisten)
- ▶ R-sig-* (special interests groups)
u.a. R-sig-finance = Special Interest Group for 'R in Finance'

Zum Eintragen und für Archive siehe <http://www.r-project.org/mail.html> bzw. <http://news.gmane.org/index.php?prefix=gmane.comp.lang.r>.

Zur Suche hilfreich ist <http://www.rseek.org>.

Bücherauswahl

Datenanalyse:

- ▶ Wollschläger (2010): Grundlagen der Datenanalyse mit R
- ▶ Dalgaard (2002): Introductory Statistics with R
- ▶ Murrell (2005): R Graphics
- ▶ Venables and Ripley (2002): Modern Applied Statistics with S (Ergänzungen u.a. für R: <http://www.stats.ox.ac.uk/pub/MASS4>)

Programmierung:

- ▶ Ligges (2009): Programmieren mit R (siehe auch: <http://www.statistik.uni-dortmund.de/~ligges/PmitR/>)
- ▶ Gentleman (2008): R Programming for Bioinformatics
- ▶ Venables and Ripley (2000): S Programming (siehe auch: <http://www.stats.ox.ac.uk/pub/MASS3/Sprog>)

Weitere:

→ <http://www.r-project.org/doc/bib/R-books.html>

Etwas Rechnerei zum Anfang

Was ist eigentlich R?

Wie bekomme ich Hilfe?

Etwas Rechnerei zum Anfang

- Datentypen

- Vektoren, Matrizen, Arrays, ...

- Operationen (elementweise bzw. vektor-/matrixweise)

Daten & Dateien

Schöne bunte Welt der Grafik

Etwas Statistik

"Höhere" Mathematik

Einstieg ins Programmieren

Literatur

Etwas Rechnerei zum Anfang

```
demo()
demo(graphics)    ## nette Grafiken ;- )
demo(persp)       ## nette 3D-Grafiken ;- )
demo(image)       ## besonders nette Grafiken ;- )

x <- 1
x <- 0 -> y
x <- y <- z <- NA      ## Missing
x <- 0/0              ## Not a Number (NaN)
x <- NULL             ## kein Wert

x <- rnorm(100)       ## Vektor von 100 N(0,1)-Zufallszahlen

hist(x, col="orange")          ## Histogramm
r <- hist(x, col="orange", freq=FALSE) ## dasselbe Histogramm?
g <- seq(-5,5,length=100)
ylim <- range(c(r$density,max(dnorm(g))))

hist(x, col="orange", freq=FALSE, ylim=ylim) ## dasselbe Histogramm?
lines(g, dnorm(g))                ## N(0,1)-Dichte dazu
```

Nützliche Tools

```
ls()                ## alle R-Objekte auflisten

x <- 1:3
x                  ## Objekt (hier Vektor: x) anzeigen

print(x)           ## Objekt (hier Vektor: x) anzeigen, auch
                  ## innerhalb von Funktionen

fun <- function(x){ sin(x) }
fun                ## Objekt (hier Funktion: fun) anzeigen

median             ## Objekt (hier Funktion: median) anzeigen

rm(x)              ## Objekt x löschen

save.image()       ## Workspace speichern (.RData, .Rhistory)
load(".RData")     ## Workspace laden      (.RData, .Rhistory)

date()             ## Datum und Uhrzeit

q()                ## Quit
```


Datentypen

numeric:

```
x <- 1  
y <- pi      ## predefined pi = 3.1415926535898
```

character:

```
x <- "a"  
y <- "Ein Text"
```

logical:

```
x <- TRUE  
y <- 1 > 2
```

```
> y  
[1] FALSE
```

Kompliziertere Datentypen sind durch Kombination dieser drei einfachen Datentypen in Form von Vektoren, Matrizen, Arrays und Listen konstruierbar.

Vektoren

```
x <- c(1,2,3)
```

```
x <- 1:3
```

```
y <- c(1,1,1)
```

```
y <- rep(2,10)
```

```
z <- as.character(1:3)
```

```
z <- c("a", "b", "c")
```

```
length(z)
```

```
names(x) <- z
```

```
x[2:3]
```

```
x["b"]
```

Alle Elemente des Vektors sind vom gleichen Typ (numeric, character, logical)!

Matrizen

```
x <- 1:20  
x <- matrix(x, 5,4)      ## matrix(x, nrow=5,ncol=4)
```

```
x[2,3]  
x[c(1,5),2:4]  
x[,2:4]
```

```
dim(x)  
nrow(x); ncol(x)
```

```
length(x)  
as.vector(x)
```

```
dimnames(x) <- list(paste("row",1:nrow(x), sep=""),  
                    c("a","b","c","d"))
```

```
x[,"b"]  
x[,c("a","b")]
```

Alle Elemente der Matrix sind vom gleichen Typ (numeric, character, logical)!

Erzeugen von Vektoren und Matrizen

Vektoren aus Vektoren konstruieren:

```
x <- c(2,6,3)
y <- 1:3
```

```
c(x,y)          ## zwei Vektoren aneinander hängen
c(x,1:5,y,6)    ## Vektoren und Skalare aneinander hängen
```

Matrizen aus Vektoren konstruieren:

```
x <- c(2,6,3)
y <- 1:3
```

```
cbind(x,y)      ## vertikal zusammensetzen
rbind(x,y)      ## horizontal zusammensetzen
```

```
cbind(x,y,rep(0,3)) ## vertikal zusammensetzen
```

Arrays

```
x <- 1:60  
x <- array(x, c(5,4,3))
```

```
x[2,3,1]  
x[1,2:4,3]  
x[, ,1]
```

```
dim(x)  
nrow(x)  
ncol(x)
```

```
length(x)  
as.vector(x)
```

```
dimnames(x) <- list(paste("row",1:nrow(x), sep=""),  
                    c("a","b","c","d"),c("x","y","z"))
```

Alle Elemente des Arrays sind vom gleichen Typ (numeric, character, logical)!

Listen

```
x <- list(Eins=11:15, Zwei=c("a","b","c"), Drei=(1:4)>0)
y <- list(x=x, Vier=1:3)
```

```
x$Eins
y$x$Eins
```

```
y$Vier
y[[2]]
```

```
length(x)
length(y)
```

```
y$Fuenf <- names(x)
```

Listen enthalten Objekte verschiedenen Typs, die Objekte können mit `$<Name>` über ihren Namen oder mit `[[<Nummer>]]` über ihre Nummer angesprochen werden.

Datenmatrizen (data.frame)

```
x <- data.frame(N=11:14, C=c("a", "b", "c", "d"), L=(1:4)>0)
```

```
dim(x)
```

```
nrow(x); ncol(x)
```

```
length(x)
```

```
as.vector(x)
```

```
names(x)
```

```
x[2,3]
```

```
x[,2:3]
```

```
x[,2]
```

```
x["C"]
```

```
x$C
```

Datenmatrizen kann man als Listen sehen, bei denen alle Spalten die gleiche Länge haben. → Excel-Tabellen, die als .csv abgespeichert wurden, werden typischerweise als Datenmatrix (data.frame) in R eingelesen.

Operationen (elementweise bzw. vektor-/matrixweise)

```
x <- matrix( 1:20, 5, 4)  ## 5x4 Matrix

x+1; x-1; x*1; x/1      ## elementweise Operationen
sin(x); exp(x)          ## elementweise Funktionsaufrufe

y <- 1:5
x * y                   ## elementweise Multiplikation

z <- 1:4
x %*% z                 ## Matrixmultiplikation

min(x)                  ## Minimum aller Elemente von x
apply(x,1,min)          ## Zeilenminima
apply(x,2,min)          ## Spaltenminima

y <- c(TRUE, TRUE, FALSE, FALSE)
y & TRUE                ## elementweise logisches "UND"
y | FALSE               ## elementweise logisches "ODER"
!y                      ## elementweise logisches "NICHT"

y && TRUE                ## hier gilt nur das erste
y || FALSE              ## Ergebnis ("UND" bzw. "ODER")
```


Daten & Dateien

Was ist eigentlich R?

Wie bekomme ich Hilfe?

Etwas Rechnerei zum Anfang

Daten & Dateien
CSV-Dateien lesen und Speichern
R-Skriptdateien
Zufallszahlen und Verteilungen

Schöne bunte Welt der Grafik

Etwas Statistik

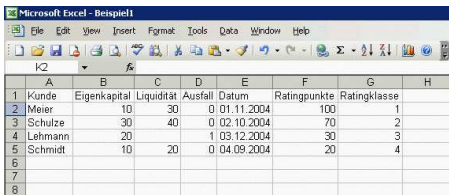
“Höhere” Mathematik

Einstieg ins Programmieren

Literatur

Daten & Dateien

Beispiel-Datei in Excel:



| | A | B | C | D | E | F | G | H |
|---|---------|--------------|------------|---------|------------|--------------|--------------|---|
| 1 | Kunde | Eigenkapital | Liquidität | Ausfall | Datum | Ratingpunkte | Ratingklasse | |
| 2 | Meier | 10 | 30 | 0 | 01.11.2004 | 100 | 1 | |
| 3 | Schulze | 30 | 40 | 0 | 02.10.2004 | 70 | 2 | |
| 4 | Lehmann | 20 | | 1 | 03.12.2004 | 30 | 3 | |
| 5 | Schmidt | 10 | 20 | 0 | 04.09.2004 | 20 | 4 | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | | | | | | | |

→ unter Excel als CSV speichern: Beispiel1.csv

```
Kunde;Eigenkapital;Liquidität;Ausfall;Datum;Ratingpunkte;Ratingklasse  
Meier;10;30;0;01.11.2004;100;1  
Schulze;30;40;0;02.10.2004;70;2  
Lehmann;20;;1;03.12.2004;30;3  
Schmidt;10;20;0;04.09.2004;20;4
```

CSV-Dateien lesen und Speichern

Lesen der Datei Beispiel1.csv:

```
x <- read.csv("Beispiel1.csv", sep=";")
```

```
dim(x)
```

```
names(x)
```

```
x
```

Ergebnis in R:

| | Kunde | Eigenkapital | Liquidität | Ausfall | Datum | Ratingpunkte | Ratingklasse |
|---|---------|--------------|------------|---------|------------|--------------|--------------|
| 1 | Meier | 10 | 30 | 0 | 01.11.2004 | 100 | 1 |
| 2 | Schulze | 30 | 40 | 0 | 02.10.2004 | 70 | 2 |
| 3 | Lehmann | 20 | NA | 1 | 03.12.2004 | 30 | 3 |
| 4 | Schmidt | 10 | 20 | 0 | 04.09.2004 | 20 | 4 |

Schreiben der Daten in Beispiel2.csv:

```
write.table(x, file="Beispiel2.csv", sep=";", row.names=FALSE, quote=FALSE)
```

Weitere Funktionen für den Dateninput bzw. -output

Andere Funktionen zum Lesen:

- ▶ `read.table` (Daten im ASCII-Format)
- ▶ `scan` (scannt beliebige Textdatei, eigene Nachbearbeitung erforderlich)

Funktionen zum evtl. Konvertieren von Spalten:

- ▶ `as.numeric`, `as.character`, `as.factor`, `as.Date`

Andere Möglichkeiten der Kommunikation mit Excel:

- ▶ `RODBC` (Zugriff auf Excel oder Access als Datenbank)
- ▶ R-Excel-Interface über DCOM-Server
(<http://cran.at.r-project.org/contrib/extra/dcom>)

R-Skriptdateien

Skript mit R-Befehlen einlesen:

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

R-Output in Datei abspeichern:

```
sink("MeinOutput.txt")    ## ab jetzt Ausgabe in Datei  
sink()                    ## und nun wieder auf Bildschirm
```

Zufallszahlen und Verteilungen

Beispiel Normalverteilung:


| | |
|-------------------------------------|---------------------------|
| <code>rnorm(n, mean=0, sd=1)</code> | Zufallszahlen |
| <code>dnorm(x, mean=0, sd=1)</code> | Dichte (pdf) |
| <code>pnorm(x, mean=0, sd=1)</code> | Verteilungsfunktion (cdf) |
| <code>qnorm(p, mean=0, sd=1)</code> | Quantile |

nach gleichem Muster:


Gleichverteilung

 `{r|d|p|q}unif`


t-Verteilung

 `{r|d|p|q}t`


Lognormalverteilung

 `{r|d|p|q}lnorm`


Gamma-Verteilung

 `{r|d|p|q}gamma`


χ^2 -verteilung

 `{r|d|p|q}chisq`


Beta-Verteilung

 `{r|d|p|q}beta`


Binomialverteilung

 `{r|d|p|q}binom`


Poisson-Verteilung

 `{r|d|p|q}pois`

Exponentialverteilung

 `{r|d|p|q}exp`

F-Verteilung

 `{r|d|p|q}f`

...

→ bei Bedarf den vorher Seed mit  `set.seed` setzen

Beispiel: Normal- vs. t-Verteilung

```
x <- rnorm(100)
mean(x)
sd(x)
```

```
plot(rnorm(10000), rnorm(10000))
```

```
x <- seq(-5,5,by=0.1)
plot(x, dnorm(x), type="l", col="black", lwd=2)
lines(x, dt(x, df=1), col="blue")
lines(x, dt(x, df=5), col="orange")
lines(x, dt(x, df=20), col="red")
```

```
qnorm(0.95)
qnorm(0.975)
```

Multivariate Normalverteilung

```
library(help=mvtnorm)
library(mvtnorm)

mu <- c(0,0)      ## Erwartungswerte
sigma <- c(1,1)   ## Standardabweichungen
rho <- 0.5        ## Korrelation

S <- matrix(NA, 2,2)
diag(S) <- sigma^2
S[1,2] <- S[2,1] <- rho*prod(sigma)

x <- rmvnorm(n=10000, mean=mu, sigma=S)
plot(x)

x <- seq(-5*sigma[1]+mu[1], 5*sigma[1]+mu[1], length = 50)
y <- seq(-5*sigma[2]+mu[2], 5*sigma[2]+mu[2], length = 50)
f <- function(x,y) { dmvnorm(cbind(x,y), mean=mu, sigma=S) }
z <- outer(x, y, f)
persp(x, y, z, theta = 10, phi = 20, expand = 0.5,
      col = "lightblue", shade = 0.75)
```


Schöne bunte Welt der Grafik

Was ist eigentlich R?

Wie bekomme ich Hilfe?

Etwas Rechnerei zum Anfang

Daten & Dateien

Schöne bunte Welt der Grafik

- Balkendiagramme

- Boxplots

- Histogramme

- Scatterplots und Kurven

- Dreidimensionale Darstellung

- Gestaltungsmöglichkeiten

- Grafiken speichern

Etwas Statistik

“Höhere” Mathematik

Einstieg ins Programmieren

Literatur

Schöne bunte Welt der Grafik

Kreditausfalldaten:

<http://www.stat.uni-muenchen.de/service/datenarchiv/kredit/kredit.html>

```
file <- read.csv("D:\\\\kredit.asc", sep=" ")
y <- 1-file$credit          ## default set to 1

prev <- (file$moral >2)+0      ## previous loans were OK
employ <- (file$beszeit >1)+0  ## employed (>=1 year)
dura <- (file$laufzeit)       ## duration
d9.12 <- ((file$laufzeit >9)&(file$laufzeit <=12))+0 ## 9-12 months
d12.18 <- ((file$laufzeit >12)&(file$laufzeit <=18))+0 ## 12-18 months
d18.24 <- ((file$laufzeit >18)&(file$laufzeit <=24))+0 ## 18-24 months
d24 <- (file$laufzeit >24)+0   ## > 24 months
amount <- file$hoehe          ## amount of loan
age <- file$alter             ## age of applicant
savings <- (file$sparkont > 4)+0 ## savings >= 1000 DM
phone <- (file$telefon==1)+0  ## applicant has telephone
foreign <- (file$gastarb==1)+0 ## non-german citizen
purpose <- ((file$verw==1)|(file$verw==2))+0 ## loan is for a car
house <- (file$verm==4)+0     ## house owner
```

Balkendiagramme

→ grafische Darstellung der Häufigkeitsverteilung diskreter Merkmale

```
table(dura)                ## Häufigkeitstabelle

barplot(table(dura), col="cyan", main="Duration of Loan")
                        ## absolute Häufigkeiten

n <- length(dura)
barplot(table(dura)/n, col="cyan", main="Duration of Loan")
                        ## relative Häufigkeiten

par(mfrow=c(1,3))  ## 1 Zeile, 3 Spalten im Display
barplot(table(dura), col="cyan", main="Duration of Loan")
barplot(table(savings), col="orange", main="Savings >1000 DM")
barplot(table(house), col="magenta", main="House Owner")
par(mfrow=c(1,1))  ## Display-Teilung zurücksetzen!
```

Beispiel: Balkendiagramme

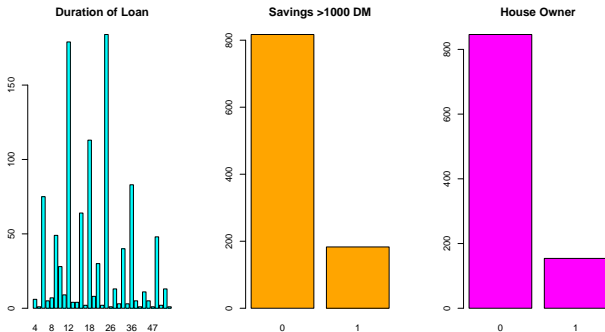


Abbildung: Beispiele für Balkendiagramme: Laufzeit des Kredits (links), Spareinlagen (mitte) und Immobilienbesitz (rechts)

Boxplots

→ grafische Darstellung von Ausreißern, Minimal-/Maximalwerten, 25%-, 50%- und 75%-Quantilen

```
boxplot(age)
```

```
boxplot(age, horizontal=TRUE)
```

```
boxplot(age, col="gray",horizontal=TRUE)
```

```
boxplot(age ~ y, col=c("gray","red"),  
        horizontal=TRUE, main="Age vs. Y")
```

```
boxplot(amount ~ y, col=c("gray","red"),  
        horizontal=TRUE, main="Amount vs. Y")
```

Beispiele: Boxplots

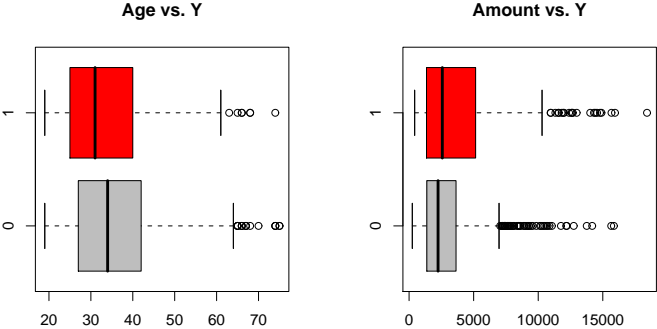


Abbildung: Alter des Kreditnehmers (links) und Kreditbetrag (rechts) gegen Ausfallbeobachtung (1 =Ausfall, 0 = Nichtausfall)

Histogramme

→ grafische Darstellung der Häufigkeitsverteilung (Häufigkeitsdichte) stetiger Merkmale

```
hist(age)
hist(age, freq=FALSE)
hist(age, freq=FALSE, col="gray")

hist(amount, freq=FALSE, col="gray", main="Amount")
xx <- seq(min(amount),max(amount), length=100)
lines(xx, dnorm(xx, mean(amount), sd(amount)), col="red")
lines(xx, dlnorm(xx, mean(log(amount)), sd(log(amount))), col="green", lwd=2)

## mit kleineren Intervallen und besserer vertikaler Skalierung
b <- seq(0,20000,by=1500)           ## neue Intervallgrenzen
h <- hist(amount, freq=FALSE, breaks=b, plot=FALSE)
                                     ## Histogramm ohne Grafik
xx <- seq(min(amount),max(amount), length=100)
d1 <- dnorm(xx, mean(amount), sd(amount))  ## Normal-Dichte
d2 <- dlnorm(xx, mean(log(amount)), sd(log(amount)))
                                     ## LogNormal-Dichte

ylim <- range( c(h$density, d1, d2) )

hist(amount, freq=FALSE, breaks=b, col="gray", main="Amount", ylim=ylim)
lines(xx, d1, col="red")
lines(xx, d2, col="green", lwd=2)
```

Beispiele: Histogramm

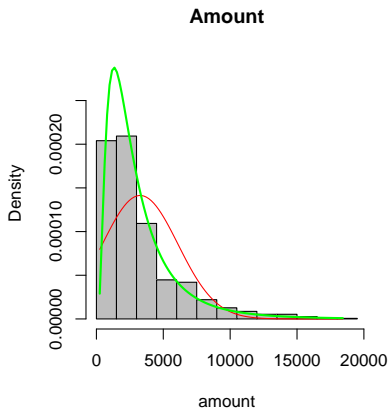


Abbildung: Verteilung des Kreditbetrags, Histogramm im Vergleich mit Normal- und LogNormaldichte

Punktwolken (Scatterplots, Streudiagramme)

```
plot(age, amount)
```

```
color <- 1*(y==1) + 2*(y==0)
```

```
plot(age, amount, col=color)
```

```
color <- rep("", length(age))
```

```
color[y==1] <- "red"
```

```
color[y==0] <- "blue"
```

```
plot(age, amount, col=color)
```

```
plot(1:20,1:20,col=1:20, pch=1:20)
```

```
text(1:20,1:20,labels=as.character(1:20), pos=4)
```

```
symbol <- 8*(y==1) + 1*(y==0)
```

```
plot(age, amount, col=color, pch=symbol)
```

Punktwolken und Kurven

```
x <- seq(-pi,pi,length=100)
plot(x, sin(x), type="l")
lines(x, cos(x), col="red")

logit <- glm(y ~ age, family=binomial(link = "logit"))

plot(age, logit$fitted.values)

plot(age, logit$fitted.values, type="l")
      ## nicht so, ...

o <- order(age)
plot(age[o], logit$fitted.values[o], type="l")
      ## ... sondern so (sortiert!)

plot(age[o], logit$fitted.values[o], type="l", lwd=2, ylim=c(0,1))
title("PDs")
points(age, y, col="red", pch=3, cex=0.5)
```

Beispiele: Punktwolke und Kurven

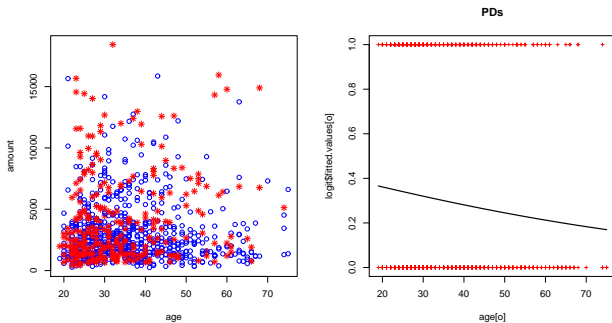


Abbildung: Scatterplot von Alter vs. Kreditbetrag (links),
Logit-Ausfallwahrscheinlichkeiten (rechts)

Oberflächen, Punktwolken und Konturkurven

```
## Bivariate Normalverteilungsdichte
library(mvtnorm)
x <- y <- seq(-5, 5, length = 50)
f <- function(x,y) { dmvnorm(cbind(x,y)) }
z <- outer(x, y, f)
persp(x, y, z, theta=10, phi=20, expand=0.5, col="lightblue")
persp(x, y, z, theta=10, phi=20, expand=0.5, col="lightblue",
      shade = 0.75)
```

```
## Konturkurven der bivariaten Normalverteilungsdichte
x <- y <- seq(-5, 5, length = 150)
z <- outer(x, y, f)
contour(x, y, z, nlevels=20)
contour(x, y, z, nlevels=20, col=rainbow(20))
contour(x, y, z, nlevels=20, col=rainbow(20), labels="")
```

```
## Dreidimensionale normalverteilte Daten
library(scatterplot3d)
x <- matrix(rnorm(15000),ncol=3)
scatterplot3d(x)
scatterplot3d(x, angle=20)
```

Beispiele: Oberflächen, Punktwolken und Konturkurven

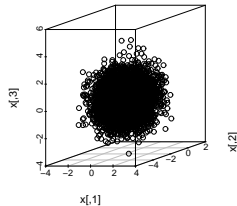
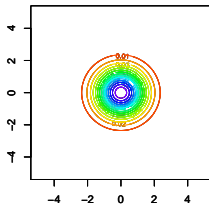
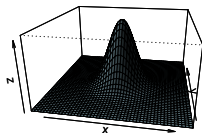


Abbildung: Bivariate Normalverteilungsdichte: 3D-Darstellung (links) und Konturkurven (mitte); 3D-Scatterplot (rechts)

Gestaltungsmöglichkeiten

Direkt in den Grafikroutinen: → Hilfe dazu mit `?par`

- ▶ Farben setzen mit `col=...`
(Generierung von → `?rainbow`, `?rgb`, `?col2rgb`)
- ▶ Symbole setzen mit `pch=...`, Größen mit `cex=...`
- ▶ Titel setzen mit `main=...`, Achsenlabels mit `xlab=...`, `ylab=...`
- ▶ Zeichenbereich setzen mit `xlim=...`, `ylim=...`

Nach Zeichnen einer Grafik:

- ▶ Linien und Punkte mit `lines(...)` bzw. `points(...)` ergänzen
- ▶ Labels (Texte) mit `text(...)` ergänzen
- ▶ Titel mit `title(...)` ergänzen
- ▶ Legende mit `legend(...)` ergänzen

Grafiken speichern

- ▶ PostScript:

```
x <- matrix(rnorm(5000),ncol=2)
plot(x)
postscript("MeinPlot.eps", width=5, height=4, horizontal=FALSE)
plot(x)
dev.off()
```

- ▶ andere Formate sind u.a. `Rpdf`, `Rpictex`, `Rxfig`, `Rpng`, `Rjpeg`
→ siehe `R?Devices`

Etwas Statistik

Was ist eigentlich R?

Wie bekomme ich Hilfe?

Etwas Rechnerei zum Anfang

Daten & Dateien

Schöne bunte Welt der Grafik

Etwas Statistik

- Kennzahlen

- Tabellen

- Lineare Regression

- Generalisiertes Lineares Modell (GLM)

- Weitere Regressions- und Zeitreihenmodelle

- Normaltests

- Vergleich von Verteilungen

- Weitere Tests

"Höhere" Mathematik

Einstieg ins Programmieren

Literatur

Etwas Statistik

Kreditausfalldaten:

<http://www.stat.uni-muenchen.de/service/datenarchiv/kredit/kredit.html>

```
file <- read.csv("D:\\\\kredit.asc", sep=" ")
y <- 1-file$kredit          ## default set to 1

prev <- (file$moral >2)+0      ## previous loans were OK
employ <- (file$beszeit >1)+0  ## employed (>=1 year)
dura <- (file$laufzeit)       ## duration
d9.12 <- ((file$laufzeit >9)&(file$laufzeit <=12))+0 ## 9-12 months
d12.18 <- ((file$laufzeit >12)&(file$laufzeit <=18))+0 ## 12-18 months
d18.24 <- ((file$laufzeit >18)&(file$laufzeit <=24))+0 ## 18-24 months
d24 <- (file$laufzeit >24)+0   ## > 24 months
amount <- file$hoehe          ## amount of loan
age <- file$alter             ## age of applicant
savings <- (file$sparkont > 4)+0 ## savings >= 1000 DM
phone <- (file$telefon==1)+0  ## applicant has telephone
foreign <- (file$gastarb==1)+0 ## non-german citizen
purpose <- ((file$verw==1)|(file$verw==2))+0 ## loan is for a car
house <- (file$verm==4)+0     ## house owner
```

Kennzahlen

```
kredit <- data.frame(y,age,amount,dura,prev,savings,house)
```

```
summary(kredit)
```

```
mean(kredit$age)
```

```
sd(kredit$age)
```

```
var(kredit$age)
```

```
cov(kredit[,1:3])
```

```
cor(kredit[,1:3])
```

```
median(kredit$age)
```

```
quantile(kredit$age,c(0.1,0.5,0.9))
```

```
library(help=e1071)
```

```
library(e1071)
```

```
skewness(kredit$age)
```

```
kurtosis(kredit$age)
```

```
skewness(rnorm(1000))
```

```
kurtosis(rnorm(1000))
```

Tabellen

```
length(kredit$age)
length(unique(kredit$age))
```

```
table(kredit$age)
table(kredit$dura)
table(kredit$savings)
```

```
table(kredit$y, kredit$savings)
table(kredit$y, kredit$savings)/nrow(kredit)
```

```
table(kredit$y, kredit$savings, kredit$house)
```

```
unique(kredit[,c("y", "savings", "house")])
```

Lineare Regression

```
plot(kredit$sage, kredit$dura)
```

```
lm <- lm( dura ~ age, data=kredit)
summary(lm)                ## Abhängigkeit von Alter
abline(lm, col="red", lwd=2)
```

```
lm2 <- lm( dura ~ age + amount, data=kredit)
summary(lm2)              ## Abhängigkeit von Alter+Betrag
```

```
lm3 <- lm( dura ~ amount, data=kredit)
summary(lm3)             ## Abhängigkeit von Betrag
plot(kredit$amount, kredit$dura)
abline(lm3, col="red", lwd=2)
```

```
lm4 <- lm( dura ~ amount + I(amount^2), data=kredit)
summary(lm4)            ## Abhängigkeit von Betrag (quadr.)
o <- order(kredit$amount)
lines(kredit$amount[o], lm4$fitted.values[o], col="blue", lwd=2)
```

Lineare Regression (Forts.)

→ Laufzeit des Kredits hängt klar vom Kreditbetrag ab:

```
> summary(lm4)
```

Call:

```
lm(formula = dura ~ amount + I(amount^2), data = kredit)
```

Residuals:

| Min | 1Q | Median | 3Q | Max |
|----------|---------|---------|--------|---------|
| -34.6115 | -5.5761 | -0.9547 | 5.0850 | 42.1110 |

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) | |
|-------------|------------|------------|---------|----------|-----|
| (Intercept) | 8.410e+00 | 6.516e-01 | 12.906 | < 2e-16 | *** |
| amount | 4.855e-03 | 2.961e-04 | 16.393 | < 2e-16 | *** |
| I(amount^2) | -1.815e-07 | 2.309e-08 | -7.863 | 9.7e-15 | *** |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.144 on 997 degrees of freedom

Multiple R-Squared: 0.4262, Adjusted R-squared: 0.425

F-statistic: 370.3 on 2 and 997 DF, p-value: < 2.2e-16

Lineare Regression (Forts.)

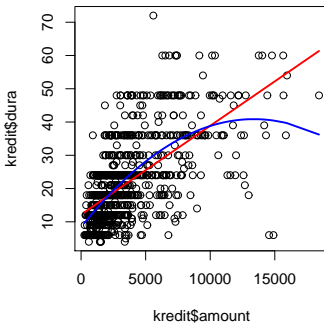
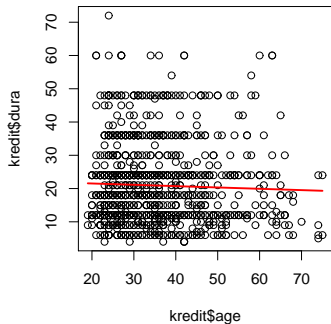


Abbildung: Abhängigkeit der Kreditlaufzeit vom Alter (links) und vom Kreditbetrag (rechts)

Generalisiertes Lineares Modell (GLM)

→ Schätzung der Ausfallwahrscheinlichkeiten mit einem Logit-Modell

```
logit <- glm(y ~ age + amount + dura + prev + savings + house,  
            family=binomial(link = "logit"))  
summary(logit)
```

```
logit2 <- glm(y ~ age + amount + I(amount^2) + dura + prev +  
             savings + house, family=binomial(link = "logit"))  
summary(logit2)
```

Beispiel: Logit-Modell

```
> summary(logit2)
```

Call:

```
glm(formula = y ~ age + amount + I(amount^2) + dura + prev +  
     savings + house, family = binomial(link = "logit"))
```

Deviance Residuals:

| Min | 1Q | Median | 3Q | Max |
|---------|---------|---------|--------|--------|
| -2.1244 | -0.8495 | -0.6196 | 1.0935 | 2.2584 |

Coefficients:

| | Estimate | Std. Error | z value | Pr(> z) |
|-------------|------------|------------|---------|--------------|
| (Intercept) | -4.637e-01 | 3.035e-01 | -1.528 | 0.12652 |
| age | -1.748e-02 | 7.159e-03 | -2.442 | 0.01460 * |
| amount | -2.070e-04 | 9.348e-05 | -2.214 | 0.02679 * |
| I(amount^2) | 1.870e-08 | 6.941e-09 | 2.694 | 0.00707 ** |
| dura | 3.992e-02 | 8.106e-03 | 4.925 | 8.46e-07 *** |
| prev | -7.589e-01 | 1.619e-01 | -4.688 | 2.76e-06 *** |
| savings | -9.897e-01 | 2.232e-01 | -4.435 | 9.22e-06 *** |
| house | 6.277e-01 | 2.073e-01 | 3.027 | 0.00247 ** |

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1221.7 on 999 degrees of freedom
Residual deviance: 1102.1 on 992 degrees of freedom
AIC: 1118.1

Number of Fisher Scoring iterations: 4

Weitere Regressions- und Zeitreihenmodelle

| Modell | Routinen / Packages |
|-------------------------------------|--|
| Linear | <code>Rlm</code> , <code>Ranova</code> |
| GLM | <code>Rglm</code> , <code>Rmgcv</code> / <code>Rgam</code> (additiv nichtparametrisch) |
| Nichtlinear | <code>Rnls</code> |
| Nichtparametrisch | <code>Rlocpoly</code> , <code>Rlocfit</code> |
| Mixed Models | <code>Rlmm</code> , <code>Rnlme</code> , <code>RglmmML</code> , <code>RglmmPQL</code> |
| Zeitreihen | <code>Rar</code> , <code>Rarma</code> , <code>Rarima</code> , <code>Rarima0</code> , <code>Rgarch</code> |
| Classification and regression trees | <code>Rtree</code> , <code>Rrpart</code> , <code>Rparty</code> |

Packages: `RMASS`, `Rstats`, `RKernSmooth`, `Rtseries`

Normaltests

```
library(KernSmooth)
f <- bkde(kredit$age)
plot(f, type="l", xlim=range(f$x), ylim=range(f$y))
                                     ## Verteilung normal?
title("Distribution of Age")

t <- shapiro.test(kredit$age)
t
t$p.value

library(tseries)
t <- jarque.bera.test(kredit$age)
t
t$p.value
```










Vergleich von Verteilungen




```
library(KernSmooth)
f0 <- bkde(kredit$age[y==0])
f1 <- bkde(kredit$age[y==1])
plot(f0, type="l", col="blue", xlim=range(c(f0$x,f1$x)),
      ylim=range(c(f0$y,f1$y)))
lines(f1, col="red")
## Verteilungen gleich?
title("Age vs. Default")

t <- ks.test(kredit$age[y==1],kredit$age[y==0])
t
t$p.value

t <- wilcox.test(kredit$age[y==1],kredit$age[y==0])
t
t$p.value
```

Weitere Tests

| Test | Routinen |
|--|--|
| Mittelwert-Tests/-Vergleiche (t-Tests) |  <code>t.test</code> |
| Varianz-Tests/-Vergleiche (F-Tests) |  <code>var.test</code> |
| Binomial-Tests |  <code>prop.test</code> ,  <code>binom.test</code> |
| Korrelation |  <code>cor.test</code> |
| Rangtests |  <code>wilcox.test</code> |
| Regression |  <code>anova</code> |
| Unit Roots (mean reversion) |  <code>adf.test</code> ,  <code>kpss.test</code> |

Packages:  `stats`,  `tsseries`,  `exactRankTests`

“Höhere” Mathematik

Was ist eigentlich R?

Wie bekomme ich Hilfe?

Etwas Rechnerei zum Anfang

Daten & Dateien

Schöne bunte Welt der Grafik

Etwas Statistik

“Höhere” Mathematik
Optimierung von Funktionen
Interpolation
Numerische Integration

Einstieg ins Programmieren

Literatur

Optimierung von Funktionen

Beispiel: lineares Modell

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i$$

Erzeugen künstlicher Daten:

```
n <- 100; b <- c(-1,3)
x <- matrix(rnorm(n*length(b)),ncol=length(b))
                                     ## Regressoren
e <- rnorm(n)/4
y <- 1 + x %*% b + e                 ## Lineares Modell
l <- lm( y~x ); summary(l)          ## built-in lm
```

→ zu optimieren ist hier:

$$QS = \sum_i (y_i - \mathbf{x}_i^T \beta)^2$$

(zugegebenermaßen kein wirklich gutes Beispiel für iterative Optimierung ;-))

Optimierung von Funktionen (Forts.)

Optimierung (ohne Konstante):

```
QS <- function(b, x, y){ sum( (y - x %*% b)^2 ) }  
                                ## Optimierungskriterium  
  
b0 <- c(0,0)  
opt <- optim(b0, QS, method="BFGS", x=x, y=y)  
                                ## Optimierung  
  
opt  
sum( (x %*% opt$par - mean(y))^2 )/sum( (y-mean(y))^2 ) ## R^2
```

→ Bestimmtheitsmaß R^2 kann hier außerhalb von $[0, 1]$ liegen

Optimierung (mit Konstante):

```
b1 <- c(0,0,0)  
x1 <- cbind(rep(1,n),x)  
opt1 <- optim(b1, QS, method="BFGS", x=x1, y=y)  
                                ## Optimierung  
  
opt1  
sum( (x1 %*% opt1$par - mean(y))^2 )/sum( (y-mean(y))^2 ) ## R^2
```

Optimierung von Funktionen (Forts.)

Optimierung mit Gradient:

$$QS = \sum_i (y_i - x_i^\top \beta)^2 = (y - X\beta)^\top (y - X\beta), \quad \frac{\partial QS}{\partial \beta} = -2X^\top y + 2X^\top X\beta$$

```
D.QS <- function(b, x, y){ -2* t(x) %*% y + 2* t(x) %*% x %*% b }  
## Gradient
```

```
opt2 <- optim(b1, QS, D.QS, method="BFGS", x=x1, y=y)  
opt2  
sum( (x1 %*% opt2$par - mean(y))^2 )/sum( (y-mean(y))^2 ) ## R^2
```

Optimierung mit Intervallrestriktion (z.B. $\beta_j \geq 0$):

```
b2 <- c(0,0,0)  
x1 <- cbind(rep(1,n),x)  
opt3 <- optim(b1, QS, D.QS, method="BFGS", lower=0, x=x1, y=y)  
opt3  
sum( (x1 %*% opt3$par - mean(y))^2 )/sum( (y-mean(y))^2 ) ## R^2
```


Optimierung von Funktionen (Forts.)

Optimierung mit linearer Restriktion (z.B. $\beta_0 \geq 0, \beta_1 + \beta_2 \leq 2$):

$$u\beta - c = \begin{pmatrix} 0 & -1 & -1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} - \begin{pmatrix} -2 \\ 0 \end{pmatrix} \geq 0$$

```
u <- cbind( c(0,1), c(-1,0), c(-1,0) )
c <- c(-2,0)
```

```
applyDefaults <- function(fn, ...) { ## um weitere Parameter
  function(x) fn(x, ...)           ## an QS, D.QS zu übergeben
}
```

```
b4 <- rep(0.5,3)
opt4 <- constrOptim(b4, applyDefaults(QS, x=x1, y=y),
                    applyDefaults(D.QS, x=x1, y=y), ui=u, ci=c)
opt4
sum( (x1 %*% opt4$par - mean(y))^2 )/sum( (y-mean(y))^2 ) ## R^2
```

Interpolation

→ `R`approx für lineare, `R`spline bzw. `R`interpSpline für Spline-Approximation



```
x <- seq(-5,5,by=1)
y <- sin(x)
```

```
xx <- seq(-5,5,by=0.1)
y.approx <- approx(x,y, xout=xx)$y
yy <- sin(xx)
```

```
plot(xx,yy, type="l", col="green")
lines(xx,y.approx, lwd=2)
```

```
library(splines)
sp <- interpSpline(x,y)
lines(predict(sp,xx), col="red")
```

Numerische Integration

-  integrate für 1-dimensionale,
-  adapt für mehrdimensionale Integration

```
pnorm(0)
```

```
it <- integrate(dnorm, -Inf,0)
```

```
it
```

```
attributes(it) ## Ergebnis ist Objekt der Klasse "integrate"
```

```
it$value
```

```
pmvnorm(c(0,0))
```

```
pmvnorm(c(0,0))[[1]]
```

```
library(adapt)
```

```
it <- adapt(2, c(-Inf,-Inf), c(0,0), functn=dmvnorm)
```

```
attributes(it) ## Ergebnis ist Objekt der Klasse "integration"
```

```
it$value
```

Einstieg ins Programmieren

Was ist eigentlich R?

Wie bekomme ich Hilfe?

Etwas Rechnerei zum Anfang

Daten & Dateien

Schöne bunte Welt der Grafik

Etwas Statistik

"Höhere" Mathematik

Einstieg ins Programmieren

Funktionen in R

Bedingte Anweisungen, Schleifen

"Mengenlehre"

Pakete (Packages)

DLLs

Tipps & Tricks

Literatur

Funktionen in R

```
myfun <- function(x, a){  
  r <- a*sin(x)  
  return(r)  
}  
myfun(pi/2,2)
```

```
myfun1 <- function(x, a){ a*sin(x) }    ## wie myfun, aber kürzer  
myfun1(pi/2,2)
```

```
myfun2 <- function(x, a=1){            ## optionaler Parameter  
  a*sin(x)                             ## mit Defaultwert=1  
}  
myfun2(pi/2,2)  
myfun2(pi/2)
```

```
myfun3 <- function(x, a=NULL){        ## optionaler Parameter  
                                       ## ohne Defaultwert  
  if (!is.null(a)){ a*sin(x) }else{ cos(x) }  
}  
myfun3(pi/2,2)  
myfun3(pi/2)
```

Funktionen in R (Forts.)

```
myfun4 <- function(x, a=1){  
  r1 <- a*sin(x); r2 <- a*cos(x)  
  return(list(r1=r1,r2=r2))  ## ein Ergebnis (Liste!)  
}  
myfun4(pi/2)
```

```
myfun5 <- function(x, a=1, b=2){  
  r1 <- a*sin(x); r2 <- b*cos(x)  
  return(list(r1=r1,r2=r2))  
}  
myfun5(pi/2)                ## a=1, b=2 (Defaults)  
myfun5(pi/2,1,2)           ## a=1, b=2 (beide explizit gegeben)  
  
myfun5(pi/2,2)             ## a=2, b=2 (nur a explizit gegeben)  
myfun5(pi/2,a=2)          ## a=2, b=2 (nur a explizit gegeben)  
  
myfun5(pi/2,b=3)          ## a=1, b=3 (nur b explizit gegeben)
```

→ Inputparameter können (wenn sinnvoll) weggelassen werden; mehrere Outputparameter sind eigentlich Listenelemente

Bedingte Anweisungen, Schleifen

- ▶ **R**if & **R**else

```
x<- 1; if (x==2){ print("x=2") }  
x<- 1; if (x==2){ print("x=2") }else{ print("x!=2") }
```

- ▶ **R**for & **R**repeat


```
for (i in 1:4){ print(i) }  
for (i in letters[1:4]){ print(i) }  
i <- 0; while(i<4){ i <- i+1; print(i)}  
i <- 0; repeat{ i <- i+1; print(i); if (i==4) break }
```

- ▶ weitere: **R**ifelse, **R**switch

“Mengenlehre”

```
A <- 1:3      ## Vektor als "Menge"  
B <- 2:6      ## Vektor als "Menge"  
A %in% B  
B %in% A  
  
C <- c("A", "B")  
D <- LETTERS[2:6]  
C %in% D  
D %in% C
```


Pakete (Packages)

- ▶ Packages umfassen (eine oder) mehrere Funktionen, werden mit `library(<Package-Name>)` geladen, in einem Package verfügbare Funktionen können mit `library(help=<Package-Name>)` abgefragt werden
- ▶ zum Erstellen eigener Packages gibt es zwei hilfreiche Funktionen
 - `package.skeleton(<Package-Name>)`
erstellt die Verzeichnisstruktur des Packages mit Templates für die notwendigen Files
 - `prompt(<Funktion>)`
erstellt ein Template für den Hilfetext zur Funktion
- ▶ Packages installiert man unter Windows mit dem entsprechenden Menüpunkt des GUI oder vom Kommando-Prompt
 `install.packages`
- ▶ alle Packages zu einem Thema findet man bei den **Task Views** (→ <http://cran.at.r-project.org/web/views/>), installieren kann man einen Task View z.B. mit

```
install.packages("ctv")
library("ctv")
update.views("Econometrics")
```

DLLs

→ Manual “Writing R Extensions”

(<http://cran.at.r-project.org/doc/manuals/R-exts.pdf>)

Beispiel mit einfacher C-Funktion unter Unix/Linux:

```
#include <stdlib.h>
#include <math.h>

/* Compile shared library: gcc -shared -O2 -o mydll.so mydll.c */

int mysum(double *dim, double *x, double *y, double *z)
{
    long i, n;
    n=dim[0];

    for (i=0; i<n; i++)    /* loop over obs */
    {
        z[i] = x[i] + y[i];
    }
    printf ("mysum in C\n");
    return 0;
}
```

DLLs: Aufruf in R

```
dyn.load("mydll.so")          ## DLL laden
is.loaded("mysum")            ## "mysum" ist da?

d <- 3
x <- 1:3
y <- 4:6
z <- rep(0,3)

r <- .C("mysum", dim=d, x=x, y=y, z=z )  ## das geht schief!
r$z

d <- as.double(3); x <- as.double(1:3)
y <- as.double(4:6); z <- rep(0.0,3)
r <- .C("mysum", dim=d, x=x, y=y, z=z )  ## so geht's
r$z
z                                         ## z ist immer noch 0

r <- .C("mysum", dim=d, x=x, y=y, z=z, DUP=FALSE)
                                         ## so geht's ohne Kopie der Param.
r$z
z                                         ## z enthält das Ergebnis

dyn.unload("mydll.so")          ## DLL unload
```

Tipps & Tricks

Editoren:

- ▶ Windows: TinnR (<http://sciviews.org/Tinn-R/>)
- ▶ Windows/Unix/Linux: ESS = "emacs speaks statistics" von <http://ess.r-project.org/> downloaden und in `.emacs` einbinden:
`(load "<Pfad zu ESS>/ess-5.1.24/lisp/ess-site")`
- ▶ Syntaxhighlightening für Windows gibt es auch in WinEdt (<http://cran.r-project.org/web/packages/RWinEdt/index.html>)

Alternative R-Umgebungen:

- ▶ R Commander (<http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/>) ist eine Menü-gestützte Umgebung zur Datenanalyse mit vielen vorgefertigten Analysemöglichkeiten, Installation durch:
`install.packages("Rcmdr")`
- ▶ R Studio (<http://rstudio.org/>) ist eine Alternative zum R-GUI, für verschiedene Betriebssysteme verfügbar

Tipps & Tricks (Forts.)

Programmierung:

- ▶ mit `eval` und `parse` können Zeichenketten als Ausdrücke ausgewertet werden:

```
eval(parse(text=paste("x.", as.character(1:2), " <- 0", sep="")))
print(x.1)
```

- ▶ es gibt in R zwei Verfahren für OOP: S3- und S4-Klassen; zur Information über die Komponenten von S3-Klassen (älterer Ansatz) sind die Funktionen `class`, `attributes` nützlich während für S4-Klassen (neuerer Ansatz) `getClass`, `slot`, `slotNames` verwendet werden
- ▶ Methoden können klassenabhängig sein, z.B. erhält man mit `methods(print)` alle zur Funktion gehörenden Methoden

Tipps & Tricks (Forts.)

Diverses:

- ▶ Runden und Formatieren von Zahlen geht mit `round`, `floor`, `ceiling`, `signif`, `formatC`
- ▶ Strings (character vectors) können bearbeitet werden mit `paste`, `substr`, `nchar`, `strsplit`, `toupper`, `tolower`, `sub`
- ▶ Datumsgenerierung mit `as.POSIXlt` und `strptime`, z.B.
`as.POSIXlt(strptime("20050101", "%Y%m%d"))+(0:364)*86400`
erzeugt alle Tage des Jahres 2005;
`d <- as.POSIXlt(strptime("20050926", "%Y%m%d"))`; `d$wday`
gibt den Wochentag des 26.9.2005 an
- ▶ mit `system` kann man Betriebssystemkommandos ausführen, z.B.
unter Linux: `system("cal 09 2005")`
- ▶ die Funktionen `Rxtable` (Package: `Rxtable`) und `Rlatex`
(Package: `RHmisc`) können R-Objekte als Latex-Kode speichern

Literatur

- Becker, R. A. and Chambers, J. M. (1984). *S. An Interactive Environment for Data Analysis and Graphics*. Wadsworth and Brooks/Cole, Monterey.
- Becker, R. A., Chambers, J. M., and Wilks, A. R. (1988). *The New S Language*. Chapman & Hall, London.
- Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*. Chapman & Hall, London.
- Dalgaard, P. (2002). *Introductory Statistics with R*. Springer. ISBN 0-387-95475-9.
- Gentleman, R. (2008). *R Programming for Bioinformatics*. Computer Science & Data Analysis. Chapman & Hall/CRC, Boca Raton, FL. ISBN 978-1-420-06367-7.
- Ligges, U. (2009). *Programmieren mit R*. Springer-Verlag, Heidelberg, 3rd edition. ISBN 978-3-540-79997-9, in German.
- Murrell, P. (2005). *R Graphics*. Chapman & Hall/CRC, Boca Raton, FL. ISBN 1-584-88486-X.
- Venables, W. N. and Ripley, B. D. (2000). *S Programming*. Springer. ISBN 0-387-98966-8.
- Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S. Fourth Edition*. Springer. ISBN 0-387-95457-0.
- Wollschläger, D. (2010). *Grundlagen der Datenanalyse mit R: Eine anwendungsorientierte Einführung*. Statistik und ihre Anwendungen. Springer.