Vogelwarte 48, 2010: 119 – 135

Das freie Statistikpaket „R“: Eine Einführung für Ornithologen

Fränzi Korner-Nievergelt & Ommo Hüppop

Korner-Nievergelt F & Hüppop O 2010: The free statistic software “R”: An introduction for ornithologists. Vogelwarte 48: 119-135.

Publishing ornithological data requires the application of adequate statistical methods. With the improvement of methods, software requirements are arising, but available proprietary programs are often far beyond the financial limits of the users. The statistic package R is a free but nevertheless very sophisticated alternative. Regrettably, it is not easy to get started with R since it is not clickable and needs the typing of code. This article is a step by step introduction for R-beginners. The reader can experience the input of data, their handling and visualisation at his computer. We guide the user through a t-test and develop a simple linear model including analysis of residuals. Finally, we suggest books for further reading.

✉ FK-N: Schweizerische Vogelwarte, Seerose 1, CH-6204 Sempach und oikostat GmbH, Ausserdorf 43, CH-6218 Ettiswil, E-mail: fraenzi.korner@oikostat.ch
OH: Institut für Vogelforschung „Vogelwarte Helgoland“, Inselstation, Postfach 1220, D-27494 Helgoland, E-mail: ommo.hueppop@ifv-vogelwarte.de

Einleitung

Die Ansprüche an Auswertungen ornithologischer Studien steigen beständig. Herausgeber, aber auch Leser von Fachzeitschriften fordern eine statistische Prüfung möglichst aller präsentierten Ergebnisse. Sie möchten wissen, wie sicher die aufgestellten Hypothesen durch die Daten belegt werden. Laufend werden neue Auswertemethoden entwickelt, die neue Anforderungen an die Software stellen oder diese sogar überfordern. Das Statistikpaket R (R Development Core Team 2010) ist ein flexibles, ausbaubares Softwarepaket, das den Ansprüchen moderner Datenanalysen standhält. Neu entwickelte Analysemethoden können einfach eingebaut werden, da die Entwickler ihre Prozeduren in Paketen der Öffentlichkeit zur Verfügung stellen können. In R lassen sich allerdings keine Auswertungsprozeduren auswählen, indem man sich durch Menüs klickt oder Häkchen setzt. Statistische Funktionen und die zu übergebenden Parameter müssen vielmehr eingetippt werden oder zumindest als Skript (siehe unten) vorliegen. Dies hat den Vorteil, dass man immer genau verstehen muss, was gerechnet werden soll. So wird auch der statistische Output verstanden und die Interpretation einfacher. Zusätzlich bietet R die Möglichkeit, individuelle, wissenschaftliche Grafiken und eigene Auswertungsfunktionen zu erstellen. Somit ist diese Software für jedermann den eigenen Bedürfnissen und Kenntnissen entsprechend beliebig erweiterbar.

Nicht zuletzt ist R eine unter www.r-project.org für verschiedene Betriebssysteme (Linux, Mac OS X, Windows) frei erhältliche Software – ein großer Vorteil, wenn man bedenkt, dass man für ein konventionelles Statistikprogramm bis über 10.000 EURO bezahlen muss. Wohl aus oben genannten Gründen hat sich R in den letzten Jahren unter Ornithologen bereits stark verbreitet. Zum Beispiel wurde im letzten Jahrgang des „Journal of Ornithology“ (Vol. 150, 2009) gut jede fünfte (22 %) Analyse mit R durchgeführt (Von 81 Artikeln, die statistische Analysen präsentieren, machen 22 keine Angaben über die verwendete Software. Die anderen verwendeten: 30 % SPSS oder SYSTAT, 22 % R, 18 % SAS oder JMP, 10 % STATISTICA).

Der Einstieg in R ist aber nicht einfach, denn R ist eine Programmiersprache, die man lernen muss und bei seltenem Gebrauch wieder vergisst. R eignet sich für Personen, die häufig mit Daten arbeiten, diese darstellen und analysieren wollen. Mit diesem Artikel möchten wir den Einstieg in R erleichtern und schmackhaft machen. Er ist für R-Anfänger geschrieben, die ein statistisches Grundwissen mitbringen. Wir zeigen u. a., wie Daten in R eingelesen, betrachtet, verändert und gespeichert, wie einfache Grafiken erstellt und klassische Tests durchgeführt werden. Sie sollten dazu den Computer gestartet und R geöffnet haben, damit Sie alle Prozeduren selbst ausführen, verändern und durch „Herumspielen“ erlernen können.

Geschichte und Philosophie von R

R wurde 1992 durch Ross Ihaka und Robert Gentleman (man beachte die Anfangsbuchstaben der beiden Vornamen!) zunächst für Lehrzwecke entwickelt (Ligges 2008). Der R-Benutzer soll interaktiv mit Daten rechnen können und selbst zum Programmierer werden. Grafiken sollen für die explorative Datenanalyse und Präsentation erstellt und bereits entwickelte Funktionen wieder verwendet werden können (Chambers 2008). R hat eine rasante Entwicklung hinter sich: 1993 erschien

die erste Binärversion. Das „R Development Core Team“ wurde 1997 gegründet. Heute sind darin 19 Personen aus Forschung und Wirtschaft aus der ganzen Welt vereinigt (Ligges 2008, www.r-project.org). 1998 wurde das „Comprehensive R Archive Network“ (CRAN) gegründet. CRAN ist ein Internet-Netzwerk für R-Benutzer und R-Entwickler. Version R-1.0.0 erschien im Jahr 2000 (seit April 2010 ist die Version 2.11.0 verfügbar). Von 2001 bis 2008 folgten die „RNews“, seit 2009 „The R Journal“, ein kostenloses Online-Journal, in dem R-relevante Artikel erscheinen, unter anderem neue Packages (von Anwendern erstellte, nachladbare Pakete von Funktionen für spezielle Aufgaben) vorgestellt und Änderungen von Updates erläutert werden. 2002 wurde die R Foundation (www.r-project.org/foundation) gegründet und seit 2004 finden inzwischen jährliche Anwenderkonferenzen „useR!“ an verschiedenen Orten statt. Seit 2007 gehören gut besuchte R-Kurse auch zum festen Programm der Jahresversammlungen der Deutschen-Ornithologen Gesellschaft (Korner-Nievergelt et al. 2007).

Vieles spricht dafür, dass R das Statistikpaket der Zukunft ist, da es frei erhältlich, sehr flexibel und endlos erweiterbar ist. Prozeduren lassen sich leicht zwischen Anwendern austauschen. Einmal geschriebener Code, z. B. einer Analyse oder für die Erstellung einer Grafik, muss für die Wiederholung mit neuen Daten oder in leicht abgeänderter Form nicht neu geschrieben werden. Lästiges Durchklicken endloser Menüs entfällt. Arbeitsabläufe können daher enorm verkürzt und für wiederkehrende Aufgaben weitgehend automatisiert werden. Nicht zuletzt verdeutlicht auch die exponentiell steigende Zahl der Packages (Ende April 2010 waren es schon 2338) die breite Akzeptanz und Verbreitung von R.

R-Installation und Dokumentation

R kann von www.r-project.org heruntergeladen und in gängiger Weise des jeweiligen Betriebssystems installiert werden. Auf dem umfassenden R Archiv Netzwerk CRAN finden Sie zahllose Dokumentationen: z. B. Einführungen, Handbücher, häufig gestellte Fragen, ebenso die RNews, The R Journal und alle Packages.

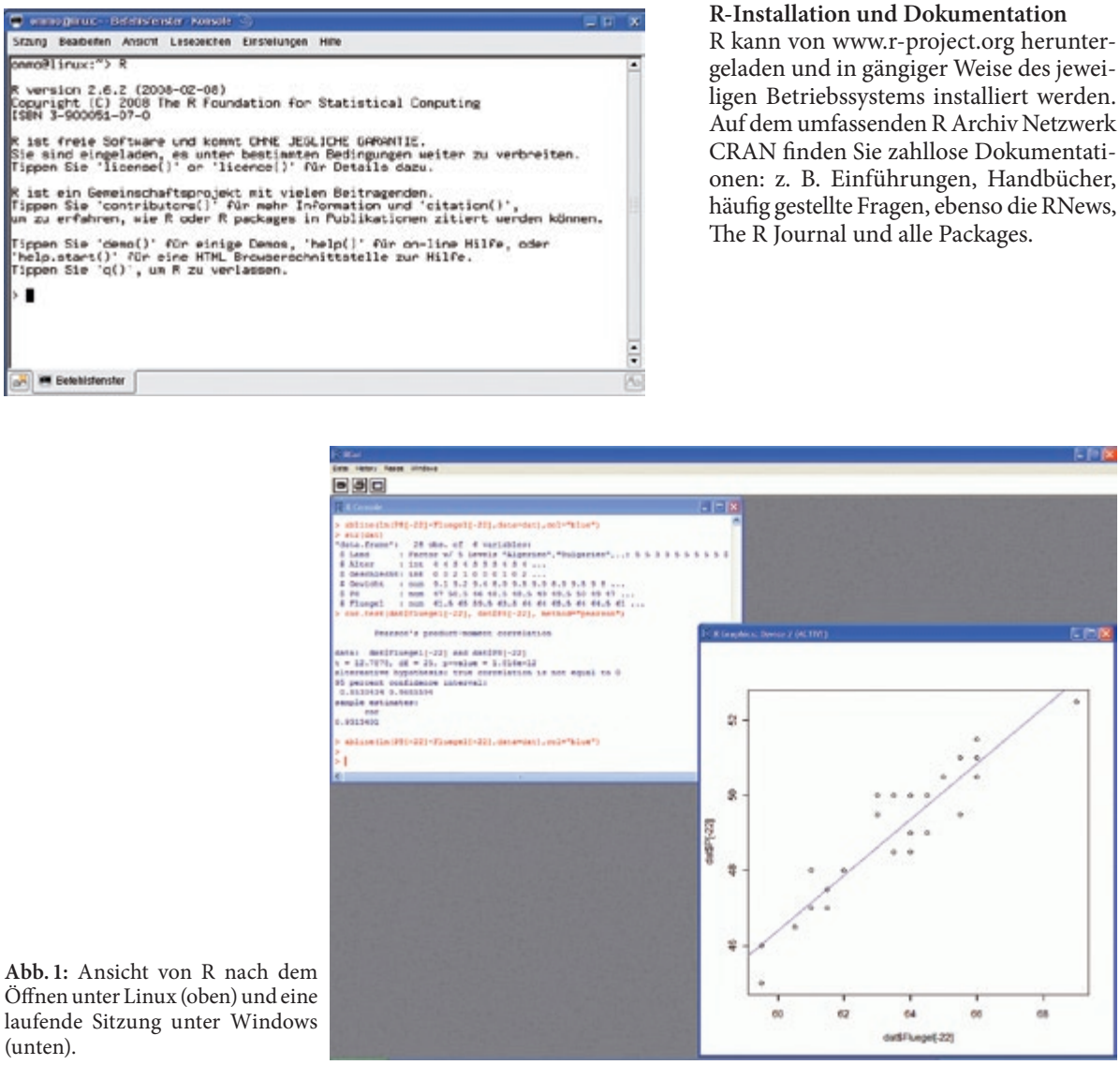

Abb. 1: Ansicht von R nach dem Öffnen unter Linux (oben) und eine laufende Sitzung unter Windows (unten).

Erste Schritte

Öffnen Sie R, indem Sie unter Linux „R“ im Befehlsfenster eintippen oder unter Windows bzw. Mac OS X das R-Symbol auf ihrem Desktop doppelklicken. R wird sich Ihnen wie in Abb. 1 präsentieren. Der wichtigste Teil im Programm ist die geöffnete R-Konsole. Darin zeigt das „>“-Zeichen („Prompt-Zeichen“), dass R bereit ist, Befehle entgegen zu nehmen. Sie können direkt in diese Konsole schreiben.

Alle R-Codes in diesem Artikel können als Text-File vom Internet heruntergeladen werden (http://www.oikostat.ch/data/korner_u_hueppop_rcode.r). Das Text-File kann z. B. mit dem R-Editor oder mit Tinn-R (siehe unten) geöffnet werden. Dies erspart Ihnen das mühsame Abtippen des Codes aus diesem Artikel.

Um etwas mit R vertraut zu werden, geben Sie zunächst ein paar ganz einfache Code-Zeilen ein (Das Sternchen ist in R wie in vielen anderen Programmen das Malzeichen):

```
15.3 * 5
```

liefert sofort ein Ergebnis nach dem Drücken der Eingabetaste. Beachten Sie unbedingt, dass R einen Dezimalpunkt und kein Dezimalkomma erwartet! Zahlen können auch einem Objekt mit einem Namen aus einer beliebigen Folge von Buchstaben zugeordnet werden:

```
x = 5.3; y <- 5
```

Die Zuweisung kann entweder über das Gleichheitszeichen oder über „<-“ erfolgen. Bei der zweiten Möglichkeit ist eine Zuweisung nach links oder rechts möglich, je nachdem in welche Richtung der Pfeil weist. Mittels Semikolon können mehrere Funktionen auf derselben Zeile hintereinander gereiht werden. Danach kann mit den so erstellten Objekten weitergerechnet werden:

```
x * y
```

gibt das Ergebnis der Multiplikation direkt aus. Sie können es aber auch einem neuen Objekt zuordnen (In diesem Beispiel weist der Zuweisungspfeil nach rechts):

```
x * y -> ergebnis
```

Durch anschließende Eingabe von ergebnis kann man sich den Wert anschauen (man sollte sich angewöhnen, selbsterklärende Objektnamen anstelle kryptischer Kürzel zu verwenden). Neben den Grundrechenarten stehen etliche weitere arithmetische Operatoren zur Verfügung. Die dazugehörige Hilfe erhält man mit ?Arithmetic (Groß- und Kleinschreibung ist bei R unbedingt zu beachten!). Von diesen arithmetischen Operatoren abgesehen, arbeitet man in R vor allem durch Aufrufe von Funktionen, denen jeweils in Klammern die nötigen Argumente übergeben werden: sqrt(ergebnis) liefert uns beispielsweise die Quadratwurzel aus dem Ergebnis unserer Multiplikation, log(ergebnis) den natürlichen Logarithmus und log10(ergebnis) den dekadischen Logarithmus. Hilfe zu diesen mathematischen Funktionen gibt es über ?log. Dort lesen sie übrigens unter **See also**, dass man mit ?sqrt und ?sin Hilfe zu weiteren mathematischen Funktionen erhält.

Objekte können statt eines einzelnen natürlich auch mehrere Werte enthalten. Einfache Datenreihen werden „Vektoren“ genannt, die man z. B. so definiert:

```
daten <- c(23, 34.5, 67.1, NA, -12. 8, .56)
```

wobei NA einen fehlenden Wert bedeutet („not available“). Die Funktion c() („combine“) fasst mehrere Werte (nicht nur Zahlenwerte, sondern auch Zeichenketten) in einem Vektor zusammen. Mit numerischen Vektoren kann man auch rechnen, probieren Sie daten * y.

Mit unseren ersten Daten können wir auch gleich etwas beschreibende Statistik ausprobieren („#“ leitet einen Kommentar ein. Text, der diesem Zeichen folgt, wird von R nicht interpretiert. Nutzen Sie diese Möglichkeit ausgiebig, damit Sie auch in ein paar Jahren Ihren Code noch verstehen):

```
summary(daten)  # verschiedene statistische Kennwerte
```

```
   Min. 1stQu.  Median   Mean 3rdQu.   Max. NA's
-12.80    0.56   23.00  22.47   34.50  67.10 1.00
```

Falls Sie hier eine Fehlermeldung erhalten, könnte es sein, dass Sie das Objekt daten noch nicht erstellt haben. Beginnen Sie zwei Abschnitte weiter vorne nochmals und lassen Sie alle Befehlszeilen der Reihe nach in R laufen.

Mittelwert und Standardabweichung kann man so berechen:

```
mean(daten, na.rm=TRUE); sd(daten, na.rm=T)
```

```
[1] 22.472
[1] 31.06791
```

Gibt man nur mean(daten) oder nur sd(daten) ein, erhält man hingegen eine Fehlermeldung:

```
[1] NA
```

Mit dem Hilfeaufruf (?mean) kommt man der Ursache auf die Schliche: Das logische Argument na.rm (für „NA remove“) bestimmt, ob fehlende Werte von der Mittelwertberechnung ausgeschlossen werden sollen. Statt TRUE kann man auch einfach ein T schreiben. Wenn einer Funktion mehrere Argumente übergeben werden sollen, werden diese innerhalb der Klammer durch Kommas getrennt.

Die Suche nach unbekannten Funktionen kann angesichts des riesigen Umfangs von R mühevoll werden. Wenn Sie help.start() eingeben, erscheint eine Hilfe-Übersicht. Klicken Sie auf „Packages“. Eine Link-Liste aller auf dem lokalen Rechner installierten Pakete erscheint. „stats“ führt uns beispielsweise zu zahlreichen Statistikfunktionen. Unter ihnen finden wir auch mean() und sd() für Mittelwert und Standardabweichung. Alternativ kann auch die Suchfunktion „Search Engine & Keywords“ benutzt werden. Geben Sie im Eingabefeld „standard deviation“ ein und klicken Sie o.k. Leider blockieren einige Anti-Virusprogramme diese Suchfunktion, so dass Sie vermutlich zuerst explizit „geblockte Inhalte zulassen“ müssen, bevor die Suche ausgeführt werden kann. Denken Sie bitte wieder an den Ausschluss fehlender Werte, wenn Sie die Funktion sd() ausführen:

library(help=“stats“) führt direkt zu einer Liste aller im Paket „stats“ enthaltenen Funktionen. Allerdings kann man in dieser nicht suchen.

Häufig benötigt man regelmäßige Reihen von Werten. Jahre <- c(1960:2010) oder Jahre <- seq(1960,2010) erzeugt einen Vektor mit allen Jahreszahlen von 1960 bis 2010, Jahre <- seq(1960,2010,10) dagegen in Zehnerschritten nur die „runden“ Jahre. Mit rep lassen sich Zahlenreihen beliebig vervielfachen, z. B. erzeugt rep(seq(0,10,2),3) drei Mal hintereinander die Folge aller geraden Zahlen von 0 bis 10. Die Länge eines Vektors, d. h. die Zahl der in ihm enthaltenen Werte, erhält man mit der Funktion length():

```
length(daten)
[1] 6
```

Einzelne Werte, wie hier der zweite unseres Vektors daten, können über einen Index in eckigen Klammern angesprochen werden:

```
daten[2]
[1] 34.5
```

R Editoren

Das Schreiben in der R-Konsole ist mühsam, da die Maus nicht als Cursor benutzt werden kann und keine praktischen Textverarbeitungsfunktionen zur Verfügung stehen. Ebenfalls ist es schwierig, in der R-Konsole erstellten Code zu speichern und später abzuändern. Deshalb empfehlen wir dringend, Code in einem Text-Editor zu erstellen und nur zur Ausführung in die R-Konsole zu kopieren. Sie können einen einfachen Text-Editor benutzen, zum Beispiel den mit R mitgelieferten. Sie erreichen ihn über das Dateimenü von R („Neues Skript“ bzw. „Öffne Skript…“, wenn mit einem bereits gespeicherten R-Code weitergearbeitet werden soll). Erstellen oder verändern Sie Ihren Code in diesem R-Editor. Wenn Sie eine Zeile ausführen möchten, dann setzen Sie einfach den Cursor in die betreffende Zeile und drücken Strg+R (Ctrl+R). Die ganze Zeile wird automatisch in die R-Konsole kopiert und dort ausgeführt. Mit derselben Tastenkombination können Sie auch einen ganzen Abschnitt in R transferieren, wenn Sie ihn vorher markiert haben. Der R-Editor bietet aber keine grafische Hilfe für das Programmieren an, wie das andere auf dem Markt erhältliche Editoren, z. B. WinEdt oder Tinn-R, können.

Tinn-R ist ebenfalls frei erhältlich und sehr empfehlenswert. Die Dokumentation ist allerdings etwas spärlich (http://www.sourceforge.net/projects/tinn-r). Tinn-R verwendet für Funktionen, Text, Kommentare und logische Ausdrücke andere Farben, so dass Tippfehler leicht auffallen. Beim Verwenden von Klammern wird jeweils die zugehörige Klammer markiert. Zusätzlich bietet Tinn-R für jede Funktion ein einfaches aber sehr nützliches Hilfefenster und besitzt eine Funktion, um nach Befehlen zu suchen. Tastenkombinationen für das Transferieren von Code in R können individuell definiert werden. WinEdt ist ein kommerzielles Programm, das wie Tinn-R eine grafische Programmierhilfe bietet, aber keine speziellen R-Hilfefiles enthält. Hinweise auf weitere Editoren, insbesondere für Mac-Benutzer, finden Sie in Ligges (2008). Unter Linux leistet Emacs (http://ess.r-project.org) gute Dienste.

Einlesen von Daten

Das oben dargestellte Eingeben von Daten als Vektoren ist unnütze Mühe, wenn die Daten schon in digitaler Form vorliegen, z. B. in einer Tabellenkalkulation. Viele Leser werden ihre Daten mit Excel oder OpenOffice erfassen

und verwalten. Wie aber macht man diese Daten für R verfügbar? Für das einfache Einlesen in R sollten die Daten als Tabulator-getrennte Text-Dateien vorliegen (unter „Speichern unter“ können Sie diese Option in Ihrer Tabellenkalkulation einstellen). Die erste Zeile kann die Variablennamen enthalten. Die Variablennamen müssen mit einem Buchstaben beginnen. Sie dürfen Punkte enthalten, mit anderen Sonderzeichen ist Vorsicht geboten, und die Überschrift der ersten Spalte darf nicht aus den beiden Großbuchstaben „ID“ bestehen! Leere Zellen, d. h. Zellen mit fehlenden Werten, müssen wieder mit „NA“ gekennzeichnet sein. Tippen Sie folgende Zeile in Ihre R-Konsole ein und drücken Sie die Enter-Taste, um eine Datei direkt von der oikostat-Website in Ihr R einzulesen.

```
dat<-read.table("http://www.oikostat.ch/data/parusatermorph.txt", header=TRUE)
```

Wenn eine Fehlermeldung erscheint, funktioniert vermutlich Ihre Internetverbindung nicht oder es ist Ihnen ein Tippfehler unterlaufen. Das „<-“-Zeichen fungiert auch hier wieder als Zuweisungspfeil. Die Funktion read.table liest ein auf dem Computer oder im Internet vorhandenes Datenfile ein. read.table braucht als erstes Argument den Namen des Datenfiles inklusive Pfad. Ein auf Ihrer Festplatte vorhandenes Datenfile lesen Sie z. B. so ein: read.table("c:/Ordner1/Ordner2/Name.txt", dec=","), wobei Sie natürlich Ordner- und Filenamen anpassen müssen. Dabei muss, anders als sonst unter Windows üblich, der normale Schrägstrich („/“) verwendet werden, nicht der Backslash („\“). Mit dem Argument dec lässt sich angeben, dass die einzulesenden Daten Dezimalkommas und keine Dezimalpunkte (die Voreinstellung) aufweisen, was z. B. der Fall ist, wenn Werte wie oben beschrieben aus einem deutschsprachigen Excel exportiert wurden. Das Argument header wird auf TRUE gesetzt, wenn die erste Zeile die Variablennamen enthält, sonst setzt man dieses Argument auf FALSE. Das Datenfile parusatermorph.txt stammt aus Korner-Nievergelt & Leisler (2004). Wie ?read.table zeigt, lassen weitere Argumente eine sehr flexible Anpassung der Funktion an die formale Struktur des einzulesenden Datenfiles zu. Mit skip lässt sich beispielsweise bestimmen, wie viele Reihen am Anfang übersprungen werden sollen (z. B. Kommentare zur Beschreibung der Daten), nrows legt die Zahl der einzulesenden Zeilen fest.

str(dat) zeigt uns die Struktur der eingelesenen Datentabelle (Data-Frame) an:

```
,data.frame':    28 obs. of  6 variables:
$ Land      :  Factor w/ 5 levels "Algerien","Bulgarien",..: 5 5
              3 3 5 5 5 5 5 5 ...
$ Alter     :  int  4 4 3 4 3 3 3 4 3 4 ...
$ Geschlecht:  int  0 0 2 1 0 0 0 1 0 2 ...
$ Gewicht   :  num  9.1 9.2 9.4 8.9 9.5 9.9 8.9 9.5 9 8 ...
$ P8        :  num  47 50.5 46 48.5 48.5 49 49.5 50 49 47 ...
$ Fluegel   :  num  61.5 65 59.5 63.5 64 64 65.5 64 64.5 61 ...
```

Unter dem Namen dat ist ein Objekt der Klasse „data.frame“ (Datentabelle) gespeichert, das 28 Beobachtungen und 6 Variablen enthält: Für 28 Tannenmeisen *Parus ater* jeweils Herkunft (Land), Alter, Geschlecht, Gewicht, Länge der 8. Handschwinge (P8) und Flügellänge. Hinter den $ -Zeichen ist jede einzelne Variable deklariert. Die Variable Land ist ein Faktor mit 5 Kategorien. Die Variablen Alter und Geschlecht hat R als Integer (Ganzzahl) und nicht als Faktoren mit 4 bzw. 3 Kategorien interpretiert (In der Variablen Alter bezeichnet eine 3 einen diesjährigen , eine 4 einen nicht diesjährigen, eine 5 einen vorjährigen und eine 0 einen Vogel unbestimmten Alters, in der Variablen Geschlecht eine 1 ein Männchen, eine 2 ein Weibchen und eine 0 alle Vögel mit unbestimmtem Geschlecht). Wir können jeder Variablen den Variablentyp aber leicht selbst zuordnen:

```
dat$Alter<-factor(dat$Alter)
dat$Geschlecht<-factor(dat$Geschlecht)
```

Betrachten Sie die Struktur der Datentabelle jetzt noch einmal mit str(dat)! Mit levels(dat$Geschlecht) können Sie sich die verschiedenen Kategorien einer faktoriellen Variablen ausgeben lassen. Tipp: Mit der Pfeil nach oben (↑) - Taste können vergangene Befehlszeilen abgerufen, dann ggf. verändert und noch einmal ausgeführt werden.

Variablen und Beobachtungen ansprechen, Daten sortieren

R-Code sollte immer (!) so viel wie möglich kommentiert werden, damit man ihn auch später noch versteht. Für Kommentare dient uns wieder das #-Zeichen. Text dahinter wird von R ignoriert und nicht ausgeführt. Tippen Sie folgende Zeilen in Ihren R-Editor (oder in ein Tinn-R-File) ein und führen Sie den Code zeilenweise auf R aus. Verstehen Sie genau, was R macht und wie R die einzelnen Zeichen interpretiert?

```
dat$Gewicht          # Variable Gewicht ansprechen
dat[,4]              # dasselbe über Indizierung
dat$Gewicht[5]       # Gewicht der 5. Beobachtung ansprechen
dat[5,4]             # dasselbe über Indizierung
```

Zur Erinnerung: Mit dem Dollarzeichen ($) können die einzelnen Variablen eines Data-Frames aufgerufen werden. Zusätzlich können mit eckigen Klammern die genauen Positionen innerhalb eines Vektors oder innerhalb eines Data-Frames angesprochen werden, wobei innerhalb der Klammern genau so viele Elemente, durch Komma getrennt, erwartet werden, wie das Objekt Dimensionen hat, also 1 für Vektoren und 2 für Data-Frames (probieren Sie dim(dat) für unsere Tannenmeisen-Daten!). Bei zweidimensionalen Data-Frames steht die erste Position innerhalb der eckigen Klammern für die Zeilennummern, die zweite für die Spaltennummern. Steht keine Zahl an der betreffenden Stelle, so bedeutet dies „alle Zeilen“ bzw. „alle Spalten“. Mit dem Doppelpunkt kann man einen Wertebereich angeben.

```
dat$Gewicht[5] <- 99                        # Gewicht der 5. Beobachtung ändern
dat$Gewicht == 99                           # Vektor, der alle Beobachtungen mit dem Wert 99 als “TRUE”, alle anderen als “FALSE” kennzeichnet
dat$Gewicht[dat$Gewicht == 99]<-9.5         # Änderung für alle Werte mit Gewicht = 99 wieder rückgängig machen
dat[3:8,]                                   # gibt den dritten bis achten Datensatz aus
```

Die logischen Operatoren == (ist gleich), != (ist nicht gleich), <=, <, >=, > sowie is.na() erzeugen logische Werte (TRUE, FALSE), die für die Auswahl von Beobachtungen innerhalb eines Data-Frames verwendet werden können. Die Funktion is.na() prüft, ob Werte als fehlend gekennzeichnet sind. Probieren Sie is.na(daten).

```
dat$Gewicht[dat$Geschlecht == “1”]          # gibt nur die Gewichte der Männchen aus
```

Da wir die Variable Geschlecht oben als Faktor definiert haben, behandelt R die Einträge dieser Variablen als Zeichenkette (String), deshalb schreiben wir die „1“ in Anführungszeichen. Wenn wir die Anführungszeichen vergessen, wählt R die im Alphabet zu vorderst liegende Stufe aus. Da in diesem Beispiel die Stufen „1“ und „2“ heißen, wäre die alphabetisch erste Stufe ebenfalls „1“, also die Männchen. Hätten wir jedoch anstelle von „1“ und „2“ „M“ und „F“ (für „males“ und „females“) genommen, dann würde mit dem Befehl dat$Gewicht[dat$Geschlecht == 1] (ohne Anführungszeichen) die Weibchen ausgewählt. Diese Anführungszeichengeschichte erscheint auf den ersten Blick als Schikane, ist aber sehr sinnvoll. Sie stellt sicher, dass die Codes von Faktorenstufen nicht als Zahlen behandelt werden (und z. B. unsinnigerweise 2 * Männchen = Weibchen gerechnet wird). Zudem helfen die Anführungszeichen uns, unseren eigenen Code besser zu verstehen: „1“ steht für eine Buchstabenfolge, hier „Männchen“, 1 bedeutet hingegen die Zahl 1.

```
dat$Geschlecht[dat$Fluegel >= 65]   # gibt das Geschlecht für alle Individuen mit mindestens 65 mm Flügellänge
which(dat$Fluegel >= 65)            # gibt den Index für alle Individuen mit mindestens 65 mm Flügellänge
```

which() ist eine Funktion. Deswegen wird das Argument in runden Klammern übergeben.

```
sort(dat$Gewicht)                                   # sortiert die Variable Gewicht in aufsteigender Reihenfolge
order(dat$Gewicht)                                  # Gibt an, wie die Werte in einem Vektor sortiert sind
datsort<-dat[order(dat$Geschlecht, dat$Alter),]     # sortiert das Datenfile zuerst nach Geschlecht, dann nach Alter
datsort                                             # das neu erzeugte Datenfile betrachten
```

t-Test und U-Test

Nachdem wir uns mit der Eingabe, dem Einlesen und dem Ansprechen von Daten befasst haben, können wir uns einfachen statistischen Tests zuwenden. Uns interessiert, ob sich die Flügellängen männlicher und weiblicher Tannenmeisen unterscheiden. Wir betrachten die Flügellängen beider Geschlechter zuerst grafisch in einem Boxplot (man sollte sich seine Daten zunächst immer anschauen, bevor man sich an statistische Tests wagt).

```
boxplot(Fluegel ~ Geschlecht, data = dat)
```

ergibt einen Boxplot mit den drei Kategorien der Variablen Geschlecht (Männchen, Weibchen, unbestimmt). Wir möchten aber die Kategorie Geschlecht = 0 (unbestimmt) nicht abbilden und weitere Verbesserungen am Plot vornehmen (Abb. 2).

```
dat$Geschlecht[dat$Geschlecht == 0] <- NA  # Nullen durch "NA" ersetzen
dat$Geschlecht <- factor(dat$Geschlecht)   # Der Faktor Geschlecht muss neu definiert werden, um die NAs
                                           auszuschließen (s.u.).
boxplot(Fluegel ~ Geschlecht, data = dat, col=c("lightgray","darkgray"), xaxt="n", ylab="Flügellänge (mm)", las=1)
axis(1, at=c(1,2), labels=c("Männchen", "Weibchen"))
```

Der Funktion werden wieder verschiedene Argumente übergeben: Mit Fluegel ~ Geschlecht sagen wir R, dass die Flügellänge gegen das Geschlecht aufgetragen werden soll, mit data wird das zu verwendende Data-Frame bezeichnet und mit col die Farbe der Boxen im Plot. Die Namen aller verfügbaren Farben erhält man mit colours(). xaxt="n" verbietet Werteangaben an der x-Achse, während sich mit ylab die Achsenlegende für die y-Achse vorgeben lässt. las=1 legt schließlich fest, dass die Werte an der y-Achse horizontal ausgerichtet sind. Mit Hilfe der Funktion axis() wird anstelle der Werte 1 und 2 der Klartext „Männchen" und „Weibchen" mittels labels=c("Männchen", "Weibchen") geplottet. Das Argument at=c(1,2) bestimmt, dass dies bei den x-Werten 1 und 2 geschehen soll.

?par gibt einen Überblick über die sehr umfangreichen Möglichkeiten zur vielfältigen Gestaltung von Grafiken in R (für Einsteiger allerdings eher verwirrend). An dieser Stelle weisen wir auch gerne auf Murell (2006) hin (siehe Literaturbesprechung unten). Um die Grafik beispielsweise als Bitmap zu speichern, brauchen wir

```
savePlot(filename="Pfadname/Plotname",
type="bmp")
```

Anstelle von „Pfadname/Plotname" setzen Sie wieder Ihre eigenen Präferenzen. Unter Windows kann man die Grafik auch einfach aus dem Datei-Menü des Grafik-Fensters heraus abspeichern. Die Qualität der so erstellten Grafiken ist in dieser Form allerdings oft nicht für Publikationen geeignet. Mit den Funktionen bmp(), jpeg(), png() und pdf() lassen sich qualitativ hochwertige Grafiken beliebiger Größe erstellen. Die erforderlichen Argumente können Sie wieder in der Hilfe nachlesen, z. B. mit ?bmp. Leider können solche Grafiken nicht direkt in R betrachtet werden. Sie werden im Hintergrund gespeichert und können mit einem Grafikprogramm (GIMP, Photoshop, Photoeditor oder ähnl.) geöffnet und ggf. bearbeitet werden.

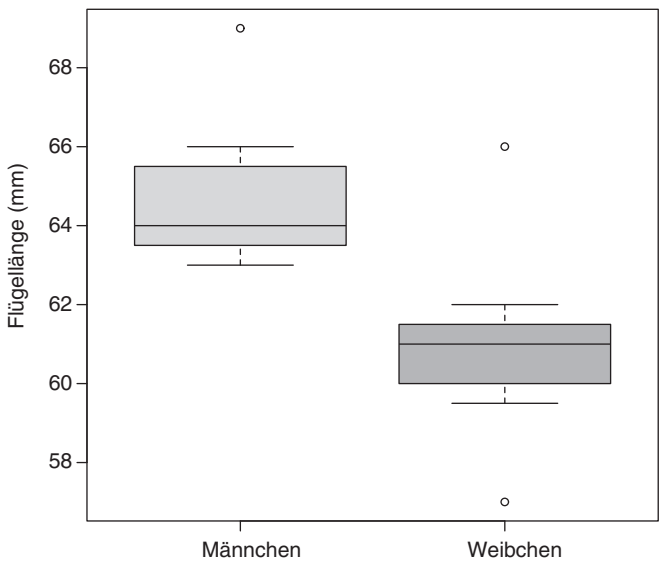

Abb. 2: Nach Geschlecht getrennter Boxplot der Flügellängen.

Aufgrund des hier sehr eindeutigen Boxplots erübrigt sich im Grunde ein Test, um die Frage nach Unterschieden zwischen den Geschlechtern zu klären. Zu Demonstrationszwecken führen wir ihn trotzdem durch. ?t.test ruft die dazugehörige Hilfefunktion auf.

t.test(Fluegel~Geschlecht, data=dat) oder auch t.test(dat$Fluegel~dat$Geschlecht)

R ist so eingestellt, dass es per Voreinstellung annimmt, die Varianzen in beiden Gruppen seien unterschiedlich. Deshalb wird die Welch-Näherung des t-Wertes inklusive Korrektur der Freiheitsgrade berechnet (Zar 1996, Sachs & Hedderich 2006). Ist man sicher, dass die Varianzen der beiden Gruppen gleich sind, dann kann mit dem Argument var.equal=TRUE die Welch-Korrektur ausgeschalten werden. Um zu überprüfen, ob die Verteilung der Daten die Anwendung eines t-Tests erlauben (Die Residuen, d. h. die Differenzen jeder einzelnen Beobachtung zum betreffenden Gruppen-Mittelwert, müssen normalverteilt sein), müssen wir zunächst die

Verteilung der Beobachtungen pro Gruppe betrachten. Dazu berechnen wir die Residuen der Ein-Weg-Varianzanalyse mit Fluegel als abhängiger Variable und Geschlecht als erklärender Variable (die hier angewandte Funktion lm() wird weiter unten erläutert) und stellen ihre Verteilung in einem Q-Q-Plot (Quantile-Quantile-Plot) der Normalverteilung gegenüber. Im Q-Q-Plot werden die Quantile zweier Verteilungen gegeneinander aufgetragen. Quantile sind jene Werte, welche die Daten in bestimmte Anteile unterteilen, z. B. liegt ein Viertel der Daten unterhalb des 25 %-Quantils und der Median entspricht dem 50 %-Quantil. Wenn zwei Verteilungen identisch sind, besitzen sie gleiche Quantile. Im Q-Q-Plot liegen in diesem Falle alle Punkte auf einer Geraden. Die Funktion qqnorm() stellt die Verteilung beliebiger Daten der Normalverteilung gegenüber.

```
qqnorm(resid(lm(Fluegel~Geschlecht, data=dat)))
qqline(resid(lm(Fluegel~Geschlecht, data=dat)))        # fügt die Gerade als optische Hilfe ein
```

Abgesehen von zwei Ausreißern passt die Normalverteilung nicht schlecht. Allerdings ist bei einer so kleinen Stichprobe die Beurteilung der Normalverteilung schwierig. Deshalb führen wir besser den nicht-parametrischen Wilcoxon Rangsummentest (Mann-Whitney U-Test) durch:

wilcox.test(Fluegel~Geschlecht, data=dat) oder auch wilcox.test(dat$Fluegel~dat$Geschlecht)

Die ausgegebene Warnung bedeutet, dass wegen der vielen verbundenen Werte (d. h. mehrfach vorhandene Werte) kein exakter p-Wert berechnet werden kann. Mit table(dat$Fluegel) kann man leicht ermitteln, dass unter 28 Werten der Wert 64 viermal, andere Werte zwei- oder dreimal vorkommen. Nur 5 Werte kommen nur einmal vor. Unter diesen Bedingungen muss der p-Wert als grobe Schätzung angesehen werden, d. h. wenn der p-Wert nahe bei 0,05 liegt, sollte das Testresultat als „Grenzfall“ deklariert werden.

Lineare Modelle, Modellwahl und Überprüfung der Modellannahmen

Häufig interessiert uns der gleichzeitige Einfluss mehrerer erklärender Variablen, z. B. ob neben dem Geschlecht auch das Alter, die Herkunft und das Gewicht der Tannenmeise mit der Flügellänge korrelieren. Dazu erstellen wir ein lineares Modell mit der Flügellänge als abhängiger Variable. Geschlecht, Alter und Land gehen als erklärende (nominale, nicht kontinuierliche) Variablen ins Modell ein, Gewicht als (kontinuierliche) Kovariable. Der Faktor Alter muss vorher noch aufbereitet werden, d. h. es müssen die Faktorstufen 4 und 5 in eine Faktorstufe vereint und Werte mit der Faktorstufe 0 als fehlende Werte markiert werden.

```
dat$Alter[dat$Alter==5] <- 4          # Alter “vorjährig” (5) und “nicht diesjährig” (4) in einer Stufe (4) “alt”
                                        zusammenfassen
dat$Alter[dat$Alter==0] <- NA         # Alter 0 (unbestimmt) als solches markieren
dat$Alter <- factor(dat$Alter)        # Den Faktor Alter neu als Faktor definieren
```

Wenn wir, wie soeben beschrieben, in einem Faktor Stufen verändern, dann sollte danach der Faktor neu definiert werden, damit R die alten Stufen (hier „0“ und „5“) vergisst. Wird diese Neudefinition ausgelassen, behält R die alten Stufen bei und lässt sie z. B. im Boxplot auf der x-Achse in unerwünschter Weise wieder auftauchen.

Die Funktion lm() steht für „linear model“. Sie erstellt aus den Daten und der Modellspezifikation ein Objekt der Klasse „lm“. Wichtig ist, dass die Variablen im Data-Frame richtig deklariert worden sind, also Faktoren als Faktoren und numerische Variablen als solche. Dies lässt sich ggf. wieder mit der Funktion str(dat) überprüfen.

```
mod1<-lm(Fluegel~Geschlecht+Alter+Geschlecht:Alter+Gewicht+Land, data=dat)        # volles Modell mit allen
                                                                                    erklärenden Variablen
```

Links der Tilde „ ~“ steht die abhängige Variable, rechts sind die erklärenden Variablen aufgelistet. Dabei bedeutet ein „+“ einen additiven Effekt und ein „:“ eine Interaktion. Der Ausdruck „Geschlecht+Alter+Geschlecht:Alter“ könnte auch in Kurzform geschrieben werden: „Geschlecht*Alter“. Das „*“ bedeutet, dass sowohl die Haupteffekte als auch die Interaktion als erklärende Variablen im Modell erscheinen. Die arithmetischen Zeichen „+“, „*“, „:“ sowie „-“ (siehe unten) erhalten also in der Formel zur Spezifikation eines Modells andere Funktionen als wenn sie außerhalb der Modellformel verwendet werden. Weitere Möglichkeiten, Modelle zu spezifizieren, finden Sie in Tabelle 9.3. in Crawley (2007).

Um Objekte der Klasse „lm“ zu betrachten, gibt es verschiedene Möglichkeiten:

anova(mod1) erstellt die Tabelle der Varianzanalyse:

```
Response: Fluegel
                    Df     Sum Sq    Mean Sq    F value       Pr(>F)
Geschlecht           1     54.158     54.158    20.8176    0.0008131  ***
Alter                1      6.517      6.517     2.5051    0.1417815
Gewicht              1      5.821      5.821     2.2373    0.1628401
Land                 3     34.987     11.662     4.4828    0.0274361    *
Geschlecht:Alter     1      0.059      0.059     0.0225    0.8834263
Residuals           11     28.617      2.602
```

Die Variablen werden hierbei hierarchisch getestet, d. h. der p-Wert in der Spalte Pr (>F) der obersten Variablen sagt, ob die Variable verglichen mit dem Mittelwert aller Werte einen Erklärungswert besitzt. Die zweitoberste Variable wird getestet, indem das Modell mit erster und zweiter Variable gegen das Modell mit nur der ersten Variablen getestet wird, usw. Demzufolge sind die p-Werte von der Reihenfolge der Variablen im Modell abhängig!

Die Funktion drop1() testet hingegen die einzelnen Variablen in einem Modell unabhängig von ihrer Reihenfolge:

drop1(mod1, test="F")

```
Single term deletions

Model:
Fluegel ~ Geschlecht * Alter + Gewicht + Land
                Df Sum of Sq    RSS    AIC F value   Pr(F)
<none>                       28.617 23.782
Gewicht          1     0.425 29.042 22.062  0.1635 0.69373
Land             3    34.961 63.578 32.949  4.4796 0.02749 *
Geschlecht:Alter 1     0.059 28.676 21.821  0.0225 0.88343
```

Für alle außer den an Interaktionen beteiligten Variablen wird ein F-Test durchgeführt. Effekte von Variablen, die an Interaktionen beteiligt sind, werden durch die Interaktion verfälscht. Deshalb gibt drop1() für solche Variablen (hier Geschlecht und Alter) kein Testergebnis aus.

Jede Variable wird geprüft, indem das volle Modell gegen das Modell ohne die betreffende Variable getestet wird. Das Argument test="F" stellt sicher, dass diese Modellvergleiche mittels F-Test durchgeführt werden. Dabei wird getestet, ob das Verhältnis der Residuenvarianz des kleineren zu jener des größeren Modells signifikant größer als 1 ist. Ist der Test signifikant, dann bedeutet das, dass die betreffende Variable einen signifikanten Anteil jener Varianz erklärt, die nicht schon durch die anderen Variablen im Modell erklärt wird. Wird das Argument test in der Funktion drop1() weggelassen, dann erscheinen die letzten beiden Spalten im Output nicht. Diese Einstellung animiert uns dazu, den AIC-Wert anstelle der F-Tests für die Modellwahl zu benutzen. Der AIC-Wert wird aus der Likelihood (= Wahrscheinlichkeit der Daten unter dem entsprechenden Modell) und der Zahl der im Modell verwendeten Parameter berechnet: AIC = -2 × log(Likelihood) + 2 × (Anzahl Parameter). Je kleiner der AIC-Wert ist, desto besser „passt“ das Modell. Dem Output oben entnehmen wir, dass der AIC-Wert des vollen Modells (Zeile „none“ = keine Variable wurde entfernt) 23,8 ist. Wird die Interaktion Geschlecht:Alter entfernt, dann sinkt der AIC-Wert auf 21,8, was bedeutet, dass das Modell ohne Interaktion besser ist als das Modell mit Interaktion. Umgekehrt steigt der AIC-Wert auf 32,9, wenn die Variable Land entfernt wird, womit wir sie im Modell belassen würden. Eine Modellwahl basierend auf Informationskriterien wie dem AIC hat gegenüber Hypothesetests (hier F-Test) den Vorteil, dass sie unabhängig von einem willkürlich gewählten Signifikanzniveau (meist 0,05) ist. Letzteres wird fragwürdig, wenn - wie oft während einer Modellwahl - viele Tests hintereinander durchgeführt werden. Eine Einführung in die Informationstheorie sowie eine Zusammenstellung ihrer Vor- und Nachteile finden Sie in Anderson et al. (2000) und sehr verständlich aufbereitet im Kapitel 2 von McCarthy (2007).

summary(mod1) fasst unser Modell zusammen:

```
Call:
lm(formula = Fluegel ~ Geschlecht * Alter + Gewicht + Land, data = dat)

Residuals:
       Min         1Q     Median         3Q        Max
-3.633e+00 -4.533e-01  2.098e-16  6.496e-01  2.051e+00

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        66.2428     6.0674  10.918 3.05e-07 ***
Geschlecht2        -3.6278     2.5493  -1.423   0.1824
Alter4              0.4879     1.1342   0.430   0.6754
Gewicht             0.2316     0.5727   0.404   0.6937
LandBulgarien      -1.6729     2.0301  -0.824   0.4274
LandRussland       -5.2916     2.3385  -2.263   0.0449 *
LandSchweiz        -4.9812     1.7453  -2.854   0.0157 *
Geschlecht2:Alter4  0.4273     2.8473   0.150   0.8834
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.613 on 11 degrees of freedom
  (9 observations deleted due to missingness)
Multiple R-squared: 0.7801,     Adjusted R-squared: 0.6402
F-statistic: 5.576 on 7 and 11 DF,  p-value: 0.006083
```

Wir erhalten die Schätzungen jedes einzelnen Parameters inklusive Standardfehler und t-Tests. Die Effekte von Geschlecht und Alter interpretieren wir erst unten, weil sie in diesem Modell durch die Interaktion Geschlecht:Alter „verfälscht“ sind, bzw. sie gelten jeweils nur für die Referenzstufe des anderen Faktors (der hier berechnete Geschlechtseffekt gilt für diesjährige und der Alterseffekt für die Männchen). Die Effekte der Variablen, die nicht an der Interaktion beteiligt sind, dürfen interpretiert werden: Je schwerer ein Vogel ist, desto länger ist sein Flügel (Steigung 0,23). Per Voreinstellung wird für Faktoren die erste Stufe (Geschlecht = 1, also Männchen, Alter = 3, also diesjährig und Land = Algerien) auf Null gesetzt und für jede weitere Stufe die Differenz zur ersten Stufe geschätzt. Vögel aus Russland („LandRussland“) und der Schweiz („LandSchweiz“) haben signifikant um 5,29 bzw. 4,98 mm kürzere Flügel als die Vögel aus Algerien. Falls Sie die Kategorien einer faktoriellen Variablen vergessen haben sollten, oder nicht wissen, welches die erste Stufe ist, können Sie sich z. B. mit levels(dat$Land) schnell einen Überblick verschaffen, table(dat$Land) gibt zusätzlich die jeweiligen Häufigkeiten aus, was manchmal ganz hilfreich ist. In unserem Fall bedeutet das Ergebnis von table(dat$Land), dass die Daten ungeeignet sind, um Unterschiede zwischen den Ländern zu testen, da die meisten (20) Tannenmeisen aus der Schweiz stammen und nur je 2 aus den anderen Ländern. Entsprechend sollten signifikante Unterschiede zwischen den Ländern mit großer Vorsicht interpretiert werden (sie beruhen lediglich auf Messungen von 2 Tannenmeisen!).

Wir wollen das Modell nun vereinfachen, indem wir die nicht signifikante Interaktion zwischen Geschlecht und Alter herausnehmen.

```
mod2<-update(mod1, .~.-Geschlecht:Alter)
```

Die Funktion update() übernimmt aus dem Modell mod1 die abhängige Variable („.“ links der Tilde) sowie alle erklärenden Variablen („.“ rechts der Tilde), entfernt jedoch aus den erklärenden Variablen die Interaktion Geschlecht:Alter („-“ bedeutet hier „ohne“). Das um die Interaktion Geschlecht:Alter reduzierte Modell wird unter mod2 gespeichert. Nun möchten wir die beiden Modelle vergleichen, um zu entscheiden, welches der Modelle besser zu unseren Daten passt. Für diesen Vergleich können wir ebenfalls die Funktion anova() verwenden.

```
anova(mod1, mod2)
```

```
Analysis of Variance Table
Model 1: Fluegel ~ Geschlecht * Alter + Gewicht + Land
Model 2: Fluegel ~ Geschlecht + Alter + Gewicht + Land
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1     11 28.6170
2     12 28.6756 -1   -0.0586 0.0225 0.8834
```

Ein nicht signifikanter Unterschied zwischen den beiden Modellen bedeutet, dass die Interaktion zwischen Geschlecht und Alter keinen zusätzlichen Erklärungswert enthält. Also werden unsere Daten gut mit dem einfacheren Modell beschrieben. Da das Modell nun keine Interaktionen mehr enthält, können wir alle Haupteffekte unverfälscht testen:

```
drop1(mod2, test="F")
```

```
Single term deletions

Model:
Fluegel ~ Geschlecht + Alter + Gewicht + Land
          Df Sum of Sq    RSS    AIC F value    Pr(F)
<none>                  28.676 21.821
Geschlecht  1    42.985 71.661 37.223 17.9882 0.001145 **
Alter       1     0.877 29.552 20.393  0.3668 0.556030
Gewicht     1     0.396 29.072 20.081  0.1659 0.690946
Land        3    34.987 63.662 30.974  4.8803 0.019172 *
```

Wenn keine Interaktionen mehr im Modell enthalten sind, gelten alle Haupteffekte für den gesamten Datensatz und können direkt interpretiert werden. Im Prinzip ist es dann nicht nötig, weitere Variablen aus dem Modell zu entfernen, außer wenn die erklärenden Variablen untereinander korrelieren. Dann beeinflusst die eine erklärende Variable den Effekt der anderen im Modell. In solchen Fällen ist es empfehlenswert, jede Variable auch einzeln (in einem Modell ohne die mit ihr korrelierten Variablen) zu testen, um die Bandbreite der möglichen Aussagen zu explorieren. Zu einer eindeutigen Aussage wird man jedoch nicht kommen, da Effekte von korrelierten Variablen ohne zusätzliche Information mathematisch nicht getrennt werden können. Mehr zu diesem Thema finden Sie in Smith et al. (2009) und Hector et al. (2009).

Wir betrachten nun die Effekte von Geschlecht, Alter, Gewicht und Land auf die Flügellänge der Tannenmeisen:

```
summary(mod2)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-3.61349 -0.45702  0.01062  0.67082  2.01062

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    66.2497     5.8148  11.393  8.6e-08 ***
Geschlecht2    -3.2648     0.7698  -4.241  0.00114 **
Alter4          0.5721     0.9447   0.606  0.55603
Gewicht         0.2223     0.5457   0.407  0.69095
LandBulgarien  -1.6398     1.9341  -0.848  0.41312
LandRussland   -5.4371     2.0396  -2.666  0.02057 *
LandSchweiz    -4.9439     1.6557  -2.986  0.01136 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.546 on 12 degrees of freedom
  (9 observations deleted due to missingness)
Multiple R-squared: 0.7797,     Adjusted R-squared: 0.6695
F-statistic: 7.078 on 6 and 12 DF,  p-value: 0.002109
```

In der Spalte „Estimate" ist für numerische Variablen (hier Gewicht) die Steigung angegeben und, wie im vorigen Fall, für kategorielle Variablen der Unterschied zur ersten Stufe. Der Unterschied in der Flügellänge zwischen Männchen und Weibchen beträgt -3,3, d. h. die Weibchen haben im Durchschnitt einen um 3,3 mm kürzeren Flügel als die Männchen.

Wir fanden also signifikante Unterschiede in der Flügellänge zwischen den Geschlechtern und zwischen den Ländern. Bevor wir dieses Resultat publizieren, überprüfen wir, ob die Modellvoraussetzungen erfüllt sind. Dazu erstellen wir vier Abbildungen der Residuen. Dies ist bei allen linearen und nicht-linearen Modellen essenziell! R macht uns die Überprüfung sehr einfach:

```
par(mfrow=c(2,2))        # unterteilt Grafikfenster in 4 Unterfenster (2 Reihen und 2 Spalten)
plot(mod2)               # grafische Analyse der Residuen
```

Die Warnung

```
Warnmeldungen:
1: Not plotting observations with leverage one:  8
2: Not plotting observations with leverage one:  8
```

zeigt an, dass die 8. Beobachtung eine Hebelwirkung von 1 besitzt. Die Hebelwirkung ist ein relatives Maß dafür, wie weit eine Beobachtung vom Mittelwert der erklärenden Variablen entfernt ist. Beobachtungen mit hoher Hebelwirkung haben einen stärkeren Einfluss auf das Modell als Beobachtungen mit kleiner Hebelwirkung. Hebelwirkungswerte von 1 dürfen nicht auftreten. Im Datenfile (tippe dat in die R-Console), sehen wir, dass die 8. vollständige Beobachtung (Zeile 14) eine Tannenmeise aus Algerien ist. Weiter sehen wir, dass die zweite Tannenmeise aus Algerien von R stillschweigend (stimmt nicht ganz: im „summary output“ oben wird angegeben, wie viele Zeilen ignoriert werden) für die Modellanpassung ignoriert wird, weil für diese Tannenmeise keine Altersangabe vorhanden ist. Als Folge davon enthält das verwendete Datenfile nur noch eine Tannenmeise aus Algerien, die entsprechend den Mittelwert für Algerien zu 100 % bestimmt, was zur Hebelwirkung von 1 führt. Die Warnmeldung teilt uns mit, dass diese Beobachtung in den Residuenplots nicht abgebildet wird. Für den Moment ignorieren wir die Warnung, regen jedoch dazu an, die Analyse nochmals allein mit den schweizerischen Tannenmeisen zu wiederholen.

In den Residuenplots (Abb. 3) überprüfen wir folgende Modellvoraussetzungen:

1.) Linearer Zusammenhang (oben links): Besteht zwischen den erklärenden Variablen und der abhängigen, wie im Modell angenommen, ein linearer Zusammenhang, dann verläuft der Glätter (die durchgezogenen Linie) entlang der gestrichelten Null-Linie. In unserem Falle ist anscheinend ein linearer Zusammenhang vorhanden. Die Linie verläuft zwar nach rechts leicht nach unten, jedoch sind lediglich 2 Beobachtungen dafür verantwortlich. Entsprechend dürfte dieser Abfall zufällig sein.
2.) Normalverteilung der Residuen (oben rechts): Sind die Residuen normalverteilt, liegen sie auf einer Geraden. Das ist in unserem Beispiel zufrieden stellend gegeben. An den beiden Enden dürfen einzelne Beobachtungen von der Linie abweichen.
3.) Homogenität der Varianzen (unten links): Wenn die Varianz im ganzen Wertebereich, wie angenommen, gleich ist, verläuft der Glätter horizontal. Auf den ersten Blick scheint die Varianz mit zunehmenden Vorhersagewerten stark anzusteigen. Dieser Anstieg wird jedoch durch eine einzige Beobachtung (27) verursacht, entsprechend stufen wir die Verletzung der Annahme gleicher Varianzen als harmlos ein.
4.) Einfluss einzelner Beobachtungen (unten rechts): Besitzt eine Beobachtung eine Cooks Distanz (Definition siehe Legende zu Abb. 3) von über 1 (d. h. liegt sie im Plot außerhalb der 1er-„Höhenlinie“), dann hat diese Beobachtung einen substanziellen Einfluss auf das Modell. In unserem Beispiel liegt keine Beobachtung außerhalb der Cooks Distanzlinie 1 (ganz oben oder ganz unten rechts in der Abbildung), so dass keine Beobachtung einen sehr ernst zu nehmenden Einfluss auf das Modell ausübt (außer natürlich die nicht abgebildete 8. Beobachtung!). Allerdings erscheinen die Beobachtungen 26 und 27 in allen vier Plots als „Ausreißer“. Entsprechend würde es sich lohnen, den Einfluss dieser beiden Beobachtungen zu überprüfen.

Für speziellere lineare Modelle, wie zum Beispiel generalisierte lineare Modelle (GLM) oder gemischte Modelle (mixed models), oder für generalisierte additive Modelle (GAM), die auch nicht-lineare Zusammenhänge zulassen, müssen wir auf die weiterführende Literatur verweisen: Crawley (2007) beschreibt die am häufigsten verwendeten Modelltypen (normale lineare Modelle, generalisierte lineare Modelle, gemischte Modelle) sehr gut verständlich. Faraway (2006) gibt einen vertieften Einblick in die Möglichkeiten von linearen Modellen (inkl. z. B. multinomialer Modelle) und nicht-parametrische Regressionen. Gelman & Hill (2007) bieten einen gut verständlichen Einstieg in das moderne kreative Modellieren mit Schwerpunkt hierarchische (gemischte) Modelle. Generalisierte additive Modelle sind ausführlich in Wood (2006) beschrieben. Alle diese Bücher enthalten R-Codebeispiele.

Korrelation, Scatterplot und einfache Regression

Oft wird anstelle der Flügellänge die Länge der 8. Handschwinge gemessen (= Federlänge), da der persönliche Messfehler für dieses Maß geringer ist (Berthold & Friedrich 1979, Jenni & Winkler 1989). Wir möchten klären, ob die Federlänge wirklich die Flügellänge ersetzten kann und wenn ja, wie gut.

Zunächst hilft auch hier wieder sehr, sich die Datenverteilung grafisch in einem Scatterplot zu veranschaulichen. Nur wenn alle Voraussetzungen für eine Korrelation oder lineare Regression (linearer Zusammenhang, Residuen normalverteilt, Varianzen ändern sich mit den y-Werten nicht, vgl. z. B. Sachs & Hedderich 2006 oder Crawley

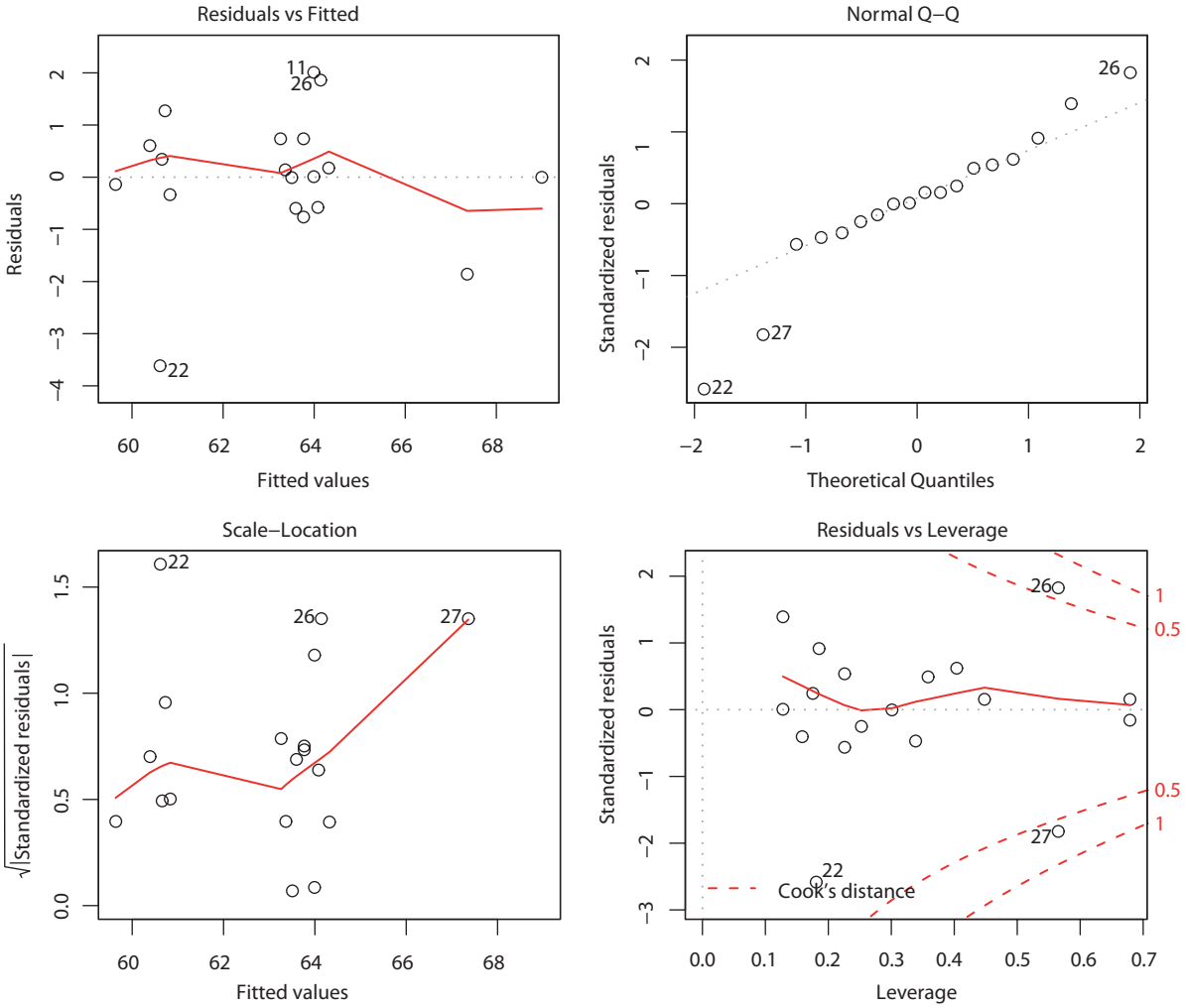

Abb. 3: Grafische Residuenanalyse für ein lineares Modell. 1.) Residuen gegen Vorhersagewerte (fitted values, oben links), 2.) Q-Q-Plot der Residuen (oben rechts), 3.) Wurzel des Absolutwertes der standardisierten Residuen gegen die Vorhersagewerte (unten links), 4.) Residuen gegen Hebelwirkung (leverage, unten rechts) inklusive Cooks Distanz („Höhenlinien" im Plot), welche die Größe der Residuen und die Hebelwirkung zu einem Maß für den Einfluss jeder einzelnen Beobachtung verrechnet.

2007), dürfen wir ein lineares Modell rechnen! Ansonsten müssten wir die Daten transformieren oder auf ein nicht-lineares Modell ausweichen.

```
plot(dat$Fluegel, dat$P8)
```

Bis auf eine einzige Beobachtung liegen die Punkte alle sehr eng um eine Gerade herum, d. h. die Voraussetzungen für eine lineare Regression sind anscheinend ansonsten erfüllt. Wir vermuten, dass für den Ausreißer ein Messfehler verantwortlich ist. Deshalb identifizieren wir diese Beobachtung interaktiv.

```
identify(dat$Fluegel, dat$P8)
```

Mit der linken Maustaste kann der Ausreißer in der Grafik angeklickt werden. Im Grafik-Fenster erscheint die Identifikationsnummer des Ausreißers, hier 22. Über die rechte Maustaste oder mit der Escape-Taste kann die Funktion gestoppt werden.

Wir können nun die Daten des Ausreißers mit `dat[22,]` anschauen.

Wir nehmen an, dass bei diesem Wert ein Mess- oder Tippfehler vorliegt und erstellen den Scatterplot ohne diese Beobachtung. In wissenschaftlichen Arbeiten ist das Weglassen von Beobachtungen ohne stichhaltigen Beweis für ihre Fehlerhaftigkeit tabu. In solchen Fällen können die sogenannten robusten Methoden helfen (Huber 1981):

```
plot(dat$Fluegel[-22], dat$P8[-22])
```

Nun können wir den Korrelationskoeffizienten mit dem dazugehörigen Signifikanzniveau unter Ausschluss des Ausreißers berechnen.

```
cor.test(dat$Fluegel[-22], dat$P8[-22], method="pearson")
```

```
        Pearson's product-moment correlation

data:  dat$Fluegel[-22] and dat$P8[-22]
t = 12.7878, df = 25, p-value = 1.816e-12
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.8533434 0.9685594
sample estimates:
      cor
0.9313401
```

Der Korrelationskoeffizient („cor“) ist sehr hoch, so dass wir annehmen dürfen, dass das Handschwingenmaß die Flügellänge relativ gut ersetzt. Mit method="spearman“ erhält man übrigens eine Rankkorrelation nach Spearman, die keinen linearen Zusammenhang und keine Normalverteilung der Residuen erfordert und zumindest für einen ersten Test auf einen Zusammenhang sehr hilfreich ist.

Wir möchten jetzt Flügellängen in das Handschwingenmaß umrechnen, d. h. wir möchten die Geradengleichung der Regression P8~Fluegel berechnen und diese im Scatterplot einzeichnen.

```
m <- lm(P8[-22]~Fluegel[-22], data=dat)      # Rechnet das Modell ohne Ausreißer
abline(m, col="blue")                        # Zeichnet eine blaue Regressionsgerade in den Plot
summary(m)                                   # Fasst das Modell zusammen
```

```
Call:
lm(formula = P8[-22] ~ Fluegel[-22], data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-1.02608 -0.49106 -0.02608  0.46175  1.37956

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.92191    3.68507   0.522    0.607
Fluegel[-22] 0.74125    0.05796  12.788 1.82e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6544 on 25 degrees of freedom
Multiple R-squared: 0.8674,     Adjusted R-squared: 0.8621
F-statistic: 163.5 on 1 and 25 DF,  p-value: 1.816e-12
```

Die Zusammenfassung liefert auch die Koeffizienten für die Regressionsgleichung („Intercept“ ist der Achsenabschnitt). Sie lautet demnach: P8 = 1,92 + 0,74 × Fluegel. Das Signifikanzniveau für die Abweichung der Steigung von Null ist übrigens identisch mit dem für den Korrelationskoeffizienten. Zuletzt überprüfen wir (routinemässig!) wieder grafisch, ob die Voraussetzungen für ein lineares Modell in diesem Fall erfüllt sind:

```
par(mfrow=c(2,2))        # unterteilt Grafikfenster in 4 Unterfenster (2 Reihen und 2 Spalten)
plot(m)                  # grafische Analyse der Residuen
```

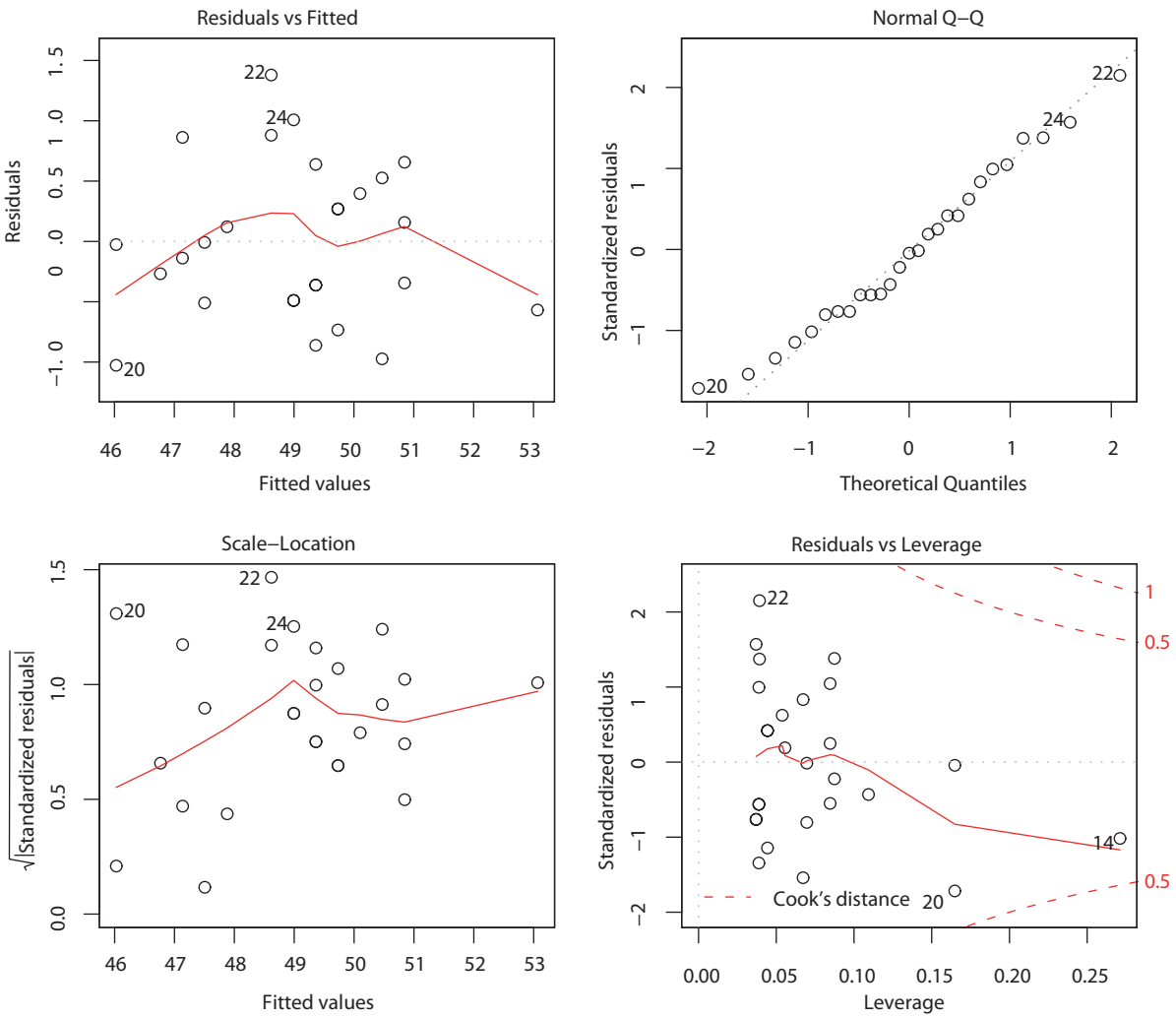

Abb. 4: siehe Legende zu Abb. 3.

Die Interpretation der Residuenplots (Abb. 4) lautet:

1.) Der lineare Zusammenhang ist einigermaßen erfüllt. Die Krümmungen nach unten an beiden Enden beruhen auf sehr wenigen Daten.
2.) Die Residuen sind perfekt normalverteilt (QQ-Plot).
3.) Die Residuen-Varianz steigt mit zunehmenden Federlängen leicht an. Da wir aber gesamthaft eher wenige Daten haben, könnte dieser Anstieg auch einfach Zufall sein. Wenn sich dieser Anstieg der Varianz bei größeren Datensets erhärtet, müsste man die zunehmende Varianz in der Regression z. B. durch Datentransformation berücksichtigen.
4.) Keine Beobachtung beeinflusst das Modell beunruhigend stark (alle Beobachtungen liegen innerhalb der Cooks Distanz von 0,5).

Berechnen und überprüfen Sie zum Vergleich ein Modell mit dem Ausreißer (Datensatz 22)!

Was aber ist zu tun, wenn eine oder mehrere der Annahmen nicht erfüllt sind? Je nach Art der Daten und Art des Modells bieten sich verschiedene Möglichkeiten, wie z. B. Transformationen, robuste Methoden, generalisierte lineare Modelle (GLM), gemischte Modelle (LME) und nicht-parametrische Methoden, an, die wir hier nicht erläutern können. Wir verweisen deshalb auf die Fachliteratur.

Empfohlene Literatur

Eine Vielzahl sehr guter Bücher liefert praktische Anleitungen für das Arbeiten mit R. Wir stellen hier eine Auswahl vor, die wir persönlich für R-Einsteiger sehr empfehlen.

„The R book“ (Crawley 2007) bietet eine sehr gut verständliche Einführung in die Sprache R und in die wichtigsten Methoden der angewandten Statistik. Crawley benutzt viele Beispiele aus der Biologie und liefert kochbuchartige Rezepte inklusive R-Code. Dieses Buch kann als Lehrbuch verwendet werden. Ein Lehrbuch in ähnlichem Stil, jedoch für Leser, die es kurz und bündig mögen, ist „Introductionary statistics with R“ (Dalgaard 2002).

Ligges (2008) befasst sich in „Programmieren mit R“ alleine mit der Sprache R. Er vermittelt darin ein tiefes Verständnis der Sprache und liefert einige Tipps für das effiziente Programmieren. Wer sich noch tiefer mit der Sprache R befassen möchte, dem sei das Buch „Software for data analysis. Programming with R“ von Chambers (2008) empfohlen. Eine große Programmierhilfe bietet auch „R graphics“ (Murrell 2006). Dieses Buch zeigt, wie Grafiken uneingeschränkt nach den eigenen Wünschen erstellt werden können. Zum Beispiel wird beschrieben, wie Bitmap-Bilder in R importiert und in eine Grafik eingebaut werden oder wie verschiedene Grafiken verschachtelt werden können. Leider ist der Index sehr spärlich ausgefallen, so dass oft das ganze Buch durchgeblättert werden muss, wenn eine spezifische Lösung gesucht wird. Es ist zu hoffen, dass dies bei einer eventuellen Neuauflage verbessert wird.

Für Leser, die gerne mit klassischen Statistikbüchern arbeiten, empfehlen wir „Angewandte Statistik, Methodensammlung mit R“ (Sachs & Hedderich 2006). In diesem Buch werden im Nachschlagewerk-Stil Rezepte für die gängigen statistischen Methoden präsentiert. In der zwölften, neuen Auflage wird erstmals zu jeder Methode der R-Code zum Abschreiben mitgeliefert.

Relativ kurze und bündige Einführungen in etliche moderne statistische Methoden bietet „Modern applied statistics with S“ (Venables & Ripley 2002). Die Sprache S ist praktisch identisch mit R (aber nicht frei erhältlich). Im Buch weisen Fußnoten auf allfällige Abweichungen hin. Zum Inhalt gehören lineare Modelle, generalisierte lineare Modelle, nicht-lineare und additive Modelle, Baum-Modelle (tree-based models), gemischte Modelle, multivariate Analysen wie Cluster-Analyse und Faktorenanalyse, Klassifizierungen wie Diskriminanzanalyse und neuronale Netzwerke, Überlebensanalyse, Zeitreihenanalyse, Räumliche Statistik und Optimisierungen.

Wer sich tiefer mit linearen Modellen befassen möchte, dem empfehlen wir die beiden Bücher von J. Faraway: „Linear Models with R“ (Faraway 2005) und „Extending the linear model with R“ (Faraway 2006). Sie sind sehr verständlich geschrieben, allerdings sei darauf hingewiesen, dass lange Listen mit Korrekturen während der Lektüre unbedingt beachtet werden müssen. Für die generalisierten additiven Modelle gibt es ein sehr gutes und verständlich geschriebenes Buch: „Generalized additive models. An introduction with R“ (Wood 2006). Neben einer fundierten Einführung in den theoretischen Hintergrund der additiven Modelle liefert Wood (2006) auch praktische Beispiele mit R-Codes dazu. Wenn verschiedene Einstellungsmöglichkeiten bestehen, z. B. für die Wahl der Glättungsart, sind diese in übersichtlichen Tabellen dargestellt, so dass das Buch nicht nur als Lehrbuch sondern auch als Nachschlagewerk dient.

In das schwierige Thema der gemischten Modelle mit R führt das Buch „Data analysis using regression and multilevel/hierarchical models“ (Gelman & Hill 2007) auf moderne Art und in verständlicher Weise ein. Gleichzeitig leiten uns die Autoren sanft an die Bayesianische Denkweise heran und ermutigen, auch unkonventionelle aber kreative Wege zu gehen. Letzteres wird gerade durch R vereinfacht.

Dank

Unser Dank gilt vor allem den Entwicklern von R sowie allen, die in vielfältiger Weise zu seiner beständigen Verbesserung, Erweiterung und Verbreitung beitragen. Franz Bairlein, Wolfgang Fiedler, Gudrun Hilgerloh und Ulrich Köppen machten wertvolle Anmerkungen zu einer früheren Version dieses Textes. Nicht zuletzt sei auch den Teilnehmer/innen an den R-Kursen auf den Jahresversammlungen der Deutschen Ornithologen-Gesellschaft gedankt, die unsere Aufmerksamkeit auf spezielle Problemfelder bei der Anwendung von R lenkten.

Zusammenfassung

Die Publikation ornithologischer Daten setzt heute voraus, dass sie mit angemessenen statistischen Methoden ausgewertet werden. Mit der Entwicklung entsprechender Verfahren steigen auch die Ansprüche an die Auswertungen und an die Software, die für solche Auswertungen nutzbar ist. Anerkannte kommerzielle Statistiksoftware ist für den Normalverbraucher oft unerschwinglich teuer. Das freie Statistikpaket R bietet eine kostenlose, aber doch professionelle Lösung. Leider ist der Einstieg in R nicht einfach, da das Programm nicht geklickt werden kann, sondern Code geschrieben werden muss. In diesem Artikel bieten wir eine Einstiegshilfe. Wir zeigen Schritt für Schritt, wie mit R gearbeitet wird. Der Leser kann direkt am eigenen Computer nachvollziehen, wie Daten in das R eingelesen werden, wie diese angesprochen und dargestellt werden. Wir begleiten den Leser durch einen t-Test und führen ein einfaches lineares Modell inklusive Residuenanalyse durch. Abschließend geben wir Empfehlungen für weiterführende Bücher.

Literatur

Anderson DR, Burnham KP & Thompson WL 2000: Null hypothesis testing: Problems, prevalence, and an alternative. J. Wildl. Manage. 64: 912-923.

Berthold P & Friedrich W 1979: Die Federlänge: Ein neues nützliches Flügelmaß. Vogelwarte 30: 11-21.

Chambers JM 2008: Software for data analysis. Programming with R. Springer, New York.

Crawley MJ 2007: The R book. John Wiley & Sons, Chichester.

Dalgaard P 2002: Introductory statistics with R. Springer, Berlin.

Faraway J 2005: Linear models with R. Chapman & Hall, Boca Raton

Faraway J 2006: Extending the linear model with R. Chapman & Hall, Boca Raton

Gelman A & Hill J 2007: Data analysis using regression and multilevel/hierarchical models. Cambridge University Press, Cambridge.

Hector A, von Felten S & Schmid B 2009: Analysis of variance with unbalanced data: an update for ecology & evolution. J. Anim. Ecol. 79: 308–316.

Huber PJ 1981: Robust statistics. John Wiley & Sons, New York.

Jenni L & Winkler R 1989: The feather-length of small passerines: a measurement for wing-length in live birds and museum skins. Bird Study 36: 1-15.

Korner-Nievergelt F & Leisler B 2004: Morphological convergence in conifer-dwelling passerines. J. Ornithol. 145: 245-255.

Korner-Nievergelt F, Hüppop O & Schmaljohann H 2007: Einführung in das freie Statistikpaket R. Vogelwarte 45: 373.

Ligges U 2008: Programmieren mit R. 3. Aufl., Springer, Berlin.

McCarthy M 2007: Bayesian methods for ecology. Cambridge University Press, Cambridge, New York u. a.

Murrell P 2006: R graphics. Chapman & Hall/CRC, Boca Raton.

R Development Core Team 2010: R: A language and environment for statistical computing. R Foundation for Statistical Computing. Wien. http://www.R-project.org.

Sachs L & Hedderich J 2006: Angewandte Statistik. Methodensammlung mit R. Springer, Berlin.

Smith AC, Koper N, Francis CM & Fahrig L 2009: Confronting collinearity: comparing methods for disentangling the effects of habitat loss and fragmentation. Landscape Ecology 24: 1271-1285.

Venables WN & Ripley BD 2002: Modern applied statistics with S. Springer, Berlin.

Wood S 2006: Generalized additive models. An introduction with R. Chapman & Hall/CRC, Boca Raton.

Zar JH 1996: Biostatistical analysis. Prentrice Hall Inc., London.