

Karsten Rincke

# Statistik mit R

Ergänzungen zu  
Rasch, Frieze, Hofmann & Naumann (2010)  
Quantitative Methoden  
Band 1 (3. Auflage). Heidelberg: Springer



Universität Regensburg  
Didaktik der Physik

Der vorliegende Text entspricht dem Bearbeitungsstand vom 21. November 2013. Ich freue mich über Anregungen und Rückfragen: [Karsten.Rincke@physik.uni-regensburg.de](mailto:Karsten.Rincke@physik.uni-regensburg.de)  
Die Herstellung des Textes erfolgt ausschließlich unter Verwendung quelloffener und kostenloser Software.  
Satz: L<sup>A</sup>T<sub>E</sub>X mit hyperref, der Fließtext ist in MinionPro gesetzt, Überschriften in Myriad, für die Grafik: R unterstützt durch Gimp.

# Inhaltsverzeichnis

<b>1</b>	<b>Zur Einführung</b>	<b>5</b>
<b>2</b>	<b>Grundlagen zum Umgang mit R</b>	<b>7</b>
2.1	Einlesen des Beispieldatensatzes	7
2.2	Datentypen	7
2.3	Datenstrukturen	8
2.3.1	Vektoren	8
2.3.2	Matrizen	8
2.3.3	Dataframes	9
2.3.4	Listen	9
2.3.5	Arrays	9
2.3.6	Faktoren	9
2.4	Anzeigen oder Eingeben von Daten	10
2.5	Elementare Rechenoperationen	11
2.6	Daten adressieren	11
2.6.1	Einzelne Datenpunkte auswählen	11
2.6.2	Datenbereiche auswählen	12
2.6.3	Daten filtern	12
2.7	Eingabe und Ausgabe von R	13
<b>3</b>	<b>Ergänzungen zu Kapitel 1: Deskriptive Statistik</b>	<b>15</b>
3.1	Grafiken erzeugen und speichern	15
3.1.1	Histogramme	15
3.1.2	Grafiken speichern	16
3.1.3	Kreisdiagramme	16
3.2	Statistische Kennwerte	18
3.3	z-Standardisierung	19
<b>4</b>	<b>Ergänzungen zu Kapitel 2: Inferenzstatistik</b>	<b>21</b>
4.1	Konfidenzintervalle	21
4.2	Balkendiagramme mit Angabe von Konfidenzintervallen	22
4.2.1	Auswählen der benötigten Daten	22
4.2.2	Bestimmen der Zeichenkoordinaten für die Konfidenzintervalle	22
4.2.3	Zeichnen des Diagramms	23
4.3	Balkendiagramme mit Angabe von Fehlerbalken	24
4.3.1	Definieren einer eigenen Funktion zum Zeichnen	24
4.3.2	Weitere effizientere Möglichkeiten zur Erzeugung von Balkendiagrammen	25

## 5 Ergänzungen zu Kapitel 3:

### Der t-Test

**27**

5.1	Durchführung eines t-Tests für unabhängige Stichproben . . . . .	27
5.1.1	Test auf Varianzhomogenität . . . . .	28
5.2	Durchführung eines t-Tests für abhängige Stichproben . . . . .	29
5.3	Durchführung eines t-Tests für eine Stichprobe . . . . .	31
5.4	Vertiefung: Vergleich t-Test für unabhängige und abhängige Stichproben in $R$ . . . . .	32
5.4.1	Bestimmung der Korrelation . . . . .	32

## 6 Ergänzungen zu Kapitel 4:

### Merkmalszusammenhänge

**35**

6.1	Streudiagramme . . . . .	35
6.2	Korrelationen . . . . .	37
6.2.1	Signifikanztest für Korrelationen . . . . .	37
6.3	Partialkorrelationen . . . . .	38
6.3.1	Signifikanztest für Partialkorrelationen . . . . .	39
6.4	Lineare Regression . . . . .	39
6.4.1	Durchführung und Interpretation an einem Beispiel . . . . .	39
6.4.2	Zusammenhang zwischen Korrelation, Regression und t-Test . . . . .	43

# 1 Zur Einführung

Der vorliegende Text möchte Hilfe und Anregung dazu sein, die Aufgaben und Beispiele im Buch *Quantitative Methoden* mit der Statistik-Programmierungsumgebung *R* zu bearbeiten. Der Text ist ab Kapitel 3 inhaltlich eng an die »Ergänzungen zu SPSS« angelehnt, die sich auf der Seite <http://www.quantitative-methoden.de> finden. *R* ist hoch leistungsfähig, quelloffen, kostenfrei und für alle gängigen Betriebssysteme verfügbar. Für nähere Informationen siehe unter anderem <http://www.r-project.org/>. *R* wird im Wesentlichen durch Befehle in Textform gesteuert. Es werden also keine Zeilen oder Spalten in einer Tabelle mit der Maus markiert und dann in ein anderes Fenster kopiert oder dergleichen. Viele grundsätzliche Eigenschaften des Programms werden der Leserin und dem Leser ungewohnt sein. Auch wenn sich der Text nicht als systematische Einführung in *R* versteht, werden die wichtigsten im vorliegenden Text gebrauchten Befehle in Kapitel 2 erklärt. Das bedeutet nicht, dass dieses Kapitel unbedingt durchzuarbeiten wäre, bevor mit der Arbeit an den Beispielen begonnen werden kann. Wer direkt in die Arbeit mit den Beispielen einsteigen möchte, findet an entsprechender Stelle Verweise auf das Grundlagenkapitel.

Zur Installation und zum Bedienkonzept von *R* findet man auf der Projektseite des Programms und an vielen anderen Stellen im Datennetz Anleitungen in Deutsch oder Englisch.

Hyperlinks sind im vorliegenden Text grundsätzlich aktiv, sie können also aus dem vorliegenden Pdf heraus direkt mit der Maus angewählt werden.



## 2 Grundlagen zum Umgang mit R

### 2.1 Einlesen des Beispieldatensatzes

Um mit dem Beispieldatensatz arbeiten zu können, muss dieser eingelesen werden. Da R nicht mit einer grafischen Benutzeroberfläche, sondern befehlsorientiert arbeitet, erfolgt das Einlesen nicht wie das Öffnen eines Textdokuments, wie man es vielleicht von Office-Programmen kennt. Das Einlesen erfolgt mit einem Befehl, mit dem der Datensatz dem Programm bekannt gemacht wird, ohne dass der Datensatz selbst nun auf dem Bildschirm erscheint:

```
> daten <- read.spss(file="BeispieldatensatzA3.sav",  
                     to.data.frame=TRUE)
```

In Verbindung mit dieser Befehlszeile ist das Folgende zu beachten:

- Wenn das Programm nach dem Drücken der Enter-Taste »nichts« macht, dann ist das ein gutes Zeichen – der Datensatz wurde erfolgreich eingelesen. Wenn Sie daran zweifeln, geben Sie nun einfach `daten` ein. Der Datensatz rauscht nun über den Bildschirm: R teilt mit, was es mit dem Objekt »daten« verbindet.
- Die Befehlszeile führt nur dann zum gewünschten Ziel, wenn sich der Beispieldatensatz im aktuellen Arbeitsverzeichnis befindet. Sollte er sich an anderer Stelle im Dateisystem befinden, muss der Pfad entsprechend angegeben sein: `...file="<Pfad>BeispieldatensatzA3.sav"`
- Der Befehl `read.spss` liest einen Datensatz ein, der mit dem Programm SPSS gespeichert wurde, also etwa einen solchen, den Sie als SPSS-Beispieldatensatz auf der Seite <http://www.quantitative-methoden.de> finden. Die Option `to.data.frame=TRUE` sorgt dafür, dass das Resultat in R als Dataframe behandelt wird (siehe dazu Abschnitt 2.3.3).

Weitere Informationen zum Importieren solcher Daten finden Sie in der Anleitung unter [http://de.wikibooks.org/wiki/GNU\\_R:\\_Datenimport\\_und\\_-export#Import\\_aus\\_SPSS](http://de.wikibooks.org/wiki/GNU_R:_Datenimport_und_-export#Import_aus_SPSS). Der dort genannte Befehl `install.packages` wird direkt in R ausgeführt!

- Der Befehl `read.spss()` entstammt der Bibliothek `foreign`, die mit `library(foreign)` aktiviert werden muss.
- Innerhalb von Dateinamen sollten Sie die Verwendung von Grundstrichen `_` und Bis-Strichen – vermeiden, weil diese vom Programm unter Umständen als Teile von Befehlen und damit fehlinterpretiert werden.

### 2.2 Datentypen

R kennt unterschiedliche Datentypen. Die wichtigsten Datentypen sind

- Zahlen (numeric),
- Buchstaben (character),
- logische Werte (logical).

Um angesichts eines vorliegenden Wertes festzustellen, von welcher Art er ist, gibt es den Befehl `mode()`, für Beispiele siehe Abschnitt 2.3.1. Man kann den Datentyp an die Erfordernisse anpassen, und zwar zum Beispiel mit den Befehlen `as.character()` und `as.numeric()`. Das folgende Beispiel zeigt, wie ein Vektor, bestehend aus Zahlen, in einen solchen umgewandelt wird, dessen Einträge als Buchstaben angesehen werden:

```
> v <- c(1, 2, 3)
> mode(v)
[1] "numeric"
> v <- as.character(v)
> v
[1] "1" "2" "3"
> mode(v)
[1] "character"
```

## 2.3 Datenstrukturen

Mit Strukturen sind hier Gebilde gemeint, in denen mehrere Daten gemeinsam vorkommen, in aller Kürze sind das Vektoren, Matrizen, Dataframes, Arrays, Listen und Faktoren.

### 2.3.1 Vektoren

Vektoren werden durch Aneinanderhängen von Daten gebildet, der Befehl hierzu lautet `c()` («concatenate»). Vektoren enthalten nur Daten eines einzigen Typs, siehe dazu die folgenden vier Beispiele:

```
> v <- c(1, 2, 3)
> mode(v)
[1] "numeric"
> b <- c("ja", 3)
> mode(b)
[1] "character"
> l <- c(TRUE, FALSE)
> mode(l)
[1] "logical"
> nl <- c("TRUE", "FALSE")
> mode(nl)
[1] "character"
```

Man erkennt, dass in `b` offenbar auch die Ziffer als Buchstabe behandelt wird. Weiterhin erkennt man (siehe `l`), dass die Schlüsselwörter »TRUE« oder »FALSE« nur dann als logische Werte erkannt werden, wenn sie nicht in Anführungszeichen gesetzt sind (vergleiche mit `nl`).

### 2.3.2 Matrizen

Eine Matrix ist eine zweidimensionale Datenstruktur, die in Zeilen und Spalten organisiert ist, und die nur Daten eines einzigen Typs enthält. Eine Matrix wird mit dem Befehl `matrix` definiert, gefolgt von der Angabe der Einträge der Matrix (Daten), dann der Anzahl der Zeilen und der Spalten:



```
> m <- matrix(c(3, 6, 9, 12, 15, 18), 2, 3)
```

Wichtig ist, dass die Einträge der Matrix bei der Eingabe durch den Befehl `c()` zu einer Struktur verbunden werden. Beachten Sie aber, dass dies nicht die gewöhnliche Art ist, in der man in R zum Beispiel Fragebogendaten eingibt (siehe dazu Abschnitt 2.4) – es diente hier nur als ein kurzer und einfacher Weg, um eine zweidimensionale Struktur zu erzeugen. Wenn man anschließend `m` eingibt und mit Enter bestätigt, wird die Matrix ausgegeben.

```
> m
      [,1] [,2] [,3]
[1,]    3    9   15
[2,]    6   12   18
```

Man erkennt an dem Beispiel, dass die Matrix spaltenweise aufgefüllt wird (dieses Verhalten ist veränderbar, siehe dazu `help(matrix)`)

### 2.3.3 Dataframes

Dataframes sind den Matrizen verwandt, jedoch allgemeiner: Sie können Daten unterschiedlichen Typs enthalten. Diese Form der Datenstruktur wird in empirischen Erhebungen sehr oft benutzt, da hier zum Beispiel Angaben zum Geschlecht (»weiblich«, »männlich«) mit Zahlenwerten kombiniert werden können. Abschnitt 2.4 zeigt, wie man ein Dataframe bequem erzeugen oder bearbeiten kann.

### 2.3.4 Listen

Listen sind sehr allgemeine Strukturen und werden durch den Befehl `list()` definiert. Listen bestehen aus der Aneinanderreihung von beliebigen Objekten, die ihrerseits auch zum Beispiel wieder Listen sein können – oder auch etwas ganz anderes. Für Näheres hierzu sei auf die Hilfe verwiesen.

### 2.3.5 Arrays

Arrays sind Strukturen, die mehr als zwei Dimensionen haben können (aber nicht müssen). Der zweidimensionale Fall entspricht der Matrix (Abschn. 2.3.2), für Näheres sei auf `help(array)` verwiesen.

### 2.3.6 Faktoren

Faktoren sind besonders wichtig, wenn es um qualitative Daten geht, die Kategorien zugewiesen werden können. Ein Beispiel dafür wäre es, wenn in einer Erhebung das Geschlecht der Befragten erhoben würde, hier würde man von einem nominalskalierten Merkmal sprechen.

```
> gender <- c("maennlich", "maennlich", "weiblich", "maennlich",
              "weiblich", "weiblich", "weiblich")
> gender <- factor(gender)
> gender
[1] maennlich maennlich weiblich  maennlich weiblich
     weiblich weiblich
Levels: maennlich weiblich
```

Indem man auf den ursprünglichen Vektor `gender` die Funktion `factor()` anwendet, weist `R` dem Vektor intern natürliche Zahlen zu, und zwar beginnend mit 1, die dem alphabetisch ersten nominalen Merkmal zugewiesen wird. Hier wird also intern die Zuordnung »1=maennlich«, »2=weiblich« vorgenommen. Die Abfrage `gender`, bestätigt mit der Enter-Taste, zeigt, dass `R` zwei Merkmalsausprägungen erkannt hat (»levels«). Mit dem Befehl `summary()` können die Anzahlen ausgegeben werden, in denen die verschiedenen Merkmale auftauchen:

```
> summary(gender)
maennlich weiblich
          3         4
```

Wenn in qualitativen Merkmalsausprägungen zudem eine Ordnungsrelation gilt, spricht man von ordinalskalierten Merkmalsausprägungen. Ein Beispiel dafür wären die Schulnoten »1« bis »6«. Gehen wir davon aus, dass in einer Klasse Noten nach der üblichen Skala erteilt wurden, wobei sich die Verteilung `ergebnis` zeigt:

```
> noten <- c("1", "2", "3", "4", "5", "6")
> ergebnis <- c(1, 3, 2, 5, 4, 1, 2, 6, 1, 3, 4, 3, 2, 5, 2, 3, 5)
> klassenspiegel <- factor(ergebnis, labels=noten, levels=noten,
                           ordered=TRUE)

> summary(klassenspiegel)
klassenspiegel
1 2 3 4 5 6
3 4 4 2 3 1
> klassenspiegel
[1] 1 3 2 5 4 1 2 6 1 3 4 3 2 5 2 3 5
Levels: 1 < 2 < 3 < 4 < 5 < 6
```

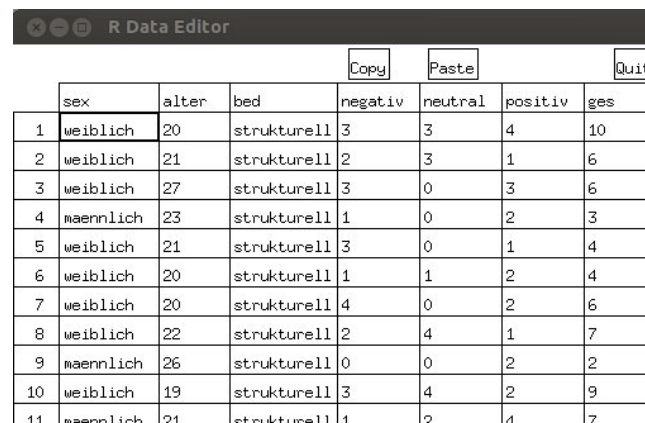
Wichtig ist in diesem Beispiel, dass die Noten, gegeben als Ziffern, nicht als rechnerische Größen missverstanden werden – es sind Namen, die ein Urteil symbolisieren. Damit `R` sie als dem Typus »character« zugehörig ansieht (vgl. Abschn. 2.2), wurden sie in Anführungszeichen gesetzt. Die allgemein übliche Mittelwertbestimmung auf der Basis solcher Daten ist höchst fragwürdig, zumal nicht sichergestellt ist, dass eine »2« doppelt so gut ist wie eine »4«. Der Befehl `summary(ergebnis)` würde eine solche, eigentlich unzulässige Rechnung durchführen und zum Beispiel ein arithmetisches Mittel ausgeben, weil die in `ergebnis` gespeicherten Daten nicht erkennen lassen, dass es sich um ein ordinalskaliertes Merkmal handelt. Erst die Zuweisungen mit `factor()` machen `R` die nötigen Informationen zugänglich.

## 2.4 Anzeigen oder Eingeben von Daten

Wenn Sie eine Anzeige in Tabellenform wünschen, in die Sie auch direkt etwas eingeben oder ändern können, dann können Sie im vorliegenden Fall den Befehl `edit(data.frame())` verwenden. Wenn Sie den Beispieldatensatz eingelesen haben (siehe Abschnitt 2.1), können Sie nun den Befehl testen:

```
> mydata <- edit(data.frame(mydata))
```

Auf dem Bildschirm erscheint eine Tabelle wie in Abbildung 2.1 gezeigt. Wenn Sie den Beispieldatensatz nicht wie beschrieben eingelesen haben, können Sie auch einfach das letzte runde Klammerpaar leer lassen – dann öffnet sich ein leeres Dataframe, in das Sie nun Einträge vornehmen können. Wenn hier



	sex	alter	bed	negativ	neutral	positiv	ges
1	weiblich	20	strukturell	3	3	4	10
2	weiblich	21	strukturell	2	3	1	6
3	weiblich	27	strukturell	3	0	3	6
4	maennlich	23	strukturell	1	0	2	3
5	weiblich	21	strukturell	3	0	1	4
6	weiblich	20	strukturell	1	1	2	4
7	weiblich	20	strukturell	4	0	2	6
8	weiblich	22	strukturell	2	4	1	7
9	maennlich	26	strukturell	0	0	2	2
10	weiblich	19	strukturell	3	4	2	9
11	maennlich	21	strukturell	1	2	4	7

**Abbildung 2.1:** Der Beispieldatensatz in tabellarischer Form. Hier können einzelne Zellen mit der Maus angewählt werden.

Änderungen vorgenommen werden, dann werden diese im Objekt `mydata` gesichert, denn durch die Befehlszeile oben wird vorgegeben, dass das, was beim Editieren des Datensatzes passiert, dem Objekt `mydata` zugewiesen wird. Gäbe man stattdessen den Befehl

```
> edit(data.frame(mydata))
```

ein, dann erschiene dasselbe Bild wie in [Abbildung 2.1](#) auf dem Bildschirm, jedoch würden Änderungen oder Ergänzungen im Datensatz nicht im Objekt `mydata` gespeichert. Sie gehen verloren, sobald der Editor für den Datensatz geschlossen wird!

## 2.5 Elementare Rechenoperationen

Für das Rechnen mit numerischen Daten stehen die üblichen Verknüpfungssymbole `+` `-` `*` `/` zur Verfügung. Um mehrere numerische Werte zu summieren, gibt es die Funktion `sum()`. Oft benötigt man eine Möglichkeit zum Runden, dies leistet der Befehl `round(sum(x), 2)`, in der hier gegebenen Form würden die Daten, die in `x` enthalten sind, summiert und das Ergebnis auf zwei Stellen nach dem Komma gerundet.

## 2.6 Daten adressieren

Datenstrukturen in *R* tragen grundsätzlich Indizes, über die einzelne Datenpunkte adressiert werden können. Die Indizes werden nicht angezeigt, sie ergeben sich daraus, an welcher Stelle ein Datenpunkt in der Datenstruktur platziert ist.

### 2.6.1 Einzelne Datenpunkte auswählen

Betrachten wir als einfaches Beispiel einen Vektor `a`, der mit dem folgenden Befehl gebildet wird:

```
> a <- c(9, 11, 23)
```

Die drei Zahlen, aus denen der Vektor besteht, tragen implizit je einen Index beginnend bei 1 und hier endend bei 3. Ein Datenpunkt in der Datenstruktur `a` kann mit Hilfe seines Indexes adressiert

werden: Die Abfrage, welcher Wert z. B. an der dritten Stelle der Datenstruktur steht, erfolgt, indem hinter dem `a` in eckigen Klammern der Index des abgefragten Datenpunktes angegeben wird. Nach dem Drücken der Enter-Taste erscheint das Ergebnis, dem eine Zeilennummer voran gestellt wird (ebenfalls in eckigen Klammern).

```
> a[3]  
[1] 23
```

In zweidimensionalen Strukturen werden die eckigen Klammern ebenfalls benutzt, dabei müssen aber zwei Argumente übergeben werden – eines für die Zeile und eines für die Spalte, an der sich der Datenpunkt befindet. Als Beispiel betrachten wir die Matrix aus Abschnitt 2.3.2. Wenn Sie diese Matrix wie dort beschrieben definieren, `m[2, 3]` eingeben und mit Enter bestätigen, wird der Wert ausgegeben, der in der zweiten Zeile und dritten Spalte steht.

### 2.6.2 Datenbereiche auswählen

In vielen Fällen möchte man nicht einen Wert, sondern ganze Bereiche einer Datenstruktur auswählen. Hierfür gibt es mehrere Möglichkeiten. Sei `a` wieder ein Vektor, für die folgenden Beispiele müsste dieser mindestens 6 Einträge haben:

- `a[c(2:6)]` wählt die Daten an zweiter bis sechster Stelle aus.
- `a[c(2, 6)]` wählt die Daten an zweiter und sechster Stelle aus,
- `a[c(-2, -6)]` wählt alle Daten außer derjenigen an zweiter und sechster Stelle aus, das ist gleichwertig zu `a[-c(2, 6)]`.
- `m[, 1]` (siehe Beispielmatrix in Abschn. 2.3.2) wählt alle Zeilen, aber nur die erste Spalte aus,
- `m[c(2:3), ]` wählt die Zeilen zwei bis drei aus, außerdem alle Spalten,
- `m[-2, ]` wählt die gesamte Matrix bis auf die zweite Zeile aus.
- `$`: In Datenstrukturen, deren Spalten Variablennamen tragen, können diese Spalten mit Hilfe des `$`-Zeichens ausgewählt werden. Im Beispieldatensatz (siehe Abschn. 2.1) kann die Spalte mit der Gesamtzahl der erinnerten Adjektive durch den Befehl `daten$ges` ausgewählt werden, weil die entsprechende Spalte den Namen `ges` trägt (siehe auch Abbildung 2.1 die Spalte am rechten Rand).

### 2.6.3 Daten filtern

Oft reichen die bisher beschriebenen Methoden nicht aus, um Daten auszuwählen. So kann es beispielsweise sein, dass man in einem sehr großen Datensatz Datenbereiche auswählen möchte, die eine bestimmte Eigenschaft haben. Man möchte dann den Datensatz nicht manuell durchsuchen und alle betroffenen Zeilen oder Spalten notieren, zumal ein solches Verfahren sehr fehleranfällig sein dürfte. Die Auswahl muss vielmehr automatisch nach bestimmten Kriterien erfolgen, die Daten sollen also automatisch gefiltert werden. Eine solche Filtermethode, die weiter unten verwendet werden wird, nutzt die Funktion `subset()`, die zum Beispiel Zeilen eines Datensatzes anhand eines vorgegebenen Merkmals auswählt (für ein Beispiel siehe Abschnitt 4.2.1 auf S. 22).

Die wichtigsten Methoden zum Filtern, Sortieren oder Gruppieren von Daten in *R* kann man sehr schön im freien Wiki-Book zu *R* nachlesen: [http://de.wikibooks.org/wiki/GNU\\_R#R\\_benutzen](http://de.wikibooks.org/wiki/GNU_R#R_benutzen) (siehe dort den Abschnitt *Umgang mit Datensätzen (Erstellen, Auswählen und Filtern)*).

## 2.7 Eingabe und Ausgabe von R

Wenn man die Ausgabe eines Befehls für die nächste Rechnung weiterverwenden möchte, schreibt man diese selbstverständlich nicht ab. Üblich ist es, der Ausgabe des einen Befehls ein neues Objekt zuzuordnen, mit dem in der folgenden Rechnung weiter gearbeitet wird. Eine zweite Möglichkeit ist es, die Befehle mit entsprechenden Klammerpaaren ineinander zu schachteln. So wurde in Abschnitt 2.6.1 dem Vektor  $c(9, 11, 23)$  ein neues Objekt  $a$  zugeordnet, mit dem weiter gearbeitet werden kann. Stattdessen könnte aber auch der ursprüngliche Vektor in folgende Rechnungen eingesetzt werden.

Wichtig ist, dass auch die Ausgabe komplexer Befehle einem Objekt zugordnet werden kann. In Abschnitt 5.2 wird zum Beispiel ein t-Test durchgeführt. Die dortige Ausgabe des t-Tests kann man einem Objekt zuweisen:

```
> x <- t.test(messw$Messung1, messw$Messung2, paired=TRUE)
```

Mit dem Befehl `names` kann man abfragen, unter welchen Namen die einzelnen Resultate des Testergebnisses intern gespeichert werden:

```
> names(x)
[1] "statistic"      "parameter"      "p.value"        "conf.int"
      "estimate"
[6] "null.value"     "alternative"     "method"         "data.name"
```

Damit ist es leicht möglich, die Ausgabe eines speziellen Wertes zu adressieren, zum Beispiel die ermittelte Wahrscheinlichkeit:

```
> x$p.value
[1] 0.3076503
```

Solche Werte könnten dann einem eigenen Objekt zugeordnet werden, sodass sie in weiteren Rechnungen leicht referenziert werden können, ohne dass man sie abschreiben müsste.



## 3 Ergänzungen zu Kapitel 1: Deskriptive Statistik

### 3.1 Grafiken erzeugen und speichern

R bietet einen großen Umfang an Möglichkeiten, Daten grafisch darzustellen. Um einen ersten Eindruck zu erhalten, können Sie

```
> demo(graphics)
```

eingeben. Im Folgenden sollen die in Kapitel 1.1 behandelten Varianten Histogramm und Kreisdiagramm behandelt werden.

#### 3.1.1 Eine erste Grafik: Ein Histogramm

Es soll die Häufigkeitsverteilung einer bestimmten Variable grafisch dargestellt werden. Wir wählen aus dem Beispieldatensatz die Variable `ges`. Der Befehl `hist()` bietet die Möglichkeit, ein solches Histogramm herzustellen. Dazu muss aber mitgeteilt werden, welche Variable des umfangreichen Datensatzes denn für die Darstellung ausgewählt werden soll. Im vorliegenden Datensatz hat die interessierende Variable den Namen `ges`. Wir können diese Variable mit dem Befehl

```
> hist(daten$ges)
```

adressieren, siehe dazu den Abschnitt 2.6.2. Vermutlich wird man das Bild, das mit dem eben angegebenen Befehl erzeugt wird, noch etwas verändern und verfeinern wollen. Dazu stellt der Befehl `hist` viele weitere Optionen zur Verfügung, die man mit `help(hist)` einsehen kann. Ein Bild wie in Abbildung 3.1 wird so erzeugt:

```
> hist(daten$ges, breaks=c(0:30), xlab="Gesamtzahl erinnelter  
Adjektive", ylab="Häufigkeit", main="", col=7)
```

Dabei bedeuten:

- `breaks` die Anzahl der Teilungen der  $x$ -Achse des Histogramms. Die Angabe `c(0:30)` erzeugt einen Vektor beginnend bei 0 und endend bei 30.
- `xlab` gibt die Beschriftung der Horizontalen Achse vor, `ylab` entsprechend,
- `main` gibt eine Hauptüberschrift vor, hier wurde sie leer gelassen,
- `col=7` gibt eine Farbe für die Balken vor, wobei 7 gelb entspricht. Um eine Liste der möglichen Farben zu erhalten, kann man den Befehl `colors()` verwenden. Anstatt der Zahl kann auch ein Farbname verwendet werden, der in der Liste auftaucht, die man mit dem eben gegebenen Befehl erhält, also zum Beispiel `col="springgreen"`.

Wenn man nicht am Histogramm in seiner grafischen Form interessiert ist, sondern an den dahinter liegenden Zahlenwerten, ergänzt man die Optionen im `hist`-Befehl einfach durch den mit einem Komma angeschlossenen Zusatz `plot=FALSE`

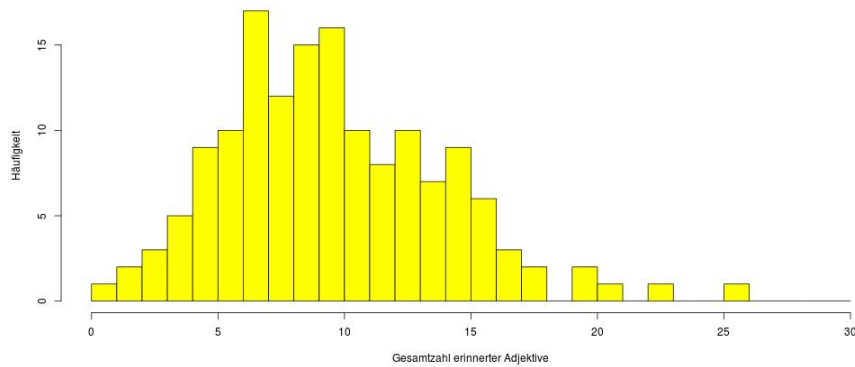


Abbildung 3.1: Ein Histogramm zum Beispieldatensatz.

### 3.1.2 Eine Grafik speichern

Das Speichern erfolgt in R vergleichsweise ungewohnt – es besteht aus drei Schritten, die mit den folgenden drei Befehlszeilen beispielhaft dargestellt sind:

```
> jpeg(file="histogramm.jpg",width=1000)
> hist(daten$ges,breaks=c(0:30),xlab="Gesamtzahl erinnelter
  Adjektive",
  ylab="Häufigkeit",main="",col=7)
> dev.off()
```

Mit der ersten Zeile wird vorgegeben, dass eine jpg-Datei mit einer Breite von 1000 Punkten erzeugt werden soll (für Details siehe `help(jpeg)`), die zweite erzeugt das Bild, und die dritte schaltet die Funktion, eine jpg-Datei zu erzeugen, wieder ab. Diese auf den ersten Blick umständliche Schrittfolge hat einen entscheidenden Vorteil: Grafiken, die auf dem Bildschirm erscheinen, können sukzessive weitere Elemente durch weitere Zeichenbefehle hinzugefügt werden, bevor mit `dev.off()` das Zeichnen des Diagramms beendet wird.

### 3.1.3 Ein Kreisdiagramm

Der Befehl zum Erzeugen eines solchen Diagramms in R lautet `pie()`. Als Aufgabe legen wir uns vor, den Anteil von Frauen und Männern im Beispieldatensatz in einem Kreisdiagramm darstellen zu wollen. Der Beispieldatensatz, wenn man ihn in der in Abschnitt 2.1 angedeuteten Weise eingelesen hat, enthält eine Spalte mit der Spaltenüberschrift `sex`, darunter Einträge wie `weiblich` oder `maennlich`. Naheliegend ist nun ein Versuch mit dem Befehl `pie(daten$sex)`. Es erscheint jedoch eine Fehlermeldung, die sagt, dass die darzustellenden Werte positiv sein müssen – das ist bei Buchstabenketten offensichtlich nicht der Fall. Es muss also zunächst dafür gesorgt werden, den beiden Geschlechtern je einen eindeutigen, positiven Wert zuzuordnen. Da es sich bei den Geschlechtern um nominale Daten handelt, machen wir diese Information R zunächst zugänglich (vgl. Abschn. 2.3.6 ab S. 9):

```
> gender <- factor(daten$sex)
> summary(gender)
maennlich weiblich
```



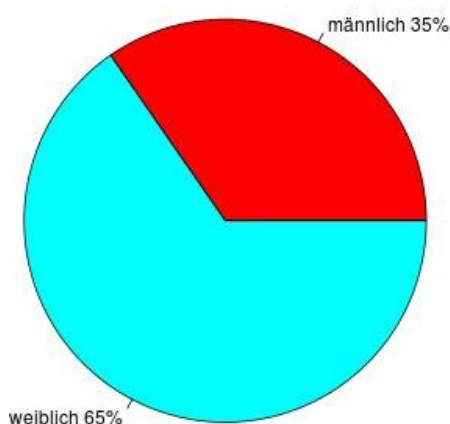
Um das eigentliche Kreisdiagramm zu erzeugen, dient der Befehl

```
> pie(summary(gender))
```

Auch hier gibt es viele zusätzliche Optionen, mit deren Hilfe man das Aussehen der Grafik verändern kann, sodass es das Aussehen wie in Abbildung 3.2 oder 3.3 erhält. Die Ansicht wie in Abbildung 3.2 entsteht wie folgt:

```
data <- summary(gender)
> percent <- c(round(data[1]/sum(data)*100),
               round(data[2]/sum(data)*100))
> names <- c("männlich", "weiblich")
> names <- paste(names, percent)
> names <- paste(names, "%", sep=" ")
> pie(data, labels=names, col=rainbow(length(names)))
```

Zunächst werden die Daten, die durch den Befehl `summary(gender)` erzeugt werden, unter dem neuen Objekt `data` gespeichert, um die nachfolgenden Zeilen übersichtlicher zu halten, denn dort kommt `data` mehrfach vor. Dann werden die prozentualen Anteile berechnet (für die Rechnungen vgl. Abschn. 2.5, für die Bedeutung der eckigen Klammern siehe Abschn. 2.6). Dann werden in drei Schritten die Beschriftungen erstellt bzw. ergänzt (`paste`), im dritten Schritt gefolgt von dem Hinweis, dass kein Leerzeichen zwischen `names` und dem %-Zeichen eingefügt werden soll (`sep=""`). Für die Auswahl der Farben wird eine Serie von Regenbogenfarben gewählt, wobei die Auswahl auf die Länge des Vektors `names` beschränkt wird (hier 2). Wer gern eine dreidimensionale Darstellungen einsetzen

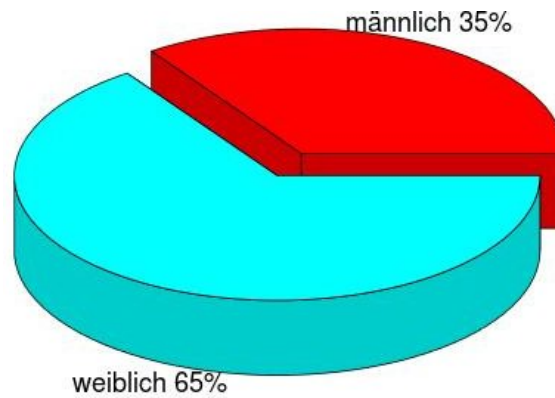


**Abbildung 3.2:** Ein Kreisdiagramm mit Beschriftungen.

möchte, installiert das Paket `plotrix` und aktiviert dann in `R` die entsprechende Bibliothek.

```
> library(plotrix)
> pie3D(data, labels=names, explode=0.1)
```

Abbildung 3.3 zeigt das Ergebnis, wobei für `names` die Befehle zu verwenden sind, die oben zur Erzeugung von Abbildung 3.2 aufgeführt wurden.



**Abbildung 3.3:** Ein dreidimensional gezeigtes Kreisdiagramm mit Beschriftungen. Der Parameter `explode` steuert, wie weit die Segmente räumlich von einander entfernt dargestellt werden.

### 3.2 Statistische Kennwerte

Für die statistischen Kennwerte steht in *R* eine Reihe von Befehlen zur Verfügung. Beispielhaft greifen wir wieder auf den Beispieldatensatz zu und betrachten die Gesamtzahl der erinnerten Adjektive (siehe Abschn. 2.1).

- **Mittelwert:** `mean()`

```
> mean(daten$ges)
[1] 10.07333
```

- **Median:** `median()`

```
> median(daten$ges)
[1] 10
```

- **Modus:** `which.max()`

```
> which.max(table(daten$ges))
7
```

Hier ist also zu beachten, dass der Befehl nicht auf dem Datenvektor direkt arbeitet, sondern auf seiner Häufigkeitsverteilung, die mit `table()` erzeugt wird.

- **Standardabweichung:** `sd()`

```
> sd(daten$ges)
[1] 4.367508
```

**Hinweis:** Der Befehl `sd()` berechnet die Standardabweichung mit Bezug auf die Stichprobenvarianz, also als  $\sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$ .

- **Varianz:** `var()`

```
> var(daten$ges)
[1] 19.07512
```

**Hinweis:** Der Befehl `var()` berechnet die Varianz in Bezug auf die Stichprobe, also mit  $\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}$ .

- **Spannweite:** `range()`

```
> range(daten$ges)
```

```
[1] 1 26
```

Einen Überblick über die Daten erhält man auch mit der Funktion `summary()`, die einige der eben erwähnten Kennwerte auflistet, dazu aber auch noch so genannte Quartile:

```
> summary(daten$ges)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.00   7.00   10.00   10.07   13.00   26.00
```

Die Eigenschaft des Medians ist, dass er die Häufigkeitsverteilung halbiert. Die Quartile (allgemeiner: Quantile, wenn die Häufigkeitsverteilung in andere als 25%-Anteile geteilt wird) lassen sich ähnlich interpretieren: Der Wert für das erste Quartil bei 7 sagt aus, dass 25% der Individuen zwischen 1 und einschließlich 7 Adjektive erinnert hat. Für das dritte Quartil lautet die Aussage, dass 75% der Individuen zwischen 1 und einschließlich 13 Adjektive erinnert hat.

### 3.3 z-Standardisierung

Um eine  $z$ -Standardisierung der Daten vornehmen zu können, muss zu jedem Datenpunkt  $x$  ein  $z$ -Wert berechnet werden, und zwar nach der Vorschrift  $z = \frac{x_i - \bar{x}}{\sigma(x)}$ . Diese Vorschrift lässt sich mit elementaren Rechenoperationen in R angeben, wir beziehen uns wieder auf das Beispiel der Anzahl erinnerter Adjektive (siehe Abschn. 2.1):

```
> z <- (daten$ges - mean(daten$ges)) / sd(daten$ges)
```

Durch diese Transformation wird jedem Probanden ein neuer Wert zugeordnet. Die sich daraus ergebende Verteilung hat die Streuung 1 und den Mittelwert 0. Verteilungen dieser Art sind untereinander eher vergleichbar als solche mit unterschiedlichen Mittelwerten und Streuungen. Die Verteilung an sich kann mit dem Befehl `table(z)` angezeigt werden, `hist(z)` liefert ein entsprechendes Diagramm (zu Details des Diagramms s. Abschn. 3.1.1). Einen Überblick über die Daten liefert auch der Befehl `summary()`:

```
> summary(z)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.07700 -0.70370 -0.01679  0.00000  0.67010  3.64700
```



## 4 Ergänzungen zu Kapitel 2: Inferenzstatistik

### 4.1 Konfidenzintervalle

Wir legen uns die Aufgabe vor, ein 95%-Konfidenzintervall für den Mittelwert der Gesamtzahl erinnerter Adjektive zu berechnen, wobei wir uns wieder auf den Beispieldatensatz beziehen (s. Abschn. 2.1). Die Berechnung soll zwei Werte für die Gesamtzahl erinnerter Adjektive liefern, die ein Intervall begrenzen. Innerhalb dieses Intervalls wird der empirisch ermittelte Mittelwert unserer Stichprobe liegen ( $\bar{x} = 10,07$ , s. Abschn. 3.2 ab S. 18). Das Intervall wird gerade so groß sein, dass wir sagen können, dass unsere Stichprobe aus einer Population stammt, deren Mittelwert mit einer Wahrscheinlichkeit von 95% ebenfalls aus diesem Intervall stammt. Die Berechnungsvorschrift lautet

$$\begin{aligned}\text{untere Grenze} &= \bar{x} - z \cdot \hat{\sigma}_{\bar{x}} \\ \text{obere Grenze} &= \bar{x} + z \cdot \hat{\sigma}_{\bar{x}}\end{aligned}$$

wobei für  $\hat{\sigma}_{\bar{x}}$  gilt:

$$\hat{\sigma}_{\bar{x}} = \frac{\hat{\sigma}_x}{\sqrt{n}}$$

$\hat{\sigma}_{\bar{x}}$  ist die Größe, die in R durch den Befehl `sd()` berechnet wird. Um die Rechnung durchzuführen, benötigen wir den Befehl `qnorm()`, der zu einem Wahrscheinlichkeitswert (hier  $1 - \alpha/2$ ) mit  $\alpha = 0,05$  den benötigten  $z$ -Wert ausgibt:

```
qnorm(1-0.05/2)
[1] 1.959964
```

Die vollständige Berechnung lautet dann für die untere bzw. obere Grenze:

```
> mean(daten$ges) - qnorm(1-0.05/2) * sd(daten$ges) /
  sqrt(length(daten$ges))
[1] 9.374399
> mean(daten$ges) + qnorm(1-0.05/2) * sd(daten$ges) /
  sqrt(length(daten$ges))
[1] 10.77227
```

Unsere Stichprobe stammt also mit einer Wahrscheinlichkeit von 95% aus einer Population, deren Mittelwert im Intervall [9,37; 10,78] liegt. Wenn der Stichprobenumfang kleiner als 30 ist, wird empfohlen, statt der  $z$ -Werte die zugehörigen  $t$ -Werte zu verwenden (mit passendem Freiheitsgrad). Für diesen Fall stellt R den Befehl `qt()` zur Verfügung, die Rechnung wäre dann gleichlautend bis auf den Ersatz von `qnorm()` durch in unserem Fall `qt(1-0.05/2, length(daten$ges)-1)`. Im Argument dieses Befehls taucht also an zweiter Stelle die Angabe des Freiheitsgrades  $150 - 1$  auf. Wenn Sie die Rechnung selbst durchführen, werden Sie feststellen, dass sich im vorliegenden Fall wegen  $n = 150$

ein Vertrauensintervall ergibt, dass bis auf zwei Stellen nach dem Komma identisch mit dem eben berechneten ist. In der Literatur wird daher oft grundsätzlich mit der Funktion `qt()` gearbeitet, da sie bei kleinen  $n$  geeigneter ist und bei großen  $n$  in die Werte von `qnorm()` übergeht.

## 4.2 Balkendiagramme mit Angabe von Konfidenzintervallen

R verfügt über umfangreiche Möglichkeiten, Grafiken zu erstellen. Gemeinsam ist ihnen, dass sie dem Nutzer sehr viele Möglichkeiten der Feineinstellung überlassen. Das ist praktisch, wenn man Grafiken für einen bestimmten Zweck erstellen möchte, es ist aber zunächst auch unpraktisch, weil man eine Reihe von Befehlen erlernen muss, um die Möglichkeiten auch tatsächlich nutzen zu können.

Wir legen uns die Aufgabe vor, ein Balkendiagramm zu erstellen, dass die Mittelwerte der erinnerten Adjektive getrennt für Frauen und Männer anzeigt, und das zusätzlich die Konfidenzintervalle zeigt.

### 4.2.1 Auswählen der benötigten Daten

Zunächst müssen aus dem Datensatz die Daten für Männer und Frauen ausgewählt werden. Damit dies nicht manuell geschehen muss, kann man zum Beispiel die Funktion `subset()` verwenden, die eine Auswahl nach einem bestimmten Merkmal ermöglicht – hier das Auftreten der Schlüsselwörter »maennlich« oder »weiblich« (s. Abschn. 2.6.3 auf S. 12). Anschließend wird gemittelt.

```
> frauen <- subset(daten, daten$sex=="weiblich")$ges
> maenner <- subset(daten, daten$sex=="maennlich")$ges
> m <- mean(maenner)
> f <- mean(frauen)
```

Die ersten beiden Zeilen sorgen dafür, dass der ursprüngliche Datensatz in zwei Teile zerfällt, und in jedem Teil wird nur die Spalte gewählt, die mit `ges` überschrieben ist. Beachten Sie, dass die Befehlskette, die mittels `subset()` eine Untermenge aus den Daten auswählen soll, die Daten zu "weiblich" und `maennlich` als Buchstabenkette erwartet, also von der Form `character`. Sollte dies nicht der Fall sein, so kann man dafür leicht mit einem zusätzlich eingeschobenen Befehl sorgen:

```
> frauen <- subset(daten, as.character(daten$sex)=="weiblich")$ges
```

### 4.2.2 Bestimmen der Zeichenkoordinaten für die Konfidenzintervalle

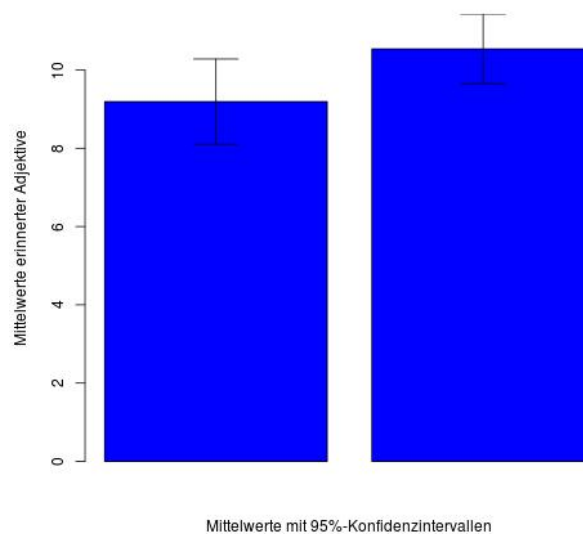
Die folgenden vier Zeilen bestimmen für Männer und Frauen jeweils die **untere** und die **obere** Grenze der 95%-Konfidenzintervalle:

```
> cim1 <- mean(maenner) - qnorm(1-0.05/2) * sd(maenner) /
  sqrt(length(as.vector(maenner)))
> cim0 <- mean(maenner) + qnorm(1-0.05/2) * sd(maenner) /
  sqrt(length(as.vector(maenner)))
> cif1 <- mean(frauen) - qnorm(1-0.05/2) * sd(frauen) /
  sqrt(length(as.vector(frauen)))
> cifo <- mean(frauen) + qnorm(1-0.05/2) * sd(frauen) /
  sqrt(length(as.vector(frauen)))
```

Im nächsten Schritt werden die  $x$ -Koordinaten der Balken im Balkendiagramm ermittelt und in `x` gespeichert, diese Information ist für das Zeichnen der Konfidenzintervalle nötig. Außerdem werden

die unteren bzw. oberen Intervallgrenzen je zu einem Vektor zusammengefasst (`ciu` bzw. `cio`), dies hat weiter unten Vorteile, wenn dort dann Vektoren als Argumente an die Zeichenbefehle übergeben werden und keine einzelnen Punkte, sodass die Eingabe etwas verkürzt wird. Ähnliches gilt für die Zusammenfassung von `m` und `f` zu `d`. Für die Zeichnung ist außerdem wichtig, wie weit die  $y$ -Achse nach oben ragen muss, dafür wird das Maximum der beiden oberen Intervallgrenzen bestimmt.

```
> x <- barplot(c(m, f))
> ciu <- c(cimu, cifu)
> cio <- c(cimo, cifo)
> d <- c(m, f)
> max <- max(cio)
```



**Abbildung 4.1:** Ein Balkendiagramm zum Vergleich der Gesamtzahl erinnelter Adjektive für Männer (links) und Frauen.

#### 4.2.3 Zeichnen des Diagramms

Nun kann mit der eigentlichen Zeichnung begonnen werden. Dazu wird zunächst die Funktion `jpeg` eingeschaltet (s. Abschn. 3.1.2) und dann das Balkendiagramm gezeichnet:

```
barplot(d, ylab="Mittelwerte erinnelter Adjektive", col="blue",
        xlab="Mittelwerte mit 95%-Konfidenzintervallen", ylim=c(0, max))
```

Die Konfidenzintervalle sind noch nicht zu sehen, diese werden nun schrittweise eingefügt: Die folgenden Zeilen führen das Zeichnen durch, dabei wird der Befehl `segments()` verwendet, der eine Linie von einem Punkt zu einem anderen zeichnet. Der erste Punkt ergibt sich aus den beiden ersten Argumenten ( $x_1$ -Koordinate,  $y_1$ -Koordinate), der zweite aus dem dritten und vierten Argument des Befehls ( $x_2$ -Koordinate,  $y_2$ -Koordinate). Hier erweist sich nun die Bestimmung der  $x$ -Werte der Balken(mitten) als wichtig, da die Konfidenzintervalle üblicherweise horizontal in der Mitte eines Balkens eingetragen werden.

```
> segments(x, d, x, cio)
> segments(x, d, x, ciu)
> segments(x-0.1, cio, x+0.1, cio)
> segments(x-0.1, ciu, x+0.1, ciu)
```

Die ersten beiden Zeilen lassen die vertikalen Teile der Intervalle zeichnen, die letzten beiden Zeilen lassen die Intervallgrenzen einzeichnen (»whiskers«). Wenn das Diagramm vollständig ist (es vervollständigt sich am Bildschirm schrittweise), wird die Zeichenfunktion mit `dev.off()` abgeschaltet – dabei wird im Hintergrund die letzte Version der Grafik unter dem Namen gespeichert, der an den `jpeg`-Befehl übergeben worden ist. Abbildung 4.1 zeigt das Ergebnis. Man erkennt, dass die Frauen deskriptiv mehr Adjektive erinnern haben als die Männer. Für den Rückschluss auf die Population gibt die Grafik aber den Hinweis, dass dies dort nicht gelten muss – die beiden Konfidenzintervalle für Frauen und Männer überlappen, das heißt, dass es eine gewisse Wahrscheinlichkeit dafür gibt, dass die Mittelwerte von Frauen und Männern in der Population übereinstimmen. Die Frage, wie solche Unterschiede zu bewerten sind, ist Thema des 3. Kapitels, das sich mit dem t-Test befasst.

### 4.3 Balkendiagramme mit Angabe von Fehlerbalken

Mit der in Abschnitt 4.2 beschriebenen Methode lassen sich Balkendiagramme sehr fein justieren. Oft werden in solchen Diagrammen nicht die Konfidenzintervalle der Mittelwerte angegeben, sondern die Standardfehler der Mittelwerte. Der Standardfehler des Mittelwertes wird durch

$$\hat{\sigma}_{\bar{x}} = \frac{\hat{\sigma}_x}{\sqrt{n}} = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n \cdot (n - 1)}}$$

geschätzt. Da  $R$  bei der Berechnung der Streuung und Standardabweichung die in Abschnitt 3.2 auf S. 18 angegebenen Rechenvorschriften verwendet, kann der Standardfehler sehr einfach in  $R$  berechnet werden.

#### 4.3.1 Definieren einer eigenen Funktion zum Zeichnen

Um die vielen Einzelbefehle wie im Beispiel oben nicht immer wieder eintippen zu müssen, kann man sich auch eine Funktion definieren, in der die Einzelbefehle gespeichert sind. Für das Zeichnen des Balkendiagramms mit Standardfehlern der Mittelwerte soll nun eine solche Funktion unter dem Namen `barplotse` definiert werden.

```
barplotse <- function(d)
{
  m <- colMeans(d, na.rm = T)
  se <- function(p) sqrt(var(p, na.rm = T) /
                           length(as.vector(p)))
  se <- apply(d, 2, se)
  x <- barplot(m)
  tu <- m + se
  tl <- m - se
  max <- max(tu)
  barplot(m, ylab="d", col="blue",
```



```

xlab="Mittelwerte mit Standardfehlern",ylim=c(0,max))
segments(x,m,x,tu)
segments(x,m,x,tl)
segments(x-0.1,tu,x+0.1,tu)
segments(x-0.1,tl,x+0.1,tl)
}

```

Dieser Text wird als `R.barplotse` gespeichert (im Sinne einer eigenen Datei). Die Definition der Funktion `barplotse()` enthält zum größten Teil das, was bereits aus Abschnitt 4.2 bekannt ist. Einiges ist aber neu und soll daher hier erläutert werden:

- In den Berechnungsvorschriften taucht `na.rm = T` auf. Damit wird dem Programm mitgeteilt, wie mit fehlenden Werten umgegangen werden soll – hier sollen sie entfernt werden (»na« steht für »not available«, »rm« für »remove« und »T« für »TRUE«). Fehlende Werte sind der Normalfall in empirisch gewonnen Datensätzen, daher muss fast immer eine Strategie vorgegeben werden, wie damit umgegangen werden soll. Sehr oft scheitern Berechnungen daran, dass man vergessen hat, eine entsprechende Strategie vorzugeben, sodass R keine Berechnung durchführen kann.
- Es taucht die Funktion `apply(d, 2, se)` auf. Diese bedeutet hier, dass die Funktion `se` (Definition direkt darüber stehend) auf `d` angewandt werden soll, und zwar spaltenweise, was durch die 2 vorgegeben wird. Eine 1 an dieser Stelle würde eine zeilenweise Anwendung erzwingen.

Nachdem die Datei `R.barplotse` mit dem obigen Inhalt gespeichert wurde, kann die durch den Inhalt der Datei definierte Funktion `barplotse()` in R bekannt gemacht werden:

```
source(file="R.barplotse")
```

Da die neue Funktion `barplotse()` einen rein numerischen Datensatz erwartet, wählen wir die entsprechenden Spalten aus dem Beispieldatensatz (s. Abschn. 2.1) aus und fassen sie zu einer Matrix `mydata` zusammen. Der Befehl `cbind()` verbindet Vektoren spaltenweise (die Entsprechung für das zeilenweise Verbinden wäre `rbind()`):

```
> mydata <- cbind(daten$negativ, daten$neutral,
                  daten$positiv, daten$ges)
```

Die neue Funktion kann nun auf diesen Datensatz angewandt werden:

```
> barplotse(mydata)
```

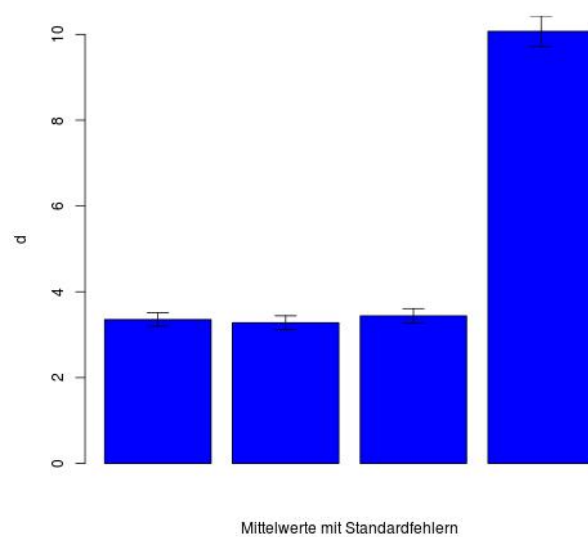
Das Ergebnis ist in Abbildung 4.2 zu sehen.

#### 4.3.2 Weitere effizientere Möglichkeiten zur Erzeugung von Balkendiagrammen

Oben wurde gezeigt, wie man das Zustandekommen eines Diagramms im Detail steuern kann, und wie man sich hierzu gegebenenfalls auch eine eigene Funktion zum Zeichnen definieren kann. Es gibt allerdings auch fertige Bibliotheken, die Befehle enthalten, die solche Dienste übernehmen können, eine davon ist `psych`. Mit den Befehlen (verwende wieder den Beispieldatensatz aus Abschn. 2.1)

```
> library(psych)
> error.bars(cbind(daten$positiv, daten$negativ), bars=TRUE,
             ylim=c(0, 4))
```

erhalten Sie bereits ein Diagramm, das den oben beschriebenen Varianten sehr ähnelt. Für Informationen über weitere Einstellmöglichkeiten siehe `help(error.bars)` (nachdem Sie die Bibliothek geladen haben) oder, ausführlicher, `help(package="psych")`.



**Abbildung 4.2:** Das Balkendiagramm zu der in Abschnitt [4.3.1](#) definierten Funktion.

## 5 Ergänzungen zu Kapitel 3: Der t-Test

### 5.1 Durchführung eines t-Tests für unabhängige Stichproben

Wir beziehen uns wieder auf den Beispieldatensatz (Abschn. 2.1) und legen uns die Frage vor, ob der Unterschied zwischen den Erinnerungsleistungen unter der Bedingung »strukturell« oder »bildhaft« eher ein zufälliges Ereignis ist, oder ob er zu der Annahme berechtigt, dass es sich hier um einen systematischen Unterschied handelt. Im ersten Fall würde man folgern, dass sich das Ergebnis bei einer Wiederholung des Versuchs sehr wahrscheinlich anders darstellen wird, womöglich könnte der Unterschied sogar in umgekehrter Richtung auftreten. Im zweiten Fall hingegen würde man folgern, dass man einen Unterschied in gleicher Richtung mit sehr hoher Wahrscheinlichkeit wieder beobachten würde.

Um den t-Test für die beiden unabhängigen Stichproben durchführen zu können, müssen aus dem Datensatz zunächst die relevanten Teile ausgewählt werden, das sind einmal alle Versuchspersonen, die unter der Bedingung »strukturell« am Experiment teilgenommen haben, und dann alle Personen, für die die Bedingung »bildhaft« lautete. Hierzu verwenden wir wieder den Befehl `subset()`. Der t-Test wird dann in der Weise durchgeführt, wie es die folgenden Zeilen zeigen. Dabei wird die Option "greater" gesetzt. Das bedeutet, dass hier ein einseitiger Test mit der Alternativhypothese durchgeführt wird, dass es einen von Null verschiedenen positiven Unterschied zwischen den beiden Mittelwerten der Stichproben gibt ("less" und "two.sided" wären die anderen möglichen Einstellungen).

```
> strk <- subset(daten, daten$bed=="strukturell")$ges
> bild <- subset(daten, daten$bed=="bildhaft")$ges
> t.test(bild, strk, alternative = "greater")
```

Die Ausgabe des Programms lautet dann wie folgt:

```
Welch Two Sample t-test

data:  bild and strk
t = 5.1575, df = 91.653, p-value = 7.186e-07
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 2.575716      Inf
sample estimates:
mean of x mean of y
 11.0      7.2
```

Der Welch-Test für zwei unabhängige Stichproben ist ein t-Test, bei dem das Programm davon ausgeht, dass die Varianzen der beiden Stichproben inhomogen sind, also eine Annahme, die die Verwendung des klassischen t-Tests nicht zulässt. Daraus ergeben sich nun zwei Fragen, und zwar erstens, wie ein klassischer t-Test (wie im Buch beschrieben) durchzuführen wäre, und zweitens, wie die in Frage

stehende Varianzhomogenität geprüft werden kann. Anders ausgedrückt muss man sich fragen, ob die Realisierung des t-Tests durch einen Welch-Test, die R ausgewählt hat, das angemessene Verfahren ist, und wenn nicht, wie ein angemessenes Verfahren für die Realisierung eines t-Tests technisch durchzuführen wäre. Zur Frage, wie ein t-Test in der im Buch beschriebenen Variante durchzuführen wäre, gibt es eine kurze Antwort: Für den t-Test bei *gleichen Stichprobenvarianzen* wird der Befehl für den t-Test in R um einen Parameter ergänzt, der genau dies mitteilt, und zwar `var.equal=TRUE`. Für den vorliegenden Fall sieht der Test dann folgendermaßen aus:

```
> t.test(bild, strk, alternative = "greater", var.equal=TRUE)

      Two Sample t-test

data:  bild and strk
t = 5.1575, df = 98, p-value = 6.523e-07
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 2.576525      Inf
sample estimates:
mean of x mean of y
   11.0      7.2
```

In dieser Variante nimmt die Anzahl der Freiheitsgrade  $df$  auch wieder den Wert an, den wir nach den Ausführungen im Buch erwartet hätten:  $100 - 2 = 98$ . Während die  $t$ -Werte der beiden Testvarianten gleich sind, führt der Welch-Test offenbar eine Korrektur der Freiheitsgrade nach unten aus (vergleiche dazu mit dem Beispiel oben, das  $df = 91.653$  enthält). Für die Beurteilung der Signifikanz des Ergebnisses hat dies im vorliegenden Fall jedoch keinen Einfluss. Ebenso hat die Frage, welche der beiden Testvarianten durchgeführt wird, für das vorliegende Beispiel keine Bedeutung für das Konfidenzintervall, das für den Mittelwertunterschied von 3,8 der beiden Stichproben als geschlossenes rechtsseitig unendliches Intervall  $[2, 58; \infty)$  angegeben wird. Für den t-Test ist ein Konfidenzniveau von 0,95 voreingestellt. Möchte man hier einen anderen Wert vorgeben, dann ergänzt man die Angaben im Argument um den Eintrag `conf.level=<Wert>`, also zum Beispiel `conf.level=0.99`. Damit ist geklärt, wie die beiden Varianten des t-Tests aussehen, um die es hier geht. Offen ist die Frage, welche der beiden Varianten hier die angemessene ist – auch dann, wenn wir bereits wissen, dass dies für die Interpretation der Ergebnisse im vorliegenden Fall unbedeutend ist, da die relevanten Werte je gleich sind. Um diese Frage zu entscheiden, muss ein Test auf Varianzhomogenität durchgeführt werden.

### 5.1.1 Test auf Varianzhomogenität

Wird die Varianzhomogenität in Zweifel gezogen, kann man mit dem Levene-Test eine Prüfung durchführen. Dazu muss die Bibliothek `car` geladen werden:

```
> library(car)
```

Der Levene-Test erwartet einen Datenvektor und einen Vektor, nach dem diese Daten gruppiert werden sollen. Diese beiden Vektoren verschaffen wir uns zunächst, indem die Daten zu einem Vektor `mydata` verbunden werden, außerdem wird für den Zweck der Gruppierung ein neuer Vektor `group` gebildet, der genau dort 1 enthält, wo `mydata` Daten unter der Bedingung »strukturell« enthält, und 2, wo `mydata` sich auf die Bedingung »bildhaft« bezieht. Da die Daten in `mydata` bereits sortiert sind, will sagen, dass an den ersten 50 Plätzen die Daten unter der Bedingung »strukturell« liegen und beginnend

mit Platz 51 bis Platz 100 die Daten zur Bedingung »bildhaft« liegen, ist der Gruppierungsvektor sehr einfach unter Verwendung des Befehls `rep()` zu bilden. `rep(1, length(strk))` bedeutet, dass eine 1 genau so oft wiederholt wird, wie der Vektor `strk` lang ist. Im Ganzen hat die Befehlskette dann folgende Gestalt:

```
> mydata <- c(strk, bild)
> group <- as.factor(c(rep(1, length(strk)), rep(2, length(bild))))
```

Der klassische Levene-Test bezieht sich auf die Mittelwerte der gruppierten Daten, daher geben wir `mean` im Argument des Befehls mit an:

```
> leveneTest(mydata, group, "mean")
Levene's Test for Homogeneity of Variance (center = "mean")
      Df F value  Pr(>F)
group  1  3.7638 0.05525 .
      98
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Der Test nutzt für die Prüfung die F-Verteilung, die uns an anderer Stelle noch beschäftigen wird. Wesentlich ist der Wert 0.05525, der über die Signifikanz des Testergebnisses entscheidet. Hier liegt ein marginal signifikantes Ergebnis vor, es ist also davon auszugehen, dass die Varianzen in der Tendenz inhomogen sind. Der Welch-Test, der als erste Variante hier vorgestellt wurde, ist daher das angemessene Verfahren zur Durchführung eines t-Tests mit den vorliegenden Daten. Hätte der Levene-Test ein nicht signifikantes Ergebnis erbracht, dann wäre die Argumentation mit der Variante des t-Tests zu führen, in der der Parameter `var.equal=TRUE` gesetzt ist.

## 5.2 Durchführung eines t-Tests für abhängige Stichproben

Häufig tritt bei statistischen Erhebungen eine Situation auf, in der eine einzige Stichprobe mehrfach für eine Messung heran gezogen wird, etwa dadurch, dass ein und dieselbe Personengruppe mehrfach befragt wird. Es liegt dann eine so genannte Messwiederholung vor. Für diese Situation ist der t-Test in einer eigenen Variante durchzuführen, in der die Tatsache berücksichtigt wird, dass die bei den unterschiedlichen Messungen erhobenen Daten nicht statistisch unabhängig sind. Eine Person, die bei einer Befragung ein bestimmtes Antwortverhalten zeigt, wird in einer Wiederholung derselben Befragung möglicherweise ein ähnliches Antwortverhalten zeigen – die Daten sind *korreliert*. Eine solche Korrelation ist nicht auf die Wiederholung derselben Befragung beschränkt – auch bei mehreren Befragungen unterschiedlichen Inhalts, aber derselben Personen, kann es sein, dass die gewonnenen Daten nicht unabhängig von einander sind, zum Beispiel dann, wenn eine Personeneigenschaft das Antwortverhalten grundsätzlich beeinflusst.

In Kapitel 3.5 ist eine Aufgabe beschrieben, in der Probanden innerhalb einer kurzen Zeitspanne möglichst oft eine bestimmte Sequenz zu tippen hatten. Als abhängige Variable wurde die Anzahl der korrekten Sequenzen erhoben. Diese Aufgabe hatten alle Versuchspersonen zweimal zu bewältigen, es liegt also die Situation einer Messwiederholung vor. Ein t-Test für abhängige Stichproben kann hier Aufschluss über die Frage geben, ob sich die Ergebnisse bei der zweiten Durchführung eher verbessern, etwa bedingt durch einen Übungseffekt, oder ob sie sich eher verschlechtern, zum Beispiel bedingt durch einen Motivationsverlust.

Um das Beispiel zu betrachten, wird der Datensatz zur Messwiederholung von der Internetseite zum

Buch kopiert und unter einem Namen gespeichert, der keine Grundstriche enthält. Als nächstes wird die Bibliothek `foreign` aktiviert, damit der SPSS-Datensatz eingelesen werden kann.

```
> library(foreign)
> messw <- read.spss(file="Messwiederholung.sav",
                     to.data.frame=TRUE)
```

Zurückkodierung von CP1252

Um sich einen Überblick über die Daten zu verschaffen, kann man einfach `messw` eingeben, wesentlich sind aber eigentlich nur die Spaltenüberschriften, die mit dem folgenden Befehl angezeigt werden.

```
> names(messw)
[1] "vp"           "Geschlecht"  "Messung1"
      "Messung2"  "Messung3"
[6] "Diff_M1_M2"  "Diff_M2_M3"  "Diff_M1_M3"
      "Medikament1" "Medikament2"
[11] "Medikament3" "bed"         "av"
      "Frage1"     "Frage2"
[16] "Frage3"
```

Um den Test durchführen zu können, werden die abhängigen Stichproben ausgewählt (Messungen 1 und 2), außerdem wird die Option `paired=TRUE` aktiviert – damit wird R mitgeteilt, dass ein Test für abhängige Stichproben durchzuführen ist:

```
> t.test(messw$Messung1, messw$Messung2, paired=TRUE)
```

Paired t-test

data: messw\$Messung1 and messw\$Messung2

t = -1.0353, df = 35, p-value = 0.3077

alternative hypothesis: true difference in means is not equal to 0  
95 percent confidence interval:

-2.1384816 0.6940372

sample estimates:

mean of the differences

-0.7222222

Das Ergebnis zeigt, dass die gefundenen Daten unter der Annahme der Nullhypothese relativ wahrscheinlich sind ( $p = 0,31$ ), die gefundene Mittelwertsdifferenz ist also nicht signifikant. Das angegebene Konfidenzintervall für die Mittelwertdifferenz fügt sich plausibel an dieses Ergebnis, denn es schließt die Null ein. Negative wie positive Werte müssen also beim Rückschluss auf die Population erwartet werden. Wichtig ist noch der Hinweis, dass im vorliegenden Fall keine Vorgabe gemacht wurde, ob ein ein- oder zweiseitiger Test durchzuführen wäre. Da es keine begründete gerichtete Hypothese gab, wird hier zweiseitig getestet, was die Standardeinstellung für den t-Test ist, siehe dazu `help(t.test)`. Mit der Option `alternative = "two.sided"` im Argument des Befehls `t.test` ergäbe sich daher ein identisches Ergebnis.

Schließlich sei noch ein Hinweis zu den Freiheitsgraden gegeben: Da der t-Test für abhängige Stichproben mit der Mittelwertdifferenz arbeitet, gehen nur  $n$  Daten in die Rechnung ein. Bei der theoretischen Berechnung denkbarer Mittelwertunterschiede sind stets  $n - 1$  Mittelwertunterschiede frei wählbar, und einer ist durch die  $n - 1$  frei gewählten Unterschiede determiniert. Da 36 Personen teilgenommen

haben, ergibt sich  $df = 35$ , und nicht  $36 - 2 = 34$ , wie es sich im Falle zweier unabhängiger Stichproben ergeben hätte.

### 5.3 Durchführung eines t-Tests für eine Stichprobe

Bei dieser Variante des t-Tests wird der Mittelwert einer Stichprobe (oder auch die Mittelwertdifferenz zweier Stichproben) mit einem bestimmten, vorzugebenden Testwert verglichen. In Bezug auf den Beispieldatensatz (s. Abschn. 2.1, S. 7) und die damit einher gegangene Untersuchung könnte eine Frage etwa lauten, ob die Versuchspersonen im Schnitt signifikant mehr als 9 Adjektive erinnert haben. Für den Test liegen unter dieser Fragestellungen die beiden Hypothesen  $H_0: \bar{x} \leq \mu$  und  $H_1: \bar{x} > \mu$  zugrunde, das bedeutet, es wird ein einseitiger t-Test durchgeführt:

```
> t.test(daten$ges, alternative="greater", mu=9)

      One Sample t-test

data:  daten$ges
t = 3.0099, df = 149, p-value = 0.001535
alternative hypothesis: true mean is greater than 9
95 percent confidence interval:
 9.4831      Inf
sample estimates:
mean of x
10.07333
```

Das Resultat ist so zu lesen, dass die empirisch gefundenen Daten unter der Annahme der Nullhypothese sehr unwahrscheinlich sind  $p < 0,01$ . Man wird die Nullhypothese daher verwerfen, der Unterschied zwischen dem beobachteten Mittelwert  $\bar{x} = 10,07$  und dem für eine Population angenommenen von  $\mu = 9$  ist hoch signifikant.

Würde die Frage anders formuliert, und zwar derart, dass es darum gehe, ob die Versuchspersonen im Schnitt eine signifikant andere Anzahl von Adjektiven erinnerten, dann müsste zweiseitig getestet werden, der Befehl zur Durchführung und die Ausgabe lauteten dann wie folgt:

```
> t.test(daten$ges, mu=9)

      One Sample t-test

data:  daten$ges
t = 3.0099, df = 149, p-value = 0.00307
alternative hypothesis: true mean is not equal to 9
95 percent confidence interval:
 9.368676 10.777991
sample estimates:
mean of x
10.07333
```

Auch hier ist das Ergebnis hoch signifikant. Während sich oben wegen der einseitigen Testung ein abgeschlossenes, rechtsseitig unendliches Konfidenzintervall ergab, ist dieses Intervall nun wegen der zweiseitigen Testung abgeschlossen (die zweiseitige Testung ist die Standardeinstellung, daher muss

die Option `alternative="two.sided"` nicht angegeben werden). Dieses Intervall schließt den angenommenen Wert von  $\mu = 9$  nicht ein, was in Zusammenhang mit dem signifikanten Ergebnis plausibel ist.

## 5.4 Vertiefung: Vergleich t-Test für unabhängige und abhängige Stichproben in R

Für diesen Abschnitt verwenden wir wieder den Datensatz zur Messwertwiederholung, siehe Abschnitt 5.2 auf S. 29. Für die im vorliegenden Abschnitt ausgeführten Überlegungen ist ein grundlegendes Verständnis des Konzepts des Zusammenhangs bzw. der Korrelation hilfreich (siehe Kap. 4 des Buches). In Abschnitt 5.2 wurde der Grundgedanke bereits skizziert, was unter abhängigen Stichproben zu verstehen ist. Im vorliegenden Abschnitt soll es nun um die Frage gehen, wie die Ergebnisse eines t-Tests für abhängige Stichproben und solche für unabhängige Stichproben zusammenhängen, wenn in Wahrheit gar keine Korrelation, also keine Abhängigkeit der Daten untereinander vorliegt.

In dem Datensatz, mit dem im vorliegenden Fall gearbeitet werden soll, finden sich die beiden Variablen `Frage1` und `Frage2`. Die Daten zu diesen beiden Fragen sind künstlich so gestaltet, dass *keine Korrelation* vorliegt, es handelt sich also um unabhängige Daten. Ihr Zustandekommen kann man sich so vorstellen, dass Frage 1 und Frage 2 gar nicht denselben Personen gestellt wurden, sondern verschiedenen Personengruppen, und dass die Ergebnisse anschließend durch die Art der Eingabe in das Dataframe so angeordnet worden seien, dass sie vermeintlich gepaart sind. Wenn Sie sich den Datensatz `messw` (sofern Sie ihn diesem Objektamen zugeordnet haben) ansehen, erkennen Sie, dass die Daten zu Frage 2 identisch sind mit den Daten, die Sie in der Spalte `av` unter der Bedingung `bed 2` finden. Sie stammen von den Probanden 37-72 und wurden in die Spalte `Frage2` zur Bedingung `bed 1` kopiert.

Im ersten Schritt wollen wir uns davon überzeugen, dass dies tatsächlich der Fall ist, im zweiten Schritt untersuchen wir die Frage, welche Resultate der t-Test für abhängige oder unabhängige Stichproben auf der Basis dieser Daten liefert.

### 5.4.1 Bestimmung der Korrelation

Zunächst wählen wir aus dem Datensatz diejenigen Zeilen (Probanden) aus, die Antworten auf `Frage1` und `Frage2` enthalten, das sind genau die Zeilen, die unter der Variablen `bed` den Eintrag 1 haben:

```
> mydata <- subset(messw, messw$bed==1)
```

Die Frage nach einem denkbaren (linearen) Zusammenhang zwischen den Daten in `Frage1` und `Frage2` kann mit der Bestimmung des so genannten Korrelationskoeffizienten  $r_{Frage1,Frage2}$  beantwortet werden. Seine Werte liegen im Intervall  $[-1, 1]$ . Ein Wert von  $r_{Frage1,Frage2} = 1$  gibt an, dass zwischen den betrachteten Daten ein perfekt positiv-linearer Zusammenhang besteht, für  $r_{Frage1,Frage2} = -1$  gilt die Aussage entsprechend für einen negativen Zusammenhang. Die Berechnung in R erfolgt mit dem Befehl `cor`:

```
> cor(mydata$Frage1, mydata$Frage2)
[1] 1.140134e-15
```

Der ermittelte Wert liegt nahe Null, wenn man auf zwei Stellen nach dem Komma rundete, ergäbe sich

```
> round(cor(mydata$Frage1, mydata$Frage2), 2)
[1] 0
```



Es ist also kein linearer Zusammenhang in den Daten erkennbar. Wäre ein solcher erkennbar, so stellte sich beim Rückschluss auf die Populationsdaten wieder die Frage, ob die gefundenen Zusammenhänge signifikant sind. Es kann durchaus sein, dass man in einer Stichprobe Zusammenhänge in Form von Korrelationskoeffizienten meint nachweisen zu können, die sich aber in der Population nicht finden lassen. Aus diesem Grund ist im Allgemeinen bei der Bestimmung von Korrelationen auch ein Signifikanztest nötig, der hier entfällt, da ohnehin kein Zusammenhang erkennbar wird (in Abschn. 6.2.1 ab S. 37 wird jedoch erläutert, wie man dazu vorgehen müsste).

Wir wenden uns nun der zweiten Frage zu, in der es darum geht, die Ergebnisse eines t-Tests für ungepaarte mit einem solchen für gepaarte (abhängige) Stichproben zu vergleichen. Vorbereitend überzeugen wir uns zunächst von der Varianzhomogenität (s. Abschn. 5.1.1 ab S. 28):

```
> group <- as.factor(c(rep(1,length(mydata$Frage1)),
                        rep(2,length(mydata$Frage2))))
> leveneTest(c(mydata$Frage1,mydata$Frage2),group,"mean")
Levene's Test for Homogeneity of Variance (center = "mean")
      Df F value Pr(>F)
group  1  1.2835 0.2611
      70
```

Es zeigt sich, dass unter der Annahme der Nullhypothese (gleiche Varianzen) die Wahrscheinlichkeit für die tatsächlich vorliegenden Varianzen recht groß ist ( $p = 0,26$ ). Wir gehen daher davon aus, dass die Varianzen homogen sind und verwenden bei den folgenden Tests daher die Option `var.equal=T`. Der Test für unabhängige Stichproben liefert:

```
> t.test(mydata$Frage1,mydata$Frage2,var.equal=T)

      Two Sample t-test

data:  mydata$Frage1 and mydata$Frage2
t = -0.6885, df = 70, p-value = 0.4934
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.815302  1.370413
sample estimates:
mean of x mean of y
 16.55556  17.27800
```

Mit der zusätzlichen Option `paired=T` führen wir den Test für abhängige Stichproben aus und erhalten:

```
> t.test(mydata$Frage1,mydata$Frage2,paired=T,var.equal=T)

      Paired t-test

data:  mydata$Frage1 and mydata$Frage2
t = -0.6885, df = 35, p-value = 0.4957
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.852733  1.407845
sample estimates:
```

```
mean of the differences  
-0.7224444
```

Es zeigt sich, dass der Test im Falle von unkorrelierten Daten gleiche Ergebnisse liefert ungeachtet der Frage, ob er als Test für gepaarte (abhängige) oder ungepaarte (unabhängige) Stichproben durchgeführt wird.

## 6 Ergänzungen zu Kapitel 4: Merkmalszusammenhänge

### 6.1 Streudiagramme

Wie Sie möglicherweise durch Ausprobieren des Befehls `demo(graphics)` gesehen haben, finden Sie in *R* zahlreiche Möglichkeiten zur Auswahl und Gestaltung von Diagrammen. Um als Beispiel Streudiagramme erstellen zu können, kehren wir zum Beispieldatensatz zurück, der im Rahmen eines Gedächtnisexperiments entstanden ist (s. Abschn. 2.1). Eine Frage könnte lauten, ob, und wenn ja, wie die Gedächtnisleistung beim Erinnern von positiven mit der von negativen Adjektiven übereinstimmt. Um eine erste Antwort auf diese Frage durch Interpretation einer Grafik zu erhalten, erzeugen wir ein Streudiagramm auf folgende Weise:

```
> plot(daten$positiv, daten$negativ, main="", xlab="erinnerte positive  
Adjektive", ylab="erinnerte negative Adjektive")
```

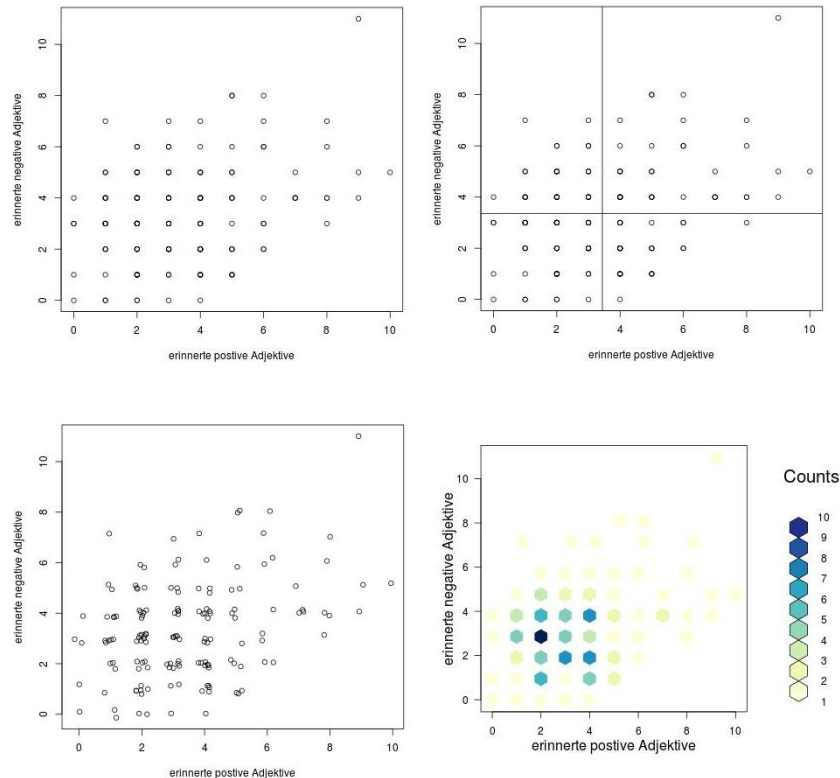
Abbildung 6.1 (oben links) zeigt das Ergebnis. Die Auswahl, welche Daten auf der *x*- und welcher auf der *y*-Achse dargestellt werden, ist hier willkürlich und drückt sich im Befehl oben in der Reihenfolge der Nennung im Argument des `plot()`-Befehls aus. Es gibt allerdings Situationen, in denen eine bestimmte Anordnung für die Interpretation hilfreicher ist.

Abbildung 6.1 (oben links) lässt sich so interpretieren, dass Personen, die viele positive Adjektive erinnern, tendenziell auch viele negative erinnern. Die Variante oben rechts in der Abbildung lässt das noch deutlicher werden, da erkennbar wird, dass Probanden, die mehr positive Adjektive als der Durchschnitt erinnern, ebenfalls mehr negative Adjektive als der Durchschnitt erinnern. Die Linien, die jeweils den Mittelwert einer der beiden Variablen repräsentieren, kann man dem Diagramm sehr einfach durch die beiden Befehle

```
> abline(v=mean(daten$positiv))  
> abline(h=mean(daten$negativ))
```

hinzu fügen. Ungelöst bleibt aber das Problem, dass sich bei Abbildungen dieser Art hinter jedem Punkt die Daten unterschiedlich vieler Probanden befinden können – diese Information kann wichtig sein, wird in den bis hierhin besprochenen Darstellungen aber nicht repräsentiert. Das Streudiagramm in Abbildung 6.1 unten links löst dies, indem zu jedem Probanden ein Punkt gedruckt wird, diese Punkte im Diagramm aber nicht genau übereinander liegen (wo man sie nicht unterscheiden könnte), sondern mit kleinen statistischen Schwanken horizontal und vertikal versetzt gedruckt werden. Dadurch werden sie unterscheidbar und man erhält einen Eindruck davon, wie vielen Probanden hier ein Punkt zugeordnet ist. Diese Zeichenfunktion wird durch den Befehl `jitter()` aktiviert:

```
> plot(jitter(daten$positiv), jitter(daten$negativ), main="",  
       xlab="erinnerte positive Adjektive",  
       ylab="erinnerte negative Adjektive")
```



**Abbildung 6.1:** Verschiedene Streudiagramme: Oben links die einfachste Form, rechts daneben ergänzt um zwei Linien, die die Mittelwerte der beiden Variablen kennzeichnen. Links unten eine Variante, die erkennen lässt, wie viele Probanden einem Datenpunkt zugeordnet sind. Unten rechts ist die Anzahl der Probanden pro Datenpunkt farblich kodiert. Viele weitere Varianten sind möglich.

Eine andere Möglichkeit, diese Information im Diagramm deutlich werden zu lassen, ist eine zusätzliche farbliche Kodierung der Punkte im Diagramm. Das Beispiel unten rechts in [Abbildung 6.1](#) zeigt dafür eine Möglichkeit. Das Diagramm wurde erzeugt unter Verwendung der Bibliotheken `hexbin` und `RColorBrewer`:

```
> library(hexbin)
> library(RColorBrewer) # muss eventuell nachinstalliert werden!
> plot(hexbin(daten$positiv, daten$negativ, xbins=20), main="",
       xlab="erinnerte positive Adjektive",
       ylab="erinnerte negative Adjektive",
       colramp=colorRampPalette(brewer.pal(9, "YlGnBu")))
```

`hexbin` sorgt dafür, dass die Punkte als Sechsecke dargestellt werden. Der Parameter `xbins` gibt die Größe der Sechsecke vor, und zwar bezogen auf die Länge der  $x$ -Achse des Diagramms. Hier wurden die Sechsecke so gezeichnet, dass ihr Durchmesser  $1/20$  der Länge der  $x$ -Achse beträgt. Mit dem Befehl `colramp` wird eine Farbpalette gewählt, und zwar in diesem Fall aus der Bibliothek `RColorBrewer`. Die Ziffer 9 gibt die Anzahl der farblichen Abstufungen an, "YlGnBu" zeigt an, welche Farbpalette genau es sein soll, hier ein Übergang von Gelb über Grün nach Blau. Die verschiedenen möglichen Farbplatten aus dieser Bibliothek kann man einsehen, wenn man `help(brewer.pal)` eingibt. Im vorliegenden Fall beträgt die maximale Anzahl von Probanden, die einem Sechseck zugeordnet sind, 10,

daher wären eigentlich zehn unterschiedliche Farbabstufungen sinnvoll. Lieder bietet die hier gewählte Farbpalette nicht mehr als neun Farben an. Trüge man dennoch 10 statt 9 ein, erhielte man einen entsprechenden Hinweis vom Programm. Die Farben sind daher im Sinne »von bis« zu lesen – gelbe Sechsecke markieren zum Beispiel Datenpunkte, die zu ein bis zwei Probanden gehören, siehe dazu die Legende zum Diagramm in Abbildung 6.1 unten rechts.

## 6.2 Korrelationen

Mit Korrelationen können Zusammenhänge zwischen Merkmalen quantifiziert werden. Je nach Skalenniveau der in Rede stehenden Merkmale gibt es unterschiedliche rechnerische Wege, zu Korrelationen gelangen. In R lautet der Befehl zur Errechnung bivariater Korrelationen `cor()`. Um zusätzlich vorzugeben, nach welcher Methode eine Korrelation bestimmt werden soll, kann die Zeichenkette `method="spearman"` ergänzt werden ("`pearson`" ist die Standardeinstellung, weiterhin ist `kendall` möglich, für weitere Informationen siehe `help(cor)`). Oft möchte man nicht nur eine Korrelation bestimmen, sondern zusätzlich erfahren, ob die berechnete Korrelation signifikant ist. Für diesen Fall ist der Befehl `cor.test()` geeignet, bei dem – ähnlich dem Befehl `t.test()` – durch die Zeichenkette `alternative="less"` (bzw. "`greater`" oder "`two.sided`", Standardeinstellung) die Art des Hypothesentests spezifiziert werden kann, für einen zweiseitigen Test lautete die Alternativhypothese zum Beispiel, dass die Korrelation nicht gleich Null sei.

Als Beispiel verwenden wir wiederum den Datensatz aus Abschnitt 2.1 (s. S. 7) und gehen der Frage nach, ob es Zusammenhänge zwischen der Anzahl negativer, neutraler und positiver erinnelter Adjektive gebe:

```
> cor(cbind(daten$negativ, daten$neutral, daten$positiv))
      [,1]      [,2]      [,3]
[1,] 1.0000000 0.2867774 0.3369060
[2,] 0.2867774 1.0000000 0.2948303
[3,] 0.3369060 0.2948303 1.0000000
```

Das Ergebnis ist in diesem Fall eine Matrix, aus der die Korrelationen (nach Pearson, diese Einstellung braucht nicht explizit angegeben zu werden) eines jeden der drei Vektoren mit jedem Vektor hervor gehen. Man erkennt, dass jede Variable mit sich selbst die Korrelation 1 hat (Werte in der Hauptdiagonalen). Alle anderen Einträge sind hier positiv, aber kleiner als 1. Man erkennt außerdem, dass die Werte im Dreieck links unterhalb der Hauptdiagonalen dieselben sind wie in jenem oberhalb rechts dieser Diagonalen – es ist also gleichgültig, ob nach der Korrelation von `daten$negativ` mit `daten$positiv` gefragt wird oder umgekehrt.

### 6.2.1 Signifikanztest für Korrelationen

Um für Korrelationen die entsprechenden Signifikanztests durchzuführen, ist folgendes Vorgehen geeignet (wir testen zweiseitig, `alternative="two.sided"` braucht nicht angegeben zu werden):

```
> cor.test(daten$negativ, daten$neutral)

Pearson's product-moment correlation

data:  daten$negativ and daten$neutral
t = 3.6418, df = 148, p-value = 0.0003738
```

```
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.1326106 0.4273961
sample estimates:
      cor
0.2867774
```

Die Korrelation ist also hoch signifikant. Für die anderen Korrelationen (mit den Kombinationen `daten$negativ`, `daten$positiv` und `daten$neutral`, `daten$positiv`) ergeben sich ebenfalls hoch signifikante Werte. Die Wahrscheinlichkeiten sind sämtlich sehr klein. In einem Bericht über diese Ergebnisse würde man im eben gezeigten Fall allerdings nicht  $p = 0.0003738$  angeben, sondern  $p < 0,001$ . Oft wird auch  $p < .001$  angegeben. Letzteres entspricht den Gepflogenheiten im angelsächsischen Sprachraum, aber nicht den Regeln deutschsprachiger Typografie, die ein Komma als Dezimaltrennsymbol vorsieht.

## 6.3 Partialkorrelationen

In Kapitel 4.1 des Buches wurde das Konzept der Partialkorrelation besprochen. Das dort vorgestellte Beispiel soll hier nun mit Hilfe von *R* nachvollzogen werden. Dafür muss der entsprechende Datensatz eingelesen werden (achten Sie darauf, ihn beim Herunterladen aus dem Netz ohne Grundstrich abzuspeichern):

```
> library(foreign)
> feuerdaten <- read.spss(file="PartialkorrelationA3.sav",
                           to.data.frame=TRUE)
```

Da der Datensatz mehr Zeilen als nötig enthält (alle Zeilen jenseits der zehnten enthalten nur fehlende Werte), wählen wir nur die ersten zehn Zeilen:

```
> feuerdaten <- feuerdaten[c(1:10),]
```

Es ist nun möglich, die Korrelationen (nach Pearson) in Form einer Matrix berechnen zu lassen:

```
> m <- cor(feuerdaten)
> m
      fleute  schaden  brand
fleute 1.0000000 0.6319160 0.7203427
schaden 0.6319160 1.0000000 0.8027745
brand  0.7203427 0.8027745 1.0000000
```

Die Partialkorrelationen können nun mit Hilfe der im Buch angegebenen Rechenvorschrift bestimmt werden:

```
> (m[1,2]-m[2,3]*m[1,3])/sqrt((1-m[2,3]^2)*(1-m[1,3]^2))
[1] 0.1297005
```

Eine andere, etwas effizientere Möglichkeit bietet der Befehl `partial.r()`, der in der Bibliothek `psych` enthalten ist:

```
> library(psych)
> partial.r(feuerdaten,c(1,2),3)
partial correlations
      fleute schaden
```

```
fleute      1.00      0.13
schaden     0.13      1.00
```

Der erste Eintrag im Argument des Befehls `partial.r` bezeichnet den Datensatz, das zweite einen Vektor bestehend aus den Variablen 1 (Feuerwehrleute) und 2 (Schaden), das dritte Argument schließlich bezeichnet die Variable, die heraus partialisiert wird (Schwere des Brandes).

### 6.3.1 Signifikanztest für Partialkorrelationen

In Abschnitt 6.3 wurde beispielhaft erklärt, wie Partialkorrelationen mit R berechnet werden können. In Zusammenhang mit Partialkorrelationen stellt sich – wie im Falle von Korrelationen – die Frage, ob sie signifikant sind oder nicht. Um diese Frage exemplarisch zu klären, kehren wir zu dem Beispiel zurück, das Gegenstand von Abschnitt 6.3 war. Es geht uns also darum, zu klären, ob die dort gefundene Partialkorrelation zwischen der Anzahl der Feuerwehrleute und dem verursachten Schaden, die sehr viel geringer ist als die nicht bereinigte Korrelation, signifikant ist. Im Buch wird dazu die Rechenvorschrift

$$t_{df} = r_{xy|z} \cdot \sqrt{\frac{N-2}{1-r_{xy|z}^2}}$$

gegeben, die zur Beantwortung der Frage heran gezogen werden soll. Wir weisen der Partialkorrelation  $r_{xy|z}$  zunächst den Wert `r` zu, um die Rechnung übersichtlicher zu gestalten (aus der Matrix, die der Befehl `partial.r()` erzeugt, wird dazu das Element in der ersten Zeile und in der zweiten Spalte ausgewählt, also `[1, 2]`):

```
> r <- partial.r(feuerdaten, c(1, 2), 3)[1, 2]
```

Für den  $t_{df}$ -Wert ermittelt man anschließend unter Verwendung der eben zitierten Formel ( $N = 10$ ):

```
> t <- r*sqrt((10-2)/(1-r^2))
> t
[1] 0.3699734
```

Um zu ermitteln, welche Fläche dieser Wert bei **zweiseitiger** Testung unter der Dichtefunktion der  $t$ -Verteilung abschneidet, verwenden wir den Befehl `pt()` unter Angabe des Freiheitsgrades von  $df = 10 - 3 = 7$ . Der Befehl `pt(t, 7)` liefert den Inhalt der Fläche unter der Dichtefunktion, der links vom  $t$ -Wert liegt, und zwar bei  $df = 7$ . Der Befehl `1-pt(t, 7)` liefert daher den Flächeninhalt rechts vom besagten  $t$ -Wert, der Faktor 2 rührt von der zweiseitigen Testung her:

```
> 2*(1-pt(t, 7))
[1] 0.7223422
```

Unter der Annahme der Nullhypothese ist die beobachtete Partialkorrelation also sehr wahrscheinlich. Sie ist also nicht nur sehr viel kleiner als der zuvor berechnete, nicht bereinigte Wert, sie ist zudem auch nicht mehr signifikant.

## 6.4 Lineare Regression

### 6.4.1 Durchführung und Interpretation an einem Beispiel

Für die Betrachtungen in diesem Abschnitt benötigen Sie den Datensatz `Regression_A3`, den Sie auf der Internetseite zum Buch finden. Beim Speichern der Datei sollten Sie ihr einen Namen geben, der keinen Grundstrich enthält.

```
> library(foreign)
> regression <- read.spss(file="RegressionA3.sav",
                           to.data.frame=TRUE)
```

Da der Datensatz jenseits der 10. Zeile weitere Zeilen ohne Werte enthält, schneiden wir den nicht benötigten Teil des Datensatzes ab:

```
> regression <- regression[c(1:10), ]
> regression
  Alko Reak
1   0.0  590
2   0.3  581
3   0.5  687
4   0.7  658
5   1.0  632
6   1.2  645
7   1.4  687
8   1.8  624
9   2.3  702
10  2.5  789
```

Die Zusammenhänge zur linearen Regression in Kapitel 4.2 des Buches bilden eine Einführung in den Themenkomplex dieser Art des Umgangs mit statistischen Daten. Aus diesem Grund bieten die Funktionen in R, die im Folgenden verwendet werden, zahlreiche weitere Optionen, die wir zunächst unbeachtet lassen.

Für den hier zu betrachtenden einfachen Fall sollen die Parameter einer einfachen Regressionsgleichung der Form

$$\hat{y} = b \cdot x + a$$

bestimmt werden. Die Anforderung, einen Parameter eines *Modells* dieser Form zu bestimmen, wird R durch den Befehl `lm(y ~ x)` mitgeteilt, wobei `lm` für *linear model* steht. Es wird sich als praktisch erweisen, dem Ergebnis dessen, was R durch diesen Befehl erzeugt, ein eigenes Objekt zuzuordnen, hier wählen wir `fm` für »fitted model«

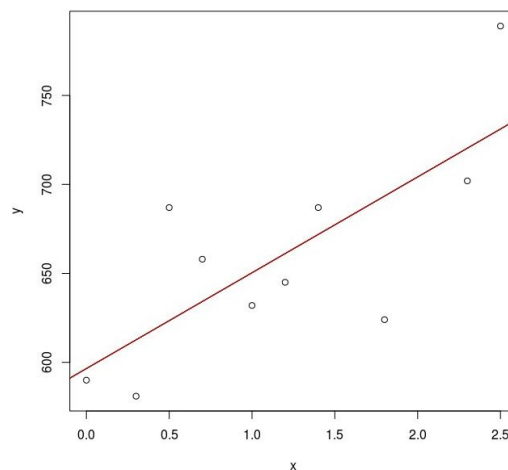
```
> y <- regression$Reak
> x <- regression$Alko
> fm <- lm(y ~ x)
> fm

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)          x
    596.50         53.84
```

Für die gesuchten Parameter  $a$  und  $b$  ergibt sich nun:  $a = 596.50$  ms,  $b = 53.84$  ms. Damit lässt sich nun bereits ein einfaches Diagramm zeichnen, wie man es in ähnlicher Form auch im Buch findet, siehe Abbildung 6.2.





**Abbildung 6.2:** Das Diagramm zeigt die Daten aus dem Datensatz RegressionA3.sav sowie eine Regressionsgerade.

```
> plot(x, y)
> abline(fm, col="red")
```

Mit den Befehlen `fitted.values()` und `residuals()` lassen sich die durch die Regressionsgleichung vorhergesagten Werte bzw. Residuen ausgeben:

```
> fitted.values(fm)
      1      2      3      4      5      6
596.5021 612.6554 623.4243 634.1932 650.3465 661.1153
      7      8      9     10
671.8842 693.4219 720.3441 731.1130
> residuals(fm)
      1      2      3      4      5
-6.502136 -31.655434  63.575700  23.806834 -18.346464
      6      7      8      9     10
-16.115330  15.115804 -69.421927 -18.344091  57.887043
```

Der Korrelationskoeffizient (im Fall der einfachen linearen Regression identisch mit dem  $\beta$ -Gewicht) wird über den Zusammenhang

$$\beta_{yx} = b \cdot \frac{\sigma_x}{\sigma_y}$$

bestimmt, was in R manuell so umgesetzt werden kann:

```
> fm$coefficients[2]*sd(x)/sd(y)
      x
0.7417239
```

Die `[2]` teilt mit, dass der zweite der beiden Koeffizienten ( $b$ ) im Modell für die Rechnung verwendet werden soll.

Für den Determinationskoeffizienten  $r^2$  erhält man:

```
> (fm$coefficients[2]*sd(x)/sd(y))^2
      x
```

0.5501544

Anstatt dieser schrittweisen Vorgehensweise lässt sich auch eine kompakte Ausgabe anfordern, und zwar mit dem Befehl `summary()`:

```
> summary(fm)

Call:
lm(formula = y ~ x)

Residuals:
    Min       1Q   Median       3Q      Max
-69.42 -18.35 -11.31   21.63   63.58

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    596.50     24.35   24.496 8.24e-09 ***
x              53.84     17.21    3.128  0.0141 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 43.28 on 8 degrees of freedom
Multiple R-squared:  0.5502,    Adjusted R-squared:  0.4939
F-statistic: 9.784 on 1 and 8 DF,  p-value: 0.01406
```

Die Ausgabe gibt zunächst Informationen darüber, wie sich die Residuen auf die Quartile verteilen. Dass die hier aufgelisteten Werte bis auf die Maximal- und Minimalwerte nicht in der Liste der Residuen von oben (Ausgabe `residuals()`) vorkommen, liegt an der Anzahl der Werte insgesamt. Da es zehn Werte sind, wird zwischen dem an 5. und 6. Stelle (in der nach Größe geordneten Reihe) liegenden Wert arithmetisch gemittelt, um den Median zu erhalten. Um zum Beispiel den Wert für das 75%-Quartil (3Q) zu erhalten, wird der an siebter Stelle stehende Wert mit 0,25, der an achter Stelle stehende Wert mit 0,75 gewichtet ( $15,116 \cdot 0,25 + 23,807 \cdot 0,75 = 21,634$ ).

In der Ausgabe folgen dann die schon bekannten Koeffizienten für den Achsenabschnitt (»Intercept«) und die Steigung (als Koeffizient für  $x$ ).

Weiterhin werden die Standardfehler für diese beiden Koeffizienten angegeben (»Std. Error«). Sie schätzen die Streuung der gefundenen Koeffizienten um den Populationswert und sind damit inferenzstatistische Kennwerte.

Für jeden der beiden Koeffizienten  $a$  und  $b$  wird ein t-Test auf Signifikanz durchgeführt. Die Nullhypothese lautet jeweils, dass der betroffene Koeffizient  $a$  bzw.  $b$  Null betrage, also zum Beispiel  $H_0: b = 0$ . Für den Achsenabschnitt ist die Frage der Signifikanz wenig bedeutsam, da es hier lediglich darum geht, in welcher Höhe die Regressionsgerade die  $y$ -Achse schneidet. Anders ist es mit der fraglichen Signifikanz für die Steigung ( $b$ ). Dieser Wert ist hoch signifikant, was bedeutet, dass die Alkoholkonzentration im Blut ein signifikanter Prädiktor für die Reaktionszeit ist. Wäre hingegen dieser Wert nicht signifikant, dann gälte die Vorhersage als nicht verlässlich.

Der Standardschätzfehler (»Residual standard error«) wird angegeben (hier 42,28), der ein Maß für die Streuung der tatsächlichen, empirischen Werte um die durch das Modell vorhergesagten Werte ist.

Schließlich ist der Determinationskoeffizient  $r^2$  angegeben, hier als »R-squared«. Da die lineare Regression auf der Basis der Stichprobenwerte den Zusammenhang, wie er in der Population vorliegt,

überschätzt, wird außerdem ein korrigierter Wert für den Determinationskoeffizienten angegeben, der entsprechend etwas niedriger ausfällt.

Die letzte Zeile der Ausgabe bezieht sich auf varianzanalytische Betrachtungen, die an dieser Stelle nicht weiter thematisiert werden sollen.

#### 6.4.2 Zusammenhang zwischen Korrelation, Regression und t-Test

Kapitel 4.1 hat gezeigt, dass eine punktbiseriale Korrelation und ein t-Test konzeptuell eng verwandt sind. Die Abbildungen 4.7 und 4.8. auf Seite 142 zeigen dies. Der t-Test betrachtet den Mittelwertunterschied, die punktbiseriale Korrelation hingegen fragt nach dem Zusammenhang zwischen einem dichotomen Merkmal (Zugehörigkeit zu einer Gruppe A oder B) und einem anderen intervallskalierten Merkmal. Kapitel 4.2 behandelte die Zusammenhänge zwischen Korrelation und Regression. Die Frage, die die folgenden Überlegungen leitet, ist nun die nach einem möglichen Zusammenhang zwischen der Regression und dem t-Test.

Wir bereiten zunächst die Berechnung einer punktbiserialen Korrelation vor. Hierfür gibt es zwar einen eigenen Befehl im Paket `ltm`, dabei wird jedoch kein Signifikanztest durchgeführt. Aus diesem Grund verwenden wir den üblichen Befehl `cor.test()`, der jedoch keine nicht-numerischen Daten akzeptiert, wie sie hier in Form einer Zeichenkette wie `weiblich` oder `maennlich` vorliegen. Um dieses Problem zu lösen, wählen wir aus dem Beispieldatensatz (Abschnitt 2.1) die Spalte aus, die die Information über das Geschlecht enthält und kodieren mittels einer Schleife in der Weise, dass Männern eine 1, Frauen eine 2 zugeordnet wird:

```
s <- mydata$sex
s <- as.character(s)
> for (i in 1:150) if (s[i] == "weiblich") s[i] <- 2 else s[i] <- 1
```

Da die Schleife Buchstabenketten ("weiblich") erwartet, müssen die Einträge in `s` entsprechend definiert werden (`as.character()`). Nach Durchlaufen der Schleife werden die gewonnenen Einträge wieder zu Zahlen rückdefiniert, um sodann den Test durchzuführen:

```
> s <- as.numeric(s)
> cor.test(s, mydata$ges)

Pearson's product-moment correlation

data:  s and mydata$ges
t = 1.8134, df = 148, p-value = 0.07179
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.01313852  0.30059259
sample estimates:
      cor
0.1474334
```

Das Ergebnis ist so zu lesen, dass Frauen marginal signifikant mehr Adjektive erinnern als Männern. Diese Aussage ergibt sich unter anderem daraus, dass Frauen mit 2 (und Männern mit 1) kodiert wurden, dass also ein höherer Wert für das Geschlecht mit einer höheren Anzahl erinnerter Adjektive einher geht (der Korrelationskoeffizient ist positiv).

Nun legen wir uns die Aufgabe vor, mit Hilfe einer einfachen linearen Regression aus dem Geschlecht

### die Erinnerungsleistung vorherzusagen:

```
> reg <- lm(mydata$ges~s)
> summary(reg)

Call:
lm(formula = mydata$ges ~ s)

Residuals:
    Min       1Q   Median       3Q      Max
-9.5408 -3.1923 -0.5408  2.8077 15.4592

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   7.8438     1.2794   6.131 7.56e-09 ***
s             1.3485     0.7436   1.813  0.0718 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.334 on 148 degrees of freedom
Multiple R-squared:  0.02174,    Adjusted R-squared:  0.01513
F-statistic: 3.289 on 1 and 148 DF,  p-value: 0.07179
```

Die Ausgabe zeigt einen Wert  $r^2 = 0,02174$ , aus dem sich  $r = 0,14744$  errechnen lässt – derselbe Koeffizient, der auch oben für die Korrelation berechnet wurde. Die letzte Zeile der Ausgabe zeigt zudem einen p-Wert, der ebenfalls aus dem Signifikanztest für die Korrelation im Beispiel oben bekannt ist. Dies passt zu den Ausführungen im Buch in Abschnitt 4.1 über die Umrechnung punktbiserialer Korrelationen in t-Werte.

Die Betrachtungen zeigen, dass Frauen in der Tendenz mehr Adjektive erinnern als Männer. Fraglich ist indessen, ob dieser Zusammenhang statistisch bedeutsam ist. Wir untersuchen dies mit einem t-Test für unabhängige Stichproben:

Zunächst werden die Datensätze zu Frauen und Männern ausgewählt:

```
> frauen <- subset(mydata, as.character(mydata$sex)=="weiblich")$ges
> maenner <- subset(mydata, as.character(mydata$sex)=="maennlich")$ges
```

und dann dem Test unterzogen:

```
> t.test(frauen,maenner,var.equal=TRUE)

Two Sample t-test

data:  frauen and maenner
t = 1.8134, df = 148, p-value = 0.07179
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.1209881  2.8180054
sample estimates:
mean of x mean of y
```

10.540816	9.192308
-----------	----------

In diesem Test wurde die Varianzhomogenität vorausgesetzt, von der man sich mit dem Levene-Test überzeugen kann (Abschnitt 5.1.1 auf S. 28) – das Ergebnis des Tests auf Varianzhomogenität zeigt ein nicht signifikantes Ergebnis. Der t-Wert entspricht dem t-Wert für das Betagewicht der linearen Regression, und auch die dazu gehörigen Wahrscheinlichkeiten stimmen überein. Außerdem entspricht das nicht standardisierte Regressionsgewicht von oben (1, 35) dem Mittelwertunterschied des t-Tests. Die Regression einer intervallskalierten Variable auf der Basis einer dichotomen Variablen und ein t-Test für unabhängige Stichproben sind offenbar konzeptuell identisch. Unter dieser Perspektive stellt sich der t-Test als Spezialfall einer Regression dar. Allerdings sind die Möglichkeiten der Regression grundsätzlich viel weiter gefasst als die des t-Tests, insbesondere kann sie mit Prädiktoren auf anderen Skalenniveaus umgehen, wie in Kapitel 4.2 gezeigt wurde. Zudem kann die Regressionsrechnung grundsätzlich auch mit mehr als einem Prädiktor arbeiten, hierzu sind im Buch im Schlussteil von Kapitel 4 Hinweise enthalten.