Kapitel

3 Erste Schritte

Dieses Kapitel soll einen ersten Einstieg in R ermöglichen. Es wird zunächst die Installation der Basisversion und optionaler Erweiterungen, sogenannter R-Packages, beschrieben. Es folgt ein Abschnitt, der ohne in die Tiefe zu gehen, auf einfache, interaktive Weise das Kennenlernen einiger Grundstrukturen ermöglichen soll. Hierzu scheint es am besten, sich vor einen Rechner zu setzen und den Text direkt nachzuvollziehen. Man kann dabei auch gleich eigene Ideen ausprobieren und umsetzen. Der Schwerpunkt liegt auf einem spielerischen Umgang mit R.

Lernziele:

Nach Durcharbeiten dieses Kapitels haben Sie Folgendes erreicht:

- Sie kennen die R-Download-Webseite und können die R-Basisversion sowie Ergänzungen herunterladen und installieren.
- Sie können R als eine Art Taschenrechner unter Anwendung der Grundrechenarten und einfacher mathematischer Funktionen benutzen und Ergebnisse speichern.
- Sie kennen den Unterschied zwischen Zahlen und Zeichenketten in R und können diese zu Vektoren und Matrizen kombinieren.
- Sie wissen, wie man Elemente aus Vektoren und Matrizen in R ansprechen und extrahieren kann.
- Sie kennen den Begriff Data Frame, können einen solchen erstellen und die darin enthaltenen Variablen mit beschreibenden Namen versehen.
- Sie wissen, wie man in R einfache Grafiken erstellen kann.

3.1 Download und Installation von R

Dieser Abschnitt zeigt, wie man die Basisversion von R sowie Ergänzungen installiert.

3.1.1 Download

Den Download-Server erreicht man direkt unter http://CRAN.R-project.org/. CRAN steht für *Comprehensive R Archive Network*, ein Netzwerk von Webseiten in vielen Ländern rund um den Globus, auf denen identisches Material (verschiedene R-Versionen im Quellcode und als Binärprogramme, Erweiterungen und Dokumentationen) verfügbar ist. R ist plattformunabhängig, d.h., man kann R unter verschiedensten Betriebssystemen (z.B. Windows, Linux, Mac-OS) verwenden. Wir werden uns hier auf die Installation unter Windows beschränken, wobei ein Querverweis zu anderen Betriebssystemen an entsprechender Stelle gegeben wird. Zunächst muss man die gewünschte Version herunterladen. In einem Internet-Browser geht man zur Seite http://CRAN.R-project.org/. Ein Ausschnitt ist in ▶ Abbildung 3.1 dargestellt.

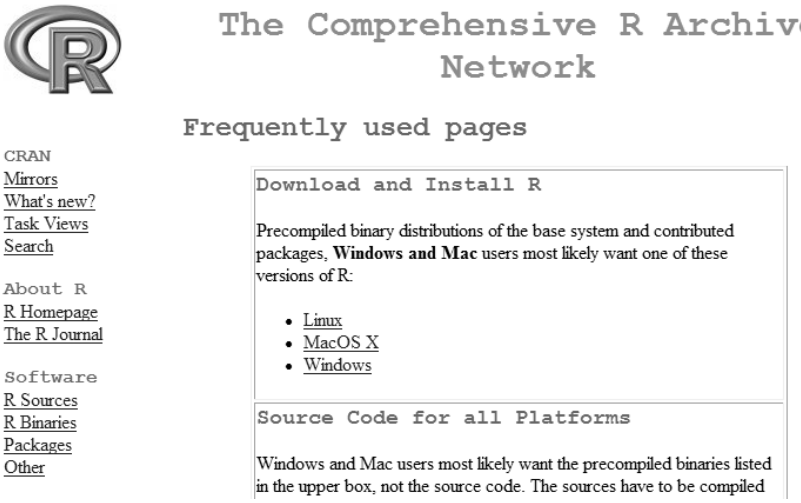

Abbildung 3.1: Ausschnitt der CRAN-Webseite (http://CRAN.R-project.org/)

Im linken Navigationsteil findet man den Link Mirrors, der zu einer Seite führt, in der man einen Server aussuchen kann, der in der Nähe liegt. Nachdem man einen Server ausgewählt hat, gelangt man zu einer Seite, die so aussieht wie in ▶ Abbildung 3.1, die allerdings jetzt vom gewählten Server kommt. Im Kasten

Download and Install R wählt man das Betriebssystem. Für Linux und Mac-OS findet man auf den sich öffnenden Seiten weitere Anweisungen. Wir klicken auf Windows und kommen zu einer Seite, die ausschnittsweise in ▶ Abbildung 3.2 dargestellt ist.

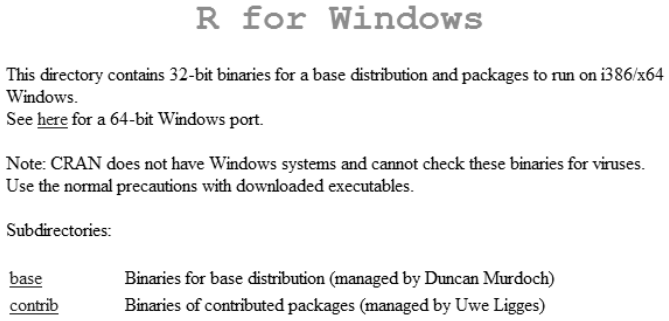

Abbildung 3.2: Ausschnitt der Windows-Download-Seite

Wenn Sie auf base klicken, erhalten Sie (die für Anfänger empfohlene) 32-bit-Version. Sollten Sie ein 64-bit-Betriebssystem verwenden, dann klicken Sie auf: See here for a 64-bit Windows port. In beiden Fällen öffnet sich die eigentliche Download-Seite (ein Ausschnitt ist in ▶ Abbildung 3.3 dargestellt). Hier findet sich immer die aktuellste R-Version.

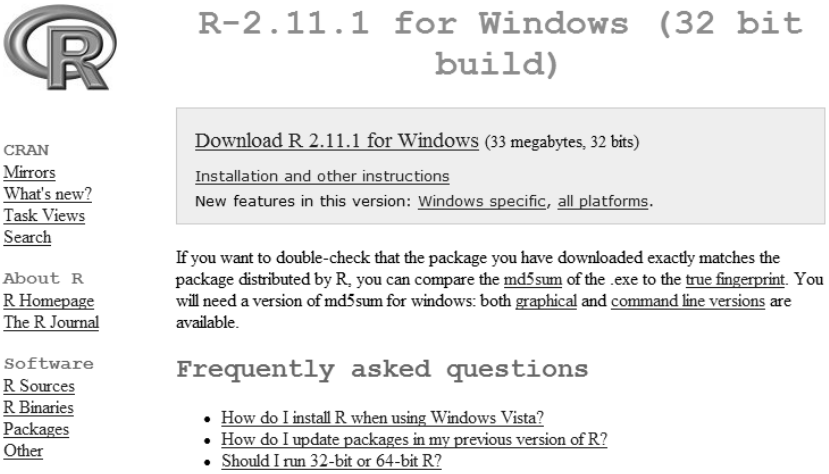

Abbildung 3.3: Ausschnitt der eigentlichen Windows-Download-Seite

Nach Klicken auf Download R x.xx.x for Windows (R 2.11.1 ist die beim Schreiben dieses Buchs verwendete Version) können Sie die R-Installationsdatei (die in unserem Fall `R-2.11.1-win32.exe` heißt) herunterladen. Wir empfehlen, sie in ein geeignetes Verzeichnis zu speichern und nicht gleich auszuführen.

3.1.2 Installation

Nach dem Herunterladen wechseln Sie in das Verzeichnis, in das Sie die Installationsdatei gespeichert haben. Sie starten die Installation durch Doppelklick auf die Installationsdatei, in unserem Beispiel `R-2.11.1-win32.exe`. Auf etwaige Fragen, ob das Programm Änderungen an Ihrem Computer durchführen darf, antworten Sie mit Ja. Es öffnet sich der Setup-Assistent. Im Prinzip können Sie bei der Installation den Voreinstellungen folgen und immer auf Weiter klicken, außer Sie wollen eine spezielle Einstellung.

Hinweis

Zu beachten ist die Wahl des Ziel-Ordners. Sie können R in ein beliebiges Verzeichnis installieren, also auch auf einen USB-Stick. Sie sollten aber beachten, dass Sie für den Zielordner Schreibrechte besitzen. Wenn Sie in Windows als Standardbenutzer angemeldet sind (und nicht als Benutzer mit Administratorrechten, bzw. Sie nicht wissen, was das ist oder was zutrifft), vermeiden Sie Verzeichnisse im System-Bereich von Windows (also z.B. die Verzeichnisse \Programme\ oder \Program Files\) und installieren R besser woanders. Wer es genau wissen will, kann weitere Informationen über die Links Installation and other instructions (für Windows XP) bzw. How do I install R when using Windows Vista? (für Windows Vista/7/Server 2008) erhalten (▶ Abbildung 3.3).

Eine Ausnahme von der Standardinstallation, die wir vorschlagen, ergibt sich bei folgendem Fenster ▶ Abbildung 3.4:

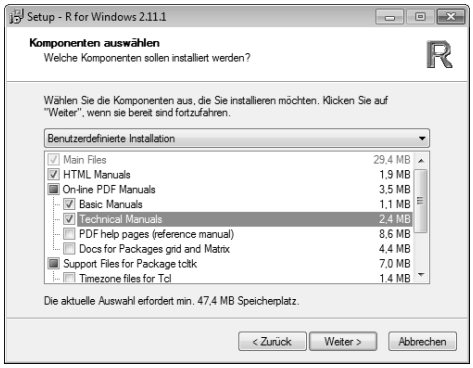

Abbildung 3.4: Auswahl von Installationskomponenten

Hier sollten Sie zusätzlich die Auswahlmöglichkeit Technical Manuals anklicken.

Es werden dann weiterführende Dokumente in PDF-Form installiert, die Sie später über die Hilfe aufrufen können. Nach Spezifikation aller Optionen wird R installiert. Zum Abschluss teilt Ihnen der Setup-Assistent mit, dass die Installation abgeschlossen ist.

3.1.3 Aufrufen und Beenden von R

Um zu prüfen, ob alles geklappt hat, wollen wir R aufrufen. Dies erfolgt entweder über das Icon am Desktop oder über das Startmenü unter ... ▷ R ▷ R 2.11.1 (statt 2.11.1 wird hier die aktuell installierte Versionsnummer stehen). Es öffnet sich das R-Fenster(▶ Abbildung 3.5).

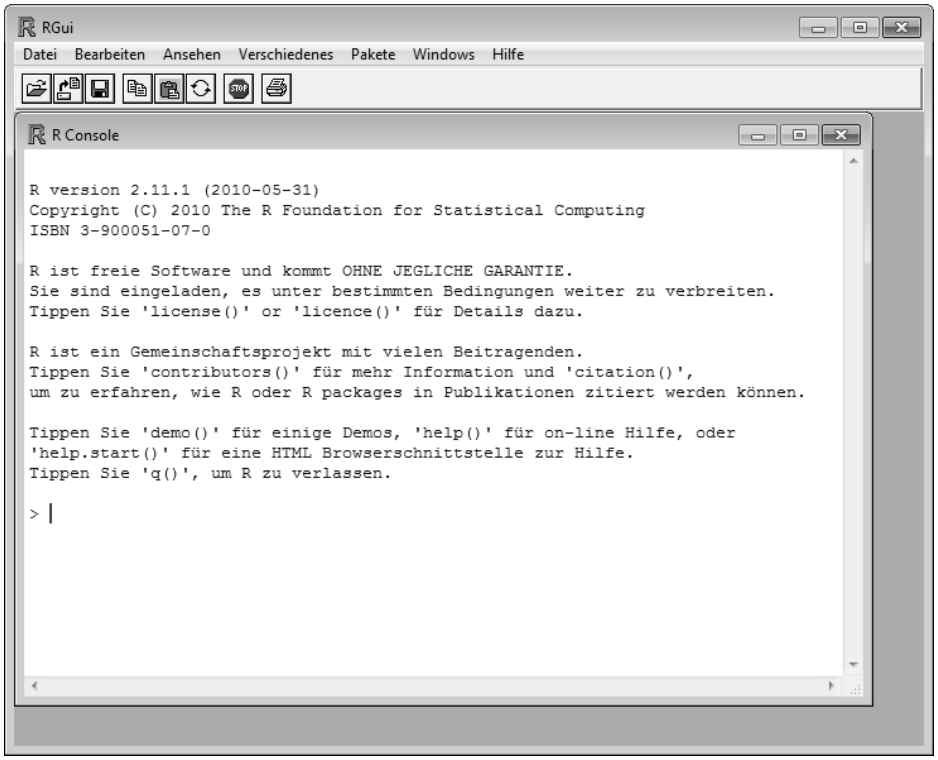

Abbildung 3.5: Das R-Fenster

Sie können R wieder beenden, indem Sie im Menü Datei ▷ Beenden oder einfach auf rechts oben klicken. Es öffnet sich dann das Fenster aus ▶ Abbildung 3.6, wo Sie mit Nein antworten. (Die Alternative JA wird in ▶ Abschnitt 5.1.2 besprochen.)

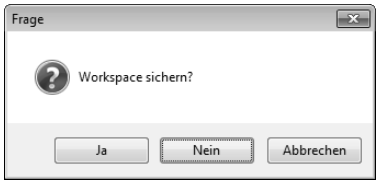

Abbildung 3.6: Dialogfenster beim Beenden von R

3.1.4 Installation von Ergänzungen (Contributed Packages)

Unter anderem ist die Erfolgsstory von R darauf zurückzuführen, dass Wissenschaftler auf der ganzen Welt Beiträge zu R liefern, die den Funktionsumfang gewaltig steigern. Diese Beiträge werden PACKAGES (oder Pakete) genannt und sind abgeschlossene Erweiterungsmodule. Sie implementieren spezialisierte statistische Methoden, erlauben Zugriff auf Daten und Hardware oder dienen zur Ergänzung bestimmter Texte oder Lehrbücher. Einige sind in der Basisversion von R enthalten, andere können von CRAN oder anderen Archiven (wie etwa http://www.bioconductor.org/) heruntergeladen werden. Momentan gibt es über 2000 Erweiterungs-Packages und ihre Zahl wächst täglich. In diesem Abschnitt wollen wir zeigen, wie man Packages installiert. In den späteren Kapiteln werden wir öfters solche Erweiterungs-Packages verwenden.

Eine Liste aller Packages, die auf CRAN verfügbar sind, sowie eine kurze Beschreibung findet man auf der CRAN-Webseite unter Packages links unten im Navigationsbereich (siehe auch ▶ Abbildung 3.1. Wir werden in diesem Buch manche dieser Packages verwenden. Diese müssen natürlich, bevor man sie verwenden will, installiert werden. Am einfachsten geht das folgendermaßen.

Zunächst müssen wir R starten (siehe ▶ Abschnitt 3.1.3). Es öffnet sich das R-Fenster (siehe auch ▶ Abbildung 3.10). In der Menüleiste ganz oben finden wir den Menüpunkt Pakete . Nach Anklicken öffnet sich ein Submenü (▶ Abbildung 3.7)

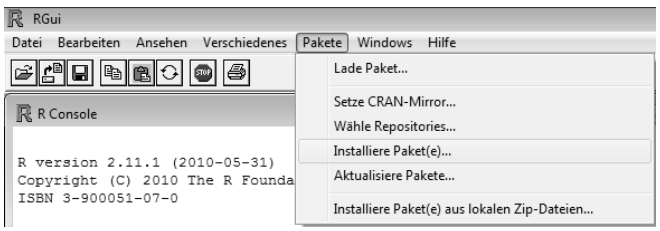

Abbildung 3.7: Das Pakete-Menü

Hier wählen wir Installiere Pakete... . Es öffnet sich ein Fenster (▶ Abbildung 3.8 links), in dem wir wieder einen nahe gelegenen Server aussuchen und OK klicken.

Im darauf folgenden Fenster (▶ Abbildung 3.8 rechts) finden wir eine Liste aller verfügbaren Packages. Wir wollen das Package maps auswählen, das wir später verwenden wollen. Nach Hinunterscrollen, Markieren von maps und Klicken von OK wird das Package installiert.

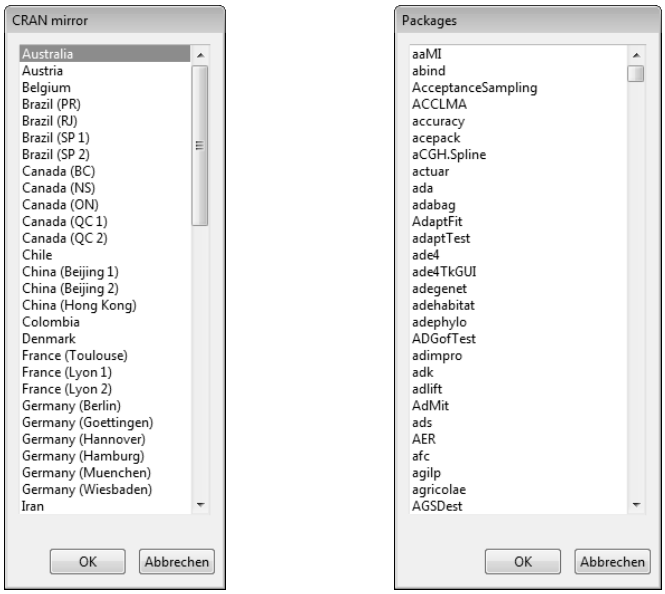

Abbildung 3.8: Auswahl des Servers und des zu installierenden Packages

Falls Windows die Frage stellt, ob eine persönliche Bibliothek eingerichtet werden soll, um dort Pakete zu installieren, antworten Sie mit Ja. Nach erfolgreichem Abschluss der Installation sehen Sie im R-Fenster eine entsprechende Meldung, die etwa so aussehen könnte (▶ Abbildung 3.9):

```
Content type 'application/zip' length 2133704 bytes (2.0 Mb)
URL geöffnet
downloaded 2.0 Mb

Paket 'maps' erfolgreich ausgepackt und MD5 Summen abgeglichen

Die heruntergeladenen Pakete sind in
        C:\Users\Ruser\AppData\Local\Temp\Rtmp3ddms6\downloaded_packages
>
```

Abbildung 3.9: Meldung zur Installation im R-Fenster

Wir haben nun die Basisversion von R (und ein Package) installiert und können beginnen, das Programm näher kennenzulernen.

3.2 Aller Anfang ist leicht

Wir starten R wie in ▶ Abschnitt 3.1.3 beschrieben. Es öffnet sich das R-Fenster (▶ Abbildung 3.10).

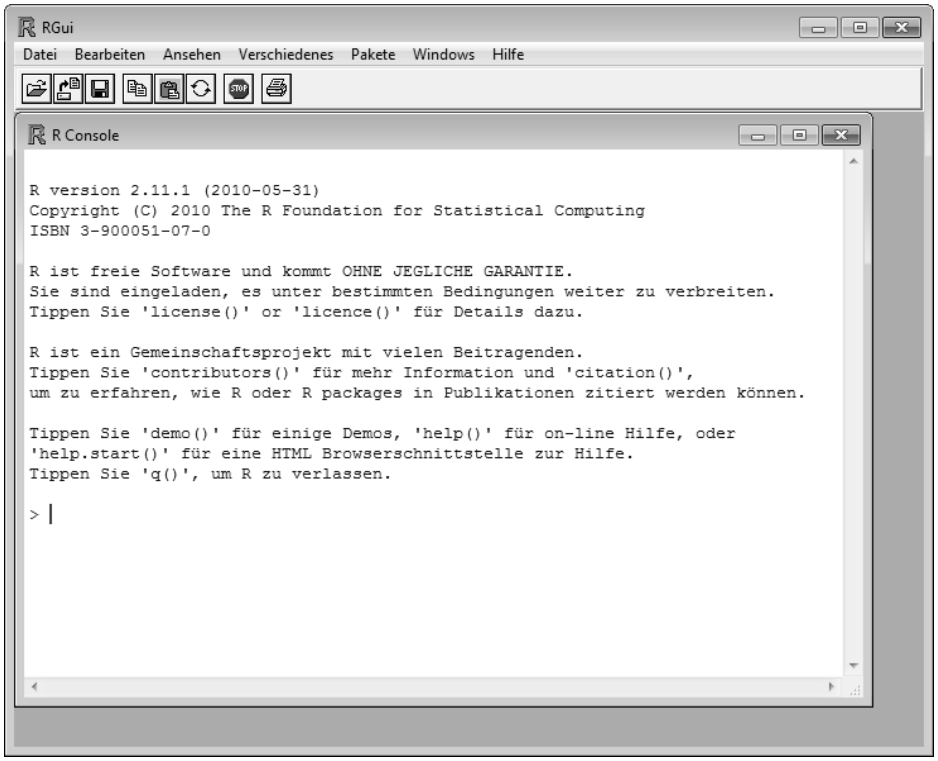

Abbildung 3.10: Das R-Fenster

Ein wesentlicher Unterschied zu den meisten Standardprogrammen, die man unter Windows verwendet, besteht darin, dass R in der Basisversion befehlsorientiert ist. Das heißt, dass man Kommandos eingeben muss, um mit R zu kommunizieren. Wie man in ▶ Abbildung 3.10 sieht, gibt es zwar einige Menüs und Schaltflächen, diese erlauben es aber nicht, Berechnungen durchzuführen oder Grafiken zu erstellen, sondern dienen eher allgemeinen Aufgaben, wie bestimmte Dateien einzulesen oder Hilfe aufzurufen. Darauf werden wir später noch näher eingehen. Im Prinzip könnte man aber alle diese Funktionen auch über Befehle ausführen.

Nachdem wir also R aufgerufen haben, sehen wir einen kurzen Text und darunter den sogenannten *Command Prompt* > (das Aufforderungszeichen für eine Eingabe) und daneben das Zeichen | für den Cursor. R wartet darauf, dass man an dieser Stelle etwas eingibt. Im einfachsten Fall ist das eine Zahl, die von R dann auch gleich ausgegeben wird. Wenn wir nach > die Zahl 13 eingeben und

die Eingabetaste ↵ drücken, sehen wir folgendes Bild (▶ Abbildung 3.11). R hat

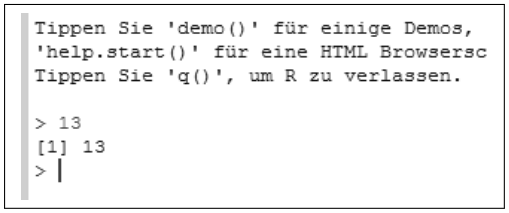

Abbildung 3.11: Ausschnitt des R-Fensters nach Eingabe einer Zahl

13 wieder ausgegeben (die Bedeutung von [1] wird später erklärt).

Im Folgenden werden die R-Befehle, die Sie eingeben sollen, und die Ausgabe, die dadurch erzeugt wird, immer so dargestellt:

R
```
> 13
```

```
[1] 13
```

Bitte beachten Sie, dass Sie in R mit der Pfeiltaste ↑ die vorher eingegebenen Befehle wieder bekommen und diese auch adaptieren können. Statt der Eingabe 13 erhalten Sie die Zahl 23, indem Sie 1 durch 2 ersetzen und dann die Eingabetaste drücken (der Cursor muss dazu nicht am Ende der Eingabezeile sein). Das ist besonders bei langen Befehlen sehr nützlich. Mit ↑ und ↓ können Sie durch die Liste der verwendeten Befehle blättern. Falls irgendetwas schief geht und Sie Fehlermeldungen erhalten, geben Sie einfach den vorherigen Befehl korrigiert nochmals ein. Wenn Sie einen Befehl nicht ordnungsgemäß beenden (zum Beispiel eine schließende Klammer vergessen) und schon die Eingabetaste gedrückt haben, dann erwartet R noch Input und zeigt das durch ein + statt > an. Falls Sie da nicht weiterwissen, drücken Sie am besten die Taste ESC. Dann wird das Vorangegangene ignoriert und Sie können am Command Prompt, d.h. an der Stelle rechts von >, wieder von Neuem beginnen.

Hinweis

Einfaches Rechnen

Wir wollen zunächst R als eine Art Rechenmaschine verwenden und die Grundrechenarten illustrieren.

R
```
> 1 + 2
```

```
[1] 3
```

R
```
> 1 - 2
```

```
[1] -1
```

R

```
> 4 * 5
```

```
[1] 20
```

Man muss zwischen den einzelnen Zeichen keine Leerzeichen eingeben (aber es erhöht die Lesbarkeit).

R

```
> 4/5
```

```
[1] 0.8
```

Hinweis

Das Resultat ist $0,8$. R stellt es aber als `0.8` dar. R verwendet den Dezimalpunkt als Dezimalzeichen. Verwenden Sie also bitte immer einen Dezimalpunkt und nicht ein Dezimalkomma. Potenzieren funktioniert so:

R

```
> 4^2
```

```
[1] 16
```

Es gibt viele Funktionen wie zum Beispiel die Exponentialfunktion

R

```
> exp(1)
```

```
[1] 2.72
```

Logarithmus

R

```
> log(1)
```

```
[1] 0
```

oder trigonometrischen Funktionen, wie z.B. Cosinus.

R

```
> cos(0)
```

```
[1] 1
```

Man kann Funktionen auch schachteln.

R

```
> log(exp(0))
```

```
[1] 0
```

Geschachtelte Funktionen werden von innen nach außen abgearbeitet. Zuerst wird e^0 berechnet, das Ergebnis = 1. Dann folgt der Logarithmus von 1, das Ergebnis ist 0.

Beachten Sie bitte, dass R *case sensitive* ist, d.h., Groß- und Kleinbuchstaben werden unterschieden. Die Eingabe von `EXP(0)` oder `Exp(0)` würde zu Fehlern führen.

Hinweis

Variablen

Zahlen können Variablen zugewiesen werden und man kann damit rechnen.

R

```
> x <- 25
> x
```

```
[1] 25
```

Der nach links gerichtete Pfeil `<-` (ein Kleiner-Zeichen und ein Bindestrich) bedeutet, dass die Zahl `25` der Variable `x` zugewiesen wird. Auf Englisch spricht man von *gets*, also „`x` gets `25`“, auf Deutsch könnte man „`x` *wird* `25`“ sagen.

Bei Zuweisungen, d.h., wird in einem R-Befehl `<-` verwendet, gibt es keinen Output. Erst durch die Eingabe der Variable wird ihr Inhalt ausgegeben. Im obigen Beispiel antwortet R nach Eingabe von `x <- 25` nur mit dem Command Prompt `>`. Erst die Eingabe von `x` zeigt, was in dieser Variable gespeichert ist.

Hinweis

Weitere Beispiele sind: Addition

R

```
> y <- x + 1
> y
```

```
[1] 26
```

Multiplikation

R

```
> x * y
```

```
[1] 650
```

oder Division

R

```
> z <- x/y
> z
```

```
[1] 0.962
```

Vektoren

Variablen können auch mehr als eine Zahl enthalten. Wir verwenden dazu die Funktion `c()` (*combine*). Die Klammern `()` deuten an, dass dazwischen kein, ein oder mehrere Argumente stehen können. Bei Funktionen müssen aber prinzipiell diese Klammern geschrieben werden. Mit Hilfe der Funktion `c()` erzeugt man Vektoren von Elementen gleichen Typs. Der Typ kann numerisch sein, wie in

R

```
> ALTER <- c(21, 24, 28)
> ALTER
```

```
[1] 21 24 28
```

Ein anderer Typ ist *character*. Die Elemente eines Character-Vektors sind Text, d.h. Zeichenketten oder engl. *strings*. Text bzw. eine Zeichenkette wird immer unter Anführungszeichen geschrieben (diese können wahlweise beide doppelt `"..."` oder beide einfach `'...'` nur nicht gemischt, z.B. `"...'`, sein).

R

```
> Geschlecht <- c("männlich", "weiblich")
> Geschlecht
```

```
[1] "männlich" "weiblich"
```

Wenn Ziffern bzw. Zahlen unter Anführungszeichen stehen, sind sie auch vom Typ character (z.B. `"15"` oder `"q20"`).

Mit `c()` kann man auch schon bestehende Variablen erweitern.

R

```
> ALTER <- c(22, ALTER, 39)
> ALTER
```

```
[1] 22 21 24 28 39
```

Hier wurde am Anfang 22 und am Ende 39 hinzugefügt, in der Mitte blieben die ursprünglichen Werte von `ALTER`, nämlich 21, 24 und 28. Die neuen Werte wurden wieder der Variablen `ALTER` zugewiesen, wodurch sich die ursprüngliche Variable `ALTER` geändert hat und nun fünf Elemente enthält.

Die Funktion `length()` gibt die Anzahl der Elemente einer Variable aus.

R

```
> length(ALTER)
```

```
[1] 5
```

R verfügt über eine Vielzahl unterschiedlicher Funktionen, die man auf Variable anwenden kann. Wir erhalten zum Beispiel den Mittelwert einer Variable mit der Funktion `mean()`.

```
> mean(ALTER)
```

```
[1] 26.8
```

Die Auswahl von Elementen eines Vektors kann durch direkte Angabe der entsprechenden Indizes (Positionen) erfolgen. Den zweiten Wert von `ALTER` erhält man, indem man die Zahl 2 in eckigen Klammern hinter `ALTER` anfügt.

```
> ALTER[2]
```

```
[1] 21
```

Man kann auch mehrere Indizes (als Vektoren) angeben. Den ersten, zweiten und vierten Wert erhält man mit

```
> ALTER[c(1, 2, 4)]
```

```
[1] 22 21 28
```

Den ersten, zweiten und dritten Wert hätten wir auch so spezifizieren können

```
> ALTER[1:3]
```

```
[1] 22 21 24
```

Der Doppelpunkt `:` ist hierbei eine Abkürzung und bedeutet „von-bis“, hier von 1 bis 3. Diese Möglichkeit ist natürlich dann besonders nützlich, wenn man fortlaufende Sequenzen wie z.B. die Zahlen von 1 bis 50 erzeugen will.

```
> x <- 1:50
> x
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
[19] 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
[37] 37 38 39 40 41 42 43 44 45 46 47 48 49 50
```

Man kann diese Sequenz auch umdrehen:

```
> x <- 50:1
> x
```

```
 [1] 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33
[19] 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15
[37] 14 13 12 11 10  9  8  7  6  5  4  3  2  1
```

Nun erkennen wir auch die Bedeutung der Zahlen in eckigen Klammern am Zeilenanfang, wenn wir uns den Wert eines Vektors anzeigen lassen. Es handelt sich um den Index der unmittelbar folgenden Komponente des Vektors. Wir sehen z.B. neben `[19] 32` und neben `[37] 14`. Das bedeutet, dass die 19. Zahl `32` und die 37. Zahl `14` ist.

Statt der combine-Funktion `c()` kann man zum Indizieren auch Variablen (Vektoren) verwenden, in denen die Indizes stehen. (Für diesen Index-Vektor können wir natürlich einen beliebigen Namen verwenden, also z.B. `u`.)

R

```
> u <- 3:5
> ALTER[u]
```

```
[1] 24 28 39
```

Man kann einzelnen (aber auch mehreren) Elementen einer Variable auch neue Werte zuweisen. Die aktuell in ALTER gespeicherten Werte sind

R

```
> ALTER
```

```
[1] 22 21 24 28 39
```

Wir ändern das dritte Element von `ALTER` auf `42` mit

R

```
> ALTER[3] <- 42
> ALTER
```

```
[1] 22 21 42 28 39
```

Generell werden Variable in R mit eckigen Klammern indiziert. Werden keine eckigen Klammern angegeben, sind immer alle Elemente der Variable gemeint.

Es gibt auch die Möglichkeit, negative Werte als Indizes zu verwenden, zum Beispiel

R

```
> ALTER[-3]
```

```
[1] 22 21 28 39
```

Dabei bedeutet ein Minus (`-`) alle Elemente ohne die mit den angegebenen Indizes. Diese Indizes können auch in Vektoren enthalten sein, wie z.B. im oben definierten `u`.

```
> ALTER[-u]
```

```
[1] 22 21
```

Verwendet man den Doppelpunkt (`:`), um mit Minus (`-`) Werte aus einem Vektor auszuschließen, dann muss man Klammern verwenden, also `-(1:3)` und nicht `-1:3`, da man sonst die Werte von -1 bis 3 erzeugen würde.

Hinweis

Möchte man mehrere Variable in Tabellenform anordnen, gibt es dafür in R das Konzept einer Matrix. Um das zu veranschaulichen, wollen wir noch zwei weitere Variablen definieren, nämlich `GEWICHT` und `GRÖSSE`.

```
> GEWICHT <- c(56, 63, 80, 49, 75)
> GRÖSSE <- c(1.64, 1.73, 1.85, 1.6, 1.81)
```

Matrizen

Mit dem Befehl `cbind()` kann man Vektoren spaltenweise, also nebeneinander (von links nach rechts) zusammenfügen (*c* in `cbind()` steht für *columns*).

```
> X <- cbind(ALTER, GEWICHT, GRÖSSE)
> X
```

```
     ALTER GEWICHT GRÖSSE
[1,]    22      56   1.64
[2,]    21      63   1.73
[3,]    42      80   1.85
[4,]    28      49   1.60
[5,]    39      75   1.81
```

(Die Bedeutung der eckigen Klammern wird später erklärt). Das Gleiche geht auch zeilenweise (von oben nach unten) mit dem Befehl `rbind()` (das *r* steht für *rows*).

```
> Z <- rbind(ALTER, GEWICHT, GRÖSSE)
> Z
```

```
         [,1]  [,2]  [,3] [,4]  [,5]
ALTER   22.00 21.00 42.00 28.0 39.00
GEWICHT 56.00 63.00 80.00 49.0 75.00
GRÖSSE   1.64  1.73  1.85  1.6  1.81
```

Die Vektoren in `cbind()` bzw. `rbind()` sollten gleich lang sein. Beide Befehle produzieren hier eine Matrix.

Wie Vektoren können Matrizen mit eckigen Klammern indiziert werden, allerdings muss man jetzt zwischen Zeilen und Spalten unterscheiden. Der Index einer Matrix enthält also jetzt zwei Spezifikationen. Will man z.B. aus der Matrix `X` den Wert, der in der 2. Zeile und in der 3. Spalte steht, dann spezifiziert man

R

```
> X[2, 3]
```

```
GRÖSSE
  1.73
```

Die erste Zahl steht für die Zeile und die zweite für die Spalte. Wie oben kann man auch Vektoren zum Indizieren verwenden. Die ersten drei Zeilen und die ersten zwei Spalten von `X` erhält man z.B. mit

R

```
> X[u, 1:2]
```

```
     ALTER GEWICHT
[1,]    42      80
[2,]    28      49
[3,]    39      75
```

Lässt man einen der beiden Indizes aus (das trennende Komma bleibt aber), dann erhält man alle Werte der entsprechenden Zeile oder Spalte, für die man einen Index angegeben hat.

R

```
> X[3, ]
```

```
ALTER GEWICHT  GRÖSSE
42.00   80.00    1.85
```

Diese Darstellungsform, also `[3,]`, haben wir schon bei der Ausgabe der Matrix `X` gesehen. Links neben den Werten der Matrix war [1,], [2,] etc. zu sehen. Dort betraf es die Beschriftung der Zeilen, wobei [1,] bedeutet: Werte aus Zeile 1, über alle Spalten. Oberhalb der Matrix waren die Spalten für `X` mit den Namen der Variablen, die wir zur Erzeugung der Matrix verwendet haben, beschriftet. Jede Matrix kann nämlich Zeilen- und Spaltennamen haben, die man mit den Funktionen `rownames()` bzw. `colnames()` spezifizieren, aber auch abfragen kann. So nennt uns R die Spaltennamen von `X` mit

R

```
> colnames(X)
```

```
[1] "ALTER"   "GEWICHT" "GRÖSSE"
```

Wir können den Zeilen von `X` neue Namen geben, zum Beispiel

```
> namen <- c("Gerda", "Karin", "Hans", "Doris", "Ludwig")
> rownames(X) <- namen
> X
```

```
        ALTER GEWICHT GRÖSSE
Gerda      22      56   1.64
Karin      21      63   1.73
Hans       42      80   1.85
Doris      28      49   1.60
Ludwig     39      75   1.81
```

In unserem Beispiel oben, als wir mit `cbind()` die Matrix `X` erzeugt haben, hat R die Spaltennamen automatisch vergeben, da die Variablennamen `ALTER` etc. schon definiert waren.

Einzelne Elemente einer Matrix, aber auch einzelne Spalten bzw. Zeilen kann man mit den Namen indizieren. Will man das Gewicht von Doris, so kann man schreiben

```
> X["Doris", "GEWICHT"]
```

```
[1] 49
```

Alle Werte von Doris (also die vierte Zeile) erhält man mit

```
> X["Doris", ]
```

```
  ALTER GEWICHT  GRÖSSE
   28.0    49.0     1.6
```

Bei Verwendung der Namen kann man aber : als Abkürzung nicht verwenden.

Eine Matrix kann, wie ein Vektor, nur Elemente eines bestimmten Typs enthalten, z.B. müssen alle Elemente numerisch oder Textzeichen (character) sein. Verwendet man beide Typen gemeinsam, dann wandelt R die numerischen Vektoren in Text-Vektoren um und das Resultat ist eine Matrix, die nur aus Text-Elementen besteht.

Hinweis

Bestimmte Eigenschaften einer Matrix lassen sich abfragen. Die Dimension, also Größe der Matrix erhält man mit der Funktion `dim()`, zum Beispiel

```
> dim(X)
```

```
[1] 5 3
```

Die erste Zahl ist die Anzahl der Zeilen, die zweite die Anzahl der Spalten. Die Anzahl aller Elemente (also Anzahl Zeilen mal Anzahl Spalten) erhält man wie vorher mit

R

```
> length(X)
```

```
[1] 15
```

Data Frames

Zum Abschluss wollen wir noch das Konzept von Data Frames vorstellen, wo verschiedene Datentypen gleichzeitig in einer matrixähnlichen Struktur vorkommen können.

Eine Möglichkeit, einen Data Frame zu erzeugen, bietet die Funktion `data.frame()`, die ähnlich wie `cbind()` funktioniert. Nur lassen sich damit Vektoren (auch Matrizen) spaltenweise zusammenfassen, die sowohl numerisch als auch character sein können. Die einzige Einschränkung ist, dass die Länge aller Vektoren (bzw. die Anzahl der Zeilen der Matrizen) gleich ist.

Als Beispiel wollen wir zur Matrix `X` davor noch eine Spalte hinzufügen, die das Geschlecht beschreibt. Dazu erzeugen wir zunächst einen Vektor `SEX`. Dann erstellen wir einen Data Frame, den wir `gewichtsdaten` nennen wollen, und zeigen ihn an.

R

```
> SEX <- c("weiblich", "weiblich", "männlich", "weiblich",
+     "männlich")
> gewichtsdaten <- data.frame(SEX, X)
> gewichtsdaten
```

```
             SEX ALTER GEWICHT GRÖSSE
Gerda   weiblich    22      56   1.64
Karin   weiblich    21      63   1.73
Hans    männlich    42      80   1.85
Doris   weiblich    28      49   1.60
Ludwig  männlich    39      75   1.81
```

Wie bei Matrizen, können Zeilen und/oder Spalten von Data Frames mit eckigen Klammern indiziert werden. Zum Beispiel erhalten wir die Daten von `Doris` wie schon oben mit

R

```
> gewichtsdaten["Doris", ]
```

```
             SEX ALTER GEWICHT GRÖSSE
Doris weiblich    28      49    1.6
```

Data Frames bieten einen natürlichen Rahmen, wie man statistische Daten repräsentieren kann, da oft sowohl metrische als auch kategoriale Variable in einem Datensatz vorkommen. Wir werden auf Data Frames in ▶ Abschnitt 4.3 detaillierter eingehen.

Natürlich kann man mit Vektoren, Matrizen und den Inhalten von Data Frames rechnen, Funktionen auf sie anwenden und sie auf beinahe beliebige Weise weiterverwenden. Vor allem statistische Funktionen werden in diesem Buch noch ausführlich behandelt werden. Dieser Abschnitt sollte Ihnen einen ersten Einstieg ermöglichen und zeigen, dass die grundsätzliche Bedienung von R im Gegensatz zu oft gehörten Meinungen eigentlich ganz einfach ist. Die Komplexität und Mächtigkeit von R entsteht vor allem aus der Kombination von, für sich genommen, relativ einfachen Konzepten.

Einfache Grafiken

Ein besondere Eigenschaft von R ist die Möglichkeit, maßgeschneiderte, druckreife Grafiken zu erstellen. Zur Illustration zeigen wir hier nur ein paar einfache Möglichkeiten, Details werden später behandelt (▶ Abschnitt 5.2). Wollen wir z.B. die `GRÖSSE` unserer fiktiven Personen darstellen, könnten wir die Funktion `barplot()` verwenden.

R

```
> barplot(GRÖSSE)
```

Es öffnet sich ein eigenes Grafikfenster (▶ Abbildung 3.12), in dem R den Output von Grafikfunktionen darstellt. Wir sehen für jede Person einen Balken, die Höhe des jeweiligen Balkens entspricht der Körpergröße. (Wir könnten uns vorstellen, es stehen fünf schematisierte Personen nebeneinander, die wir bezüglich ihrer Körpergröße vergleichen wollen.)

Ein weiteres Beispiel ist `plot()`, die vielleicht wichtigste Grafikfunktion in R. Aus der Vielzahl an Möglichkeiten, die `plot()` bietet, wollen wir zwei herausgreifen.

In einem datenanalytischen Kontext ist es oft wichtig zu untersuchen, ob zwei Variablen in Zusammenhang stehen. Bei den von uns vorher verwendeten Variablen könnten das Gewicht und Körpergröße sein, die wir gegeneinander in einem sogenannten Streudiagramm (siehe auch ▶ Abschnitt 9.1.1) darstellen wollen. In der `plot()`-Funktion gibt man dann die beiden Variablen an, wobei die erste auf der x-Achse und die zweite auf der y-Achse dargestellt wird. Der Befehl ist

R

```
> plot(GRÖSSE, GEWICHT)
```

Im Output (▶ Abbildung 3.13) sieht man für jede Person einen Punkt, der ihr Gewicht und ihre Körpergröße beschreibt. Man sieht auch: je größer eine Person

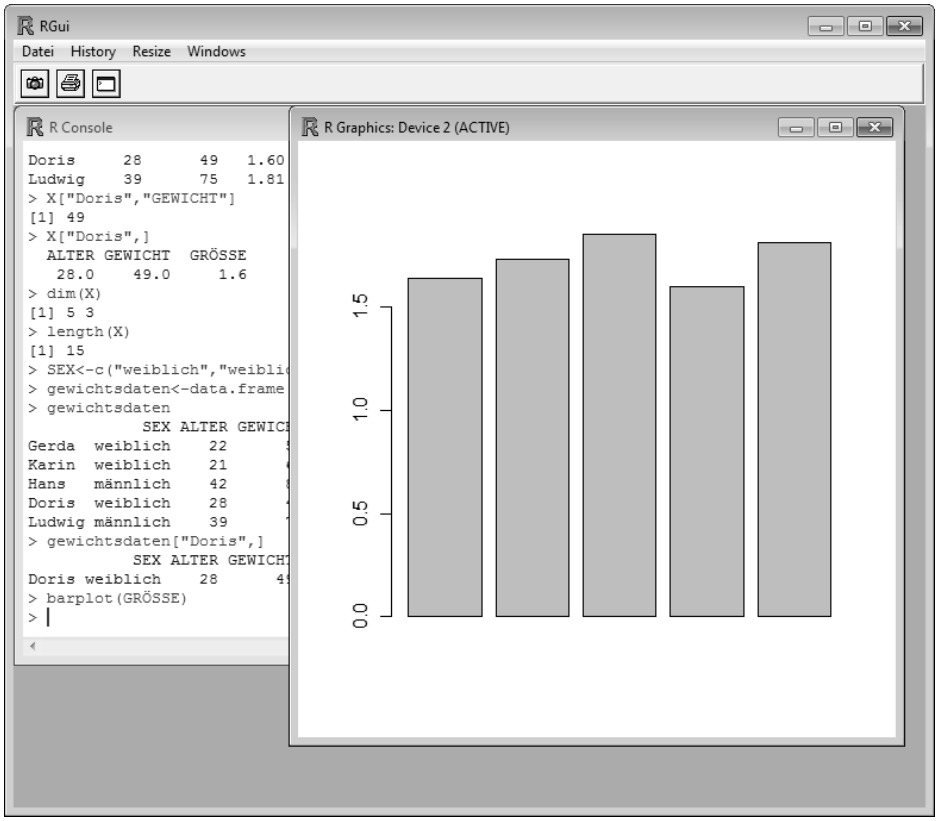

Abbildung 3.12: Einfache R-Grafik

ist, umso schwerer ist sie. Es besteht also ein Zusammenhang zwischen den beiden Variablen.

Eine andere, sehr praktische Verwendungsmöglichkeit von `plot()` ergibt sich, wenn man eine mathematische Funktion visualisieren möchte. Zum Beispiel könnten wir die Logarithmusfunktion folgendermaßen darstellen. Mit dem Operator `:` erzeugen wir eine Sequenz von Zahlen, für die wir dann die Funktionswerte ausrechnen. Legen wir also für `x` die Zahlen 1 bis 10 fest und berechnen wir die logarithmierten Werte von `x`, die wir in `y` speichern. Dann erhalten wir eine Grafik der Funktion (▶ Abbildung 3.14) so:

R

```
> x <- 1:10
> y <- log(x)
> plot(x, y)
```

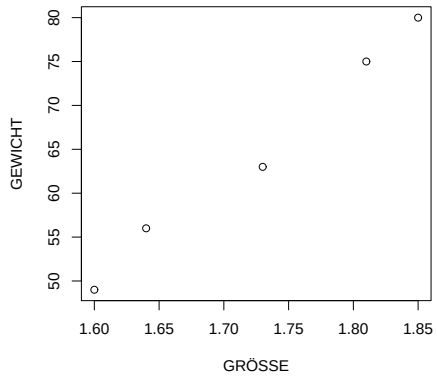

Abbildung 3.13: Einfache R-Grafik

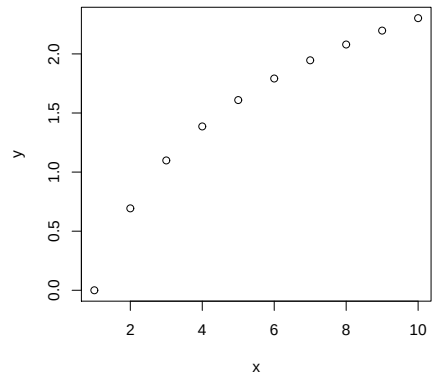

Abbildung 3.14: Die Logarithmusfunktion

Die bisher dargestellten Grafiken sind bewusst einfach gehalten. Im Prinzip lassen sie sich nahezu beliebig erweitern und an spezielle Anforderungen anpassen. Dazu gehören Änderungen der Skalierung und Beschriftungen der Achsen, Legenden, Hinzufügen weiterer Elemente wie Linien oder Texte, Farben und vieles mehr. Wir werden darauf in ▶ Abschnitt 5.2 detaillierter eingehen.

Zum Abschluss dieser Einführung wollen wir noch eine komplexere Grafik demonstrieren. Hierzu benötigen wir das R-Package maps (Becker et al., 2010), das wir über die Menüpunkte Pakete ▷ Installiere Pakete... installieren müssen (▶ Abschnitt 3.1.4). In diesem Package sind geografische Daten für mehrere Länder enthalten. Nachdem wir das Package installiert und mit `library()` geladen haben, erhalten wir eine Landkarte von z.B. Italien (▶ Abbildung 3.15) mittels

R

```
> library("maps")
> map("italy")
```

Abbildung 3.15: Eine Landkarte von Italien

Einen Überblick, was in R alles an Grafiken möglich ist, gibt die *R Graph Gallery* (http://addictedtor.free.fr/graphiques/).

3.3 R-Befehle im Überblick

`+ - * / ^` Grundrechenarten. Zum Beispiel: `1 + 3`

`<-` Zuweisung. Daten werden in eine Variable gespeichert. Zum Beispiel: `x <- 25`. `x` hat jetzt den Wert 25 bzw. `x` „wird“ 25.

`:` Erzeugt eine Sequenz von ganzen Zahlen von:bis. Zum Beispiel: `1:3` ergibt die Zahlen 1,2,3.

`[] bzw. [,]` dient zur Indizierung von Vektoren, Matrizen oder Data Frames. Zum Beispiel: `ALTER[2]` beschreibt das 2. Element des Vektors `ALTER`.

`barplot(height)` erzeugt ein Balkendiagramm. Die Höhe der Balken wird mit `height` angegeben.

`c(...)` Kombiniert das, was unter `...` steht, zu einem Vektor. Zum Beispiel: c(21, 24, 28)

`cbind()` Vektoren werden nebeneinander, spaltenweise zusammengefügt (von links nach rechts, *c* in `cbind()` steht für *columns*), es entsteht eine Matrix.

`colnames(x)` Setzen oder Abrufen von Spaltennamen einer Matrix oder eines Data Frame.

`cos(x)` Kosinusfunktion.

`data.frame(...)` erzeugt einen Data Frame, d.h. eine rechteckige Kollektion von Vektoren, die verschiedenen Typs sein dürfen. Sie werden durch Kommas getrennt an der Stelle ... spezifiziert. Ansonsten verhalten sich Data Frames ähnlich wie Matrizen. Sie bilden die fundamentale Datenstruktur für viele Statistikfunktionen in R.

`dim(x)` Setzen oder Abrufen der Dimension einer Matrix oder eines Data Frame (d.h. Anzahl der Zeilen bzw. Anzahl der Spalten)

`exp(x)` Exponentialfunktion (zur Basis e)

`length(x)` Setzen oder Abrufen der Länge eines Vektors (d.h. Anzahl der Elmeente)

`library(package)` Laden und zur Verfügungstellung von (installierten) Packages

`log(x)` Logarithmusfunktion (natürlicher Logarithmus)

`mean(x)` Arithmetisches Mittel

`plot(x, y, ...)` Funktion zum Plot von R-Objekten. Im einfachen Fall werden Punkte mit den Koordinaten x und y dargestellt.

`rbind()` Vektoren werden untereinander, zeilenweise zusammengefügt (von oben nach unten, *r* in `rbind()` steht für *rows*), es entsteht eine Matrix.

`rownames(x)` Setzen oder Abrufen von Zeilennamen einer Matrix oder eines Data Frame

3.4 Übungen

1. Weisen Sie einer Variable `x` den Wert 15 zu und erstellen Sie einen Vektor `y` mit den Werten {1, 2, 3, 10, 100}. Multiplizieren Sie diese miteinander und speichern Sie das Ergebnis in einem neuen Objekt `z`. Bilden Sie anschließend die Summe aller Elemente von `z`.
2. Erzeugen Sie eine Sequenz von 0 bis 10 und eine Sequenz von 5 bis −5.
3. Erzeugen Sie eine Sequenz von −3 bis + 3 in 0.1-Schritten. (Tipp: Erzeugen Sie die Sequenz von −30 bis 30 und dividieren Sie durch 10.)
   Zeichnen Sie die Funktion $y = x^2$, wobei Sie für x die erzeugte Sequenz verwenden. Wie sieht die Funktion für $y = 2 + x^2$ bzw. $y = 5 - x^2$ aus?
4. Definieren Sie zwei Vektoren mit folgenden Daten: `t` enthält {mo, di, mi, do, fr, sa} und `m` enthält {90, 80, 50, 20, 5, 50}.
   Verbinden Sie beide Vektoren spaltenweise zu einer Matrix mit 5 Zeilen und 2 Spalten und speichern Sie diese im Objekt `studie` ab. Vergeben Sie anschließend die Spaltennamen *Wochentag* für `t` und *Motivation* für `m`.
   Fügen Sie nun am unteren Ende der Matrix eine Zeile mit den Elementen {so, 100} hinzu und überschreiben Sie mit dem Ergebnis das Objekt `studie`.
5. Gehen Sie genauso vor wie im vorigen Beispiel, aber erzeugen Sie einen Data Frame (den Sie `studie2` nennen) anstelle einer Matrix.

Datenfiles sowie Lösungen finden Sie auf der Webseite des Verlags.

CWS