Angewandte Statistik für die biologischen Wissenschaften

2., durchgesehene, aktualisierte, überarbeitete und erweiterte Auflage

Dr. Carsten F. Dormann Dr. Ingolf Kühn
Helmholtz Zentrum für Umweltforschung-UFZ

13. November 2012

Inhaltsverzeichnis

Vorwort

Disclaimer

Beim Verfassen dieses Textes haben die Autoren großen Wert auf Verständlichkeit gelegt. Sollten sich dabei Ungenauigkeiten und mathematische Inkorrektheiten eingeschlichen haben, so tut uns das leid. Wir weisen deshalb hier darauf hin, dass wir keine Garantie für die Korrektheit aller beschriebener Verfahren geben können.

Die hier vorliegenden Seiten sind aus einem Vorlesungsskript erwachsen. Sie sind notwendigerweise etwas unvollständig, und u.U. sogar inkorrekt (Tippfehler, Missverständnisse, Inkompetenz unsererseits). Die Arbeit an diesem Text verteilte sich gemäß univariater und multivariater Statistik: für erstere zeichnet sich CFD, für letztere IK verantwortlich. Über Hinweise auf Fehler und fehlende wichtige Themengebiete wären wir sehr dankbar! Bitte per Email an: `carsten.dormann@ufz.de`. Dank für entsprechende Kommentare und Korrekturvorschläge geht an Jan Hanspach, Ingo Holz, Ronny Goldberg, Felix May, Ulrike Obertegger, Armin Schmitt, Karl-Manfred Schweinzer und Annegret Werzner!

Ebenso wären wir dem werten Leser dankbar, wenn er/sie bei Benutzung dieses Skripts für Lehre oder in der Anwendung auf unsere Urheberschaft hinweist ...

Herzlichen Dank!

Die Natur biostatistischer Daten und die damit einhergehenden didaktischen Probleme

Kein anderer Zweig naturwissenschaftlicher Forschung trifft so häufig, so unausweichlich und vor allem so schlecht ausgebildet auf komplexe Daten wie die Biologie. Nahezu jede erdenkliche Verteilung, unmögliche Versuchsdesigns, Abhängigkeiten und Messergebnisse sind hier die Norm. Entsprechend greifen viele einführende Bücher der Statistik viel zu kurz, wenn es um die Situationen geht, in denen sich Biologen häufig finden. Insbesondere drei Themenfelder sind im allgemeinen nur sehr unzureichend abgedeckt: *resampling*-Verfahren (*bootstrap*, Monte Carlo-Simulationen), nicht-normalverteilte Antwortvariablen (verallgemeinertes lineares Modell, *generalised linear model*) und die saubere Einbeziehung des Versuchsdesigns in die Auswertung (feste/zufällige Effekte; *split-plot* und *nested* Designs).

Hier versuchen wir, auch diesen Fällen Rechnung zu tragen. Für einen Biologen wird es dabei kaum möglich/nötig sein, die mathematischen Hintergründe genau zu verstehen. Wichtiger ist unser Ansicht nach ein Verständnis der Prinzipien und ihre Umsetzung in einer Software. An mancher Stelle werden auch hier Formeln auftreten, die das Verständnis des ein oder anderen Lesers übertreffen. Dies ist zumeist nicht so schlimm. Wir haben versucht, auf die wichtigen Elemente besonders (und wiederholt) hinzuweisen, so dass die Formeln eigentlich vor allem zum Nachschlagen oder fortgeschrittenen Verständnis dienen sollen.

Ein Problem didaktischer Natur, dass wir nicht zu lösen vermochten, ist das der hierarchische Aufbau der parametrischen Statistik und seine geschichtliche Entwicklung nicht mit einem sukzessiven Aufbau eines Lehrbuches einhergehen. Ein Beispiel: Regression und ANOVA sind von einer höheren Warte aus gesehen obsolete Ausdrücke. Beide sind Spezialfälle eines linearen Modells. Trotzdem können wir nicht mit einem linearen Modell anfangen, denn dies ist in seiner Formulierung zu mathematisch-abstrakt, um als Einstieg zu dienen. Entsprechend folgen wir hier im Aufbau der klassischen Route, die als Nachteil eine umständliche Nomenklatur mit sich bringt. Vergleichbar ist dies mit der Genetik und der Taxonomie. Heutzutage werden zwar alle modernen Stammbäume aufgrund genetischer Erkenntnisse aufgebaut, und

Artengruppen („Würmer“, „Grasmücken“) sind dabei häufig anders eingeteilt, als die deutsche oder lateinische Namensgebung vermuten lässt. Andererseits ist es sehr schwierig sich vorzustellen, wie ein Student im ersten Semester mit ATCGATAC-Sequenzen an die Taxonomie herangeführt werden kann. Also müssen wir beides lernen: die deutschen/lateinischen Namen fürs Grobe, die Genetik fürs Spezielle.

„Statistik kann ich nicht.“

Statistik ist eine hügelige Landschaft, in der man meistens nur geringe Teile überblicken kann, und in der man sich jeden Hügel, jeden Berg auf Neue erobern muss. Statistisches Denken ist niemandem in die Wiege gelegt, aber trotzdem für alle zugänglich. Die wenigen Grundgedanken statistischer Tests etwa, die wir hier einfach hinschreiben könnten, helfen uns nur im Zusammenhang mit einer konkreten Fragestellung weiter. Wie Erich Fromm in seinem Buch „Die Kunst des Liebens“ schreibt, bräuchte er eigentlich nur eine Seite, aber er weiss, dass der Leser nicht genug über diese eine Seite nachdenken würde. Deshalb hat er jeden Gedanken dieser einen Seite ausgeführt und so oft wiederholt, wie er glaubt, dass der Leser sich damit beschäftigen sollte. Ganz so schlimm soll es hier nicht werden. Nichtsdestotrotz werden einzelne Aussagen wiederholt auftreten, als statistisches Mantra sozusagen.

Weiterführende und ergänzende Literatur

Von den deutschen Bücher zum Thema Statistik wollen wir vier hervorheben: (1) Werner A. Stahel: Statistische Datenanalyse – eine Einführung für Naturwissenschaftler (Vieweg, Braunschweig, 1995, 359 Seiten). Das Wort Einführung ist angemessen, denn es geht bis ANOVA und sogar etwas Zeitreihenanalyse, berührt aber nicht GLM oder kompliziertere Versuchsdesigns. (2) Joachim Werner: Lineare Statistik. Das Allgemeine Lineare Modell (Beltz, PsychologieVerlagsUnion, Weinheim, 1997, 633 Seiten). Dieses Werk ist sehr gründlich und mit SAS-Beispielen durchsetzt, behandelt aber nur, wie im Titel angekündigt, das lineare Modell, ohne seine Verallgemeinerung und ohne nicht-parametrische Verfahren. (3) Wolfgang Köhler, Gabriel Schachtel, Peter Voleske: Biostatistik. Eine Einführung für Biologen und Agrarwissenschaftler (Springer, Berlin, 2002, 320 Seiten). Dies Buch ist sehr nett für viele statistische Probleme. Was fehlt ist sowohl der größere Überblick, das Verallgemeinerte Lineare Modell, *resampling*-Verfahren sowie detailiertere multivariate Statistik. (4) Jürgen Engel: Signifikante Schule der schlichten Statistik (Filander, Fürth, 1997, 113 Seiten). Ein einfacher, kurzer und humorvoller Ratgeber, welcher Test wann zu benutzen ist. Leider werden ANOVA und GLM nur ganz kurz gestreift, Grundlagen nahezu vollständig vernachlässigt. Besser als keine Statistik, aber kein Buch, um Statistik zu verstehen.[1]

Darüberhinaus gibt es eine Reihe englischer Lehrbücher, die hier unbedingt positiv hervorzuheben sind:

1. Legendre & Legendre *Numerical Ecology* (1998). Dieses Buch ist dick, schlecht indiziert und vergleichsweise schlecht lesbar. Es ist aber gleichzeit voll mit sehr nützlichen Prozeduren, und es ist das Standardwerk zur multivariaten Statistik. Ein Muss in jedem Methodenregal! Häufig wenn wir glauben ein neues Problem gefunden zu haben reichte ein Blick in den Legendre, um die Lösung zu finden. Ein toller Schmöker!

[1] Da vergleichende Werbung ja seit ein paar Jahren erlaubt ist, hier eine kurze Anmerkung zum häufigst verkauften Statistikbuch in Deutschland: Der “Sachs” ist das unverständlichste, kompliziertes, unübersichtlichste, unanwendbarste und unstrukturierteste Machwerk was uns jemals untergekommen ist. Wahrscheinlich ist das Meiste richtig, was darin steht. Aber Statistik lernen ist damit unmöglich und selbst zum Nachschlagen taugt es nicht. Seit neustem gibt es eine R-Gänzung dazu: sehr dünn zeigt sie, dass die Hunderte von Tests im Sachs mit generischen R-Funktionen lösbar sind. Als Briefbeschwerer ist es zu labberig und im Regal sieht es wegen seiner gräßlichen Farben auch nicht schön aus. Unser Rat: Finger weg!

2. Crawley *Statistical Computing: An Introduction to Data Analysis using S-plus* (2002[2]). Ein tolles Buch für R-Einsteiger, denn praktisch alle Beispiele laufen auch in R. Sehr viele Themen extrem lesbar dargeboten und mit ausführlichem R-Kode unterlegt. Alle Datensätze gibt es auf Crawleys *homepage*. Diesem Buch sind auch einige Beispiele entlehnt.

3. Quinn & Keough *Experimental Design and Data Analysis for Biologists* (2002). Das unser Meinung nach beste allgemeine Statistikbuch für Biologen; viele Beispiele, allerdings keine Hinweise, wie man die Methoden denn in Software umsetzen kann.

4. Faraways Doppelschlag *Linear Models with R* und *Extending Linear Models with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models* (2005; 2006). Faraway führt ähnlich wie Crawley durch alle wichtigen Gebiete. Etwas mehr verschiedene Methoden und gelegentlich kompetenter wirkend, allerdings ohne den Crawleyschen Schwung. Besonders Teil II (*Extending ...*) ist im Anschluss an die Themen unseres Machwerks zu empfehlen.

5. Fox *An R and S-Plus Companion to Applied Regression* (2002). Ähnlich wie Faraway oder Crawley, nur preiswerter und in manchen Kapiteln ausführlicher. Es lohnt auf jeden Fall ein Blick auf John Fox *homepage*, wo er verschiedene Zusatzkapitel zur Verfügung stellt (etwa Strukturgleichungsmodelle).

Weiterführende Literatur, die sich also den hier behandelten Themen anschließt, gibt es deutlich weniger, meist werden sie sehr speziell und sind dann beim entsprechenden Abschnitt in unserem Werk aufgeführt. Vier Bücher wollen wir trotzdem hier schon empfehlen:

1. Hastie, Tibshirani & Friedman *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2008). Eine wahnsinnig dichte Darstellung einer möglichen statistischen Denkweise. Genau wie Harrell (s.u.) stellen HTF das ganze Buch unter eine Sichtweise, nämlich das der Klassifikationsproblematik. Darin gehen dann die Beispiele von Schrifterkennung bis zur Überlebensanalyse. Sehr inspirierend und die eigene Denkweise hinterfragend, zudem sehr kompetent geschrieben, wenngleich dadurch gelegentlich auch recht mathematisch. Obwohl alle Anwendungen in R oder S-plus ausgeführt wurden gibt es in diesem Buch keinen R-Kode!

2. Harrell *Regression Modeling Strategies - with Applications to Linear Models, Logistic Regression, and Survival Analysis* (2001). Die Ähnlichkeit mit HTF ist nur in der Struktur: eine grundsätzliche Denkrichtung bestimmt das ganze Buch. Hier ist die Denkweise aber ganz anders. Harrell betrachtet Statistik vor allem vor dem Problem ein sinnvolles Modell zu finden. Variablenselektion, Nicht-Linearität der funktionalen Beziehungen und der Ansatz des *bootstrap* für den gesamten Analyseprozess sind hier zentrale Gedanken. Harrell betont immer wieder die Generalisierbarkeit von Ergebnissen, bevorzugt schlanke Modelle, legt Wert auf Quantifizierung von Unsicherheiten und Fehlern auf Schätzern. Die Darstellung ist bei Harrell ganz anders als bei allen bisher erwähnten Büchern. Er reißt die Grundlagen ziemlich kurz ab (unter 100 Seiten), um dann ganz ausführlich an Fallstudien bestimmte Probleme abzuarbeiten (etwa unvollständige Datensätze, *survival analysis*, Modellvereinfachung). Wenn man erst einmal weiß, was in diesem Buch steckt, dann ist es toll zum Nachschlagen. Harrell hat zwei R-*packages* geschrieben, die das Buch begleiten und ein Fülle an Schätzen bietet (**Hmisc** und **Design**); alle Beispiele werden in R vorgerechnet.

3. Zuur, Ieno & Smith *Analysing Ecological Data* (2007). ZIS steigen auf fortgeschrittenem Niveau ein (also etwa beim GLM) und stellen im kürzeren ersten Teil statistische

[2]Sein *R-Book* ist der gleiche Wein in anderem Schlauch und ein Beispiel für gute Vermarktung, nicht für Weiterentwicklung.

Methoden vor, mit Schwerpunkt auf gemischten Modellen (sowohl linear wie auch additiv), aber auch Zeitreihenanalyse, räumliche Statistik und vieles mehr. Dann folgen anderthalb Dutzend ausführliche Fallstudien aus der Ökologie, jeweils mit einem exemplarischen statistischen Problem: super! ZIS leiten ein Statistikbüro in Schottland (Highland Statistics Ltd., obwohl der Laden an der Küste liegt) und haben über die Jahre viel gelehrt und beraten. Das kommt in diesem Buch für Biologen, speziell deskriptiv arbeitende Ökologen, voll zum Tragen. Die Autoren haben eine eigene proprietäre Software (*bodgar*) geschrieben, die als GUI[3] auf R aufsitzt. Deshalb gibt es auch in diesem Buch keinen R-Kode, obwohl wie in HTF alles in R gerechnet wurde. Die meisten Beispiele sind aber so detailiert, dass das Nachbauen kein Problem sein sollte. Alle Daten stehen auf der *webpage* des Buches zur Verfügung. Nach so viel Lob: Es fehlen manche Bereiche (vor allem gemischte Modelle) vollständig. Und: Der Preis ist heftig, aber das Buch auch sehr dick (700 Seiten).

4. Venables & Ripley *Modern Applied Statistics with S*[4] 2002. Dieses Buch, unter Freunden MASS genannt, wird häufig unter R-Novizen als 'das' R-Buch bezeichnet, inkorrekterweise, denn die R-*books* sind nach Farben benannt (siehe Literaturverzeichnis der R beiliegenden *Introduction to R*). MASS war in seiner ersten Auflage ein Meilenstein, denn es hat die ganze Nutzung von R als Statistiksoftware auf hohem Niveau angeregt. Beide Autoren sind maßgeblich an der Entwicklung von S-plus und R beteiligt (und an der R-*help mailing list*) und MASS ist immer noch ein tolles Werk. Meines Wissens taucht in keinem der anderen obigen Bücher etwas zur robusten Regression auf, werden gemischte Verteilungen zum Nachbauen konstruiert usw. Als Leser muss man sich bei MASS gelegentlich mit einem Statistikbuch bewaffnen, um noch zu verstehen, was da abgeht. Da aber ein erheblicher Prozentsatz der Kommentare in der R-*help* auf MASS verweisen, geht um dieses Buch kaum ein Weg drumherum.

Bücher sind natürlich im Internetzeitalter nicht mehr die Hauptresource. Verschiedene *webpages* bieten umfangreiche Dokumentationen und Erklärungen. Besonders die mathematischen Grundlagen sind bei *Wikipedia* oder *mathworld* hervorragend abrufbar. Bei manchen Themen stößt man auch auf Vorlesungskripts, Rechenübungen oder Seiten, die eine bestimmte Berechnung auf dem *webserver* rechnen. Schließlich seien noch die Foren ans Herz gelegt, die sich mit entweder mit spezieller Software beschäftigen (da gibt es dann für jede Software ein eigenes Forum, für R etwa die R-*help mailing list*) oder mit bestimmten Themengebieten (etwas die *multivariate statistics mailing list*). In Form von Fragen und Antworten, abgelegt in *threads*, kann man da Tips finden und bei Problemen um Hilfe bitten. Eine Regel gilt für all diese *mailing*-Listen: Erst suchen, dann fragen! Es ist ausgesprochen lästig, wenn die gleiche Frage jeden Monat wieder auftaucht, obwohl sie seit drei Jahren beantwortet im (durchsuchbaren) Archiv liegt. Die gelegentlich rüde Antwort ist dann ein "RTFM"[5]. Wer will das schon als Antwort erhalten[6]?

Statistik-Software

Die Statistik-Software lässt sich in zwei Gruppen teilen (mit einigen Übergängen): die der Schaltflächen-*pop-up*-Fenster-Software (Minitab, SPSS, Statistica, Systat, usw.), und die der Syntax-gesteuerten Programme (Genstat, Mathematica, Matlab, R/S-plus, SAS, XploRe, usw.). Erstere haben den Vorteil leicht erlernbar zu sein, letztere den Vorteil, dass man auch

[3] *Graphic User Interface*: Nutzeroberfläche, in diesem Fall Tcl/Tk-basiert

[4] `S` ist eine `C`-ähnliche Programmiersprache, in der R und S-plus geschrieben wurden. Eine dritte Software in `S` ist BUGS (*Bayesian analysis Using Gibbs Sampler*).

[5] Read the fucking manual!

[6] Ein R-*package* namens **fortunes** sammelt Kommentare aus der R-*help mailing list* und gibt sie zufällig nach dem Aufruf `fortune()` aus. Zugegeben ein bisschen albern, aber für den fortgeschrittenen R-Nutzer, der mit den ganzen Namen inzwischen etwas anfangen kann recht lustig.

nach längere Zeit noch nachvollziehen kann, was man gemacht hat (da der Syntax gespeichert, bzw. mit ausgegeben wird). Aus didaktischen Gründen ist Syntax-Software zu bevorzugen: Anstelle von Abbildungen der Graphikoberfläche und umständlicher Beschreibung welche Boxen angeklickt und Felder wie ausgefüllt werden müssen, können wir die Syntax einfach niederschreiben. Da die Syntax zudem den gesamten Analysebefehl enthält, gibt es keine optionalen Fensterchen, die man beim zweiten Analyseversuch vergessen hat. Und schließlich muss der Benutzer wissen, was er tut: ein einfaches Klicken ist nicht möglich. (Nichtsdestotrotz kann man auch mit Syntax-Software noch genug Fehler machen.) Diese Vorteile werden durch einen hohen Preis erkauft: Wir müssen erst die Syntax lernen, quasi eine Programmiersprache, und die ist nicht immer intuitiv, erst recht nicht in einer fremden Sprache.

Das alles entscheidende Argument ist aber die Qualität der Software. Wer Excel benutzt, weiss nicht, was er tut. Die Funktionsbeschreibungen sind so mager wie die Programmierfehler Legion. Fehlerhaften Routinen von Excel, Minitab oder selbst Statistica und SPSS füllen ganze *webpages*. In eingeschränktem Maße gilt dies auch für alle anderen kommerziellen Programme, da der *source code* nicht offen liegt.

Wir haben uns für die Software R entschieden. Sie ist umsonst, der *source code* ist frei zugänglich, durch ihre Syntaxsteuerung unglaublich vielseitig, vielfach dokumentiert, für alle Computersysteme verfügbar (Windows, Unix, Mac) und verfügt über eine der umfassendsten Sammlungen an Analysefunktionen überhaupt. Die kommerzielle Version von R , S-plus, bietet *software support*, mehr nicht[7]. Für Institute mag dies oder SAS eine Alternative darstellen.

Wir können und wollen auf diesen Seiten keine umfassende Einführung in R geben. Im Appendix ist eine kurze Einführung enthalten, und im Text werden sich immer wieder Beispiele finden, die bestimmte Funktionen erklären. Wer R (oder S-plus) gründlicher erlernen will, kommt um englische Bücher nicht herum (Venables and Ripley 2002; Crawley 2002).

Zum Gebrauch dieses Buches

Es sollen verschiedene Formen der Benutzung möglich werden. Zum einen soll mit Hilfe der Beispiele dem Leser ermöglicht werden, sich selbständig in eine statistische Methodik einzuarbeiten. Zweitens kann man dieses Buch zum Nachschlagen von Methoden benutzen, allerdings weniger zum Nachschlagen von Formeln. Und schließlich sollen die Beispiele in R das Erlernen dieses Programmes ermöglichen. Da hier sowohl die Syntax als auch die Interpretation des *outputs* wichtig ist, wird beides angeboten. Für Benutzer anderer Programme ist dies kein Verlust: aus dem Syntax werden die Prinzipien deutlich, und andere Programme haben (so sie gut sind) einen vergleichbaren *output*.

Terminologie und Kommasetzung

Wie die meisten Forschungsgebiete so sind auch statistische Konzepte heute durch angloamerikanische Forscher dominiert. Wenngleich sicherlich deutsche Kollegen ihren Teil dazu beitragen, so ist die Sprache des Austauschs doch Englisch. Entsprechend sind deutsche Ausdrücke für manche Begriffe, um das Mindeste zu sagen, ungebräuchlich. Beispielsweise ist das deutsche Wort „Wahrscheinlichkeitsdichte“ zwar korrekt, aber weitaus weniger gebräuchlich und eingängig als „*likelihood*“. Dieser Verenglischung der deutschen Wissenschaftssprache fröhnen wir auf diesen Seiten ebenfalls. Das hat immerhin das Gute, dass es englische Bücher zum Thema etwas zugänglicher macht.

Unseres Wissens sind die Länder des deutschsprachige Raums, Schweden und Spanien unter den wenigen Ländern, in denen Kommazahlen auch wirklich Kommazahlen sind. Im Rest der Welt sind Kommazahlen „Punktzahlen“, und das Komma markiert etwa 1,000er Stellen (wie bei uns der Punkt). Wir werden uns in diesem Werk dem Rest der Welt anschließen und etwa

[7] In der letzten Fassung stand hier noch „ein paar Zusatzfunktionen“. Inzwischen ist der Funktionsumfang von R viel größer als der von S-plus und es gibt kaum noch eine S-plus Funktion, die nicht auch in R vorliegt.

π durch (gerundet) 3.141 darstellen, nicht durch 3,141. Dies hat vor allem zwei Gründe: (1) Wenn wir Kommazahlen benutzen, und dann aber die Software Punktzahlen erwartet (was viele, aber nicht alle Programme tun), so kommen wir unnötig durcheinander. (2) In R wird das Komma als Trennzeichen etwa zwischen Elementen eines Vektors oder zwischen Optionen eines Befehls benutzt. Es ist hier (wie auch in SAS oder Matlab) nicht möglich, den Syntax zu verändern, damit er die deutsche Rechtschreibung versteht.

Wir bitten um Nachsicht.

C. F. D.
I. K.

Teil I.

Grundlagen

1. Einleitung

1.1. Wissenschaftliche Methodik: Beobachtungen, Ideen, Modellvorstellungen, Hypothesen, Experimente, Tests und wieder von vorn

Die meisten Biologen hat die Faszination am Lebenden in dieses Berufsfeld getrieben. Unser täglicher Umgang mit einer extrem vielfältigen und komplizierten Welt stellt den Reiz dar, die Prozesse und Muster der lebenden Umwelt zu verstehen ist die Herausforderung. über lange Jahrzehnte haben Biologen der Natur auf die Finger geschaut, sie nachgebaut, nachgemessen, manipuliert oder mathematisch-simplifiziert beschrieben. Während z.B. Artbeschreibungen oder Enzymnachweise qualitative Erkenntnisse sind, spielt seit etwa 50 Jahren die Quantifizierung biologischer Prozesse ein immer größere Rolle. Zudem sind, wie es scheint, die meisten qualitativen Aussagen gemacht, und weitere lassen sich nur mühsam der Natur abringen. Der typische, iterative Gang eines wissenschaftlichen Erkenntnisprozesses sei hier kurz nachgezeichnet, um die Rolle der Statistik darin besser zu verstehen.

Es beginnt mit einer Beobachtung; uns fällt ein Muster auf, eine Regelmäßigkeit im sonstigen Durcheinander. Nach etwas Nachdenken kann es sein, dass wir eine Idee entwickeln, wodurch dieses Muster entstanden sein könnte. Wir entwickeln also eine Modellvorstellung, wie dieser kleine Teil der Welt zusammensitzt. Von hier ist es nur ein kurzer Schritt, sich Hypothesen zu überlegen, denen man mit geeigneten Experimenten nachspüren könnte. Wir führen das Experiment, die entsprechende Beobachtung durch, und können nun die gewonnenen Daten auf die Konsistenz mit unseren Vorstellungen, unserer Hypothese, testen. Wenn der Test positiv ausfällt, könnten wir uns mit einem „Aha!“ in den Sessel fallen lassen, oder uns neuen Themen zuwenden. Wenn der Test negativ ausfällt, dann wissen wir zumindest, dass es in diesem Fall nicht der von uns verdächtigte Mechanismus war. Dann können wir nach neuen Wegen Ausschau halten, und das Spielchen beginnt von vorne.

Was hier so lax dargestellt ist, stellt den Alltag und Lebensinhalt vieler Forscher dar. Entsprechend hat sich dieses System auch recht weit entwickelt. Es gibt eine wissenschaftsphilosophische Grundlage, die unterschiedliche Forschungsansätze einander gegenüberstellt (Stichworte „induktiv“ und „deduktiv“). Induktive Forschung geht davon aus, dass sich durch Beobachtungen allein ein Phänomen erklären lässt, und dass alle Probleme der Welt so lösbar sind. Dies war vor allem eine im antiken Griechenland vorherrschende Meinung. Inzwischen hat sich in der Naturwissenschaft der hypothetisch-deduktiven Ansatz durchgesetzt. Er geht üblicherweise auch von einer Beobachtung aus, aber diese führt zu Hypothesen über die Mechanismen, die wir wiederum in Experimenten überprüfen können.

1.2. Das Testen von Hypothesen

Statistik ist eine Methode der Entscheidungshilfe; ihr obliegt nicht die Entscheidung selbst. Das Element der Statistik ist die Hypothesenprüfung. Wenn wir Daten gesammelt haben wissen wir noch nicht, ob diese Daten nicht rein zufällig zustande gekommen sind. Auch nach einer statistischen Analyse wissen wir dies noch nicht! Was die Statistik aber für uns leistet, ist eine Abschätzung der *Wahrscheinlichkeit*, dass diese Zahlen zufällig zustande gekommen sind. Um die genaue Rolle der Statistik wertschätzen zu können, müssen wir uns zunächst ein Kochrezept für den Test von Hypothesen anschauen.

1.2.1. Kochrezept für den Test von Hypothesen

Am Anfang steht eine Frage: Sind alle Schwäne weiß?, Können Eisbären länger schwimmen als Pinguine? oder Folgt die Enzymkinetik der Esterase X der Michaelis-Menton-Gleichung? Diese Frage formulieren wir dann so um, dass sie eine Arbeitshypothese darstellt: Alle Schwäne sind weiß. Eisbären können länger schwimmen als Pinguine. und Die Esterase X folgt der MM-Kinetik. Nun stellen wir fest, dass wir diese Arbeitshypothesen schlecht „beweisen“ können. Wir müssten alle Schwäne anschauen, die Ausdauer aller Eisbären und Pinguine messen und unsere Esterase X unter allen Umweltbedingungen und Substratkonzentrationen reagieren lassen. Da wir realistisch nur einen kleinen Teil dieser Messungen durchführen können, gilt unsere Schlussfolgerung immer nur solange, bis weitere Daten sie widerlegen.

Seit den 50er Jahren hat sich zudem eine auf Karl Popper zurückgehende Wissenschaftsphilosophie etabliert. Popper argumentierte, dass wir eine Hypothese nicht beweisen, sondern nur widerlegen können (Falsifikation). Deshalb schlug er vor, jeder Arbeitshypothese (H_1) eine komplementäre Nullhypothese (H_0) zur Seite zu stellen, und diese stattdessen zu untersuchen. Wenn wir die Nullhypothese widerlegen können, so ist damit die Arbeitshypothese bis auf weiteres angenommen. Entsprechend Beispiele wären: Es gibt nicht-weiße Schwäne. Pinguine schwimmen mindestens genauso lange wie Eisbären. und Die Esterase X zeigt keine MM-Kinetik.

Jetzt führen wir unsere Messungen und Experimente durch: Wir suchen 100 Schwäne und schreiben auf, welche Farbe sie haben; wir fahren mit dem Schlauchboot neben Eisbären und Pinguinen und messen ihre Schwimmdauer; wir pipettieren unterschiedliche Substratkonzentrationen zu unsere Esterase und messen die Zeit bis zum vollständigen Abbau.

Endlich schlägt die Stunde der Statistik! Wenn alle Schwäne weiß sind, so weisen wir die Nullhypothese, dass es nämlich andersfarbige Schwäne gibt, zurück. Ein einzelner grüner oder schwarzer Schwan hingegen bestätigt die Null- und widerlegt die Arbeitshypothese. Das war einfach. Bei den Pinguinen und Eisbären wird es schon schwieriger. Manche Individuen sind nur kurz schwimmen gewesen, manchmal waren die Pinguine länger im Wasser, manchmal die Eisbären. Wie geht's hier weiter? Nun, stellen wir uns vor, Pinguine und Eisbären wären gleich ausdauernd. Dies bedeutet, dass ihre Daten*verteilungen* sich nicht unterscheiden dürften, sie also aus der gleichen statistischen Grundgesamtheit stammen. Die genauen sich anschließenden statistischen Tests tun hier nichts zur Sache. Die Wahrscheinlichkeit dafür kann man mittels statistischer Methoden berechnen. Und schließlich zur Esterase. Den Messungen der Reaktionsgeschwindigkeit je Substratkonzentration können wir eine Regressionslinie anpassen. Je besser der Fit (die Anpassung), desto wahrscheinlicher ist, dass die Esterase X tatsächlich einer MM-Kinetik folgt. Ergeben andere Enzymkinetikmodelle einen besseren Fit, dann müssen wir u.U. unsere Nullhypothese annehmen.

1.2.2. Testfehler

Wir können bei unserer Entscheidung für oder gegen die Arbeitshypothese vier Situationen unterscheiden (Tabelle 1.1):

Sollten wir uns für unsere Arbeitshypothese entscheiden (fälschlicherweise, denn es gibt ja schwarze Schwäne), so begehen wir einen Fehler 1. Ordnung. Wenn unsere Enzymkinetikdaten nicht mit der MM-Kinetik in Einklang sind (obwohl das Enzym der MM-Kinetik folgt), so begehen wir einen Fehler 2. Ordnung.

1.3. Tests

An den Beispielen wird deutlich, dass der Stichprobenumfang, d.h. die Anzahl der Messungen, einen Einfluss auf unsere Entscheidung haben wird. Irgendwann laufen wir in einem Park dem schwarzen Trompeterschwan aus Südafrika über den Weg, und unsere H_1 ist hinfällig.

Tabelle 1.1.: Abhängig vom wahren Zustand (Spalten 2-3) können wir aufgrund unserer statistischen Analyse zu Annahme der Nullhypothese oder Arbeitshypothese kommen (Zeilen 2 bzw. 3). Wenn wir die *Arbeitshypothese* annehmen, obwohl sie falsch ist, begehen wir einen Fehler 1. Ordnung (unten links). Wenn wir hingegen die *Nullhypothese* annehmen, obwohl sie falsch ist, begehen wir einen Fehler 2. Ordnung (rechts oben). Die bei einer statistischen Analyse berechneten Wahrscheinlichkeiten (p) geben den Fehler 1. Ordnung an (α). Der Fehler 2. Ordnung (β) wird selten berechnet. Die Teststärke (*power*) eines Test ($1 - \beta$) hingegen wird bei nicht-signifikanten Effekten gelegentlich berichtet.

	Nullhypothese wahr	Arbeitshypothese wahr
Nullhypothese angenommen	richtig; $p = 1 - \alpha$	falsch; Fehler 2. Ordnung $p = \beta$
Arbeitshypothese angenommen	falsch; Fehler 1. Ordnung $p = \alpha$	richtig; $p = 1 - \beta$

Neben dem Stichprobenumfang gibt es andere Faktoren, die einen Einfluss auf die Teststärke (also eine wahre Arbeitshypothese als wahr zu erkennen) haben. Je größer der Unterschied zwischen den Daten der Nullhypothese und der Arbeitshypothese, desto wahrscheinlicher werden wir diesen Unterschied auch finden. ähnlich verursachen stark streuende Daten natürlich auch eine Reduktion der Teststärke.

Grundsätzlich finden wir also, dass ein möglicher Unterschied zwischen Arbeits- und Nullhypothese umso eher detektierbar ist, je größer der wahre Unterschied, je geringer die Streuung der Daten und je größer der Stichprobenumfang ist.

1.3.1. Weitere Begriffe zum Thema Tests

Ein Test gilt als *einseitig* (*one-tailed*), wenn eine gerichtete Hypothese getestet wird: Schneehuhnmännchen sind größer als Schneehuhnweibchen. Soll nur getestet werden, ob überhaupt ein Größenunterschied besteht (egal in welcher Richtung), so müssen wir einen *zweiseitigen* (*two-tailed*) Test benutzen. (Bei den symmetrischen Testverteilungen geschieht dies einfach durch Verdopplung des P-Wertes.) Die meisten Statistikprogramme sind automatisch auf den weitaus häufiger genutzten zweiseitigen Test eingestellt.

Allen Tests liegen **Annahmen** zugrunde. Welche Annahmen dies im speziellen sind, müssen wir in der Dokumentation des speziellen Tests nachschlagen. Grundsätzlich gilt für alle Tests, dass die Daten **unabhängig** voneinander sein müssen. Wenn wir bei einem Jungen und einem Mädchen jeweils 17 Mal das Gewicht messen, so gibt uns dies keine Aufschluss über den Unterschied zwischen Jungen und Mädchen, sondern nur zwischen *diesem* Jungem und *diesem* Mädchen. Die einfachste Art und Weise Unabhängigkeit zu erzeugen ist die zufällige Auswahl der Stichprobe. Zufälligkeit an sich ist aber kein statistisches Muss, denn auch regelmäßige („jeder fünfte“) Auswahl kann unabhängige Daten generieren. Wichtig ist, dass jedes Objekt die gleiche Chance hat gewählt zu werden.

Desweiteren verlangen manche Tests eine bestimmte Verteilung der Daten. Die auf Verteilungsannahmen beruhenden Tests nennen wir *parametrisch* (diese Parameter beschreiben eben die Datenverteilung). Dem stehen die nicht-parametrischen Tests gegenüber, in denen nicht der absolute Messwert sondern sein Vorzeichen oder sein Rang innerhalb aller Messwerte analysiert wird. Beispiele für beide Test finden sich im Abschnitt 4.

Ein Problem taucht auf, wenn wir an Objekten viele Messungen machen (etwa an 50 Menschen jeweils Schuhgröße, Nasenlänge, Spannweite, Schrittlänge, Ohrlänge usw.) und nachher auf Unterschiede zwischen Objekten testen (etwa zwischen den Geschlechtern). Einfach aus purem Zufall werden wir einen signifikanten Unterschied finden; je mehr Vergleiche wir ma-

chen, desto mehr Unterschiede werden signifikant sein. Dann nehmen wir die Arbeitshypothese an, obwohl sie verkehrt ist. Dieses Phänomen der **multiplen Tests** nennt sich Fehlerinflation (*type 1 error inflation*). Es gibt verschiedene Wege dafür zu korrigieren (etwa den erhaltenen P-Wert durch die Anzahl durchgeführter Vergleiche teilen: Bonferroni-Korrektur; gleichzeitiges Testen aller Messgrößen etwa in einer MANOVA). Wichtig ist hier erst einmal, sich des Problems bewusst zu werden.

Schließlich taucht bei parametrischen wie nicht-parametrischen Tests gelegentlich das Problem der **Datenbindung** auf (*ties*). Dabei gibt es einen oder mehrere Messwerte mehrfach. Besonders bei nicht-parametrischen Tests kann dies zu Verzerrungen der Ergebnisse führen. Während Alternativen existieren (Permutationstest sind nicht anfällig für Datenbindung), so ist doch das Wissen um dieses Problem das Wichtigste.

1.3.2. Schlussbemerkungen zu Tests

Der magische P-Wert von üblicherweise 0.05 (oder 5% Irrtumswahrscheinlichkeit) ist eine Konvention und somit willkürlich (als Richtgröße, nicht als starre Marke vorgeschlagen von Fisher 1956). Manchem mag diese Irrtumswahrscheinlichkeit zu hoch, anderen zu niedrig sein. Die Fixierung auf Signifikanztest ist vielfach kritisiert worden (siehe etwa Shaver 1993; Johnson 1999), aber derzeit noch immer das Paradigma (Oakes 1986; Ford 2000). Vor allem: erst wer diese klassische Form des Signifikanztests beherrscht sollte sich daran machen, die kritischen P-Werte zu modifizieren (Simberloff 1983).

Eine Versuchung, der viele Menschen erliegen, ist das **Extrapolieren**. Während es durchaus verständlich erscheint, dass wir Werte, die zwischen den von uns gemessenen liegen, interpoliert, so ist es ebenso problematisch, auf Werte außerhalb des Messbereichs zu schließen. Wenn wir eine Regression der metabolischen Grundumsatzrate gegen das Körpergewicht machen (Kleiber-Funktion), so können wir nicht auf den Grundumsatz eines 0 g schweren Tieres extrapolieren, oder den eines Säugetieres von der Größe unseres Planeten. Hier spielen u.U. ganz andere Faktoren eine limitierende Rolle als das Körpergewicht. Mindestens aber müssen wir solche Vorhersagen mit vielen begleitenden Worten und mit einer Fehlerabschätzung machen.

2. Grundlagen

Statistik versucht die Wahrscheinlichkeit zu bestimmen, mit der ein Ereignis eintrifft. Dafür muss man wissen, wieviele Möglichkeiten es gibt, von denen eines (oder mehrere) dann das gesuchte Ereignis ist (sind). Um abzuschätzen, wie wahrscheinlich es ist, dass Newton tatsächlich ein Apfel auf den Kopf fiel, als er unter einem Apfelbaum saß, müssten wir wissen, wie groß der Baum, wieviele Äpfel, welches Datum, welche Wetterbedingungen an diesem Tag und den Wochen zuvor, wie lange Newton dort saß, wie häufig in den letzten Tagen er sich dort aufhielt, und, schlussendlich, wie groß sein Kopf war. Die meisten dieser Daten werden wir nicht oder nur sehr ungenau rekonstruieren können, eine Abschätzung der Wahrscheinlichkeit bleibt damit sehr ungenau. Um die statistischen Verfahren hinter diesen komplexen Fragen zu verstehen, müssen wir wohl oder übel ein paar Grundlagen verstehen, auch wenn ihre Relevanz nicht unmittelbar deutlich wird.

2.1. Wahrscheinlichkeit

2.1.1. Kombinatorik

Kombinatorik unterliegt aller Statistik. Ein Verständnis wenigstens der einfachsten Begriffe ist deshalb wünschenswert. Diese seien im folgenden kurz dargestellt.[1]

Als **Permutationen** bezeichnen wir die verschiedenen Möglichkeiten, eine Anzahl Elemente anzuordnen. So kann man die Elemente a, b und c auf 6 verschiedene Weisen anordnen: abc, acb, bac, bca, cab, cba. Allgemein gibt es für n unterscheidbare Objekte $n!$ Permutationen[2].

Für n Objekte, die in k Gruppen nicht weiter unterscheidbarer Objekte vorliegend, gibt es

$$\frac{n!}{n_1! \cdot n_2! \cdots n_k!}$$

Permutationen, wobei n_k jeweils die Anzahl nicht-unterscheidbarer Objekte sind.

Beispiel 1: 2 rote, 3 grüne, 7 blaue Kugeln ($k = 3$): $n_1 = 2$, $n_2 = 3$ und $n_3 = 7$, $n = 12$, können in $\frac{12!}{2! \cdot 3! \cdot 7!} = 7920$ Permutationen vorliegen. Beispiel 2: Beim Skat gibt es 32 Karten, die zu je 10 an einen von drei Spielern gehen, und zwei Karten in den Skat. Entsprechend gibt es $\frac{32!}{10! \cdot 10! \cdot 10! \cdot 2!} \approx 2.8 \cdot 10^{15}$ Kartenvarianten.

Demgegenüber stehen **Kombinationen**: Wenn aus einer Menge von n Elementen nicht alle Elemente gezogen werden (wie unter Permutationen angenommen), sondern nur k, spricht man von Kombinationen. Diese können mit Zurücklegen erfolgen (es wird also ein Element gezogen, vermerkt und zurückgetan), oder ohne. Dann kommt es noch darauf an, ob die Reihenfolge eine Rolle spielen soll, ob also die Sequenz abaab eine andere sein soll als abbaa. Zur Berechnung dieser Kombinationen benutzt man eine besondere mathematische Schreibweise, den Binomialkoeffizienten $\binom{n}{k}$ (sprich: n über k). Er berechnet sich wie folgt:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n \cdot (n-1) \cdot (n-2) \cdots (n-k+2) \cdot (n-k+1)}{1 \cdot 2 \cdot 3 \cdots (k-1) \cdot k}$$

Dann lassen sich k Elemente aus einer Menge von n Elementen zu folgender Anzahl Kombinationen zusammensetzen:

[1] Die folgenden Passagen zur Kombinatorik sind eng an ein Skript von Klaus Doden (1991) angelehnt.

[2] „$n!$"bedeutet „n Fakultät", und symbolisiert ein Aufmultipizieren aller ganzen Zahlen bis n. Beispiel: $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$.

1. ohne Zurücklegen,
   a) ohne Berücksichtigung der Reihenfolge: $\binom{n}{k}$
   b) mit Berücksichtigung der Reihenfolge: $k! \cdot \binom{n}{k}$
2. mit Zurücklegen,
   a) ohne Berücksichtigung der Reihenfolge: $\binom{n+k-1}{k}$
   b) mit Berücksichtigung der Reihenfolge: n^k

2.1.2. Wahrscheinlichkeit

Die relative Häufigkeit des Eintreffens eines Ereignisses A nennt man Wahrscheinlichkeit des Eintreffens von A, $P(A)$. Der Wurf einer Münze illustriert diese recht technische Definition. Es gibt zwei mögliche Ereignisse, Kopf und Zahl. Jedes trifft im Mittel in der Hälfte aller Fälle auf, also ist $P(Kopf) = P(Zahl) = 0.5$.

Für ein Experiment, mit k möglichen, gleich wahrscheinlichen Ausgängen gilt:

$$P(\text{Ausgang } 1) = P(\text{Ausgang } 2) = \cdots = P(\text{Ausgang } k) = 1/k$$

Alle diese Wahrscheinlichkeiten aufsummiert müssen 1 ergeben, da ein Experiment ja schließlich einen Ausgang haben muss:

$$\sum_{i=1}^{k} P(\text{Ausgang } i) = 1$$

.

Schließen sich zwei Ereignisse A und B aus (sind sie *unvereinbar*), so ist die Wahrscheinlichkeit, eines von beiden zu erhalten ($P(A \cup B)$, $\cup$ bedeutet „oder“: Vereinigungsmenge), gleich der Summe der Wahrscheinlichkeiten der Einzelereignisse. Oder allgemein:

$$P(A_1 \cup A_2 \cup \cdots \cup A_i) = P(A_1) + P(A_2) + \cdots + P(A_i) = \sum_{k=1}^{i} P(A_k) \tag{2.1}$$

Zur Verwirrung ein auf den ersten Blick ähnlicher Fall: Sind zwei Ereignisse A und B voneinander *unabhängig* (wie etwa Kopf oder Zahl bei zwei aufeinanderfolgenden Würfen), so ist die Wahrscheinlichkeit des Auftretens einer Ereignissequenz „AB“ gegeben durch das Produkt der Wahrscheinlichkeiten der Einzelereignisse: $P(A \cap B) = P(A) \cdot P(B)$ ($\cap$ bedeutet „und“: Schnittmenge). Oder allgemein:

$$P(A_1 \cap A_2 \cap \cdots \cap A_i) = P(A_1) \cdot P(A_2) \cdot \cdots \cdot P(A_i) = \prod_{k=1}^{i} P(A_k) \tag{2.2}$$

Ganz allgemein gilt für zwei beliebige Ereignisse A und B:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \tag{2.3}$$

In Worten: Die Wahrscheinlichkeit, dass genau A oder genau B eintritt, ist gleich der Wahrscheinlichkeit des Eintretens von entweder A oder B, abzüglich der Wahrscheinlichkeit dass beide gleichzeitig eintreten. Für *unvereinbare* Ereignisse führt dies zu Formel 2.1, da ja $P(A \cap B) = 0$.

Wenn A eingetreten ist, was ist dann die Wahrscheinlichkeit, dass auch noch B eintritt (= bedingte Wahrscheinlichkeit von B unter der Hypothese A)? Dies ist natürlich die Wahrscheinlichkeit dass beide eintreten, hochgerechnet auf alle Fälle in denen A eingetreten ist. Das „Hochgerechnet“ ist mathematisch eine Division durch $P(A)$:

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

2.2. Datentypen

Im wesentlichen kann man drei Typen von Daten unterscheiden: kontinuierliche, kategorische und diskrete[3]. Am besten wird dies an Beispielen deutlich: Wenn wir die Jagdgeschwindigkeit eines Geparden messen, so kann das Tier jede Geschwindigkeit zwischen 0 und 100 $\mathrm{km\,h^{-1}}$ annehmen, also sind die gemessenen Geschwindigkeitsdaten kontinuierlich. Wenn wir hingegen die Augenfarbe von Geparden erfassen, so sind dies kategorische Werte: gelb, grau oder schwarz. Darüberhinaus gibt es diskrete Werte, wie die Anzahl Blütenköpfe je Pflanze oder die Größe einer Orang-Utan-Gruppe, die also ganzzahlig-kontinuierlich sind. Solche Daten entstehen vor allem durch Zählungen (*count data*).

Dieser Unterschied ist wichtig, wenn es an die Analyse der Daten geht, da sie unterschiedliche *Verteilungen* haben.

2.3. Verteilungen

Ein Wort vorweg. Wir müssen zwischen *Grundgesamtheit* und *Stichprobe* unterscheiden. Die Parameter einer Verteilung (zumeist durch griechische Buchstaben symbolisiert) beschreiben die Grundgesamtheit. Die statistischen Parameter gleichen Namens (etwa: Mittelwert), symbolisiert durch lateinische Buchstaben, sind ein Schätzer für die Grundgesamtheit. Sie werden meist etwas anders berechnet und sind für uns die wichtigeren Parameter. Der handelsübliche Mittelwert einer Normalverteilung etwa (symbolisiert als $\bar{x}$) dient uns als *erwartungstreuer Schätzer* des Mittelwerts der Grundgesamtheit (μ). Die Mittelwerte, Standardabweichungen usw. die wir für einen Datensatz berechnen sind also grundsätzlich Mittelwerte der Stichprobe und also lateinische Buchstaben. Zur Verwirrung werden hier aber Verteilungsformeln für die Grundgesamtheit angegeben - mit griechischen Buchstaben.

2.3.1. Die Normalverteilung

Wenn wir 100 Mal die Jagdgeschwindigkeit unterschiedlicher Geparden messen, dann werden diese natürlich variieren. Man kann jetzt die gemessenen Geschwindigkeiten in Klassen zusammenfassen (z.B. 51-60, 61-70, 71-80, 81-90, 91-100, 101-110 $\mathrm{km\,h^{-1}}$) und dann auszählen, wieviele Werte in jeder Klasse liegen. Wenn man dann die Anzahl der Werte pro Gruppe über die Klassengrößen aufträgt erhält man etwa Abbildung 2.1.

Diese Verteilung ähnelt der häufigsten Verteilungsform, der **Normalverteilung** (auch Gauss'sche Glockenkurve). Diese lässt sich mittels der Gauss'schen Formel berechnen:

[3] Es gibt im Deutschen einen ganzen Wust an Namen für diese Grundtypen und ihre Spezialformen. Diese werden im folgenden nicht benutzt, da sie mehr Jargon als hilfreich sind. Hier ihre Bedeutung: **nominal**: Merkmale sind unterscheidbar, aber nicht in einer logischen Reihenfolge sortierbar (Augenfarbe); **ordinal**: unterscheidbar und sortierbar, aber nicht direkt miteinander verrechenbar (Größe in Kategorien: klein, mittel, groß), **kardinal**: diskret und sortierbar, mit Zahlenwerten, die interpretierbar sind (4 ist doppelt so groß wie 2, und 3 liegt dazwischen; Alter in Jahren); **metrisch**: gleiche Abstände zwischen Variablenstufen (umfasst kardinale, stetige und quasistetige Variablen); **stetig**: Parameter nimmt kontinuierliche, sortiert-zunehmende Werte an (Temperatur); **quasistetig**: Parameter nimmt sortiert-zunehmende, aber nicht kontinuierliche Werte an (Schuhgrößen).

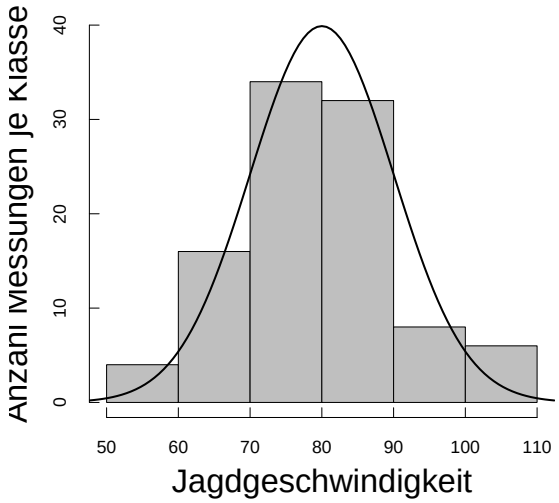


Abbildung 2.1.: Verteilung von 100 Messungen zur Jagdgeschwindigkeit von Geparden. Die durchgezogenen Linie ist die Normalverteilung, aus der die 100 Geschwindigkeitswerte zufällig gezogen wurden ($\mu = 80, \sigma = 10$).

$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{2.4}$$

mit Mittelwertschätzer

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2.5}$$

und Standardabweichung

$$s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}} \tag{2.6}$$

In dieser Formel sind zwei Parameter entscheidend: Der Mittelwert μ (sprich: „müh“, mit Schätzer $\bar{x}$, “X-quer”) und die Standardabweichung σ („sigma“, mit Schätzer s). Sie geben an, um welchen Wert sich diese Verteilung gruppiert (in Abbildung 2.1 also $\mu = 80$) und wie stark zufällig gezogenen Werte x um diesen Mittelwert streuen. Der Mittelwert μ schiebt also die Verteilung nach rechts und links auf der x-Achse, während die Standardabweichung σ die Form der Glocke flacher oder steiler macht.

Aus Gründen, über die es sich in diesem Zusammenhang nicht lohnt näher zu sprechen, liegen ca. 65% aller Werte zwischen den x-Werten $\mu - \sigma$ und $\mu + \sigma$ (abgekürzt: im Interval $[\mu - \sigma; \mu + \sigma]$), bzw. ca. 95% der Werte im Interval $[\mu - 2\sigma; \mu + 2\sigma]$. Die durch Formel 2.4 beschriebene Verteilung kann man über die Daten in Abbildung 2.1 legen (durchgezogene Linie).

Wir wollen bisweilen prüfen, ob unsere Daten denn tatsächlich normalverteilt sind. Dies geschieht am einfachsten mit dem Kolmogorov-Smirnov-Test. Dabei geben wir neben den Daten auch den Mittelwert der Daten, bzw. den bekannten Mittelwert der Verteilung, an.

```
> speed <- rnorm(20, 80, 10)
> ks.test(speed, "pnorm", mean=mean(speed), sd=sd(speed))

        Two-sample Kolmogorov-Smirnov test
```

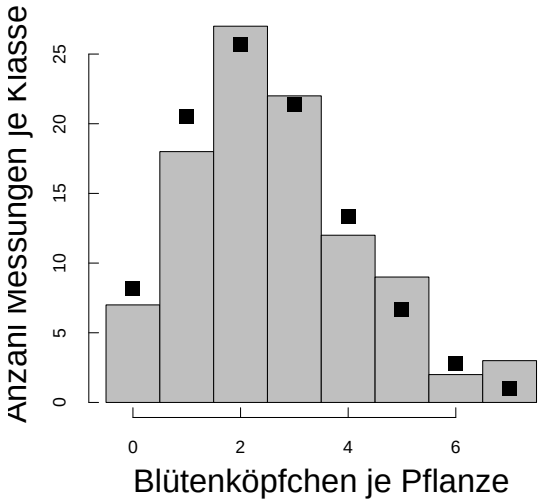


Abbildung 2.2.: Verteilung von 100 Zählungen zu Blütenköpfchen am Kleinen Wiesenknopf. Die schwarzen Punkte markieren die Poissonverteilung, aus der die 100 Blütenkopfzahlen zufällig gezogen wurden ($\mu = 2.5$).

```
data:  speed and mean(speed)
D = 0.152, p-value = 0.6894
alternative hypothesis: two.sided
```

Die Hypothese, dass die Daten speed nicht einer Normalverteilung mit Mittelwert `mean(speed)` entstammen, muss mit einer Wahrscheinlichkeit von 0.95 zurückgewiesen werden.

Es bleibt zu erwähnen, dass kontinuierliche Daten nicht notwendigerweise so schön normalverteilt sein müssen. Leider werden wir uns später noch viel damit beschäftigen, wie wir mit nicht-normalverteilten Daten arbeiten. Im folgenden wollen wir uns aber erst einmal andere Verteilungen anschauen, um dann doch wieder zur Normalverteilung zurückzukehren.

2.3.2. Die Poisson-Verteilung

Zähldaten (wenn man denn *count data* so übersetzen will) weisen eine besondere Eigenschaft auf. Beim Zählen beginnt man immer bei 0, und Zahlenwerte weit von 0 entfernt werden immer seltener.

Zählen wir also beispielsweise Blütenköpfchen an einer Pflanze (z.B. dem Kleinen Wiesenknopf, *Sanguisorba minor*). Manche Pflanzen haben keine Blütenköpfchen, ein ganzer Schwung einen, zwei oder drei. Die Verteilung sieht also aus wie in Abbildung 2.2. Offensichtlich ist die Verteilung asymmetrisch, und zwar nach rechts ausgezogen. Da es sich um diskrete Werte handelt, ist die Verteilung auch diskret, keine durchgezogenen Linie wie bei der Normalverteilung. Die Berechnung der einzelnen Werte dieser sogenannten **Poisson-Verteilung** erfolgt mittels Formel 2.7.

$$p(x) = \frac{\lambda^x}{x! e^\lambda} \tag{2.7}$$

mit dem Verteilungsparameter λ (Mittelwert), der sich für eine Stichprobe (dann $\hat{\lambda}$)) genauso berechnet wie der Mittelwert der Normalverteilung:

$$\hat{\lambda} = \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{2.8}$$

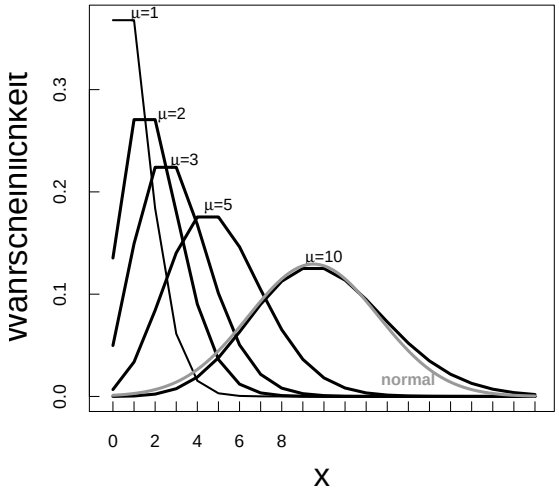


Abbildung 2.3.: Poisson-Verteilungen für verschiedene Mittelwerte (μ). Die graue Verteilung ist eine Normalverteilung mit $\mu = 9.5, \sigma = \sqrt{9.5}$. Die Poisson-Verteilung ist eine diskrete Verteilung, die Linien verbinden also diskrete Werte.

y ist die Wahrscheinlichkeit, dass die Anzahl Ereignisse einen bestimmten Wert ($x = 0, 1, 2, \ldots$) haben.

Wir sehen, dass die Poisson-Verteilung nur einen Parameter besitzt, den Mittelwert λ („lambda“). Dieser ist bei der Poisson-Verteilung gleich der Varianz (Quadrat der Standardabweichung): $\lambda = s^2$. Traditionell sagt man, dass die Poisson-Verteilung die Wahrscheinlichkeitsverteilung für seltene Ereignisse ist. Der Grund hierfür ist, dass der häufigste Wert, in unserem Beispiel also 2, 30% der Werte stellt. Wenn man höhere Werte haben will, so muss man die Verteilung nach links verschieben, was aber dazu führt, dass die Anzahl an 0-Werten zunimmt. D.h., es ist nicht möglich, Ereignisse größer Null mit einer Häufigkeit von über 38% zu erhalten. Dies ist in Abbildung 2.3 dargestellt.

Wir sehen auch, dass die Poisson-Verteilung für hohe λ-Werte sich einer Normalverteilung annähert. Dies ist der Fall für alle Verteilungen, und wird als „Zentrales Grenzwerttheorem“ (*central limit theorem*, CLS) bezeichnet. Für den anwendenden Statistiker bedeutet dies, dass bei hohen Mittelwerten eine Normalverteilung die Datenverteilung genügend genau annähert.

Für die Poisson-Verteilung etwa wäre dies ein Beispiel: Man stelle sich also vor, wir zählten die Seiten von Büchern. Nun wird es *per definitionem* kein Buch unter 100 Seiten geben, die meisten werden irgendetwas zwischen 250 und 400 Seiten haben, also etwa einen Mittelwert von 330. Da schon bei $\mu = 10$ die Annäherung an die Normalverteilung recht ordentlich ist, ist bei dem Bücherbeispiel der Poisson-Ursprung der Daten nicht mehr erkennbar.

2.3.3. Die Binomialverteilung

Die dritte wichtige Verteilungsform ergibt sich aus wiederholten Zufallsexperimenten, wie etwa dem Wurf eines Würfels oder einer Münze. Fangen wir mit der Münze an, deren Wurf zu dem Ereignis „Kopf“ (K) oder „Zahl“ (Z), jeweils in der Hälfte aller Fälle führt. Dafür können wir schreiben: $P(\mathsf{K}) = P(\mathsf{Z}) = 0.5$. Der entscheidende Schritt zum Verständnis der sich hier entwickelnden **Binomialverteilung** ist folgender: Interessant ist nicht der wiederholte Wurf *einer* Münze, sondern mehrerer. Wenn wir also gleichzeitig 4 Münzen werfen, so können wir etwa folgenden Sequenzen erhalten: KKZK, KZKZ, KKZZ, ZKKK, ZZZK, usw., insgesamt $n^k = 2^4 = 16$ verschiedene. Die Wahrscheinlichkeit, 3 Mal ein Ereignis und nur 1 Mal das

andere zu erhalten ($P(3+1)$) berechnet sich als das Produkt der Wahrscheinlichkeit der Einzelereignisse (Formel 2.2): $0.5^3 * 0.5 = 0.0625$. Jedes Ereignis jeweils 2 Mal zu erhalten ($P(2+2)$) ist entsprechend $0.5^2 \cdot 0.5^2 = 0.0625$. Genauso ist die Wahrscheinlichkeit 4 Mal das gleiche Ereignis zu erhalten ($P(4)$) $= 0.0625$.

Nun gibt es unter den 2^4 möglichen Sequenzen nur 2 mit 4 gleichen Ereignissen (KKKK und ZZZZ), 8 für Ereignisverhältnisse von 3 zu 1 (KKKZ, KKZK, KZKK, ZKKK, ZZZK, ZZKZ, ZKZZ, KZZZ) und 6 für Paare (KKZZ, KZZK, ZZKK, ZKKZ, KZKZ, ZKZK). Entsprechend können wir nun unsere Erwartungen für 4er, 3+1er und Paare berechnen als das Produkt der Wahrscheinlichkeit ein Ergebnis zu erhalten und der Häufigkeit dieses Ereignisses: $0.0625 \cdot 2 = 0.125, 0.0625 \cdot 8 = 0.5, 0.0625 \cdot 6 = 0.375$ (sie addieren sich natürlich zu 1).

Diese Berechnungen werden etwas mühselig für größere Zahlenmengen oder unterschiedliche Ereigniswahrscheinlichkeiten p. Beim Würfel beträgt diese $p_{\text{Würfel}} = 1/6$, für Sonnenschein in Darmstadt $p_{\text{Sonne in Darmstadt}} = 298/365$, usw. Insgesamt kann man die Binomialverteilung mit folgender Formel für beliebige Ereigniswahrscheinlichkeiten p und Probengrößen n berechnen:

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x} \tag{2.9}$$

Mittelwert und Varianz der Binomialverteilung berechnen sich als

$$\bar{x} = np \tag{2.10}$$
$$s^2 = npq \tag{2.11}$$

In unser Münzenbeispiel ist $p = 0.5$, $n = 4$ und $x = 1, 2, 3$ oder 4. Für $x = 2$ erhalten wir also (genau wie vorher zu Fuß):

$$p(2) = \binom{4}{2} 0.5^2 (1-0.5)^2 = 6 \cdot 0.5^4 = 0.375$$

Abbildung 2.4 vermittelt einen Eindruck über die Binomialverteilung für unterschiedliche Ereigniswahrscheinlichkeiten. Nur die für $p = 0.5$ ist symmetrisch.

Diese Funktion wird in R von der Funktion `dbinom(x, n, p)` ausgeführt.
Wenn wir darüberhinaus noch wissen wollen, wie wahrscheinlich es ist, dass die Daten einer Binomialverteilung entstammen, so können wir, analog zum Test auf Normalverteilung, den verallgemeinerten Kolomogorov-Smirnov-Test benutzen. Der Syntax ist ebenfalls ähnlich: wir müssen die Wahrscheinlichkeitsverteilung und ihre Parameter spezifizieren.

```
> survival <- c(12, 13, 14, 15, 16)
> total <- rep(20, 5)
> ks.test(survival, "pbinom", prob = (sum(survival)/sum(total)),
+     size = total)

        One-sample Kolmogorov-Smirnov test

data:  survival
D = 0.2277, p-value = 0.9064
alternative hypothesis: two.sided
```

Die Hypothese, dass die Daten `survival` *nicht* einer Binomialverteilung mit einem p von `sum(survival) /sum(total)` entstammen, muss mit einer Wahrscheinlichkeit von 0.96 zurückgewiesen werden.

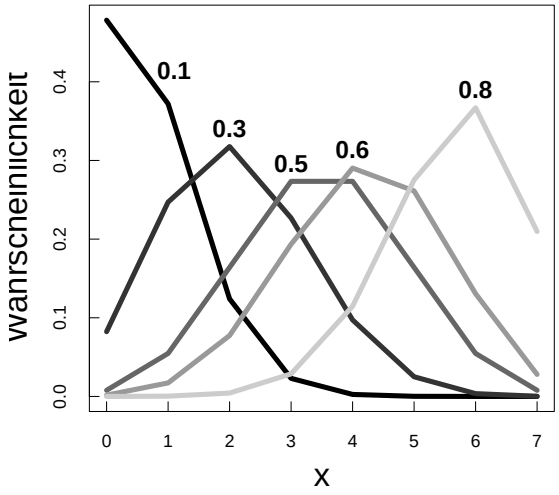


Abbildung 2.4.: Binomialverteilungen für verschiedene Ereigniswahrscheinlichkeiten p. Ebenso wie die Poisson-Verteilung ist auch die binomiale eine diskrete Verteilung, die Linien verbinden also Werte mit ganzzahligem x. $n = 7$

2.3.4. Weitere Verteilungen

Neben diesen drei wichtigsten gibt es zahllose weitere Verteilungen. Sie teilen sich in die Gruppe der Test-Verteilungen (X^2, t, F, usw.), die der mathematisch-deduzierten (exponential, Gamma, Beta, Cauchy, lognormale, logistische, uniforme, Weibull, usw.) und die der erweiterten Binomialverteilung (Bernoulli, negativ-binomiale, geometrische, hypergeometrische, multinomiale, usw.). Sie alle haben eine spezielle Anwendung, z.B. die uniforme für Zufallszahlen, die Hypergeometrische für Wahrscheinlichkeiten à la binomial *ohne* Zurücklegen, usw.

Hier wollen wir die wichtigsten weiteren Verteilungen nur kurz erwähnt haben. Für ausführlichere Betrachtungen sei auf Evans et al. (2000) verwiesen.

Die γ (Gamma)-Verteilung

Wenngleich die γ-Verteilung für den Laien etwas abschreckend definiert ist, so hat sie doch die schöne Eigenschaft nach rechts ein ausgezogenes Ende zu besitzen. Diese Art von Verteilung ist sehr häufig bei biologischen Daten (was dazu führt, dass die Varianz mit dem Mittelwert zunimmt), und deshalb kann die γ-Verteilung gelegentlich bei nicht-normalverteilten Daten als Fehlerverteilung in einem GLM benutzt werden.

Die γ-Verteilung hat zwei Parameter, von denen der eine verwirrenderweise entweder b ("scale") oder $\lambda = 1/b$ ("rate") heißt. Der andere Parameter der γ-Verteilung heißt c ("shape parameter"). Diese Parameter sind echt größer Null. Die Verteilungsfunktion hier hinzuschreiben ist zwar verführend (da sie mit dem griechischen Großbuchstaben Γ ein sehr hübsches Symbol beinhaltet), aber wenig hilfreich für unsere Ziele. Hier sei auf das exzellent Verteilungsbuch von Evans et al. (2000) oder verschiedene Internetresourcen hingewiesen[4].

Mittelwert und Varianz der γ-Verteilung berechnen sich als

$$\bar{x} = bc \tag{2.12}$$
$$s^2 = b^2 c \tag{2.13}$$

[4] siehe Wikipedia oder mathworld

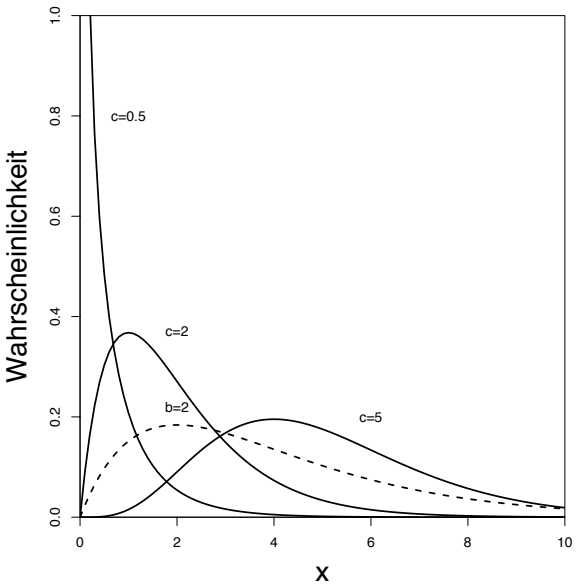


Abbildung 2.5.: γ-Verteilungen für verschiedene Verteilungsparameter c (durchgezogene Linien: $b = 2$) und b (gestrichelte Linie: $c = 2$).

Ausgesprochen wichtig ist, dass die γ-Verteilung nur für das Interval $[0, \infty]$ definiert ist (siehe auch Abb. 2.5). Für ein GLM in R bedeutet dies, dass alle Werte echt größer Null sein müssen!

Gestutzte Verteilungen

Die häufigste gestutzte (oder abgeschnittene; *truncated*) Verteilung ist die gestutzte Poisson-Verteilung: Nehmen wir an, wir zählen Eier in Vogelnestern. Kein Vogel baut ein Nest um kein Ei hineinzulegen, d.h. unsere Daten fangen nicht mit Null an, sondern mit eins. Anstelle einer echten Poisson-Verteilung haben wir dann eine, die erst bei 1 beginnt, eben eine gestutzte Verteilung.

Für eine 0-gestutzte Poisson-Verteilung gilt:

$$p(x) = \frac{e^{-\lambda}\lambda^x}{x!(1 - e^x)} \tag{2.14}$$

mit $x = 1, 2, 3, \ldots$. Mit ihr lassen sich entsprechende Erwartungswerte errechnen und mit den gefundenen Werten vergleichen.

2.4. Wichtige Parameter und Statistiken

An dieser Stelle wird eine wichtige Unterscheidung zwischen **Parametern** und **Statistiken** fällig. Wenn wir aus einer Grundgesamtheit (= zugrunde liegende Verteilung) zufällig Proben ziehen, so wird diese **Grundgesamtheit** von **Parametern** beschrieben, die **Stichprobe** hingegen von **Statistiken**. Wenn wir einen Datensatz beschreiben wollen, so sind wir an verschiedenen Parametern (der Grundgesamtheit) interessiert. Wir nutzen aber die Stichprobe, um eine Abschätzung dieser Parameter auf Grundlage der Statistik vorzunehmen. Im folgenden wird dies deutlicher.

2.4.1. Maße für das Zentrum einer Verteilung

Vor allem interessiert uns meistens ein Maß für das Zentrum der Verteilung. Der häufigste davon ist das arithmetische Mittel (kurz: **Mittelwert**; *mean*):

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \tag{2.15}$$

Wir berechnen also einen Mittelwert $\bar{y}$ für unsere Datenpunkte, und sagen damit etwas über den Mittelwert der Grundgesamtheit (μ), nämlich $\mu \approx \bar{y}$. Dies würde vermutlich kein Mathematiker so schreiben, sondern stattdessen sagen: Der Mittelwert der Stichprobe ($\bar{y}$) ist ein „erwartungstreuer" (*unbiased*) Schätzer des Mittelwertes der Grundgesamtheit (μ).

Es gibt zwei weitere Maße für das Verteilungszentrum: der **Median** (*median*) gibt den Wert an, der genau in der Mitte des sortierten Datensatzes liegt (bei einem geradzahligen Stichprobenumfang: Mittelwert der zwei Werte um die Mitte):

$$\text{Median} = \begin{cases} y_{(n+1)/2}, & \text{für ungerade } n; \\ (y_{n/2} + y_{(n/2)+1})/2, & \text{für gerade } n. \end{cases} \tag{2.16}$$

Der **Mode** (*mode*) hingegen ist der Wert, der am häufigsten gemessen wurde. Haben wir beispielsweise folgende Werte gemessen: $\{3, 4, 5, 6, 7, 8, 4, 5, 6, 3, 5, 7\}$, so ist der Mode = 5.

Während die Funktionen `mean` und `median` in R zur Verfügung stehen, muss der Mode zu Fuß berechnet werden (die Funktion `mode` ist bereits belegt!):

```
> xx <- c(3, 4, 5, 6, 7, 8, 4, 5, 6, 3, 5, 7)
> my.mode <- function(x) {as.numeric(names(which.max(table(x))))}
> my.mode(xx)

[1] 5
```

2.4.2. Maße für die Breite der Verteilung

Neben einem Maß für das Zentrum eines Datensatzes interessiert uns häufig auch die Breite der Verteilung, angegeben für Normalverteilungen durch den Parameter Standardabweichung (σ; *standard deviation*), bzw. geschätzt als Datensatzstatistik Standardabweichung (s). Da die Standardabweichung die Wurzel aus der Varianz (s^2; *variance*) ist, hier erst die Formel für die Varianz der Daten um den Mittelwert $\bar{y}$ (Gleichung 2.17, dann für die Standardabweichung der Daten vom Mittelwert (Gleichung 2.18):

$$\text{Varianz} \qquad s^2 = \sum_{i=1}^{n} \frac{(y_i - \bar{y})^2}{n-1} \tag{2.17}$$

$$\text{Standardabweichung} \qquad s = \sqrt{\sum_{i=1}^{n} \frac{(y_i - \bar{y})^2}{n-1}} \tag{2.18}$$

Mit diesen Formeln schätzen wir also (für eine Normalverteilung) den Mittelwert μ der Grundgesamtheit durch Berechnung des Mittelwertes der Stichprobe ($\bar{y}$). Dito für die Varianz/Standardabweichung der Grundgesamtheit (σ^2, σ), geschätzt durch die Varianz/Standardabweichung der Stichprobe (s^2, s).

2.4.3. Güte der Parameterschätzung

Desweiteren wollen wir ein Gütemaß für die Schätzung des Stichprobenmittelwerts $\bar{y}$ haben. Da die Genauigkeit mit zunehmender Stichprobengröße zunimmt, erwarten wir also ein n in der Formel. Es haben sich drei verschiedene Maße durchgesetzt: der Varianzkoeffizient (CV;

coefficient of variance), der Standardfehler des Mittelwertes (SE; *standard error of the mean*) und das 95% Konfidenzinterval um den Mittelwert (95%-CI; *confidence interval*).

Der Varianzkoeffizient erlaubt den Vergleich unterschiedlicher Datensätze, da er unabhängig von den Absolutwerten ist:

$$CV = s/\bar{y} * 100 \tag{2.19}$$

Ein CV-Wert von 10% ist entsprechend sehr niedrig, einer von 140% weißt hingegen auf eine hohe Messungenauigkeit hin.

Der Standardfehler besitzt einige Attraktivität, weil er die Visualisierung eines t-Tests ist. Wenn die Standardfehlerbalken zweier Mittelwerte sich nicht überlappen, so sind diese Mittelwerte signifikant unterschiedlich. Allerdings gilt dies nur für normalverteilte Daten und ein-faktorielle Tests (siehe Abschnitt 4.1.2). Er wird wie folgt berechnet:

$$SE = s/\sqrt{n} \tag{2.20}$$

Im Unterschied zu obigen beiden Mittelwertgütemaßen gibt das 95% Konfidenzinterval die Grenzen an, in denen mit 95%er Wahrscheinlichkeit der Grundgesamtheitsmittelwert μ liegt.[5] Es wird vor allem für nicht-normalverteilte Daten eingesetzt (die obigen beiden Maße nehmen zumindest eine symmetrische Verteilung der Daten um den Mittelwert an). Die Berechnung erfolgt ganz unterschiedlich, etwa über *bootstrapp* (siehe 2.5.2) oder, wenn die zugrundeliegende Verteilung bekannt ist, parametrisch.

Der Berechnung des 95%-CIs für normalverteilte Daten liegt die Stichprobenverteilungsfunktion t zugrunde. Diese t-Funktion ist identische mit der Normalverteilung, wenn die gesamte Grundgesamtheit erfasst wurde, ist aber sonst breiter. Dies reflektiert die Unsicherheit der Statistik, also beim Schätzen der exakten Parameter μ und σ^2. Die folgende Formel gibt die 95% Konfidenzintervalle für normalverteilte Daten an.

$$\bar{y} - t_{0.025(n-1)}\frac{s}{\sqrt{n}} \leq \mu \leq \bar{y} + t_{0.025(n-1)}\frac{s}{\sqrt{n}}, \tag{2.21}$$

wobei $t_{0.025(n-1)}$ die Quantilfunktion der t-Verteilung für einen P-Wert von 0.025 bei $n-1$ Freiheitsgraden ist. Wir benutzen 0.025 für den oberen und den unteren Teil der Verteilung, so dass insgesamt 5% der Verteilung außerhalb der gewählten Werte liegen.

Fangen wir mit der Annahme normal-verteilter Daten an. Dann berechnet R die CIs mittels der Funktion `t.test`. über die Option `conf.level=0.99` (Grundeinstellung ist 0.95) können wir das Konfidenzinterval bestimmen. Um nicht den ganzen `t.test` *output* zu erhalten, wählen wir nur das Attribut `$conf.int` (`$conf.int[1:2]` gibt nur die CI-Werte aus).

```
> xx <- c(3, 4, 5, 6, 7, 8, 4, 5, 6, 3, 5, 7)
> t.test(xx, conf.level = 0.99)$conf.int

[1] 3.813199 6.686801
attr(,"conf.level")
[1] 0.99
```

Nicht-normalverteilte Daten (z.B. Poisson oder binomial) lassen sich nicht so leicht fassen. Der Berechnungsansatz hier ist etwas umständlich, und führt über das *likelihood*-Profil der entsprechenden Dichtefunktion (eine hervorragende Einführung zu diesem Thema ist Hilborn and Mangel 1997). Für die Poisson-Verteilung stellt sich die Dichte- (= *likelihood*)-Funktion so dar:

$$\mathscr{L} = \prod_{i=1}^{n} \frac{\lambda^{y_i} e^{-\lambda}}{y_i!} = \lambda^{\sum y_i} e^{-n\lambda} \tag{2.22}$$

[5] Wieder etwas verwirrend! Bei einer Normalverteilung mit den Parametern μ und σ liegen 95% aller Werte zwischen $\mu - 1.96\sigma$ und $\mu + 1.96\sigma$. Hier beschäftigen wir uns aber mit der Güte des Schätzers, nicht mit der Werteverteilung der Grundgesamtheit. Es ist etwas unglücklich, dass Parameter und Schätzerstatistik immer den gleichen Namen haben.

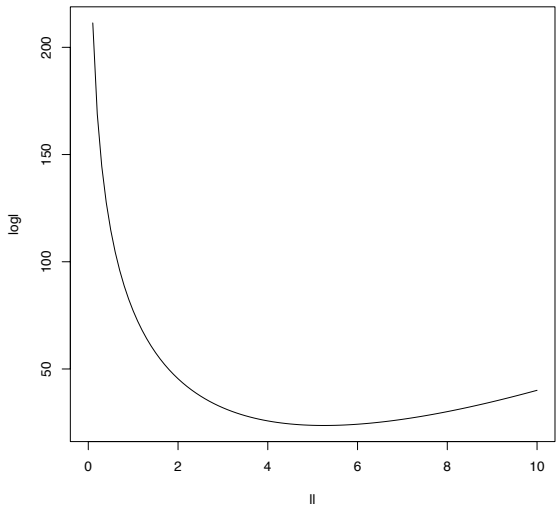


Abbildung 2.6.: −log-*likelihood*-Profil für die Bestimmung eines Mittelwertes für den Datensatz `xx`. Der Wert von `ll`, an dem die Funktion ihr Minimum hat, ist derjenige Mittelwert, der für `xx` am plausibelsten ist.

Bei gegebenen Daten (y_i) beschreiben natürlich unterschiedliche Mittelwerte λ die Daten verschieden gut. Indem wir nach dem Maximalwert dieser Beschreibungsgüte (*likelihood*; häufiger dem Minimalwert der negativen log-*likelihood*) suchen, finden wir auch den optimalen Verteilungsparameter λ^*. Diese Suche ist nur iterativ möglich.

Für Poisson-verteilte Daten benutzen wir folgenden Syntax, um diesen Vorgang zu visualisieren (siehe dabei entstehende Abb. 2.6):

```
> xx <- c(3, 4, 5, 6, 7, 8, 4, 5, 6, 3, 5, 7)
> ll <- seq(0, 10, length = 101)
> loglik.pois <- function(x) {
+     -sum(log(dpois(x = xx, lambda = x)))
+ }
> logl <- sapply(ll, loglik.pois)
> plot(ll, logl, type = "l")
> max.loglik <- optim(par = 2, fn = loglik.pois)
> max.loglik$par

[1] 5.25

> loglik.CI <- function(x) {
+     abs(loglik.pois(x) - (loglik.pois(max.loglik$par) + 1.92))
+ }
> lower.CI <- optim(par = 1, fn = loglik.CI)$par
> lower.CI

[1] 4.058250

> upper.CI <- optim(par = 100, fn = loglik.CI)$par
> upper.CI

[1] 6.65493
```

Was hier abläuft ist im wesentlichen folgendes: `dpois` ist die Dichte- bzw. *likelihood*-Funktion der Poisson-Verteilung. Entsprechend berechnet die neu definierte Funktion `loglik.pois` die negative log-*likelihood*,

nachdem die Poisson-Verteilung an den gemessenen xx-Werten gefittet wurde. Das entstehende Bild nennen wir log-*likelihood*-Profil, das Minimum gibt den wahrscheinlichsten Wert für den Mittelwerte λ an. Dieses Minimum wird mittels der optim-Funktion gefunden (iterativ). Die Konfidenzlimits liegen nun 1.92 über diesem Minimum[6], und werden ebenfalls iterativ gefunden (Hilborn and Mangel 1997), indem einmal von links (par=1) und einmal von rechts (par=100) die absolute Differenz zwischen dem um 1.92 verschobenen Minimum und dem aktuellen Wert minimiert wird.

Alternativ kann man hier die Stichprobe *bootstrappen* (siehe Abschnitt 2.5.2 auf Seite 24) und daraus Konfidenzintervalle berechnen. In R steht neben dem hier benutzen *package* **bootstrap** auch noch ein *package* **boot** zur Verfügung. In vielen Fällen müssen wir uns aber auch die Funktionen für den bootstrap selbst schreiben, wobei dann die R-function sample die wichtigste Rolle spielt.

```
> library(bootstrap)
> xx <- c(3, 4, 5, 6, 7, 8, 4, 5, 6, 3, 5, 7)
> sort(bootstrap(xx, nboot = 1000, mean)$thetastar)[c(75, 975)]

[1] 4.583333 6.166667
```

2.4.4. Ein Beispiel

Spielen wir die Berechnung der verschiedenen Parameterschätzer doch einmal an einem fiktiven Datensatz durch. Der Datensatz ist log-normal-verteilt, um die Unterschiede zwischen den verschiedenen Zentralitätsparametern zu verdeutlichen (das Ergebnis ist in Abbildung 2.7 zu sehen).

```
> set.seed(5)
> bsp <- rlnorm(101, meanlog = 0.75, sdlog = 0.2)
> hist(bsp, breaks = seq(1, 3.5, by = 0.25), freq = F, xlab = "",
+     ylab = "Dichte", xlim = c(1, 3.5), col = "grey80", main = "")
> my.mode <- function(x) {
+     gnu <- hist(x, breaks = seq(1, 3.5, by = 0.25), freq = F,
+         xlab = "", ylab = "Dichte", xlim = c(1, 3.5), plot = F)
+     gnu$mids[which(gnu$counts == max(gnu$counts))]
+ }
> curve(dlnorm(x, meanlog = 0.75, sdlog = 0.2), add = TRUE, lwd = 2,
+     lty = 1)
> curve(dnorm(x, mean(bsp), sd(bsp)), add = TRUE, lwd = 2, lty = 2)
> mean(bsp)

[1] 2.161315

> median(bsp)

[1] 2.070023

> my.mode(bsp)

[1] 1.875

> points(mean(bsp), -0.02, pch = 17, cex = 2, col = "black")
> points(median(bsp), -0.02, pch = 17, cex = 2, col = "grey50")
> points(my.mode(bsp), -0.02, pch = 17, cex = 2, col = "grey70")
> CIlow <- t.test(bsp)$conf.int[1]
> CIup <- t.test(bsp)$conf.int[2]
> for (k in seq(CIlow, CIup, length = 100)) {
+     lines(c(k, k), c(0, dnorm(k, mean(bsp), sd(bsp))), pch = 3,
+         cex = 3, col = "grey30")
+ }
> sort(bootstrap(bsp, nboot = 1000, mean)$thetastar)[c(75, 975)]
```

[6] Woher diese 1.92 kommt? Aus dem sogenannten *likelihood-ratio*-Test, dessen Werte X^2-verteilt sind. Der kritische X^2-Wert (für einen Freiheitsgrad, da ein Parameter) ist aber gerade 3.84, also 1.92 hüben und drüben des Minimums (Hilborn and Mangel 1997, S. 154f.).

```
[1] 2.102294 2.240263
```

In diesem Fall sind die *bootstrap*-Fehler des Mittelwertes sehr ähnlich den `t.test`-Schätzern.

```
> sample.95 <- numeric(2)
> sample.95[1] <- min(sort(bsp)[-c(1, 2, 3, 99, 100, 101)])
> sample.95[2] <- max(sort(bsp)[-c(1, 2, 3, 99, 100, 101)])
> para.95 <- qnorm(c(0.05, 0.95), mean(bsp), sd(bsp))
> lines(c(sample.95[1], sample.95[1]), c(0, dnorm(sample.95[1],
+     mean(bsp), sd(bsp))), lwd = 3, col = "white")
> lines(c(sample.95[2], sample.95[2]), c(0, dnorm(sample.95[2],
+     mean(bsp), sd(bsp))), lwd = 3, col = "white")
```

Beachte, dass die Grenzen, die tatsächlich 95% der Daten beinhalten (weiß), nicht symmetrisch um den Mittelwert sind.

```
> lines(c(para.95[1], para.95[1]), c(0, dnorm(para.95[1], mean(bsp),
+     sd(bsp))), lwd = 3, col = "black")
> lines(c(para.95[2], para.95[2]), c(0, dnorm(para.95[2], mean(bsp),
+     sd(bsp))), lwd = 3, col = "black")
```

Die mittels der angenäherten Normalverteilung vorhergesagte 95%ige Datenspanne (schwarz) hingegen ist symmetrisch. Sie unterschätzt die Linkslastigkeit der Daten.

Die oben besprochenen Schätzer des Verteilungszentrums (Mittelwert und Median) sind sogenannte L-Schätzer (*L-estimators*). Wenn wir alle Werte ihrer Größe nach ordnen (von klein nach groß), so berechnen sich L-Schätzer als eine lineare Kombination (Summierung) der gewichteten Daten. Im Falle des Mittelwertes ist die Wichtung $1/n$, im Falle des Medians 0, für alle Werte bis auf den mittleren, für den die Wichtung 1 ist. D.h., nicht die Wichtung ist linear, sondern die Kombination der gewichteten Daten.

Desweiteren gibt es M- und R-Schätzer. Beim M-Schätzer nimmt die Wichtung der Daten vom Mittelwert nach außen ab (etwa Huber M-Schätzer, implementiert in R im *package* **MASS** als `huber`). R-Schätzer benutzen nicht die Datenwerte, sondern nur ihren Rang. Sie werden häufiger unter Transformationen oder nicht-parametrische Statistik gefasst als unter dem Namen R-Schätzer. Ihr „häufigster" ist der Hodges-Lehmann-Schätzer für die Differenz zweier Stichproben. Nur der Vollständigkeit halber kurz das Prinzip des HL-Schätzers (derzeit in R nur implizit in `wilcox.test` implementiert): Man berechnet die Differenz von jedem Wert x_i der einen und jedem Wert y_j der anderen Stichprobe. Somit werden $n \cdot m$ Differenzen berechnet. Der Median dieser Differenzen ist der HL-Schätzer des Medians der Unterschiede zwischen x und y.

2.5. Methoden der Parameterschätzung

2.5.1. Schätzung von Verteilungsparametern

Bei der **Schätzung der Parameter** geht es darum, den Wert einer Modellkonstanten zu finden, der das Modell den Daten am nächsten bringt. Für einen Datensatz x wollen wir dasjenige Modell finden, das den Datensatz am wahrscheinlichsten macht. Dies ist der Fall, wenn die Wahrscheinlichkeit, einen Wert unter einem Modell mit Parameter(n) θ zu beobachten, für alle Datenpunkte x_i maximiert ist. Für jedes Modell berechnet man also die „maximale Dichte". Wir benutzen aber lieber den englischen Ausdruck *maximum likelihood*, nämlich das Produkt ($\prod$) der Wahrscheinlichkeiten jedes einzelnen Datenpunktes (x_i):

$$\mathscr{L}(x|\theta) = \prod_{i=1}^{n} f(x_i|\theta) \tag{2.23}$$

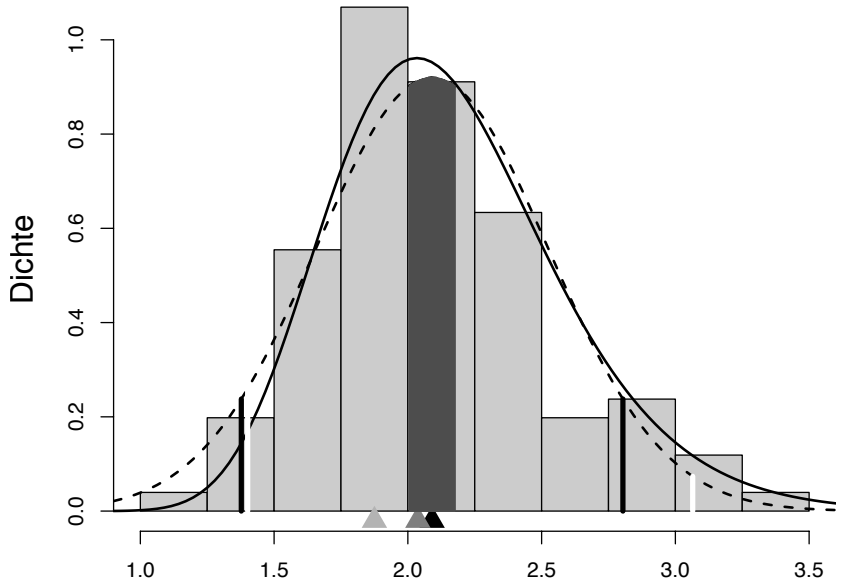


Abbildung 2.7.: Ergebnis des vorgerechneten Beispiels. Die grauen Säulen fassen die Daten zusammen, die durchgezogene Linie ist die Grundgesamtheit, aus der sie stammen, die gestrichelte hingegen stellt die angenäherte Normalverteilung dar. Die Dreiecke unter den Säulen stellen (von rechts nach links) arithmetisches Mittel, Median und Mode dar. Der schwarz eingefärbte Bereich der Säulen ist der parametrische (`t.test`-basierte) Konfidenzbereich der Mittelwertschätzung. Die weißen Strich umschließen 95% der Daten, während die benachbarten schwarzen Linien 95% der vorhergesagten (normalverteilten) Grundgesamtheit umfassen.

Dabei ist $f(.)$ eine noch näher zu spezifizierenden Funktion mit den Parametern θ, die die Datenpunkte x_i annähert.

Noch mal langsam: Jeder Datensatz kann durch verschiedene Funktionen angenähert werden. In jedem Modell befinden sich ein paar Parameter, die beispielsweise die Form einer Kurve durch die gemessenen Punkte beschreibt. Im einfachsten Fall liegen alle gemessenen Punkt parallel zur x-Achse (auch Abszisse genannt), und das Modell hat die Form $y = c$, wobei c eine Konstante ist. Für Punkte, die einen linearen Zusammenhang nahelegen, müssen für die Regressionsgerade $y = ax + c$ zwei Parameter geschätzt werden: das c des vorherigen Falles und zusätzlich die Geradensteigung a. Nächstkomplizierter ist beispielsweise eine quadratische Form ($y = ax + bx^2 + c$) oder etwa eine ansteigende Sinuswelle ($y = dx\, sin\, ex$). Alle diese Modelle können auf einen Datensatz losgelassen werden, aber manche werden besser passen als andere.

Für jedes dieser Modelle kann nun ein maximale Wahrscheinlichkeit berechnet werden. Die Konstanten der Modelle sind dabei in Gleichung 2.23 durch das θ symbolisiert.

$$\mathscr{L}_1(x|y = c) = \prod_{i=1}^{n} c = nc \tag{2.24}$$

$$\mathscr{L}_2(x|y = ax + c) = \prod_{i=1}^{n} (ax_i + c) = (ax_1 + c)(ax_2 + c) \cdots (ax_n + c) \tag{2.25}$$

$$\mathscr{L}_3(x|y = ax + bx^2 + c) = \prod_{i=1}^{n}(ax + bx^2 + c) = \cdots \tag{2.26}$$

$$\mathscr{L}_4(x|y = dx\ sin\ ex) = \prod_{i=1}^{n}(dx\ sin\ ex) = \cdots \tag{2.27}$$

Produkte von Wahrscheinlichkeiten sind sehr kleine Zahlen (da ja jede Wahrscheinlichkeit ≤ 1 ist). Deshalb benutzt man meist nicht die *likelihood*, sondern ihren Logarithmus, die *log-likelihood*. Aus Gleichung 2.23 wird somit:

$$log\,\mathscr{L}(x|\theta) = log\left[\prod_{i=1}^{n} f(x_i|\theta)\right] = \sum_{i=1}^{n} log f(x_i|\theta) \tag{2.28}$$

Und aus Gleichung 2.26 wird:

$$log\,\mathscr{L}(x|y = ax + bx^2 + c) = \sum_{i=1}^{n} log(ax_i + bx_i^2 + c) \tag{2.29}$$

Für jede Wertekombination von a,b und c kann man nun mit Hilfe eines Computers die *log-likelihood* berechnen. Die Kombination, die den höchsten Wert hat (die *maximum likelihood*, ML) ist das beste Modell. Manchmal kann diese ML algebraisch berechnet werden, aber vor allem bei nicht-normalverteilten Daten sind iterative Berechnungen unumgänglich.[7]

Diese ganzen Formeln sind keine überflüssige Mathematik. Hilborn and Mangel (1997) geben der ML-Methode ein ganzes, einführendes Buch, in dem eine klassische Regression oder Varianzanalyse nicht einmal vorkommt. Sie nutzen ML, um verschiedene ökologische Fragestellungen zu beantworten.

Vor allem aber gibt es eigentlich kein anderes verteilungsbasiertes Schätzverfahren! Das viel häufiger beschriebene Verfahren der geringsten Abweichungsquadrate (*ordinary least squares*, OLS) ist ein Spezialfall der ML für normalverteilte Daten. Um dies zu sehen, muss man sich an die Form der Normalverteilung erinnern (Gleichung 2.4):

$$f(\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Ersetzt man x durch den Regressionsterm (also etwa $ax+c$), so sehen wir, dass die Abweichung von der Regressionsfunktion im Quadrat in der Berechnung der Normalverteilung eingeht. Je geringer dieser Abweichungsterm, desto größer der Wert von $f(.)$. Da in der ML-Formel (Gleichung 2.23) der Wert für $\mathscr{L}$ maximiert werden soll, müssen entsprechend die Parameter a und c so gewählt werden, dass die Abweichungen minimiert werden. Genau dies ist das Prinzip der *ordinary least square* Regression.[8]

Zwar wird das OLS Schätzverfahren auch für andere Verteilungen benutzt, aber dies eigentlich vor allem aus Mangel an ML-Erfahrung. Alle verallgemeinerten linearen Modelle (GLM),

[7] Noch ein Wort zur Verwirrung: Im englischen unterscheidet man zwei Wörter für „Wahrscheinlichkeit“: *likelihood* und *probability*. Zum einen bestehen Unterschiede in der Philosophie der Modellierung: Bei der *likelihood* sind die Daten fest, und die Modellparameter variieren, während bei der *probability* Daten zu einem festen Modell gesucht werden. Anders formuliert: Bei der *likelihood* fragt man, wie wahrscheinlich es ist, dass ein Modell den Daten passt; bei der *probability*, wie wahrscheinlich es ist, dass die Daten einem bestimmten Modell entstammen. Beide sind über Bayes Theorem (Gleichung 2.41) miteinander verknüpft (Hilborn and Mangel 1997, p. 133). Zum anderen ist die Summe der *likelihoods* einer Verteilung (= die Fläche unter einer Verteilung) = 1. Bei einer Normalverteilung mit $s << 1$ muss dafür die *likelihood* an manchen Stellen >1 sein, was natürlich für eine Wahrscheinlichkeit nicht möglich ist. Allgemein sind also *likelihoods* den Wahrscheinlichkeiten nur verwandt, aber ihnen nicht gleich.

[8] Nur der Vollständigkeit halber: Es gibt noch weitere *least square* Regressionen, z.B. *two-stage least square* (2SLS, auch *instrumental-variable regression* genannt) implementiert in R in den *packages* **systemfit**, Funktion `twostage.systemfit` und *package* **sem**, Funktion `tsls`. Das 2SLS-Verfahren wird benutzt, wenn die erklärende Variable (x) mit dem Fehler korreliert ist. Man nennt dann x eine endogene Variable. Für die mathematische Grundlage siehe etwa `http://stat-www.berkeley.edu/~census/ivy.eps`.

die wir auf späteren Seiten sehen werden, nutzen *maximum likelihood*-Schätzungen. Es kann gezeigt werden, dass unter Annahme der Normalverteilung der Daten (genauer: der Residuen) die ML-Verfahren mit dem OLS-Schätzer identisch ist. Da diese Herleitung der OLS recht instruktiv ist, und gleichzeitig das ML etwas entmystifiziert, sei es hier kurz dargestellt:

Wir beginnen mit der Formel für eine Normalverteilung (s. oben oder Gleichung 2.4), und erinnern uns das gilt: $\prod e^x = e^{\sum x}$. Damit können wir die Normalverteilung als Funktion $f(.)$ in Gleichung 2.23 einsetzen:

$$\mathscr{L} = \frac{1}{(\sigma\sqrt{2\pi})^n} e^{-\frac{1}{2\sigma^2}\sum (x_i-\mu)^2}$$

Logarithmieren führt zu:

$$\begin{aligned} log\mathscr{L} &= l(\mu,\sigma) = log\left[\frac{1}{(\sigma\sqrt{2\pi})^n} e^{-\frac{1}{2\sigma^2}\sum (x_i-\mu)^2}\right] && (2.30)\\ &= log\left[\sigma^{-n}(2\pi)^{-n/2}\right] - \frac{1}{2\sigma^2}\sum (x_i-\mu)^2 && (2.31)\\ &= -n\ \log\sigma - \frac{n}{2} log(2\pi) - \frac{1}{2\sigma^2}\sum (x_i-\mu)^2 && (2.32) \end{aligned}$$

Um das Maximum zu finden, setzen wir die Ableitung dieses Terms gleich Null:

$$\frac{d\,l}{d\mu} = 0 = \frac{1}{\sigma^2}\sum (x_i-\mu)$$

Da $\frac{1}{\sigma^2} \neq 0$, muss der Rest Null sein. Somit ist

$$0 = \sum x_i - \sum \mu = \sum x_i - n\mu$$

da μ konstant ist und n mal addiert wird. Nach Umformung sehen wir, dass

$$\mu = \frac{\sum x_i}{n}$$

Für die Varianz differenzieren wir Gleichung 2.32 nach σ, wobei der konstante Term wegfällt, und setzen gleich Null (wir erinnern uns dass die Ableitung von $log\, x = 1/x$ und die Ableitung von $-1/x^2 = 2/x^3$):

$$\frac{d\,l}{d\sigma} = 0 = \frac{-n}{\sigma} + \frac{\sum (x_i-\mu)^2}{\sigma^3}$$

Multiplizieren auf beiden Seiten mit σ^3 führt zu

$$0 = -n\sigma^2 + \sum (x_i-\mu)^2$$

$$\iff \sigma^2 = \frac{\sum (x_i-\mu)^2}{n}$$

Voilà! Auch nach dem Ansatz der ML berechnet sich der Mittelwert der Normalverteilung wie gewohnt als Summe aller Werte durch deren Anzahl, und die Varianz als Summe der Abweichungsquadrate geteilt durch die Anzahl.

Das Gleiche können wir mit Poisson-verteilten Daten machen. Da

$$\mathscr{L}(\mu) = \frac{\lambda^x}{x!e^\lambda}$$

(siehe Gleichung 2.7), ergibt sich für die *likelihood*-Funktion:

$$\mathscr{L}(x_i|\lambda) = \frac{\lambda^{x_1}}{x_1!e^\lambda} \cdot \frac{\lambda^{x_2}}{x_2!e^\lambda} \cdots \frac{\lambda^{x_i}}{x_n!e^\lambda} = \frac{\lambda^{\sum x_n}}{x_1! \cdots x_n! e^{n\lambda}}$$

Wir logarithmieren und erhalten

$$ln\mathscr{L} = -n\lambda + (ln\lambda)\sum x_i - ln(\prod x_i!)$$

Differenzieren nach λ und Nullsetzen ergibt:

$$\frac{d}{d\lambda} ln\mathscr{L} = -n + \frac{\sum x_i}{\lambda} = 0$$

$$\lambda = \frac{\sum x_i}{n}$$

In der Tat ist dies genau die Formel für den Mittelwert Poisson-verteilter Daten.

Im Vergleich zum OLS-Schätzverfahren benutzt entsprechend eine ML Poisson-Regression nicht Abweichungsquadrate. Wir ersetzen analog zum obigen Beispiel λ durch $ax_i + c$. Um $\mathscr{L}$ zu maximieren, müssen a und c so gewählt werden, dass der Term $\ln(\mathscr{L})$ maximiert wird. Dies ist analytisch nicht lösbar (da zwei Unbekannte in nur einer Gleichung), sondern wird vom Computer iterativ berechnet.

Der Vollständigkeit halber sei auch noch die *likelihood*-Funktion für eine Binomialverteilung angeführt:

$$\mathscr{L}(p|n, x_i) = \binom{n}{x_i} p_i^x (1-p)^{n-x_i}$$

Dem aufmerksamen Beobachter wird auffallen, dass obige Formel identisch mit der Verteilungsfunktion der Binomialverteilung ist. Während also die Verteilungsfunktion die Wahrscheinlichkeit der Daten (x_i) (bei gegebener Stichprobengröße n und Eintrittswahrscheinlichkeit p) berechnet, gibt die *likelihood*-Funktion die Wahrscheinlichkeit der Funktionsparameter (bei gegebenen Daten) an[9].

2.5.2. Parameterschätzung mittels verteilungsfreier Methoden

Seit Rechenkapazität kein Problem in der Statistik mehr ist, finden verteilungsfreie Parameterschätzungen vermehrt Anwendung. Verteilungsfrei bedeutet dabei, dass wir nicht eine bestimmte Form der Datenverteilung zugrunde legen (z.B. Normalverteilung oder Poisson-Verteilung), sondern die Daten und ihre Verteilung benutzen.

Um dies zu verstehen, müssen wir eine Schritt zurückgehen. Wofür brauchen wir die Parameter einer Verteilung (parametrisch oder nicht) eigentlich? Wir wollen z.B. wissen, ob ein Datensatz einer Normalverteilung entspricht, oder, im Falle einer Regression, ob eine Poisson-Regression den Datensatz gut erklärt. D.h., uns interessieren nicht vorrangig die Parameter, sondern wir wollen eine Hypothese testen. Oder wir wollen eine Vorhersage machen auf Grundlage des Datensatzes, und die Güte der Vorhersage ist wichtig, nicht aber die Parameter der Formel, mit der wir diese Vorhersage getroffen haben.

In beiden Fällen sind wir aber nicht darauf angewiesen, dass wir die genaue Verteilung der Daten kennen. Stattdessen erlauben es sog. *resampling*-Verfahren die Güte unserer Schätzung zu quantifizieren. Zur Illustration dient folgendes Beispiel:

Wir wollen beispielsweise die Zeit zwischen zwei Geysirausbrüchen wissen[10]. Da die Daten keiner offensichtlichen Verteilung folgen (Abbildung 2.8), bietet sich der Median als Schätzer an.

In diesem Fall beträgt der Wert 76. Aber wie genau ist dieser Schätzer? Wir können eine Dichtekurve über die Daten legen, die sich als Mittelwert (oder anderer Schätzer) von jeweils n Punkten berechnet. Je mehr Punkte in die Berechnung einbezogen werden (je breiter also die Bandbreite des Mittelwertfilters), desto flacher wird die Dichtekurve (Abbildung 2.8). Diese Dichtekurve können wir auch benutzen, um die Zeit zwischen zwei Ausbrüchen zu schätzen.

[9] Siehe z.B. `http://www.cnr.colostate.edu/class_info/fw663/BinomialLikelihood.eps`

[10] Dieses Beispiel stammt aus Venables and Ripley (1994).

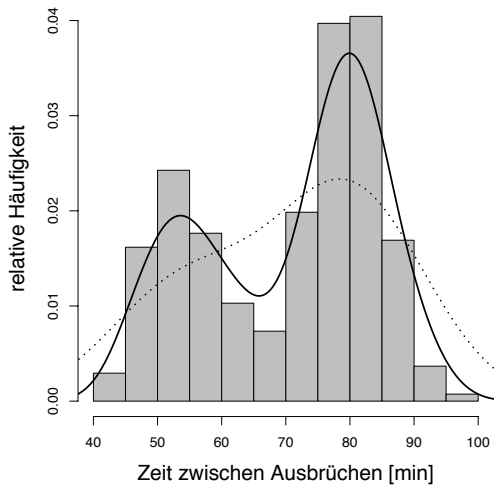


Abbildung 2.8.: Zeitlicher Abstand zwischen zwei Ausbrüchen des Geysirs „Old Faithful“, Yellowstone National Park, USA. Das Histogramm stellt die gemessenen Zeiten dar, die durchgezogene Linie die Dichtekurve (Bandbreite = 4 (durchgezogene Linie) und 10 (gestrichelt)).

Sowohl für eine Bandbreite von 4 als auch von 10 liegt der Median bei 69.5. Welcher Schätzer ist also besser? Der der Rohdaten, oder der der Dichtekurven? Und welcher Fehler mag auf den Schätzern lasten?

2.5.3. Der bootstrap

Hier kommen wir zur Technik des *bootstrap*. Die Idee des *bootstrap* ist, dass man die n gemessenen Werte x zufällig unter Zurücklegen beprobt. Somit liegt dem neu entstehenden Datensatz (üblicherweise auch der Größe n) genau die gleiche Verteilung zugrunde, wie den Originaldaten. Von diesem neuen Datensatz x' können wir ebenfalls den Median berechnen. Und wenn wir diese Prozedur sagen wir 1000 mal wiederholen, können wir den Mittelwert und die Varianz der Mediane berechnen.[11] Aus den nach Wert geordneten *bootstrap samples* können wir dann auch 95% Konfidenzintervalle ableiten (nämlich bei dem 25 und 975 Wert).

In R gibt es für das *bootstrap* eine eigene Funktionen `bootstrap` im *package* **bootstrap**. Dies ist eine Implementierung der Ideen des Standardwerkes zum *bootstrap* (Efron and Tibshirani 1993). Diese Funktionen bietet verschiedene Zusatzoptionen, die hier nicht besprochen werden können. Anschließend machen wir es der Deutlichkeit halber nochmals zu Fuß. Für obiges Beispiel entsteht damit in R folgender Code:

```
> library(bootstrap)
> data(faithful)
> boot.sample <- bootstrap(faithful$waiting, median, nboot = 1000)
```

Leider ist der *output* von `bootstrap` etwas hässlich (tippe einfach `boot.sample`, um ihn anzusehen. Die *bootstrap*-Proben sind als `boot.sample$thetastar` abrufbar.

```
> mean(boot.sample$thetastar)

[1] 75.643

> sd(boot.sample$thetastar)

[1] 0.9762192
```

[11] Dank der Theorie der großen Stichproben wissen wir, dass diese Mediane annähernd normalverteilt sind.

Und hier die umständlichere aber fürs Verständnis einfachere Variante:

```
> median(faithful$waiting)

[1] 76

> set.seed(10)
> newdata <- numeric(1000)
> for (i in 1:1000) newdata[i] <- median(sample(faithful$waiting,
+     length(faithful$waiting), replace = TRUE))
> mean(newdata - median(faithful$waiting))

[1] -0.32

> sd(newdata)

[1] 1.022813

> hist(newdata, breaks = 8)
> CI.median <- sort(newdata)[c(25, 975)]
> CI.median

[1] 73 77
```

Tatsächlich sehen wir, dass der Mittelwert der Mediane nur wenig von dem ursprünglichen Median abweicht, und zwar um etwa -0.36, bei einer Standardabweichung von etwa 1 (die exakten Werte variieren natürlich je Durchlauf). Wenn wir uns das Histogramm der Mediane anschauen (hier nicht abgebildet, wird aber mittels obrigen Codes in R erzeugt), so sehen die Daten recht normalverteilt aus. (Bei mehr als 8 Kategorien scheinen halb-minütige Abstände unterrepräsentiert zu sein, was daran liegt, dass die Originaldaten ganzzahlig sind.) `CI.median` gibt uns 95% Konfidenzintervalle.

2.5.4. Das jackknife

Es gibt noch ein „Gegenstück“ zum *bootstrap*, das *jackknife*. Dieses misst die Güte eines Schätzers (θ^*) anhand seiner Sensibilität gegenüber Datenveränderungen, indem die *jackknife*-Prozedur jeden Datenpunkt einzeln entfernt, und dann den Schätzer θ^*_{-i} neu berechnet. Schließlich wird dann der *jackknife*-korrigierte Schätzer ($\tilde{\theta}_{\text{jack}}$) berechnet[12]:

$$\tilde{\theta}_{\text{jack}} = n\theta^* - \frac{(n-1)}{n}\sum_{i=1}^{n}\theta^*_{-i} \tag{2.33}$$

Die Standardabweichung auf dem korrigierten Schätzer berechnet sich als

$$SD_{\hat{\theta}} = \sqrt{\frac{n-1}{n}\sum(\theta^*_{-i} - \hat{\theta}^*)^2} \tag{2.34}$$

wobei $\hat{\theta}^*$ der Mittelwert der *jackknife*-Daten ist.

In R gibt es für das *jackknife* eine eigene Funktion `jackknife` im *package* **bootstrap**. Hier machen wir es aber ebenfalls der Deutlichkeit halber zunächst zu Fuß. Eine Augenblick der Reflexion lässt uns erkennen, dass wir nicht den Median sinnvoll als Schätzer jackknifen können: Bei 272 Datenpunkten verändert sich der zentrale Wert natürlich nahezu gar nicht, egal welchen Datenpunkt wir eliminieren. Der Median ist also ein extrem robuster Schätzer. Um das Prinzip des *jackknife* illustrieren zu können, benutzen wir das arithmetische Mittel.

```
> data(faithful)
> wait <- faithful$waiting
> mean(wait)
```

[12] Dies ist einfach der Mittelwert der *jackknife* θ^*_{-i}.

```
[1] 70.89706

> n <- length(wait)
> jack <- numeric(n)
> for (i in 1:n) jack[i] <- mean(wait[-i])
> mean.jack <- n * mean(wait) - (n - 1)/n * sum(jack)
> SD.jack <- sqrt(((n - 1)/n) * sum((jack - mean(jack))^2))
> mean.jack

[1] 70.89706

> SD.jack

[1] 0.8243164
```

Zum Vergleich: Mittelwert und Standardfehler der *jackknife*-Proben:

```
> mean(jack)

[1] 70.89706

> sd(jack)/sqrt(n)

[1] 0.003041758
```

Hier die gleiche Analyse mittels der entsprechenden R-Funktion.

```
> library(bootstrap)
> jack2 <- jackknife(wait, mean)
> mean(jack2$jack.val)

[1] 70.89706

> jack2$jack.se

[1] 0.8243164
```

BEACHTE: *variable*`$jack.se` ist der Standardfehler der *jackknife*-Proben (und die Standardabweichung des Schätzers), nicht der Standardfehler der Schätzung! Der Standardfehler der Schätzung berechnet sich (wie gezeigt) als Standardabweichung geteilt durch Wurzel aus Stichprobenumfang.

Nach einer anfänglichen Euphorie (siehe etwa Sokal and Rohlf 1995) ist die *jackknife*-Technik heute seltener in Gebrauch. Bei vielen Datenpunkten ist der Wert des einzelnen naturgemäß gering, und somit eine *jackknife*-korrigierte Statistik oft eine Unterschätzung der wahren Variabilität der Daten. Natürlich kann man mehr als einen Datenpunkt von der Schätzung ausnehmen, und sozusagen den *jackknife* bootstrappen. Diese Technik wird dann allgemein als Kreuzvalidierung (*cross-validation*) bezeichnet und soll uns hier nicht weiter interessieren.

Hingegen hat das Gegenstück, *jackknife-after-bootstrap*, recht weite Anwendung gefunden. Nehmen wir an, dass wir eine statistische Analyse durchführen (etwa eine Regression). Nun können wir diese Analyse wiederholt mit zufällig unter Zurücklegen gezogenen Daten wiederholen, um ihre Robustheit zu prüfen. Dies ist dann entsprechend ein *bootstrap* der Analyse. Die *bootstrap*-Analyseergebnisse müssen aber keinesfalls normalverteilt sein, und so liegt es nahe, sie zu jackknifen. Wir erhalten dann robuste Fehlerschätzungen für eine robuste Analyse, was etwas doppelt-gemoppelt wirkt, aber nicht ist.

Die oben bereits vorgestellte Funktion `bootstrap` (siehe wiederum Efron and Tibshirani 1993) berechnet automatisch den *jackknife-after-bootstrap*, wenn als Option `func=mean` benutzt wird.

```
> library(bootstrap)
> x <- 1:30
> y <- 2 * x + 2 + rnorm(30, 0, 15)
> xdata <- cbind(x, y)
> t.intercept <- function(x, xdata) {
+     coef(lm(xdata[x, 1] ~ xdata[x, 2]))[1]
+ }
> results1 <- bootstrap(1:30, 30, t.intercept, xdata, func = mean)
> results1
```

```
$thetastar
 [1] 7.471075 4.728679 7.347634 6.256474 4.971537 6.595669 5.046793 6.582090
 [9] 5.644995 5.906513 4.866621 5.557942 4.887078 5.665790 8.123228 6.419654
[17] 6.136722 6.424393 4.829017 7.027734 4.983329 6.561887 7.294802 7.656352
[25] 4.462863 7.561732 5.802776 5.422696 6.794669 5.794038

$func.thetastar
[1] 6.09416

$jack.boot.val
 [1] 6.374082 6.686976 6.688685 6.250165 5.288295 6.062643 6.090965 6.241435
 [9] 6.430189 6.501408 6.285086 5.887587 5.954695 6.094878 6.549196 6.770500
[17] 6.108882 5.702080 5.815113 6.224004 6.162543 6.272109 5.638173 5.869550
[25] 5.427326 5.659438 6.270584 6.414696 6.186123 6.073885

$jack.boot.se
[1] 1.915768

$call
bootstrap(x = 1:30, nboot = 30, theta = t.intercept, xdata, func = mean)
```

```
> t.slope <- function(x, xdata) {
+     coef(lm(xdata[x, 1] ~ xdata[x, 2]))[2]
+ }
> results2 <- bootstrap(1:30, 30, t.slope, xdata, func = mean)
> results2$jack.boot.se
```

```
[1] 0.04181107
```

Dabei geben `$func.thetastar` und `$jack.boot.se` die gesuchten Mittelwert und Standardfehler an. Leider lässt sich diese Funktion nicht gleichzeitig auf y-Achsenabschnitt und Steigung anwenden.

2.6. Der Ansatz der Bayesische Statistik

Der zentrale Gedanke hinter Bayesischer Statistik ist die Nutzung von Vorinformationen, sei es aus früheren Experimenten oder aus Expertenwissen. Während sich dies einfach und vernünftig anhört, sind die speziellen mathematischen Details alles andere.

2.6.1. Noch mehr Mengenlehre

Dafür müssen wir uns noch ein wenig mehr mit der Notation von konditionalen Wahrscheinlichkeiten auseinandersetzen (Fortsetzung von S. 8). Abbildung 2.9 zeigt die Wahrscheinlichkeit zweier Ereignisse (A und B) als Fläche, die sie im Wahrscheinlichkeitsraum S einnehmen.

Die Wahrscheinlichkeit, dass A eintritt (notiert als $P(A)$) ist also Fläche(A)/Fläche(S). Die Wahrscheinlichkeit, dass A **oder** B eintreten ($P(A \cup B)$) ist damit gegeben als:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \tag{2.35}$$

Da die Schnittmenge von $P(A)$ mit der von $P(B)$ überlappt, müssen wir sie einmal abziehen. (Für die Schnittmenge schreiben wir verkürzt $P(A \text{ und } B) = P(A \cap B)$.)

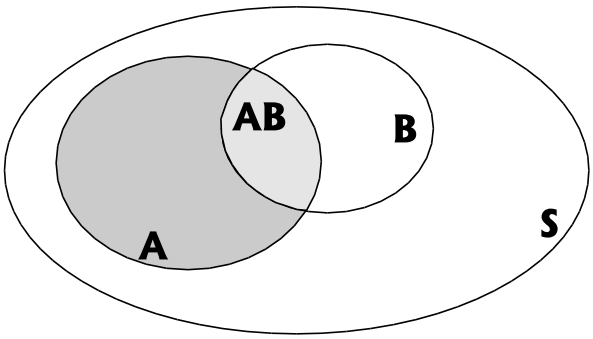


Abbildung 2.9.: Wahrscheinlichkeit des Eintreffens von A, B und AB. S bezeichnet alle die Fläche aller möglichen Ereignisse. Verändert nach Hilborn and Mangel (1997).

2.6.2. Konditionale Wahrscheinlichkeit

Konditionalität drückt sich in unserer Sprache als „wenn“ aus. Wir fragen also: Wie groß ist die Wahrscheinlichkeit, dass A eintritt, wenn wir wissen, dass B eintritt. Dies schreiben wir kurz als $P(A|B)$ und lesen es als: „Wahrscheinlichkeit von A bei Eintreten von B “(engl.: *probability of A, given B*). Diese ist ja nun gleich der Schnittmenge von A und B (AB in Abbildung 2.9), geteilt durch die Fläche von B. (Der Grund, durch die Fläche von A zu teilen, liegt darin, dass wir die Fläche von B an die Stelle von S setzen. Wir wissen ja, dass B eintritt, und fragen nur nach der Wahrscheinlichkeit, dass darüberhinaus noch A eintritt.) Wir schreiben dies als:

$$P(A|B) = P(A \cap B)/P(B) \tag{2.36}$$

Wenn A und B unabhängige Ereignisse sind, der Eintritt des einen also keinen Einfluss auf das Eintreten des anderen hat[13], so folgt:

$$P(A|B) = P(A) \text{ und } P(B|A) = P(B) \tag{2.37}$$

Eine dieser Gleichungen können wir in Gleichung 2.36 einsetzen, und erhalten als Wahrscheinlichkeit des Eintritts zweier unabhängige Ereignisse A und B:

$$P(A \cap B) = P(A) \cdot P(B) \tag{2.38}$$

Wir können Gleichung 2.36 umformen, so dass

$$P(A \cap B) = P(A|B) \cdot P(B) \tag{2.39}$$

Da A und B austauschbare Variablen sind, gilt ebenso:

$$P(A \cap B) = P(B|A) \cdot P(A) \tag{2.40}$$

2.6.3. Bayes Theorem und einfache Beispiele seine Anwendung

Wenn wir jetzt Gleichung 2.40 in Gleichung 2.36 einsetzen, so erhalten wir **Bayes Theorem**:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.41}$$

Diesen Sachverhalt in Worten zu fassen ist nicht wirklich hilfreich. Durch die Herleitung können wir wenigstens einsehen, dass Bayes Theorem wahr ist. Und wir sehen, dass wir die Wahrscheinlichkeit des Eintritts von A, gegeben B, berechnen können aus dem Wissen um die Eintrittswahrscheinlichkeiten von A, B und B gegeben A.

[13] Dies bedeutet, dass sich ihre „Wahrscheinlichkeitsflächen“ *nicht* überlappen.

Wenden wir uns stattdessen einem Beispiel zu (aus Crawley 2002): In einer Zeitung wird der Vorwurf erhoben, dass Frauen (etwa im öffentlichen Dienst) seltener befördert werden als Männer: Nur 2% Frauen wurden befördert! Verglichen mit einem Geschlechterverhältnis in der Bevölkerung von etwa 50:50 scheint dies ein klarer Fall von Diskriminierung zu sein. Nun stehen im weiteren folgende Daten zur Verfügung: Von 200 Beförderungen entfielen nur 4 auf Frauen (also tatsächlich 2%). Es hatten sich, so lesen wir, 40 Frauen um eine Beförderung beworben (also 10% Erfolgschance!), während 3270 Männer einen entsprechenden Antrag gestellt hatten (also eine Erfolgschance von $(200-4)/3270 * 100\% = 5.99\%$!). Damit hat sich die "Diskriminierung" umgedreht[14].

Hier wird eine saubere Wahrscheinlichkeitsformulierung fällig. Die Wahrscheinlichkeit, befördert zu werden, wenn man eine Frau ist, $P(B|F)$, ist 10%. Entsprechend ist $P(B|M) = 6\%$. Was wir wissen wollen, ist die Wahrscheinlichkeit befördert zu werden, wenn man eine Frau ist: $P(B|F)$. Und was gibt die Zeitung? Sie geben die Wahrscheinlichkeit eine Frau zu sein, wenn man befördert wurde: $P(F|B)$. Was wir brauchen, um aus der Schlagzeile den wirklichen gesuchten Wert zu berechnen, ist die Wahrscheinlichkeit, Frau zu sein ($P(F)$), und die befördert zu werden ($P(B)$). Jetzt haben wir alle Zahlen für Bayes Theorem zusammen:

$$P(B|F) = \frac{P(F|B)P(B)}{P(F)} = \frac{0.02 \cdot 40/(3270+40)}{200/(3270+40)} = 0.1$$

Die Umkehrung der konditionalen Wahrscheinlichkeit (statt $P(B|F)$ $P(F|B)$ anzugeben) nennt man den „Trugschluss der transponierten Konditionalität". Hilborn and Mangel (1997, S. 208ff.) bringen ein beunruhigendes Beispiel für eine folgenschwere inkorrekte Berechnung der konditionalen Wahrscheinlichkeit:

1990 wurde ein Britte zu 16 Jahren Gefängnis wegen dreifacher Vergewaltigung verurteilt. Das Urteil fußte auf einer DNA-Übereinstimmung, die, laut Experten, mit einer Wahrscheinlichkeit von 1 in 3 Millionen zufällig war ($P(M|U)$). Kein Zweifel möglich? Wir müssen doch eigentlich nach $P(U|M)$ (unschuldig trotz DNA-*match*) fragen:

$$P(U|M) = \frac{P(M|U)P(U)}{P(M)}$$

$P(M)$ (die Wahrscheinlichkeit eines DNA-*matches*) berechnet sich in Abhängigkeit der Anzahl getesteter Männer. Nehmen wir an, dass alle englischen Männer entsprechenden Alters in einer DNA-Datenbank sind, so dass die Grundgesamtheit hier 10 Millionen Menschen sind. Dann ist $P(M)$ die Wahrscheinlichkeit, schuldig zu sein (also 1/10 Millionen; unkonditional!) plus der Wahrscheinlichkeit unschuldig, aber trotzdem ein positives DNA-match zu haben (also 9 999 999/10 Millionen $\cdot$ 1/3 Millionen). Jetzt setzen wir ein Bayes Theorem ein:

$$P(U|M) = \frac{P(M|U)P(U)}{P(M)} = \frac{1/3 \text{ Mio} \cdot 9\,999\,999/10 \text{ Mio}}{1/3 \text{ Mio} \cdot 9\,999\,999/10 \text{ Mio} + 1/10 \text{ Mio}} = 0.77$$

Mit 77%er Wahrscheinlichkeit wäre unser Angeklagter also unschuldig!

Wahrscheinlich ist die Datengrundlage aber spärlicher, es wurden möglicherweise nur Männer der näheren Umgebung getestet, sagen wir 10 000. Dann verändern sich die Zahlen entsprechend zu:

$$P(U|M) = \frac{1/3 \text{ Mio} \cdot 9\,999/10\,000}{1/3 \text{ Mio} \cdot 9\,999/10\,000 + 1/10\,000} = 0.0033$$

Damit ist unser Angeklagter zwar „hochsignifikant" schuldig, aber aus einer Schuldigkeitswahrscheinlichkeit von 1 zu 3 Millionen wurde eine korrektere von 3 in 1000.

[14] Ob es sich tatsächlich um Diskriminierung von Männer handelt, lässt sich an den Zahlen nicht erkennen. Grundlage der Beförderung ist schließlich die Leistung, nicht der Antrag.

Was wir aus diesem Beispiel lernen, ist *nicht*, dass man so wenig wie möglich DNA-Proben nehmen sollte, um die Wahrscheinlichkeit der Verurteilung eines Unschuldigen zu minimieren, sondern dass viele Wahrscheinlichkeiten, die uns im täglichen Leben begegnen, konditional sind, und somit mit Vorsicht und Ruhe zu betrachten sind!

2.6.4. Bayesische Statistik

Bei weiterführenden Verfahren der Bayesischen Statistik („Bayesische Regression", „Bayesische Mittelwerte") haben gegenüber den üblichen *likelihood*-Verfahren expliziet die Zielsetzung, Vorinformationen mit in die Berechnungen einzubeziehen. Ein anderes Anliegen der Bayesischen Statistik ist, den konkurrierenden Hypothesen zur Erklärung eines Datensatzes unterschiedliche Erwartungen zuzuordnen. Während es eine Vielzahl guter Bücher zum Thema Bayesische Statistik gibt (Berger 1980; Martz and Waller 1982; Gelman et al. 1995), führen nur ganz wenige allgemeine Statistikbücher in diese Form der Analyse ein (etwa Quinn and Keough 2002). Der Grund ist vor allem, dass explorative, „ journalistische" Datenanalyse wenig von dem Bayesischen Ansatz profitiert. Zudem sind die Unterschiede zur konventionellen *likelihood*-Methodik nur im Naturschutz wirklich auffällig (Hilborn and Mangel 1997). Eine kurze Einführung in die Nutzung Bayesischer Statistik für Fragestellungen des Naturschutzes gibt Wade (2000). Grundtenor letzterer Publikation ist, dass wir beim Schutz etwa einer Population seltener Tiere nicht an der Wahrscheinlichkeit, dass die Art ausstirbt interessiert sind, sondern an der Wahrscheinlichkeit, dass die Art *schnell* ausstirbt, so dass wir Handlungsprioritäten aus der Populationsentwicklung ableiten können.

Um zu sehen, wie wir Bayes Theorem für diese Anwendungen nutzen können, müssen wir Gleichung 2.41 etwas umformulieren. Wir suchen die Wahrscheinlichkeit, dass eine Hypothese H_i wahr ist, für gegebene Daten ($P(\mathrm{H}_i|\mathrm{Daten})$, auch *posterior probability* genannt). An die Stelle des Ereignisses B tritt die *likelihood* der Daten für diese Hypothese ($\mathscr{L}(\mathrm{Daten}|\mathrm{H}_i)$). Damit wird aus $P(B) \rightarrow P(Daten)$. Und an die Stelle von $P(A)$ setzen wir eine Eintrittswahrscheinlichkeit, die aus Vordaten bekannt ist (oder abgeschätzt wird): $\mathrm{Prior}(\mathrm{H}_i)$.

$$P(\mathrm{H}_i|\mathrm{Daten}) = \frac{\mathscr{L}(\mathrm{data}|\mathrm{H}_i)\mathrm{Prior}(\mathrm{H}_i)}{P(\mathrm{Daten})} \tag{2.42}$$

Der Nenner stellt die Gesamtwahrscheinlichkeit der Daten dar, also die Summe der Wahrscheinlichkeiten aller möglichen Hypothesen: $\sum_j \mathscr{L}(\mathrm{data}|\mathrm{H}_j)\mathrm{Prior}(\mathrm{H}_j)$. Dann wird aus Gleichung 2.42:

$$P(\mathrm{H}_i|\mathrm{Daten}) = \frac{\mathscr{L}(\mathrm{data}|\mathrm{H}_i)\mathrm{Prior}(\mathrm{H}_i)}{\sum_j \mathscr{L}(\mathrm{Daten}|\mathrm{H}_j)\mathrm{Prior}(\mathrm{H}_j)} \tag{2.43}$$

Wenn wir jetzt noch durch $\mathscr{L}(\mathrm{H}_i|\mathrm{Daten})$ teilen, so erhalten wir:

$$P(\mathrm{H}_i|\mathrm{Daten}) = \frac{\mathrm{Prior}(\mathrm{H}_i)}{\sum_j (\mathscr{L}(\mathrm{Daten}|\mathrm{H}_j)/\mathscr{L}(\mathrm{data}|\mathrm{H}_i))\mathrm{Prior}(\mathrm{H}_j)} \tag{2.44}$$

Das Verhältnis $\mathscr{L}(\mathrm{Daten}|\mathrm{H}_j)/\mathscr{L}(\mathrm{data}|\mathrm{H}_i)$ (*odds ratio* genannt) berechnet sich z.B. als Quotient der *maximum likelihood* der rivalisierenden Hypothesen H_i und H_j. Wir könnten uns also vorstellen, dass wir einen Datensatz mit einer lineare (H_i) und einer quadratischen (H_j) Funktion annähern. Wenn wir dann das *odds ratio* berechnen, so ist dies einfach ein *likelihood*-Vergleich mittels X^2-Test. Der Bayesische Teil entsteht dadurch, dass wir den Hypothesen unterschiedliche Wahrscheinlichkeiten zuordnen können, etwas aus dem Wissen, dass der gleiche Versuch beim letzten Mal Hypothese H_1 20% und H_2 80% Wahrscheinlichkeit zugewiesen hat. Diese gehen dann als $\mathrm{Prior}(\mathrm{H}_i)$ und $\mathrm{Prior}(\mathrm{H}_j)$ in Gleichung 2.44 ein.

Für weitere Ausführungen und technischere Anwendungen kommen wir um eines der obigen speziellen Lehrbücher nicht herum.

3. Visualisierung und beschreibende Statistik

Ziel der Visualisierung ist das Erkennen von Mustern in unseren Daten. Wir wollen ein Gefühl für unseren Datensatz bekommen und für mögliche Zusammenhänge. Bei einer explorativen Visualisierung der verschiedenen erklärenden und Antwortvariablen entdecken wir mögliche Tippfehler, deutliche Unterschiede in der Anzahl der Stichpunkte je Faktorlevel usw. Die intensive Auseinandersetzung mit den Daten *vor* einer konkreten statistischen Analyse rüstet uns besser gegen statistische Fehlleistungen, als weitreichendes statistisches Wissen oder hochgerüstete Software!

3.1. Univariate Daten

Vor jeder Form der statistischen Auswertung müssen wir uns mit den Daten vertraut machen. Dafür ist der einfachste Weg eine grafische Darstellung. Für eine Variable bieten sich *boxplots* und Histogramme an; für zwei Variablen *scatterplots* oder konditinale *plots*; und für mehr Variablen Interaktionsplots oder ebenfalls konditionale *plots*. Sie sollen uns ermöglichen, Ausreißer zu finden, die Verteilung der Daten abzuschätzen, usw.

Wir beginnen mit einer Variablen. Eine numerische Zusammenfassung beinhaltet den **Mittelwert**, die **Standardabweichung**, den **Median**, die **Minimal-** und **Maximalwerte**, sowie häufig die **Quartilen 1** und **3**.[1] Für zwei Datensätze (x und y) sehen diese Werte so aus (Tabelle 3.1). Wir suchen in solchen Zusammenfassungen nach Zeichen von mangelnder Normalverteilung, auffälligen Ausreißern, oder besonders breiter Verteilung. Im Datensatz x fällt auf, dass alle Werte (außer dem Mittelwert) ganzzahlig sind. Dies deutet darauf hin, dass es sich möglicherweise um Zählungen handelt, also Poisson-verteilte Daten. Ebenfalls auffällig ist der hohe Maximalwert von y, der 2.2 Standardabweichungen über dem Mittelwert liegt. Eine grafische Darstellung würde uns hier vielleicht weiter helfen.

Tabelle 3.1.: Numerische Zusammenfassung der Datensätze x und y, $N = 100$.

	x	y
max	6	11.63
3. Quantile	3	7.31
Mittelwert	1.96	5.02
Standardabw.	1.43	2.97
Median	2	4.93
1. Quantile	1	3.20
min	0	−3.67

In R lässt sich diese Kurzzusammenfassung mittels des Befehls `summary()` angeben.

```
> set.seed(2)
> x <- rpois(100, lambda = 2)
> set.seed(20)
> y <- rnorm(100, mean = 5, sd = 3)
> summary(x)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.00    1.00    2.00    1.96    3.00    6.00

> summary(y)
```

[1] Die Quartilen teilen den der Größe nach sortierten Datensatz in 4 Teile: die kleinsten 25%, 25-50%, 50-75% und die über 75%. Ein beliebiger Prozentsatz (etwa 32.7%) kann natürlich auch erfragt werden. Den bezeichnen wir als dann als 32.7%-Quantil.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -3.669   3.199   4.925   5.015   7.305  11.630
```

Die Visualisierung erfolgt, etwas komplizierter, da es sich um mehrere Abbildungen handelt (siehe Abb. 3.1), mittels folgender Befehle:

```
> par(mfrow = c(2, 3), mar = c(5, 5, 3, 1))
> hist(x, cex.lab = 1.5, ylab = "Frequenz", main = "Histogramm")
> boxplot(x)
> title("Boxplot", ylab = "Mittelwert", cex.lab = 1.5)
> qqnorm(x, cex.lab = 1.5, ylab = "gemessene Quantilen", xlab = "theoretische
+                         Quantilen", main = "normaler Q-Q-Plot")
> qqline(x)
> hist(y, cex.lab = 1.5, ylab = "Frequenz", main = "Histogramm")
> boxplot(y)
> title("Boxplot", ylab = "Mittelwert", cex.lab = 1.5)
> qqnorm(y, cex.lab = 1.5, ylab = "gemessene Quantilen", xlab = "theoretische
+                         Quantilen", main = "normaler Q-Q-Plot")
> qqline(y)
```

Es gibt drei häufige Formen der Visualisierung univariater Daten: **Histogramme**, **Boxplots** und **Q-Q-plots**. Ein Histogramm stellt dar, wieviele Werte pro Werteklasse vorhanden sind (Abbildung 3.1, erste Spalte). Boxplots stellen Median und Quantilen 1 und 3 als Box, sowie Minimum und Maximum als Fehlerbalken dar (*whiskers*; einer Konvention folgend haben diese maximal die anderthalbfache Länge der Box. Werte darüber hinaus werden als Punkte dargestellt.), wie in der zweiten Spalte der Abbildung 3.1. In der letzten Spalte werden in einem *Q-Q-plot* die Quantilen der gefundenen Daten mit normalverteilten verglichen (Mittelwert und Standardabweichung aus den beobachteten Daten berechnet). Die durchgezogene Linie gibt die erwartete Verteilung an.

Hier wird deutlicher, was wir bei der numerischen Zusammenfassung bereits vermutet haben. Der Datensatz x hat eine typische Poisson-Verteilung im Histogramm, die Fehlerbalken im Boxplot sind nach oben länger als nach unten, und im *Q-Q-plot* liegen die Punkte etwas *J*-förmiger als die Normalverteilung.

Der Datensatz y hingegen scheint normalverteilt zu sein und weist nicht die gerade beschriebenen Abweichungen auf. Tatsächlich entstammen die oberen Daten einer Poisson-Verteilung (mit $\lambda = 2$) und die unteren einer Normalverteilung ($\mu = 5$, $\sigma = 3$).

3.2. Bivariate Daten

Liegen zwei Datensätze vor, die miteinander in Beziehung stehen, so bieten sich *weitere* Formen der numerischen und graphischen Darstellung an. Natürlich muss man weiterhin jede einzelne Variable für sich verstehen (als univariaten Datensatz also).

3.2.1. Kontinuierliche Daten

Wir können die beiden Datensätze miteinander korrelieren und gegeneinander auftragen. Beachte: Wenn die beiden Variablen zwar zusammenhängen, aber diese Beziehung nicht konstant über den Wertebereich ist (etwa 1:1-Verhältnis bei niedrigen, aber 10:1-Verhältnis bei hohen Werten), so spricht man von nicht-linearen Zusammenhängen. Eine Auftragung der Variablen `y` gegen `x` (oder umgekehrt) vermittelt ein Gefühl für ihre gegenseitige Abhängigkeit (*scatterplot*: Abbildung 3.2).

Die folgenden `par`-Argumente setzen zwei Abbildungen nebeneinander (`mfrow`), reduzierenden den äußeren Rand jeder einzelnen Graphik (`oma`) und vergrößern die Gesamtabbildungsfläche (`plt`). Die verschiedenen `plot`-Optionen sind in der `plot.default`-Hilfe beschrieben.

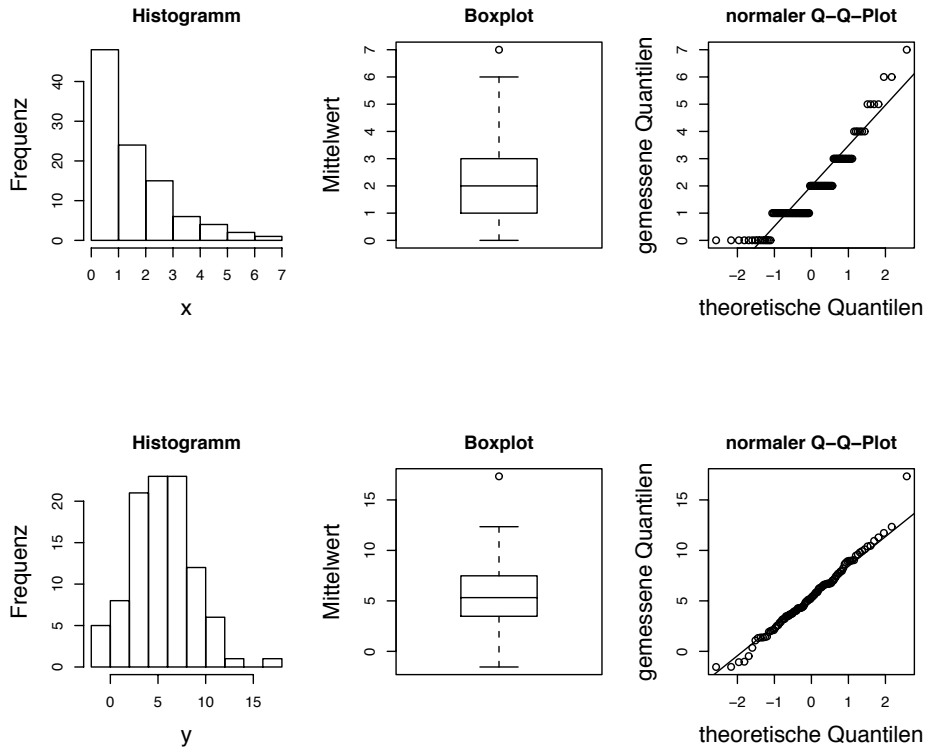


Abbildung 3.1.: Explorative Datenanalyse mittels Histogramm, Boxplot und Q-Q-plot: In den oberen drei Abbildungen ist ein Poisson-verteilter Datensatz aufgetragen, in den unteren ein normalverteilter.

```
> par(mfrow = c(1, 2), oma = c(0, 0.5, 0, 0), plt = c(0.135, 0.98,
+     0.15, 0.98))
> set.seed(20)
> y1 <- round(sort(rnorm(15, mean = 7, sd = 3)), 2)
> y2 <- sort(y1 + runif(15, 0, 5))
> y3 <- 10 * (y1^0.1) + runif(15, 0, 0.2)
> plot(y2 ~ y1, xlab = "", ylab = "", cex.lab = 1.5, pch = 16,
+     cex = 1.5, tcl = 0.5, cex.axis = 1.3)
> plot(y3 ~ y1, xlab = "", ylab = "", cex.lab = 1.5, pch = 16,
+     cex = 1.5, tcl = 0.5, cex.axis = 1.3)
```

3.2.2. Daten mit kategorischer Variablen

Wenn eine Variable als kategorische Daten vorliegt (unabhängig ob sie eine erklärende Variable ist oder nicht), so bietet sich ein kategorischer Boxplot an. Dabei wird die kontinuierliche Variable gegen die kategorische aufgetragen. Dies ist natürlich die bevorzugte Form, wenn eine Kausalität $A \Rightarrow B$ bekannt ist (Abbildung 3.3).

Im folgenden Beispiel benutzen wir zusätzlich zum `plot`-Befehl selber noch einige Parameter der Abbildungsgestaltung vor. Dafür verändern wir die Grundeinstellungen, die wir vorher unter der Variablen op speichern, um sie nachher wiederherstellen zu können. Siehe in R `?par` für die Erklärung der einzelnen Optionen.

```
> op <- par(no.readonly = TRUE)
> par(mfrow = c(1, 1), mar = (c(5, 5, 4, 2) + 0.1))
> set.seed(10)
```

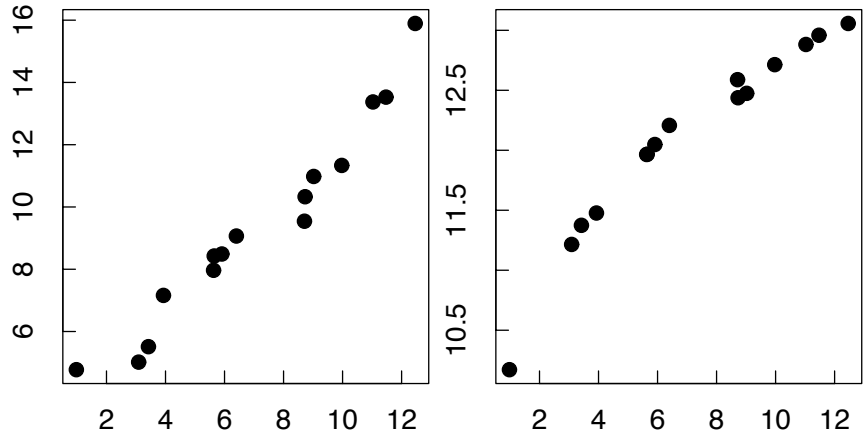


Abbildung 3.2.: Zwei kontinuierliche Variablen (links eine lineare, rechts eine nicht-lineare Beziehung darstellend).

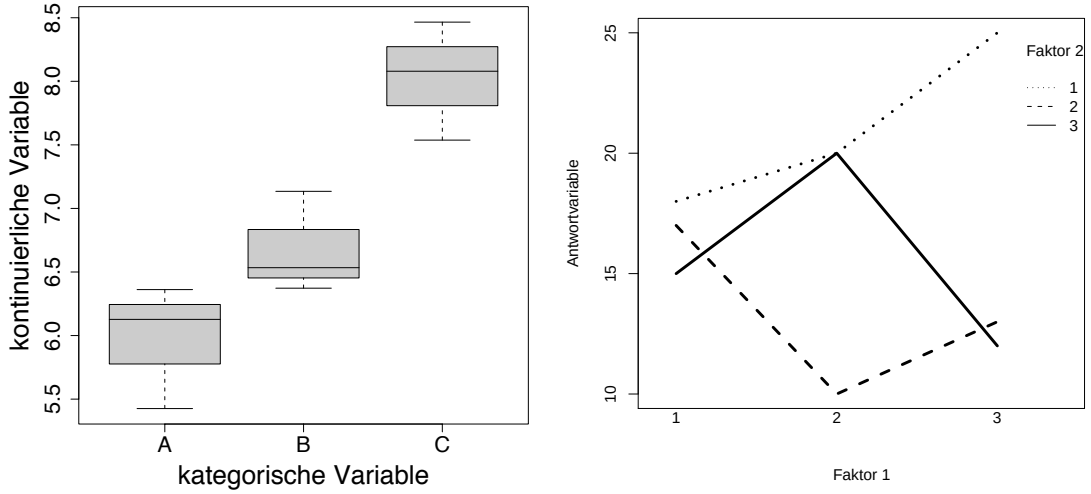


Abbildung 3.3.: Boxplot einer kontinuierlichen und einer kategorischen Variablen (links), sowie ein Interaktionsplot von zwei Faktoren mit einer Antwortvariablen (rechts).

```
> y <- sort(runif(9, 0, 5) + 5)
> x <- factor(rep(c("A", "B", "C"), each = 3))
> plot(y ~ x, xlab = "kategorische Variable", ylab = "kontinuierliche Variable",
+     pch = 16, cex = 2, cex.lab = 2, tcl = 1.5, cex.axis = 1.7,
+     col = "grey80")
> par(op)
```

Wenn mehrere Faktoren vorhanden sind, so wird die grafische Darstellung etwas komplexer. Am wichtigsten hier ist, dass wir versuchen, solange die Daten darzustellen, bis wir ein Gefühl für ihre Beziehungen haben. Bei zwei Faktoren und einer Antwortvariablen bietet sich der Interaktionsplot an (Abbildung **??**).

```
> op <- par(no.readonly = TRUE)
> par(mfrow = c(1, 1), mar = (c(5, 5, 4, 2) + 0.1))
> counts <- c(18, 17, 15, 20, 10, 20, 25, 13, 12)
> outcome <- gl(3, 1, 9)
> treatment <- gl(3, 3)
```

```
> interaction.plot(treatment, outcome, counts, lwd = 3, trace.label = "Faktor 2",
+     xlab = "Faktor 1", ylab = "Antwortvariable", legend = T,
+     cex.lab = 1.7, cex.axis = 1.5, cex = 2)
> par(op)
```

3.3. Transformationen und ihre Visualisierung

Die Natur zieht die Maße für ihre Organismen oder Prozesse nicht aus einem Hut voller Zettel mit normalverteilten Zahlen. Die Messungen, die wir anstellen, und die Zahlen, die wir mit Experimenten der Natur abringen, sind leider allzu oft nicht normalverteilt. Die einfachen statistischen Verfahren, bis hinein in die recht aufwendige ANOVA, haben aber als Voraussetzung, dass die zu analysierenden Zahlen normalverteilt sind. Um also trotzdem die schiefen Verteilungen realer Daten analysierbar zu machen, benutzt man **Transformationen**.

Gelegentlich überkommt einen Anfänger Unwohlsein bei dem Gedanken, Daten einfach mittels einer mathematischen Operation in einen ganz anderen Datensatz zu überführen, der dann lustig ausgewertet werden kann. Und leider kommt es immer wieder vor, dass Daten, aus Unwissenheit oder mit Vorsatz, fehlerhaft transformiert werden. Dies hat der Transformation keinen guten Namen beschert.

Andererseits ist nichts prinzipiell gegen Transformationen einzuwenden. Schließlich benutzen wir sie tagtäglich, ohne uns dessen bewusst zu sein: pH-Werte sind genauso auf einer logarithmischen Skala gemessen wie die Dezibel der Lautstärke. Die Uhrzeit liegt in einem wilden Misch aus Zahlensystemen vor (60 Sekunden zur Minute, aber 24 Stunden zum Tag), ohne dass wir damit ein Problem hätten. Nur weil wir also ein linear skaliertes Metermaß benutzen, um Spinnenbeinlängen zu messen, heißt das doch nicht, dass Spinnenbeinlängen ebenfalls auf diesem linearen Maß normalverteilt sein müssen.

Zellkulturen wachsen ebenso wie Babies exponentiell (zumindest anfänglich), während Bierschaum expontiell zerfällt, genau wie radioaktive Atomkerne. Der exponentielle und logarithmische Maßstab sollte uns also einigermaßen vertraut sein, und einer Logarithmierung von Daten sollte zumindest keine Emotion im Wege stehen. Ähnlich könnte man für andere Transformationen argumentieren. Stattdessen schauen wir uns einfach ein paar Transformationen und ihre Indikation an.

3.3.1. Die logarithmische Transformation

Die wahrscheinlich häufigste Transformation biologischer Daten ist die logarithmische: $y = ax + b \rightarrow \log(y) = a'x + b'$. Sie wird aus mehreren Gründen benutzt:

1. weisen viele Datensätze ein Rechtslastigkeit auf, ihre Histogramme sind also nach recht ausgezogen. Diese kann u.U. durch eine Logarithmierung aufgehoben werden (siehe Abbildung 3.4).

2. liegt manchen Daten eine Exponentialverteilung zugrunde, die mittels der log-Transformation linearisiert wird: $log(y = be^{ax}) \rightarrow log(y) = ax + \log(b)$.

3. kann eine Beziehung zwischen Varianz und Mittelwert (die ja eine Verletzung der Annahme varianzanalytischer Verfahren darstellt) häufig durch diese Transformation getilgt werden. (Ob Mittelwert und Varianz dann tatsächlich unkorreliert sind, muss man nach der Analyse überprüfen. Siehe dafür die Ausführungen in den entsprechenden Kapiteln zu ANOVA und GLM.)

Ein Problem stellt sich, wenn der Datensatz Nullen enthält, deren Logarithmus ja nicht definiert ist. Statt $\log(y)$ wird dann $\log(y + c)$ berechnet, wobei allerdings der genaue Wert von c einen Einfluss auf die Ergebnisse haben kann! Als Daumenregel sollte c halb so groß

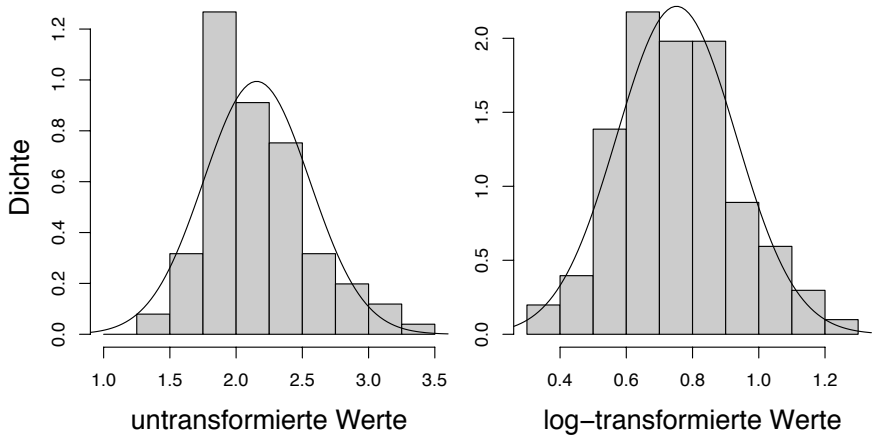


Abbildung 3.4.: Effekt der log-Transformation auf die Verteilung einer Variablen. Zur Verdeutlichung stellen die durchgezogenen Linien die aus Mittelwerten und Standardabweichungen abgeleiteten Normalverteilungen dar.

wie der kleinste Wert von y gewählt werden. Idealerweise wählt man c so, dass die Residuen nach der Analyse möglichst symmetrisch sind (Berry 1987).[2]

Generell sollten nach der Analyse die Ergebnisse nicht in transformierten Werten angegeben werden, sondern rücktransformiert. Im Beispiel des Logarithmus wird also aus $\log_b(y) \rightarrow b^{\log_b(y)}$, wobei b die Basis des Logarithmus ist (üblicherweise 10 oder e). Entsprechendes geschieht mit den Standardfehlern oder Konfidenzintervallen. Beachten müssen wir hierbei, dass wir die transformierten Konfindenzintervalle ausrechnen, und diese dann rücktransformieren (siehe Formel 2.21):

$$CI = b^{(\log(y) \pm t_{0.05(n-1)} \sqrt{s^2_{\log y}/n})}$$

Dabei werden die Konfidenzintervalle asymmetrisch! Schließlich sind ja die logarithmierten CIs symmetrisch, und ihre „Ent-logarithmisierung“ führt zur Asymmetrie.

log-Transformationen sind in R einfach durch den Befehl `log(x+c)` (zur Basis e), bzw. `log10(x+c)` (zur Basis 10) durchführbar. Die Rücktransformation ist mittels `exp(x+c)` bzw `10^(x+c)` möglich.

3.3.2. Die Wurzel-Transformation

Bei Zähldaten kann oft mit Wurzel-Transformation eine Entkopplung von Mittelwert und Varianz erreicht werden (Sokal and Rohlf 1995). Da diese Zähldaten aller Wahrscheinlichkeit nach Poisson-verteilt sind, können wir eine Transformation häufig vermeiden und stattdessen die Poisson-Verteilung als Fehlerverteilung definieren (siehe Abschnitt 8 auf Seite 141). Mittelwert und Varianz sind bei der Poisson-Verteilung ja identisch (siehe Abschnitt 2.7 auf Seite 11), und eine Wurzeltransformation kann diesen Zusammenhang brechen. Ähnlich wie bei der log-Transformation können wir eine Konstante dazuzählen (0.5 oder 0.375; Sokal and Rohlf 1995), wenn Nullen im Datensatz vorkommen.

Wurzeltransformationen werden in R durch den Befehl `sqrt(x+c)` umgesetzt.

3.3.3. Die arcsin-Wurzel-Transformation

Die arcsin-Transformation (auch Winkeltransformation; *angular transformation*) ist angezeigt, wenn wir es mit Prozentzahlen oder Anteilen zu tun haben. Diese Daten folgen der Binomialverteilung (siehe Gleichung 2.9), mit Standardabweichung $\sigma = \sqrt{pq/k}$. Da Mittelwert $\mu = p$ und $q = 1 - p$ (k = konstant), ist die Varianz (σ^2) eine Funktion des Mittelwertes. Die arcsin-Transformation entkoppelt diese Abhängigkeit.

[2] Eine *Minimierung* der Residuen kommt nicht als Ziel in Frage: Dies würde erreicht für $c \rightarrow \infty$!

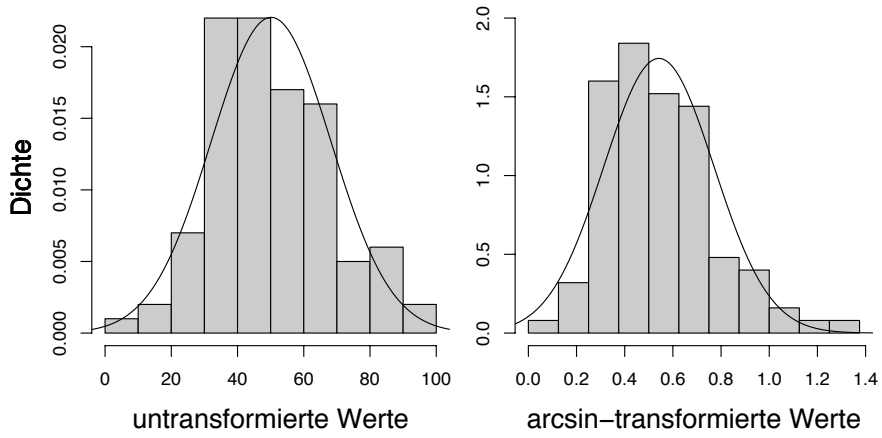


Abbildung 3.5.: Effekt der arcsin-Transformation auf die Verteilung einer Variablen. Zur Verdeutlichung stellen die durchgezogenen Linien die aus Mittelwerten und Standardabweichungen abgeleiteten Normalverteilungen dar.

Wir stellen uns dafür eine Veränderung von 50 nach 55% vor, und vergleichen diese mit der von 5 nach 10%. Offensichtlich ist erstere nur eine marginale Änderung, während letztere substantiell ist. Die Winkeltransformation zieht die Zahlen nahe 0 und 100 auseinander, während sie die mittleren Werte staucht (siehe Tabelle und Sokal and Rohlf 1995).

Tatsächlich wird nicht allein der Arcsinus von y berechnet, sondern der Arcsinus $\sqrt{y/100\%}$. Für die Werte von 0 bis 100% berechnen sich entsprechend folgende transformierten Werte (y'):

y in %	0	10	20	30	40	50	60	70	80	90	100
y'	0	18.4	26.6	33.2	39.2	45.0	50.8	56.8	63.4	71.6	90

Die Rücktransformation eines berechneten Wertes (θ') erfolgt einfach durch $\theta = \sin^2(\theta')$.[3]

Der `asin`-Befehl kann nur mit Werten zwischen -1 und 1 umgehen, da ja der Sinus eines Winkels eben zwischen -1 und 1 liegt. Entsprechend müssen wir bei der Transformation durch entsprechende Division dafür sorgen, dass das Ergebnis des Wurzelziehens kleiner gleich 1 ist. Der `round(., 1)`-Befehl rundet auf die erste Kommastelle:

```
> xx <- seq(0, 100, by = 10)
> yy <- round(asin(sqrt(xx/100)), 1)
> xx

 [1]   0  10  20  30  40  50  60  70  80  90 100

> yy

 [1] 0.0 0.3 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.6
```

Oder, für gerundete y'-Werte in Grad:

```
> round(asin(sqrt(xx / 100)) / pi / 2 * 360, 1)

[1]  0.0 18.4 26.6 33.2 39.2 45.0 50.8 56.8 63.4 71.6 90.0
```

[3] Dabei ist $\sin^2(x) = (\sin(x))^2$.

3.3.4. Box-Cox-Transformationen

Statt eine ganze Reihe von verschiedenen Transformationen durchzuprobieren wäre es natürlich schön, eine Funktion zu haben, die die beste Transformation herausfindet. Diese Idee findet sich in der Box-Cox-Transformation umgesetzt. Dabei wird ein Parameter λ gesucht, mit dessen Hilfe der transformierte Wert (y') entsprechend folgender Formel berechnet wird:

$$y' = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{für } \lambda \neq 0; \\ log(y) & \text{für } \lambda = 0. \end{cases} \tag{3.1}$$

Spezialfälle dieser Transformation sind die logarithmische (für $\lambda = 0$), die Wurzeltransformation (für $\lambda = 1/2$) sowie weitere Wurzeln (kubisch für $\lambda = 1/3$, usw.), und schließlich die reziproke Transformation (für $\lambda = -1$).

Den optimalen Wert für λ berechnen wir über die *likelihood*-Funktion obiger Transformation (Box and Cox 1964; Venables and Ripley 2002):

$$\mathscr{L}(\lambda) = c - \frac{n}{2}\text{RSS}(z^{(\lambda)}) \tag{3.2}$$

wobei RSS die Residuensumme der Abweichungsquadrate (*residual sum of squares*) der Regression gegen $z^{(\lambda)}$ ist. Etwas umständlich ist $z^{(\lambda)} = y^{(\lambda)}/\dot{y}^{\lambda-1}$, wobei wiederum $\dot{y}$ das geometrische Mittel[4] der Beobachtungen ist.

Glücklicherweise haben nahezu alle Statistikprogramme eine Routine, um diesen Wert zu finden, so dass diese Feinheiten der Berechnung für uns folgenlos sind. Die Box-Cox-Transformation wird nahezu ausschließlich bei linearen Modellen und ANOVAs benutzt wird, und zwar *nach* einer Analyse untransformierter Daten. Dann wird ein Wert für λ gesucht, der die Residuen nach der Analyse möglichst nah in eine Normalverteilung überführt.

Schauen wir uns dies am Beispiel eines Datensatzes über Höhe, Umfang und Holzvolumen von Schwarzkirschbäumen an (wiederum in R standardmäßig implementiert; siehe auch Venables and Ripley 1994). Für die Funktion boxcox müssen wir zunächst das **MASS**-*package* laden. Dann berechnen wir λ und schauen uns das log-*likelihood*-Profil dieser Funktion an, bevor wir den neuen, transformierten Datensatz berechnen.

```
> library(MASS)
> data(trees)
> lambda.fm1 <- boxcox(Volume ~ 1, data = trees)
```

Diese Funktion produziert eine Liste mit x- und y-Werten, die in der mit ausgegebenen Abbildung dargestellt werden (nicht abgebildet). Um an den x-Wert (λ) zu kommen, der den maximalen y-Wert (log-*likelihood*) liefert, schreiben wir:

```
> lambda.max <- lambda.fm1$x[which.max(lambda.fm1$y)]
```

Dieser Wert ist sehr nahe 0, wie auch in der Abbildung zu sehen. Dies spricht für eine log-Transformation. Wir lassen uns einfach den transformierten Datensatz mit genau diesem λ von R berechnen:

```
> require(car)
> bc.Vol <- box.cox(trees$Volume, p = lambda.max)
```

Vergleichen wir die beiden Datenverteilungen (Abbildung 3.6):

```
> par(mfrow = c(1, 2), oma = c(0, 2, 0, 0))
> hist(trees$Volume, xlab = "untransformierte Werte", freq = F,
+     ylab = "", col = "grey80", main = "", cex = 1.5, cex.lab = 1.5)
> hist(bc.Vol, xlab = "transformierte Werte", freq = F, ylab = "",
+     col = "grey80", main = "", cex = 1.5, cex.lab = 1.5)
> mtext("Dichte", side = 2, outer = T, cex = 1.5, at = 0.6)
```

[4] Während beim arithmetischen Mittel alle Datenwerte aufsummiert und dann durch n geteilt werden, werden sie beim geometrischen Mittel *aufmultipliziert*, und dann wird die n-te Wurzel gezogen: $\dot{y} = \left(\prod_{i=1}^{n} y_i\right)^{1/n}$. Dies ist offensichtlich nur sinnvoll definiert, wenn alle $y_i > 0$ sind.

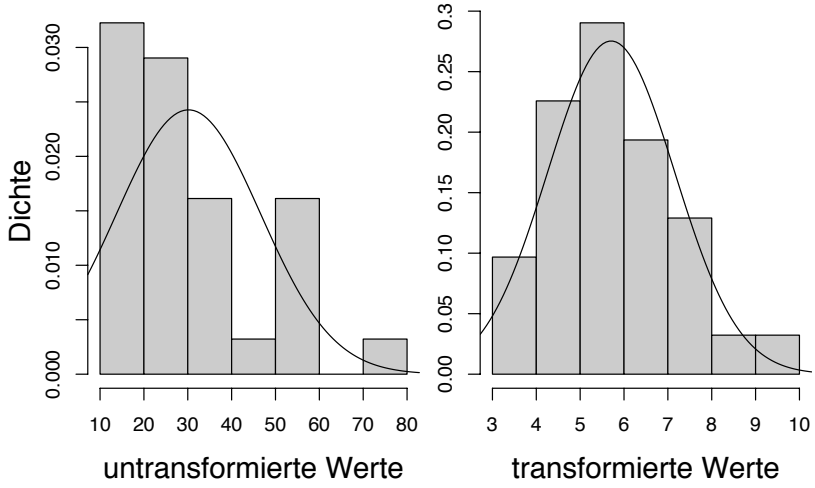


Abbildung 3.6.: Effekt der Box-Cox-Transformation auf die Verteilung der Variablen „Kirschholzvolumen“. Zur Verdeutlichung stellen die durchgezogenen Linien die aus Mittelwerten und Standardabweichungen abgeleiteten Normalverteilungen dar.

3.3.5. Rang-Transformation

Bei einer Rang-Transformation werden die Werte eines Datensatzes durch ihre Ränge ersetzt. Damit wird ein neuer Datensatz mit Werten zwischen 1 und n (der Anzahl Datenpunkte) generiert. An sich sind alle Datenpunkte äquidistant, außer wenn ein Wert mehrfach vorkommt (Mehrfachnennung (engl.: *ties*); diese werden bei der Rang-Transformation durch den Mittelwert ihrer Ränge ersetzt: Aus „1, 3, 3, 4“ wird „1, 2.5, 2.5, 4“). Ein Beispiel ist in Abbildung 3.7 gegeben.

Der Informationsverlust ist bei dieser Transformation natürlich erheblich. Sie wird deshalb entweder als konservative Transformation benutzt oder wenn die Daten tatsächlich nicht anders sinnvoll transformierbar sind. Allerdings müssen wir dann bedenken, dass auch die Ergebnisse, die wir aus den transformierten Daten gewinnen, u.U. nicht sehr informativ sind.

Außerdem sind Rang-transformierte Daten üblicherweise nicht normalverteilt (Abbildung 3.7). Desweiteren müssen wir uns bewusst sein, dass beispielsweise eine ANOVA mit Rangtransformierten Daten weiterhin auf der Annahme der Varianzhomogenität fußt, die sich ebenfalls nicht notwendigerweise mit einer Transformation einstellt.

Die Rang-Transformation wird für nicht-parametrischen Statistiken benutzt, denn die Koeffizienten (Steigung und y-Achsenabschnitt) etwa einer Regression von Rang-transformierten Daten sind natürlich aussagelos, nur ihre Signifikanz ist von Interesse.

Eine ANOVA rank-transformierter Daten entspricht einem Kruskal-Wallis-Test, auch wenn dieser eine andere Herleitung hat.

3.3.6. Standardisierung

Vor allem für die Analyse multivariater Daten und für die (kontinuierlichen) erklärenden Variablen in multiplen Regressioen und GLMs wird häufig eine Standardisierung (auch Normalisierung genannt) vorgenommen. Darunter verstehen wir eine Transformation, die die Datenmittelwerte auf Null und die Standardabweichung auf 1 normiert. Der Effekt einer Standardisierung ist dass alle Variablen dann einen Mittelwert von 0 und eine Standardabweichung von 1 haben. Damit sind die Einheiten, auf denen die Variablen vorliegen, miteinander vergleichbar, und groß-wertige Variablen dominieren nicht mehr die Interaktionen. Dies wird

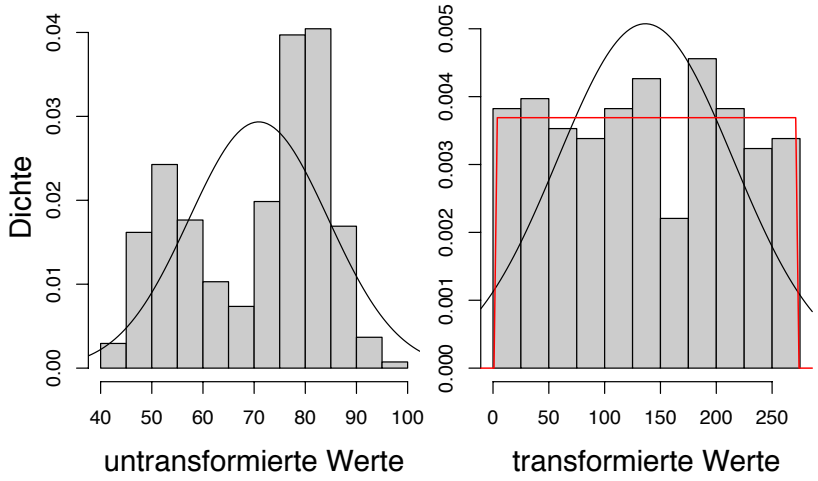


Abbildung 3.7.: Effekt der Rang-Transformation auf die Verteilung der Variablen „Abstand zwischen Ausbrüchen des Geysirs *Old Faithful*“. Zur Verdeutlichung stellen die durchgezogenen Linien die aus Mittelwerten und Standardabweichungen abgeleiteten (und offensichtlich vollkommen unangemessenen) Normalverteilungen dar. Die uniforme Verteilung der transformierten Daten ist durch die rote Linie gekennzeichnet.

durch folgende einfache Formel erreicht:

$$y_i' = \frac{y_i - \bar{y}}{s_y} \tag{3.3}$$

wobei $\bar{y}$ der Mittelwert des Datensatzes ist und s_y dessen Standardabweichung. Man kann sich dies als einen zweistufigen Prozess vorstellen: (1) Zentrierung (*centering* oder *translation*) der Daten und (2) Standardisierung im engeren Sinn (Fläche unter der Normalverteilung gleich 1 setzen). Standardisierte Werte werden im englischen häufig als *z-scores* bezeichnet.

Der Sinn einer Standardisierung wird deutlich, wenn wir uns vorstellen, dass zwei Variablen in einer Analyse als Antwortvariablen parallel im Fokus des Interesses stehen, z.B. Reaktionsgeschwindigkeit (in s) und Aggressivität (als Angriffe pro s) in Abhängigkeit von Testosteronimpfungen bei Mäusen. Wenn eine Variable jetzt viel höhere Werte (und damit eine höhere Varianz) aufweist als die andere, so wird diese die Analyse dominieren: Wir vergleichen sozusagen Äpfel mit Birnen. Durch eine Standardisierung wird die Varianz ($= s^2$) und der Mittelwert beider Variablen gleich (nämlich 1 und 0, respektive).

Für andere Zwecke gibt es weitere Formen der Standardisierung. So empfahl etwa eine physikalische Fachzeitschrift in den 90ern, dass alle Abbildungen auf x- und y-Achse von 0 bis 1 skaliert sein sollten. Dies wird durch folgende Standardisierung erreicht (Legendre and Legendre 1998):

$$y_i' = \frac{y_i - y_{\min}}{y_{\max} - y_{\min}} \tag{3.4}$$

Diese Form der Standardisierung nennt man *ranging*, und sie ist ebenfalls bei manchen multivariaten Verfahren von Nutzen, z.B. bei stark unterschiedlichen Häufigkeiten oder Deckungsgraden.

3.3.7. Ausreißer und das Beschneiden (trimming) von Datensätzen

Bisweilen erscheinen in unseren Daten Werte, die weit außerhalb der erwarteten Verteilung liegen, sogenannte Ausreißer. Ihnen kommt bei der Berechnung z.B. des Mittelwertes sehr großes Gewicht zu, uns so können sie eine Analyse stark beeinflussen. Manchmal kann man

diese Punkte auf methodische Probleme zurückführen: die Waage klemmte, einen andere Person hat Zeiten gestoppt, eine Probefläche erhielt versehentlich die 10-fachen Düngemenge, ein Tippfehler usw. Dann können wir mit gutem Grund den entsprechenden Punkt von der Analyse ausschließen. (Hier sehen wir den Wert sorgfältiger Dokumentation eines Versuchablaufs!)

Obwohl die Versuchung, auch nicht-erklärbare Ausreißer von der Analyse auszuschließen, groß ist, so müssen wir ihr doch widerstehen. Schließlich wollen wir unsere Hypothesen mit unseren Daten testen, und nicht auf Gedeih und Verderb eine Signifikanz erzeugen (häufig unter dem zynischen Euphemismus „ergebnisorientierte Analyse“).

Wenn ein Datensatz also partout nicht entsprechend unserer Bedürfnisse transformierbar ist, so müssen wir auf robuste Analysen zurückgreifen (Median statt Mittelwert, Kruska-Wallis statt ANOVA, usw.).

Um ein Gefühl für den Effekt von Ausreißern zu bekommen, können wir in R der Funktion `mean` eine Option `trim` mitgeben. Dabei bestimmt der angegebene Wert, welcher Teil des Datensatzes von jedem Ende der Verteilung abgeschnitten werden soll, bevor der Mittelwert berechnet wird. `trim=0.1` entfernt also 20%. Wenngleich dies den Datensatz verändert, so findet doch wenigstens keine Selektion der einzelnen Ausreißer statt, sondern eine symmetrische Begrenzung. Das folgende Beispiel ist aber keinesfalls eine Empfehlung für das *trimming* von Datensätzen!

```
> xx <- rlnorm(100, 0.5, 0.5)
> mean(xx)

[1] 1.855726

> mean(xx, trim = 0.1)

[1] 1.717062

> mean(xx, trim = 0.2)

[1] 1.664248

> median(xx)

[1] 1.654721
```

Wir sehen, dass der Mittelwert bei zunehmender Beschneidung der Datengrundlage dem Median immer ähnlicher wird, letzterer also die “robuste” Form des Mittelwerts ist.

4. Einfache Fragen und die Methodik ihrer Beantwortung: Klassische Tests

4.1. Klassische Tests

Es gibt einen ganzen Schwung einfacher Tests, „Klassische Tests“ genannt, die für die Analyse von einem oder zwei Stichprobengruppen benutzt werden können. Wenn es die Fragestellung zulässt, sollten wir diese einfachen Tests benutzen. Im folgenden werden wir uns sowohl klassische parametrische als auch nicht-parametrische Tests anschauen.

Die Unterscheidung in parametrische und nicht-parametrische Tests ist etwas irreführend. Eigentlich wollen wir natürlich genau die Daten analysieren, die wir erhoben haben, d.h. unter voller Berücksichtigung der Werte. Gelegentlich werden aber Ausreißer oder extrem schiefe Verteilungen unserem Bemühen einen Strich durch die Rechnung machen. Während mancher große Mühen unternimmt, um mittels Datentransformation (Abschnitt 3.3) oder nicht-normaler Fehlerverteilungen (Abschnitt 8) die Originaldaten analysierbar zu machen (Crawley 2002), sind andere bereit, einen Teil der Informationen zu opfern, um die Datenanalyse einfach zu halten (Venables and Ripley 1994; Quinn and Keough 2002). Geopfert wird der *absolute* Wert der Beobachtung, während der *relative* Wert (der Rang im Vergleich zu den anderen Datenpunkten) erhalten bleibt. Entsprechend sind Parameter wie der Mittelwert der Rangdaten kein Parameter der ursprünglichen Verteilung mehr: deshalb der Name nicht-parametrische Verfahren.

Der Verlust an Information ist dabei überraschend gering, und für viele Tests ist die nicht-parametrische Option zu erwägen. Andererseits kann man keine Parameter schätzen, die Koeffizienten eines Tests haben also keine interpretierbare Bedeutung. Somit ist beispielsweise eine nicht-parametrische Regression natürlich nicht nutzbar um Beziehungen zu *quant*ifizieren.

Es gibt drei Hauptideen zu einfachen Tests. Erstens kann man den Unterschied zweier Stichproben in Beziehung setzen zur Variabilität der Daten. Dies führt zum t-Test in seiner weiteren Form (Formel 4.4). Oder man kann zweitens analysieren, ob eine Stichprobe mehr Daten über einem kritischen Wert hat als eine andere Stichprobe. Dieser Ansatz ist in den meisten nicht-parametrischen Test wie etwa dem Wilcoxon-Test umgesetzt. Und schließlich lässt sich für manche Fälle die Anzahl an Möglichkeiten direkt kombinatorisch berechnen, und zur vorgefundenen Häufigkeit in Beziehung setzen (etwa der Binomialtest).

4.1.1. Statistik mit nur einer Stichprobe

Nehmen wir an, wir haben nur eine Stichprobe, etwa die Schuhgröße von 100 Personen, und wir wollen wissen, ob sie von einem erwarteten Wert abweicht (z.B. vom „wahren“ deutschen Schuhgrößenmittelwert von 40.2). Nehmen wir weiterhin an, dass die Daten normalverteilt sind, dann kann man den t-Test (auch `Student's t-Test`[1]) benutzen, um diesen Vergleich zu testen. Es ist dann

$$t_s = \frac{\bar{x} - A}{SE_{\bar{x}}} = \frac{\bar{x} - A}{sd_x/\sqrt{n}} \tag{4.1}$$

mit $\bar{x}$ gleich dem Mittelwert der Stichprobe, A einer Konstanten, gegen deren Wert getesten werden soll (für uns also 40.2) und $SE_{\bar{x}}$ dem Standardfehler auf dem Mittelwert der Stichpro-

[1] An dieser Stelle ist der Hinweis obligatorisch, dass „Student“ das Pseudonym von W.S. Gossett war, als er, für die Guinness-Brauerei arbeitend, den t-Test veröffentlichte. Sein Arbeitgeber wollte wohl nicht mit so etwas Vergeistigten wie der t-Verteilung assoziiert werden.

be (im rechten Teil der Gleichung aufgelöst als Standardabweichung geteilt durch die Wurzel aus dem Stichprobenumfang).

Kritische t-Werte sind in vielen Statistikbüchern tabelliert (z.B. Zar 1996). Die t-Verteilung ist eine abgeplattete Normalverteilung, was der Tatsache Rechnung trägt, dass wir die Statistik einer Stichprobe und nicht die Grundgesamtheit betrachten.

Wir könnten in R die eingebaute t-Verteilungsfunktionen `pt` nutzen, um uns die Wahrscheinlichkeiten der errechneten t-Werte anzeigen zu lassen (etwa: `pt(4.12, 99)`). Einfacher aber ist die Nutzung der Funktion `t.test`:

```
> schuhgroesse <- c(41, 40, 37, 45, 41, 39, 38, 20, 43, 44, 44,
+     36, 41, 42)
> t.test(schuhgroesse, mu = 40.2)

        One Sample t-test

data:  schuhgroesse
t = -0.5099, df = 13, p-value = 0.6186
alternative hypothesis: true mean is not equal to 40.2
95 percent confidence interval:
 35.78636 42.92793
sample estimates:
mean of x
 39.35714
```

Wenn die Daten allerdings *nicht* normalverteilt sind, so können wir natürlich auch nicht den t-Test nutzen. Statt der echten Werte unserer Stichprobe nutzen wir jetzt nur den Rang der Beobachtung (relativ zu unserem konstanten Wert A). Der Test heißt **Wilcoxon Vorzeichen-Rang-Test** (*Wilcoxon signed rank test*). Er berechnet sich wie folgt:

$$T+ = \sum_{y>0} Rang|y| \text{ , wobei } y = x - A \tag{4.2}$$

In Worten: Berechne die Differenzen zwischen den Messwerten und der Konstanten, berechne den Rang der *absoluten* Differenzen ($Rang|y|$), und summiere die Ränge der positiven Differenzen[2]. Wie beim t-Test finden sich die kritischen Werte in Tabellen.

Wie für die t-Verteilung gibt es auch den Wilcoxon-Test als Wahrscheinlichkeitsfunktion: `psignrank`. Einfacher ist folgender Syntax:

```
> wilcox.test(schuhgroesse, mu = 40.2)

        Wilcoxon signed rank test with continuity correction

data:  schuhgroesse
V = 57, p-value = 0.8015
alternative hypothesis: true location is not equal to 40.2
```

Noch einfacher berechnet sich **Fischers Vorzeichentest** (*Fisher's sign test*), auch **Binomialtest** genannt, der nur die Vorzeichen der Differenzen zwischen Messwert und Konstanten berücksichtigt (n_+ und n_-). Er vergleicht wie häufig ein Messwert über der Konstanten gelegen hat, im Vergleich zu gleichhäufigen positiven und negativen Werten. Dazu berechnet man den Binomialkoeffizienten $B = \binom{N}{n_+}$. Der p-Wert dieses Tests[3] errechnet sich als Summe aller Koeffizienten die $\leq B$ (d.h. $\binom{N}{n_+-1}, \binom{N}{n_+-2}$, etc.), dividiert durch 2^N.

Dieser Test steht nicht zur Verfügung, aber eine Funktion ist leicht definiert:

[2] http://mathworld.wolfram.com/WilcoxonSignedRankTest.html
[3] http://mathworld.wolfram.com/FisherSignTest.html

```
> fisher.sign <- function(x, A) {
+     N <- length(x)
+     n <- ifelse(mean(x) > mean(A), sum(ifelse(x - A >= 0, 1,
+         0)), sum(ifelse(x - A <= 0, 1, 0)))
+     B <- numeric(n)
+     for (i in 0:n) B[i] <- choose(N, i)
+     ifelse(n < N/2, res <- round((sum(B)/2^N), 5), res <- 1 -
+         round((sum(B)/2^N), 5))
+     paste("p =", res, sep = " ")
+ }
> fisher.sign(schuhgroesse, 40.2)

[1] "p = 0.3952"
```

4.1.2. Vergleich zweier Stichproben

Wenn zwei **gepaarte Stichproben** vorliegen, d.h. jedem $i$ten Wert x_{1i} der einen Stichprobe (x_1) ist eindeutig der $i$te Wert x_{2i} der anderen Stichprobe (x_2) zugeordnet, so unterscheidet sich die Analyse nur wenig von dem Vergleich einer Stichprobe mit einem konstanten Wert (Gleichungen 4.1 und 4.2). Statt der Konstanten A wird jetzt der gepaarte x_{2i}-Wert subtrahiert. Die t-Test-Gleichung sieht jetzt so aus:

$$t_s = \frac{\sum_{i=1}^{n}(x_{1i} - x_{2i})}{SE_{x_{1i}-x_{2i}}} \tag{4.3}$$

Diese gepaarten Stichproben tauchen sowohl bei einfachen experimentellen Anordnungen auf (z.B. Behandlung und Kontrolle in einem Block), als auch bei deskriptiven Untersuchungen, etwa der Arthropodenartenzahl auf der südlichen und nördlichen Seite mehrerer Bäume.

Für einen Datensatz mit den beiden Variablen `arten.nord` und `arten.sued` kann man entsprechend den folgenden t-Test durchführen:

```
> arten.nord <- c(12, 23, 15, 18, 20)
> arten.sued <- c(5, 8, 7, 9, 9)
> t.test(arten.nord, arten.sued, paired = TRUE, alternative = "two.sided")

        Paired t-test

data:  arten.nord and arten.sued
t = 7.0711, df = 4, p-value = 0.002111
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  6.073514 13.926486
sample estimates:
mean of the differences
                     10
```

Für nicht-parametrische Verfahren wird die Wilcoxon-Funktion verändert (x_2 statt A) und unser selbstdefinierte Fishers Vorzeichentest ist ebenfalls anwendbar.

```
> wilcox.test(arten.nord, arten.sued, paired = TRUE)

        Wilcoxon signed rank test

data:  arten.nord and arten.sued
V = 15, p-value = 0.0625
alternative hypothesis: true location shift is not equal to 0

> fisher.sign(arten.nord, arten.sued)
```

```
[1] "p = 0.03125"
```

Liegen die Stichproben **ungepaart** vor, d.h. der erste Wert von x_1 hat keinen Bezug zum ersten Wert von x_2, so berechnet sich der t-Test wie folgt:

$$t = \frac{\text{Differenz der Mittelwerte}}{\text{Standardfehler der Differenzen}} = \frac{\bar{x}_1 - \bar{x}_2}{SE_{\text{Differenzen}}} \ , \tag{4.4}$$

$$\text{wobei } SE_{\text{Differenzen}} = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

Dies ist der t-Test in seiner allgemeinen Form. Wie zu sehen ist, müssen die beiden Stichproben nicht die gleiche Varianz haben: sie gehen als s_1^2 bzw. s_2^2 in die Formel ein. Beachte, dass der t-Test für *ungleiche* Varianzen ungleich einer *oneway*-ANOVA ist (Verzani 2002)!

Der Syntax in R vereinfacht sich dabei, da in der Funktion `t.test` die Option `paired=FALSE` die Grundeinstellung ist (wie übrigens auch `alternative="two.sided"`), und entsprechend weggelassen werden kann.

```
> t.test(arten.nord, arten.sued)

        Welch Two Sample t-test

data:  arten.nord and arten.sued
t = 4.8679, df = 5.196, p-value = 0.00415
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  4.778713 15.221287
sample estimates:
mean of x mean of y
     17.6       7.6
```

Beachte, dass sich das Signifikanzniveau verschlechtert hat, da ja nun weniger Informationen genutzt wurden.

4.1.3. Vergleich der Varianzen zweier Stichproben

Eine der wichtigsten Annahmen der Varianzanalyse (siehe unten) ist die der Varianzhomogenität oder Homoskedastizität. Um zu testen, ob zwei Stichproben x_1 und x_2 die gleich Varianz haben, benutzt man den F-Test. Wir berechnen die Varianz (s^2) der jeweiligen Stichprobe, s_1^2 und s_2^2, und teilen den größeren Wert durch den kleineren: $F = s_1^2/s_2^2$. Kritische F-Werte können in Tabellen nachgeschlagen werden.

Der p-Wert für ein bestimmtes F-Verhältnis kann mit der Wahrscheinlichkeitsfunktion der F-Verteilung berechnet werden: `pf(F-Wert, n1-1, n2-1)` Alternativ kann man den kritischen F-Wert für ein bestimmtes Signifikanzniveau (etwa 0.05%) mit der Quantilenfunktion nachschlagen: `qf(0.95, n1-1, n2-1)`.
Noch einfacher geht's mit dem R-Befehl textttvar.test. Dieser kann sogar den *output* linearer Modelle vergleichen (siehe Abschnitt 6).

```
> set.seed(3)
> x <- rnorm(50, mean = 0, sd = 2)
> y <- rnorm(30, mean = 1, sd = 1)
> var.test(x, y)

        F test to compare two variances

data:  x and y
F = 4.3741, num df = 49, denom df = 29, p-value = 6.326e-05
```

```
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 2.197665 8.229562
sample estimates:
ratio of variances
          4.374131
```

```
> var.test(lm(x ~ 1), lm(y ~ 1))

        F test to compare two variances

data:  lm(x ~ 1) and lm(y ~ 1)
F = 4.3741, num df = 49, denom df = 29, p-value = 6.326e-05
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 2.197665 8.229562
sample estimates:
ratio of variances
          4.374131
```

Beide Test liefern hier natürlich das gleiche Ergebnis, da die linearen Modelle ja nur eine Konstante beinhalten.

4.1.4. Vergleich von Proportionen

Der einfachste Test hier nutzt die z-Statistik. Nehmen wir zwei Proportionen $\hat{p}_1 = x_1/n_1$ und $\hat{p}_2 = x_2/n_2$, dann ist:

$$z = \frac{\hat{p}_1 - \hat{p}_2}{\sqrt{\frac{\hat{p}_1(1-\hat{p}_2)}{n_1} + \frac{\hat{p}_2(1-\hat{p}_2)}{n_2}}} \tag{4.5}$$

Der z-Wert kann entsprechend nachgeschlagen werden.

Nehmen wir als Beispiel an, dass 2 von 20 Jungen einer Schule eine *base cap* tragen, aber 7 von 120 Mädchen, so ist $\hat{p}_{\text{boys}} = 2/20$ und $\hat{p}_{\text{girls}} = 7/120$. Dann errechnet sich die Wahrscheinlichkeit, dass Jungen und Mädchen in unterschiedlichen Proportionen *base caps* tragen als:

```
> prop.test(c(2, 7), c(20, 120))

        2-sample test for equality of proportions with continuity correction

data:  c(2, 7) out of c(20, 120)
X-squared = 0.0445, df = 1, p-value = 0.8329
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.1255037  0.2088370
sample estimates:
    prop 1     prop 2
0.10000000 0.05833333
```

Beachte, dass der erste Vektor die „Erfolge“ (hier *base cap* tragende SchülerInnen) und er zweite die Gesamtmenge (Junge bzw. Mädchen) enthält. Siehe `?prop.test` für weiter Hinweise und Beispiele.

Im Prinzip kann dieser Test auch wiederholt auf mehr als zwei Proportionen angewandt werden (etwa ein Vergleich von drei verschiedenen Bestäubungsmechanismen – Wind, Insekten, selbstbestäubt – einer tropischen mit einer temperaten Flora). Dann fassen wir zwei Klassen zusammen (etwas Wind und Insekten) und testen dies gegen die dritte (selbstbestäubt). Danach wiederholen wir diese Prozedur für alle drei Kombinationen. Damit vergleichen wir in jeder Klasse die Proportion des Bestäubungsmechanismus zwischen der tropischen und temperaten Flora. Zum Schluss müssen wir noch den erhaltenen p-Wert durch die Anzahl durchgeführter Vergleiche, hier also 3, teilen, da wir die gleichen Daten für wiederholte Test benutzt haben (Bonferroni-Korrektur; siehe Abschnitt 1.3.1 auf Seite 5).

4.1.5. Test auf Assoziation: der X^2 Test

Daten zweier kategorischer Faktoren lassen sich in einer sog. Kontingenztabelle (auch Kreuztabelle oder 2×2-Tabelle, engl. *contingency table*), zusammenfassen. Im folgenden wollen wir uns anschauen, ob die zwei Faktoren, die diese Tabelle aufbauen, miteinander assoziiert sind.

Hier haben wir beispielsweise das Ergebnis der Befragung einer Gruppe Rentner bezüglich des Zusammenhangs zwischen Rauchen und Geschlecht:

	Raucher	Nichtraucher
Männer	1	14
Frauen	12	26

oder allgemein:

	A1	A2
B1	a	b
B2	c	d

Nähmen wir an, dass Geschlecht und Rauchen unabhängig voneinander sind, so ließe sich für jede Zelle ein Erwartungswert berechnen, und zwar als Wahrscheinlichkeit das Geschlecht zu besitzen multipliziert mit der Wahrscheinlichkeit zu rauchen (Gleichung 2.2), multipliziert mit der Gesamtanzahl Beobachtungen ($N = a + b + c + d$).

	Raucher	Nichtraucher
Männer	$\frac{(1+14)}{53} \cdot \frac{(1+12)}{53} \cdot 53$	$\frac{(1+14)}{53} \cdot \frac{(14+26)}{53} \cdot 53$
Frauen	$\frac{(1+12)}{53} \cdot \frac{12+26)}{53} \cdot 53$	$\frac{(12+26)}{53} \cdot \frac{(14+26)}{53} \cdot 53$

bzw.

	A1	A2
B1	$\frac{(a+b)\cdot(a+c)}{N}$	$\frac{(a+b)\cdot(b+d)}{N}$
B2	$\frac{(a+c)\cdot(c+d)}{N}$	$\frac{(b+c)\cdot(c+d)}{N}$

Der X^2**-Test** wird dann wie folgt berechnet:

$$X^2 = \sum \frac{(O-E)^2}{E} \tag{4.6}$$

wobei O = *observed* die beobachteten Daten sind, und E = *expected* die erwarteten, errechneten Wahrscheinlichkeiten aus dem zweiten Satz Tabellen. Die erste Zelle ist also $O_1 = a$, $E_1 = \frac{(a+b)\cdot(a+c)}{N}$, und entsprechend ist der erste Summand

$$X_1^2 = (a - \frac{(a+b)\cdot(a+c)}{N})^2 / \frac{(a+b)\cdot(a+c)}{N}$$

Die entspechende Verteilung, wenig überraschend X^2-Verteilung genannt, ist in diversen Bücher tabelliert.

Ein Wort der Vorsicht: Der X^2- (Chi-Quadrat)-Test darf nur benutzt werden, wenn die Anzahl *erwarteter* Elemente einer Kategorie > 5. Sonst müssen wir auf Fishers Exakten Test zurückgreifen, eine Erweiterung des binomialen Tests (= Fishers Vorzeichentest) zum Test von zwei Proportionen[4]. Da bei so geringen Datenmengen aber der Zufall eine zu große Rolle spielt, können wir erwarten, dass dieser Test sehr konservativ ist, und nur in extremen Fällen eine Assoziation erkennen wird.

Um den X^2-Test durchzuführen, müssen die Daten erst in die Form einer 2×2-Matrix gebracht werden. (Die Option `byrow=FALSE` führt dazu, dass die Daten spaltenweise in die Matrix plaziert werden.)

```
> Rauchen <- matrix(c(1, 12, 14, 26), nrow = 2, byrow = FALSE)
> Rauchen
```

```
     [,1] [,2]
[1,]    1   14
[2,]   12   26
```

Jetzt können wir den Test durchführen:

```
> chisq.test(Rauchen)
```

[4] http://mathworld.wolfram.com/FishersExactTest.html

```
        Pearson's Chi-squared test with Yates' continuity correction

data:  Rauchen
X-squared = 2.3854, df = 1, p-value = 0.1225
```

In diesem Datensatz besteht anscheinend kein Zusammenhang zwischen Geschlecht und Rauchen.
Mit folgendem Befehl kann man R dazu bringen, die Erwartungswerte auszugeben. Diese müssen alle > 5 sein, damit der Test gültig ist.

```
> chisq.test(Rauchen)$expected

         [,1]     [,2]
[1,] 3.679245 11.32075
[2,] 9.320755 28.67925
```

Wir sehen, dass für die erste Eintragung die Anzahl erwarteter Werte kleiner als 5 ist. Entsprechend sollten wir den X^2-Test nicht benutzen, sondern Fischer's Exact Test.
Fishers Test als Alternative zum X^2-Test ist in R auch als solcher implementiert:

```
> fisher.test(Rauchen)

        Fisher's Exact Test for Count Data

data:  Rauchen
p-value = 0.08013
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.003387818 1.292748350
sample estimates:
odds ratio
 0.1590004
```

Hier entdecken wir also einen signifikanten Zusammenhang zwischen Geschlecht und Rauchen.

4.1.6. Permutationstests

Abschließend sei noch auf **Permutationstests** hingewiesen, bei denen die Zuordnung von Daten zufällig verändert wird, nicht aber die Datenwerte. Beispielsweise (Crawley 2002, S. 204f.) haben wir Schnecken in zwei verschiedenen Vegetationstypen gezählte (A und B), jeweils 20 Replikate. Unsere Tabelle hat also in der ersten Spalte immer A oder B stehen, in der zweiten die Zahl der Schnecken. Wir können jetzt einfach die Vegetationsbezeichung randomisieren (ohne Zurücklegen) und die jeweiligen Mittelwerte neu berechnen. Wenn wir dies 1000 mal machen, sehen wir, ob unsere ursprüngliche Erhebung von einer zufällig erwarteten abweicht.

```
> Schnecken <- read.table("schnecken.txt", header = T)
> attach(Schnecken)
> set.seed(578)
> tapply(Anzahl, Vegetationstyp, mean)

    A     B
1.275 2.275

> diff(tapply(Anzahl, Vegetationstyp, mean))

B
1
```

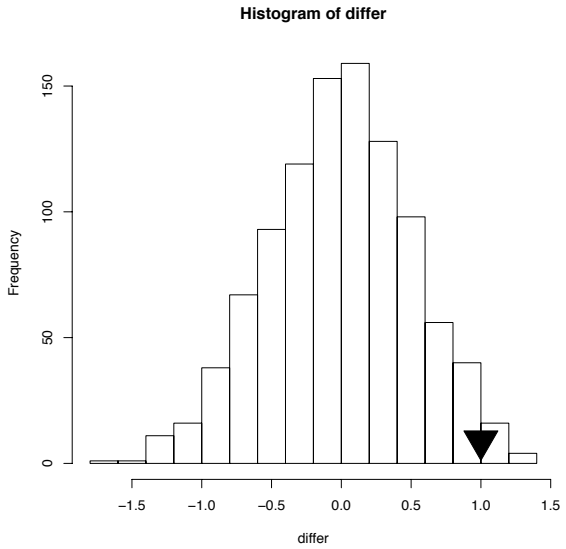


Abbildung 4.1.: Verteilung der Unterschiede zwischen den Schneckenzahlen beider Vegetationstypen für 1000 Randomisierungen. Das Dreieck gibt den Wert für den beobachteten Unterschied an.

```
> differ <- numeric(1000)
> for (i in 1:1000) {
+     Vegetationstypsample <- sample(Vegetationstyp)
+     differ[i] <- diff(tapply(Anzahl, Vegetationstypsample, mean))
+ }
> hist(differ)
> sum(differ < 1)

[1] 977

> points(1, 9, cex = 4, pch = 25, bg = "black")
> quantile(differ, 0.975)

97.5%
 0.95
```

In diesem Beispiel unterscheiden sich also unsere Vegetationstypen signifikant in ihrer Schneckendichte, da 977 der 1000 Permutationsdifferenzen kleiner als der beobachtete Wert von 1 sind. Die Quantilenfunktion `quantile` gibt den Wert an, ab dem der Unterschied signifikant ist (hier also ab Differenzen von 0.95). Abbildung 4.1 vermittelt ein Gefühl für die „Ungewöhnlichkeit“ unserer Messung. In diesem Fall sind viele Werte deutlich kleiner als der beobachtete Mittelwert, und Letzterer mithin signifikant nicht-zufällig.

4.2. Kovarianz und Korrelation

Die Beziehung zweier kontinuierlicher Datensätze werden durch die **Kovarianz**, die **parametrische Korrelation** oder die **nicht-parametrische Korrelation** numerisch zusammengefasst.

Kovarianz beschreibt, wie stark zwei Variablen in der gleichen Richtung variieren. Sie wird berechnet als:

$$s_{Y_1Y_2} = \frac{\sum_{i=1}^{n}(y_{i1} - \bar{y}_1)(y_{i2} - \bar{y}_2)}{n-1} \tag{4.7}$$

Dabei steht $s_{Y_1Y_2}$ für die Kovarianz der Variablen Y_1 mit Y_2, $\bar{y}_i$ für den Mittelwert des Datensatzes Y_i, und y_{i1} für einen Datenpunkt aus dem Datensatz Y_1.

Wenn wir uns Daten mit großen Werten anschauen, so wird auch die Kovarianz dieser Daten sehr groß sein. Damit ist ein Vergleich von Kovarianzen aus unterschiedlichen Datensätzen nicht wirklich möglich. Eine Möglichkeit ist, die Kovarianz durch die Standardabweichungen der beiden Variablen zu teilen, so dass das sich ergebende Maß zwischen -1 und 1 liegt. Dies nennt man **Pearson-Korrelation**, und sie wird wie folgt berechnet:

$$r = \frac{\sum_{i=1}^{n}(y_{i1} - \bar{y}_1)(y_{i2} - \bar{y}_2)}{\sqrt{\sum_{i=1}^{n}(y_{i1} - \bar{y}_1)^2 \sum_{i=1}^{n}(y_{i2} - \bar{y}_2)^2}} \tag{4.8}$$

Der Standardfehler von $r_{Y_1Y_2}$ kann leicht berechnet werden als:

$$s_r = \sqrt{\frac{1 - r^2}{n - 2}} \tag{4.9}$$

Wenn die zwei Datensätze Y_1 und Y_2 in der gleichen Richtung gehen, so sind sie positiv korreliert, und $r_{Y_1Y_2}$ ist positiv. Sind sie gegenläufig (Y_1 nimmt ab, wenn Y_2 zunimmt), so sind sie negative korreliert, und $r_{Y_1Y_2} < 0$. Je näher $r_{Y_1Y_2}$ an 1 oder -1 ist, umso stärker ist die Korrelation der beiden Variablen. Dieser Stärke kann auch ein Signifikanzwert zugeordnet werden, wenn die Daten normalverteilt sind. Dieser Test ist ein t-test, der die Korrelation mit ihrem Standardfehler vergleicht: $t = \frac{r}{s_r}$.

Bei der Pearson-Korrelation gehen die Werte der Variablen in die Berechnung der Korrelation ein. Da aber der t-test eine Normalverteilung der Daten voraussetzt, können nur entsprechende Daten auf Korrelation getestet werden. **Nicht-parametrische Korrelation** hingegen ersetzt die absoluten Werte durch den Rang innerhalb des Datensatzes, und korreliert dann die Ränge (genannt Spearmans Rang Korrelation). Eine Alternative ist Kendalls τ („Tau“).[5]

Schauen wir uns das an einem Beispiel an. Abbildung 4.2 zeigt einen *Scatterplot*, also eine Auftragung von Y_2 gegen Y_1.[6] Die Kovarianz entsprechend obiger Formel berechnet sich mittels der Werte in Tabelle 4.1.

Somit beträgt die Kovarianz der Variablen Y_1 und Y_2, $s_{Y_1Y_2} = 114.47/9 = 12.73$. Die Korrelation errechnet sich als $114.47/\sqrt{94.20 \cdot 168.41} = 0.909$, mit einem Standardfehler von $s_r = \sqrt{(1 - 0.909^2)/(10 - 2)} = 0.147$.[7] Dasselbe kann man nun mit den Rängen machen (in Klammern angegeben in Tabelle 4.1). Dabei kommt dann folgenden Wert für Spearmans r_s heraus: $r_s = 0.879$. Und für Kendalls Korrelation: $\tau = 0.778$.

Generell sind die nicht-parametrischen (nicht die wirklichen Zahlen, sondern nur ihre Rangfolge berücksichtigenden) Methoden etwas weniger sensibel als parametrische, d.h. ein möglicher Zusammenhang wird eher übersehen. Andererseits sind sie deutlich robuster hinsichtlich der Datenverteilung.

Die Berechnung der Kovarianz und Korrelation ist in R mittels der Funktionen `var` bzw. `cor` implementiert.

[5] Diese berechnet sich wie folgt: Alle Werte der Variablen Y_1 und Y_2 werden Rang-transformiert. Dann wird nach Y_1 sortiert. Wenn ein Wertepaar (y_1, y_2) den gleichen Rang hat, so zählt dies als *concordant*, andernfalls als *discordant* (*ties*, d.i. gleiche Ränge innerhalb einer Variablen, werden nicht gezählt). Sei K = Anzahl *concordant* − Anzahl *discordant* Datenpaare. Dann ist $\tau = \frac{2K}{n(n-1)}$.

[6] Es werden Ys als Namen gewählt, um deutlich zu machen, dass keine der beiden kausalen Einfluss auf die andere hat. X und Y würden suggerieren, dass Y die von X abhängige Variable ist.

[7] Der t-Test liefert somit den Wert $t = 0.909/0.147 = 6.187$, das entspricht einer Wahrscheinlichkeit von 0.000133 bei einer t-Verteilung mit $n - 2 = 8$ Freiheitsgraden (`pt(6.187, 8, lower.tail=F)`). Diesen Wert müssen wir noch mit 2 multiplizieren, da wir ja einen zweiseitigen Test durchführen.

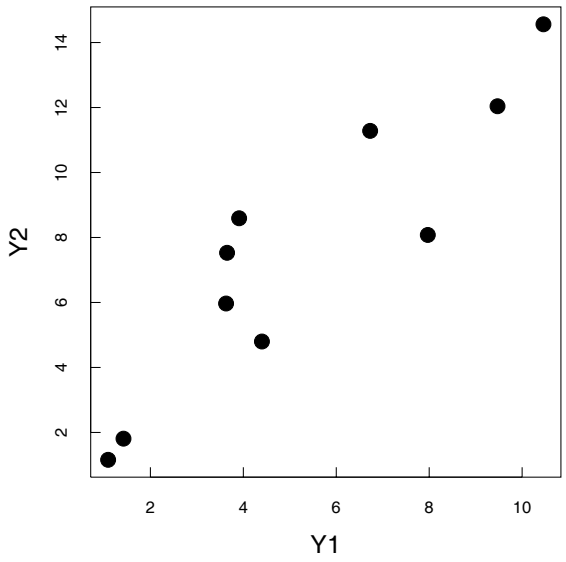


Abbildung 4.2.: Scatterplot der Variablen Y_1 und Y_2.

Tabelle 4.1.: Beispielrechnung zur Kovarianz und Korrelation.

Y_1 (Rang)	Y_2 (Rang)	$Y_1 - \bar{Y}_1$	$Y_2 - \bar{Y}_2$	$(Y_1 - \bar{Y}_1)(Y_2 - \bar{Y}_2)$	$(Y_{i1} - \bar{Y}_1)^2$	$(Y_{i2} - \bar{Y}_2)^2$
1.08 (1)	1.16 (1)	−4.18	−6.42	26.86	17.50	41.24
1.42 (2)	1.81 (2)	−3.85	−5.77	22.24	14.85	33.32
3.63 (3)	5.97 (4)	−1.64	−1.61	2.65	2.70	2.60
3.65 (4)	7.53 (5)	−1.62	−0.05	0.08	2.63	0.00
3.91 (5)	8.59 (7)	−1.36	1.01	−1.37	1.86	1.02
4.40 (6)	4.80 (3)	−0.87	−2.78	2.43	0.76	7.74
6.73 (7)	11.28 (8)	1.46	3.70	5.39	2.12	13.68
7.97 (8)	8.08 (6)	2.70	0.50	1.34	7.27	0.25
9.47 (9)	12.04 (9)	4.20	4.46	18.71	17.61	19.87
10.46 (10)	14.55 (10)	5.19	6.97	36.19	26.90	48.69
$\bar{Y}_1 = 5.27$	$\bar{Y}_2 = 7.58$			$\sum = 114.52$	$\sum = 94.20$	$\sum = 168.41$

```
> y1 <- c(1.08, 1.42, 3.63, 3.65, 3.91, 4.4, 6.73, 7.97, 9.47, 10.46)
> y2 <- c(1.16, 1.81, 5.97, 7.53, 8.59, 4.8, 11.28, 8.08, 12.04, 14.55)
> var(y1, y2)

[1] 12.72655

> cor(y1, y2)

[1] 0.909302
```

Spearmans und Kendalls Korrelation ist, ebenso wie der Signifikanztest der Pearson Korrelation, mit der Funktion `cor.test` abrufbar.

```
> cor.test(y1, y2)

        Pearson's product-moment correlation

data:  y1 and y2
t = 6.1804, df = 8, p-value = 0.0002650
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
```

```
 0.6542436 0.9786380
sample estimates:
     cor
0.909302

> cor.test(y1, y2, method = "spearman")

        Spearman's rank correlation rho

data:  y1 and y2
S = 20, p-value = 0.001666
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.8787879

> cor.test(y1, y2, method = "kendall")

        Kendall's rank correlation tau

data:  y1 and y2
T = 40, p-value = 0.0009463
alternative hypothesis: true tau is not equal to 0
sample estimates:
      tau
0.7777778
```

Teil II.

Univariate Statistik I: Das Lineare Modell

Die univariate Statistik beschäftigt sich mit nur einer Antwortvariablen zur Zeit. Statistische Modelle werden also immer nur auf eine einzige abhängige Variable zugeschnitten, unabhängig von der Anzahl erklärender Variablen. Damit wird implizit die Annahme gemacht, dass es Ursache und Wirkung gibt, und dass wir uns dafür interessieren, welche Variable ursächlich für den Wert einer abhängigen Variablen ist.

Als Beispiel: Die Körpergröße von Kindern hängt mit der von Mutter und Vater zusammen, genauso wie mit frühkindlicher Ernährung, körperlicher Belastung im Kindesalter, Luftverschmutzung, Stabilität des familiären Umfeldes usw. Jedes Element dieser Liste kann also als mitbestimmend für die eine Antwortvariable Körpergröße in der univariaten Statistik benutzt werden.

Die univariate Statistik ist so besonders beliebt, weil es uns anscheinend extrem schwer fällt, uns mit mehreren Dingen gleichzeitig zu befassen. Zwei Antwortvariablen im Auge zu behalten, während verschiedene Einflüsse sich verändern, ist uns kaum möglich. Natürlich gibt es auch mathematische und technische Gründe, weshalb univariate Verfahren einfacher, eleganter oder schneller lösbar sind.

Darüber hinaus liegt aber die univariate Statistik auch sämtlichen Verfahren der multivariaten Analyse zugrunde, stellt sozusagen einen vereinfachten Spezialfall dar. Ein Verständnis der univariater Methoden und Hintergründe ist deshalb auch für die multivariate Statistik unverzichtbar.

Wie so häufig in der Mathematik, lässt sich auch das Prinzip des Linearen Modells einfacher in einer Formel, als in einem Satz fassen. Ein Versuch: Als lineares Modell bezeichnet man ein statistisches Modell, dessen erklärende Variable(n) linear kombiniert werden, um die abhängige Variable vorherzusagen. Linear kombiniert bedeutet dabei (eigentlich) reine Additivität der erklärenden Variablen. Durch Rekodierung ist allerdings auch die Interaktion von Variablen in einem linearen Modell repräsentierbar.

Formal schreiben wir hierfür:

$$Y = \beta_0 + \sum_{i=1}^{n} \beta_i X_i + \sigma Z \tag{4.10}$$

Dabei ist Y der Datensatz der abhängigen Variablen, X_i sind die n unabhängigen Variablen, die über jeweils einen Koeffizienten β_0 und β_i mit Y verbunden sind, und σZ stellt einen normal-verteilten Fehlerterm (Mittelwert 0, Varianz σ) dar (häufig als ϵ dargestellt; Z wird auch als „Zufallsvariable“ bezeichnet).

Gemäß des einleitenden Beispiels ist also Y die Körpergröße und X_i die lange Liste an möglichen Einflussfaktoren. Die Stärke des Einflusses spiegelt sich in den spezifischen Koeffizienten β_i wider.

Wir werden uns dieser Formel in mehreren Schritten nähern. Der erste Schritt ergibt sich durch die extreme Vereinfachung auf eine erklärende Variable.

5. Das (Einfache) Lineare Modell: Eine Erklärende Variable

Mit nur einer erklärenden Variablen vereinfacht sich Gleichung 4.10 zu:

$$Y = \beta_0 + \beta_1 X + \epsilon \tag{5.1}$$

Diese Formel dürfte uns aus der Schulzeit bekannt vorkommen: $y = mx + b$, plus einem Fehlerterm (ϵ, entspricht einem „Rauschen"), da die gemessenen Daten ja immer etwas streuen. Wenn X also eine kontinuierliche Variable ist, dann haben wir es in dieser Formel mit einer Geradengleichung zu tun, und die entsprechende Statistik ist die Regression. Wenn hingegen X ein Faktor ist (mit beliebig vielen Stufen; etwa beim Vergleich der Reaktionsgeschwindigkeit verschiedene Enzyme), so gelangen wir zur einfaktoriellen ANOVA. Beginnen wir mit dem gebräuchlichsten Fall, der linearen Regression.

5.1. Lineare Regression

In den klassischen Test haben wir schon die Korrelation kennengelernt (Abschnitt 4.2 S. 52). Der Hauptunterschied zwischen Korrelation und Regression ist der folgende: Bei der Regression wird eine Variable als in ihren Werten von der anderen Variablen abhängig angenommen. Wir haben also eine *unabhängige* oder *erklärende Variable* (X; *predictor variable*) und eine *abhängige* oder *Antwortvariable* (y; *response variable*).

Die Ausweisung von x und y erfolgt aufgrund unseres Verständnisses, der Fragestellung oder unseren Interesses. Auch bei einer Regression wird, genau wie bei der Korrelation, *keine Ursache-Wirkung-Zusammenhang* postuliert, sondern nur nahegelegt! Wenn wir aber Variable x (Luftdruck) zur Vorhersage der Werte von y (Regenwahrscheinlichkeit) nutzen wollen, so heißt dies nicht, dass wir hier den Luftdruck als *ursächlich* für den Regen halten, sondern nur als damit korreliert.

Regression hat üblicherweise drei Zielstellungen (Quinn and Keough 2002): (1) Beschreibung des Zusammenhangs zwischen abhängiger und unabhängiger Variablen, (2) Beschreibung der Güte dieses Zusammenhangs, also wieviel Variabilität von y durch x vorhersagbar ist, und (3) Nutzung von x zur Vorhersage von y.

Das berühmte Beispiel vom synchronen Storch- und Babyrückgang ist ja kein Hinweis auf eine kausale Verknüpfung[1]. Nichtsdestotrotz können wir natürlich versuchen, die Geburtenrate durch die Häufigkeit von Störchen (oder umgekehrt) vorherzusagen. Die Genauigkeit dieser Vorhersage mag dann allerdings gering sein und logisch minderwertig.

5.1.1. Berechnung der Parameter

Wie finden wir die Gerade, die die gemessenen Wertepaare (x_i, y_i) am besten annähert? Ein Punkt dieser Geraden ist einfach gefunden: ($\bar{x}$, $\bar{y}$), der Mittelwert von x und y (Schwerpunkt genannt). Dieser Schwerpunkt liegt sozusagen genau in der Mitte aller Datenpunkte, und deshalb muss eine Regression auch durch diesen Punkt gehen.

[1] Es gäbe hier ja mindestens zwei: Der Babyrückgang kann zur Massenabwanderung aus Arbeitslosigkeit bei Störchen geführt haben, oder (weit weniger intuitiv), die Störche bringen die Babies, ein Rückgang in der Storchpopulation hat entsprechende Effekte für die Babylieferung.

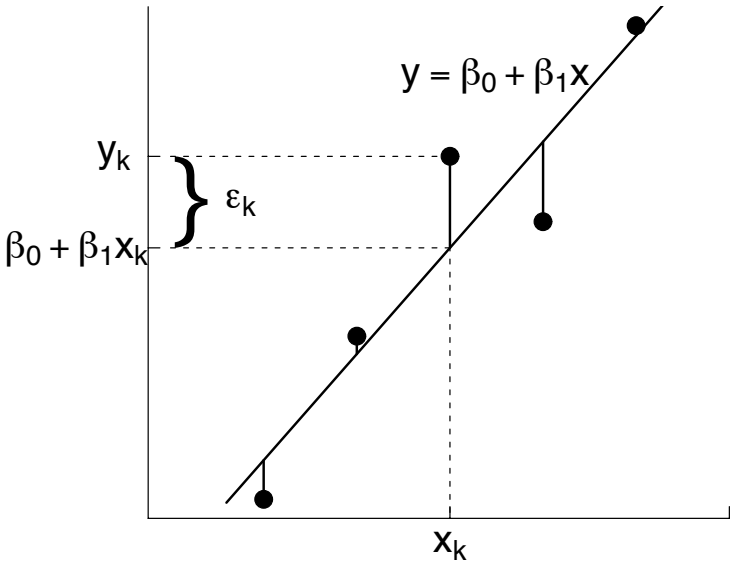


Abbildung 5.1.: Ein Regression durch fünf Datenpunkte. Die Linien zwischen der Regressionsgeraden und den Messpunkten stellen die Abweichung des Modells von den Daten dar. Für einen Wert x_k kann entsprechend ein Abweichungswert ϵ_k berechnet werden, als Differenz zwischen y_k und dem Regressionswert $\beta_0 + \beta_1 x_k$.

Die beste Regression ist diejenige, die den geringsten Abstand zu allen Punkten hat. Mathematisch ist dies durch das Abweichungsquadrate berechenbar (Abbildung 5.1). Wir addieren die quadrierten Unterschiede zwischen den vorhergesagten und tatsächlichen y_i-Werten und suchen den Wert von β_0 und β_1, die diesen Wert minimieren.

$$f(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_i))^2 = \sum_{i=1}^{n} (y_i^2 - 2\beta_0 y_i - 2\beta_1 x_i y_i + 2\beta_0\beta_1 x_i + \beta_1^2 x_i^2 + \beta_0^2) \tag{5.2}$$

Das Quadrieren führt dazu, dass nicht die Richtung der Abweichung zählt, sondern allein ihr Wert. Wie finden wir nun das Minimum? Wir erinnern uns, dass bei einem Minimum (und Maximum) einer Funktion ihre erste Ableitung 0 ist. Da hier zwei Parameter bestimmt werden müssen, setzen wir ihre partiellen Ableitung 0 und erhalten:

$$f'_{\beta_0}(\beta_0, \beta_1) = 2\sum_{i=1}^{n} (\beta_1 x_i + \beta_0 - y_i) = 0$$

$$f'_{\beta_1}(\beta_0, \beta_1) = 2\sum_{i=1}^{n} x_i(\beta_1 x_i + \beta_0 - y_i) = 0$$

Mit einer gewissen mathematischen Hartnäckigkeit, die hier nicht am Platze ist (siehe etwa Crawley 2002, S. 226), lässt sich zeigen, dass es genau eine Lösung gibt, dass hier tatsächlich ein Minimum vorliegt, und dass sich β_0 und β_1 dann wie folgt berechnen:

$$\beta_0 = \frac{\left(\sum_{i=1}^{n} x_i^2\right)\left(\sum_{i=1}^{n} y_i\right) - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} x_i y_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

$$\beta_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

Da der Schwerpunkt $(\bar{x}, \bar{y})$ auf der Regressionsgeraden liegt, reicht es einen der beiden obigen Parameter mit den angegebenen Formeln zu berechnen, und den anderen mittels Lösen der Gleichung: $\bar{y} = \beta_0 + \beta_1 \bar{x}$.

Einfacher können wir auch die Kurzversion schreiben (bei der wir allerdings zuerst die Mittelwerte von x und y berechnen müssen):

$$\beta_1 = \frac{\sum_{i=1}^{n}((x_i - \bar{x})(y_i - \bar{y}))}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \qquad \text{mit} \qquad se_{\beta_1} = \sqrt{\frac{\sum (y_i - \bar{y})^2}{(n-2)\sum (x_i - \bar{x})^2}} \tag{5.3}$$

$$\beta_0 = \bar{y} - \beta_1 x_i \qquad \text{mit} \qquad se_{\beta_0} = \sqrt{\frac{\sum (y_i - \bar{y})^2}{n-2}\left(\frac{1}{n} + \frac{\bar{x}^2}{\sum (x_i - \bar{x})^2}\right)} \tag{5.4}$$

Schließlich sei der Vollständigkeit halber noch die Berechnung des Fehlerterms aufgeführt, auch wenn dies ein wenig trivial ist:

$$\epsilon_i = y_i - \bar{y} \qquad \text{mit} \qquad se_\epsilon \approx \sqrt{\frac{\sum (y_i - \bar{y})^2}{n-2}} \tag{5.5}$$

Der Quotient $\frac{\sum (y_i - \bar{y})^2}{n-2}$ steht für die Varianz der Residuen. Das $n-2$ im Nenner ergibt sich aus der Tatsache, dass wir zwei Parameter schätzen müssen, und deshalb zwei Freiheitsgrade bei der Berechnung verlieren.

Die Nähe zur Korrelation zeigt sich an der Formel, die den Korrelationkoeffizienten (r; siehe Gleichung 4.8 auf Seite 53) von x und y mit β_1 verbindet:

$$\beta_1 = r\frac{s_y}{s_x} \tag{5.6}$$

wobei s_y und s_x die Standardabweichungen von y und x sind.

Wie in allen anderen Statistikprogrammen sehen wir auch in R wenig von diesen ganzen Formeln. Stattdessen erhalten wir die Parameterberechnungen und meist auch einen Signifikanztest dazu. Gehen wir die an einem Beispiel durch, z.B. mit den Daten aus Abbildung 5.1.

```
> x <- c(1, 2, 3, 4, 5)
> y <- c(4, 14, 25, 21, 33)
> fm <- lm(y ~ x)
> summary(fm)

Call:
lm(formula = y ~ x)

Residuals:
   1    2    3    4    5
-2.4  1.1  5.6 -4.9  0.6

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.100      4.795  -0.021   0.9847
x              6.500      1.446   4.496   0.0205 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.572 on 3 degrees of freedom
Multiple R-Squared: 0.8708,        Adjusted R-squared: 0.8277
F-statistic: 20.22 on 1 and 3 DF,  p-value: 0.02054
```

```
> anova(fm)

Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value  Pr(>F)
x          1  422.5   422.5  20.215 0.02054 *
Residuals  3   62.7    20.9
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Gehen wir den *output* kurz durch: Zunächst gibt uns R die Abweichungen zwischen beobachteten und berechneten y-Werten (*residuals*). Dann erhalten wir die Koeffizientenberechnungen. R nennt β_0 angemessenermaßen den y-Achsenabschnitt (*intercept*) und β_1 wird durch den Namen der erklärenden Variablen, hier x, kodiert. Die Parameterberechnung ist eine Schätzung der „wahren" Parameter, die diesem Zusammenhang zugrunde liegen, deshalb nennt R die Parameter *estimate*. Neben dem Parameterwert erhalten wir dessen Standardfehler und die Ergebnisse eines t-Tests, ob dieser Wert ungleich 0 ist. In unserem Fall ist der y-Achsenabschnitt nicht signifikant von 0 unterschiedlich, aber die Steigung schon. Die Sternchen hinter den p-Werten sind in der dann folgenden Zeile erklärt. Die letzten Zeilen fassen die Regression zusammen: Von der ursprünglichen Gesamtvarianz von y sind noch 4.57 übrig, berechnet mit 3 Freiheitgraden ($n-$Anzahl geschätzter Parameter). Die Regression reduziert damit die Varianz in y um 0.87 (siehe Abschnitt 5.1.4) oder (bei korrigierter Berechnung) 0.83. Analog zur ein-faktoriellen ANOVA ist somit ein F-Test möglich, ob die Regression signifikant ist, dessen Ergebnisse in der letzten Zeile angegeben sind. Da wir nur einen Faktor (x) haben, ist dieser Test identisch mit dem t-Test der Steigung.

Jetzt zur Ausgabe der Funktion `anova`[2]. Diese Art von Tabelle nennt man eine ANOVA-Tabelle, denn in ihr werden die Abweichungsquadrate (*sum of squares* = SS) für die einzelnen Effekte im Modell angegeben und mit den unerklärten Abweichungsquadraten verglichen. Entsprechend enthält die erste Zeile den Namen der Variablen, `x`, die Freiheitsgrade (`Df`) des Effekts, die Abweichungsquadrate (`Sum Sq`) für den Effekt, die mittleren Abweichungsquadrate (MSS = `Mean Sq` = `Sum Sq` / `Df`), den F-Wert (`F-value` = `Mean Sq` des Effekts / `Mean Sq` der Residuen) und ihre Signifikanz (`Pr(>F)`). In der nächsten Zeile sehen wir die entsprechenden Angaben für die Residuen, natürlich ohne Test auf Signifikanz.

Nur für Modelle mit einer erklärenden Variablen sind die P-Werte des `summary` und `anova`-Befehl identisch. Grundsätzlich ist der `anova`-Befehl der richtige zur Extraktion der Signifikanz, und der `summary`-Befehl lediglich für die Werte der Koeffizienten und ihrer Fehler nützlich.

5.1.2. Signifikanztests bei Regressionen

Grundsätzlich gibt es hier zwei unterschiedliche Dinge, die uns interessieren sollten: (1) Sind die Koeffizienten (Steigung und y-Achsenabschnitt) signifikant von Null unterschiedlich? und (2) Ist das Regressionsmodell als solches signifikant? Die erste Frage wird mittels eines t-Tests der Parameterschätzungen und seines Fehlers beantwortet.

Die zweite Frage hingegen testen wir mit einem F-Test genau wie bei einer einfaktoriellen Varianzanalyse (siehe Abschnitt 5.3.1, Seite 81). Zur Erinnerung: Beim F-Test wird der durch ein statistisches Modell erklärte Varianz mit der unerklärten Varianz verglichen. Bei Regressionen wird in diesem Fall nur eine erklärende Variable ins Modell aufgenommen („x"). Deshalb wird das Regressionsmodell nur dann als signifikant erkannt, wenn die Steigung signifikant ist. Entsprechend sind bei Regressionen die Ergebnisse des t-Tests und des F-Tests identisch.

Diese Unterscheidung soll an dieser Stelle nur helfen, den *output* verschiedener Statistikprogramme interpretierbar zu machen. Wenn wir später mehrere erklärende Variablen in ein

[2] Mit der Option `test="Chisq"` oder `test="F"` (siehe `?anova.lm`) im `anova`-Aufruf können wir wählen, welchen Test wir für die Signifikanzwerte wählen wollen. Dabei ist für die linearen Modelle die Annahme, dass das Verhältnis von Faktor- und Residuen-MSS einer F-Verteilung entspricht. Dies ist nur der Fall für normalverteilte Daten! Für nicht-normalverteilte Daten (siehe Das Verallgemeinerte Lineare Modell) nimmt man allgemeiner eine X^2-Verteilung an. Auf normalverteilte Daten angewandt ist diese etwas konservativer, aber meistens sehr ähnlich.

Modell hineinnehmen wird der Unterschied sichtbar. Dann sind u.U. mehrere Koeffizienten signifikant, ohne dass das Gesamtmodell signifikant ist!

5.1.3. Diagnostik/Transformationen

Nach dem Berechnen eines Modells müssen wir überprüfen, ob die der Methode zugrundeliegenden Annahmen auch erfüllt sind. Dies ist bei der Regression vor allem eine Untersuchung der Residuen auf Normalverteilung. Bei kleinen Stichproben ist natürlich keine klare Aussage über die Verteilungsform möglich: Wie soll man aus fünf Werten ein Histogramm konstruieren? Entsprechend müssen wir die im folgenden betrachteten Tests mit einer entspannten Zurückhaltung betrachten. Eine nicht-parametrische Regression, *jackknifing* oder robuste Verfahren (s.u.) können benutzt werden, um die Regressionergebnisse auf Robustheit zu überprüfen.

Der häufigst verwandte Test auf Normalverteilung ist der Kolmogorov-Smirnov-Test. Dabei werden die Werte der zu testenden Variablen ihrer Größe nach sortiert, und ihre kumulative Summe wird dann mit der einer spezifizierten kumulativen Normalverteilung (abgekürzt: $\Phi(\mu, \sigma)$) verglichen. Der maximale Unterschied zwischen einen Summenwert und der Normalverteilung, genannt D, darf dabei einen bestimmten Schwellenwert nicht übersteigen.[3]

Ein anderer Test auf Normalverteilung ist der Shapiro-Wilk-Test. Dafür wird eine Teststatistik W berechnet als

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_i\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{5.7}$$

Dabei sind a_i Wichtungskonstanten der Daten, die intern aus einem normalverteilten Vergleichdatensatz gleicher Größe berechnet werden. $\bar{x}$ ist der Mittelwert der Stichprobe.

Der Kolmogorov-Smirnov-Test ist in R als `ks.test` implementiert. Zusätzlich zu den Daten muss dabei der Mittelwert und die Standardabweichung der Grundgesamtheit angegeben werden. Dies sind schlicht Mittelwert und Standardabweichung der Residuen. Für den Shapiro-Wilk-Test geben wir nur die zu testenden Daten an.

```
> Residuen <- resid(fm)
> ks.test(Residuen, mean(Residuen), sd(Residuen))

        Two-sample Kolmogorov-Smirnov test

data:  Residuen and mean(Residuen)
D = 0.6, p-value = 1
alternative hypothesis: two.sided


> shapiro.test(Residuen)

        Shapiro-Wilk normality test

data:  Residuen
W = 0.9762, p-value = 0.9132
```

Beide halten also im vorliegenden Fall normalverteilte Residuen für extrem wahrscheinlich. Beachte: Die Null-Hypothese lautet, dass die Daten aus einer Normalverteilung stammen, und diese These wird nicht zurückgewiesen.

Deutlicher und instruktiver ist ein graphischer Normalverteilungs„test“. Dabei ist die Auftragungen der beobachteten Quantilen gegen die erwarteten sehr nützlich (*qq-plot*). Hier sehen wir die Abweichung von der Normalverteilung als eine Abweichung der Punkte von der Diagonalen. Wir suchen vor allem nach systematischen Abweichungen, z.B. nur im oberen oder

[3] Der Kolmogorov-Smirnov-Test ist auf andere Verteilungen verallgemeinerbar, sowie zum Vergleich zweier (normalverteilter) Stichproben nutzbar. Für Hinweise zur Umsetzung in R siehe Hilfe zur Funktion `ks.test`.

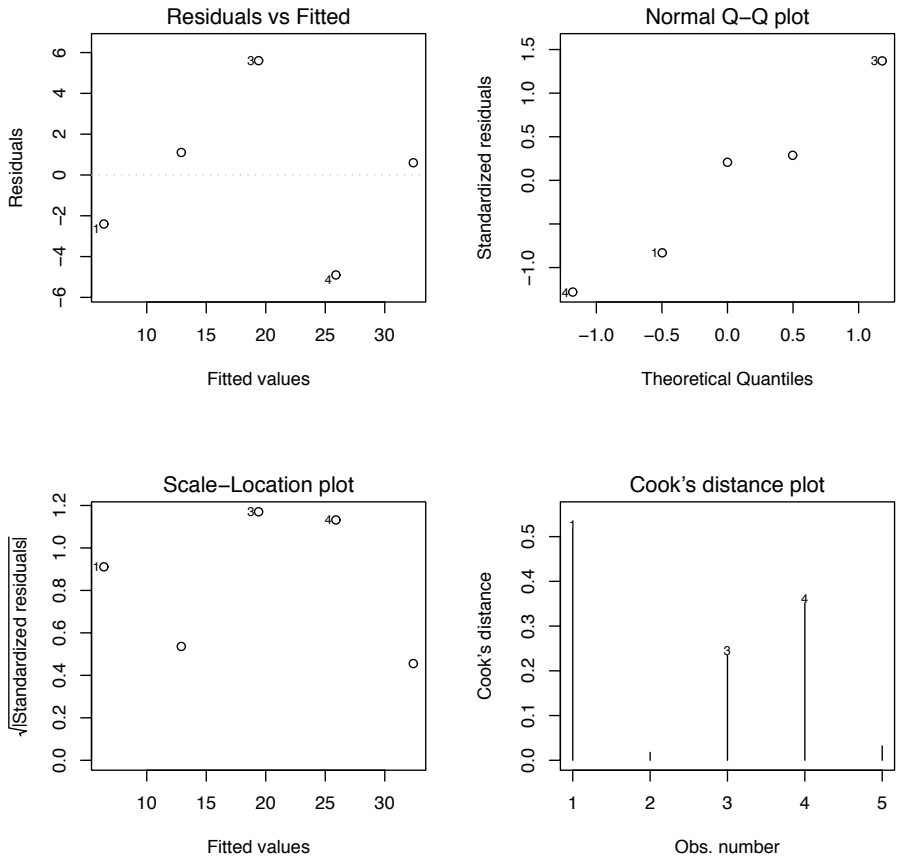


Abbildung 5.2.: Diagnostischer Standardplot für lineare Modelle in R. Siehe Text für Erklärungen.

unteren Teil der Quantilen. Eine gewisse Streuung ist allerdings unvermeidlich. Hier müssen wir mit der Zeit etwas Gefühl entwickeln.

Neben der Normalverteilung der Residuen müssen wir uns für das Phänomen der Varianzhomogenität interessieren. Wenn die gemessenen Werte bei hohen Werten stärker von den vorhergesagten abweichen als bei niedrigen Werten, so bedeutet dies, dass die Varianz der Daten mit ihren Werten zunimmt. Dies ist allerdings ein Verstoß gegen die Annahmen der Regression (wie auch anderer Analyseformen, wie etwa ANOVA). Hier wäre dann eine Transformation der Daten notwendig.

Für unsere Residuen können wir einen Quantilen-*plot* einfach mittels des Befehls `qqnorm(Residuen)` herstellen (nicht abgebildet). Für die oben durchgeführte Regression bietet R allerdings eine komfortablere Lösung an. Mittels des Befehls `plot(fm)` können wir verschiedene diagnostische Darstellungen der Regressionsanalyse ausgeben lassen (Abb. 5.2).

```
> qqnorm(Residuen)
> par(mfrow = c(2, 2))
> plot(fm)
```

Die Ausgabe des letzten Befehls sind vier *plots* unterschiedlicher Aussage. Im ersten werden schlicht die Residuen über die vorhergesagten Werte aufgetragen. Wir suchen nach einer systematisch größeren Streuung am rechten Ende, die natürlich bei so wenigen Datenpunkten nicht auffallen würde. Der zweite *plot* (oben rechts) ist eine Variante des `qqnorm`-*plots*. Dabei wurden die Werte standardisiert (siehe Abschnitt 3.3). Die dritte Abbildung zeigt ähnlich der ersten die Residuen gegen vorhergesagte Werte. Diesmal ist die Wurzel aus den Absolutwerten der Residuen abgetragen. Und schließlich zeigt die letzte Abbildung

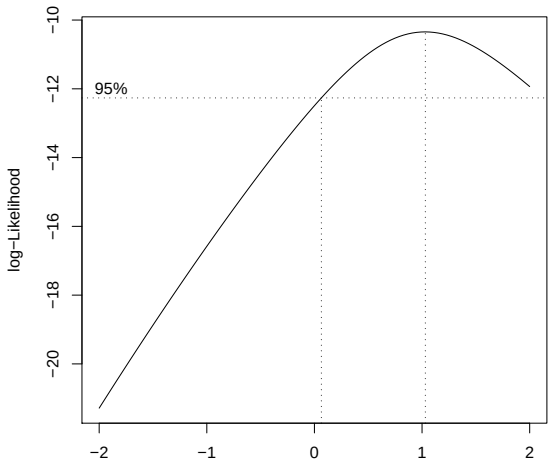


Abbildung 5.3.: Box-Cox-Transformation: Je größer die *log-likelihood* (d.h. je weniger negativ), desto erfolgreicher ist die Transformation. Der optimale Wert für λ kann dann auf der x-Achse abgelesen werden.

den Einfluss jeder Beobachtung für das Regressionsergebnis. Je größer der Wert (bezeichnet als *Cook's distance*), desto wichtiger ist diese Beobachtung.

Um zu sehen, ob eine Transformation die Qualität dieser Regression verbessern kann, führen wir eine Box-Cox-Transformation durch (siehe Gleichung 3.1).

```
> library(MASS)
> boxcox(y ~ x)
```

Wie wir sehen (Abbildung 5.3), schlägt R ein λ von 1 vor, d.h. keine Transformation der Daten! Hiermit ist die Diagnostik der Annahmen unserer Regression beendet. Läge λ abseits der 1, so müssten wir eine entsprechende Transformation durchführen, und danach nochmals die Residuen analysieren, wie oben beschrieben.

An dieser Stelle sei nochmals auf die Wichtigkeit einer grafischen Exploration der Daten hingewiesen. R^2-Werte geben keinen Aufschluss über die Verteilung der Datenpunkte. Abbildung 5.4 zeigt welch unterschiedliche Punktverteilungen die gleichen R^2-Werte liefern können. Ein *plot* der Residuen ist hier aufschlussreich. Dieser Datensatz (in R mitgeliefert als `data(Anscombe)`) ist so konstruiert, dass alle vier Regressionen die gleiche Steigung und das gleiche R^2 liefern. Bei oberflächlicher Betrachtung ohne Residuenanalyse hätten wir die Modelle also vielleicht akzeptiert!

5.1.4. Güte des Regressionsmodells: erklärte Varianz

Die Qualität einer Regression zeigt sich daran, wie gut die beobachteten Daten durch die Regression angenähert werden. Dafür können wir für jeden beobachteten Punkt mittels der Regressionsgeradengleichung einen Vorhersagewert berechnen. Diese zwei x-Werte-Datensätze können dann mittels Korrelation verglichen werden (siehe Abschnitt 4.2). Pearson's r (als Maß für die Stärke der Korrelation) wird quadriert als Maß für die Güte der Regression genommen (dann als r^2 oder R^2 bezeichnet). Dieses R^2 nimmt Werte zwischen 0 und 1 an, die dem Anteil von der Regression erklärter Variabilität angibt. Eine Regression mit einem R^2 von 0.4 erklärt also 40% der Varianz des ursprünglichen Datensatzes. Anders formuliert ist dies der Anteil, den die Residuen (d.i. beobachtete - vorhergesagte y-Werte) weniger streuen als die ursprünglichen Daten. Die Berechnung erfolgt genau wie in Formel 4.8 angegeben.

Andererseits interessiert uns, besonders bei komplizierteren Modellen, ob genau dieses Modell, mit genau diesen Parametern, von hoher oder niedriger Qualität ist. Dafür wird dann in die Berechnung der Varianz die Anzahl der Modellparameter miteinbezogen. Eine Regression hat ja üblicherweise zwei Parameter (y-Achsenabschnitt und Steigung), und die Varianz wird

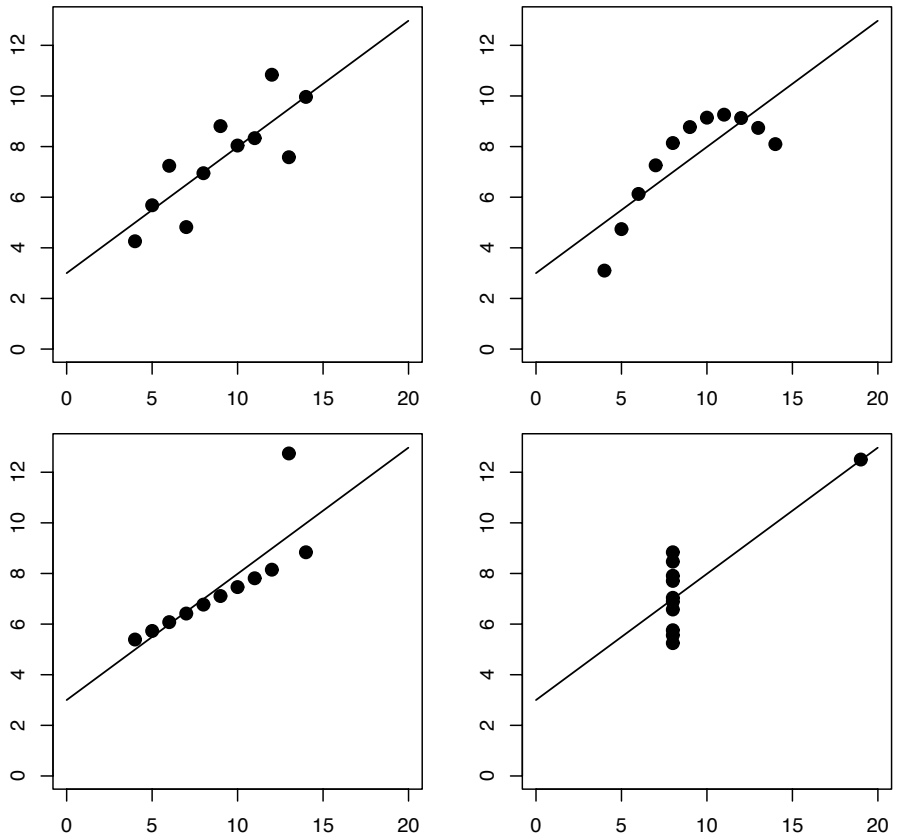


Abbildung 5.4.: Vier unterschiedliche Datensätze, die folgendes gemeinsam haben: Regressionsgerade: $y = 3.0 + 0.5x$, Anzahl Datenpunkte = 11, $R^2 = 0.67$, Abweichungsquadrate (SS) = 27.5 und der t-Test der Steigung = 4.24, $P = 0.002$. (Daten von Anscombe 1973).

dann nicht durch $n-1$ geteilt, sondern durch $n-$ Anzahl zu schätzender Parameter)$= n-2$. Diesen neuen R^2 bezeichnen wir als *adjusted* R^2. Wird hingegen die Regression durch den Ursprung erzwungen, oder soll nur ein Achsenabschnitt geschätzt werden, so bleibt es bei $n-1$, und der eigentliche und der *adjusted* R^2 sind identisch.

R gibt bei der `summary(lm(.))` gleich beide R^2-Werte an. Bemerke, dass der Standardfehler der Residuen (*residual standard error*) ebenfalls nach der Formel für die *adjusted* R^2 berechnet werden, d.h. auf Basis der Summe der Abweichungsquadrate durch $n-$ Anzahl Parameter. Siehe das obige Beispiel für entsprechenden R-*output*.

5.1.5. Regression durch den Ursprung und gewichtete Regression

Es gibt nur wenige Situationen, in denen wir eine Regression durch den Ursprung zwingen wollen. Denn wenn wir dies tun, testen wir nicht mehr die erhobenen Daten auf einen Zusammenhang, sondern prüfen direkt eine Hypothese, die einen y-Achsenabschnitt von 0 ergibt. Damit wird uns aber gleichzeitig ein Test auf die Signifikanz dieses Achsenabschnitts verwehrt.

Trotz dieser Vorbemerkungen, können wir natürlich eine Regression durch den Ursprung herbeiführen, z.B. um ein statistisches Modell zu vereinfachen: Wenn der y-Achsenabschnitt nicht signifikant ist, können wir ihn dann nicht weglassen?

Dazu müssen wir die im linearen Modell benutzte Formel etwas abwandeln.

```
> fm2 <- lm(y ~ x - 1)
> summary(fm2)
```

```
Call:
lm(formula = y ~ x - 1)

Residuals:
      1       2       3       4       5
-2.4727  1.0545  5.5818 -4.8909  0.6364

Coefficients:
  Estimate Std. Error t value Pr(>|t|)
x   6.4727     0.5339   12.12 0.000266 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.959 on 4 degrees of freedom
Multiple R-Squared: 0.9735,        Adjusted R-squared: 0.9669
F-statistic:   147 on 1 and 4 DF,  p-value: 0.0002656
```

Unsere Regressionsgeradensteigung hat sich also kaum verändert, ihr Standardfehler ist aber auf ein Drittel geschrumpft. Entsprechend sinkt auch der Wert des t-Tests auf ein hochsignifikantes Niveau. Die Residuen haben abgenommen, die erklärte Varianz zu, der F-Test verändert sich analog dem t-Test, allerdings mit einem Freiheitsgrad mehr, da wir ja jetzt nur einen Parameter schätzen. Ist dieses Regressionsmodell nun besser als das erste? Dafür betrachten wir die Gesamtresiduen (Summe der quadrierten Residuen).

```
> sum(resid(fm)^2)

[1] 62.7

> sum(resid(fm2)^2)

[1] 62.70909
```

Wir haben also einen Freiheitsgrad Unterschied, und die Modelle unterscheiden sich praktisch nicht in der Variabilität ihrer Residuen. Somit ist das einfachere Modell (ohne y-Achsenabschnitt) dem anderen überlegen. Im Zweifelsfall ist das Modell *mit* y-Achsenabschnitt vorzuziehen, da es weniger Annahmen an die Form des Modells stellt (weshalb es auch die Grundeinstellung in R ist).

Die gewichtete Regression sei hier auch nur kurz erwähnt, da sie mehr eine methodische Abwandlung der normalen Regression ist. Dafür schauen wir uns zunächst die gewichtete Berechnung eines Mittelwertes an. Nehmen wir an, wir haben die Länge eines Spatzenflügels an 12 Individuen gemessen, was zu einem Mittelwert von 8.7 cm führte. Zwei Kollegen haben dieselbe Messung an anderen Spatzen durchgeführt, und sie erhalten folgende Werte: 7.8 cm (20 Messungen) und 8.5 (5 Messungen). Wie können wir jetzt diese Mittelwerte am sinnvollsten zu einem zusammenfassen? Der Mittelwert aus 8.7, 7.8 und 8.5 ist 8.33. Sinnvoller ist es sicherlich, dem Mittelwert aus 20 Messungen mehr Genauigkeit beizumessen, als dem aus 5 Messungen. Wir könnten also einen Mittelwert berechnen, der gewichtet ist nach der Anzahl der Messungen: $(12 \cdot 8.7 + 20 \cdot 7.8 + 5 \cdot 8.5)/(12 + 20 + 5) = 8.19$. Dies bezeichnet man als gewichteten Mittelwert. Die allgemeine Formel dafür ist:

$$\bar{x}_w = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i} \tag{5.8}$$

wobei w_i der Wichtungsfaktor ist, im obigen Fall die Anzahl der Messungen.

Dies Prinzip können wir nun auf eine Regression übertragen. Nehmen wir an, dass wir neben den y-Werten auch noch eine Abschätzung über die Qualität dieses Wertes haben. So können diese y-Werte etwa Mittelwerte aus anderen Experimenten sein, die mit einem Maß für Variabilität qualifiziert sind.[4] Wenn nun ein y-Wert durch seine Standardabweichung

[4] Wenn wir alle Rohdaten haben, aus denen diese Mittelwerte berechnet wurden, dann sollten wir natürlich diese benutzen. Vor allem bei der Kombination von Literaturdaten ist dies aber nicht möglich.

als viel ungenauer erkennbar ist, als ein anderer, so wollen wir diesem weniger Gewicht bei der Berechnung der Regression geben: wir wollen die Regression nach der *Genauigkeit* der Datenpunkte wichten. üblich ist dabei, dass die Varianz des i-ten Datenpunktes (σ_i^2) zur Berechnung des Wichtungsfaktors (w_i) genutzt wird:

$$w_i = 1/\sigma_i^2 \tag{5.9}$$

Dies bedeutet einfach nur, dass je ungenauer ein Datenpunkt, desto weniger trägt er zur Regression bei. Die Wahl des Wichtungsfaktors hängt dabei ganz von den Umständen ab. Wenn wir Fehlerabschätzungen haben, so können wir Formel 5.9 benutzen, im Beispiel davor war es einfach die Stichprobengröße.[5]

Die Berechnung der Regressionsparameter erfolgt dann mit folgender Formel:

$$\beta_0 = \frac{\left(\sum_{i=1}^n w_i x_i^2\right)\left(\sum_{i=1}^n w_i y_i\right) - \left(\sum_{i=1}^n w_i x_i\right)\left(\sum_{i=1}^n w_i x_i y_i\right)}{\sum_{i=1}^n w_i \sum_{i=1}^n w_i x_i^2 - \left(\sum_{i=1}^n w_i x_i\right)^2}$$

$$\beta_1 = \frac{\sum_{i=1}^n w_i \sum_{i=1}^n w_i x_i y_i - \left(\sum_{i=1}^n w_i x_i\right)\left(\sum_{i=1}^n w_i y_i\right)}{\sum_{i=1}^n w_i \sum_{i=1}^n w_i x_i^2 - \left(\sum_{i=1}^n w_i x_i\right)^2}$$

Der Syntax hierfür ist denkbar einfach. Zusätzlich zu den abhängigen und unabhängigen Variablen geben wir noch eine Variable als Wichtungsfaktor in der Berechnung an. Im folgenden Beispiel ist der Einfluss der Wichtung allerdings nicht sonderlich stark. Nichtsdestotrotz erhalten wir andere Parameter für die Regression.

```
> x <- runif(10, 1, 5)
> y <- 2 * x + 4 + rnorm(10, 0, 1)
> w <- runif(10, 0, 1)
> summary(lm(y ~ x))

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-0.6401 -0.4207 -0.2455  0.5962  0.8777

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.5749     0.5960   7.675 5.88e-05 ***
x             1.8487     0.1848  10.003 8.47e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.622 on 8 degrees of freedom
Multiple R-Squared: 0.926,        Adjusted R-squared: 0.9167
F-statistic: 100.1 on 1 and 8 DF,  p-value: 8.468e-06

> summary(lm(y ~ x, weights = w))


Call:
lm(formula = y ~ x, weights = w)

Residuals:
```

[5] Ein Vorgriff auf spätere Kapitel: Wenn wir beispielsweise wissen, dass unerfahrene Vogelvermesser ungenauer messen als erfahrene, dann könnten wir das Alter als Wichtungsfaktor benutzen. Viel besser wäre es aber, Alter als weitere erklärende Variable mit in das linear Modell aufzunehmen.

```
    Min      1Q  Median      3Q     Max
-0.5243 -0.3214 -0.1771  0.3202  0.6878

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.8148     0.5775   8.338 3.24e-05 ***
x             1.8070     0.1887   9.576 1.17e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4671 on 8 degrees of freedom
Multiple R-Squared: 0.9198,        Adjusted R-squared: 0.9097
F-statistic:  91.7 on 1 and 8 DF,  p-value: 1.172e-05
```

5.1.6. Modell II und III Regression

Modell II und III-Regressionen (=Typ II und III-Regressionen) sind in der biologischen Literatur relativ selten anzutreffen. Im Zweifelsfall raten wir, dieses Thema noch einmal in einem formaleren Statistiklehrbuch nachzuschlagen (wie etwa in Legendre and Legendre 1998).

Bei biologischen Daten sind oftmals nicht nur die y-Werte mit einem Messfehler behaftet, sondern auch die x-Werte. Wenn wir zum Beispiel die Höhe eines Baumes gegen seinem Umfang regressieren, so haben wir beide Variablen im Feld mit einer gewissen Ungenauigkeit erfasst. Die klassische, oben durchgeführte Regression geht aber von fehlerfreien x-Daten aus (sogenannte Model I-Regression), und in ihre werden die Abweichungsquadrate nur in y-Richtung minimiert (also immer vertikal vom Messpunkt zur Regressionsgeraden).

Bei Typ II-Regressionen wird ein Fehler in der Messung der x-Werte angenommen, während die y-Werte fehlerfrei sind, so dass hier die Abweichungsquadrate in x-Richtung minimiert werden (also immer horizontal vom Messpunkt zur Regressionsgeraden).

Solange wir die x-Variable manipulieren (z.B. wenn wir Phytoplanktondichte entlang eines von uns erzeugten Nährstoffgradienten messen, der natürlich auch etwas schwanken kann), gilt der *Berkson case* (Sokal and Rohlf 1995), und wir können weiterhin mit der konventionellen Model I-Regression arbeiten.

Im Beispiel der Baumumfänge und -höhen ist dies aber nicht der Fall. Dort müssen wir auf eine neue Methode ausweichen, die *(reduced) major axis regression* (RMA, „Hauptachsenregression“). Dahinter verbirgt sich eine Verdrehung des Koordinatensystems, so dass x- und y-Achse so in den Datenpunkten liegen, dass sie möglichst viel der Varianz erklären. Ohne im Detail darauf eingehen zu wollen zeigt sich (Sokal and Rohlf 1995), dass die Steigung der Geraden (ν_{yx}) berechnet werden kann als der Quotient der Standardabweichungen von y und x:

$$\nu_{yx} = \pm\sqrt{\frac{\sum y^2}{\sum x^2}} = \pm\frac{s_y}{x_x} \tag{5.10}$$

Wenn die Variablen auf unterschiedlichen Skalen gemessen sind (wie im Beispiel Höhe und Umfang), so müssen wir vor der Berechnung der Standardabweichung (die ja von den Absolutwerten abhängig ist) eine Standardisierung der Werte vornehmen (siehe Gleichung 3.3 Seite 42). Dann erst können wir Formel 5.10 anwenden.

Was in einer RMA passiert ist, dass nicht die Abweichung zwischen der Regressionsgeraden an der Stelle x_i und dem dazugehörigen y_i-Wert berechnet wird, also der Abstand parallel zur y-Achse, sondern senkrecht zur Regressionsgeraden.

Wie wir im Teil zur multivariaten Statistik sehen werden, gibt es ein einfaches Verfahren, um diese *reduced major axis regression* durchzuführen: die Hauptkomponentenanalyse (*principle component analysis*: PCA). Die erste Achse der PCA ist genau die Gerade, die die Residuen senkrecht zur Geraden minimiert. Die Steigung dieser Graden, β_{RMA}, berechnet

sich etwas umständlich als geometrischer Mittelwert der Steigungen von x gegen y und y gegen x: $\beta_{RMA} = \sqrt{\beta_{xy} \cdot \beta_{yx}}$.

Eine selbstgeschriebene Funktion von Shigenobu Aoki[6] berechnet Steigung, Achsenabschnitt und die jeweiligen Standardfehler und 95%-Konfidenzintervalle. Beachte, dass bei der PCA die Daten standardisiert werden sollten, was zu Veränderungen im Wert des y-Achsenabschnitts führt.

```
> RMA <- function(x, y) {
+     CL95intercept <- (intercept <- mean(y) - (slope <- sign(cor(x,
+         y)) * sqrt(var(y)/var(x))) * mean(x)) + c(1, -1) * qt(0.025,
+         df <- (n1 <- (n <- length(x)) - 1) - 1) * (SEintercept <- sqrt((MSE
+         <- (var(y) - cov(x, y)^2/var(x)) * n1/df) *
+                         (1/n + mean(x)^2/var(x)/n1)))
+     CL95slope <- slope + c(1, -1) * qt(0.025, df) * (SEslope <-
+                         sqrt(MSE/var(x)/n1))
+     result1 <- c(slope, SEslope, CL95slope)
+     result2 <- c(intercept, SEintercept, CL95intercept)
+     names(result1) <- c("slope", "SE[slope]", "95% LCL", "95% UCL")
+     names(result2) <- c("intercept", "SE[int.]", "95% LCL", "95% UCL")
+     list(RMAslope = result1, RMAintercept = result2)
+ }
> y <- c(61, 37, 65, 69, 54, 93, 87, 89, 100, 90, 97)
> x <- c(14, 17, 24, 25, 27, 33, 34, 37, 40, 41, 42)
> RMA(x, y)

$RMAslope
    slope SE[slope]   95% LCL   95% UCL
2.1193664 0.3324959 1.3672085 2.8715243

$RMAintercept
intercept  SE[int.]   95% LCL   95% UCL
 12.19378  10.54975 -11.67141  36.05898
```

5.1.7. Vorhersage von y und x

Wenn wir eine Regressionsgerade wie oben berechnet haben, können wir auch für unbekannte x-Werte einen y-Wert vorhersagen. Der Wert ergibt sich einfach durch Einsetzen des gewünschten Wert x_1 in die Geradengleichung $y_1 = \beta_0 + \beta_1 x_1$.

Besonders komfortable ist in R für diesen Zweck die Funktion `predict`. Mit ihrer Hilfe können wir y-Werte sowohl zu einzelnen als auch zu mehreren gegebenen x-Werten ausgeben lassen.

```
> y <- c(61, 37, 65, 69, 54, 93, 87, 89, 100, 90, 97)
> x <- c(14, 17, 24, 25, 27, 33, 34, 37, 40, 41, 42)
> fm <- lm(y ~ x)
> predict(fm, list(x = 5))

[1] 29.11659
```

Für mehrere Werte können wir entweder der Liste diese Werte geben, etwa: `list(x=c(1,2,3))`, oder besser noch die Option `newdata` nutzen. Dann müssen wir einen `data.frame` mit den x-Werten vorgeben. Zusätzlich können wir mittels der Option `se.fit=T` auch noch Standardfehler für eben diese vorhergesagten y-Werte berechnen lassen.

```
> neu <- data.frame(x = 11:20)
> predict(fm, neu, se.fit = T)
```

[6] `http://aoki2.si.gunma-u.ac.jp/R/RMA.html`

```
$fit
       1        2        3        4        5        6        7        8
40.33632 42.20628 44.07623 45.94619 47.81614 49.68610 51.55605 53.42601
       9       10
55.29596 57.16592

$se.fit
       1        2        3        4        5        6        7        8
7.129127 6.830342 6.534815 6.243007 5.955466 5.672840 5.395903 5.125576
       9       10
4.862962 4.609379

$df
[1] 9

$residual.scale
[1] 10.15363
```

Wenn wir die Geradengleichung $y_1 = \beta_0 + \beta_1 x_1$ nach x_1 umformen, können wir auch aus bekanntem y den dazugehörigen x-Wert vorhersagen: $x_1 = \frac{y_1 - \beta_0}{\beta_1}$. Deutlich schwieriger wird die Berechnung des Fehlers auf den vorhergesagten x-Werten. Der Grund ist, dass ja der y-Wert mit einem Fehler behaftet ist, der x-Wert aber nicht. Für eine Methode dazu siehe Sokal and Rohlf (1995, S. 492).

Dieses Problem tritt häufig auf in Versuchen, in denen experimentell die Mortalität in Abhängigkeit von der Konzentration eines Stoffes gemessen wird, und nachher, im Umkehrschluss, diejenige Konzentration gefunden werden soll, die gerade zum Tod der Hälfte der Testorganismen führte (LD50 = *lethal dose* 50% genannt). Da hier oft mehrere Faktoren eine Rolle spielen, und die Antwortvariable nicht kontinuierlich ist (sondern etwa % tot), sollte dieses Problem der *inverse prediction* eigentlich erst im Verallgemeinerten Linearen Modell behandelt werden.

Erst für die letzte Ausgabe ihres Buches haben Venables and Ripley (2002) eine entsprechende Funktion zur Verfügung gestellt. Diese ist zwar von ihrem Syntax eindeutig genau auf die Vorhersage des LD50 ausgelegt, kann aber genauso gut für unsere Zwecke „missbraucht" werden. Dafür müssen wir der Funktion `dose.p` aus dem *package* **MASS** drei Argumente mitgeben: 1. das Regressionsmodell, 2. welche Koeffizienten des Modells es zur Vorhersage nutzen soll, und 3. einen Vektor mit den zu benutzenden y-Werten. Wir benutzen die letzte Regressionsgerade mit ihrem Modell `fm`. Die Koeffizienten sind Achsenabschnitt und Steigung, kodiert als `c(1,2)`. Die Grundeinstellung ist alle Koeffizienten zu benutzen, und entsprechend können wir das Argument auch weglassen. Bei komplizierten Modellen will man meist nur die Abhängigkeit des x von einen Faktor wissen und spezifiziert hier eben die entsprechenden Koeffizienten (siehe Hilfe zu `dose.p`). Neben dem vorhergesagten x-Wert wird auch ein Standardfehler angegeben.[7]

```
> library(MASS)
> dose.p(fm, cf = c(1, 2), p = c(70, 80, 90))

            Dose       SE
p = 70: 26.86331 1.751484
p = 80: 32.21103 1.669798
p = 90: 37.55875 2.077760
```

5.1.8. Steigung und Achsenabschnitt unterschiedlicher Regressionen vergleichen

Häufig will man die Ergebnisse verschiedener Regressionen vergleichen. Dabei gibt es zwei Wege. Der erste und zu bevorzugende ist eine Kovarianzanalyse (ANCOVA, siehe unten), in

[7] In Sokal and Rohlf (1995, S. 492f.) ist der Fehler auf dem geschätzten x-Wert leicht asymmetrisch, bei dieser Funktion hingegen symmetrisch. Auf je mehr Werten die Regression beruht, umso geringer ist aber die Asymmetrie.

der man die zu vergleichenden Systeme als Faktoren kodiert. Wollen wir beispielsweise vergleichen, ob sich die Abnahme der Artenzahl mit dem Breitengrad für Moose und Gefäßpflanzen unterscheidet, führen wir nicht zwei Regressionen durch und vergleichen diese, sondern fügen alle Daten in einer ANCOVA zusammen, wobei dann die Pflanzentyp×Breitengrad-Interaktion die obige Frage beantwortet. Das gleiche ist für logistische Regressionen und ähnliches möglich (dann allerdings in einem *generalized linear model*, GLM). Entstammen die Daten Veröffentlichungen und tragen deshalb eine Maß für ihre Genauigkeit (etwa Standardfehlerbalken), so kann man dies Maß in der ANCOVA als Wichtungsvariable mit integrieren.

Die Alternative zu einer ANCOVA ist ein einfacher t-Test. Es gilt, dass die Schätzung für Parameter normalverteilt sind, und somit einem Vergleich durch parametrische Verfahren zugänglich. In diesem Fall kann ein t-Test benutzt werden: der Vergleich zweier Mittelwerte ($\bar{x}$) mit unbekannter Varianz (σ^2) der Grundgesamtheit (Crawley 2002, S. 173f.). Es ist:

$$t = \frac{\bar{x}_A - \bar{x}_B}{\sigma^2_{\bar{x}_A - \bar{x}_B}} = \frac{\bar{x}_A - \bar{x}_B}{\sigma^2_A + \sigma^2_B} \tag{5.11}$$

Der gleichen Logik folgend können wir mit den Ergebnissen einer Regressionsanalyse auch testen, ob die Regression von einem erwarteten Wert abweicht. Dies ist dann einfach ein t-Test gegen einen festen Wert (siehe Abschnitt 4.1.1 auf Seite 45).

5.2. Nicht-lineare Regression und stückweise Regression

Die folgenden Themen sitzen im Abschnitt Lineares Modell eindeutig verkehrt: hier sind gerade Alternativen zum Linearen Modell dargestellt. Andererseits gehören nicht-linear Regression und stückweise Regression zum Thema Regression, und passen insofern doch hierhin.

5.2.1. Nicht-lineare Regression

Bei der nicht-linearen Regression wird eine Funktion durch die Daten gelegt, die keine Gerade ist. Da viele nicht-lineare Zusammenhänge aber linearisierbar sind (etwa Potenzfunktionen durch Logarithmisierung der x- und y-Werte), sprechen wir bei der nicht-linearen Regression im engeren Sinn nur dann, wenn eine solche Linearisierung nicht möglich ist. Die nicht-lineare Regression könnte dazu verleiten, nach einer Gleichung zu suchen, die die Daten möglichst gut annähert, ohne Rücksicht auf die Angemessenheit der Gleichung. In der Praxis spielt dieses Herumstochern keine Rolle, da es eine Unmenge Gleichungen gibt, die wir aber nicht alle fitten können oder wollen. Der Sinn wird vielmehr deutlich, wenn wir eine bestimmte, theoretisch abgeleitete Formel mit den Daten in Einklang bringen wollen.

Die Theorie hinter der nicht-linearen Regression ist denkbar einfach: Wir suchen die Parameter eines nicht-linearen Regressionsmodells, die dazu führen, dass die Abweichungsquadrate zwischen beobachteten und vorhergesagten y-Werten minimal sind.[8] Der Rest besteht in der technischen Umsetzung in der jeweiligen Software.

Beginnen wir mit einem einfachen Beispiel: Wir haben die Aktivität eines Enzyms in Abhängigkeit seiner Konzentration gemessen. Wir erwarten eine Michaelis-Menten-Kinetik der Form $v = \frac{v_{max} * c}{K_m + c}$, wobei v die Aktivität (Umsatzgeschwindigkeit) des Enzyms, v_{max} die maximale Aktivität, K_m die Halbaktivitätskonzentration (= MM-Konstante) und schließlich c die Konzentration des Enzyms ist. Zunächst generieren wir ein paar entsprechende Daten.

```
> noise <- rnorm(101, 0, 0.1)
> c.e <- seq(0, 10, 0.1)
> Km <- 1
> vmax <- 2
```

[8] Dieser Methode implizit ist die Annahme, dass die Daten normalverteilt sind. Nur dann sind die Abweichungsquadrate aus dem *maximum likelihood*-Ansatz ableitbar (siehe S. 20).

```
> v.e <- (vmax * c.e)/(Km + c.e) + noise
> plot(v.e ~ c.e)
```

Jetzt können wir unsere MM-Kinetik fitten. Dazu benutzen wir die R-Funktion `nls`. Die Modellformel wird spezifiziert, und den zu bestimmenden Parametern werden Anfangswerte zugedacht. Diese sollten ungleich Null sein! Sie sind lediglich der Ausgangspunkt einer iterativen Annäherung an die besten Parameterwerte und ihr Wert nicht mehr als eine erste Schätzung.

```
> fm <- nls(v.e ~ (vmax * c.e)/(Km + c.e), start = list(vmax = 1,
+     Km = 0.5))
> summary(fm)

Formula: v.e ~ (vmax * c.e)/(Km + c.e)

Parameters:
     Estimate Std. Error t value Pr(>|t|)
vmax  1.95864    0.02254   86.89   <2e-16 ***
Km    0.89835    0.05427   16.55   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.09252 on 99 degrees of freedom
```

Im *output* erscheint zusätzlich zu den Parametern (und einem t-Test auf Unterschiedlichkeit von 0) der Standardfehler der Residuen sowie eine Korrelationsmatrix der Parameter. Mit dieser hat es folgende Bewandtnis: Wenn wir einen Wert für `vmax` vorgeben, so gibt es einen optimalen `Km`-Wert. ändern wir `vmax`, so ändert sich auch dieser optimale `Km`. Die Korrelation der beiden gibt an, wie unabhängig sich der eine Parameter vom anderen wählen lässt, bzw. eben wie stark die beiden korreliert sind.
Um eine Regressionskurve in die Datenpunkte zu legen benutzen wir wieder die Funktion `predict()`. Das Ergebnis ist in Abbildung 5.5 dargestellt.

Und - nein, es gibt keinen R^2-Wert für nicht-lineare Regressionsmodelle! Dieser ist allgemein nicht definiert, so wurde uns berichtet. So ist der R^2-Wert einer linearen Regression ja ein Maß für die Verbesserung gegenüber dem Nullmodell, in dem nur der Achsenabschnitt gefittet wird. Bei nicht-linearen Modellen aber gibt es kein Standardnullmodell, mit dem man es vergleichen kann. Zudem muss das Nullmodell ja auch im Regressionsmodell genestet sein: was soll das in der nicht-linearen Regression sin? In der R-*help mailing list* kann man lange Email-Wechsel darüber lesen, wieso ein R^2-Wert kein sinnvolles Maß ist, was man stattdessen benutzen soll, und wie man mit Herausgebern umgehen soll, die einen R^2-Wert fordern. Diese Lektüre sei empfohlen, aber nicht hier abgedruckt.

```
> lines(seq(0, 10, by = 0.1), predict(fm, newdata = data.frame(c.e = seq(0,
+     10, by = 0.1))))
```

Als nächstes wollen wir eine Regression für zwei unterschiedliche Enzyme rechnen, also einen Faktor mit in die Regression hineinnehmen. Dazu simulieren wir zunächst einen Datensatz mit zwei leicht unterschiedlichen Enzymkinetiken, die wir dann in eine Tabelle (`data.frame`) umformen.

```
> noise <- rnorm(50, 0.1, 0.1)
> c.e <- seq(0.2, 10, 0.2)
> Km1 <- 1
> vmax1 <- 2
> v.e1 <- (vmax1 * c.e)/(Km1 + c.e) + noise
> Km2 <- 1.2
> vmax2 <- 2.2
> v.e2 <- (vmax2 * c.e)/(Km2 + c.e) + noise
> MM2 <- data.frame(c.e = c(c.e, c.e), stack(data.frame(v.e1, v.e2)))
> colnames(MM2) <- c("c.e", "v", "enzym")
> MM2$enzym <- factor(MM2$enzym, labels = c("enz1", "enz2"))
```

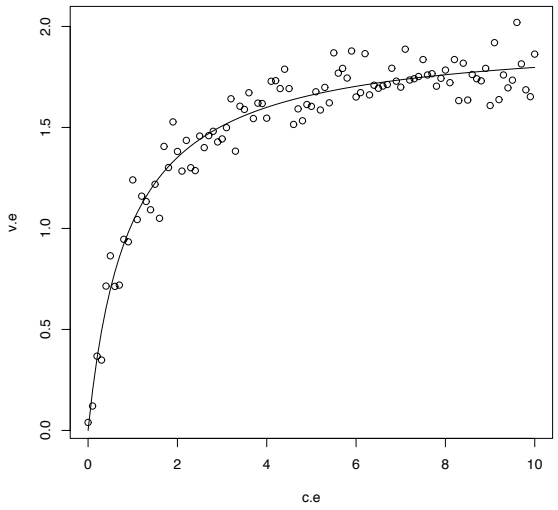


Abbildung 5.5.: Michaelis-Menton-Kinetik-Daten und nicht-linear gefittete Funktion.

Jetzt können wir aus dem *package* **nlme** die Funktion `nlsList` benutzen, um eine nicht-lineare Regression durchzuführen. Durch einen senkrechten Strich (|) wird diese Regression bedingt nach dem Faktor `enzym` berechnet.

```
> library(nlme)
> fm2 <- nlsList(v ~ (vmax * c.e)/(Km + c.e) | enzym, start = c(vmax = 1,
+     Km = 0.5), data = MM2)
> summary(fm2)

Call:
  Model: v ~ (vmax * c.e)/(Km + c.e) | enzym
   Data: MM2

Coefficients:
   vmax
     Estimate Std. Error  t value     Pr(>|t|)
enz1 2.109743 0.03498065 60.31173 6.828369e-47
enz2 2.303018 0.03828896 60.14834 7.124948e-47
   Km
      Estimate Std. Error  t value     Pr(>|t|)
enz1 0.9266136 0.07953195 11.65083 1.395194e-15
enz2 1.0998460 0.08629158 12.74569 4.921672e-17

Residual standard error: 0.1004273 on 96 degrees of freedom
```

Wir erhalten also nur die Koeffizienten der beiden Modelle und keinen Test auf ihre Unterschiedlichkeit. Der t-Test auf Unterschiedlichkeit funktioniert wie folgt (siehe Gleichung 4.4 für die t-Testformel). Zunächst speichern wir die Koeffizienten und ihre Standardfehler unter neuem Namen.

```
> coef.vmax <- coef(summary(fm2))[, , 1][1:2]
> coef.vmax.se <- coef(summary(fm2))[, , 1][3:4]
```

Dann wenden wir Formel 4.4 an, wobei natürlich $se = \frac{s}{\sqrt{(n)}}$, und somit $se^2 = \frac{s^2}{n}$.

```
> t.wert <- as.numeric(coef.vmax[1] - coef.vmax[2]/sqrt(sum(coef.vmax.se^2)))
> pt(t.wert, 48)

[1] 5.951791e-40
```

Wir erhalten also eine t-Wert von -37 (minus, da die Differenz der Mittelwerte negativ ist; das Vorzeichen ist beim t-Test aber irrelevant). Unsere Daten fußen auf 50 Datenpunkten, wir ziehen für die zwei Gruppen zwei Freiheitsgrade und erhalten df=48. Der Wahrscheinlichkeitswert wird mittels der pt-Funktion ermittelt und ist extrem klein. Also sind die Koeffizienten für `vmax` signifikant verschieden. Entsprechendes für `Km` sieht dann so aus:

```
> coef.Km <- coef(summary(fm2))[, , 2][1:2]
> coef.Km.se <- coef(summary(fm2))[, , 2][3:4]
> t.wert2 <- as.numeric(coef.Km[1] - coef.Km[2]/sqrt(sum(coef.Km.se^2)))
> pt(t.wert2, 48)

[1] 2.355718e-11
```

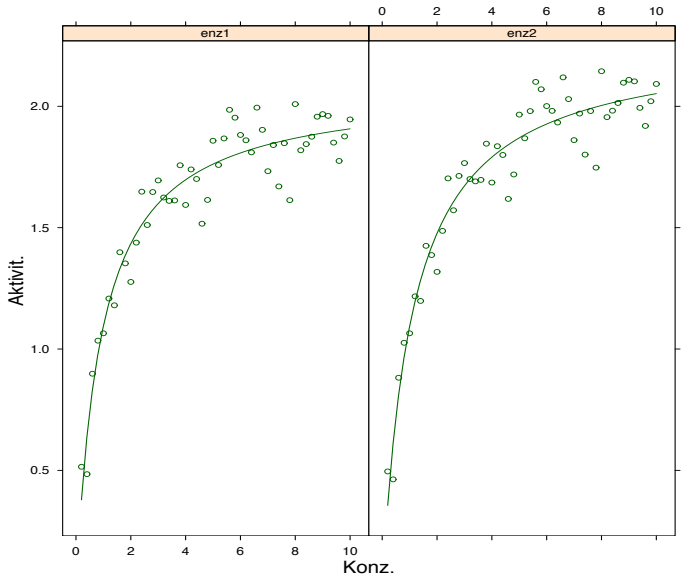


Abbildung 5.6.: Vergleich zweier Enzyme bezüglich ihrer Aktivität. Diese Abbildung ist mit dem `xyplot`-Befehl erzeugt worden.

Im folgenden wollen wir diese beiden Kurven einmal zusammen plotten (hier nicht abgebildet!).

```
> plot(v[51:100] ~ c.e[51:100], xlab = "Enzymkonzentration", ylab = "Aktivit.",
+     data = MM2)
> points(v[1:50] ~ c.e[1:50], pch = 16, data = MM2)
> xx <- seq(0, 10, by = 0.1)
> for (i in 1:2) {
+     pred <- coef(fm2)[i, 1] * xx/(coef(fm2)[i, 2] + xx)
+     lines(xx, pred, lty = i)
+ }
```

Im Zusatzpacket **lattice** sind graphische Zusatzfunktionen enthalten, die ein sehr komfortables bedingtes plotten erlauben. Damit kann man sowohl einfache xy-*plots* machen (siehe den ersten Befehl), als auch (über das *package* **nlme**) xy-*plots* mit vorhergesagter Regressionlinie (Abbildung 5.6).

```
> library(lattice)
> attach(MM2)
> trellis.par.set(background = list(col = "white"))
> xyplot(v ~ c.e | enzym, data = MM2)
> plot(augPred(fm2, primary = ~c.e), xlab = list("Konz.", cex = 1.5),
+     ylab = list("Aktivit.", cex = 1.5))
```

5.2.2. Häufig benutzte nicht-lineare Regressionen

Natürlich gibt es in jeder Disziplin spezielle nicht-lineare Funktionen, die häufig benutzt werden. In der Biologie ist dies z.B. die Michaelis-Menten-Kinetik aus dem letzten Abschnitt. Im folgenden wollen wir nur ein paar andere häufige nützliche Funktionen darstellen[9]. Die Spezifizierung und das Fitten in R ist vollkommen analog zur Michaelis-Menten-Kinetik: In der Formel steht die Antwortvariable links der Tilde, dann rechts davon eine Formel, die die erklärende(n) Variable(n) enthält[10]. Alle zu schätzenden Parameter müssen dann in der `start`-Option als Liste spezifiziert werden.

Hyperbel $y = y_0 + a/x$, mit y_0 als Asymptote.

Gaußsche Glockenkurve $y = ae^{-0.5(\frac{x-x_0}{b})^2}$, mit Mittelwert x_0 und Standardabweichung b.

Logistische Kurve I (sigmoidal) $y = y_0 + \frac{a}{(1+e^{\frac{-(x-x_0)}{b}})}$, mit *offset* y_0.

Logistische Kurve II (sigmoidal) $y = y_0 + \frac{a}{1+\left(\frac{x}{x_0}\right)^b}$, mit *offset* y_0.

Weibull-Funktion (sigmoidal) $y = y_0 + a\left(1 - e^{-\left(\frac{x-x_0-b\ln 2^{1/c}}{b}\right)^b}\right)$

Gompertz-Funktion (sigmoidal) $y = y_0 + ae^{-e^{-\left(\frac{x-x_0}{b}\right)}}$

Hill-Funktion (sigmoidal) $y = y_0 + \frac{ax^b}{c^b+x^b}$

Chapman-Funktion (sigmoidal) $y = y_0 + a\left(1 - e^{-bx}\right)^c$

exponentieller Abfall $y = y_0 + ae^{-bx}$

exponentiell asymptotisch aufsteigend $y = y_0 + a(1 - e^{-bx})$

asymptotisch aufsteigend $y = y_0 + a(1 - b^x)$

exponentielles Wachstum $y = y_0 + ae^{bx}$

Stirling-Funktion $y = y_0 + \frac{a(e^{bx}-1}{b}$

power Funktion $y = y_0 + ab^x$

aufsteigend-asymptotische Hyperbel(Michaelis-Menten-Kinetik) $y = y_0 + \frac{ax}{b+x}$

absteigend-asymptotische Hyperbel $y = y_0 + \frac{ab}{b+x}$

Sinus $y = y_0 + a\sin\left(\frac{2\pi x}{b} + c\right)$

[9] Diese Aufstellung bleibt zwangsläufig stark ausschnittshaft. Zu jeder Funktion gibt es Variationen. Unsere Auswahl folgt dem sehr guten *regression wizard* von SigmaPlot (StatSoft Inc.).

[10] Auch wenn wir hier nur univariate nicht-lineare Funktionen aufführen, so können wir doch auch höherdimensionale Funktionen mit `nls` fitten. Ein Beispiel ist die aufgeführte Lorentz-Funktion. Erst wenn wir selbstgeschriebene Funktionen parametrisieren wollen müssen wir auf eine allgemeine Optimierungsfunktion ausweichen. R bietet hierfür die Funktion `optim`. Hier wird nicht eine Formel spezifiziert, sondern eine Funktion übergeben, die minimiert/maximiert wird. Diese Funktion kann dann auch beispielsweise gekoppelte Differentialgleichungssysteme enthalten, die an beobachtete Daten gefittet werden sollen (Dormann and Roxburgh 2005, benutzten `optim` um sieben gekoppelte Lotka-Volterra-Konkurrenzgleichungen an ein aufwändiges Pflanzenkonkurrenzexperiment zu parametrisieren). Für Beispiele siehe die Hilfe zu `optim`.

gedämpfte Schwingung $y = y_0 + ae^{-\frac{x}{d}} \sin\left(\frac{2\pi x}{b} + c\right)$

Pareto-Funktion $y = 1 - \frac{1}{x^a}$

symmetrische V-Funktion $y = y_0 + a|x - x_0|^b$

asymptotisch-aufsteigende power-Funktion $y = a(1 - x^{-b})$

logarithmische Funktion $y = y_0 + a\ln(x - x_0)$

Lorentz-Funktion (3D) $z = \frac{a}{\left(1+(\frac{x-x_0}{b})^2\right)\left(1+(\frac{y-y_0}{c})^2\right)}$

5.2.3. Stückweise Regression

Wenn Daten nur über eine Teilbereich eine offensichtliche Korrelation beinhalten, kann man versuchen, Regressionen zu „stückeln“ (Toms and Lesperance 2003).

Nehmen wir ein Beispiel: Die Summe der Temperatur bis zur Keimung eines Samen nennt man *thermal time*. Der Gedanke ist, dass für eine Art diese *thermal time* konstant ist, d.h. je kälter es ist, desto später wird sie erreicht, und um so später setzt die Keimung ein. Bei vorliegenden Daten (aus Dormann et al. 2002) wurden die Bulbillen von *Polygonum viviparum* bei Temperaturen zwischen 2 und 25 °C bis zur Keimung feucht aufbewahrt. Die Temperatursummen wurden dann gegen die Aufbewahrungstemperatur aufgetragen (Abbildung 5.7).

```
> tt <- read.table("thermaltime.txt", header = T)
> attach(tt)
> plot(thermal.time ~ temperature, pch = 16, cex = 1.5)
```

Dies sieht nach einer linearen Beziehung bis etwa 19 Grad aus. Darüber ist kein Muster mehr erkennbar.

```
> fm <- lm(thermal.time ~ temperature)
> abline(fm)
```

Die Regressionslinie könnte besser passen. Lassen wir also erst einmal alle Punkte jenseits der 20 Grad weg:

```
> fm2 <- lm(thermal.time[temperature < 20] ~ temperature[temperature <
+     20])
> abline(fm2)
```

Jetzt passt die Regressionslinie zumindest für die ersten Punkte deutlich besser. Versuchen wir ein nicht-lineares, stückweises Regressionsmodel zu finden, dass die Daten am besten beschreibt:

```
> fm3 <- nls(thermal.time ~ a1 + a2 * temperature * (temperature <
+     20) + a3 * temperature * (temperature >= 20), start = list(a1 = 0,
+     a2 = 1.2, a3 = 1.4))
> summary(fm3)

Formula: thermal.time ~ a1 + a2 * temperature *
                (temperature < 20) + a3 * temperature * (temperature >= 20)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
a1  22.7966     4.8430   4.707 3.14e-05 ***
a2   5.5905     0.3810  14.675  < 2e-16 ***
a3   4.7290     0.2457  19.246  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.12 on 39 degrees of freedom
```

Dies ist ein Modell mit einem Schwellenwert bei 20°C. Definieren wir uns doch diese Formel als Funktion, dann können wir verschiedene Schwellenwerte einfach mittels `sapply` abfragen:

```
> nlsfun <- function(cc) {
+     summary(nls(thermal.time ~ a1 + a2 * temperature * (temperature <
+         cc) + a3 * temperature * (temperature >= cc), start = list(a1 = 0,
+         a2 = 1.2, a3 = 1.4)))$sigma
+ }
> sapply(15:25, nlsfun)

 [1] 12.167287 12.167287 11.938526 11.938526 11.671277 11.118475  9.689628
 [8]  9.689628 11.266358 11.946190 12.325808
```

Wir sehen, dass der Schwellenwert von 21°C die geringsten Abweichungsquadrate erzeugen.

```
> fm4 <- nls(thermal.time ~ a1 * (temperature < 21) + a2 * temperature *
+     (temperature < 21) + a4 * temperature * (temperature >= 21),
+     start = list(a1 = 0, a2 = 1.2, a4 = 2))
> summary(fm4)

Formula: thermal.time ~ a1 * (temperature < 21) + a2 * temperature * (temperature <
    21) + a4 * temperature * (temperature >= 21)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
a1  21.1887     4.1101   5.155 7.67e-06 ***
a2   5.7051     0.3015  18.926  < 2e-16 ***
a4   5.5343     0.1192  46.434  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.714 on 39 degrees of freedom

> summary(fm)

Call:
lm(formula = thermal.time ~ temperature)

Residuals:
     Min       1Q   Median       3Q      Max
-23.7314  -8.8249   0.6533   8.9039  22.4696

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  31.7230     4.6483   6.825 3.29e-08 ***
temperature   4.5620     0.2725  16.740  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.57 on 40 degrees of freedom
Multiple R-Squared: 0.8751,        Adjusted R-squared: 0.872
F-statistic: 280.2 on 1 and 40 DF,  p-value: < 2.2e-16
```

Verglichen mit dem ursprünglichen Modell (`fm`) sehen wir in den Residuen eine deutliche Verbesserung. Einen R^2 zu berechnen macht keinen Sinn für nicht-lineare Modelle! Schauen wir uns die erhaltenen Regressiongeraden einmal an:

```
> neu <- seq(2, 25, by = 0.1)
> regpred1 <- predict(fm4, newdata = data.frame(temperature = neu))
> plot(thermal.time ~ temperature, pch = 16, cex = 1.5)
> lines(neu[neu <= 20], regpred1[neu <= 20], lwd = 2)
> lines(neu[neu > 22], regpred1[neu > 22], lwd = 2, lty = 2)
```

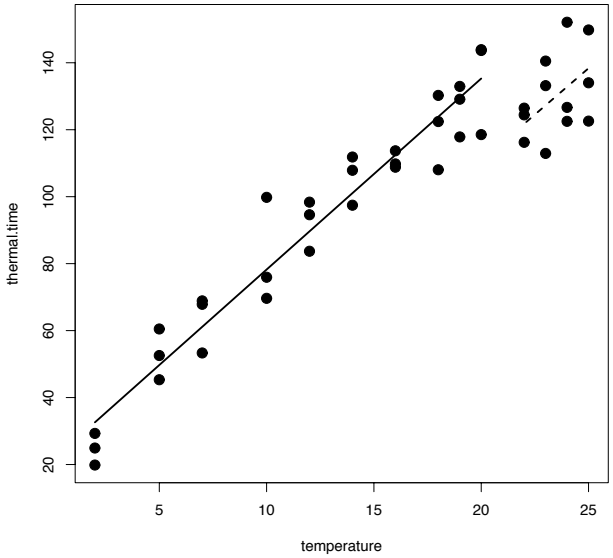


Abbildung 5.7.: Abhängigkeit der Keimungsrate von den Temperatursummen. Durchgezogene und gestrichelte Linie sind die Ergebnisse der stückweisen Regression.

Bezüglich der ursprünglichen These, dass die *thermal time* innerhalb einer Art konstant ist, bedeuten unsere Ergebnisse folgendes: Diese Art ist in ihrem Keimverhalten nahezu temperaturunabhängig, erst bei hohen Temperaturen (über 20 °C) finden wir ein Verhalten, dass einer konstanten *thermal time* entspricht.

5.3. Faktoren statt kontinuierliche erklärende Variablen: oneway-ANOVA

5.3.1. Einfaktorielle Varianzanalyse

Die Ergebnisse einer einfaktoriellen Varianzanalyse (*one-way ANOVA*: analysis of variance) und eines t-Tests sind identisch, *wenn die beiden Stichproben die gleiche Varianz haben.* Der allgemeine t-Test (Gleichung 4.4) erlaubt auch den Vergleich von Stichproben unterschiedlicher Varianz, wohingegen dies bei der ANOVA nicht möglich ist. Das hinter einer ANOVA stehende Konzept allerdings ist sehr viel mächtiger und generell sehr wichtig (Underwood 1997). Deshalb lohnt es sich hier kurz die Grundlagen am Beispiel einer einfaktoriellen ANOVA durchzugehen.

Der Grundgedanke ist, dass man die Varianz eines Datensatzes in zwei Gruppen aufteilen kann: diejenige, die mit den Behandlungseffekten assoziiert ist (s^2_{Effect}), und die sogenannte residuelle Varianz, die also nach Abzug der erklärten übrig bleibt (s^2_{Fehler} oder s^2_{resid} genannt). Das Verhältnis dieser Varianzen kann als F-Test genutzt werden. Im Detail:

Wir definieren die **Summe der Abweichungsquadrate** (SS: *sum of squares*) als

$$SS_{\text{total}} = \sum (y - \bar{\bar{y}})^2$$

wobei $\bar{\bar{y}}$ = der Gesamtmittelwert über alle y ist. Für die zwei Behandlungsgruppen A und B können wir einen Gruppenmittelwert berechnen: $\bar{y}_A$ und $\bar{y}_B$. Damit ist die Summe der Abweichungsquadrate der Gruppen, SS_A und SS_B, definiert als: $SS_A = \sum (y - \bar{y}_A)^2$ und $SS_B = \sum (y - \bar{y}_B)^2$. Die Summe dieser beiden Summen ist die verbleibende Variabilität innerhalb der Gruppen, genannt Abweichungsquadrate des Fehlers, oder *error sum of squares*:

$$SS_{\text{Fehler}} = SS_A + SS_B = \sum (y - \bar{y}_A)^2 + \sum (y - \bar{y}_B)^2$$

(Beachte, dass für den Fall dass es keinen Unterschied zwischen A und B gibt, $\bar{y}_A = \bar{y}_B = \bar{\bar{y}}$, und somit $SS_{\text{Fehler}} = SS_{\text{total}}$.)

Die Differenz zwischen SS_{gesamt} und SS_{Fehler} kann man entsprechend der Behandlung zuschreiben (*treatments sum of squares*):

$$SS_{\text{Behandl}} = SS_{\text{gesamt}} - SS_{\text{Fehler}}$$

In anderen Worten: SS_{Behandl} ist die Summe der Differenz zwischen dem Gesamtmittelwert $\bar{\bar{y}}$ und den geschätzten Gruppenmitteln $\hat{y}$ (d.i. $\hat{y}_A$ und $\hat{y}_B$, respektive):

$$SS_{\text{Behandl}} = \sum(\hat{y}_A - \bar{\bar{y}})^2 + \sum(\hat{y}_B - \bar{\bar{y}})^2$$

Analog zum Vergleich zweier Varianzen (siehe Abschnitt 4.1.3, Seite 48) kann das F-Verhältnis wie folgt berechnet werden[11]:

Quelle	SS	d.f.	MS	F
Behandlung	SS_{Behandl}	$k-1$	$\frac{SS_{\text{Behandl}}}{k-1}$	$\frac{MS_{\text{Behandl}}}{MS_{\text{Fehler}}}$
Fehler	SS_{Fehler}	$k(n-1)$	$s^2 = \frac{SS_{\text{Fehler}}}{k(n-1)}$	
Gesamt	SS_{gesamt}	$kn-1$		

k verschiedene Behandlungen mit einem Gesamtprobenumfang von n

Wenn die Genauigkeit der Messung zunimmt, so dass $(y - \bar{y}_A)^2$ abnimmt, so nimmt auch SS_{Fehler} ab. Perfekte Replikation innerhalb einer Gruppe bedeutet das SS_{Fehler} gegen 0 geht, weshalb das F-Verhältnis stärker wächst. Die kritischen F-Werte können wiederum in Tabellen nachgeschlagen werden.

Ein Mass für die Erklärungskraft des Modells ist der sogenannte R^2-Wert. Er berechnet sich als Verhältnis von $SS_{\text{Behandl.}}$ und SS_{gesamt}. Somit gibt er an, wieviel der Varianz des gesamten Datensatzes durch die Behandlung erklärt wird. Diese erklärte Varianz wird häufig in % angegeben (also $R^2 \cdot 100\%$).

Wenn die Daten nicht varianzhomogen sind, so ist eine wichtige Voraussetzung der ANOVA verletzt. Der **Kruskal-Wallis-Test** bietet hier einen Ausweg. Dafür werden die Daten Rang-transformiert und danach mit einer ANOVA analysiert (Underwood 1997). Wenn die Daten zu häufig den gleichen Wert haben (sog. „*ties*“) kann auch dieser Test nicht angewandt werden, da dann ebenfalls die Gefahr besteht, dass die Varianzen heterogen sind.

Eine Analyse wie oben kann auf verschiedene Art und Weise implementiert werden. Am schlüssigsten sich hier anschließend ist die Funktion `aov` (**a***nalysis* **o***f* **v***ariance*). Beispieldaten sind enthalten in `aov.bsp`. Im folgenden Beispiel wird zunächst eine Tabelle mit normalverteilen Werten generiert, die dann im weiteren analysiert wird. Da wir vorgeben, dass die Daten normalverteilt sind, können wir uns den Test auf Normalität schenken. Ebenso haben beide Variablen die gleiche Standardabweichung, unterscheiden sich also nur im Mittelwert.

```
> set.seed(1)
> aov.bsp <- matrix(ncol = 2, nrow = 40)
> aov.bsp[1:20, 2] <- rnorm(20, 10, 4)
> aov.bsp[21:40, 2] <- rnorm(20, 13, 4)
> aov.bsp[1:20, 1] <- 1
> aov.bsp[21:40, 1] <- 2
> colnames(aov.bsp) <- c("x", "y")
> aov.bsp <- as.data.frame(aov.bsp)
> fm1 <- aov(y ~ x, data = aov.bsp)
> summary(fm1)
```

[11] siehe etwa (Crawley 2002) oder `http://mathworld.wolfram.com/ANOVA.html`

```
            Df Sum Sq Mean Sq F value  Pr(>F)
x            1  48.93   48.93  3.8387 0.05745 .
Residuals   38 484.37   12.75
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Im Vergleich dazu der Ansatz des linearen Modells und sein output:

```
> fm2 <- lm(y ~ x, data = aov.bsp)
> summary(fm2)

Call:
lm(formula = y ~ x, data = aov.bsp)

Residuals:
    Min      1Q  Median      3Q     Max
-9.6209 -1.6972  0.1483  2.5096  5.6190

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    8.550      1.785   4.790 2.56e-05 ***
x              2.212      1.129   1.959   0.0574 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.57 on 38 degrees of freedom
Multiple R-Squared: 0.09175,        Adjusted R-squared: 0.06785
F-statistic: 3.839 on 1 and 38 DF,  p-value: 0.05745
```

Wir sehen, dass das linear Modell noch einen Achsenabschnitt fitted, und zusätzlich zu den Ergebnissen der aov die Koeffizienten angibt. Diese sind bei aov mittels des Befehls `fm1$coef` abrufbar. Hingegen bietet der `summary(fm2)`-*output* keine Werte für die SS an. Diese sind mittels des Befehls `anova(fm2)` verfügbar. (Der *output* ist dann identisch mit der Anwendung der summary auf die aov-Funktion.) Insgesamt bietet also das linear Modell mehr Angaben als die ANOVA-Variante (Koeffizienten und ihre Fehler, R^2-Werte, t-Test für Achsenabschnitt und Faktor), die aber nicht immer von Interesse sind.

5.3.2. Von der Regression zur ANOVA

Bei der Regression haben wir zwei kontinuierliche Variablen einander gegenübergestellt. Anstelle der kontinuierlichen erklärenden, könnten wir aber auch eine zweistufige, also kategorische, erklärende Variable haben, kurz: einen Faktor. Etwa könnten wir schlicht im Beispiel der *thermal time* in niedrige und hohe Temperaturen gruppieren, und diese beide einander gegenüberstellen. Wenn wir dies tun, wird aus der Regression ein faktorielles lineares Modell. Dieses ist deckungsgleich mit einer *oneway*-ANOVA. Dabei teilen wir die Varianz im Datensatz in die zwischen den zwei Gruppen (hohe, bzw. niedrige Temperatur) und die innerhalb der Gruppen. Deren Verhältnis gibt uns Aufschluss über den Einfluss des Faktors 5.3.1.

Damit haben wir gelernt, dass Regression und ANOVA eigentlich das gleiche lineare Modell zur Berechnung nutzen, und der einzige Unterschied die Form der erklärenden Variablen ist. Wenn wir im folgenden mehrere erklärende Variablen in das Modell aufnehmen, so wird dies oft ein Misch aus kontinuierlichen und kategorischen Variablen sein. Deshalb ist es wichtig zu verstehen, dass die Behandlung beider prinzipiell gleich ist.

Am einfachsten kann man sich dies dadurch veranschaulichen, dass die Level eines Faktors (die Gruppen) bei Statistikprogrammen intern als 0 und 1 kodiert werden. Was sich dann anschließt, ist eine Regression durch die Punkte, wobei eben alle Werte der Antwortvariablen jeweils mit einem Wert 0 oder 1 der erklärenden Variablen korrespondieren.

In R gibt es für die Regression nur die Funktion `lm`. Für die ANOVA können wir hingegen auch die Funktion `aov` benutzen. Im folgenden sei kurz dargestellt, dass diese (bei orthogonalen Designs) equivalent und austauschbar sind. Wir laden die Keimungsdaten und verwandeln die Variable `temperature` in einen Faktor `temp`, mit 2 *leveln* ($<$ und $>$ 17°C). Der *plot* verändert sich entsprechend in einen *boxplot*.

```
> tt <- read.table("thermaltime.txt", header = T)
> attach(tt)
> temp <- as.factor(ifelse(temperature < 17, "LOW", "HIGH"))
> plot(thermal.time ~ temp)
> nm <- aov(thermal.time ~ temp)
> summary(nm)

            Df  Sum Sq Mean Sq F value    Pr(>F)
temp         1 29551.2 29551.2  56.099 3.934e-09 ***
Residuals   40 21070.8   526.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Im *output* erhalten wir die Angaben über die Summe der Abweichungsquadrate, die dem Faktor zukommt, dem F-Test auf diese *sum of squares*, sowie die Abweichungsquadrate der Residuen. Vergleichen wir dies mit einer linearen Regression:

```
> nm2 <- lm(thermal.time ~ temp)
> summary(nm2)

Call:
lm(formula = thermal.time ~ temp)

Residuals:
    Min      1Q  Median      3Q     Max
-56.058 -10.931  -1.131  17.762  37.809

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  128.960      5.008   25.75  < 2e-16 ***
tempLOW      -53.051      7.083   -7.49 3.93e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 22.95 on 40 degrees of freedom
Multiple R-Squared: 0.5838,        Adjusted R-squared: 0.5734
F-statistic:  56.1 on 1 and 40 DF,  p-value: 3.934e-09
```

Hier bekommen wir vor allem die Regressionskoeffizienten (mit Fehlern und t-Tests), und ganz unten auch die F-Statistik. Diese ist identisch mit der der Funktion `aov`. Beachte, dass der Faktor nicht als `temp` geführt wird, sondern als `tempLOW`. D.h., wenn wir den Wert für `LOW` haben wollen, müssen wir von dem *intercept* noch den angegeben Wert abziehen. Der *intercept* ist hingegen der Mittelwert von `HIGH`.
Auf jeden Fall können wir aus beiden Modellen die jeweils fehlenden Informationen abfragen. So erhalten wir mittels `anova(nm2)` eine Tabelle identisch der aus der `aov`. Umgekehrt können wir mit dem Befehl `coef(nm)` die Koeffizienten aus der `aov` abfragen.

5.4. Modelldiagnostik

Die oben beschriebenen Verfahren setzen drei Eigenschaften der Daten voraus.

1. Unabhängigkeit der einzelnen Datenpunkte. Dies bedeutet, dass die Tatsache, dass wir einen bestimmten Wert gemessen haben, keinen Einfluss darauf hat, welchen Wert wir als nächstes messen. *Data independence* ist eine ebenso einfache wie fundamentale Voraussetzung.

2. Varianzhomogenität (Homoskedatizität). Dieses Kriterium wird viel zu oft übersehen, und ist wegen seiner Nähe zum vorigen Punkt etwas mühselig zu verstehen. Nehmen wir an, die Messungen unserer Baumgrößen ergeben, dass die Art A im Mittel 50 m hoch ist, mit einer Standardabweichung (= Wurzel der Varianz) von $s_A = 10$. Art B ist im Mittel 40 m hoch, mit einer Standardabweichung von $s_B = 7$. Es ist leider bei vielen Daten der Fall, dass die Varianz mit dem Mittelwert zunimmt. Nach Korrektur für die Unterschiede in den Mittelwerten haben wir dann immer noch einen Unterschied in der Varianz der Höhe der zwei Baumarten. Wenn wir aber aus zwei Verteilungen mit unterschiedlichen Varianzen nun wenige Stichproben ziehen, so kann es leicht passieren, dass wir trotz gleichen Mittelwertes sehr unterschiedliche Werte erhalten. In den letzten Jahren wird in der ökologischen Literatur zunehmend auf die Wichtigkeit dieser Varianzhomogenität hingewiesen, während das Paradigma der Normalverteilung an Glanz verliert.

3. Normalverteilung der Daten. Diese Annahme ist sprachlich so sehr unexakt benannt. Nichtsdestotrotz wird diese Voraussetzung meistens so formuliert. Was eigentlich gemeint ist, ist dass das unerklärte Rauschen (*residual noise*) auf dem Datensatz normalverteilt ist[12]. Wenn wir also beispielsweise die Größe zweier Baumarten vergleichen, so entstammen natürlich nicht alle diese Messungen einer normalverteilten Grundgesamtheit, sondern zweier (einer je Baumart). Wenn wir aber bei unserem linearen Modell für die Unterschiede zwischen den Baumarten korrigiert haben (d.h. die jeweiligen Mittelwert von allen Messwerten abgezogen haben), dann sollten die übrig gebliebenen Abweichungen (Residuen) aus einer Grundgesamtheit sein. Entsprechend führen wir eine Untersuchung auf Normalverteilung erst *nachträglich* durch.

Einer Analyse (etwa Regression oder ANOVA) schließen wir also die Untersuchung dieser Testannahmen an. Die Unabhängigkeit der Datenpunkte ergibt sich aus dem Design ihrer Erfassung und ist den Daten nicht anzusehen (siehe Abschnitt 13). Wenden wir uns also Punkt 2 zu.

Zur **Varianzhomogenität**: Nach Durchführung einer Regression können wir leicht die Residuen berechnen, indem wir von jedem beobachteten Wert den vorhergesagten abziehen. Dies wird uns meist von der Software abgenommen, aber zur Illustration sei es hier einmal durchgeführt. Für einen Datenpunkt $x_1 = (2, 5)$ und eine Regressionsgerade $y = 2.5x - 1$ können wir eine vorhergesagten Wert für x_1 von $y_1 = 2.5 \cdot 2 - 1 = 4$ berechnen. Die Abweichung beträgt also 5−4=1. Dies machen wir nun für alle Datenpunkte, und tragen dann die Residuen (auf der y-Achse) gegen die vorhergesagten Werte (auf der x-Achse) auf. Wenn jetzt alle Punkte ohne erkennbares Muster in der Ebene verteilt sind, sind die Varianzen homogen.

Bei den Residuen einer ANOVA sieht dies natürlich anders aus, da wir dort ja nur wenige „vorhergesagte Werte“ haben, nämlich die Level des erklärenden Faktors. Entsprechend verteilen sich die Residuen auch nicht über die ganze Ebene, sondern nur innerhalb der Faktorlevel. Dort sollte aber ebenfalls kein Muster erkennbar sein.

Ein formeller Test auf Varianzhomogenität ist der F-Test und ist auf Seite 48 beschrieben. Für ANOVAs kommen dann noch Cochrans, Bartletts oder Levenes Test in Frage. Der Grund weshalb die meisten Anwender auf diese formellen Tests verzichten (siehe etwa Crawley 2002; Dalgaard 2002; Quinn and Keough 2002) ist, dass diese Tests sehr empfindlich von der Anzahl

[12] Auch das ist nicht ganz korrekt, aber einfacher zu merken. Die wirkliche Annahme ist, dass alle beobachteten Werte aus einer Normalverteilung entstammen, deren Mittelwert durch das Regressionsmodell bestimmt wird. Der Erwartungswert der Regression für einen Datenpunkt ist also der Mittelwert der Normalverteilung, aus der der beobachtete Wert eine zufällige Realisation darstellt. Oder, mathematischer: y ist eine Zufallsvariable mit Erwartungswert $E(y_i) = \beta_0 + \beta_1 x_i$. Entsprechend wird auch die Varianzhomogenitätsannahme verständlicher: Während der Mittelwert für jeden Datenpunkt aus der Regression berechnet wird, ist die Standardabweichung der Normalverteilung über alle Punkte hinweg konstant, sonst wäre das Modell nicht eindeutig definiert, da ja dann sowohl Mittelwert als auch Standardabweichung für jeden Punkt geschätzt würden. Alles klar?

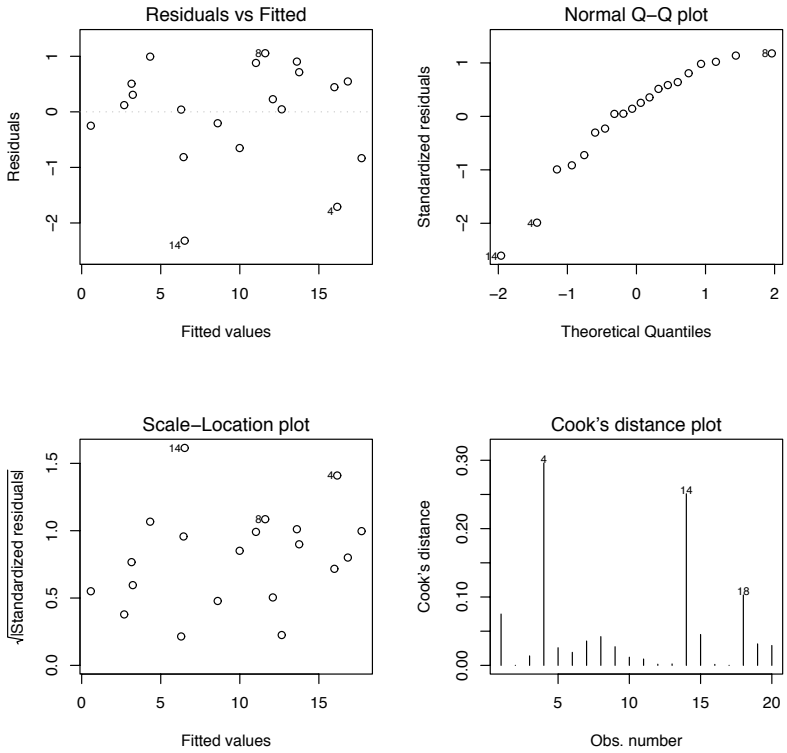


Abbildung 5.8.: Diagnostische Abbildungen in R nach einer Regression. Grafik 3 ist ähnlich Grafik 1, aber aus die Residuen wurden standardisiert und aus ihnen die Wurzel gezogen, um einen Trend in der Streuung leichter erkennbar zu machen (Dalgaard 2002). Grafik 4 gibt an, welche Datenpunkte einen besonders starken Einfluss auf das Regressionsmodell haben.

der Datenpunkte abhängen. Bei großen Stichprobenumfang kann deshalb immer ein Varianzunterschied festgestellt werden, obwohl er praktisch irrelevant sein mag. Umgekehrt können bei geringem Stichprobenumfang auch gravierende Varianzunterschiede nicht als solche erkannt werden. Deshalb argumentieren obige Autoren für eine grafische Darstellung und rein visuelle Entscheidung (etwa Boxplots innerhalb jeder Gruppe).

Die **Normalverteilung** der Residuen lässt sich ebenfalls gut grafisch evaluieren. Dafür lassen wir den Computer einen Quantilen-Quantilen-plot (*qq-plot* oder *normal quantile plot*) anfertigen, in dem die standardisierten Residuen aufgetragen werden gegen den Wert, der aus einer Normalverteilung zu erwarten wäre. Die tatsächliche Berechnung des Erwartungswertes ist etwas umständlich (siehe etwa Sokal and Rohlf 1995, S. 116ff), deshalb überlassen wir dies dem Rechner. In dieser Grafik sollten alle Punkte entlang der Winkelhalbierenden liegen. Abweichungen davon deuten auf eine Abweichung von einer Normalverteilung hin.[13]

Alternativ können wir den Kolmogorov-Smirnov-Test auf Normalverteilung durchführen (siehe Seite 10). Dieser Test ist sozusagen die Formalisierung der grafischen Methode und somit recht zuverlässig zu gebrauchen (d.h. es gelten nicht die Einwände die unter Varianzhomogenität gegen formelle Test gebracht wurden).

Günstigerweise sind die beiden beschriebenen *plots* in R standardmäßig nach der Berechnung eines linearen Modells abrufbar. Insgesamt werden mit dem Befehl `plot(lineares.Modell)` vier Grafiken produziert[14].

[13] Wenngleich selten von Interesse kann man aus der Form der Abweichung Rückschlüsse auf die zugrundeliegende Verteilung ziehen. Siehe dazu ebenfalls (Sokal and Rohlf 1995, S. 116ff).

[14] Insbesondere der vierte Plot unterscheidet sich von R-Version zu R-Version. Wahrscheinlich ist er inzwischen schon wieder anders.

Als Beispiel generieren wir einfach eine Regression und führen dann die diagnostischen *plots* durch:

```
> set.seed(1)
> x <- runif(20, 0, 10)
> y <- 2 * x - 1.5 + rnorm(20, 0, 1)
> plot(y ~ x)
> fm <- lm(y ~ x)
> par(mfrow = c(2, 2))
> plot(fm)
```

Das Ergebnis ist in Abbildung 5.8 dargestellt. Wir sehen, dass die Punkte 4 und 14 einen besonders starken Einfluss haben. Die absoluten Werte für *Cook's distance* sind allerdings recht gering, eine Veränderung der Regressionsparameter ohne diese Punkte deshalb wahrscheinlich gering. Prüfen wir das nach, indem wir uns die geschätzten Koeffizienten mit und ohne diese Punkte ansehen:

```
> fm2 <- lm(y[c(-4, -14)] ~ x[c(-4, -14)])
> summary(fm)$coef[1:2, 1:2]

             Estimate Std. Error
(Intercept) -0.5587681 0.45934033
x            1.8408324 0.07393528

> summary(fm2)$coef[1:2, 1:2]

                Estimate Std. Error
(Intercept)   -0.4314135  0.3183410
x[c(-4, -14)]  1.8585389  0.0519955
```

In der Tat unterscheiden sich die beiden Regressionen nur geringfügig in ihren Parametern. Die Steigung ist nahezu identisch, während der (nicht signifikant von 0 verschiedene) Achsenabschnitt etwas stärker variiert. Hieraus schließen wir, dass die beiden Punkte 4 und 14 die Regression nicht substantiell verzerren.

6. Das Allgemeine Lineare Modell: Mehrere Erklärende Variablen

Vom einfachen zum allgemeinen linearen Modell ist es nur ein kleiner Schritt: Statt einer erklärenden Variablen gibt es derer mehrere. Die allgemeine Form (Formel 4.10) sagt einfach, dass eine Antwortvariable Y durch mehrere Variablen $X_i = X_1, X_2, \ldots, X_n$ erklärt werden soll. Übrig bleibt ein Fehlerterm (ϵ), der durch die Variablen X nicht erklärbar ist. Bis auf weiteres nehmen wir an, dass der Fehlerterm ϵ normalverteilt ist. Erst im Verallgemeinerten Linearen Modell lassen wir diese Annahme fallen.

Sind alle Variablen X kontinuierlich, so sprechen wir von multipler Regression. Sind die Variablen X allesamt kategoriale Faktoren, so sprechen wir von einer multifaktoriellen ANOVA. Kommen sowohl kontinuierliche als auch kategoriale Variablen vor, so haben wir es mit einer ANCOVA (*analysis of co-variance*) zu tun. Wie aus diesen Zeilen deutlich wird, unterscheiden sich diese Methoden eigentlich nur geringfügig voneinander, nämlich in der Struktur der erklärenden Variablen. Mit einer moderneren Definition könnten wir diese Fälle allesamt einfach unter das Thema Allgemeines Lineares Modell fassen. Um mit praktisch allen Übrigen angewandten Statistikbüchern kompatibel zu sein, schauen wir uns diese nur leicht unterschiedlichen Modelltypen aber unter ihrem konventionellen Namen an.[1]

Mit mehreren erklärenden Variablen tauchen eine Reihe von Komplikationen auf, die wir bisher noch nicht zu berücksichtigen hatten.

1. Korrelation von erklärenden Variablen (auch Kollinearität genannt[2]): Mit der Abnahme an Babies und der Abnahme an Störchen nahm auch der CO_2-Gehalt in der Atmosphäre zu. Wenn wir jetzt die Anzahl Neugeborener mit Storchzahlen und CO_2 erklären wollen, so wird nur einer der Faktoren als wichtig erkannt (und zwar aus rechentechnischen Gründen zumeist der erste), da für den zweiten kaum noch Varianz zur Erklärung übrig bleibt.

2. Interaktionen: So nennt man das Phänomen, dass ein Faktor nur unter bestimmten Bedingungen (nämlich in Abhängigkeit des Niveaus eines anderen Faktors) einen bestimmten Effekt hat. Wenn wir also beispielsweise Pferde und/oder Kühe einen Probefläche beweiden lassen, dann finden wir u.U. nur dann einen Effekt der Kühe auf Pflanzengemeinschaften, wenn *keine* Pferde mit auf der Weide stehen.

3. Nicht-lineare Zusammenhänge mit manchen Variablen: Eine erklärende Variable muss ja nicht streng linear mit einer Antwortvariablen verbunden sein. Meist kann man einfache Nicht-Linearität durch ein Polynom annähern. Dazu füttert man nicht nur eine erklärende Variable, sondern auch ihre zweite (und dritte) Potenz mit in das Modell.

[1] Die linearen Modelle sind noch nicht im vollen Umfang ins Standardrepertoire von Biologen gelangt. Dies liegt u.a. an einer gewissen Mystik, die das lineare Modell ungerechtfertigterweise umgibt. An dieser Stelle sei jedem mit guten Englischkenntnissen und Sinn für Humor ein Pamphlet eines Guru moderner Statistik ans Herz gelegt (Venables 2000). Es behandelt die Probleme, Missverständnisse und Software-Limitationen im Bereich linearer Modelle.

[2] Gelegentlich hört man jemanden hier von "Autokorrelation" reden. Das ist falsch. Wie das Präfix "auto" (griech. für "selbst") andeutet, geht es bei Autokorrelation um die Korrelation einer Variablen mit sich selbst. So sind etwa frühe Messwerte in einer Zeitreihe oder im Raum enger mit ihren zeitlichen/räumlichen Nachbarn korreliert, als mit späteren/entfernteren Datenpunkten: zeitliche und/oder räumliche Autokorrelation. Richtige Synonyme für Kollinearität sind hingegen die Ausdrücke "Interkorrelation" oder "multiple Korrelation".

4. Modellbildung: Wenn wir von jeder erklärenden Variablen noch die zweite Potenz ins Modell nehmen, und dann alle möglichen Interaktionen zwischen den erklärenden Variablen (und ihren 2. Potenzen!), haben wir schnell ein Modell, dass mehr Antwortvariablen als Datenpunkte hat. Leider gibt es keinen goldenen Weg, welche Interaktionen/Potenzen man in welcher Reihenfolge ins Modell nimmt.

5. Modellvereinfachung: Der Umgang mit nicht signifikanten Variablen in einem Modell. Wenn unser lineares Modell vier Faktoren enthält, von denen einer keine Einfluss auf die Antwortvariable zu haben scheint, dann können wir diesen (unter bestimmten Bedingungen) aus dem Modell herausnehmen. Die Prozedur fällt unter das Stichwort *Modellvereinfachung* und hat sich inzwischen als Standardprozedur durchgesetzt. Über die Vorteile dieser Vereinfachung sprechen wir dann an gegebener Stelle.

6.1. Mehrere, kontinuierliche erklärende Variablen: Multiple Regression

Wie so häufig, lernt man am besten am Beispiel. Dieses ist dem Buch von Quinn and Keough (2002) entlehnt, und beruht auf einem Datensatz publiziert von Paruelo and Lauenroth (1996). Dabei geht es um C_3-Pflanzen in 73 Lokationen in Nord-Amerika. Wir wollen versuchen, deren Häufigkeit mittels 6 kontinuierlicher Variablen zu erklären: Längen- und Breitengrad (`LONG` und `LAT`), Jahresdurchschnittsniederschlag und -temperatur (*mean annual precipitation/temperature*, `MAP` und `MAT`) und dem Sommer- bzw. Winterniederschlagsanteil (*June/July/August proportion of MAP*: `JJAMAP` und *December/January/February proportion of MAP*: `DJFMAP`).

Zuerst laden wir den Datensatz, schauen uns die Verteilung der Antwortvariablen an, und transformieren sie in Richtung Normalverteilung. Aufgrund relativ vieler Nullen wird dies nicht sehr gut, aber für unsere Analyse soll es reichen.

```
> paruelo <- read.table("paruelo.txt", header = T)
> paruelo$C3 <- log10(paruelo$C3 + 0.1)
> attach(paruelo)
```

Jetzt schauen wir uns mit einem einfachen Befehl die Korrelation aller Variablen an.

```
> pairs(paruelo)
```

Wir sehen (Abbildung 6.1), dass C3 vor allem mit `LAT` korreliert, während die erklärenden Variablen untereinander ebenfalls korreliert zu sein scheinen (etwa `MAP` mit `LONG`). Schauen wir uns also einmal die Korrelationsmatrix an:

```
> round(cor(paruelo[, -1]), 2)
```

```
          MAP   MAT JJAMAP DJFMAP  LONG   LAT
MAP      1.00  0.36   0.11  -0.40 -0.73 -0.25
MAT      0.36  1.00  -0.08   0.00 -0.21 -0.84
JJAMAP   0.11 -0.08   1.00  -0.79 -0.49  0.07
DJFMAP -0.40  0.00  -0.79   1.00  0.77 -0.07
LONG    -0.73 -0.21  -0.49   0.77  1.00  0.10
LAT     -0.25 -0.84   0.07  -0.07  0.10  1.00
```

Es fallen vier Korrelationen auf: `MAP/LONG`, `MAT/LAT`, `JJAMAP/DJFMAP`, `DJFMAP/LONG`. Das damit auftretende Problem multipler Autokorrelation nennt man Kollinearität (*collinearity*).
Schauen wir uns erst einmal die geographischen Variablen an, und regressieren diese (und ihre Interaktion) gegen die Häufigkeit der C_3-Pflanzen:

```
> summary(lm(C3 ~ LAT * LONG))
```

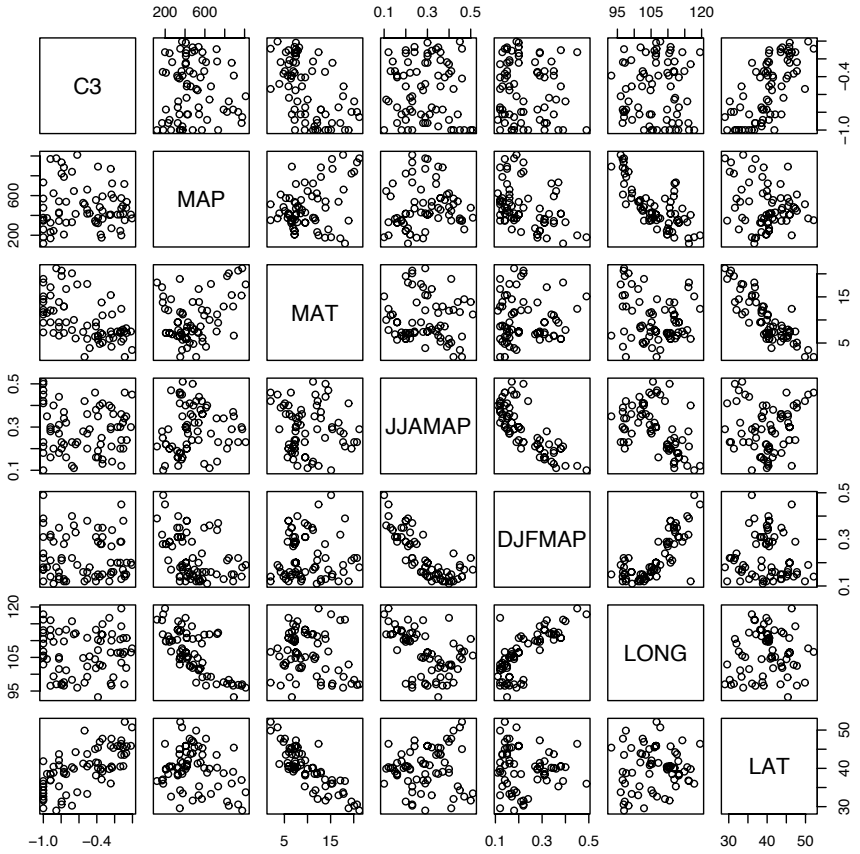


Abbildung 6.1.: R's `pairs`-*plot*, in dem jede Variable gegen jede andere aufgetragen wird.

```
Call:
lm(formula = C3 ~ LAT * LONG)

Residuals:
     Min       1Q   Median       3Q      Max
-0.54185 -0.13298 -0.02287  0.16807  0.43410

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  7.3909798  3.6253486   2.039  0.04531 *
LAT         -0.1912401  0.0910018  -2.101  0.03925 *
LONG        -0.0929020  0.0349331  -2.659  0.00972 **
LAT:LONG     0.0022522  0.0008757   2.572  0.01227 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2334 on 69 degrees of freedom
Multiple R-Squared: 0.5137,        Adjusted R-squared: 0.4926
F-statistic:  24.3 on 3 and 69 DF,  p-value: 7.657e-11
```

Zu unserem Erstaunen ist die Steigung von `LAT` *negativ*, obwohl wir im `pairs`-plot doch eine positive Korrelation bemerkt haben! Wie kann das sein?
Der Grund ist einfach: Die Interaktion von `LAT` und `LONG` wird (zufälligerweise) von `LAT` dominiert. Wir können dies sehen, wenn wir `LAT*LONG` gegen `LAT` auftragen:

```
> LL <- LAT * LONG
> plot(LL ~ LAT)
```

Dies ist nicht der Fall für LONG:

```
> plot(LL ~ LONG)
```

Die positive Korrelation mit der Interaktion erklärt also die meiste Streuung der C_3-Daten, so dass die Residuen nur schwach (und negativ) mit LAT korrelieren. Diesen Effekt der Kollinearität kann man durch die Berechnung der sogenannten Toleranz quantifiziern. Im vorliegenden Fall berechnet man dazu den Korrelationskoeffizienten R^2 für die Regression von C3 gegen LONG+LAT:LONG. Die Toleranz für LAT berechnet sich dann als $1 - R^2$. Allgemein wird die Toleranz also durch die Regression aller übrigen erklärenden Variablen gegen die jeweils betrachtete berechnet. Je näher an 1, desto unproblematischer ist eine Regression. Berechnen wir also einmal die Toleranz für LAT:

```
> 1 - summary(lm(LAT ~ LONG + LAT:LONG))$r.squared

[1] 0.003249445
```

Und entsprechend für LONG und die Interaktion:

```
> 1 - summary(lm(LONG ~ LAT + LAT:LONG))$r.squared

[1] 0.01497357

> 1 - summary(lm(LAT * LONG ~ LONG + LAT))$r.squared

[1] 0.002494144
```

Alle drei sind also nahe Null, was problematisch ist. Abhilfe schafft bisweilen ein Zentrieren der Daten:

```
> CLAT <- LAT - mean(LAT)
> CLONG <- LONG - mean(LONG)
> summary(lm(C3 ~ CLAT * CLONG))

Call:
lm(formula = C3 ~ CLAT * CLONG)

Residuals:
     Min       1Q   Median       3Q      Max
-0.54185 -0.13298 -0.02287  0.16807  0.43410

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.5529416  0.0274679 -20.130  < 2e-16 ***
CLAT         0.0483954  0.0057047   8.483 2.61e-12 ***
CLONG       -0.0025787  0.0043182  -0.597   0.5523
CLAT:CLONG   0.0022522  0.0008757   2.572   0.0123 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2334 on 69 degrees of freedom
Multiple R-Squared: 0.5137,        Adjusted R-squared: 0.4926
F-statistic:  24.3 on 3 and 69 DF,  p-value: 7.657e-11

> 1 - summary(lm(CLAT ~ CLONG + CLAT:CLONG))$r.squared

[1] 0.8268942
```

```
> 1 - summary(lm(CLONG ~ CLAT + CLAT:CLONG))$r.squared

[1] 0.9799097

> 1 - summary(lm(CLAT * CLONG ~ CLONG + CLAT))$r.squared

[1] 0.8195915

> 1 - summary(lm(CLAT * CLONG ~ CLONG + CLAT))$r.squared

[1] 0.8195915
```

Offensichtlich haben wir damit das Kollinearitätsproblem gelöst.
Jetzt können wir uns möglichen Nicht-Linearitäten zuwenden (obwohl der `pairs`-*plot* nicht darauf hindeutete). Dafür nehmen wir die 2. Potenz der Variablen mit ins Modell auf (kodiert als `I(CLAT^2)`). Außerdem wenden wir uns jetzt von den Koeffizienten (Steigungen) und schauen mehr auf die Signifikanz des Effekts im Modell. Dafür benötigen wir statt des `summary`-Befehls `anova`.
Hier vielleicht ein paar Worte zum Unterschied zwischen `anova`- und `summary`-Signifikanzen. Erstere geben an, ob ein Effekt signifikant zur Erklärung der Gesamtvarianz beiträgt. Zweitere ist "nur" ein Test, ob die Steigung signifikant unterschiedlich von 0 ist. Nun können wir uns Fälle vorstellen, in denen eine Steigung sehr gut bestimmbar ist (d.h. wenig Streuung aufweist), so dass selbst eine schwache Steigung signifikant ist. Diese Variable wird aber nicht notwendigerweise wichtig sein, da die geringe Steigung auf einen geringen funktionalen Effekt hinweist. Deshalb ist es wichtig, dass wir nicht die t-Test-Signifikanzen als Maß für die Wichtigkeit eines Effekts nehmen, sondern seinen Erklärungsanteil in der ANOVA. Später, im GLM, ersetzt die *deviance* die Abweichungsquadrate und der X^2-Test den F-Test; alles andere bleibt zutreffend!

```
> fmm2 <- lm(C3 ~ CLAT * CLONG + I(CLAT^2) * CLONG + CLAT * I(CLONG^2) +
+     I(CLAT^2) * I(CLONG^2))
> anova(fmm2)

Analysis of Variance Table

Response: C3
                     Df    Sum Sq   Mean Sq F value    Pr(>F)
CLAT                  1    3.5703    3.5703 61.4510 6.196e-11 ***
CLONG                 1    0.0412    0.0412  0.7099   0.40261
I(CLAT^2)             1    0.0073    0.0073  0.1250   0.72489
I(CLONG^2)            1 3.415e-05 3.415e-05  0.0006   0.98073
CLAT:CLONG            1    0.3601    0.3601  6.1975   0.01540 *
CLONG:I(CLAT^2)       1    0.0249    0.0249  0.4283   0.51516
CLAT:I(CLONG^2)       1    0.0034    0.0034  0.0593   0.80833
I(CLAT^2):I(CLONG^2)  1    0.0058    0.0058  0.1006   0.75211
Residuals            64    3.7184    0.0581
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Das maximale Modell, mit sämtlichen quadratischen Variablenkombinationen, ergibt keine zusätzlichen signifikanten Effekte, d.h. wir bleiben bei unserem ursprünglichen, linearen Modell mit Interaktion.
Damit können wir uns jetzt den übrigen Variablen zuwenden. Auch hier führen wir erst einmal alle Variablen ins Modell ein.

```
> anova(fmm3 <- lm(C3 ~ MAP * MAT * DJFMAP * JJAMAP))

Analysis of Variance Table

Response: C3
                    Df Sum Sq Mean Sq F value   Pr(>F)
MAP                  1 0.0017  0.0017  0.0250  0.87500
```

```
MAT                     1 2.5766  2.5766 37.9523 7.84e-08 ***
DJFMAP                  1 0.0003  0.0003  0.0051  0.94353
JJAMAP                  1 0.2951  0.2951  4.3461  0.04158 *
MAP:MAT                 1 0.0031  0.0031  0.0455  0.83191
MAP:DJFMAP              1 0.0178  0.0178  0.2626  0.61030
MAT:DJFMAP              1 0.1482  0.1482  2.1836  0.14499
MAP:JJAMAP              1 0.2063  0.2063  3.0385  0.08670 .
MAT:JJAMAP              1 0.1567  0.1567  2.3077  0.13426
DJFMAP:JJAMAP           1 0.0567  0.0567  0.8348  0.36475
MAP:MAT:DJFMAP          1 0.0161  0.0161  0.2373  0.62806
MAP:MAT:JJAMAP          1 0.2912  0.2912  4.2891  0.04290 *
MAP:DJFMAP:JJAMAP       1 0.0423  0.0423  0.6230  0.43320
MAT:DJFMAP:JJAMAP       1 0.0196  0.0196  0.2881  0.59350
MAP:MAT:DJFMAP:JJAMAP   1 0.0300  0.0300  0.4419  0.50889
Residuals              57 3.8698  0.0679
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

An dieser Stelle wir ein Schritt notwendig, der einen eigenen Abschnitt rechtfertig.

6.2. Modellvereinfachung

Die erklärenden Variablen können nicht nur linear, sondern auch nicht-linear mit der Antwortvariablen zusammenhängen. Durch die Hinzunahme der zweiten, dritten, usw. Ordnung des Effektes können wir diese Nicht-Linearität analysieren. Weiterhin können wir zusätzlich zu den Haupteffekten (*main effects*), d.h. den erklärenden Variablen, auch ihre Interaktionen mit ins Modell nehmen. Dies gilt analog für die höheren Ordnungen der erklärenden Variablen. Dies führt dazu, dass das Erklärungsmodell theoretisch unglaublich kompliziert werden kann. Schon bald gibt es dann mehr erklärende Faktoren als Datenpunkte, und eine Analyse ist nicht mehr möglich.

Das Problem besteht aus zwei Teilen: (1) Aufteilung der erklärenden Variablen und (2) Vereinfachung des Modells. Im obigen Beispiel haben wir (ohne dies weiter zu kommentieren) nur Breiten- und Längengrade regressiert, und die anderen erklärenden Variablen beiseite gelassen. Dies ist ein Fall von Variablenaufteilung[3], um das Modell handhabbar zu halten. Eine bessere Herangehensweise wird weiter unten vorgestellt (Abschnitt 6.3).

Inzwischen sind wir in dem Beispiel am Punkt angelangt, wo durch die Interaktionen der vier übrigen erklärenden Variablen 15 Effekte im Modell geschätzt werden müssen. Wie das Ergebnis zeigt, sind die meisten davon nicht signifikant. Es folgt also der nächste Schritt: Modellvereinfachung.

Um das Modell zu vereinfachen, löschen wir einfach einen Effekt (Haupt- oder Interaktionseffekt) aus dem Modell. Nur welchen? Hier gibt es einen wichtigen Punkte zu bedenken: **Wir können nur solche Effekte löschen, die nicht Teil einer Interaktion sind** (*Marginalitätstheorem*). D.h. wenn der Effekt `A` gelöscht werden soll, aber die Interaktion `A:B` signifikant ist, so dürfen wir auch `A` nicht ausschließen. Welchen Effekt wir im nächsten Schritt löschen entscheiden wir einfach am Signifikanzniveau: den am wenigsten signifikanten Effekt nehmen wir aus dem Modell.

Indem wir, Schritt für Schritt, einen Effekt aus dem Modell entfernen, wird die Varianz neu auf die übrigen Effekte verteilt. Dadurch verändern sich möglicherweise die Varianzanteile der übrig gebliebenen Effekte und entsprechend ihre Signifikanzniveaus. Deshalb ist eine *schrittweise* Eliminierung wichtig. Dies können wir solange treiben, bis nur noch Effekt im Modell

[3] Dies ist kein offizielles Verfahren oder gar eine akzeptierte Herangehensweise. Im Gegenteil, man kann sogar zeigen, dass sie zu falschen Ergebnissen führen kann. Was nicht heißt, dass diese Herangehensweise nicht doch häufig praktiziert wird.

sind, die (1) signifikant ($P < 0.05$)[4] oder (2) Teil eines signifikanten Interaktionseffektes sind (Crawley 2002).

Wir unterscheiden *backward selection* und *forward selection*. Die *forward*-Selektion der Variablen unterscheidet sich von der gerade beschriebenen *backward* dadurch, dass sie nicht mit dem vollen Modell anfängt, und dann Effekte herausnimmt, sondern umgekehrt, mit dem leeren Modell anfängt, und Effekt zufügt. Da hierbei die Reihenfolge der Effektzufügung einen starken Einfluss auf das Endergebnis hat, wird meist die *backward-selection* bevorzugt[5].

Das Problem mit beiden Ansätzen ist, dass sie beide wiederholte Tests darstellen, ohne dass eine entsprechende Korrektur zur Vermeidung Fehler 1. Ordnung stattfindet (siehe Abschnitt 1.3.1 auf Seite 5). Crawley (2002) schlägt deshalb deutlich niedrige P-Werte als Selektionsschwellenwert vor (etwa $P < 0.01$). Gleichzeitig argumentieren andere (etwa Bowerman and O'Connell 1990), dass dieser Schwellenwert deutlich *höher* als 0.05 liegen sollte, da sonst die Gefahr besteht, dass wichtige Variablen ausgelassen werden. Insgesamt liefert eine *backward selection* unserer Erfahrung nach recht gute und zuverlässige Ergebnisse.[6]

Diese Argumente können um andere erweitert werden: Wenn wir ein Experiment durchgeführt haben, dann steht damit auch das statistische Analysemodell fest. Ein nachträgliches (*a posterori*) Modifizieren dieses Modells widerspricht damit der Logik hypothesegeleiteter Tests (Quinn and Keough 2002). Andererseits können bei fehlenden Daten die Parameter eines komplexeren Modells mit deutlich geringerer Genauigkeit geschätzt werden, als in einem reduzierten Modell. Darob raten erfahrene Nutzer (Crawley 2002) ebenso wie Berater von Statistiksoftwareherstellern (Inc. 1999) zu einer Reduzierung der Modellkomplexität bei unausgewogenen (*unbalanced*) Datensätzen.

Simulationen haben gezeigt, dass ein „gutes" Modell nicht notwendigerweise die tatsächlich wichtigen Variablen beinhalten muss (Flack and Chang 1987), dass gleiche Daten zu unterschiedlichen Endmodellen führen können (James and McCulloch 1990), und dass kollineare erklärende Variablen zu großer Variabilität um die Regressionsgerade führen, was dann zum Ausschluss dieser Variablen führt (Chaterjee and Price 1991). Mac Nally (2000) schlägt deshalb unterschiedliche Verfahren vor, je nachdem ob wir Hypothesen testen wollen, oder mit möglichst wenigen Variablen möglichst guter Vorhersagen machen wollen. Wie diese verschiedenen Positionen, die mit oft überraschender Schärfe vorgebracht werden, zeigen, hat sich im Bezug auf Variablenselektion noch kein Standardverfahren durchgesetzt, und jedes Jahr kommen neue Methoden hinzu (siehe Reineking and Schröder 2004b, für einen aktuellen Überblick).

Erwähnen sollten wir noch das Verfahren der „*best subset regression*". Dieses berechnet alle möglichen Kombinationen von Variablen und sucht dasjenige Modell mit der höchsten Güte als bestes heraus. Der Rechenaufwand bei dieser Prozedur ist bei mehreren Variablen enorm und oft aus praktischen Gründen nicht möglich. Außerdem kann man auch gegen dieses Verfahren mehrere der obigen Argumente anführen (etwa multiple Tests). Trotzdem wird es in vielen Lehrbüchern zur linearen Regression empfohlen (etwa James and McCulloch 1990; Chaterjee and Price 1991; Quinn and Keough 2002). [7]

Auch die Auswahl der Effekte kann nach anderen Gesichtspunkten erfolgen als dem Signifikanzniveau: *log-likelihood*, Walds-Test, *AIC-based selection* usw. führen meist zu unterschiedlich konservativen Modellen (siehe Reineking and Schröder 2004a, für eine Übersicht).

[4] In der ANOVA-Tabelle, nicht im t-Test der Steigungen!

[5] Auch eine Kombination von beiden ist möglich, indem die Effekte *backwards* eliminiert werden, aber ihre Signifikanz *forward* getestet wird. Dies ist auch die Grundeinstellung in R.

[6] Einzig bei sehr vielen erklärenden Variablen neigt dieses Verfahren zu etwas arg großen Modellen. Hier kann durch eine Option in der `step` bzw `stepAIC`-Funktion Abhilfe geschaffen werden, indem man `k=log(N)` setzt, wobei `N` die Anzahl der Datenpunkte ist. Dieses Maß heißt BIC, im Gegensatz zur Grundeinstellung des AIC, und darunter findet man auch Informationen in der weiterführenden Literatur (Burnham and Anderson 2002).

[7] In R ist die „*best subset*"-Selektion im *package* **leaps** durch die Funktion `regsubsets` implementiert. Siehe dort für Beispiele.

Schlussendlich müssen wir das Ergebnis doch noch auf seine Sinnhaftigkeit überprüfen (siehe Beispiel).

Die automatische Modellvereinfachung in R ist mittels der Funktion `step` implementiert (siehe auf `stepAIC` im *package* **MASS** für erweiterte Optionen und die Anwendung auf `glm`). Zunächst gehen wir die entsprechenden Schritte aber zu „Fuß".
(Fortsetzung.)
Zunächst löschen wir die 4-Wege-Interaktion, dann sukzessive alle weiteren nicht-signifikanten nicht-marginalen Effekte. Dafür benutzen wir den Befehl `update`, der uns das selektive Löschen (und Hinzufügen) eines Terms erlaubt.

```
> anova(fmm4 <- update(fmm3, . ~ . - MAP:MAT:DJFMAP:JJAMAP))

Analysis of Variance Table

Response: C3
                  Df Sum Sq Mean Sq F value    Pr(>F)
MAP                1 0.0017  0.0017  0.0252   0.87439
MAT                1 2.5766  2.5766 38.3211 6.601e-08 ***
DJFMAP             1 0.0003  0.0003  0.0051   0.94326
JJAMAP             1 0.2951  0.2951  4.3884   0.04056 *
MAP:MAT            1 0.0031  0.0031  0.0459   0.83110
MAP:DJFMAP         1 0.0178  0.0178  0.2652   0.60854
MAT:DJFMAP         1 0.1482  0.1482  2.2048   0.14299
MAP:JJAMAP         1 0.2063  0.2063  3.0680   0.08513 .
MAT:JJAMAP         1 0.1567  0.1567  2.3301   0.13233
DJFMAP:JJAMAP      1 0.0567  0.0567  0.8429   0.36238
MAP:MAT:DJFMAP     1 0.0161  0.0161  0.2396   0.62637
MAP:MAT:JJAMAP     1 0.2912  0.2912  4.3308   0.04185 *
MAP:DJFMAP:JJAMAP  1 0.0423  0.0423  0.6291   0.43093
MAT:DJFMAP:JJAMAP  1 0.0196  0.0196  0.2909   0.59168
Residuals         58 3.8998  0.0672
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> anova(fmm5 <- update(fmm4, . ~ . - MAP:DJFMAP:JJAMAP))

Analysis of Variance Table

Response: C3
                  Df Sum Sq Mean Sq F value    Pr(>F)
MAP                1 0.0017 0.00170  0.0256   0.87352
MAT                1 2.5766 2.57664 38.8551 5.284e-08 ***
DJFMAP             1 0.0003 0.00034  0.0052   0.94286
JJAMAP             1 0.2951 0.29507  4.4495   0.03916 *
MAP:MAT            1 0.0031 0.00309  0.0465   0.82993
MAP:DJFMAP         1 0.0178 0.01783  0.2689   0.60603
MAT:DJFMAP         1 0.1482 0.14825  2.2356   0.14020
MAP:JJAMAP         1 0.2063 0.20629  3.1108   0.08295 .
MAT:JJAMAP         1 0.1567 0.15667  2.3626   0.12962
DJFMAP:JJAMAP      1 0.0567 0.05667  0.8546   0.35901
MAP:MAT:DJFMAP     1 0.0161 0.01611  0.2429   0.62395
MAP:MAT:JJAMAP     1 0.2912 0.29120  4.3912   0.04043 *
MAT:DJFMAP:JJAMAP  1 0.0491 0.04915  0.7411   0.39279
Residuals         59 3.9125 0.06631
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> anova(fmm6 <- update(fmm5, . ~ . - MAT:DJFMAP:JJAMAP))
```

```
Analysis of Variance Table

Response: C3
               Df Sum Sq Mean Sq F value    Pr(>F)
MAP             1 0.0017 0.00170  0.0257   0.87324
MAT             1 2.5766 2.57664 39.0235 4.737e-08 ***
DJFMAP          1 0.0003 0.00034  0.0052   0.94273
JJAMAP          1 0.2951 0.29507  4.4688   0.03868 *
MAP:MAT         1 0.0031 0.00309  0.0467   0.82955
MAP:DJFMAP      1 0.0178 0.01783  0.2700   0.60522
MAT:DJFMAP      1 0.1482 0.14825  2.2452   0.13927
MAP:JJAMAP      1 0.2063 0.20629  3.1242   0.08222 .
MAT:JJAMAP      1 0.1567 0.15667  2.3729   0.12872
DJFMAP:JJAMAP   1 0.0567 0.05667  0.8583   0.35792
MAP:MAT:DJFMAP  1 0.0161 0.01611  0.2440   0.62317
MAP:MAT:JJAMAP  1 0.2912 0.29120  4.4102   0.03994 *
Residuals      60 3.9617 0.06603
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

An dieser Stelle würden manche aufhören, da wir eine signifikante Dreifach-Interaktion haben. Dies würde in der Tat auch die automatisierte Selektion tun:

```
> anova(fmmstep <- step(fmm3, trace = F))

Analysis of Variance Table

Response: C3
               Df Sum Sq Mean Sq F value    Pr(>F)
MAP             1 0.0017  0.0017  0.0257   0.87324
MAT             1 2.5766  2.5766 39.0235 4.737e-08 ***
DJFMAP          1 0.0003  0.0003  0.0052   0.94273
JJAMAP          1 0.2951  0.2951  4.4688   0.03868 *
MAP:MAT         1 0.0031  0.0031  0.0467   0.82955
MAP:DJFMAP      1 0.0178  0.0178  0.2700   0.60522
MAT:DJFMAP      1 0.1482  0.1482  2.2452   0.13927
MAP:JJAMAP      1 0.2063  0.2063  3.1242   0.08222 .
MAT:JJAMAP      1 0.1567  0.1567  2.3729   0.12872
DJFMAP:JJAMAP   1 0.0567  0.0567  0.8583   0.35792
MAP:MAT:DJFMAP  1 0.0161  0.0161  0.2440   0.62317
MAP:MAT:JJAMAP  1 0.2912  0.2912  4.4102   0.03994 *
Residuals      60 3.9617  0.0660
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Wir setzen unsere Suche aber fort, und nehmen $P < 0.05$ als Selektionsschwellenwert. Die Zwischenschritt sind hier nicht ausgedruckt.

```
> anova(fmm7 <- update(fmm6, . ~ . -MAP:MAT:DJFMAP))
> anova(fmm8 <- update(fmm7, . ~ . -MAP:MAT:JJAMAP))
> anova(fmm9 <- update(fmm8, . ~ . -MAP:MAT))
> anova(fmm10 <- update(fmm9, . ~ . -MAP:DJFMAP))
> anova(fmm11 <- update(fmm10, . ~ . -DJFMAP:JJAMAP))
> anova(fmm12 <- update(fmm11, . ~ . -MAP:JJAMAP))
> anova(fmm13 <- update(fmm12, . ~ . -MAT:DJFMAP))
> anova(fmm14 <- update(fmm13, . ~ . -DJFMAP))

> anova(fmm15 <- update(fmm14, . ~ . - MAP))
```

```
Analysis of Variance Table

Response: C3
           Df Sum Sq Mean Sq F value    Pr(>F)
MAT         1 2.2958  2.2958 32.5202 2.693e-07 ***
JJAMAP      1 0.0594  0.0594  0.8411   0.36227
MAT:JJAMAP  1 0.5051  0.5051  7.1544   0.00933 **
Residuals  69 4.8712  0.0706
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Dies ist unser Endmodell. Es enthält einen nicht signifikante Effekt (JJAMAP, der als Teil einer signifikanten Interaktion im Modell bleiben muss. Der R^2-Wert liegt nur ein kleines bisschen unter dem der automatischen Selektion, aber unser Modell ist deutlich schlanker (mit nur 3 statt 12 erklärende Variablen).
Ein Problem ist, dass wir mit einem anderen Ausgangsmodell zu anderen Ergebnissen kommen. Wenn wir beispielsweise nur mit den Haupteffekten beginnen, erhalten wir andere Effekte als signifikant:

```
> anova(fmm15 <- step(lm(C3 ~ MAP + MAT + DJFMAP + JJAMAP), trace = F))

Analysis of Variance Table

Response: C3
          Df Sum Sq Mean Sq F value   Pr(>F)
MAT        1 2.2958  2.2958 32.1462 3.07e-07 ***
DJFMAP     1 0.0456  0.0456  0.6387  0.42692
JJAMAP     1 0.4622  0.4622  6.4713  0.01321 *
Residuals 69 4.9279  0.0714
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> anova(fmm16 <- update(fmm15, . ~ . - DJFMAP))

Analysis of Variance Table

Response: C3
          Df Sum Sq Mean Sq F value   Pr(>F)
MAT        1 2.2958  2.2958 29.8921 6.63e-07 ***
JJAMAP     1 0.0594  0.0594  0.7732   0.3823
Residuals 70 5.3763  0.0768
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> anova(fmm17 <- update(fmm16, . ~ . - JJAMAP))

Analysis of Variance Table

Response: C3
          Df Sum Sq Mean Sq F value    Pr(>F)
MAT        1 2.2958  2.2958  29.988 6.215e-07 ***
Residuals 71 5.4357  0.0766
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Jetzt ist nur noch MAT signifikant. Und noch ein weiteres Ausgangsmodell:

```
> anova(fmm18 <- step(lm(C3 ~ MAP + I(MAP^2) + MAT + I(MAT^2) +
+     DJFMAP + I(DJFMAP^2) + JJAMAP + I(JJAMAP^2)), trace = F))
```

```
Analysis of Variance Table

Response: C3
            Df Sum Sq Mean Sq F value    Pr(>F)
MAT          1 2.2958  2.2958 32.8982 2.447e-07 ***
I(MAT^2)     1 0.1914  0.1914  2.7427   0.10231
DJFMAP       1 0.0171  0.0171  0.2446   0.62250
I(JJAMAP^2)  1 0.4817  0.4817  6.9030   0.01063 *
Residuals   68 4.7455  0.0698
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Auch hier müssen wir noch etwas nachbessern, da ja ein Effekt mit den verwendeten Kriterien nicht aus dem Modell herausgenommen wurde:

```
> summary(fmm19 <- update(fmm18, . ~ . - I(MAT^2)))

Call:
lm(formula = C3 ~ MAT + DJFMAP + I(JJAMAP^2))

Residuals:
     Min       1Q   Median       3Q      Max
-0.58235 -0.19550 -0.00913  0.18879  0.62977

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.303958   0.185435   1.639  0.10573
MAT         -0.040349   0.006748  -5.979 8.88e-08 ***
DJFMAP      -1.180366   0.477335  -2.473  0.01587 *
I(JJAMAP^2) -1.887286   0.705421  -2.675  0.00931 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.266 on 69 degrees of freedom
Multiple R-Squared: 0.3684,        Adjusted R-squared: 0.3409
F-statistic: 13.41 on 3 and 69 DF,  p-value: 5.436e-07
```

Und wieder ein anderes Endmodell! Aus diesem Prozedere wird deutlich, dass es keinen goldenen Weg zur Wahrheit gibt. Wenn wir die Toleranzen für die Effekte der unterschiedlichen Modelle berechneten, dann sähen wir, dass das erste Modell (`fmm`) extrem schlechte Werte bekommt, die anderen hingegen nicht. Ein Vergleich der erklärten Varianz der unterschiedlichen Modelle ist auch oft aufschlussreich:

```
> summary(fmmstep)$adj.r.squared

[1] 0.3851132

> summary(fmm15)$adj.r.squared

[1] 0.3349114

> summary(fmm18)$adj.r.squared

[1] 0.3501132

> summary(fmm19)$adj.r.squared

[1] 0.3409063
```

Das erste, überfrachtete Modell und seine auf quadratischen Termen bauender Kollege mit drei Variablen erklären also am Meisten der Varianz, die anderen beiden sind etwa gleichwertig. Sie unterscheiden sich auch nur darin, dass im letzten JJAMAP in der zweiten statt der ersten Ordnung eingeht. Aus Gründen der Interpretierbarkeit würden wir aus Modell 3, 4 oder 2 wählen. Schon die Signifikanz der Variablen JJAMAP ist im Bild kaum nachvollziehbar, der Wust an Interaktionen im ersten Modell hingegen entbehrt jeder ökologischer Interpretation. Nichtsdestotrotz berücksichtigte Modell 2 ja aller Interaktionen. Dass nur eine übrig geblieben ist spricht für ihre Bedeutung.
Jetzt müssen wir noch die beiden Teilanalysen zusammenführen. Dazu nehmen wir das zweite Modell und das Interaktionsmodell für Langen- und Breitengrade von oben:

```
> anova(fn <- lm(C3 ~ LAT * LONG + MAT + DJFMAP + JJAMAP + MAT:JJAMAP))

Analysis of Variance Table

Response: C3
           Df Sum Sq Mean Sq F value    Pr(>F)
LAT         1 3.5703  3.5703 64.3866 2.644e-11 ***
LONG        1 0.0412  0.0412  0.7438   0.39161
MAT         1 0.0064  0.0064  0.1160   0.73451
DJFMAP      1 0.0090  0.0090  0.1615   0.68909
JJAMAP      1 0.2531  0.2531  4.5651   0.03640 *
LAT:LONG    1 0.2236  0.2236  4.0326   0.04879 *
MAT:JJAMAP  1 0.0235  0.0235  0.4237   0.51741
Residuals  65 3.6043  0.0555
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> anova(step(fn, trace = F))

Analysis of Variance Table

Response: C3
          Df Sum Sq Mean Sq F value    Pr(>F)
LAT        1 3.5703  3.5703  65.527 1.340e-11 ***
LONG       1 0.0412  0.0412   0.757   0.38728
LAT:LONG   1 0.3604  0.3604   6.615   0.01227 *
Residuals 69 3.7595  0.0545
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nach Korrektur für geografische Länge und Breite sind die Umweltparameter nicht mehr signifikant! Dies Ergebnis erhalten wir auch, wenn wir alle erklärenden Variablen additiv (plus die Interaktion von LAT und LONG) ins Modell nehmen. Lassen wir die Interaktion von LAT und LONG weg, so erhalten wir ein anderes Modell:

```
> anova(fo <- step(lm(C3 ~ LAT + LONG + MAT + DJFMAP + JJAMAP + MAT:JJAMAP),
+     trace = F))

Analysis of Variance Table

Response: C3
          Df  Sum Sq Mean Sq F value    Pr(>F)
LAT        1 2.18392 2.18392  56.895 1.175e-10 ***
Residuals 71 2.72534 0.03839
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(fo)$r.squared
```

```
[1] 0.4448568
```

Was lernen wir aus dieser Übung? Welches Modell ist denn nun das „Richtige“?

Diese Frage ist normativ zu beantworten, d.h. der Anwender der Statistik, also wir, müssen uns über unsere Fragestellung und über die Biologie dahinter im Klaren sein. Wenn wir den Effekt von Niederschlag (`MAP`) und Temperatur (`MAT`) deskriptiv erfassen wollen, so sind wir auf unterschiedliche Orte mit unterschiedlichem Klima angewiesen. Diese liegen aber in einem bestimmten Muster in der Landschaft. Im C_3-Pflanzenbeispiel aus Nordamerika nimmt also die Jahresdurchschnittstemperatur mit der Breite ab, und der Regen mit der Länge. Somit erklären Breiten- und Längengrad einen Großteil der Varianz im Klima und damit im Anteil C_3-Pflanzen an einem bestimmten Ort. Es wäre aber auch legitim, nur nach dem Einfluss des Klimas zu fragen, und `LAT` und `LONG` als Variablen aus der Erklärungsliste zu streichen. Dies ist, wie gesagt, eine nicht-statistische Entscheidung.

Wenn wir das Modell suchen, dass die Varianz in den Daten am besten erklärt, dann geraten wir möglicherweise an ein komplexes Modell mit vielen ökologisch uninterpretierbaren Interaktionen (siehe obiges Beispiel). Andererseits ist das letzte Modell mit weniger Variablen nahezu genauso gut, ist aber deutlich interpretierbarer. Als ökologisch interessant kann es trotzdem kaum gelten: Breitengrade haben sicherlich keinen biologischen Effekt auf den Anteil C_3-Pflanzen, Photoperiode (vom Breitengrad direkt abhängig) und Temperaturen aber bestimmt.

Ein ganz anderer Ansatz, in dem aus den verschiedenen kollinearen Variablen tatsächlich unabhängige Pseudovariablen (*compound variable*) konstruiert werden (z.B. Hauptkomponentenanalyse), wird uns bei den multivariaten Verfahren begegnen.

6.3. Vorsortierung der Variablen: explorative Datenanalyse

Manchmal haben wir einfach zu viele erklärende Variablen in unserem Datensatz. Dann wäre es natürlich schön, wenn wir die unwichtigen aus der Analyse ausschließen könnten. Welche Variablen unwichtig sind entscheiden wir vorrangig anhand unseres biologischen Verständnisses!

Sollten jetzt trotzdem noch viele erklärende Variablen übrig sein, so stehen wir vor einem Problem, dem sich das *data mining* verschrieben hat (Hastie et al. 2008). Es geht darum, aus der Vielzahl möglicher Zusammenhänge die wichtigen herauszufiltern, ohne dabei Scheinzusammenhängen aufzusitzen. Die zumeist rechenaufwändigen Verfahren werden in der Literatur häufig unter *machine learning* erfasst. Hier seien nur zwei Methoden exemplarisch genannt: hierarchische Partitionierung und randomForest.

6.3.1. Hierarchische Partitionierung

Wenn erklärende Variablen untereinander kausal verknüpft sind (oder zumindest miteinander korrelieren), wie etwa Breitengrad und Jahresdurchschnittstemperatur, wirkt sich jede Variable sowohl direkt (*unabhängig*) als auch auf dem Umweg über die mit ihr korrelierten Variablen auf die abhängige aus (*gemeinsamer* Beitrag). Um nachzuzeichnen welche Erklärungsanteile direkt und welche im Verbund mit anderen Variablen zustandekommen, gibt es die Methode der hierarchischen Partitionierung (*hierarchical partitioning*, Chevan and Sutherland 1991; Mac Nally 2000).

Die Idee dahinter ist einfach, ihre Umsetzung etwas rechenaufwendig. Stellen wir uns ein multiples Regressionsmodell vor (etwa das aus obigem Beispiel, in dem alle fünf Variablen additiv ins Modell eingehen). Dann können wir die Wichtigkeit einer Variablen dadurch ermitteln, dass wir sie aus dem Modell nehmen und schauen, wieviel schlechter das neue Modell ist. Als Maß für den Fit des Modells nimmt man üblicherweise (bei normalverteilten Daten) einfach den (*adjusted*) R^2-Wert. Dies tun wir nun *für alle möglichen Variablenkombinationen.*

Bei drei erklärenden Variablen `A`, `B` und `C` ergeben sich $2^3 = 8$ Modelle: keine Variable (0), `A`, `B`, `C`, `A+B`, `A+C`, `B+C` und `A+B+C`. Wir haben hier 4 Hierarchiestufen (0, 1, 1, 1, 2, 2, 2, 3). Um den Effekt von `A` zu studieren, würden wir es zu jedem Modell ohne `A` hinzufügen und die Veränderung analysieren: `A` vs. 0, `A+B` vs. `B`, `A+C` vs. `C`, `A+B+C` vs. `B+C`. Innerhalb einer Hierarchiestufe werden diese Veränderungen dann gemittelt, und schließlich über alle Stufen hinweg gemittelt. Dies ergibt den *unabhängigen* Einfluss von `A` auf die Antwortvariable. Der gemeinsame Beitrag von `A` wird aus der Differenz des Fits des kompletten Modells und den Fits der Teilmodelle errechnet (Chevan and Sutherland 1991).

Leider können die Ergebnisse der hierarchischen Partitionierung nicht mit Signifikanzwerten versehen werden, da sie ja über alle möglichen Kombination hinweg gemittelt sind. Sie dienen damit mehr als Interpretations- oder Auswahlmittel bei einer multiplen Regression.

Schauen wir uns die Umsetzung anhand der Beispieldaten von Paruelo and Lauenroth (1996) an.

```
> library(hier.part)

Loading required package: gtools

> hipa <- hier.part(C3, data.frame(CLAT = CLAT, CLONG = CLONG,
+     MAP = MAP, MAT = MAT, DJFMAP = DJFMAP, JJAMAP = JJAMAP))
> hipa

$gfs
 [1] 2.780562 2.039904 2.780492 2.780257 2.331455 2.772174 2.777950 2.029769
 [9] 1.992298 2.035930 2.037785 2.023089 2.779287 2.304933 2.762604 2.776470
[17] 2.270064 2.768612 2.777810 2.321652 2.318685 2.730877 1.984486 2.028184
[25] 2.027007 1.985771 1.992294 1.989350 1.965855 2.034321 2.019621 1.963045
[33] 2.269588 2.762343 2.771894 2.302638 2.249906 2.702313 2.269988 2.246265
[41] 2.709324 2.219886 1.984434 1.984107 1.965725 2.025975 1.985770 1.962527
[49] 1.989349 1.965751 1.946604 1.962952 2.269436 2.238248 2.700565 2.217098
[57] 2.204037 1.984019 1.965586 1.939353 1.962507 1.946259 2.202556 1.939275

$IJ
                  I            J         Total
CLAT   -0.515027195 -0.225630582 -7.406578e-01
CLONG  -0.007766134  0.007696324 -6.981013e-05
MAP    -0.022685327  0.022380490 -3.048370e-04
MAT    -0.239205484 -0.209901540 -4.491070e-01
DJFMAP -0.021897520  0.013509350 -8.388170e-03
JJAMAP -0.034705317  0.032092957 -2.612360e-03

$I.perc
                I
CLAT   61.2189668
CLONG   0.9231254
MAP     2.6965028
MAT    28.4332802
DJFMAP  2.6028597
JJAMAP  4.1252650
```

Wir erhalten als *output* auch eine grafische Darstellung (Abbildung 6.2) der unabhängigen Beiträge jeder Variablen an der erklärten Varianz. Interessanter ist die Textausgabe, in der neben den *independent effects I* auch die *joined effects* ausgegeben werden. Beachte, dass die `I.perc`-Werte sich zu 100 addieren, auch wenn nur 4% der Varianz von dem Modell erklärt würde! Beachte weiterhin, dass negative indirekte Effekt daraufhinweisen, dass *de facto* das Modell indirekt Erklärungskraft durch diese Variable verliert.
An den Werten erkennen wir, dass vor allem `CLAT` und und `MAT` wichtig sind, wobei `MAT` ebenso stark indirekt wie direkt den Anteil C_3-Pflanzen beeinflusst. Wenn wir das Modell jetzt verändern, so verändern sich auch diese Werte. Wie wir aus obiger multipler Regression wissen, ist die Interaktion von `LAT` und `LONG` auch wichtig. Ziehen wir diese also einmal mit ins hierarchische Partitionieren:

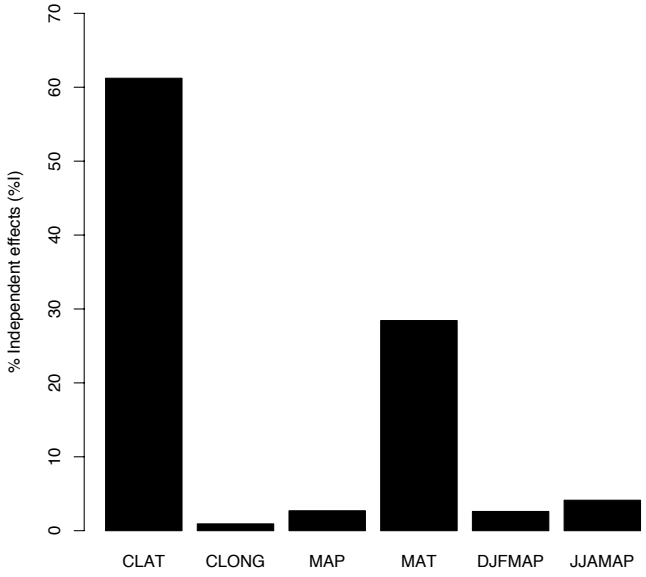


Abbildung 6.2.: Visualisierung der Anteile unabhängiger Beiträge der Variablen an der insgesamt durch das Modell erklärten Varianz.

```
> hipa2 <- hier.part(C3, data.frame(CLAT = CLAT, CLONG = CLONG,
+     CLL = CLAT * CLONG, MAP = MAP, MAT = MAT, DJFMAP = DJFMAP,
+     JJAMAP = JJAMAP))
> hipa2$IJ.perc
```

Praktisch keine Veränderung: `CLAT` und `MAT` bleiben als wichtigste Effekt erhalten.
Dies scheint doch ein ganz gutes Modell zu sein. Schauen wir uns die entsprechend multiple Regression an (nur die Tabelle hier wiedergegeben):

```
> anova(lm(C3 ~ CLAT * CLONG + MAT))

Analysis of Variance Table

Response: C3
           Df Sum Sq Mean Sq F value    Pr(>F)
CLAT        1 3.5703  3.5703 65.1210 1.637e-11 ***
CLONG       1 0.0412  0.0412  0.7523   0.38880
MAT         1 0.0064  0.0064  0.1173   0.73301
CLAT:CLONG  1 0.3854  0.3854  7.0290   0.00997 **
Residuals  68 3.7282  0.0548
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(lm(C3 ~ CLAT * CLONG + MAT))$r.squared

[1] 0.5177974
```

Hier sehen wir, dass hierarchisches Partitionieren nur ein Hilfsmittel ist: `MAT` ist nicht signifikant! Wir würden also wieder vereinfachen und zu der Minimalvariate, die auch schon oben vorgerechnet wurde kommen: `C3`$\sim$`CLAT*CLONG`. Dieses reduzierte Modell hat sogar einen höheren *adjusted* R^2.

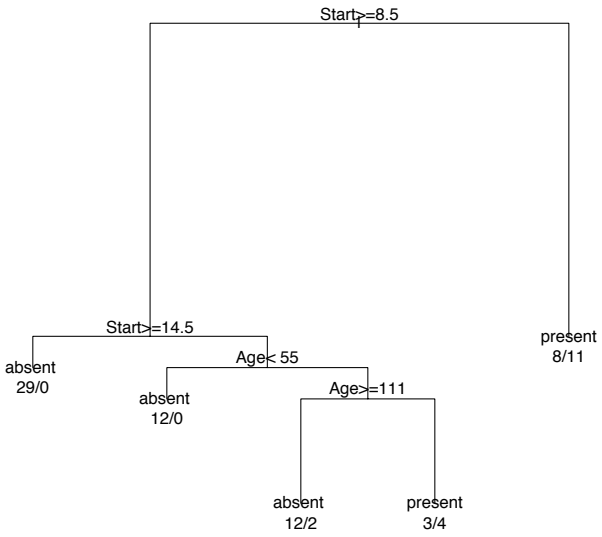


Abbildung 6.3.: Beispiel für einen *classification and regression tree*. Siehe `?rpart` im *package* **rpart** für Details und Optionen.

6.3.2. Random Forest

Um dieses Verfahren effizient zu nutzen muss man eine Methode kennen, die wir noch nicht kennengelernt haben: *classification and regression trees*, kurz CART. CARTs funktionieren nach einem dichotomen Prinzip. Aus einer erklärenden kontinuierlichen Variablen wird ein Faktor mit zwei Leveln gemacht, und zwar so, dass die zum Level 1 gehörenden y-Werte sich maximal von denen des Level 2 unterscheiden. Diesen Schritt bezeichnet man als *branching*, und wenn man ihn häufig wiederholt (für alle erklärenden Variablen entsprechend ihrer Klassifikationsgüte), so entsteht ein ganzer Entscheidungsbaum. Wir wollen das hier nicht weiter vertiefen, und es wird am einfachsten deutlich, wenn man sich einfach einmal ein Beispiel anschaut (Abb. 6.3).

In diesem Fall soll das Auftreten von Kyphosen nach einer Wirbelsäulenoperation bei Kindern. Erklärende Variablen sind Alter (`Age`), Anzahl Wirbel, die operiert wurden (`Number`) und die Wirbelnummer des obersten operierten Wirbels (`Start`). Wie wir erkennen können, ist der erste Teilungsschritt durch die Wirbelnummer bestimmt, und dann geht es iterativ so weiter. Die Zahlen am Ende jedes Zweigs (*terminal node*) gibt den Klassifikationserfolg an: `8/11` bedeutet dass 8 Kyphosen korrekt und 11 inkorrekt vorhergesagt wurden.

CARTs bilden, was auf den ersten Blick nicht erkennbar ist, automatisch nicht-lineares Verhalten und Interaktionen zwischen Variablen ab. Ersteres weil die Verzweigung ja nicht zentral erfolgen muss und in jedem Zweig wiederholt auftauchen kann. Letzteres, weil eine "interagierende" Variable nur in einem, nicht aber im anderen Zweig zur weiteren Verzweigung beiträgt. CARTs sind deshalb trotz ihrer dichotom-gestuften Modellierung oft gut geeignet, um ökologische Zusammenhänge grob zu fitten. Andererseits neigen CARTS zur Überparametrisierung und damit zu einer wenig robusten Modellierung.

Was uns aber hier eigentlich interessiert, ist nicht CART, sondern *random forest* (Breiman 2001). Ein Wald besteht aus Bäumen, ein *random forest* aus *random trees*: *random forest* wählt zufällig Datenpunkte und Variablen aus und baut daraus einen CART. Diese Prozedur wird ein paar Hundert Mal wiederholt, woraus ein "Wald" entsteht. Jeder "Baum" wird zudem hinsichtlich seiner Regressions- bzw. Klassifikationsgüte an den nicht benutzten Datenpunkten

evaluiert (*out-of-bag evaluation*). Schließlich werden zur Vorhersage alle Bäume gemittelt, und zwar gewichtet nach ihrer *oob*-Güte. Als Schmankerl bietet *random forest* die Option die Wichtigkeit von Variablen zu quantifizieren, je nachdem wie entscheidend sie in den Bäumen zur Verzweigung beiträgt. *Random forest* leidet nicht an den Nachteilen der CARTs, wohl aber bietet es ihre Vorteile. Wir wollen hier vor allem die Wichtigkeit nutzen, um eindeutig unkorrelierte Variablen aus unserem Datensatz zu eliminieren, bevor wir die übrigen wieder dem traditionellen linearen Modell zuführen.
`randomForst`, im gleichnamigen **package**, ist denkbar einfach in der Anwendung[8]:

```
> library(randomForest)
> rf <- randomForest(x=cbind(CLAT, CLONG, MAT, MAP, JJAMAP, DJFMAP), y=C3,
+                         importance=T, mtry=4, ntree=1000)
> importance(rf)

          %IncMSE IncNodePurity
CLAT   23.0193294     2.4553557
CLONG   7.6079874     0.7609013
MAT    16.7284853     1.8148605
MAP     0.9867019     0.6690566
JJAMAP  6.9370069     0.7709668
DJFMAP  5.5569596     0.5360729
```

`randomForest` misst die Wichtigkeit der Variablen auf zweierlei Art[9]. `%IncMSE` ist der mittlere Abfall in der Genauigkeit des Modells, wenn diese Variable nicht zur Verfügung steht; `IncNodePurity` ist der mittlere Abfall in den mittleren Abweichungsquadraten (MSE). In beiden Fällen bedeuten höhere Werte auch einen größerern Einfluss der Variablen.

Wir sehen, dass zentrierter Breitengrad (`CLAT`) und mittlere Jahrestemperatur mit Abstand vor den Niederschlagsvariablen führen. Dies Ergebnis ist also in etwa kompatible mit dem aus dem hierarchischen Partitioniern (Abb. 6.2). Da *random forest* auf vielen Wiederholungen beruht und intern validiert ist, vertrauen *wir* diesem Verfahren mehr als der hierarchischen Partitionierung.
Da wir nicht-lineare Zusammenhänge aufgrund unser früheren Untersuchungen ausschließen können, lautet unser `randomForest`-basiertes lineares Modell also etwa:

```
> anova(lm(C3 ~ CLAT+MAT))

Analysis of Variance Table

Response: C3
          Df  Sum Sq Mean Sq F value    Pr(>F)
CLAT       1 2.18392 2.18392  56.887 1.270e-10 ***
MAT        1 0.03801 0.03801   0.990    0.3232
Residuals 70 2.68734 0.03839
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(lm(C3 ~ CLAT+MAT))$r.squared

[1] 0.4525985
```

Dieses Modell beinhaltet also weiterhin einen nicht-signifikanten Effekt, der bei schrittweiser Vereinfachung herausfallen würde. Übrig bliebe dann genau das gleiche Modell, wie das, das direkt aus dem `step` entstanden war. Das ist in diesem Fall didaktisch ungünstig. Das Filtern von Variablen mittels `randomForest` kann nämlich grundsätzlich schon zu anderen Modellen führen.

[8] Da die Bäume mit einem zufälligen Element konstruiert werden, wird jeder Lauf von randomForest leicht andere Ergebnisse liefern.

[9] Im *package* **caret** steht mit `varImp` eine ähnliche Funktion auch für andere Modellierungsverfahren zur Verfügung, etwa für CARTs.

6.4. Mehrere, kategoriale erklärende Variablen: ANOVA und LM

Wenn wir unsere abhängige Variable durch Variablen erklären wollen, die nicht kontinuierlich (wie bei der multiplen Regression) sondern kategorial sind, stehen uns zwei Verfahren zur Verfügung: die Varianzanalyse (ANalysis Of VAriance) und das allgemeine lineare Modell (*linear model*). Betrachten wir Letztes zuerst, um uns dann ausgiebig mit der ANOVA zu beschäftigen.

6.4.1. Kategoriale erklärende Variablen im LM

Das LM folgt unmittelbar aus der multiplen Regression, denn hier werden die erklärenden Variablen genauso verrechnet.

Ein Beispiel: Wir wollen wissen, ob die beiden europäischen Unterarten des Kormoran (*Phalacrocorax carbo sinensis* und *P. c. carbo*) unterschiedlich lange nach Nahrung tauchen, und zwar je nach Jahreszeit. Unsere Antwortvariable (`Tauchzeit`) soll also erklärt werden durch die kategorialen Variablen `Unterart` (Werte `S` und `C`) und `Jahreszeit` (F, S, H, W). Schauen wir uns erst einmal die Daten an (Abbildung 6.4).

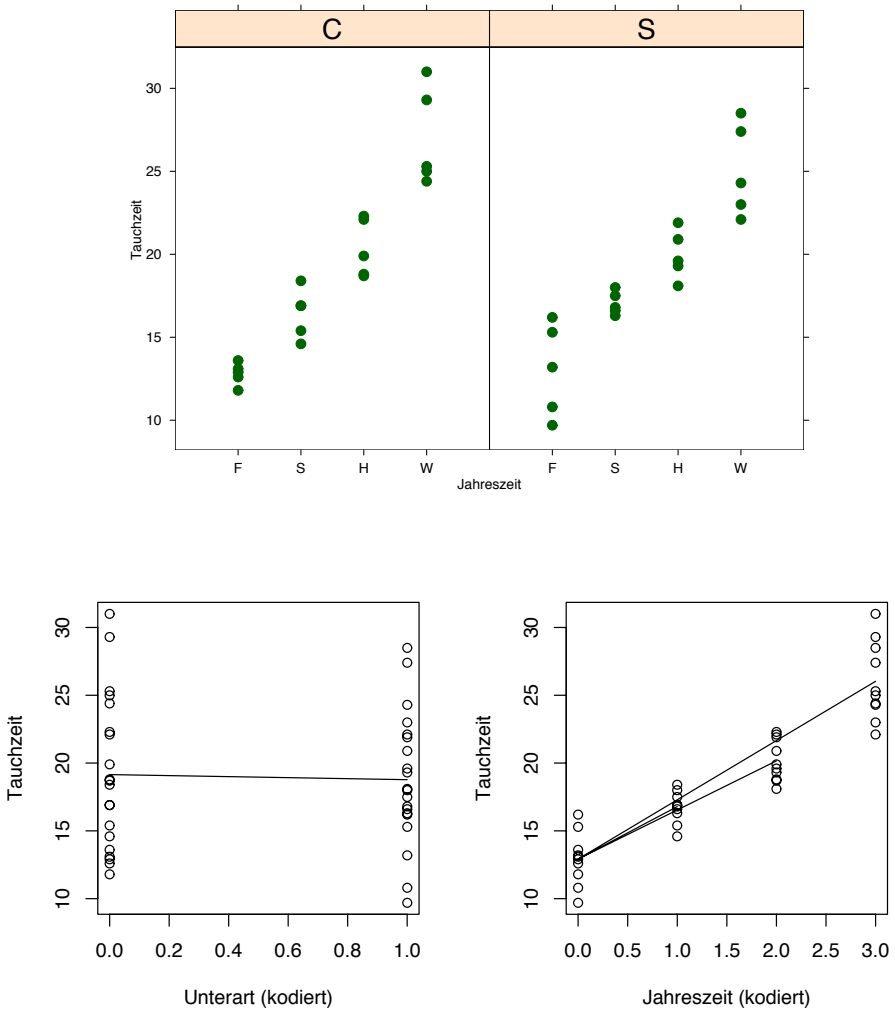


Abbildung 6.4.: Tauchzeit [in Minuten] der beiden europäischen Kormoranunterarten *Phalacrocorax carbo carbo* (`C`) und *sinensis* (`S`) in den vier Jahreszeiten in jeweils 5 Messungen. Die beiden unteren Abbildungen bilden die Effekte nur jeweils eines Faktors ab, und die aus dieser Regression resultierende Gerade. (Daten erfunden.)

Wir sehen einen klaren jahreszeitlichen Effekt und die größere Unterart (`C`) scheint tatsächlich etwas länger zu tauchen als die kleinere (`S`). Wir können jetzt diese Faktoren numerisch kodieren, indem wir die Unterarten nicht `C` und `S` nennen, sondern 0 und 1. Ähnlich können wir für jede Jahreszeit eine sog. *dummy*-Variable anlegen, die in dieser Jahreszeit 1 und sonst 0 ist. Jetzt haben wir aus den kategorialen Variablen numerische gemacht (wenn auch keine kontinuierlichen) und können mit einer Regression nach Effekten suchen.

Die Steigung des Effekts `UnterartS` (also die Steigung von `C` ausgehend nach `S`) lässt sich (mit Formel 5.4 auf Seite 63) berechnen als -1.84. `S` hat also eine um etwa 2 min kürzere Tauchzeit als `C`. Der Wert für `C` wird durch den y-Achsenabschnitt gegeben (12.78), da der x-Kode für `C` ja 0 ist (siehe Abbildung 6.4, links unten).

Für die Jahreszeiten gibt es jetzt drei Geradensteigungen: von `F` nach `S`, `H` und `W` mit Werten von 3.88, 8.10 und 13.04. Der Achsenabschnitt für `F` ist (analog zur Unterart `C`) 12.78 (siehe Abbildung 6.4, links unten).

Die gerade berechneten Regressionskoeffizienten berücksichtigen nicht mögliche Wechselwirkungen von Unterart und Jahreszeit. Sie sind nur die bivariaten Regressionen. Da Unterart und Jahreszeit miteinander interagieren können, also die Unterarten etwa im Sommer gleiche, aber im Winter unterschiedliche Tauchzeiten haben, müssen wir drei weitere Koeffizienten bestimmen: `JahreszeitS:UnterartS`, `JahreszeitH:UnterartS` und `JahreszeitW:UnterartS`. Diese drei Koeffizienten geben also die Abweichung für die drei Jahreszeiten von dem Wert, der sich aus `JahreszeitF` und `UnterartC` errechnen lässt, an.

Bevor wir dies am Beispiel durchführen sei noch erwähnt, dass sich, wie bei der Regression, für jeden Koeffizienten ein t-Test auf Abweichung von Null durchführen lässt.

Hier kommt der R-code für Abbildung 6.4 und das dazugehörige lineare Modell:

```
> kormoran <- read.table("Kormoran.txt", header = T)
> attach(kormoran)
> Jahreszeit <- factor(Jahreszeit, levels = c("F", "S", "H", "W"))
> library(lattice)
> trellis.par.set("background$col" = "white")
> xyplot(Tauchzeit ~ Jahreszeit | Unterart, pch = 16, cex = 1.3,
+     par.strip.text = list(cex = 2), scales = list(alternating = F))
> par(mfrow = c(1, 2))
> plot(Tauchzeit ~ I(as.numeric(Unterart) - 1), xlab = "Unterart (kodiert)")
> lines(c(0, 1), c(19.15, 19.15 - 0.375))
> plot(Tauchzeit ~ I(as.numeric(Jahreszeit) - 1), xlab = "Jahreszeit (kodiert)")
> lines(c(0, 1), c(12.92, 12.92 + 3.87))
> lines(c(0, 2), c(12.92, 12.92 + 7.24))
> lines(c(0, 3), c(12.92, 12.92 + 13.11))
> summary(fm <- lm(Tauchzeit ~ Unterart * Jahreszeit))

Call:
lm(formula = Tauchzeit ~ Unterart * Jahreszeit)

Residuals:
   Min     1Q Median     3Q    Max
-3.280 -0.905 -0.290  0.945  4.580

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)           12.7800     0.7288  17.535  < 2e-16 ***
UnterartS             -1.8400     1.0307  -1.785 0.083720 .
JahreszeitS            3.8800     1.0307   3.764 0.000676 ***
JahreszeitH            8.1000     1.0307   7.859 5.76e-09 ***
JahreszeitW           13.0400     1.0307  12.651 5.38e-14 ***
UnterartS:JahreszeitS -1.3000     1.4577  -0.892 0.379139
UnterartS:JahreszeitH -1.4600     1.4577  -1.002 0.324051
```

```
UnterartS:JahreszeitW  -2.9600     1.4577  -2.031 0.050671 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.63 on 32 degrees of freedom
Multiple R-Squared: 0.9114,        Adjusted R-squared: 0.892
F-statistic: 47.02 on 7 and 32 DF,  p-value: 4.638e-15
```

Da der Koeffizient für `UnterartS` nicht signifikant von 0 unterschiedlich ist, deutet sich an, dass die Unterarten bezüglich ihrer Tauchzeit wahrscheinlich nicht unterscheidbar sind. Um dies herauszubekommen, lassen wir uns die ANOVA-Tabelle ausgeben. Darin wird auf die Signifikanz der Effekte getestet.

```
> anova(fm)

Analysis of Variance Table

Response: Tauchzeit
                   Df Sum Sq Mean Sq F value    Pr(>F)
Unterart            1 106.93  106.93 40.2594 4.013e-07 ***
Jahreszeit          3 756.17  252.06 94.9009 5.185e-16 ***
Unterart:Jahreszeit  3  11.01    3.67  1.3817    0.2661
Residuals           32  84.99    2.66
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Da im vorliegenden Fall die Interaktionenskoeffizienten (und in der ANOVA-Tabelle die Interaktion insgesamt) nicht signifikant sind können wir das Modell vereinfachen:

```
> anova(lm(Tauchzeit ~ Unterart + Jahreszeit))

Analysis of Variance Table

Response: Tauchzeit
           Df Sum Sq Mean Sq F value    Pr(>F)
Unterart    1 106.93  106.93  38.984 3.691e-07 ***
Jahreszeit  3 756.17  252.06  91.895 < 2.2e-16 ***
Residuals  35  96.00    2.74
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Wenn wir alle diese Koeffizienten berechnet haben, können wir den vorhergesagten Wert einer Beobachtung daraus rekonstruieren. Dies ist praktisch eine dreidimensionale Rekonstruktion der Daten. Dimensionen 1 und 2 sind Unterart und Jahrezeit, während die Tauchzeit die dritte Dimension darstellt. Wenn wir den vorhergesagten Mittelwert für die Unterart `S` im Herbst wissen wollen, so addieren wir folgende Koeffizienten des obigen Modells:
`(Intercept)+UnterartS+JahreszeitH+UnterartS:JahreszeitH` $= 12.78 - 1.84 + 8.10 - 1.46 = 17.58$. [10]

6.4.2. Kategoriale erklärende Variablen in der ANOVA

In einer Varianzanalyse verläuft der Test auf Signifikanz beteiligter Variablen etwas anders ab, als in der Regression dargelegt. Vielmehr ist es eine Erweiterung der Ideen, die in Abschnitt 5.3.1 auf Seite 81 für nur einen Faktor ausgeführt wurde.

[10] In R ist dafür die Funktion `predict` verfügbar: `predict(fm, newdata=data.frame(Unterart="S", Jahreszeit="H"))`

Man kann argumentieren, dass die ANOVA nur eine andere Formulierung des LM ist, und ihr deshalb keinen besonderen Raum einräumen. Andererseits hat kein anderes modernes statistisches Verfahren so schnell und fundamental Einzug in die biologische Statistik gehalten wie die ANOVA. Die von ihr durchgeführten Analysen sind in allen statistischen Softwareprogrammen verfügbar, während dies für LM nicht immer der Fall ist. Somit ist der ANOVA inzwischen eine Sonderstellung angediehen, die es erforderlich macht, auch hier ein paar Absätze über grundsätzliche Unterschiede und Ähnlichkeiten von LM und ANOVA darzulegen. Wir beginnen mit dem gleichen Beispiel wie gerade, den Kormoranen.

Die Berechnungen der ANOVA erfordern alle die Berechnung verschiedener Mittelwerte und der Abweichungen der Datenpunkte von diesen Mittelwerten. Im Beispiel beträgt der Mittelwert über alle Datenpunkte 17.4, und die Summe der Abweichungsquadrate (kurz SS_{res}, für *sum of squares* der Residuen) etwa 959.1 (`sum((Tauchzeit-mean(Tauchzeit))^2)`). Jetzt berechnen wir für die beiden Unterarten einen eigenen Mittelwert (Gruppenmittel) ($\bar{x}_C = 19.03$, $\bar{x}_S = 15.77$), und die SS_{res} um die jeweiligen Gruppenmittel (mit dem Taschenrechner R):

```
> sum((Tauchzeit[Unterart == "C"] - mean(Tauchzeit[Unterart ==
+     "C"]))^2)

[1] 532.1055
```

und

```
> sum((Tauchzeit[Unterart == "S"] - mean(Tauchzeit[Unterart ==
+     "S"]))^2)

[1] 320.0655
```

Es ist also $\mathrm{SS}_C = 532.11$ und $\mathrm{SS}_S = 320.07$, so dass die SS_{res} unter Berücksichtigung des Faktors Unterart = 532.11 + 320.07 = 852.18. Der Effekt (in SS) von Unterart ist also $959.1 - 852.18 = 106.92$.

Treiben wir dieses Spiel für den Faktor `Jahreszeit` (mit entsprechend vier Mittelwerten und SS), so erhalten wir $\bar{x}_F = 11.86$, $\bar{x}_S = 15.09$, $\bar{x}_H = 19.23$, $\bar{x}_W = 23.42$, $\mathrm{SS}_F = 29.22$, $\mathrm{SS}_S = 33.87$, $\mathrm{SS}_H = 45.36$ und $\mathrm{SS}_W = 94.48$. Aufsummiert sind dies 202.93 unerklärte SS für die Residuen. Entsprechend erklärte der Faktor `Jahreszeit` (959 − 203 =) 756 von 959.

Abbildung 6.5 macht deutlich, um wieviel besser die Mittelwerte je Jahreszeit die Datenpunkte annähern als der Gesamtmittelwert.

Wir berechnen die SS_{Faktor}, indem wir einfach die Abweichungen der Gruppenmittel vom Gesamtmittel berechnen. Dies ist ja gerade der Faktoreffekt. $SS_{Unterart} = ((\bar{x} - \bar{x}_C)^2 + (\bar{x} - \bar{x}_S)^2) \cdot 20 = 107$ Wir multiplizieren mit 20, da der Mittelwert der Gruppen auf 20 Messwerten beruht.[11] Dies ist der gleiche Wert, den wir weiter oben als Differenz zwischen den Gesamtabweichungsquadraten (SS_{total}) und den Abweichungsquadraten der Residuen für `Unterart` ($SS_{Unterart}$) berechnet haben. Welchen Weg wir beschreiten ist gleichgültig.

Aus diesen Werten können wir die F-Statistik für `Unterart` berechnen:

$$F = \frac{SS_{\text{Unterart}}/df_{\text{Unterart}}}{SS_{\text{Residuen}}/df_{\text{Residuen}}} = \frac{107/1}{(959-107)/(40-1-1)} = \frac{107}{22.42} = 4.77$$

Der assoziiert P-Wert ist 0.035.

Entsprechende Berechnungen für (vier!) `Jahreszeit` sehen so aus:

$$F = \frac{SS_{\text{Jahreszeit}}/df_{\text{Jahreszeit}}}{SS_{\text{Residuen}}/df_{\text{Residuen}}} = \frac{756/3}{(959-756)/(40-3-1)} = \frac{252}{5.34} = 47.17$$

Der assoziiert P-Wert ist viel kleiner als 0.001:

[11] Bei nur zwei Kategorien eines Faktors sind die Abweichungen der Gruppenmittel vom Gesamtmittelwert jeder Gruppe gleich. Deshalb könnten wir auch diesen Wert in diesem Fall auch nur für eine Gruppe berechnen, und mit 40 multiplizieren. Bei drei oder mehr Gruppen ist dies natürlich nicht mehr möglich.

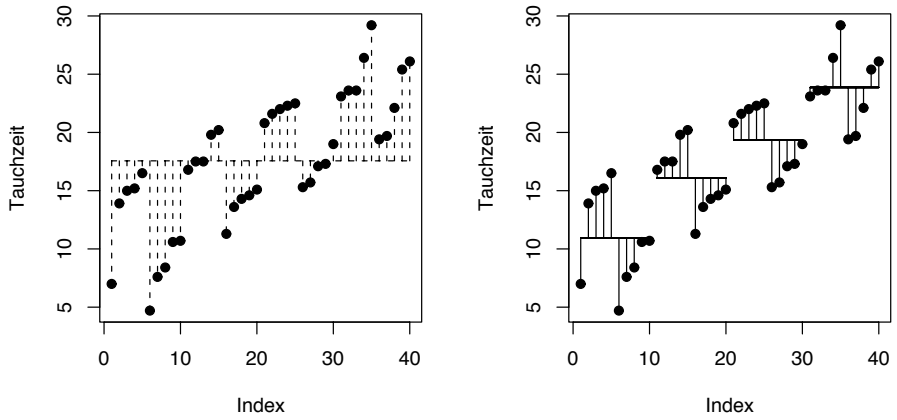


Abbildung 6.5.: Abweichung der Datenpunkte vom Gesamtmittelwert (links) und von den Gruppenmittelwerten (rechts). Diese Abweichungen quadriert sind die $\mathrm{SS}_{\mathrm{Jahreszeit}}$. Daten wie Abbildung 6.4.

```
> pf(47.17, 1, 38, lower.tail = F)

[1] 3.711755e-08
```

Was wir hier berechnet haben sind die Einzeleffekte der Faktoren `Unterart` und `Jahreszeit`. Unser Experiment war aber so angelegt, dass beide manipuliert wurden, und wir deshalb eine kombinierte Analyse vornehmen können. Dies hat zur Folge, dass sich die Werte für die Residuen stark reduzieren:

Zusammen addieren sich die SS_{Faktor} unserer beiden Faktoren `Unterart` und `Jahreszeit` also zu $107+756 = 863$. Somit bleiben für die Residuen nur noch $959-863 = 96$ übrig. Da wir jetzt die F-Statistik neu berechnen, nämlich auf Grundlage dieses viel kleineren Wertes für SS_{res}, werden unsere Faktoren noch stärker signifikant. Zu berücksichtigen ist dabei allerdings, dass sich die Freiheitsgrade für die Residuen jetzt auf das Gesamtmodell beziehen, nicht alleine auf einen Faktor. Statt $40-1-1 = 38$ für die Residuen von `Unterart` haben wir jetzt $40-1-3-1 = 35$ für beide Faktoren:

$$F = \frac{SS_{\mathrm{Unterart}}/df_{\mathrm{Unterart}}}{SS_{\mathrm{res}}/df_{\mathrm{res}}} = \frac{107/1}{96/(40-4-1)} = \frac{107}{2.74} = 39.0$$

Der assoziiert P-Wert ist viel kleiner als 0.001.

Dito für `Jahreszeit`:

$$F = \frac{SS_{\mathrm{Jahreszeit}}/df_{\mathrm{Jahreszeit}}}{SS_{\mathrm{res}}/df_{\mathrm{res}}} = \frac{756/3}{96/(40-4-1)} = \frac{252}{2.74} = 92.0$$

Der assoziiert P-Wert ist ebenfalls viel kleiner als 0.001.

Wir sehen also, dass aufgrund der Kombination beider Faktoren in einer Analyse die Erklärungskraft unseres statistischen Modells dramatisch zunimmt, und beide Faktoren jetzt hochsignifikant sind.

Für das additive ANOVA-Modell (also mit `Unterart` und `Jahreszeit`, aber ohne Interaktionen) können wir jetzt eine ANOVA-Tabelle konstruieren:

Effekt	SS	df	MS	F	P
Unterart	107	1	107	39.0	<0.0001
Jahreszeit	756	3	252	92.0	<0.0001
Residuen	96	35	2.7		
Gesamt	959	40			

Schließlich können wir auch die Interaktion auf diese Weise berechnen, indem wir Gruppen bilden, in der sowohl `Unterart` als auch `Jahreszeit` spezifiziert sind ($2 \cdot 4 = 8$ Gruppen). Dies bringt nur eine geringe (nicht signifikante) Reduzierung der SS_{res} gegenüber dem additiven Modell: 175 gegenüber 185.

Diese Berechnungsweise können wir auf weit mehr Faktoren ausweiten. Spätestens wenn signifikante Interaktionen dazukommen wird jedoch die Interpretierbarkeit gefährdet. Dieses Beispiel soll illustrieren, dass hinter der ANOVA auch keine Magie steckt, sondern dass sich diese Berechnungen auch mit dem Taschenrechner durchführen lassen.

Die ANOVA ist in R durch die Funktion `aov` implementiert[12]. Wir können aber auch aus dem linearen Modell eine ANOVA-Tabelle konstruieren, wie im letzten Beispiel gezeigt. Der Syntax von `aov` ist analog zu `lm`.

```
> summary(aov(Tauchzeit ~ Unterart + Jahreszeit))

            Df Sum Sq Mean Sq F value    Pr(>F)
Unterart     1 106.93  106.93  38.984 3.691e-07 ***
Jahreszeit   3 756.17  252.06  91.895 < 2.2e-16 ***
Residuals   35  96.00    2.74
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> detach(kormoran)
```

6.5. Kontinuierliche und kategoriale erklärende Variablen

Nachdem wir zunächst mehrere kontinuierliche erklärende Variablen, dann mehrere kategorische erklärende Variablen betrachtet haben, wenden wir uns nun deren Kombination zu. Ob wir dies mittels eines linearen Modells oder durch die ANOVA tun, ist bezüglich des Ergebnisses irrelevant (siehe die letzten beiden Abschnitte. In beiden Fällen liegt den Berechnungen das allgemeine lineare Modell zugrunde, dessen mathematische Ausarbeitung in einem späteren Abschnitt dargelegt wird (Abschnitt 6.6, Seite 117).

In der ANOVA bezeichnet man kontinuierliche erklärende Variablen als *Kovariablen* (und aus der ANOVA wird dann eine AN**C**OVA). Dies liegt daran, dass ANOVA in den 80er Jahren vor allem für die Analyse manipulativer Experimente benutzt wurde. Die manipulierten Faktoren waren nahezu ausschließlich kategorial (mit und ohne Dünger, Herbivoren oder Konkurrenz). Variablen, die etwa die Versuchseinheiten näher beschrieben, ohne Teil der Manipulation zu sein (etwa Alter, pH) wurden dann in die Analyse als möglicherweise erklärende, aber eigentlich nicht im zentralen Interesse stehende Ko-Variablen mit untersucht. Inzwischen sind die Experimente bisweilen komplexer (10 Düngestufen, Untersuchung von Alterseffekten) und die Ko-Variablen sind ebenfalls zum Gegenstand der Forschung geworden. Nur aus nostalgischen Gründen hängen wir noch immer an der Bezeichnung AN**C**OVA.

Nehmen wir als Beispiel zwei erklärende Variablen, eine kategorial, die andere kontinuierlich. Dann können wir die im Abschnitt 6.4.2 durchgeführten Berechnungen sowohl für die eine als auch für die andere Variable durchführen. Dabei unterscheiden sich nur die Art und Weise, wie wir die vorhergesagten Werte berechnen. Bei der kontinuierlichen geschieht dies identisch zur Regression (also über die Regressionsgeradengleichung), und bei der kategorialen (wie oben vorgeführt) über den Mittelwert der Gruppen. Auf eine expliziete mathematische Formulierung sei hier verzichtet, da sie sich nicht von den obigen unterscheidet. Wer trotzdem gerne die Herleitung jedes einzelnen SS-Wertes vorgerechnet bekommen möchte, sei an Underwood (1997) verwiesen.

[12] Zur Kompatibilität mit späteren GLMs könne wir auch `anova(lm(.))` benutzen, wobei die Ergebnistabelle ein leicht verändertes Layout aufweist.

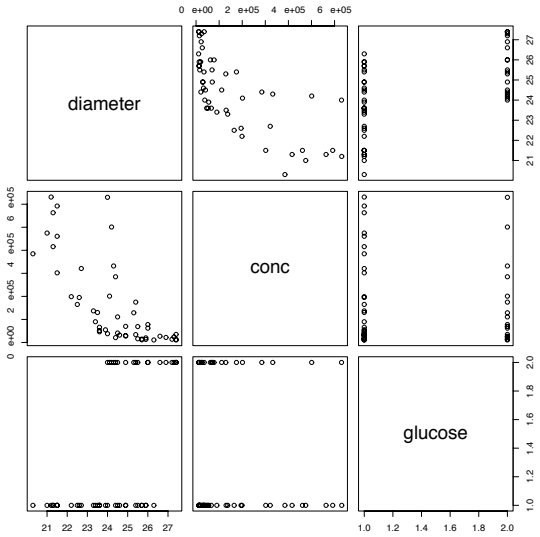


Abbildung 6.6.: Scatterplot der Variablen Zelldurchmesser, Zellendichte und Glukosezusatz. Wir sehen einen deutlichen nicht-lineare Zusammenhang zwischen Zelldurchmesser und Zellendichte. Der Effekt von Glukose ist ebenfalls erahnbar.

6.5.1. Interaktionen und ihre Interpretation

In den vorigen Beispielen waren die Interaktionen nicht signifikant. Um Interaktionen zwischen Variablen verstehen zu können, sollten wir uns erst einmal ein entsprechendes Beispiel anschauen.

Dieses Beispiel kommt aus Dalgaard (2002, etwas modifiziert!). In einem Versuch wurde der Durchmesser von *Tetrahymena*-Zellen (ein Süßwasserziliat) in Kulturen unterschiedlicher Zellendichte und mit/ohne Glukosezusatz gemessen. Zunächst laden wir die Daten ein und plotten alle Variablen gegeneinander.

```
> ancova <- read.table("ancova.data.txt", header = T)
> attach(ancova)
> plot(ancova)
```

Wir sehen (Abb. 6.6), dass der Zusammenhang zwischen `diameter` und `conc` nicht-linear ist, und untersuchen deshalb im Weiteren logarithmierte Daten. Den Faktor mit oder ohne Glukose benennen wir entsprechend. Dann schauen wir uns die Daten für beide erklärenden Variablen zusammen in einer Grafik an (Abbildung 6.7).

```
> glucose <- factor(glucose, labels = c("Ja", "Nein"))
> plot(diameter ~ log10(conc), pch = as.numeric(glucose))
> legend(4.2, 22, legend = c("Glukose", "keine Glukose"), pch = 2:1,
+     bty = "n")
> abline(lm(diameter ~ log10(conc)))
```

Jetzt benutzen wir den Befehl `lm`, um für beide Glukosekonzentrationen unterschiedliche Regressionen fitten und in die Abbildung einzuzeichnen.

```
> tethym.gluc <- ancova[glucose == "Ja", ]
> tethym.nogluc <- ancova[glucose == "Nein", ]
> lm.nogluc <- lm(diameter ~ log10(conc), data = tethym.nogluc)
> lm.gluc <- lm(diameter ~ log10(conc), data = tethym.gluc)
> abline(lm.nogluc, lty = 2)
> abline(lm.gluc, lty = 2)
```

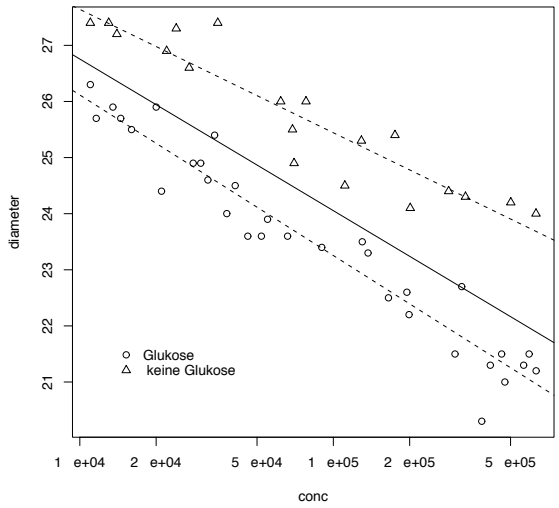


Abbildung 6.7.: Zelldurchmesser von *Tetrahymena* in Abhängigkeit von der logarithmierten Zellendichte, für Kulturen mit und ohne Glukose. Die durchgezogene Linie stellt die Regression für alle Datenpunkte dar, die gestrichelten für die beiden Glukose-Behandlungen separat.

Aber unterscheiden sich diese Regressionen denn? Zur Beantwortung dieser Frage betrachten wir die Interaktion zwischen `log10(conc)` und `glucose`:

```
> fm3 <- aov(diameter ~ log10(conc) * glucose)
> summary(fm3)

                   Df  Sum Sq Mean Sq  F value  Pr(>F)
log10(conc)         1 115.217 115.217 530.7983 < 2e-16 ***
glucose             1  53.086  53.086 244.5652 < 2e-16 ***
log10(conc):glucose 1   1.548   1.548   7.1296 0.01038 *
Residuals          47  10.202   0.217
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In der Tat liegt hier eine signifikante Interaktion zwischen Zellenkonzentration und Glukosebehandlung vor. Lassen wir uns um die Koeffizienten zu erhalten, auch noch das lineare Modell ausgeben:

```
> summary(lm(fm3))

Call:
lm(formula = fm3)

Residuals:
    Min      1Q  Median      3Q     Max
-1.2794 -0.1912  0.0118  0.2656  0.9552

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)             37.5594     0.7138  52.618   <2e-16 ***
log10(conc)             -2.8610     0.1444 -19.820   <2e-16 ***
glucoseNein             -1.1224     1.2187  -0.921   0.3617
log10(conc):glucoseNein  0.6621     0.2479   2.670   0.0104 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*'0.05 '.' 0.1 ' ' 1
```

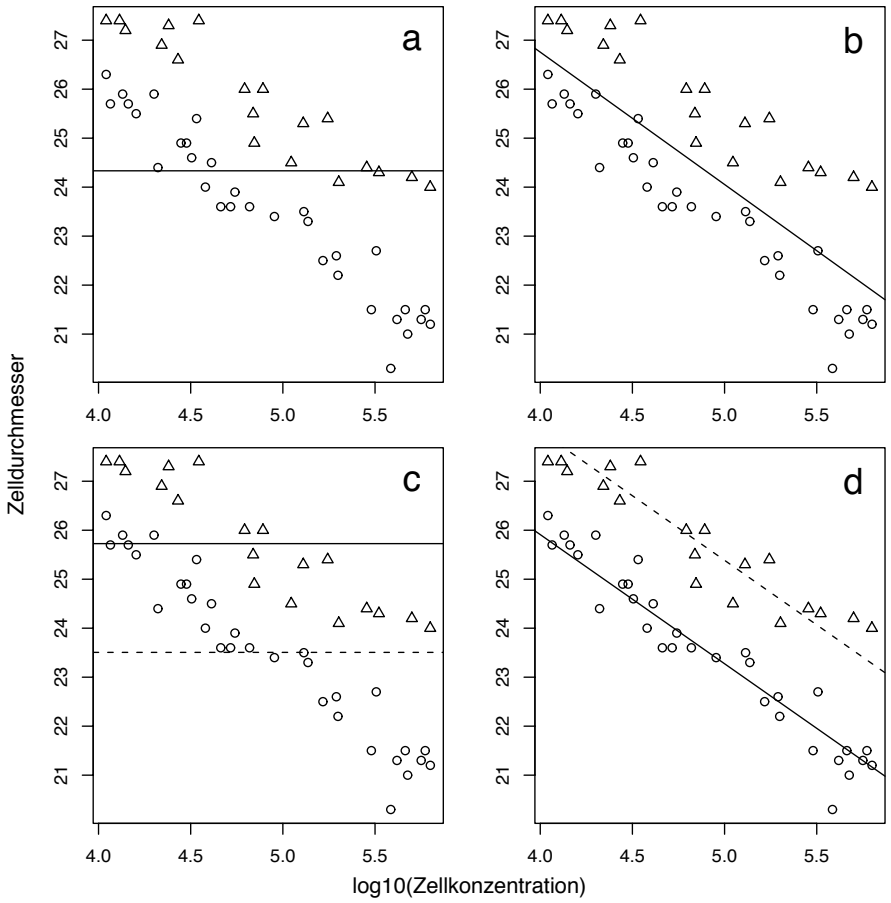


Abbildung 6.8.: Zelldurchmesser von *Tetrahymena* in Abhängigkeit von der logarithmierten Zellendichte, für Kulturen mit und ohne Glukose. Die Linien stellt die Regressionen für das jeweilige Modell dar: a) nur Achsenabschnitt, b) nur log10(Zellendichte), c) nur Glukoseeffekt, d) log10(Zellendichte) plus Glukoseeffekt. Die Interaktion von Zellendichte und Glukose ist in Abbildung 6.7 dargestellt. Gestrichelte Linie sind ohne, durchgezogene mit Glukose.

```
Residual standard error: 0.4659 on 47 degrees of freedom
Multiple R-Squared: 0.9433,     Adjusted R-squared: 0.9397
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Was bedeutet eine signifikante Interaktion zwischen einer kategorischen und einer kontinuierlichen erklärenden Variablen? Sie bedeutet schlicht, dass die Steigung der Ko-Variablen (d.h. der kontinuierlichen Variablen) für die Level der kategorischen Variablen signifikant unterschiedlich ist.

Schauen wir uns obige `lm`-Tabelle an: Zunächst haben wir einen signifikanten Achsenabschnitt (`(Intercept)`), der dadurch zustandekommt, dass die Werte für den Zelldurchmesser nicht um 0 zentriert sind (Abbildung 6.8a). Dies entspricht einer Regression in der lediglich ein Achsenabschnitt gefittet wird (Der `par(mfrow(...))`-Befehl lässt uns vier Graphiken in eine Abbildung bringen; mit dem `mtext`-Befehl erzeugen wir die Beschriftung der x- und y-Achsen.):

```
> par(mfrow = c(2, 2))
> bsp1 <- lm(diameter ~ 1)
> plot(diameter ~ log10(conc), pch = as.numeric(glucose), ylab = "")
```

```
> abline(h = coef(bsp1))
> text(5.7, 27, "a", cex = 2)
> mtext("Zelldurchmesser", side = 2, line = 0, outer = T)
> mtext("log10(Zellkonzentration)", side = 1, line = 0, outer = T)
```

Als nächstes sehen wir, dass der Effekt der Zellkonzentration signifikant ist (`log10(conc)`, Abbildung 6.8b). Also ist der Zelldurchmesser abhängig von der Zellkonzentration. Dies entspricht einem Modell mit Achsenabschnitt und Zellkonzentration:

```
> bsp2 <- lm(diameter ~ log10(conc))
> plot(diameter ~ log10(conc), pch = as.numeric(glucose), ylab = "")
> abline(bsp2)
> text(5.7, 27, "b", cex = 2)
```

Der Effekt der Glukose ist ebenfalls signifikant (`glucoseNein`, Abbildung 6.8c)). Dies bedeutet, dass Zellen in glukosehaltigen Kulturen nicht signifikant größer werden. Unser Modell enthält nun neben dem Achsenabschnitt noch den Glukoseeffekt:

```
> bsp3 <- lm(diameter ~ glucose)
> plot(diameter ~ log10(conc), pch = as.numeric(glucose), ylab = "")
> abline(h = coef(bsp3)[1] + coef(bsp3)[2])
> abline(h = coef(bsp3)[1], lty = 2)
> text(5.7, 27, "c", cex = 2)
```

Im nächsten Schritt fügen wir sowohl den Zellendichte-Term als auch den Glukoseeffekt ein, und zwar additiv:

```
> bsp4 <- lm(diameter ~ log10(conc) + glucose)
> plot(diameter ~ log10(conc), pch = as.numeric(glucose), ylab = "")
> abline(a = bsp4$coef[1], b = bsp4$coef[2], lty = 1)
> abline(a = (bsp4$coef[1] + bsp4$coef[3]), b = bsp4$coef[2], lty = 2)
> text(5.7, 27, "d", cex = 2)
> par(mfrow = c(1, 1))
```

Und schlussendlich ist die Interaktion signifikant, d.h. der Effekt der Zellendichte ist unterschiedlich in Kulturen mit und ohne Glukose (Abb. 6.7). Somit hat Glukose dann doch einen Effekt, aber dieser ist nur dann bemerkbar, wenn wir die Zellendichte berücksichtigen.

Anders formuliert stellt der Test der Glukosebehandlung einen Test auf unterschiedliche Achsenabschnitte der Zellendichte-Regressionsgeraden dar. Ist er signifikant, so sind die Regressionsgeraden parallel, aber nicht identisch (Abb. 6.8d). Der Interaktionseffekt hingegen stellt einen Test auf Unterschiedlichkeit der Steigung der Regressiongeraden dar. Ist er signifikant, so haben zwar beide den gleichen Achsenabschnitt, aber unterschiedliche Steigungen. Sind sowohl die kategoriale Variable als auch die Interaktion signifikant, so liegen sowohl unterschiedliche Achsenabschnitte und Steigungen vor (Abb. 6.7).

Schlussendlich wollen wir aber auch noch die Geradengleichungen für diese Regressionen aus der Tabelle ablesen können. Wie geht das? Zunächst erinnern wir uns, dass die Geradengleichung aus einem y-Achsenabschnitt und der Steigung besteht. Desweiteren vergegenwärtigen wir uns nochmal, dass der Effekt `glucoseNein` und `log10(conc):glucoseNein` in der oberen `summary` den *Unterschied* zum Level `glucoseJa` bzw. `log10(conc)` darstellen. Dann ergibt sich die Geradengleichung für die `glucoseJa`-Regression als:

$$(Intercept) + est_{log10(conc)} \cdot log10(conc) = 37.6 - 2.9 \cdot log10(conc)$$

Für die `glucoseJa`-Regression werden Achsenabschnitt und Steigung um den Schätzwert `glucoseNein` bzw. `log10(conc):glucoseNein` verändert:

$$\begin{aligned}((Intercept) + est_{glucoseNein}) + (est_{log10(conc)} + est_{log10(conc):glucoseNein}) \cdot log10(conc)\\ = (37.6 - 1.1) + (-2.9 + 0.7) \cdot log10(conc)\end{aligned}$$

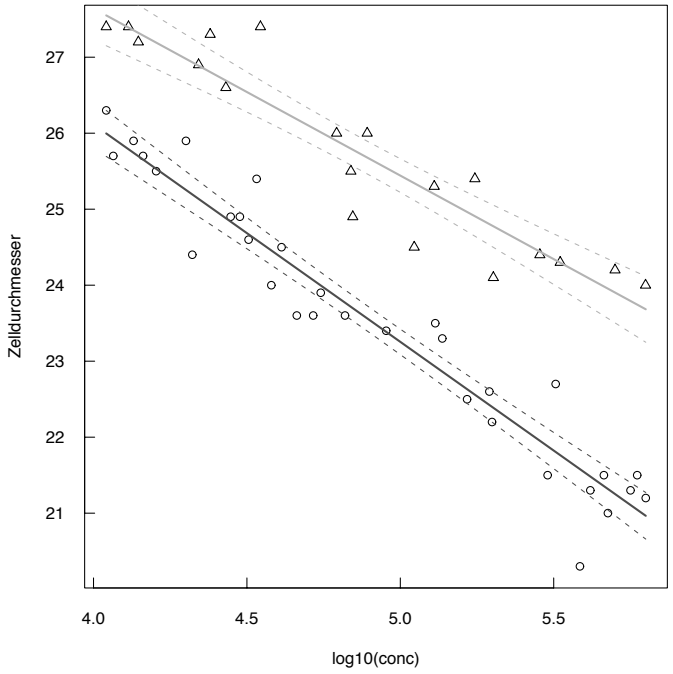


Abbildung 6.9.: Zelldurchmesser von *Tetrahymena* in Abhängigkeit von der logarithmierten Zellendichte, für Kulturen mit (Dreiecke) und ohne Glukose (Punkte). Die Linien stellt die Regressionen für das jeweilige Modell dar, jeweils mit 95%-Konfidenzinterval.

Mit Hilfe dieser Logik haben wir ja auch gerade die Geradengleichung in die Abbildung eingezeichnet.

Während Abbildung 6.7 nützlich ist, um zu verstehen, was die unterschiedlichen Modelle bedeuten, und was eine Interaktion ist, widerspricht es doch der guten Praxis, Regressionslinien über den Wertebereich der Datenpunkte hinaus zu verlängern. Außerdem wäre sinnvoll, auch die Konfidenzintervalle für diese Regression abbilden zu können. Da die Mathematik hinter der Berechnung der Konfidenzintervalle hier zu weit führen würde, beschränken wir uns auf den R-code für das Einfügen einer Linie über den Wertebereich sowie deren Konfidenzintervalle[13].

Wir beginnen mit dem Plot der Datenpunkte, benutzen dann das Modell mit der Interaktion (`fm3`), um die Regressionsgeraden und ihre Konfidenzintervalle zu plotten. Dazu berechnen wir für neue Datenpunkte im Wertebereich den Vorhersagewert nebst Konfidenzinterval. Mit der Funktion `matlines` können wir dann mehrere Linien auf einmal in die Abbildung legen. Diese Prozedur machen wir einmal für die eine Gerade, dann nochmals für die andere. Die (rein optische) Wirkung der Optionen `las` und `tcl` kann man unter `?par` nachlesen!

```
> plot(diameter ~ log10(conc), pch = as.numeric(glucose), ylab="Zelldurchmesser",
+ las=1, tcl=0.5)
> newconc <- seq(min(conc), max(conc), len=50)
> # Ja-Gerade:
> newdat.Ja <- data.frame("conc"=newconc, glucose=c("Ja"))
> pred.Ja <- predict(fm3, newdata=newdat.Ja, interval="confidence")
> matlines(log10(newdat.Ja$conc), pred.Ja, lty=c(1,2,2), lwd=c(2,1,1), col="grey30")
> # Nein-Gerade:
```

[13] Es gibt unterschiedliche Konfidenzintervalle! Wir beschränken uns hier auf den Konfidenzbereich, mit dem die Daten die Schätzung der Regressionsgerade zulassen (Option `interval='confidence'`). Darüberhinaus gibt es ein (größeres) Konfidenzintervall, dass den Bereich angibt, in den mit 95%-iger Wahrscheinlichkeit *neue* Werte fallen würden (Option `interval='prediction'`)(Abb. 6.9).

```
> newdat.Nein <- data.frame("conc"=newconc, glucose=c("Nein"))
> pred.Nein <- predict(fm3, newdata=newdat.Nein, interval="confidence")
> matlines(log10(newdat.Nein$conc), pred.Nein, lty=c(1,2,2), lwd=c(2,1,1),
+ col="grey70")
```

6.6. Die Mathematik hinter dem linearen Modell

Die Berechnungen des linearen Modells erfolgen mittels algebraischer Methoden, in diesem Fall Matrixoperationen. Stellen wir uns ein lineares Modell (unabhängig davon, ob die erklärenden Variablen kategorisch oder kontinuierlich sind) vor, das aus drei Teilen besteht: (1) ein Vektor[14] $\mathbf{Y}$, der die n gemessenen Werte der abhängigen Variablen darstellt:

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix};$$

(2) mehreren (p) Vektoren erklärender Variablen $X_1 \cdots X_p$, jeder mit ebenfalls n Werten, arrangiert als Matrix $\mathbf{X}$ (diese hat $p+1$ Spalten, die erste ($x_{\cdot 0}$) für den Achsenabschnitt):

$$\mathbf{X} = \begin{pmatrix} x_{1\,0} & x_{1\,1} & x_{1\,2} & \cdots & x_{1\,p} \\ x_{2\,0} & x_{2\,1} & x_{2\,2} & \cdots & x_{2\,p} \\ \vdots & & & \ddots & \vdots \\ x_{n\,0} & x_{n\,1} & x_{n\,2} & \cdots & x_{n\,p} \end{pmatrix};$$

und (3) einem Vektor β mit den Koeffizienten des Modells, das $\mathbf{Y}$ und $\mathbf{X}$ verbindet:

$$\beta = \begin{pmatrix} \beta_1 & \beta_2 & \ldots & \beta_p \end{pmatrix}^T.$$

Somit stellt sich das lineare Modell dar als: $\mathbf{Y} = \mathbf{X}\beta + \epsilon$, wobei ϵ ein Fehlervektor ist, mit der gleichen Struktur wie $\mathbf{Y}$.

Aus dieser Gleichung lassen sich die Koeffizienten (β_i) des Modells berechnen, durch Lösen der sogenannten Normalgleichung:

$$\mathbf{X}^T\mathbf{X}\mathbf{b} = \mathbf{X}^T\mathbf{Y},$$

wobei $\mathbf{b}$ der *ordinary least square*-Schätzer für β und Vektor der partiellen Regressionskoeffizienten ist. $\mathbf{X}^T$ ist die transponierte (um 90° gedrehte) Matrix $\mathbf{X}$.

Es folgt, dass

$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{Y}),$$

wobei $\mathbf{X}^{-1}$ die inverse Matrix von $\mathbf{X}$ ist. Dabei gilt (und definiert) $\mathbf{X}\mathbf{X}^{-1} = \mathbf{I}$, und $\mathbf{I}$, die Einheitsmatrix, hat die Werte 1 entlang der Diagonalen, und 0 sonst.

Die Varianz der partiellen Regressionskoeffizienten, sowie die Kovarianz zwischen den erklärenden Variablen lässt sich ebenfalls berechnen als:

$$s^2 = MS_{\text{Residuen}}(\mathbf{X}^T\mathbf{X})^{-1}.$$

Eine weitere nützliche Berechnung greift auf die sogenannte *hat*-Matrix $\mathbf{H}$ zurück (leider gibt es keine deutsche Übersetzung; „Dach-Matrix“ wäre angemessen, ist aber ungebräuchlich). $\mathbf{H}$ ist eine $n \times n$-Matrix. Es ist

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T.$$

[14] Vektoren und Matrizen werden entweder durch Pfeile über dem Symbol ($\vec{X}$), oder durch Fettdruck dargestellt ($\mathbf{X}$). Wir folgenden der letzten Symbolik.

H enthält als Diagonale die Einflusswerte der einzelnen Datenpunkte auf die Regression. Zudem, und das ist ihre Hauptfunktion, können wir mit ihr aus den Datenpunkten $\mathbf{Y}$ die vorhergesagten Werte $\widehat{\mathbf{Y}}$ (gesprochen: Ypsilon Dach, engl. *wai-hätt* berechnen:

$$\widehat{\mathbf{Y}} = \mathbf{HY}.$$

Und schließlich lassen sich auch die Effektquadrate der ANOVA aus diesen Matrizen berechnen. Dafür brauchen wir noch einen Korrekturfaktor $\mathbf{K} = \mathbf{Y}^T\mathbf{1}\mathbf{1}^T\mathbf{Y}/n$. Dabei ist $\mathbf{1}$ ein Vektor von Länge n, bei dem alle Einträge 1 sind.

Die Gesamtquadrate berechnen sich als:

$$SS_{\text{gesamt}} = \mathbf{Y}^T\mathbf{Y} - \mathbf{K}.$$

Die Quadrate der Residuen sind:

$$SS_{\text{resid}} = \mathbf{b}^T\mathbf{Y}^T\mathbf{Y} - \mathbf{K}.$$

Die Quadrate für die Effekte sind die Differenz:

$$SS_{\text{Effekt}} = \mathbf{Y}^T\mathbf{Y} - \mathbf{b}^T\mathbf{Y}^T\mathbf{Y}.$$

Wahrscheinlich werden wir dies nie auf diese Art selbst berechnen müssen. Trotzdem sei kurz ein Beispiel aus Crawley (2002) hier wiedergegeben. Es ist eine Regression von Holzvolumen gegen Umfang und Höhe der Bäume.

```
> Holz <- read.table("Holz.txt", header = T, dec = ",")
> attach(Holz)
```

Jetzt konstruieren wir **X**, indem wir den Achsenabschnitten durch den Wert 1 und die anderen erklärenden Variablenwerte zu einer Matrix zusammenfügen:

```
> X <- cbind(1, Umfang, Hoehe)
```

Dann berechnen wir die Inverse von **X**, **Xp** (wegen des Hochkommas, engl. *prime*).

```
> Xp <- t(X)
```

Zunächst haben wir dann die Matrix $\mathbf{X}^T\mathbf{X}$ (Matrixmultiplikationen müssen in Prozentzeichen gesetzt werden):

```
> Xp %*% X
```

```
                  Umfang      Hoehe
        31.0000  32.77230   706.8000
Umfang  32.7723  36.52706   754.6654
Hoehe  706.8000 754.66542 16224.6600
```

In der Diagonalen finden wir zunächst den Stichprobenumfang $n = 31$, dann die SS für Umfang und dann die SS für Höhe. Der zweite und dritte Wert in Spalte 1 ist jeweils die Summe der Werte für Umfang bzw. Höhe. Der noch fehlende Wert (unten Mitte bzw. rechts Mitte) ist die Summe der Produkte von Umfänge und Höhen. Wir rechnen kurz nach:

```
> sum(Umfang)
```

```
[1] 32.7723
```

```
> sum(Hoehe)
```

```
[1] 706.8
```

```
> sum(Umfang^2)

[1] 36.52706

> sum(Hoehe^2)

[1] 16224.66

> sum(Hoehe * Umfang)

[1] 754.6654
```

Als nächstes interessieren uns die Koeffizienten **b**:

```
> b <- solve(Xp %*% X) %*% Xp %*% Volumen
> b

              [,1]
       -4.19899732
Umfang  4.27251096
Hoehe   0.08188343
```

Wir vergleichen dies mit den Koeffizienten aus dem linearen Modell:

```
> lm(Volumen ~ Umfang + Hoehe)

Call:
lm(formula = Volumen ~ Umfang + Hoehe)

Coefficients:
(Intercept)       Umfang        Hoehe
   -4.19900      4.27251      0.08188
```

Nur der Vollständigkeit halber berechnen wir jetzt noch die $SS_{Residuen}$ des linearen Modells:

```
> t(Volumen) %*% Volumen - t(b) %*% Xp %*% Volumen

         [,1]
[1,] 2.212518

> detach(Holz)
```

Ob dies stimmt, kann man dann mittels `summary(aov(Volumen~Umfang+Hoehe))` nachprüfen ...

6.7. Post-hoc Vergleiche und Kontraste

Wenn eine Behandlung mehrere Level hat, dann wissen wir nach einer ANOVA noch nicht, welche dieser verschiedenen Level voneinander verschieden sind, selbst wenn die Behandlung selbst signifikant ist. Das gleiche Problem taucht auf, wenn wir eine signifikante Interaktion zwischen zwei kategorischen erklärenden Variablen haben. Welche der vier Kombinationen (bei zwei Leveln je Faktor) sind denn nun signifikant voneinander unterschiedlich?

An dieser Stelle können wir einen *post-hoc*-Vergleich durchführen (*post hoc*, da diese Tests erst nach dem eigentlichen Test auf Signifikanz der Effekte und Interaktionen durchgeführt werden.). Dabei werden die Werte der verschiedenen Level miteinander verglichen. Leider wird diese Situation durch mehrere Problem verkompliziert. Zum einen scheint es einen Glaubenskrieg zu geben zwischen den Befürwortern und den Ablehnern von *post-hoc*-Tests. Die

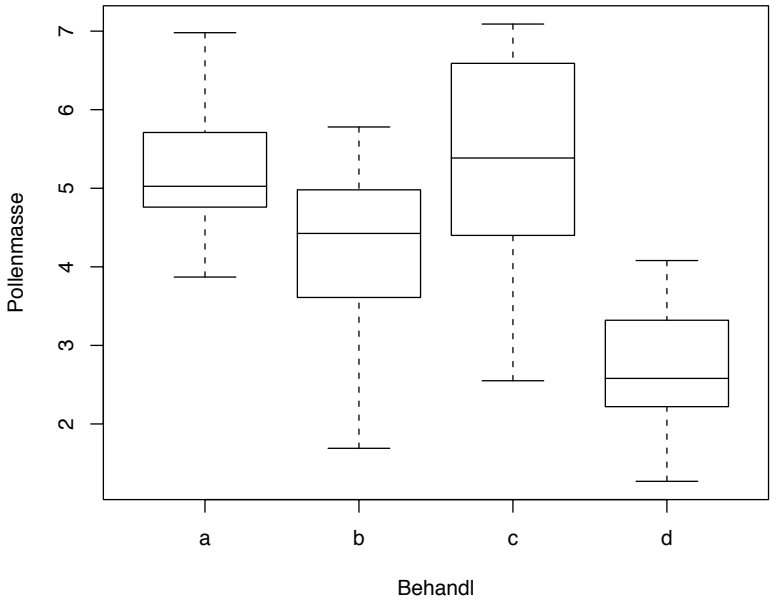


Abbildung 6.10.: Box-Whiskers-Plot der Pollenmasse [mg] in vier verschiedenen Bestäubungsbehandlungen (a-d).

Ablehner (unter ihnen die Hauptengagierten bei R, Bill Venables und Brian Ripley u.a.) argumentieren dass Anwender *post-hoc*-Test missbrauchen[15].

Weiterhin besteht das Problem der Inflation der Fehler 1. Ordnung (siehe Abschnitt 1.3.1, Seite 6). Dem kann man mit einer Bonferroni-Korrektur entgegentreten oder speziellen, konservativen *post-hoc*-Testverfahren (etwa Tukey, Duncan, Scheffé, usw.). Die Veröffentlichung von Day and Quinn (1989) beschäftigt sich ausschließlich mit diesen *post-hoc*-Tests, und empfiehlt in unterschiedlichen Situationen unterschiedliche Tests. In der Praxis haben sich jedoch zwei Gruppen von *post-hoc*-Tests etabliert: Erstens die unkorrigierten t-Test-basierten LSD-Tests (*least significant difference*), die nicht konservativ sind und der Fehlerinflation anheim fallen. Die zweite Gruppe beinhaltet konservative Tests, die robust gegenüber einer Fehlerinflation sind: die Bonferroni-Korrektur (des LSD-Tests), sowie Tukey's HSD-Test (*honest significant difference*). Alle weiteren *post-hoc*-Tests werden weitaus seltener gebraucht und sind im Ergebnis den erwähnten Test sehr ähnlich.

Schließlich taucht noch ein weiteres Problem auf, dass mit der Struktur der Daten zusammenhängt. Wenn wir einen komplizierten Versuchsaufbau haben, dann können wir den *post-hoc*-Tests nicht ohne weiteres diese Struktur vermitteln, und ein *post-hoc*-Test wird u.U. fehlerhaft Gruppenmittel vergleichen. Wir müssen also sicherstellen, dass die benutzte Software auch tatsächlich die *vorhergesagten* Gruppenmittel vergleicht, und nicht die tatsächlich gemessenen, da letztere in *nested* oder *split-plot*-Designs nicht für *blocking* und *subsampling*-Effekte korrigiert sind (siehe Abschnitt 13 für eine Erklärung der unterschiedlichen Designs).

Zur Illustration der verschiedenen post-hoc-Tests in R erfinden wir uns einfach einen einfachen Datensatz, etwa die Auswirkung von vier unterschiedlichen Bestäubungsarten (a-d) auf die Pollenmenge (in mg; siehe Abbildung 6.10).

[15] "I have also a feeling that providing software would encourage people to use these, and they are only appropriate in very special circumstances (including no data cleaning and no model selection, as well as close to normal iid errors)", Prof. Brian D. Ripley, 2002: `https://stat.ethz.ch/pipermail/r-help/2002-February/017318.html`

```
> set.seed(2)
> a <- round(rnorm(10, 5, 1), 2)
> b <- round(rnorm(10, 4, 1), 2)
> cc <- round(rnorm(10, 5, 1), 2)
> d <- round(rnorm(10, 3, 1), 2)
> posthoc <- data.frame(Behandl = rep(c("a", "b", "c", "d"), each = 10),
+     Pollenmasse = c(a, b, cc, d))
> attach(posthoc)
> plot(Pollenmasse ~ Behandl)
> fm <- aov(Pollenmasse ~ Behandl)
> summary(fm)

            Df Sum Sq Mean Sq F value    Pr(>F)
Behandl      3 43.689  14.563  11.396 2.114e-05 ***
Residuals   36 46.005   1.278
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Jetzt führen wir eine *post-hoc*-Test durch, indem die Gruppenmittelwerte und ihre Standardfehler mittels eines t-Test verglichen werden. Dabei vergleichen wir einen LSD (`p.adj="none"`) mit einem Bonferroni-korrigierten *post-hoc*-Test (`p.adj="bonf"`).

```
> pairwise.t.test(Pollenmasse, Behandl, p.adj = "none")

        Pairwise comparisons using t tests with pooled SD

data:  Pollenmasse and Behandl

  a       b     c
b 0.049   -     -
c 0.869   0.034 -
d 1.7e-05 0.006 1.0e-05

P value adjustment method: none

> pairwise.t.test(Pollenmasse, Behandl, p.adj = "bonf")

        Pairwise comparisons using t tests with pooled SD

data:  Pollenmasse and Behandl

  a       b       c
b 0.29409 -       -
c 1.00000 0.20424 -
d 0.00010 0.03607 6.2e-05

P value adjustment method: bonferroni
```

Wir sehen, dass in der Bonferroni-Korrektur die LSD-p-Werte einfach mit der Anzahl Vergleiche (6) multipliziert wurde. Signifikante Unterschiede findet der Bonferroni-korrigierte post-hoc-Test zwischen den Behandlungen `d-a`, `d-b` und `d-c`. Schauen wir uns jetzt noch Tukey's HSD an:

```
> TukeyHSD(fm)

  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = Pollenmasse ~ Behandl)
```

```
$Behandl
      diff        lwr        upr     p adj
b-a -1.030 -2.3915660  0.3315660 0.1934949
c-a  0.084 -1.2775660  1.4455660 0.9983371
d-a -2.506 -3.8675660 -1.1444340 0.0000976
c-b  1.114 -0.2475660  2.4755660 0.1416218
d-b -1.476 -2.8375660 -0.1144340 0.0293079
d-c -2.590 -3.9515660 -1.2284340 0.0000590
```

Dieser *output* ist anders organisiert. In der ersten Spalte werden alle Vergleiche aufgelistet, in der zweiten der absolute Unterschied der Mittelwerte, in der dritten und vierten die unteren und oberen 95%-Konfidenzintervalle (mit der Option `conf.level=0.99` können auch andere Konfidenzintervalle berechnet werden). Nur wenn beide Konfidenzintervalgrenzen oberhalb oder unterhalb von 0 liegen, ist der betrachtete Unterschied signifikant (hier also `d-a`, `d-b` und `d-c`). Somit ist das Testergebnis für Tukey und Bonferroni in diesem Fall identisch. Liebhaber von Konfidenzintervallen, und das scheinen die Mehrzahl der Statistiker zu sein, bevorzugen einen *output* wie beim TukeyHSD, Biologen sind hingegen oft nur an der Signifikanz eines Unterschiedes interessiert. [16]

6.7.1. Kontraste

Gelegentlich sind wir nur am Vergleich zweier Kombinationen oder Level interessiert („Kontrast"). Das kann sein, weil wir unser Experiment auf den Vergleich gerade dieser beiden Kombinationen ausgerichtet hatten (*a priori*), oder weil uns im Nachhinein zwei Kombinationen als besonders vergleichenswert erscheinen (*a posterori*). Der letztere Fall ist sozusagen „moralisch verwerflich", da dieser Vergleich nicht geplant war. Nichtsdestotrotz werten viele Wissenschaftler ihre Daten so aus. Auf jeden Fall müssen wir aber darauf achten, dass wir Kontraste nur für solche Vergleiche berechnen, die in der Varianzanalyse zuvor signifikant waren (also der Faktor mit den vielen Leveln war signifikant, oder die Interaktion, von der uns nur zwei Kombinationen interessieren, war signifikant). Wenn wir in der ANOVA keinen signifikanten Unterschied innerhalb eines Faktors gefunden haben, dann dürfen wir nachher auch nicht den größten und kleinsten Wert einfach miteinander durch Kontraste vergleichen!

Bei Kontrasten konstruieren wir eine sog. Kontrastmatrix. Diese gibt an, welcher Level mit welchem verglichen werden soll. Die Elemente der Matrix addieren sich dabei zu Null. Wollen wir also etwa nur Level `a` mit Level `c` vergleichen, so sieht unsere Matrix (hier nur ein Vektor) wie folgt aus: (1, 0, -1, 0). Bei zwei Faktoren (A und B) mit je zwei Leveln (+ und −), von denen wir nur A−B− mit A+B− vergleichen wollen sieht die Matrix dann etwa so aus: $\begin{pmatrix} 0 & 1 \\ 0 & -1 \end{pmatrix}$. Wir können aber auch Level zusammen (`a` und `b`) gegen einen anderen Level (`c`) testen, also etwa $(1, 1, -2, 0)$. Die Level mit dem gleichen Vorzeichen werden dann zusammengefasst. Damit sich das ganze zu Null addiert erhält der einzelne Level dann eine -2 (alternativ: $(0.5, 0.5, -1, 0)$).

In R lassen sich Kontraste leicht spezifizieren. Zudem erlaubt R mehrfach Kontraste durchzuführen, die dann wie Faktoren in einer multiplen Regression in einem gemeinsamen Modell verrechnet werden. Dies hat zur Folge, dass ein Kontrast, der alleine nicht signifikant ist, gemeinsam mit anderen Kontrasten wohl signifikant sein kann. Ein solches Beispiel ist hier kurz aufgeführt. Dafür greifen wir auf die Pollenmassen des vorherigen Beispiels zurück. Vergleichen wir zunächst `a` und `c` mit `b`. Anstelle des normalen `summary`-Befehls benutzen wir hier `summary.lm`, um uns die Koeffizienten mit angeben zu lassen.

[16] Eine gute Übersicht über *post-hoc*-Vergleiche gibt Werner (1997). Leider sind nicht alle *post-hoc*-Verfahren in R implementiert. Neben den erwähnten gibt es standartmäßig drei weitere, eher unbekannte Verfahren die p-Werte zu korrigieren (siehe `?p.adjust`, zur Anwendung in der Funktion `pairwise.t.test`). Im *package* **multicomp** sind dann weitere neun *post-hoc*-Verfahren verfügbar, von denen die bekanntesten Dunnett und Tukey sind. Mit der Funktion `simtest` können `glm`-Ergebnisse in *post-hoc*-Verfahren analysiert werden. Siehe `?simtest` für weitere Details.

```
> Beha <- Behandl
> contrasts(Beha, how.many = 1) <- c(-1, 2, -1, 0)
> summary.lm(aov(Pollenmasse ~ Beha))


Call:
aov(formula = Pollenmasse ~ Beha)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0770 -0.9350  0.1857  1.0257  2.3857

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.3470     0.2323  18.712   <2e-16 ***
Beha1        -0.3573     0.1897  -1.884   0.0673 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.469 on 38 degrees of freedom
Multiple R-Squared: 0.08542,        Adjusted R-squared: 0.06135
F-statistic: 3.549 on 1 and 38 DF,  p-value: 0.06725
```

Von dem ganzen *output* interessiert uns eigentlich nur der Kontrast (namens `Beha1`): er ist gerade nicht signifikant. Der Wert des Achsenabschnitts ist der Mittelwert der drei beteiligten Level. Der Kontrast weicht davon um 0.36 ab, aber dieser Unterschied ist eben nicht ganz signifikant.
Vergleichen wir dies mit zweifachen Kontrasten (`a` und `c` gegen `b`, und `b` gegen `d`):

```
> contrasts(Beha, how.many = 2) <- cbind(c(-1, 2, -1, 0), c(0,
+     1, 0, -1))
> summary.lm(aov(Pollenmasse ~ Beha))


Call:
aov(formula = Pollenmasse ~ Beha)

Residuals:
    Min      1Q  Median      3Q     Max
-2.7020 -0.5115 -0.0280  0.7250  1.8380

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.3470     0.1764  24.646  < 2e-16 ***
Beha1        -0.9050     0.1764  -5.131 9.40e-06 ***
Beha2         1.6430     0.3055   5.378 4.36e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.115 on 37 degrees of freedom
Multiple R-Squared: 0.4867,        Adjusted R-squared: 0.459
F-statistic: 17.54 on 2 and 37 DF,  p-value: 4.384e-06
```

Dass jetzt der ursprünglich nicht-signifikante Kontrast signifikant geworden ist liegt daran, dass der vierte Level (`d`) mit ins Gesamtmodell genommen wurde, durch den zweiten Kontrast (`Beha2`) erklärt wird, und entsprechend weniger Varianz für die Residuen übrig ist. Beachte, dass das Gesamtmodell jetzt 46% der Varianz erklärt, gegenüber vorher nur 6%.

6.7.2. Pooling of levels

Unterscheiden sich zwei oder mehr Level eines Faktors nicht untereinander, aber wohl von einem weiteren Level, so kann man die Level vereinigen („*poolen*“). Dies führt zu einer Vereinfachung des Modells, da ja nun weniger Freiheitsgrade für den Effekt und mehr für die Residuen zur Verfügung stehen.

Die Möglichkeit Level zu poolen wird in der biologischen Literatur überraschend wenig genutzt. Zumeist liegt dies daran, dass wir wissen wollen, welche der verschiedenen Level sich unterscheiden (etwa welche Populationen einer Vogelart hinsichtlich ihrer Gelegegröße heraussticht). Dann ist mit dem *post-hoc*-Test schon alle Information gewonnen, und ein Poolen ist nicht mehr nötig. Wenn wir allerdings Dutzende von Genlinien hinsichtlich einer Funktion vergleichen, so könnten wir sicherlich gelegentlich diese Methodik verwenden.

Im obigen Beispiel sahen wir, dass ein Bonferroni-korrigierter *post-hoc*-Test keine Unterschiede zwischen a, b und c finden konnte. In der Abbildung sehen Behandlung a und c sehr ähnlich aus (Abb. 6.10 auf Seite 120). Wir können also erst einmal diese beiden Level zu einem kollabieren, die ANOVA wiederholen, und das alte und neu Modell vergleichen.

```
> Behandl.neu <- factor(1 + (Behandl == "b" | Behandl == "d") +
+     (Behandl == "d"))
```

Dieser umständlich wirkende Befehl evaluiert die Aussagen (`Behandl=="b"`) usw., und ergibt bei Zutreffen eine 1, sonst eine 0. Der senkrechte Strich (|) bedeutet „oder“. Damit erhalten a und c den Wert 1, b den Wert 2 und d den Wert 3.

```
> fm2 <- aov(Pollenmasse ~ Behandl.neu)
> summary(fm2)

            Df Sum Sq Mean Sq F value    Pr(>F)
Behandl.neu  2 43.654  21.827  17.541 4.384e-06 ***
Residuals   37 46.040   1.244
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> anova(fm, fm2)

Analysis of Variance Table

Model 1: Pollenmasse ~ Behandl
Model 2: Pollenmasse ~ Behandl.neu
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     36 46.005
2     37 46.040 -1    -0.035 0.0276  0.869
```

Der Vergleich ergibt keinen signifikanten Unterschied zwischen den Modellen. Also ist eine Zusammenlegung der Level a und c gerechtfertigt: Ihre Unterscheidung birgt keine zusätzliche Varianz. Fügen wir jetzt noch den Level b hinzu:

```
> Behandl.neu2 <- factor(1 + (Behandl == "d"))
> fm3 <- aov(Pollenmasse ~ Behandl.neu2)
> summary(fm3)

             Df Sum Sq Mean Sq F value    Pr(>F)
Behandl.neu2  1 35.993  35.993  25.469 1.147e-05 ***
Residuals    38 53.701   1.413
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> anova(fm2, fm3)

Analysis of Variance Table

Model 1: Pollenmasse ~ Behandl.neu
Model 2: Pollenmasse ~ Behandl.neu2
  Res.Df    RSS Df Sum of Sq      F  Pr(>F)
1     37 46.040
2     38 53.701 -1    -7.661 6.1569 0.01776 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Diesmal verändert sich das Modell signifikant. Das bedeutet, dass wir Level `b` nicht mit `a` und `c` vereinigen dürfen, ohne dass unser Modell signifikant an Erklärungskraft einbüßt. Somit bleibt `fm2` unser bestes Modell. Dies spiegelt sich meist auch in den R^2-Werten der Modelle wieder:

```
> summary(lm(fm2))$adj.r.squared

[1] 0.4589516

> summary(lm(fm3))$adj.r.squared

[1] 0.3855268

> detach(posthoc)
```

Und in der Tat ist auch hier Model `fm2` um 7% erklärte Varianz besser als `fm3`.

7. Lineare Gemischte Modelle (LMM)

7.1. Feste und zufällige Faktoren

Bisher haben wir in unserem statistischen Modell nur solche Variablen gehabt, von denen wir einen Zusammenhang mit den Antwortvariablen vermutet haben. Diese Faktoren nennt man **feste Faktoren** (*fixed effects*). Gerade bei experimentellen Studien gibt es aber noch eine weitere Gruppe Faktoren, die auch Varianz im Datensatz erklären können, aber nicht von primärem Interesse für die Untersuchung sind, sogenannte **zufällige Faktoren** (*random effects*). Dies können beispielsweise der Name eines Versuchstieres, die Nummer eines experimentellen Blocks oder ähnliches sein. Wenn wir ein Versuchstier mehrfach innerhalb eines Experimentes verwenden, dann birgt das Wissen, welches Tier einen bestimmtes Ergebnis geliefert hat natürlich Information: Die Schnecke Paul hat beispielsweise unterdurchschnittlichen Hunger während einer Reihe von Palatabilitätstests, und zieht so den Mittelwert nach unten. Hier wäre also der Tiername ein Zufallseffekt, den wir mit in unser Modell hineinnehmen sollten.

Die Unterscheidung in feste und zufällige Faktoren bereitet erfahrungsgemäß größere Schwierigkeiten. Deshalb hier noch ein paar weitergehende Fragen, die dabei helfen sollen:

- Interessiere ich mich für diesen Faktor? Nein → ZF (Testindividuum).
- Sind die Level dieses Faktors informationslose Nummern oder Buchstaben? Ja → ZF (Testname).
- Habe ich die Level dieses Faktors zufällig aus einer größeren Grundgesamtheit gezogen? Ja → ZF (Ein paar Schnecken zufällig aus einer Population, da ja jede Schnecke gut genug wäre.).
- Schleppe ich diesen Faktor nur mit, damit ich mein statistisches Modell vernünftig beschreiben kann? Ja → ZF (Block oder *subsample*).

Modelle, die sowohl feste als auch zufällige Faktoren beinhalten, nennen wir **gemischte Modelle** (*mixed models, mixed effect models*). Im wesentlichen unterscheiden wir hier zwei Formen der gemischten Modelle, nämlich (1) *split-plot* ANOVA und (2) *nested* ANOVA.

7.2. Split-plot ANOVA

Dies ist die klassische Form der notgedrungenen Applikation von Behandlungen auf unterschiedlichen Flächengrößen, z.B. Dünger auf Feldgröße, aber Mäusekäfige nur auf 1 m^2. Aber auch der schon erwähnte Block-Effekt ist ein Fall für die *split-plot* ANOVA.

7.2.1. Blockeffekt

Häufig werden in experimentellen Untersuchungen die verschiedenen Behandlungen in einem Block zusammengefasst, der dann repliziert wird (siehe Abschnitt 13). Wir können uns hier verschiedene Düngestufen vorstellen, aber auch 4 Leistungstests, die nur an einer Versuchsperson pro Tag, deshalb an verschiedenen Versuchspersonen an verschiedenen Wochentagen stattfinden. Hierbei wäre dann der Tag der Block.

Den Blockeffekt können wir einfach berechnen, indem wir über die Behandlungen (Leistungstests) innerhalb eines Blocks (Tag) mitteln, und ausrechnen, wieviel der Gesamtvarianz

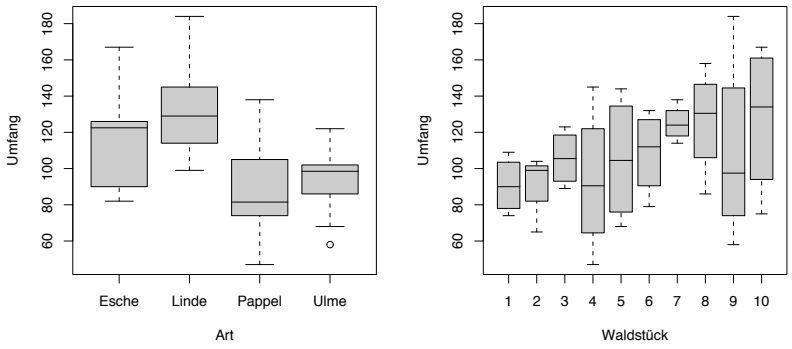


Abbildung 7.1.: Box-Whiskers-Plot des Umfangs [cm] von vier verschiedenen Baumarten in 10 Waldstücken, links gemittelt über die Waldstücke, recht über die Baumarten.

zwischen den Blöcken, und wieviel innerhalb der Blöcke liegt. Allerdings ist bei einfachem Design und einem balancierten Datensatz egal, ob wir den Blockeffekt als zufälligen oder festen Effekt berechnen.

Ein kurzes Beispiel für die letzte Bemerkung. Wir haben den Baumumfang von 4 Arten in 10 Waldgebieten (Block) gemessen und interessieren uns für Unterschiede zwischen den Arten (Abbildung 7.1. Zunächst behandeln wir block als festen Effekt:

```
> baum <- read.table(file = "baum.txt", header = T)
> baum$block <- factor(baum$block)
> summary(aov(baum ~ block + Art, data = baum))

            Df  Sum Sq Mean Sq F value   Pr(>F)
block        9  7180.1   797.8  1.2399 0.313096
Art          3 12308.1  4102.7  6.3762 0.002079 **
Residuals   27 17372.9   643.4
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Wir sehen, dass der Blockeffekt zwar nicht signifikant ist, er aber doch einige Varianz aus dem Datensatz nimmt (nämlich 7180.1). Jetzt zur Illustration der Blockeffekt als zufälliger Effekt:

```
> summary(aov(baum ~ Art + Error(block), data = baum))

Error: block
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals  9 7180.1   797.8

Error: Within
          Df  Sum Sq Mean Sq F value   Pr(>F)
Art        3 12308.1  4102.7  6.3762 0.002079 **
Residuals 27 17372.9   643.4
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Das Ergebnis ist das Gleiche, aber in einer etwas veränderten Darstellung. R teilt die Analyse in zwei Strata auf, wobei das erste durch den `Error()`-Ausdruck bestimmt wird. Dies ist der Blockeffekt. Er wird nicht mit einem Signifikanztest versehen, da wir ja eigentlich nicht am Blockeffekt interessiert sind (sonst hätten wir ihn ja als festen Effekt ins Modell genommen). Im zweiten Stratum findet dann der eigentliche Test auf Artunterschiede statt.

7.2.2. Verschachtelte Versuchseinheiten unterschiedlicher Größe: split plots

Wenn wir eine Behandlung auf eine Untereinheit anderen Behandlung anwenden, dann sprechen wir von *split-plots*. Gerade in der landwirtschaftlichen Forschung sind *split-plots* sehr verbreitet. Beispielsweise wird auf einem Feld die eine Hälfte bewässert, die andere nicht, innerhalb der jeweiligen Teilflächen wird dann noch ein Düngeexperiment durchgeführt. Somit findet das Düngeexperiment auf einer Untereinheit der Behandlung Bewässerung statt. Die Replikation findet auf der Ebene des Feldes statt. Für Einzelheiten zu diesem Design verweisen wir auf den Abschnitt 13.

Das Problem in der Auswertung liegt darin, dass wir sicherstellen müssen, dass wir die zwei Düngebehandlungen innerhalb der Bewässerung nicht als unabhängige Datenpunkte verwerten, wenn wir den Bewässerungseffektbetrachten wollen (Pseudoreplikation; siehe wiederum Abschnitt 13).

Bei der Auswertung erhalten wir mehrere Strata: (1) den Block, (2) die Bewässerung und (3) die Düngung×Bewässerung.

Hier haben wir ein Beispiel bei dem auf 4 Feldern (`block A-D`) die eine Hälfte bewässert wurde bzw. eben nicht (`bewaessert ja/nein`), und auf jeder Bewässerungs- und Nichtbewässerungsfläche gibt es zwei Düngevarianten (`geduengt N/P`)[1]. Insgesamt haben wir also 16 Versuchseinheiten für den Dünger, aber nur 8 für die Bewässerung. Im `Error()`-Term müssen wir diese Struktur abbilden.

```
> duenger <- read.table(file = "splitplotduenger.txt", header = T)
> attach(duenger)
> summary(aov(ertrag ~ geduengt * bewaessert + Error(block/bewaessert/geduengt)))

Error: block
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals  3 548.69  182.90

Error: block:bewaessert
           Df Sum Sq Mean Sq F value  Pr(>F)
bewaessert  1 6440.1  6440.1  24.748 0.01561 *
Residuals   3  780.7   260.2
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Error: block:bewaessert:geduengt
                    Df Sum Sq Mean Sq F value  Pr(>F)
geduengt             1 637.56  637.56  5.8818 0.05149 .
geduengt:bewaessert  1  85.56   85.56  0.7894 0.40850
Residuals            6 650.37  108.40
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Während also das Bewässern einen signifikanten Einfluss auf den Ertrag hatte, hat Düngen nur einen marginal signifikanten Effekt. Auch wenn wir die Interaktion herausnehmen ändert sich dies nicht.
Ein kurzer Blick auf die Freiheitgrade der Residuen. Hier erkennen wir, ob wir R haben deutlich machen können, wie die Behandlungen verschachtelt sind. Im ersten Stratum (`block`) haben wir $4-1=3$, stimmt. Im zweiten Stratum (`bewaessert`) haben wir $8-3$(von den Residuen des ersten Stratums)-1(für die Bewässerung)$-1=3$, stimmt auch. (Hier taucht bei vielen „einfachen" Statistikprogrammen der Fehler auf, dass nicht die 8 sondern alle 16 Versuchseinheiten zugrunde gelegt werden.) Und im dritten Stratum (`within`, da zwischen den Versuchseinheiten) $16-3$(Residuen I)-3(Residuen II)-1(Effekte Stratum II)-2(Effekte Stratum III)$-1=6$. Voilà!

Ein weiteres Beispiel rechnen wir in Abschnitt 13, im Anschluss an einen sorgfältigeren Blick auf das *split-plot*-Design.

[1] Daten aus Crawley (2002), gekürzt.

7.3. Nesting(nesting(nesting)) & Wiederholungsmessungen

Nesting entsteht durch das mehrfache Beproben einer Versuchseinheit, entweder räumlich (*subsampling*) oder zeitliche (*repeated measurements*). In jedem Fall wird eine Untersuchungseinheit mehrfach zur Datengenerierung genutzt. Damit sind die so gewonnen Daten nicht mehr voneinander unabhängig. Da sich sich die Beprobung auf unterschiedlichen hierarchischen Ebenen abspielt, sind auch die Varianzen der Residuen wahrscheinlich unterschiedlich. Dies muss in der Analyse Berücksichtigung finden.

Am besten erklärt sich das Nesting wiederum am Beispiel.[2] In diesem Fall geht es um den Einfluss von 3 verschiedenen Behandlungen auf den Glykogengehalt von Rattenlebern. Die Lebern von 2 Ratten wurden in je 9 Teile zerlegt, von denen jeweils 3 in 3 verschiedene Lösungen eingelegt wurden (Kontrolle, Substanz „217“und 217 plus Zucker). Somit haben wir je Behandlung 3 *subsamples* je Ratte. Dann wurde von jedem *subsample* zweimal der Glykogengehalt gemessen. Alles klar?

Entsprechend des Versuchsdesigns haben wir dann auch 4 Strata: (1) Behandlung, (2) Ratte in Behandlung, (3) *subsample* in Ratte in Behandlung und (4) Residuen nach Berücksichtigung der drei Strata.

Das Entscheidende bei *nested designs* ist die Formulierung der Modellstruktur. Es muss klar werden, welcher Faktor in welchem genested ist. In R wird dies durch den Schrägstrich im `Error()`-Term kodiert. Dabei fangen wir mit der höchsten Einheit an (hier `Behandlung`), und werden sukzessive kleiner.[3]

```
> rattenleber <- read.table(file = "Ratten.txt", header = T)
> rattenleber$subs <- factor(rattenleber$subs)
> rattenleber$Ratte <- factor(rattenleber$Ratte)
> attach(rattenleber)
> summary(aov(Glykogen ~ Behandlung + Error(Behandlung/Ratte/subs)))

Error: Behandlung
           Df  Sum Sq Mean Sq
Behandlung  2 1557.56  778.78

Error: Behandlung:Ratte
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals  3 797.67  265.89

Error: Behandlung:Ratte:subs
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals 12  594.0    49.5

Error: Within
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals 18 381.00   21.17
```

Wir sind zunächst ja am Einfluss der Behandlung interessiert. Wo findet sich dieser im *output*? Da `Behandlung` das oberste Stratum ist, finden sich dort auch die Daten, diesen Effekt zu testen. Dies müssen wir aber per Hand machen. Dafür wird der MS-Wert von `Behandlung` (778.78) geteilt durch den MS des nächsttieferen Stratums (265.89). An den Freiheitgraden kann man sich dies wieder am besten deutlich machen: Für den Test des Behandlungseffekts stehen ja nur 6 Datenpunkte zur Verfügung (zwei Ratten mal drei Behandlungen). In den Residuen stecken aber noch 18 Freiheitsgrade. Testen wir allerdings `Behandlung`

[2] Dieses konkrete Beispiel ziert mehrere Statistikbücher und es entstammt ursprünglich Sokal and Rohlf (1995, S. 289ff.). Der Reiz liegt in seiner extremen Datenarmut und den gleichzeitigen Verständnisschwierigkeiten, die das Design produziert! Nicht zu übersehen ist auch, dass dies ein Gedankenexperiment ist, denn die Beschreibung des Versuchs ist, gelinde gesagt, dürftig.

[3] Siehe Baron and Li (2003, S. 27ff.) für weitere Beispiele zur Art und Weise, wie der `Error()`-Term spezifiziert werden kann.

gegen `Behandlung:Ratte`, so haben wir unsere 6 Freiheitsgrade insgesamt -2 Freiheitsgrade für `Behandlung` -1 Freiheitsgrad $= 3$ Freiheitsgrade. Genau soviele hat auch das Stratum `Behandlung:Ratte`.
Jetzt müssen wir nur noch den Quotienten der *mean sum of squares* in einen p-Wert umrechnen. Der kritsche F-Wert ($p = 0.05$) für 2 und 3 Freiheitgrade lässt sich in R berechnen als:

```
> qf(0.05, 2, 3, lower.tail = FALSE)

[1] 9.552094
```

Oder wir lassen uns den p-Wert selbst berechnen als:

```
> pf(778.78/265.89, 2, 3, lower.tail = FALSE)

[1] 0.1970992
```

Die Option `lower.tail=FALSE` lässt R die Wahrscheinlichkeit berechnen, dass ein Wert der F-Verteilung *größer* ist als unser gemessener F-Wert; also den Prozentsatz der Verteilung, der über unserem Wert liegt. Nämliches gilt für die Quantilenfunktion `qf`: Wir suchen den F-Wert, von dem aus 95% der Verteilungswerte *größer* sind.
Wir schließen daraus, dass unsere Behandlung keinen signifikanten Einfluss auf den Glykogengehalt der Lebern hatte. Für die anderen Strata lässt sich das gleiche Spielchen betreiben, und wir erhalten eine signifikante Interaktion `Behandlung:Ratte` (`pf(265.89/49.5, 3, 12, lower.tail=F) = 0.0141`), sowie `Behandlung:Ratte:subs` (`pf(49.5/21.17, 12, 18, lower.tail=F) = 0.0503`).
Für Menschen, die die harte 0.05-Grenze des p-Wertes nicht mögen: Es interessant, dass sowohl Sokal and Rohlf (1995, S. 291) als auch Crawley (2002, S. 368) den `Behandlung:Ratte:subs`-Effekt als signifikant bezeichnen.

Für zufällige Faktoren können wir sog. **Varianzkomponenten** berechnen. Diese geben an, wieviel der Variabilität in welchem der Strata steckt (in den zufälligen Faktoren). Im obigen Beispiel können wir also drei Varianzkomponenten berechnen: (1) Varianz in den Residuen, (2) Varianz in den *subsamples* und (3) Varianz in den Ratten (innerhalb der Behandlungen).

Die Varianz in den Residuen wird uns im obigen Beispiel direkt angegeben: 21.17. Die Varianz im nächsten Stratum berechnet sich als die Differenz der MS des Stratums und des Stratums darunter, geteilt durch die Anzahl Messungen auf diesem Level. Die Varianzkomponente in den *subsamples* ($n = 2$) beträgt also $(49.50 - 21.17)/2 = 14.17$. Und die Varianz zwischen den Ratten innerhalb der Behandlungen ($n = 6$) liegt bei $(265.9 - 49.5)/6 = 36.06$. Als Merksatz nimmt die Anzahl Messungen vom unteren zum oberen Stratum immer zu. Im vorliegenden Fall ist also der Hauptteil der Variabilität zwischen den Ratten zu finden, nämlich $100\% \cdot 36.06/(21.17 + 14.17 + 36.03) = 50.7\%$, während die Wiederholungsmessungen (subsamples) am wenigsten zur Gesamtvarianz durch Zufallsfaktoren beigetragen haben (20.0%).

7.3.1. Räumliches Nesting

Für ein Beispiel für die Analyse räumlichen Nestings sei auf Abschnitt 13 verwiesen.

7.3.2. Zeitliches Nesting: Wiederholungsmessungen

In diesem Zusammenhang wollen wir uns mit der nacheinander wiederholten Messung einer Versuchseinheit beschäftigen. Bei Psychologen heißt diese Form der Datenerhebung Langzeitstudie (im Gegensatz zur Querschnittstudie, die nur einmal durchgeführt wird), oder auch longitudinale Untersuchung (*longitudinal data, repeated measurements*). Es geht hierbei *nicht* um Zeitreihenanalyse. Bei letzterer liegt der Schwerpunkt der Analyse auf den zeitlichen Mustern innerhalb einer Zeitreihe (etwa Kursschwankungen am Aktienmarkt oder Hochwasserstände). In diesem Abschnitt hingegen geht es um die Effekte fester Faktoren (experimentelle

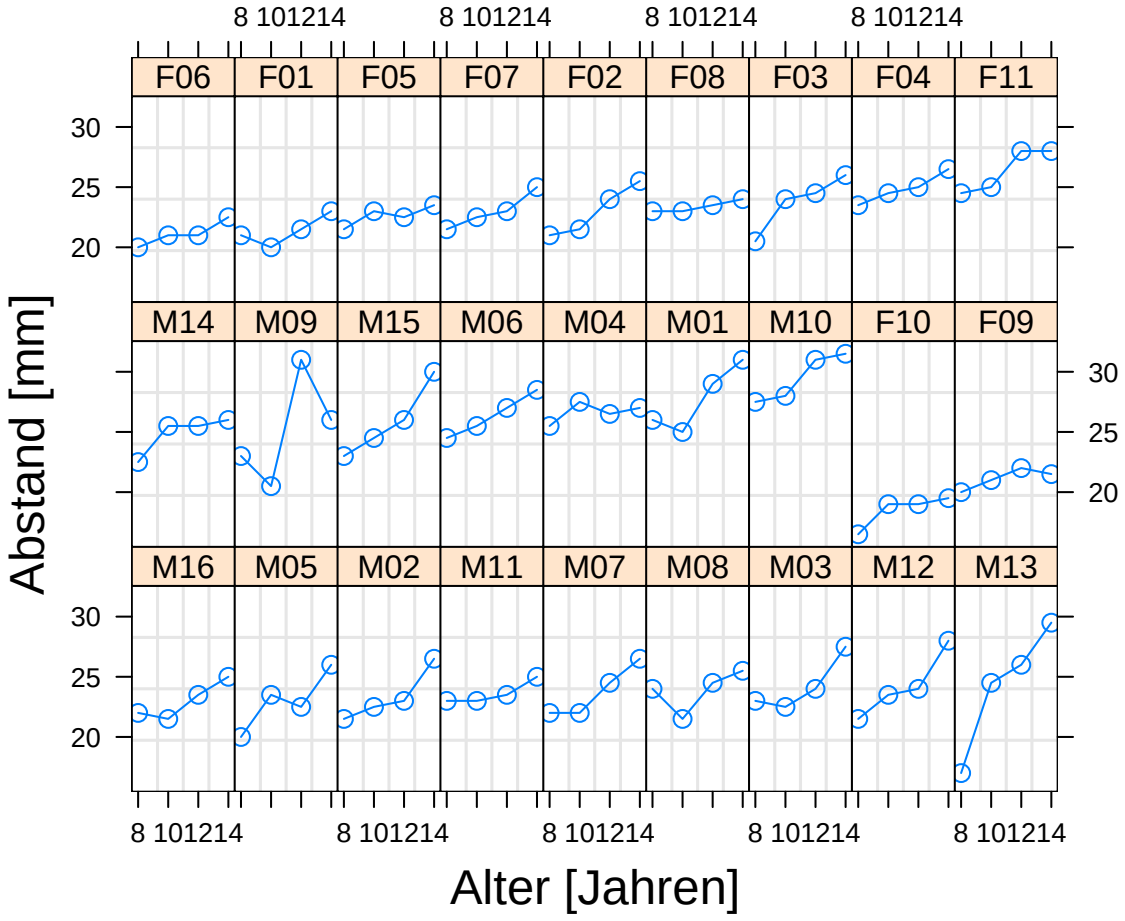


Abbildung 7.2.: Abstand zwischen der hypophysären und der pterygomaxillaren Kluft von 11 Mädchen (F01-F11) und 16 Jungen (M01-M16), aufgetragen gegen ihr Alter. Nach Pinheiro and Bates (2000).

Behandlungen oder deskriptive erklärende Faktoren), die über einen Zeitraum mehrfach erfasst wurden. Damit haben wir für jede Versuchseinheit mehrere abhängige Messpunkte, die im Faktor Zeit genestet sind.

Je nachdem ob wir an den zeitlichen Veränderungen innerhalb der Versuchsflächen interessiert sind oder nicht, brauchen wir viele Wiederholungsmessungen, oder nur ein paar. Die Analyse vieler Wiederholungsmessungen wird allerdings durch zu erwartende Nicht-Linearitäten (wie etwa zyklische Populationsentwicklungen bei nordischen Nagetieren) oder ähnliches erschwert. Hier ein genaues Modell zu fitten ist deutlich schwieriger, und überschreitet die Zielsetzung dieses Werkes[4].

Auch hier lernen wir wieder am besten am Beispiel[5]. An 27 Kindern (16 Jungen, 11 Mädchen) wurde die Kieferbildung im Alter von 8 alle zwei Jahre durch Röntgenaufnahmen gemessen, bis die Kinder 14 waren (Abbildung 7.2). Als R-Funktion benutzen wir `lme`, die eine flexiblere (und in der Berechnung genauere) Spezifizierung der Nesting-Struktur ermöglicht. `lme` steht für *linear mixed effect*.

[4]Für einfache Nichtlinearitäten kann man mittels `correlation` Funktionen zur Modifizierung der Varianz-Kovarianz-Matrix angeben. Diese können für zyklische Zeitreihen z.B. einen Sinus-Term enthalten (Pinheiro and Bates 2000). *De facto* werden ökologische Zeitreihen aber meistens mittels eines theoretischen populationsdynamischen Modells analysiert, das an die Daten gefittet wird. Dieser Vorgang hat dann weniger mit schließender Statistik zu tun sondern fällt unter das Thema Optimierung.

[5]Aus Pinheiro and Bates (2000, S. 146ff.), Daten publiziert von Potthoff and Roy (1964)

```
> library(nlme)
> data(Orthodont)
> fm1 <- lme(distance ~ I(age - 11), random = ~I(age - 11) | Subject,
+     data = Orthodont)
> summary(fm1)

Linear mixed-effects model fit by REML
 Data: Orthodont
       AIC      BIC    logLik
  454.6367 470.6173 -221.3183

Random effects:
 Formula: ~I(age - 11) | Subject
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr
(Intercept) 2.1343289 (Intr)
I(age - 11) 0.2264278 0.503
Residual    1.3100402

Fixed effects: distance ~ I(age - 11)
                Value Std.Error DF  t-value p-value
(Intercept) 24.023148 0.4296601 80 55.91198       0
I(age - 11)  0.660185 0.0712533 80  9.26533       0
 Correlation:
            (Intr)
I(age - 11) 0.294

Standardized Within-Group Residuals:
         Min           Q1          Med           Q3          Max
-3.223106873 -0.493760899  0.007316483  0.472151219  3.916031757

Number of Observations: 108
Number of Groups: 27
```

Offensichtlich hat das Alter einen starken Effekt. Jetzt wollen wir den Effekt von Alter und Geschlecht untersuchen. Dafür benutzen wir der Einfachheit halber den update-Befehl, der nur die Veränderung gegenüber dem vorigen Modell angibt.

```
> fm2 <- update(fm1, fixed = . ~ Sex * I(age - 11))
> summary(fm2)

Linear mixed-effects model fit by REML
 Data: Orthodont
       AIC      BIC    logLik
  448.5817 469.7368 -216.2908

Random effects:
 Formula: ~I(age - 11) | Subject
 Structure: General positive-definite, Log-Cholesky parametrization
            StdDev    Corr
(Intercept) 1.8303268 (Intr)
I(age - 11) 0.1803454 0.206
Residual    1.3100397

Fixed effects: distance ~ Sex + I(age - 11) + Sex:I(age - 11)
                          Value Std.Error DF  t-value p-value
(Intercept)           24.968750 0.4860007 79 51.37595  0.0000
SexFemale             -2.321023 0.7614168 25 -3.04829  0.0054
I(age - 11)            0.784375 0.0859995 79  9.12069  0.0000
```

```
SexFemale:I(age - 11) -0.304830 0.1347353 79 -2.26243  0.0264
 Correlation:
                      (Intr) SexFml I(-11)
SexFemale             -0.638
I(age - 11)            0.102 -0.065
SexFemale:I(age - 11) -0.065  0.102 -0.638

Standardized Within-Group Residuals:
         Min           Q1          Med           Q3          Max
-3.168078325 -0.385939104  0.007103934  0.445154637  3.849463326

Number of Observations: 108
Number of Groups: 27
```

```
> anova(fm2)
```

```
                numDF denDF  F-value p-value
(Intercept)         1    79 4035.594  <.0001
Sex                 1    25    8.023  0.0090
I(age - 11)         1    79   99.445  <.0001
Sex:I(age - 11)     1    79    5.119  0.0264
```

Der Abstand nimmt also für die beiden Geschlechter unterschiedlich stark zu. Den Koeffizienten können wir entnehmen, dass bei Mädchen der Abstand sowohl von Beginn an geringer ist (negativer `SexFemale`-Koeffizient), als auch langsamer zunimmt (negativer `SexFemale:I(age-11)`-Interaktion). Bei den Freiheitsgraden sehen wir für alle Effekte den Wert 79, außer bei `Sex`. Für den Vergleich von Mädchen und Jungen stehen ja schließlich nur die 27 Mittelwerte der Testindividuen zur Verfügung.
Schauen wir uns hier doch zum Vergleich einmal den *output* eines `aov`-Modells an. Der Unterschied besteht in der Berechnungsmethode. `aov` basiert auf Matrixalgebra, wobei der `Error()`-Term in eine Kontrastematrix übersetzt wird (Baron and Li 2003), während `lme` einen iterativen *likelihood*-Ansatz benutzt, und zwar genauer gesagt *restricted maximum likelihood* (REML). Wenn wir stattdessen *maximum likelihood* benutzen, müssten `aov` und `lme` die gleichen Werte produzieren. `lme` ist `aov` aber im allgemeinen deutlich überlegen, da sich kompliziertere Modellstrukturen formulieren lassen und wir sofort die Koeffizienten der Effekte erhalten.

```
> fm4 <- aov(distance ~ Sex * I(age - 11) + Error(Subject), data = Orthodont)
> summary(fm4)
```

```
Error: Subject
          Df Sum Sq Mean Sq F value   Pr(>F)
Sex        1 140.46  140.46  9.2921 0.005375 **
Residuals 25 377.91   15.12
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Error: Within
                Df  Sum Sq Mean Sq  F value  Pr(>F)
I(age - 11)      1 235.356 235.356 122.4502 < 2e-16 ***
Sex:I(age - 11)  1  12.114  12.114   6.3027 0.01410 *
Residuals       79 151.842   1.922
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Wir erhalten qualitativ die gleichen Ergebnisse, wenngleich im zweiten Stratum der p-Wert für die Interaktion im `aov`-Modell höher ist.

Wenn die Antwortvariable nicht wie hier linear mit der Zeit zu- oder abnimmt, dann können wir durch Transformation der Zeitachse eine Linearisierung anstreben. Wenn allerdings

oszillierende oder nicht-linearisierbare Funktionen vermutet werden, dann kann man auf nicht-lineare gemischt Modelle ausweichen. Dies bietet auch die Möglichkeit, räumliche Korrelationen mit ins Modell hineinzunehmen. Dies Thema ist aber zu fortgeschritten für diesen Text. Siehe Pinheiro and Bates (2000) für eine ausführliche Betrachtung dieses Themas.

Teil III.

Univariate Statistik II: Das Verallgemeinerte Lineare Modell (Generalised Linear Model, GLM)

Bis zu diesem Punkt haben wir uns ausschließlich mit normalverteilten Daten auseinandergesetzt (von ein paar Ausnahmen in den Klassischen Tests einmal abgesehen). Im Abschnitt 2.5 haben wir bereits gesehen, dass es einen allgemeinen Ansatz zur Berechnung von Parametern statistischer Modelle gibt, der zwar verteilungsgebunden, aber nicht abhängig von der Normalverteilung ist. Mittels dieses *maximum likelihood*-Ansatzes können wir auch Poisson-, binomial- oder anders verteilte Datensätze analysieren.

Im Abschnitt 2.5.1 haben wir gezeigt, dass für normalverteilte Daten der *maximum likelihood*-Ansatz zu der Methode der minimalen Abweichungsquadrate (*ordinary least squares*, OLS) führt. OLS-Verfahren (etwa Regression und ANOVA) sind analytisch lösbar, mittels Matrixalgebra (Abschnitt 6.6). Dies ist leider nicht der Fall, wenn wir uns den anderen Verteilungen zuwenden. Hier werden die Parameter iterativ numerisch[6] gelöst. Wenn die Parameter bis auf 10-15 Stellen genau geschätzt sind, dann wird dieser Berechnungsvorgang abgebrochen.

Bei manchen Daten, etwa Prozentzahlen oder Zähldaten, besteht u.U. die Möglichkeit, durch Transformationen eine Normalverteilung anzunähern (siehe Abschnitt 3.3). Aber natürlich ist es besser, die tatsächlichen Daten zu modellieren, statt transformierter, denn bei der Transformation geht doch möglicherweise einige Information verloren.

Darüber hinaus gibt es natürlich Daten, die nicht normalisierbar sind, etwa binäre Antwortvariablen (0/1), extrem schiefe Verteilungen usw. Auch diese sind dem Verallgemeinerten Linearen Modell zugänglich.

Mit dem numerischen Lösen der *maximum likelihood*-Funktion ist ein erheblicher Rechenaufwand verbunden. Noch bis vor wenigen Jahren war dieses Verfahren deshalb nur sehr wenig verbreitet; inzwischen hat es sich, häufig auch unter anderen Namen (logistische Regression und log-lineare Modelle) etabliert (McCullough and Nelder 1989) und ist Teil aller aktueller Statistiksoftware.

Ebenso wie das Allgemeine Linear Modell eine Ausweitung der Regression und einfaktoriellen ANOVA darstellt, ist das Verallgemeinerte Lineare Modell eine Erweiterung des Allgemeinen Linearen Modells. Hier haben wir diese Ausdrücke letztmalig benutzt, denn wir haben ja gesehen, dass das Lineare Modell mathematisch unabhängig von der Anzahl erklärender Variablen ist, und somit das „verallgemeinerte“ eigentlich gleich dem „einfachen“ Linearen Modell ist.

Das Verallgemeinerte Linear Modell (*Generalised Linear Model*, im Englisch-Amerikanischen mal als GLM, mal als GLZ bezeichnet) ist hingegen ein methodischer Sprung im Vergleich mit dem LM. Während zwar in beiden implizit auf den *maximum likelihood*-Ansatz zurückgegriffen wird, ist er nur im GLM auch als solcher implementiert (beim LM wie gesagt statt dessen Matrixalgebra). Um Daten mit einem GLM analysieren zu können, müssen wir wissen/raten, welche Verteilung dem Fehlerterm zugrunde liegt.

Stellen wir uns etwa vor, wir wollten die Gelegegröße von Kohlmeisen analysieren. Diese Werte werden Poisson-verteilt sein, und der Fehler in unserem Modell entsprechen nicht normalverteilt um die Mittelwerte. Entweder hat ein Nest ein oder zwei Eier mehr oder weniger, aber nicht 0.3. Entsprechend müssen wir der analysierenden Software mitteilen, dass der Fehlerterm in diesem Fall möglicherweise Poisson-verteilt ist.

Über all diese Veränderung dürfen wir die zentralen Annahmen nicht vergessen: Die Unabhängigkeit der Datenpunkte und die Homogenität der Varianzen. [7] Dies bedeutet, dass auch

[6] „Numerisch“ bedeutet hier etwa „durch ausrechnen“, während „iterativ“ bedeutet, dass aufgrund der letzten Berechnung eine neue, besser geschätzte durchgeführt werden kann.

[7] Zwar werden im GLM nicht Varianzen modelliert, sondern eine „Abweichung“, die als *deviance* bezeichnet wird. Sie entspricht jedoch den Abweichungsquadraten in der ANOVA/Regression, und ist identisch zu diesen für normalverteilte Daten (McCullough and Nelder 1989). Für andere Verteilungen ist die *deviance* etwas komplizierter und für das Verständnis der weiteren Methoden nicht wesentlich. Sie ist wie folgt definiert ($\hat{\mu}$ ist der vorhergesagte Mittelwert):
normalverteilt: $\sum (y - \hat{\mu})^2$
Poisson-verteilt: $2 \sum (y \log(y/\hat{\mu}) - (y - \hat{\mu}))$

im GLM beispielsweise kein systematischer Zusammenhang zwischen Gruppenmittelwerten und ihrer Varianz/*deviance* vorliegen darf.

binomial-verteilt: $2\sum(y\log(y/\hat{\mu}) + (m-y)\log((m-y)/(m-\hat{\mu})))$
gamma-verteilt: $2\sum(-log(y/\hat{\mu}) + (y-\hat{\mu})/\hat{\mu})$
reziprok-normalverteilt (inverse Gaussian): $\sum(y-\hat{\mu})^2/(\hat{\mu}^2 y)$.
Wie man an diesen Formeln sieht, leitet sich die *deviance* aus den *log-likelihood*s der jeweiligen Verteilungen ab, genauer aus der Differenz zwischen dem tatsächlichen und dem maximalen Modell.

8. Verallgemeinertes Lineares Modell

Neben den verschiedenen Möglichkeiten der Fehlerverteilung besitzt das GLM noch ein weiteres Merkmal, dass es vom bisherigen LM unterscheidet, die sog. *link*-Funktion. Sie beschreibt die Art und Weise, in der die abhängige Variable von den erklärenden abhängig ist. Mathematisch kann man dies so schreiben:

$$g(y_i) = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \epsilon$$

oder näher an der Wirklichkeit und in Matrixschreibweise,

$$E(y) = g^{-1}(\mathbf{X}\beta)$$

wobei y_i der vorhergesagt Wert für den $i$ten Datenpunkt, X_{1i} usf. die Werte der erklärenden Variablen für den $i$ten Punkt, β der Vektor der zu schätzenden Modellparameter und $E(\mathbf{y})$ die "Erwartungswerte" von $\mathbf{y}$ sind. Die Aufgabe der *link*-Funktion $g(.)$ ist die Übertragung der verschiedenen Verteilungen auf (zumindest potentiell) die gesamte reale Zahlenachse.

Wieso "näher an der Wirklichkeit"? Nun, die erste obige Formel suggeriert, dass die y-Werte mittels der link-Funktion "transformiert" werden, und dann ein lineares Modell gefittet wird. So ist es nicht. Vielmehr ist diese Schreibweise nur einfacher und doch für alle *link*-Funktionen gültig. Wenn das Modell gefittet wird, so geschieht dies aber in der folgenden Form (am Beispiel des *logit-link*, s.u.):

$$y = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots}}{1 + e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots}}$$

An dieser Formel sehen wir, dass nicht die y-Werte transformiert werden, sondern dass diese durch die *link*-Funktion an das (verallgemeinerte) lineare Modell gekoppelt sind. In dieser Formel nicht ablesbar ist, dass die Anpassung der Koeffizienten (β_i) nicht mittels der Minimierung der Abweichungsquadrate zwischen Modellvorhersage und Beobachutung erfolgt (wie im Abschnitt Regression behandelt), sondern abhängig ist von der zugrundeliegenden Verteilung.

Hier kommen wir zum Kerngedanken der *maximum likelihood*: Wenn durch die Parameter eine Verteilung festgelegt ist (etwa durch Mittelwert und Standardabweichung die Normalverteilung), dann kann man jedem beobachten Wert eine Wahrscheinlichkeit zuordnen, dass dieser oder ein kleinerer Wert beobachtet wurde. Durch die Koeffizienten des GLM (die βs) werden die Parameter der Verteilung für jeden beobachteten Punkt von den erklärenden Variablen abhängig gemacht (z.B. wird der Mittelwert in der Normalverteilung verschoben). Dann ist die Wahrscheinlichkeit einen Wert $\leq y_i$ zu beobachten also von den Modellparametern und den Verteilungsparametern abhängig[1]. Es ist jetzt mit wenig Mühe vorstellbar, dass es einen optimalen Satz an Parametern gibt, so dass die Wahrscheinlichkeiten maximal sind. Das ist die *maximum likelihood*.

Für die üblichen Verteilungen wird immer nur ein Verteilungsparameter an die Modellparameter gekoppelt: bei der Normalverteilung der Mittelwert μ, bei der Poissonverteilung

[1] Wieso $\leq y_i$? Nun, die Wahrscheinlichkeit den Wert y_i zu beobachten ist Null! Die Fläche unter der Wahrscheinlichkeitsdichtefunktion ergibt, per definitionem, 1. Sie wird als Integral der Wahrscheinlichkeitsverteilung berechnet, also durch Ausummieren infinitesimal kleiner Schrittweiten. Da infinitesimal bedeutet, dass die Funktion in unendlich viele Werte zerlegt wird, deren Summen 1 ergeben, muss die Wahrscheinlichkeit jedes einzelnen Wertes $1/\infty = 0$ sein. Was die *maximum likelihood* also als Wahrscheinlichkeitswert angibt ist nicht die Wahrscheinlichkeit einen Wert y_i zu beobachten, sondern einen Wert y_i oder kleiner.

der Mittelwert λ, der hier gleich der Varianz ist) und bei der Binomialverteilung die Wahrscheinlichkeit p. Da wir bei Poisson- und Binomialverteilung durch die Festlegung von λ und p nicht nur den Mittelwert sondern auch die Varianz spezifizieren, tritt bei Daten, die nicht genau diesen Verteilungen folgen, das Problem der "Über- bzw. Unterverteilung" (*over- and underdispersion*) auf. Dazu mehr weiter unten.

Vier verschieden *link*-Funktionen ($g(\mu)$) sind allgemein gebräuchlich, und sie seien hier beschrieben. Die Form dieser "Transformationen" ist in Abbildung 8.1 dargestellt. Die folgende Beschreibung ist sehr holprig. Das liegt daran, dass wir im wesentlichen sagen wollen: Die beobachteten Werte werden auf der *link-scale* modelliert. Das wiederum kann man nur verstehen, wenn man es schon weiß. Deshalb folgt hier der Versuch, mit ungelenken Worten das 'Phänomen' der *link-scale* zu umschreiben.

1. **identity link** Keine Verformung des Regressionsmodells, d.h. $g(\mathbf{y}) = \mathbf{y} = \mathbf{X}\beta + \epsilon$. Die Grundeinstellung für **normalverteilte** Daten. Da keine Verformung vorgenommen wird, kann aus den Regressionsparametern direkt der vorhergesagt Wert berechnet werden.

2. **log link** Der vorhergesagte Wert ist der Logarithmus des beobachteten. Zur Berechnung aus den Modellparametern müssen die Werte $\hat{y}$ als $e^{\hat{y}}$umgerechnet werden. Das Regressionsmodell sieht wie folgt aus: $\mathbf{y} = e^{\beta_0+\beta_1\mathbf{X}_1+\beta_2\mathbf{X}_2+\cdots} + \epsilon$. Der *link* ist also log, der anti-link e. Die Grundeinstellung für **Poisson-verteilte** Daten, da dort die Varianz gleich dem Mittelwert ist. Durch die Logarithmierung wird die Varianz bei steigendem Mittelwert als etwas konstant stabilisiert. Außerdem ist durch den log sichergestellt, dass die Werte immer echt größer Null sind, wie dies ja auch für Poisson-verteilte Daten der Fall ist.

3. **logit link** Das Regressionsmodell wird als anti-logit mit den beobachteten Werten verknüpft. Der logit ist so definiert: $g(y) = ln(y/(1-y))$ und die Umkehrfunktion $g^{-1}(y) = e^y/(1+e^y)$. Somit sieht das Regressionsmodell wie folgt aus: $\mathbf{y} = \frac{e^{\beta_0+\beta_1\mathbf{X}_1+\beta_2\mathbf{X}_2+\cdots}}{1+e^{\beta_0+\beta_1\mathbf{X}_1+\beta_2\mathbf{X}_2+\cdots}} + \epsilon$. Dies ist die Grundeinstellung für **binomial-verteilte** Daten. Für maximale/minimale Parameterwerte nähert sich dieser Ausdruck 0 bzw. 1 an. D.h., die beobachteten 0 und 1 können theoretisch nie erreicht werden, es bleibt immer eine kleine (u.U. sehr kleine) Diskrepanz[2].

4. **inverse link** Statt des gewohnten Regressionsmodells wird sein Reziprok modelliert: $\mathbf{y} = \frac{1}{\beta_0+\beta_1\mathbf{X}_1+\beta_2\mathbf{X}_2+\cdots} + \epsilon$. Der anti-link ist logischerweise identisch zum link. Dies ist die Grundeinstellung für **gamma-verteilte** Daten, bei denen die Varianz mit dem Mittelwert zunimmt. (Reziprok-normalverteilte Daten werden übrigens nicht mit einem *inverse-link* modelliert, sondern mit einem reziprok-quadrierten, wobei das Quadrat aus der Normalverteilung stammt McCullough and Nelder 1989). Die gamma-Verteilung ist in der Biologie stark unterrepräsentiert, ist sie doch oft geeignet, um biologische Daten zu beschreiben. Beispielsweise der Blattfraß phytophager Insekten ist häufig gammaverteilt, da viele Blätter überhaupt nicht, andere aber nahezu vollständig abgefressen sind. Beachte, dass die y-Werte alle echt größer 0 sein müssen!

Bevor wir uns spezifischen GLMs zuwenden sei noch eine Bemerkung zu den Verteilungsannahmen erlaubt. Selbst wenn wir Grund haben, eine bestimmte Verteilung des Fehlers zu erwarten, mag der Fit im GLM gelegentlich nur mäßig sein. Dies ist vor allem bei Poisson- und binomialverteilten Daten der Fall. Dann kommt es zum Phänomen der *overdispersion* (etwa: Überverteilung), bei der die Form der Verteilung nur in etwa getroffen wurde. Um diese Daten trotzdem analysieren zu können gibt es eine Familie der `quasi`-Funktionen, die

[2] Ob der Rechengenauigkeit werden in R Werte kleiner 10^-16 als 0 interpretiert und gelegentlich eine Warnung erzeugen, dass eine exakte 0 oder 1 errechnet wurde.

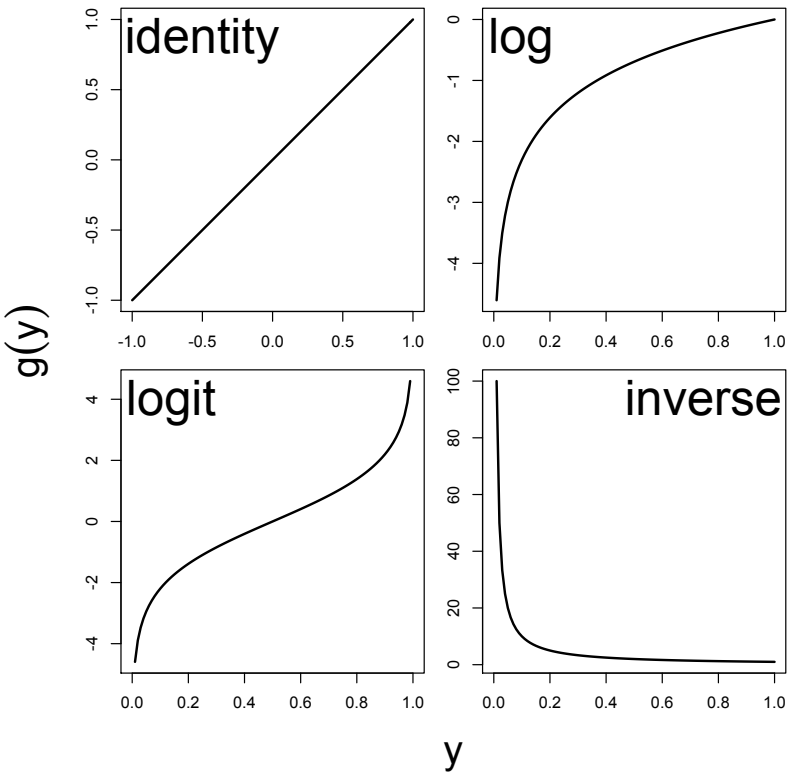


Abbildung 8.1.: Übliche *link*-Funktionen für GLMs. a) *identity link*, b) *log link*, c) *logit link* und d) *inverse link*.

mittels Formparametern die Verteilung in die Richtige Form „biegen“. Da *overdispersion* ein großes Problem bei GLMs sein kann, muss man ihm Rechnung tragen. Wir werden dies dann am konkreten Beispiel tun.

8.1. Binomial-verteilte Fehler: logistische Regression

Für binomial-verteilte Daten (am wichtigsten binäre Daten, also 0/1, Ja/Nein, anwesend/abwesend usw., aber auch Erfolge/Versuche im allgemeinen) wird als Fehlerverteilung die Binomialverteilung gewählt, und standardmäßig auf den *logit link* zurückgegriffen. Deshalb wird die Analyse binomialverteilter Daten mittels eines GLM auch logistische Regression genannt, und sie ist genau das. Wir führen eine Art Regression durch, und benutzen dabei die Annahme, dass unsere Fehler der Binomialverteilung entstammen.

Die Ergebnisse die wir erhalten, gleichen denen einer gewöhnlichen Regression mittels des linearen Modells: einen y-Achsenabschnitt und eine Steigung je Variable im Modell. Wenn wir allerdings aus diesen Koeffizienten die vorhergesagten Werte berechnen wollen, so stehen wir vor dem Problem der Rücktransformation. Unsere Analyse beruht auf folgender Gleichung:

$$logit(y) = mx + b$$

Wenn wir jetzt aber y vorhersagen wollen, so müssen wir nach y umformen, und es ergibt sich:

$$y = \frac{e^{mx+b}}{1 + e^{mx+b}}$$

Beim Nachrechnen berücksichtige, dass $logit(y) = ln(y/(1-y))$ ist.

Alle GLM-Varianten sind in R mit der Funktion `glm` implementiert. Als weiteren Befehl müssen wir die “Familie” (*family*) der Fehlerverteilung angeben, hier also `binomial`. Die *link*-Funktion ist standardmäßig

der logit für diese Familie (siehe Abschnitt 8 auf Seite 142). Als Beispiel benutzen wir das Vorkommen (oder Fehlen) von Nagetieren in Schluchten unterschiedlich langer Isolation in Südkalifornien (nach Daten aus Bolger et al. 1997).

```
> bolger <- read.table("bolger.txt", header = T)
> attach(bolger)
> names(bolger)

[1] "PERSHRUB" "DISTX"    "AGE"      "RODENTSP"

> plot(RODENTSP ~ AGE, pch = 16, cex = 1.5)
> fm1 <- glm(RODENTSP ~ AGE, family = binomial)
> summary(fm1)

Call:
glm(formula = RODENTSP ~ AGE, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4329  -1.1078  -0.4999   0.8762   2.1993

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.46429    0.82751   1.770   0.0768 .
AGE         -0.04406    0.02138  -2.061   0.0393 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 34.617  on 24  degrees of freedom
Residual deviance: 28.923  on 23  degrees of freedom
AIC: 32.923

Number of Fisher Scoring iterations: 4
```

Wir sehen, dass `AGE` signifikant von 0 verschieden ist. Im Gegensatz zur Regression mittels `lm` erhalten wir hier aber keinen F-Test, da dieser für nicht-normalverteilte Daten unzulässig ist. Anstelle der Abweichungsquadrate erhalten wir hier die *deviance* sowie einen AIC-Wert. AIC steht für *Akaike Information Criterion* und ist eine Abwandlung der *log-likelihood*: $AIC = -2logL + 2K$. Dabei ist K die Anzahl der Parameter im Modell, hier also 2. Der AIC berücksichtigt also neben der Güte des *fits* (gegeben durch die Wahrscheinlichkeit *log likelihood*) auch die Anzahl Parameter. Je mehr, desto größer wird der AIC. Ideal ist also ein möglichst niedriger AIC.

Es besteht die Möglichkeit sich von R an dieser Stelle eine ANOVA-artige Tabelle erstellen zu lassen (mittels der Funktion `Anova` aus dem *package* **car**; beachte den Großbuchstaben!). Der Standardbefehl `anova(., test="Chisq")` liefert ähnliche Informationen in einem anderen Layout[3].

[3] `Anova` und `anova` unterscheiden sich vor allem darin, dass `Anova` eine "*type II sum of squares*" berechnet, bzw. eine *type III* (siehe `?Anova`). `anova` hingegen berechnet eine "*type I sum of squares*", was nach Meinung von Venables (1994) auch korrekter ist. Die Typen unterscheiden sich dann, wenn die Variablen korreliert sind oder wir Interaktionen ins Modell nehmen. *Type II* und *III* berechnen den Erklärungswert (*deviance*) jeder Variablen separat, während *type I* ihn in der Reihenfolge der Variablen im Modell berechnet. Wenn wir also vor einer Analyse für eine Kovariate korrigieren wollen, dann nehmen wir sie zuerst ins Modell und benutzen *type I SS*. Wenn wir hingegen drei gleichwertige Variablen haben, und uns nicht sicher sind, welche wichtiger ist, dann ist *type III SS* die richtige Wahl. In der *R-help mailing list* gibt es einige interessante Details über diese Unterschiede zu lesen. Als *take-home-message* gilt: immer *type I* (also `anova`), außer wenn man sich ganz sicher ist, dass *type I* ungeeignet ist.

Danach lassen wir uns vom Modell Werte für alle Fragmentalter von 0 bis 100 ausrechnen und als Linie in die Abbildung einfügen (siehe Abbildung 8.2). Der Befehl `type="response"` berechnet die vorhergesagten Werte wie wir sie wollen. Die Grundeinstellung berechnet sie aus den Koeffizienten ohne Rücktransformation!

```
> require(car)
> Anova(fm1)

Anova Table (Type II tests)

Response: RODENTSP
    LR Chisq Df Pr(>Chisq)
AGE   5.6943  1    0.01702 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> anova(fm1, test = "Chisq")

Analysis of Deviance Table

Model: binomial, link: logit

Response: RODENTSP

Terms added sequentially (first to last)


     Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                    24     34.617
AGE   1    5.694        23     28.923     0.017

> new <- data.frame(AGE = seq(0, 100, by = 1))
> pred <- predict(fm1, new, type = "response")
> lines(new$AGE, pred, lwd = 2, col = "red")
```

Wie die Abbildung 8.2 zeigt, wird durch den *logit link* eine sigmoidale Funktion gefittet. Die Steigung ist umso stärker, je größer der Koeffizient ist.`(Intercept)` gibt in der Tat den y-Achsenabschnitt an, allerdings erst nach Rücktransformation ($\frac{e^b}{1+e^b}$).

Häufig interessiert uns auch noch ein anderer Wert, nämlich der Wert der x-Achse, bei der die Wahrscheinlichkeit gerade 50% ist (in der Toxikologie auch als LD50 bezeichnet). Zu seiner Berechnung benutzen wir die Funktion `dose.p`:

```
> library(MASS)
> dose.p(fm1, p = 0.5)

             Dose       SE
p = 0.5: 33.23231 10.22205

> dose.p(fm1, p = 0.1)

             Dose       SE
p = 0.1: 83.09863 26.59438
```

Ab einem Isolationszeitraum von 33 Jahren (± 10) können wir also nur noch in weniger als der Hälfte aller Schluchten mit Nagetieren rechnen. Dramatischer fällt der Fehler für das Vorkommen von nur noch 10% aus, nach nämlich 83 $\pm$ 27 Jahren.
In diesem Fall brauchen wir keine weiteren Angaben zu machen, da die Grundeinstellungen gerade für den 50%-Wert sind. Für weitere Informationen zum Gebrauch nutze die Hilfe: `?dose.p`.

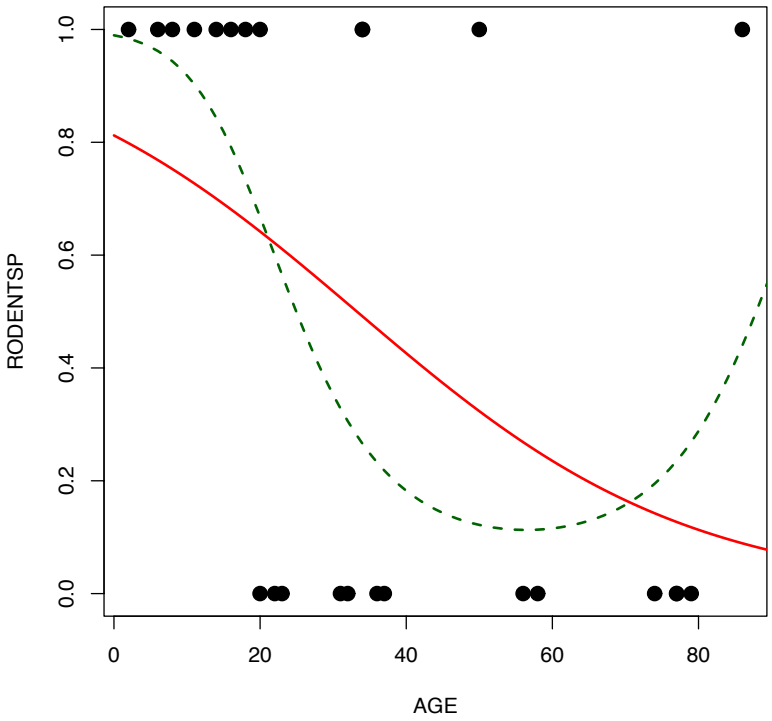


Abbildung 8.2.: Gemessene (Punkte) und vorhergesagte Werte für das Vorkommen/Fehlen von Nagetieren in Schluchten, die vor unterschiedlich langer Zeit durch Besiedlung des Umlandes isoliert wurden (AGE). Die durchgezogene Linie stellt das einfache, die gestrichelte das quadratische Modell dar.

Genau wie beim linearen Modell kann auch das GLM mehrere erklärende Variablen bearbeiten. Diese können kategorial oder kontinuierlich sein. Auch quadratische Effekte können berücksichtigt werden. Fahren wir also mit unserem Beispiel fort, indem wir zunächst einmal einen quadratischen Alterseffekt fitten, und dann die Strauchdeckung mit ins Modell nehmen:

```
> fm2 <- glm(RODENTSP ~ AGE + I(AGE^2), binomial)
> summary(fm2)

Call:
glm(formula = RODENTSP ~ AGE + I(AGE^2), family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4847  -0.7321  -0.4897   0.6826   2.0519

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.561473   2.148216   2.123   0.0337 *
AGE         -0.234754   0.112223  -2.092   0.0365 *
I(AGE^2)     0.002080   0.001133   1.836   0.0664 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 34.617  on 24  degrees of freedom
```

```
Residual deviance: 24.677  on 22  degrees of freedom
AIC: 30.677

Number of Fisher Scoring iterations: 5
```

```
> lines(new$AGE, predict(fm2, new, type = "response"), lwd = 2,
+     col = "darkgreen", lty = 2)
```

Das neue Modell mit dem quadratischen Altersterm hat eine Glockenform (Abbildung 8.2; je nach Vorzeichen des quadratischen Terms nach oben oder nach unten geöffnet). Das sollten wir im Kopf behalten, wenn wir quadratische Effekt in einer logistischen Regression fitten.
Dieses zweite Modell passt etwas besser (siehe etwas niedrigere *deviance*), und der Unterschied ist signifikant:

```
> anova(fm1, fm2, test = "Chisq")

Analysis of Deviance Table

Model 1: RODENTSP ~ AGE
Model 2: RODENTSP ~ AGE + I(AGE^2)
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1        23    28.9231
2        22    24.6765  1   4.2466    0.0393
```

D.h., da `fm2` signifikant besser ist als `fm1`, bleiben wir bei dem (ökologisch zunächst wenig einleuchtenden) zweiten Modell.
Ziehen wir jetzt den Faktor `PERSHRUB` hinzu. Er beschreibt den Prozentsatz Geträuch in der Schlucht.

```
> fm3 <- glm(RODENTSP ~ PERSHRUB + AGE + I(AGE^2), binomial)

1: Algorithm did not converge in: (if (is.empty.model(mt)) glm.fit.null else
glm.fit)(x = X, y = Y,
2: fitted probabilities numerically 0 or 1 occurred in: (if
(is.empty.model(mt)) glm.fit.null else glm.fit)(x = X, y = Y,
```

An der Fehlermeldung erkennen wir, dass das Modell überfittet ist. In diesem Fall sind Strauchdeckung und (Alter^2) so stark korreliert, dass der Iterationsprozess die Parameter nicht richtig schätzen konnte. Lassen wir also `I(AGE^2)` weg, und probieren wir es nochmals:

```
> fm3 <- glm(RODENTSP ~ AGE + PERSHRUB, binomial)
> summary(fm3)

Call:
glm(formula = RODENTSP ~ AGE + PERSHRUB, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5737  -0.5507  -0.3082   0.4261   2.2185

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.56411    2.88732  -1.927   0.0540 .
AGE          0.02623    0.03636   0.721   0.4706
PERSHRUB     0.09368    0.03921   2.389   0.0169 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)
```

```
    Null deviance: 34.617  on 24  degrees of freedom
Residual deviance: 19.514  on 22  degrees of freedom
AIC: 25.514

Number of Fisher Scoring iterations: 5
```

Der ökologisch gut interpretierbare Faktor Strauchdeckung entzieht offensichtlich dem Alterseffekt seine Grundlage. Nehmen wir noch den letzten Faktor hinzu:

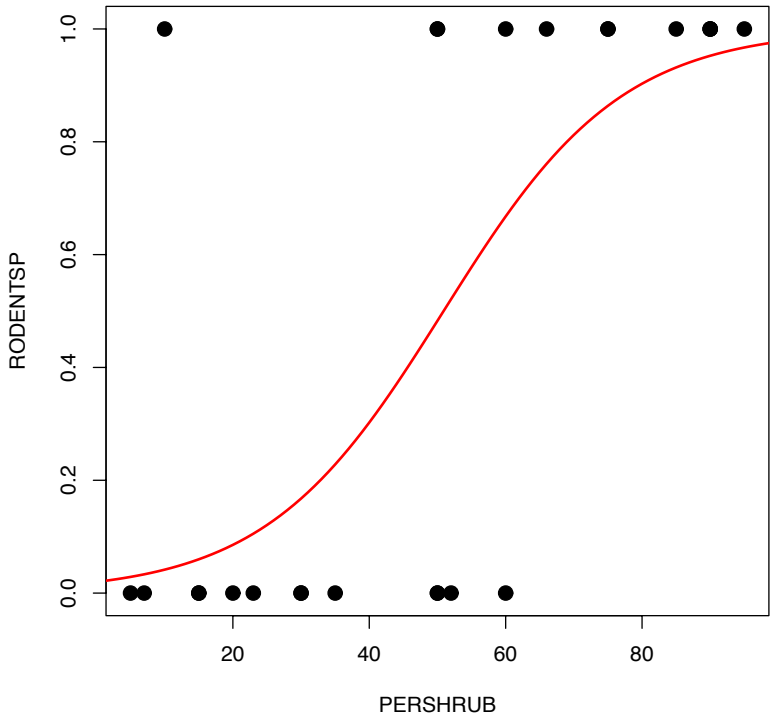


Abbildung 8.3.: Gemessene (Punkte) und vorhergesagte Werte für das Vorkommen/Fehlen von Nagetieren in Schluchten unterschiedlicher Strauchdeckung (PERSHRUB).

```
> fm4 <- glm(RODENTSP ~ AGE + PERSHRUB + DISTX, binomial)
> summary(fm4)

Call:
glm(formula = RODENTSP ~ AGE + PERSHRUB + DISTX, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5823  -0.5985  -0.2813   0.3699   2.1702

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.9099159  3.1125426  -1.899   0.0576 .
AGE          0.0250077  0.0376618   0.664   0.5067
PERSHRUB     0.0958695  0.0406119   2.361   0.0182 *
DISTX        0.0003087  0.0007741   0.399   0.6900
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 34.617  on 24  degrees of freedom
Residual deviance: 19.358  on 21  degrees of freedom
AIC: 27.358

Number of Fisher Scoring iterations: 5
```

Der Abstand zur nächsten mit Nagetieren belebten Schlucht besitzt keine erklärende Kraft. Wir können jetzt die `stepAIC`-Funktion benutzen, um uns das Modell vereinfachen zu lassen:

```
> fm5 <- stepAIC(fm4, trace = F)
> summary(fm5)

Call:
glm(formula = RODENTSP ~ PERSHRUB, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.4827  -0.6052  -0.2415   0.5421   2.5218

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.90342    1.55775  -2.506  0.01222 *
PERSHRUB     0.07662    0.02878   2.663  0.00775 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 34.617  on 24  degrees of freedom
Residual deviance: 20.049  on 23  degrees of freedom
AIC: 24.049

Number of Fisher Scoring iterations: 5
```

Es bleibt also nur die Strauchdeckung als Effekt erhalten. Dies sei dann in Abbildung 8.3 dargestellt.

```
> plot(RODENTSP ~ PERSHRUB, pch = 16, cex = 1.5)
> new <- data.frame(PERSHRUB = 1:100)
> lines(new$PERSHRUB, predict(fm5, new, type = "response"), lwd = 2,
+     col = "red")
```

Ein Wort noch zur emphoverdispersion an dieser Stelle. Da die *residual deviance* hier nicht größer ist als die Freiheitsgrade (20 vs. 23), liegt keine *overdispersion* vor, und wir nehmen die binomiale Verteilung als passend an.

Besonders für kategoriale Variablen ist es schwierig die Güte eines logistischen Regressionsmodells zu beschreiben. Menard (2000) schlägt ein Pendant zum r^2 des linearen Modells vor:

$$r_L^2 = 1 - \frac{\text{log-likelihood(fitted modelonly intercept-model)}}{\text{log-likelihood(only intercept-model)}}$$

Um dieses r_L^2 berechnen zu können, müssen wir unser Modell mit dem Modell ohne erklärende Variablen vergleichen. Wir benutzen aus der `summary(glm(.))` die *deviance* statt der *log-likelihood*[4]. 1 minus dem Quotient der beiden *residual deviances* ergibt dann unser neues r_L^2.
Im vorherigen Beispiel lassen sich die *deviances* der Residuen mit folgenden Befehlen extrahieren:

[4]Die *deviance* ist bis auf eine Konstant identisch der *log-likelihood*. Wir könnten aber auch die Funktion `logLik` benutzen, um die *log-likelihood* selbst aus einem GLM zu extrahieren.

```
> fm5$deviance

[1] 20.04929

> fm5$null.deviance

[1] 34.61735

> 1 - fm5$null.deviance/fm5$deviance

[1] -0.7266123
```

Unser Modell erklärt also etwa 42% der *deviance* der Daten.

Neben dem "R^2" der Modelle ist es bei binären Daten üblich, die Fähigkeit des Modells zur Diskriminierung zwischen 0 und 1 anzugeben. Auf eine ausführliche Herleitung und Diskussion verschiedener Maße sei hier verzichtet und auf Fielding and Bell (1997) oder Reineking and Schröder (2004a) verwiesen. Am gebräuchlichsten ist sicherlich die Fläche unter der *receiver-operator-characteristic curve* (ROC), kurz als AUC bezeichnet (*area under curve*). Sie liegt zwischen 0.5 (so schlecht wie raten) und 1 (perfekt). Modelle mit AUC-Werten unter 0.7 sollte man mit Skepsis betrachten.

In R gibt es mindestens ein halbes Dutzend Implementierungen des AUC. In **Hmisc** steht er unter `somers2` zur Verfügung, hier benutzen wir aber die Funktion `roc.area` aus **verification**. Mit `verify` stehen hier eine ganze Armada an Maßen zur Verfügung.

```
> library(verification)
> roc.area(obs = RODENTSP, pred = predict(fm5, type = "response"))

$A.tilda
[1] 0.8878205

$n.total
[1] 25

$n.events
[1] 12

$n.noevents
[1] 13

$U
[1] 17.5

$p
[1] 0.0004995746

$p.adj
[1] 0.0004796195
```

Der erste Wert, `$A.tilda`, gibt den gesuchten AUC-Wert an. Unser Modell ist also ganz gut.

Kodierung für Erfolge/Versuche-Daten

Der Ursprung binomialer Daten ist das berüchtigte Urnenmodell: Aus einer Urne mit roten und grünen Kugeln werden (ohne Zurücklegen) n Kugeln gezogen („Versuche"). Uns interessiert wieviele der Gezogenen sind rot (k, "Erfolge"). Daten dieser Art stammen etwa aus sog. Bioassays, in denen ein Organismus benutzt wird, um die Schädlichkeit einer Substanz

zu testen. Ein Beispiel wäre ein Versuch in dem jeweils 20 Essigfliegen *Drosophila* verschiedenen Dosen eines Insektizides ausgesetzt werden. Die Anzahl toter *Drosophila* ist dann der "Erfolg".

Der entsprechende Datensatz würde etwa so aussehen: $n = 20, 20, 20, 20$, $k = 4, 7, 13, 15$, Dosis $= 0.1, 0.2, 0.3, 0.4$ ml l^{-1}. Offensichtlich müssen wir hier der Software nicht nur die "Erfolge" mitteilen, sondern auch die Anzahl der Versuche. Stellen wir uns folgende Situation vor: Aus Mangel an *Drosophila*s werden in den Versuchen unterschiedlich viele Individuen benutzt. Unser neuer Datensatz sieht so aus: $n = 14, 8, 12, 6$, $k = 4, 5, 9, 5$, Dosis $= 0.1, 0.2, 0.3, 0.4$ ml l^{-1}. Jetzt ist ein möglicher Effekt der Insektizides nicht mehr ohne weiteres aus den Daten ersichtlich.

Beginnen wir mit der ersten Datensatz. In R fügen wir die Erfolge und die **Misserfolge** zu einer zweispaltigen Antwortvariablen zusammen (durch den Befehl `cbind`[5]).

```
> Versuche <- rep(20, 4)
> Erfolge <- c(4, 7, 13, 15)
> Dosis <- seq(0.1, 0.4, 0.1)
> y <- cbind(Erfolge, Versuche - Erfolge)
> fm <- glm(y ~ Dosis, binomial)
> summary(fm)

Call:
glm(formula = y ~ Dosis, family = binomial)

Deviance Residuals:
        1          2          3          4
-0.008547  -0.251709   0.519564  -0.304389

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -2.2613     0.6555  -3.450 0.000561 ***
Dosis         8.7982     2.4010   3.664 0.000248 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 16.54831  on 3  degrees of freedom
Residual deviance:  0.42603  on 2  degrees of freedom
AIC: 17.429

Number of Fisher Scoring iterations: 3
```

Der Effekt des Insektizids ist also hochsignifikant. Das Verhältnis von *residual deviance* und Freiheitsgraden ist etwas unausgewogen (0.42/2=0.21), was auf *underdispersion* hindeutet. Für nur vier Datenpunkte ist dies kein Problem, oder? Wir können mittels einer `anova(., type="Chisq")`-Abfrage die Signifikanz der Effekte testen[6]:

```
> anova(fm, test = "Chisq")

Analysis of Deviance Table

Model: binomial, link: logit
```

[5] was soviel wie *column bind*, also "verknüpfe Spalten", bedeutet

[6] In S-plus führt die Wahl des F-Tests in der Funktion `anova` zu einer Berücksichtigung *dispersion*. Dies geschieht in R nicht. In R benutzen wir den X^2-Test mit `anova`, außer bei normalverteilten Daten. Liegt *overdispersion* vor, so müssen wir auf die `quasi`-Familien ausweichen oder die `Anova(., test="F")`-Funktion bemühen, die, wie S-plus, auf *overdispersion* korrigiert.

```
Response: y

Terms added sequentially (first to last)


      Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                      3    16.5483
Dosis  1  16.1223         2     0.4260    0.0001

> Anova(fm, test = "F")

Anova Table (Type II tests)

Response: y
              SS Df      F  Pr(>F)
Dosis     16.1223  1 76.127 0.01288 *
Residuals  0.4236  2
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Wir sehen, dass der F-Test niedrigere, wenn auch noch stets signifikante Werte für die Effekte herausgibt. Alternativ (und statistisch befriedigender) können wir diese Frage anders beantworten: Wir geben R als Form der Verteilung `quasibinomial` an, in der das Verhältnis von *residual deviance* und Freiheitsgraden als *dispersion* in die Berechnung mit hineingenommen.

```
> fm2 <- glm(y ~ Dosis, quasibinomial)
> anova(fm2, test = "F")

Analysis of Deviance Table

Model: quasibinomial, link: logit

Response: y

Terms added sequentially (first to last)


      Df Deviance Resid. Df Resid. Dev      F  Pr(>F)
NULL                      3     16.548
Dosis  1   16.122         2      0.426 76.128 0.01288 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Der p-Wert ist jetzt identisch dem der vorigen `Anova`-Variante, d.h. die *dispersion* wird in beiden Fällen angemessen berücksichtigt.
Wie gemutmaßt ist ein *deviance*/Freiheitsgrade-Verhältnis von 0.2 bei nur 4 Datenpunkten im „grünen Bereich“. Dabei sehen wir aber, dass in der Tat die p-Werte sich um nahezu zwei Größenordnungen zum konservativen verschoben haben. Ein weniger eindeutiger Datensatz mag also durch eine *underdispersion* deutlicher verändert werden.
Wenden wir uns jetzt dem zweiten Datensatz zu:

```
> Versuche2 <- c(14, 8, 12, 6)
> Erfolge2 <- c(4, 5, 9, 5)
> y2 <- cbind(Erfolge2, Versuche2 - Erfolge2)
> fm2 <- glm(y2 ~ Dosis, binomial)
> summary(fm2)
```

```
Call:
glm(formula = y2 ~ Dosis, family = binomial)

Deviance Residuals:
       1         2         3         4
-0.27728   0.48963   0.04405  -0.32529

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.6658     0.8177  -2.037   0.0416 *
Dosis         9.1170     3.5687   2.555   0.0106 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 8.30903  on 3  degrees of freedom
Residual deviance: 0.42438  on 2  degrees of freedom
AIC: 14.425

Number of Fisher Scoring iterations: 4

> anova(fm2, test = "Chi")

Analysis of Deviance Table

Model: binomial, link: logit

Response: y2

Terms added sequentially (first to last)


      Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                      3     8.3090
Dosis  1   7.8847         2     0.4244    0.0050

> fm3 <- glm(y2 ~ Dosis, quasibinomial)
> anova(fm3, test = "Chi")

Analysis of Deviance Table

Model: quasibinomial, link: logit

Response: y2

Terms added sequentially (first to last)


      Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                      3     8.3090
Dosis  1   7.8847         2     0.4244 1.398e-09
```

Und hier **erhöht** die Korrektur für *underdispersion* die Signifikanz des Effektes.

Wenn wir mit GLMs arbeiten und uns für die Signifikanz eines Terms interessieren, führen wir folgende Berechnungen durch. Wir vergleichen zwei unterschiedliche Modelle: das mit dem Term, und das identische ohne den Term. Wenn nur eine erklärende Variable vorhanden ist, wie im letzten Beispiel die Dosis des Insektizid, so ist dieser Test einfach der Vergleich

des Modells mit dem "leeren" Modell nur mit y-Achsenabschnitt. Bei mehreren erklärenden Variablen müssen wir erst ein um den entsprechenden Term reduziertes Modell schaffen.

Dies sei mit den bereits oben geladenen Beispieldaten von Bolger et al. vorgeführt.

```
> names(bolger)

[1] "PERSHRUB" "DISTX"    "AGE"      "RODENTSP"

> bolfm1 <- glm(RODENTSP ~ AGE + PERSHRUB, binomial)
> bolfm2 <- glm(RODENTSP ~ AGE, binomial)
> bolfm3 <- glm(RODENTSP ~ PERSHRUB, binomial)
> anova(bolfm1, bolfm2, test = "Chi")

Analysis of Deviance Table

Model 1: RODENTSP ~ AGE + PERSHRUB
Model 2: RODENTSP ~ AGE
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1        22    19.5135
2        23    28.9231 -1  -9.4096    0.0022

> anova(bolfm1, bolfm3, test = "Chi")

Analysis of Deviance Table

Model 1: RODENTSP ~ AGE + PERSHRUB
Model 2: RODENTSP ~ PERSHRUB
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1        22    19.5135
2        23    20.0493 -1  -0.5358    0.4642
```

Der Unterschied zwischen Modell `bolfm1` und `bolfm2` ist signifikant, d.h. die Eliminierung von `PERSHRUB` reduziert die Erklärungskraft des Modells signifikant. Hingegen beeinflusst die Eliminierung von `AGE` das Modell nicht bedeutend. Beachte auch, dass alle drei Modelle ein akzeptables Verhältnis von *deviance* und Freiheitsgraden zeigen (0.88, 1.26, 0.87, respektive).

```
> anova(bolfm1, bolfm2, test = "Chi")

Analysis of Deviance Table

Model 1: RODENTSP ~ AGE + PERSHRUB
Model 2: RODENTSP ~ AGE
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1        22    19.5135
2        23    28.9231 -1  -9.4096    0.0022
```

Die Elimination von AGE führte in beiden Fällen zu einer Veränderung in der *deviance* (für `anova`) bzw. *log-likelihood* (für `anova`) von 5.7. Da *deviance*-Verhältnisse (*deviance ratio*) ebenso wie *log-likelihood-ratios* X^2-verteilt sind, ist der damit assoziierte p-Wert 0.46.
Wir können also entweder per Hand die Effekte eliminieren und mittels `anova` die Ergebnisse vergleichen, oder wir lassen `anova` diese Arbeit tun.

8.2. Log-lineare Modelle: Poisson Regression

Log-lineare Modelle sind in GLMs für Poisson-verteilte Daten zuständig. Wir erinnern uns, dass vor allem Zähldaten zu einer Poisson-Verteilung führen. Wenn wir also Daten erheben, in denen bspws. die Anzahl Kaffee-trinkender Kollegen in Abhängigkeit vom Alter analysiert

werden sollen, so geschieht dies mittels log-linearer Modelle. Der Name "log-linear" rührt aus der *link*-Funktion, die für Poisson-Daten standardmäßig der *log link* ist.

Im Grunde ist die Analyse Poisson-verteilter Daten vollständig analog zu den gerade besprochenen binomialverteilten Daten. Auch hier spezifizieren wir im GLM die Fehlerverteilung (und damit den *link*), achten bei den Ergebnissen auf die *dispersion*, und selektieren unser Modell bei Modellvereinfachung mittels eines *deviance*-Tests.

Untersuchen wir im folgenden die Frage, ob Halsbandschnäpperweibchen durch ihre Wahl eines attraktiveren Männchens auch die Befütterung ihrer gemeinsamen Brut verbessern. Dafür betrachten wir die Anzahl Futterstücke, die ein Halsbandschnäppermännchen seinen Jungen pro Stunde ans Nest liefert, in Abhängigkeit von der Attraktivität des Männchens. (Die Auswertung, die für die Erstellung der Linien nötig ist wird gleich behandelt.)

```
> schnaepper <- read.table("schnaepper.txt", header = T)
> attach(schnaepper)
> names(schnaepper)

[1] "stuecke" "attrakt"

> par(mfrow = c(1, 2), mar = c(5, 5, 1, 1))
> plot(log(stuecke) ~ attrakt, cex = 2, cex.lab = 1.7, pch = 16)
> points(seq(1, 5, by = 0.1), predict(glm(stuecke ~ attrakt, poisson),
+     list(attrakt = seq(1, 5, by = 0.1))), type = "l", lwd = 3)
> plot(stuecke ~ attrakt, cex = 2, cex.lab = 1.7, pch = 16)
> points(seq(1, 5, by = 0.1), predict(glm(stuecke ~ attrakt, poisson),
+     list(attrakt = seq(1, 5, by = 0.1)), type = "response"), type = "l",
+     lwd = 3)
```

Hat entsprechend dieser Daten nun die Attraktivität einen Einfluss auf die Versorgungsrate? Offensichtlich handelt es sich um Poisson-verteilte Daten, und entsprechend sind nur ganzzahlige Werte auf der y-Achse aufgetragen. Dadurch wirkt die Streuung sehr groß. Die Frage nach einer Signifikanz des Attraktivitätseffekts gehen wir wie folgt nach:

An diesem Beispiel wollen wir einmal den *maximum likelihood*-Ansatz zu Fuß durchrechnen. Für die Poisson-Verteilung hat die *likelihood*-Funktion folgende Form:

$$\mathscr{L}(x_i|\lambda) = \frac{\lambda^{x_1}}{x_1!e^{\lambda}} \cdot \frac{\lambda^{x_2}}{x_2!e^{\lambda}} \cdots \frac{\lambda^{x_n}}{x_n!e^{\lambda}} = \frac{\lambda^{\sum x_n}}{x_1! \cdots x_n!e^{n\lambda}}$$

Wir logarithmieren und erhalten die *log-likelihood*:

$$ln\mathscr{L} = \sum (-\lambda + (ln\lambda) \cdot x_i) - ln(\prod x_i!)$$

Der erste Schritt ist die Formulierung der Regressionsgleichung: $y = \beta_0 + \beta_1 x$. Wir sehen, dass wir zwei Parameter gleichzeitig berechnen müssen (die beiden βs). Nehmen wir der einfachheitshalber zunächst an, wir wüssten, dass der y-Achsenabschnitt den Wert 4.4 hätte. Dann wäre unsere Regressionsgleichung: $y = 4.4 + \beta_1 x$. Mit dieser können wir jetzt für jeden beobachteten y-Wert (d.i. `stuecke`) eine Wert aus der Attraktivität vorhersagen, wenn wir für β_1 einen Wert vorgeben. So sind für $\beta_1 = 1$ die beobachtete und vorhergesagten x-Werte:

```
> stuecke

 [1]  3  6  8  4  2  7  6  8 10  3  5  7  6  7  5  6  7 11  8 11 13 11  7  7  6

> 4.4 + attrakt * 1

 [1] 5.4 5.4 5.4 5.4 5.4 6.4 6.4 6.4 6.4 6.4 7.4 7.4 7.4 7.4 7.4 8.4 8.4 8.4 8.4
[20] 8.4 9.4 9.4 9.4 9.4 9.4
```

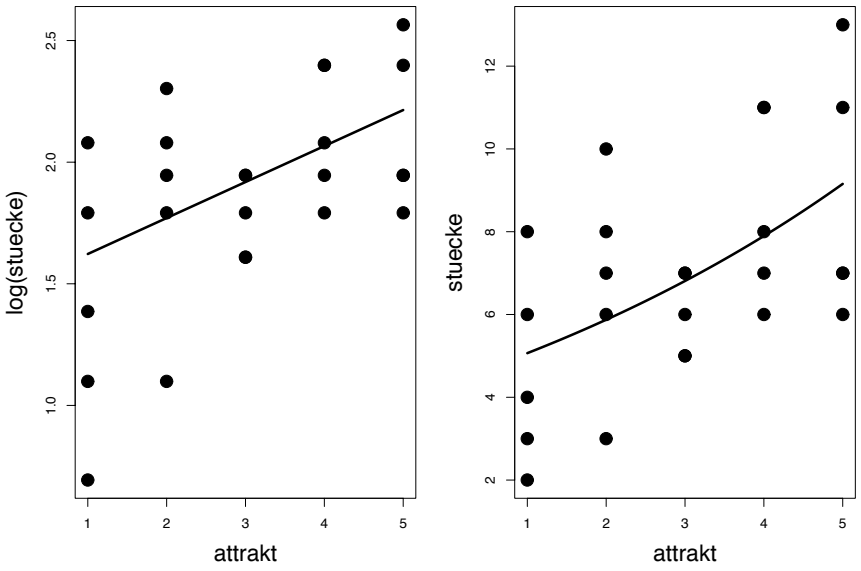


Abbildung 8.4.: Vom Halsbandschnäppermännchen an die Brut gelieferte Anzahl Futterstücke pro Stunde in Abhängigkeit der Attraktivität des Männchens, links mit logarithmisch dargestellten Anzahlen, recht normal. Die durchgezogenen Linien stellt das Ergebnis einer Poisson-Regression dar, links auf der *link-scale*, rechts rücktransformiert. Beachte, dass aus dem Regressionsgeraden mit *log-link* eine e-Funktion auf der normalen (=*response*) Skala wird. Dies mag manchmal ungewohnt erscheinen, und so werden die meisten Daten, die mittels Poisson-Regression analysiert wurden, auch als logarithmiert dargestellt. Nichtsdestotrotz ist die rechte Abbildung genauso korrekt.

```
> points(seq(1, 5, by = 0.1), 4.4 + seq(1, 5, by = 0.1), type = "l")
```

Bei der klassichen Regression normalverteilter Daten würden wir jetzt die Residuen berechnen, quadrieren und aufsummieren. Für Poisson-Daten berechnen wir aber nicht die *sum of squares*, sondern gemäß obiger Formel die *log-likelihood* der Daten, wobei wir λ durch $4.4+$`attrakt`$\cdot 1$ersetzen.

Für jeden Datenpunkt x_i können wir jetzt die Wahrscheinlichkeit berechnen, dass er einer Poisson-Verteilung mit einem bestimmten Mittelwert λ_i entstammt. Diese wird in R berechnet als `dpois(`x_i`,`λ_i`)`. Diese logarithmieren wir und summieren sie auf. Damit erhalten wir unsere *log-likelihood*. Beachte, dass wir ja die beobachteten Werte mit dem *log-link* an das Modell koppeln!

```
> sum(log(dpois(stuecke, exp(1.5 + attrakt * 0.2))))

[1] -59.86768
```

Das Ergebnis ist stets eine negative Zahl, da ja die Wahrscheinlichkeiten zwischen 0 und 1 liegen, der log davon also negativ ist (und somit auch die Summe der logs).

Wiederholen wir dies für eine Reihe an Werte für β_1, etwa von 0.01 bis 0.5, und bilden die *log-likelihood*-Summen ab, so erhalten wir Abb. 8.5.

```
> loglik <- 1:50
> beta1 <- seq(0.01, 0.5, len = 50)
> for (i in 1:50) loglik[i] <- sum(log(dpois(stuecke, exp(1.5 +
+     attrakt * beta1[i]))))
> plot(beta1, loglik, type = "l", xlab = expression(beta[1]), cex.lab = 1.5)
```

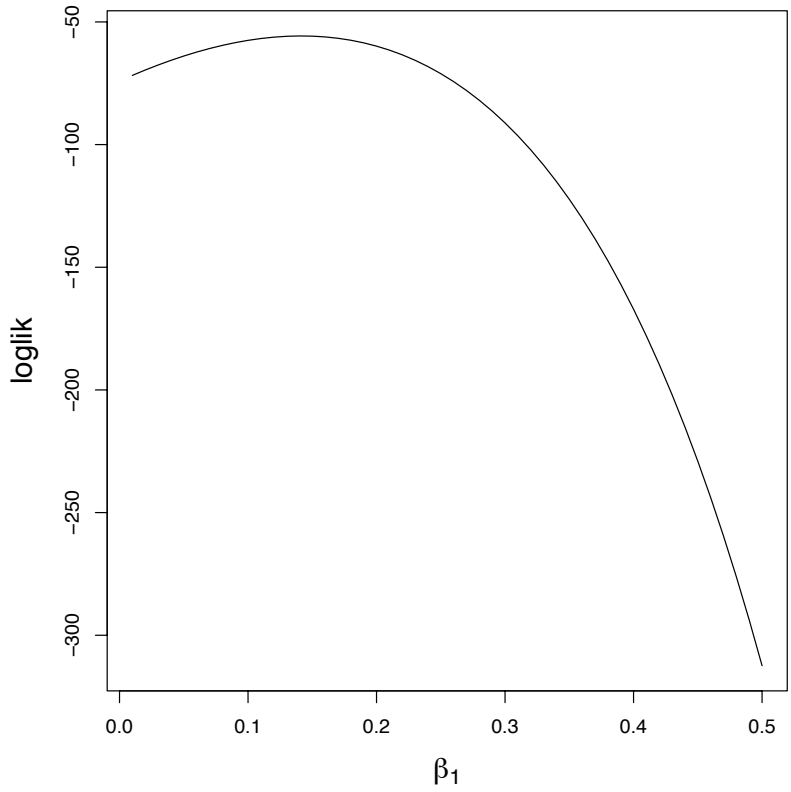


Abbildung 8.5.: *Log-likelihood* der verschiedenen Werte von β_1 bei gegebenem $\beta_0 = 4.4$.

Offensichtlich gibt es einen Wert für β_1, für den `loglik` maximal ist. Dies ist der gesuchte Wert für β_1 (in diesem Fall etwa 0.15), denn er produziert die maximale Wahrscheinlichkeit.

Entsprechend können wir vorgehen, wenn wir den y-Achsenabschnitt berechnen wollen. Interessant wird es, wenn wir beide Parameter *gleichzeitig* schätzen müssen. Das Ergebnis ist dann eine dreidimensionale Fläche, die einem Berg ähneln sollte. Beginnen wir, indem wir für Steigung und y-Achsenabschnitt 100 Werte von 0.01 bis 0.5 bzw. 0.1 bis 2 wählen, und die Poisson-*likelihood* für jede Kombination dieser Werte berechnen[7]. Anschließend plotten wir das Ergebnis (Abb. 8.6), einmal dreidimensional mittels `persp`, dann besser erkennbar zweidimensional mittels `contour`[8].

```
> beta0 <- seq(0.1, 2, length = 100)
> beta1 <- seq(0.01, 0.5, length = 100)
> llfun <- function(x, y) {
+     sum(log(dpois(stuecke, exp(x + attrakt * y))))
+ }
> loglik.m <- matrix(ncol = 100, nrow = 100)
> for (i in 1:100) {
+     for (j in 1:100) {
+         loglik.m[i, j] <- llfun(beta0[i], beta1[j])
+     }
+ }
> par(mfrow = c(1, 2))
> persp(beta0, beta1, loglik.m, phi = 30, theta = 90, xlab = "beta0",
+     ylab = "beta1", zlab = "loglik")
> contour(beta0, beta1, loglik.m, nlevels = 30, xlab = expression(beta[0]),
+     ylab = expression(beta[1]), cex.lab = 1.5)
```

[7] Hier ginge es mit der Funktion `outer` mit weniger Code, aber die `for`-Schleifen sind didaktisch klarer.
[8] Hiervon gibt es eine Variante `filled.contour`, die bunte Farbverläufe statt Konturen benutzt.

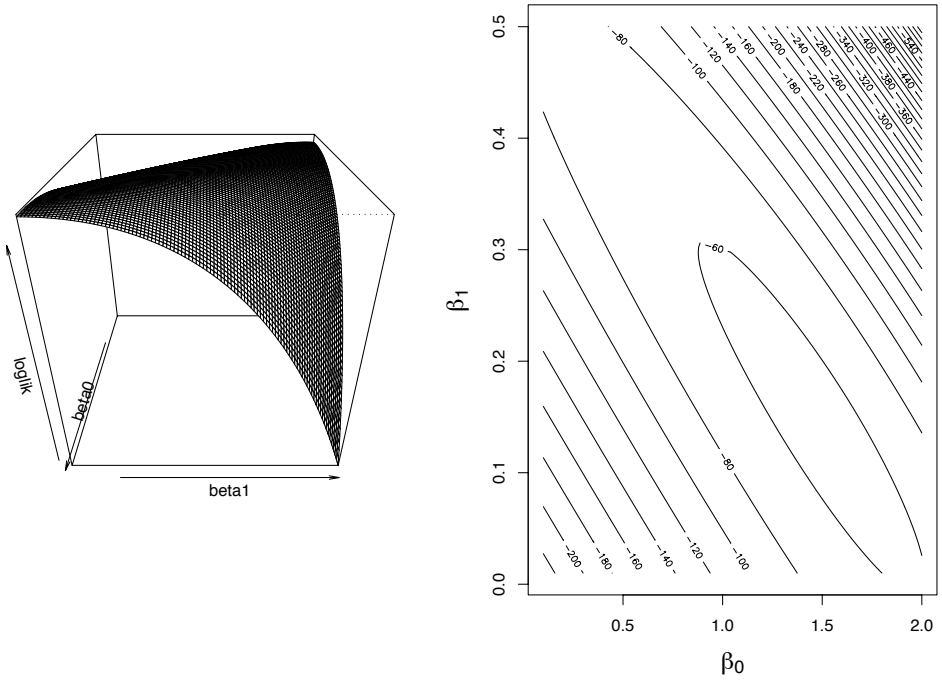


Abbildung 8.6.: 3D- und Konturenabbildung der *log-likelihood*-Berechnung für die Schnäpperdaten. Wir sehen deutlich, dass die Koeffizienten nicht unabhängig voneinander sind: je höher der Wert des einen, desto niedriger der des anderen. Für die Korrelation zwischen Achsenabschnitt und Steigung ist dies praktisch immer der Fall. Sorgen sollten wir uns machen, wenn zwei Faktoren, also zwei Steigungen, diese Bild aufweisen. Dann können wir den Effekt der einen Variablen nicht von dem der anderen trennen (*collinearity*).

Jetzt wollen wir natürlich noch die Werte für β_0 und β_1 haben. Um diese zu extrahieren benutzen wir die Funktion `which.max`:

```
> loglik.m[which(loglik.m == max(loglik.m))]

[1] -55.71565

> llmax <- which(loglik.m == max(loglik.m), arr.ind = T)
> round(beta0[llmax[1]], 2)

[1] 1.48

> round(beta1[llmax[2]], 2)

[1] 0.14
```

Die gesuchten Werte sind also $\beta_0 = 1.48$ und $\beta_0 = 0.14$.
Wenn wir diese Werte mit `glm` errechnen wollen, so müssen wir nur die Fehlerverteilung als Poisson definieren:

```
> summary(glm(stuecke ~ attrakt, poisson))

Call:
glm(formula = stuecke ~ attrakt, family = poisson)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
```

```
-1.55377  -0.72834   0.03699   0.59093   1.54584

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.47459    0.19443   7.584 3.34e-14 ***
attrakt      0.14794    0.05437   2.721  0.00651 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 25.829  on 24  degrees of freedom
Residual deviance: 18.320  on 23  degrees of freedom
AIC: 115.42

Number of Fisher Scoring iterations: 4
```

Das ist doch überzeugend. Und natürlich hat der Optimierungsalgorithmus im GLM eine viel höhere Genauigkeit als unser handgestrickter *grid-search*.

Dieses Beispiel hat uns gezeigt, wie zwei Parameter gleichzeitig geschätzt werden können. Wir haben dabei auch gesehen, dass die Fehler auf diesen Parametern sehr unterschiedlich sein können. Je flacher die Spitze des *likelihood*-Berges, desto größer ist der Fehler auf dem geschätzten Koeffizienten. In unserem Fall hatten wir eine Bergschulter; die Steigung war gut, der Achsenabschnitt schlechter zu schätzen. Wenn wir uns Abb. 8.4 anschauen, dann sehen wir, dass eine Zunahme der y-Werte mit ansteigenden x-Werten in der Tat sehr offensichtlich ist. Den *genauen* Schnittpunkt der Regressionsgraden mit der y-Achse hingegen können wir nicht wirklich abschätzen.

8.3. Verallgemeinerte Lineare Gemischte Modelle (GLMM)

Hier sei nur kurz erwähnt, dass es gemischte Effekte nicht nur für normalverteilte Daten gibt (Abschnitt 7.3.2 auf Seite 131). Tatsächlich erfolgt die Berechnung hier noch nicht einmal durch die *maximum likelihood*, sondern durch eine Schätzung derselben. Das dafür verwandte Verfahren nennt sich *penalised quasi-likelihood*, kurz PQL. Auf die Details einzugehen führte hier zu weit (siehe Venables and Ripley 2002, für weitere Details).

Ein Problem mit der Benutzung von PQL ist, dass das *likelihood-ratio* verschiedener Modelle nicht mehr benutzt werden kann, um Modellselektion zu betreiben. Unseres Wissens ist dieses Problem noch immer ungelöst, und wir benutzen also weiterhin **fälschlicherweise** den X^2-Test à la `anova(., test="Chisq")`.

Die Funktion, die in R Verallgemeinerte Lineare Gemischte Modelle implementiert liegt erst seit kurzer Zeit vor (Venables and Ripley 2002). Sie greift auf die `lme`-Funktion zurück, die wir bereits kennengelernt haben (Abschnitt 7.3.2 auf Seite 131). Für GLMMs lautet die Funktion `glmmPQL`. Wir benutzen sie zur Analyse eines genesteten Experiments mit binärer Antwortvariabler. Die Beispielrechnung dazu ist in Abschnitt 13.3.5 auf Seite 222 vorgeführt.

9. Nicht-parametrische Regression: loess, splines and GAM

Bisher haben wir uns damit auseinandergesetzt, wie die Antwortvariablen auf die Werte bzw. Level von erklärenden Variablen reagieren: Wir berechneten die Koeffizienten für einen bestimmten Effekt. Dabei setzen wir eine bestimmte Form der Abhängigkeit voraus. Entweder wirken die Effekte linear oder quadratisch. Durch Transformation der erklärenden Variablen können wir andere erwartete Funktionen anpassen.

In diesem Kapitel geht es darum *lokale* Annäherungen an die beobachteten Daten auf Basis der Werte der erklärenden Variablen zu benutzen. Dadurch können wir die Daten besser untersuchen, z.B. auf die zugrundeliegende Form des Zusammenhangs zwischen abhängiger und unabhängiger.

Die Methodik dafür basiert darauf, dass wir die zu erklärende Variable immer nur auf einem Ausschnitt der x-Achse betrachten. Dieses Betrachtungsfenster verschieben wir dann über den gesamten Bereich der x-Werte. Im englischen Sprachraum bezeichnet man dieses Verfahren als *moving window, running mean* oder *smoothing*, wobei jeder dieser Begriffe für etwas leicht anderes steht.

Berechnen wir den Mittelwert für die Daten eines Fensters, dann können wir den weiter am Rand des Fensters liegenden Werten weniger Gewicht geben, als den in der Mitte. Die bezeichnet man als lokale Gewichtung (*local weighting*). Die Stärke dieser Gewichtung bestimmt natürlich maßgeblich den berechneten Wert: Haben alle Punkte das gleiche Gewicht ($= 1$), so haben wir unseren *running mean*. Üblicher ist es das Gewicht (w_i) mit der Distanz abnehmen zu lassen,und zwar entsprechend folgender Formel:

$$w_i \propto (1 - (dist/Spanne)^3)^3 \tag{9.1}$$

Wir erahnen, dass die Breite des Fensters (Spanne, *span*, α) einen starken Einfluss auf das Ergebnis haben wird. Je weiter die Spanne, desto größer der Punktebereich, über den lokal gefittet wird. Je enger das Fenster, desto genauer werden die gemessenen Punkte angeglichen, aber desto "zackeliger" wird auch die Regressionslinie. Häufig genutzte Werte für α sind 2/3 oder 3/4 des Wertebereichs.

Am besten wird dies vielleicht an einem Beispiel deutlich. Dieses stammt aus Venables and Ripley (1994) und stellt die Beschleunigung des Kopfes einer Testpuppe in den ersten 50 Millisekunden nach einem Motorradunfall dar (`?mcycle` für weitere Informationen und Literaturhinweis). Wir laden die Daten und plotten sie erst einmal. Dann legen wir eine polynomische Regression dritter Ordnung hindurch, und danach einen *loess-smoother*[1] mit zwei verschiedenen Fensterweiten (Grundeinstellung = 0.75 und 0.2).

```
> library(MASS)
> data(mcycle)
> attach(mcycle)
> par(mfrow = c(1, 3), mar = c(2, 3, 2, 0), oma = c(3, 3, 0, 0.5))
> plot(accel ~ times, pch = 16, main = "Polynomische Regression")
> lines(times, fitted(lm(accel ~ poly(times, 3))))
> lines(times, fitted(lm(accel ~ poly(times, 5))), lty = 5)
> legend(30, -110, c("dritter Ordnung", "fünfter Ordnung"), lty = c(1,
+     5), cex = 1.1, bty = "n", y.intersp = 1.25)
> plot(accel ~ times, pch = 16, main = "loess-Regression")
```

[1] loess (oder auch lowess): *locally weighted scatterplot smooting*

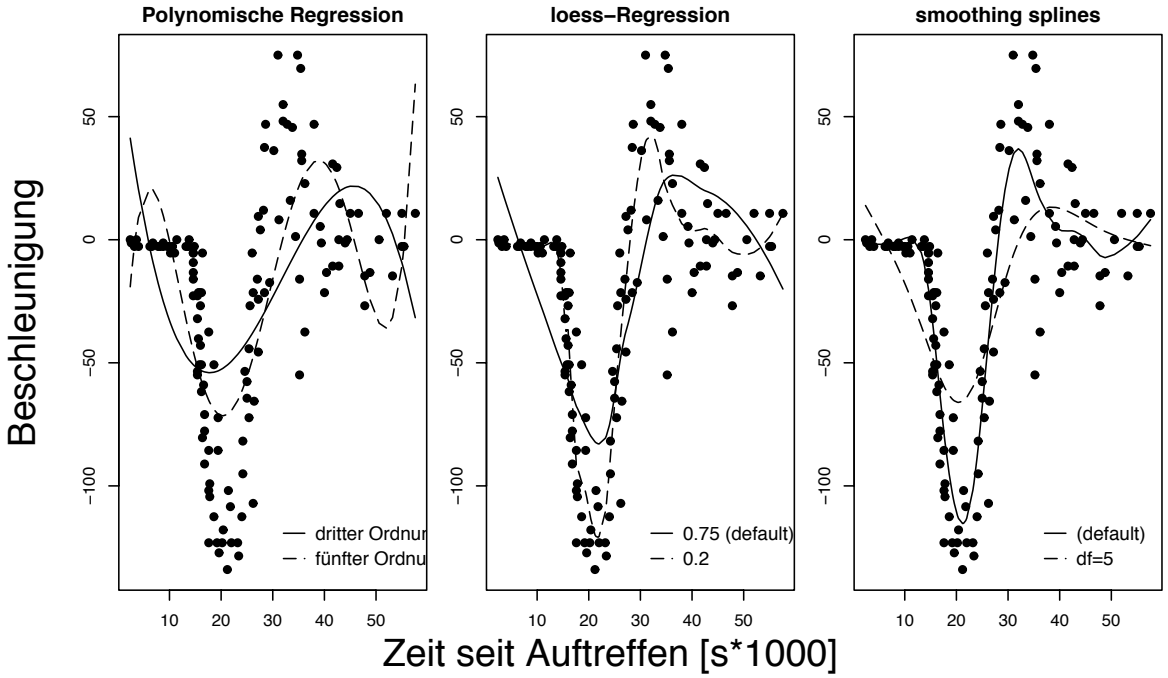


Abbildung 9.1.: Beschleunigung nach Aufprall als Beispiel für einen Datensatz, der mit drei verschiedenen Regressionen beschrieben werden soll. Links sind eine polynomische Regression 3. und 5. Ordnung dargestellt; in der Mitte *loess*-Regression mit Spanne 0.75 und 0.2; und rechts ein quadratischer *smoothing spline* mit 5 und 12 Knoten (letztere errechnen sich aus den Daten wenn wir die Grundeinstellung von `loess` benutzen; durch `smooth.spline(accel~times)` können wir sie uns anzeigen lassen).

```
> lines(times, fitted(loess(accel ~ times)))
> lines(times, fitted(loess(accel ~ times, span = 0.2)), lty = 5)
> legend(30, -110, c("0.75 (default)", "0.2"), lty = c(1, 5), cex = 1.1,
+     bty = "n", y.intersp = 1.25)
> plot(accel ~ times, pch = 16, main = "smoothing splines")
> lines(smooth.spline(times, accel))
> lines(smooth.spline(times, accel, df = 5), lty = 5)
> legend(35, -110, c("(default)", "df=5"), lty = c(1, 5), cex = 1.1,
+     bty = "n", y.intersp = 1.25)
> mtext("Zeit seit Auftreffen [s*1000]", side = 1, outer = T, line = 1,
+     cex = 1.5)
> mtext("Beschleunigung", side = 2, outer = T, line = 1, cex = 1.5)
> detach(mcycle)
```

Schließlich gibt es noch die *smoothing splines*. Der Datensatz wird zunächst durch sogenannte Knoten (*knots*) in Teilstücke zerlegt. Die Knotenzahl kann man entweder vorgeben, oder automatisch aus den Daten berechnen lassen. Zwischen zwei Knoten wird dann eine quadratische Regression durchgeführt. An den Knoten selbst werden diese dann wieder zusammengefügt. Je mehr Knoten wir einem Datensatz zubilligen, desto näher werden die Daten gefittet, aber desto trivialer wird auch die Regression. Deshalb geht die Knotenzahl negativ in ein Gütekriterium ein, dass den Fit bewertet (GCV = *Generalised Cross Validation*). Durch mehrfaches Ausprobieren der Knotenzahl wird dann der "optimale" *smoothing spline* bestimmt. Ein Beispiel ist in Abb. 9.1 rechts gegeben.

Im Vergleich der drei Verfahren von Abb. 9.1 zeigen sich *loess* und *spline* der polynomischen Regression deutlich überlegen. Und damit haben sie auch ihre Daseinsberechtigung. Allerdings

deutet die Wahlmöglichkeit der Parameter wie Spanne und Knotenzahl an, dass *loess* und *smoothing splines* eher explorative Werkzeuge sind, oder dann zum Tragen kommen, wenn wir wirklich keine Funktion für den Zusammenhang zwischen erklärender und abhängiger Variable formulieren können.

Beachte dass `loess` und `smooth.spline` genauso angewendet werden wie `lm`, d.h. durch Spezifizieren einer Formel. Die ältere Funktion `lowess` hingegen wird anders geplottet (`line(lowess(accel, times))`) und besitzt auch andere Grundeinstellungen (etwa `span=0.67`).

9.0.1. Mehr als eine erklärende Variable

Polynomische Regressionen und *loess*-Regressionen können wir auch mit mehr als einer erklärenden Variablen durchführen. Wir können uns einen Datensatz vorstellen, der durch eine unterliegende Variable einen systematischen Trend erhält. Beispiel hierfür wäre etwa die Dichte einer Lemmingpopulation entlang eines geographischen Gradienten. Wenn wir diesen Gradienten herausrechnen wollen, so können wir Längen- und Breitengrad als Variable fitten. Dies sei hier nur als Beispiel für eine multiple *loess*-Regression angebracht, nicht als Beispiel für räumlich Statistik!

Auch hier steht uns ein schöner Datensatz in R zur Verfügung: `topo` (aus **MASS**). An 52 Punkten in einer Fläche von 6.5 $\times$ 6.5 km wurde mittels Fallen die Lemmingzahl/ha gemessen.

Zunächst fitten wir eine *trend surface* auf Basis eines 2. Ordnung Polynoms mit den Variablen `lat` und `long`. Dann eine *loess*-Regression mit etwas schmalerer Spanne. Schließlich lassen wir uns zum Vergleich diese *trend surfaces* ausdrucken. Dafür sind einige Optionen nötig, die über die Hilfe für die jeweilige Funktion erklärt werden (Sichtwinkel, Schattierung, usw.).

```
> library(MASS)
> data(topo)
> colnames(topo) <- c("long", "lat", "lemming")
> flm <- lm(lemming ~ long * lat + I(long^2) * I(lat^2), topo)
> floe <- loess(lemming ~ long * lat, topo, span = 0.4)
> new <- expand.grid(long = seq(0, 6.5, len = 20), lat = seq(0,
+     6.5, len = 20))
> par(mfrow = c(1, 2), mar = c(2, 1, 1, 0), oma = c(0, 0, 0, 0))
> persp(z = matrix(predict(flm, new), ncol = 20), theta = 180,
+     xlab = "long", ylab = "lat", zlab = "Lemminge", main = "polynomisch",
+     shade = 0.5)
> persp(z = predict(floe, new), theta = 180, xlab = "long", ylab = "lat",
+     zlab = "Lemminge", main = "loess", shade = 0.5)
```

Neben dem visuellen Eindruck, der nahelegt, dass die *loess-trend surface* stärker den Daten angepasst wurde, als die polynomische, können wir uns auch Angaben zum *loess*-Modell machen lassen.

```
> floe

Call:
loess(formula = lemming ~ long * lat, data = topo, span = 0.4)

Number of Observations: 52
Equivalent Number of Parameters: 16.1
Residual Standard Error: 19.04

> summary(flm)

Call:
lm(formula = lemming ~ long * lat + I(long^2) * I(lat^2), data = topo)
```

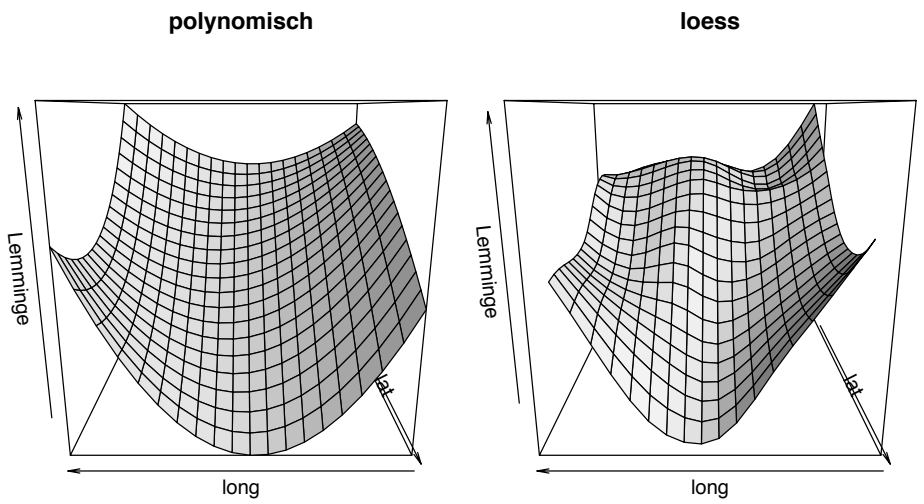


Abbildung 9.2.: Beispiel einer polynomischen (links) und *loess-trend surface* (rechts). Letztere zeichnet das Datenprofil deutlich stärker nach.

```
Residuals:
    Min      1Q  Median      3Q     Max
-73.166 -14.676  -4.334  16.116  88.116

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        933.94940   26.99823  34.593  < 2e-16 ***
long               -28.18114   14.29351  -1.972  0.05482 .
lat                 -7.09786   13.44023  -0.528  0.60002
I(long^2)            4.80830    1.67013   2.879  0.00609 **
I(lat^2)            -1.50105    1.60147  -0.937  0.35361
long:lat            -6.88946    3.35043  -2.056  0.04558 *
I(long^2):I(lat^2)   0.17531    0.07647   2.292  0.02661 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 28.2 on 45 degrees of freedom
Multiple R-Squared: 0.8175,        Adjusted R-squared: 0.7931
F-statistic: 33.59 on 6 and 45 DF,  p-value: 4.698e-15
```

Wir sehen, dass der *residual standard error* bei `floe` um 9 niedriger ist als bei `flm`. Allerdings sind die mit einiger Vorsicht zu genießen, da 1.) mit schmalerer Spanner der Fit notwendigerweise besser wird; dies aber 2.) zu Kosten der Freiheitsgrade geht. Bei `flm` haben wir 7 Parameter schätzen lassen, bei `floe` 16. Würden wir den gesamten Datenbereich als Spanne angeben, dann erhalten wir ebenfalls 7 Parameter und der *residual standard error* ist nur um 4 niedriger als `flm`:

```
> loess(lemming ~ long * lat, topo, span = 1)

Call:
loess(formula = lemming ~ long * lat, data = topo, span = 1)

Number of Observations: 52
Equivalent Number of Parameters: 7.12
Residual Standard Error: 24
```

Neben den oben aufgeführten Funktionen gibt es einige weitere, die dasselbe anders, oder tatsächlich anders machen. Erwähnt sei vor allem die Funktion `plsmo` aus dem *package* **Hmisc**. Hier kann man neben der

loess-Variante auch den *supersmoother* (`supsmu`) benutzen. Außerdem erlauben verschiedene Optionen eine Darstellung mehrerer Gruppen. `locfit` aus gleichnamigem *package* berechnet lokale *likelihoods*; `gssanova` {**gss**} berechnet "gesmoothte" ANOVA-artige Modelle für nicht-normalverteilte Antwortvariablen; `sm` {**sm**} illustriert verschiedene *smoothing*-Funktionen (für verschiedene Verteilungen) ebenfalls auf der Basis lokaler *likelihood* Modelle. Siehe auch `polySpline` {**splines**} oder `smoothSurvReg` {**smoothSurv**} für weitere, spezielle Beispiele dieses Themengebietes.

9.0.2. Generalised Additive Models

Generalised Additive Models sind ein modernes Werkzeug. Sie sind, laienhaft ausgedrückt, eine Mischung aus dem Verallgemeinerten Linearen Modell (GLM) und *smoothing splines*. Mit den GLMs haben sie gemein, dass wir verschiedene Fehlerverteilungen nebst dazugehöriger *link*-Funktionen angeben können. Allerdings verbinden sie die Antwortvariable nicht mit den erklärenden Variablen über den gesamten Wertebereich, sondern lokal, wie eben *smoothing splines* auch.

GAMs ist keine Methode, sondern ein mehr oder weniger loses Agglomerat von Methoden. Gemeinsam ist ihnen das Ziel, durch flexible Kurven die Daten nachzuzeichnen. *Splines* und *smoother* sind grundverschiedene Ansätze und doch beide in GAMs wiederzufinden. Je nach Implementierung können GAMs also ganz unterschiedliche Sachen machen. In der Literatur finden wir GAMs entsprechend mal unter nicht-parametrischen Verfahren (wie wir es für *smoother* erwarten würden), mal unter semi-parametrischen Verfahren (was eher den *splines* entspricht).

Harrell (2001) schlägt die routinemäßige Nutzung von smoothern zur explorativen Datenanalyse und zur Visualisierung vor. Wenn wir nämlich erst einmal ein einfaches linears GLM an die Daten legen zwingen wir den Daten ein lineares Korsett über. GAMs, *smoother* und *splines* zeichnen erst einmal die Daten so nach, wie sie sind. Sich zu sehr von gekrümmten Linien verführen lassen ist auch problematisch (siehe unten, Abb. 9.3), und die im *spline* investierten Freiheitsgrade mögen gelegentlich verschwendet sein (siehe Harrell 2001, für einige sehr instruktive Beispiele, wie man smoother-Modelle vereinfacht und die Freiheitsgrade berücksichtigt).

Ein großer Unterschied zwischen GLMs und GAMs ist, dass im GLM Interaktionen analysiert werden können. Dies ist in GAMs so nicht möglich (2-D *smoothing splines* sind allerdings meistens möglich; nur bei höheren Interaktionen ist dann Schluss). Das ist kein allzu großer Nachteil, da die Fits wegen der hohen Flexibilität der GAMs zumeist besser sind als GLMs.[2] Der wirklich Nachteil ist der Mangel an Interpretierbarkeit, wenn wir 15 Knoten in einer Funktion finden.

Im GAM wird für jede erklärende Variable ein eigener *smoothing spline* konstruiert. Damit sieht das Modell etwa so aus:

$$y \sim f1(x_1) + f2(x_2) + f3(x_3, x_4) + x_5$$

wobei f für eine *smoothing*- oder *spline*-Funktion steht und x für die erklärenden Variablen. An diesem Beispiel sehen wir auch, dass wir das Modell so einsetzen können, dass zwei Variablen mit der gleichen *smoothing*-Funktion gefittet werden (hier x_3 und x_4). Darüberhinaus kann das Modell auch noch Variablen enthalten, die parametrisch gefitted werden sollen.

Als *smoothing*-Funktionen (f) stehen üblicherweise zwei verschiedene zur Verfügung. Die Mathematik hinter diesen Funktionen ist allerdings ausgesprochen unübersichtlich.[3]

[2] Für Interaktionen mit Faktoren gibt es die Möglichkeit eine Interaktion "zu Fuß" zu konstruieren. Siehe dazu etwa das letzte Beispiel in der Hilfe zu `s`.

[3] Um hier nur kurz aus der R-Hilfe zur Funktion `gam` zu zitieren: "The built in alternatives for univariate smooths terms are: a conventional penalized cubic regression spline basis, parameterized in terms of the function values at the knots; a cyclic cubic spline with a similar parameterization and thin plate regression splines. The cubic spline bases are computationally very efficient, but require 'knot' locations to be chosen

```
> library(mgcv)
> soay <- read.table("soaytest.txt", head = T)
> names(soay)

[1] "density" "jan"     "feb"     "mar"     "fm"      "all"     "delta"

> attach(soay)
> mod <- gam(delta ~ s(density))
> summary(mod)

Family: gaussian
Link function: identity

Formula:
delta ~ s(density)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.02151    0.03998   0.538    0.594

Approximate significance of smooth terms:
             edf Est.rank     F  p-value
s(density) 2.398        5 7.917 3.89e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.468   Deviance explained = 50.1%
GCV score = 0.069861   Scale est. = 0.063926  n = 40

> mod2 <- gam(delta ~ s(density) + s(all))
> summary(mod2)

Family: gaussian
Link function: identity

Formula:
delta ~ s(density) + s(all)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.02151    0.03788   0.568    0.574

Approximate significance of smooth terms:
           edf Est.rank      F  p-value
s(density)   1        1 34.825 8.54e-07 ***
s(all)       1        1  8.391   0.0063 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.523   Deviance explained = 54.7%
GCV score = 0.062046   Scale est. = 0.057393  n = 40
```

(automatically by default). The thin plate regression splines are optimal low rank smooths which do not have knots, but are more computationally costly to set up. Multivariate terms can be represented using thin plate regression splines, or tensor products of any available basis including user defined bases (tensor product penalties are obtained automatically form the marginal basis penalties). The t.p.r.s. basis is isotropic, so if this is not appropriate tensor product terms should be used. Tensor product smooths have one penalty and smoothing parameter per marginal basis, which means that the relative scaling of covariates is essentially determined automatically by GCV/UBRE."

```
> mod3 <- lm(delta ~ density * all)
> summary(mod3)

Call:
lm(formula = delta ~ density * all)

Residuals:
     Min       1Q   Median       3Q      Max
-0.71409 -0.12795 -0.01159  0.18123  0.34635

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.36296    2.55970   1.704    0.097 .
density     -0.58458    0.36724  -1.592    0.120
all          0.10606    0.13526   0.784    0.438
density:all -0.01736    0.01946  -0.892    0.378
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2402 on 36 degrees of freedom
Multiple R-Squared: 0.557,        Adjusted R-squared: 0.5201
F-statistic: 15.09 on 3 and 36 DF,  p-value: 1.607e-06

> plot(mod, ylim = range(delta, na.rm = T), rug = F, ylab = "delta")
> points(delta ~ density, pch = 16)
> lines(density, predict(lm(delta ~ density)), lty = 5)
```

Die Abb. 9.3 zeigt, dass unser GAM ein recht unruhiges Bild abgibt. Die linearen Regression liegt praktisch nie außerhalb der zwei Standardfehler um den *smoothing spline* (der in der Grundeinstellung übrigens als *thin plate regression spline* berechnet wird: die aufwendigste und am stärksten nachzeichnende Variante).

Wir wollen nicht unerwähnt lassen, dass es diverse verschiedene Implementierungen von GAMs in R gibt. Die beiden ausgereiftesten und deshalb am häufigsten unter Ökologen benutzen sind in den *packages* **mgcv** und **gam** zu finden. Beide sind von versierten und mit ökologischen Daten erfahrenen Statistikprofessoren entwickelt worden (Simon Wood und Trevor Hastie, respektive) und sind in verschiedenen Büchern beschrieben (Hastie and Tibshirani 1990; Wood 2006; Hastie et al. 2008). Während das **mgcv**-`gam` viele sinnvolle Grundeinstellungen hat, ist das **gam**-`gam` in seiner Knotenwahl besser steuerbar. Je nach Geschmack tendiert die NutzerIn zum einen oder anderen. Darüberhinaus hat Trevor Hastie noch das *package* **mda** entwickelt, dass mit `bruto` eine besonders schnelle Implementierung von GAMs beinhaltet.

Nicht unerwähnt wollen wir auch **VGAM**, ein sehr ambitioniertes *package*, in dem nicht nur GAMs sondern auch GLMs vollkommen neu implementiert sind. Dies war nach Ansicht des Entwicklers Thomas Yee nötig, um eine unglaubliche Vielfalt an Verteilungen einbeziehen zu können. Wenn der werte Leser, oder die werte Leserin, also jemals eine etwas ungewöhnliche Verteilung fitten muss (etwa eine *zero-inflated Poisson-* oder eine Dirichlet-Verteilung), dann ist **VGAM** ein prima Fundus! Weiterhin bietet **splines** genau was der Titel verspricht; **sm** eine gute Implementierung von *smoothing* Methoden; weniger genutzt aber genauso verfügbar sind `lowess`, `smooth.spline` und `supsmu` im Hauptpacket **stats**. Auf sie greift etwa der schicke *smoother-plot*-Befehl `plsmo` in **Hmisc** zu. Bei derzeit an die 1000 sich ständig verändernden *packages* ist eine vollständige Auflistung allerdings leider nicht möglich.

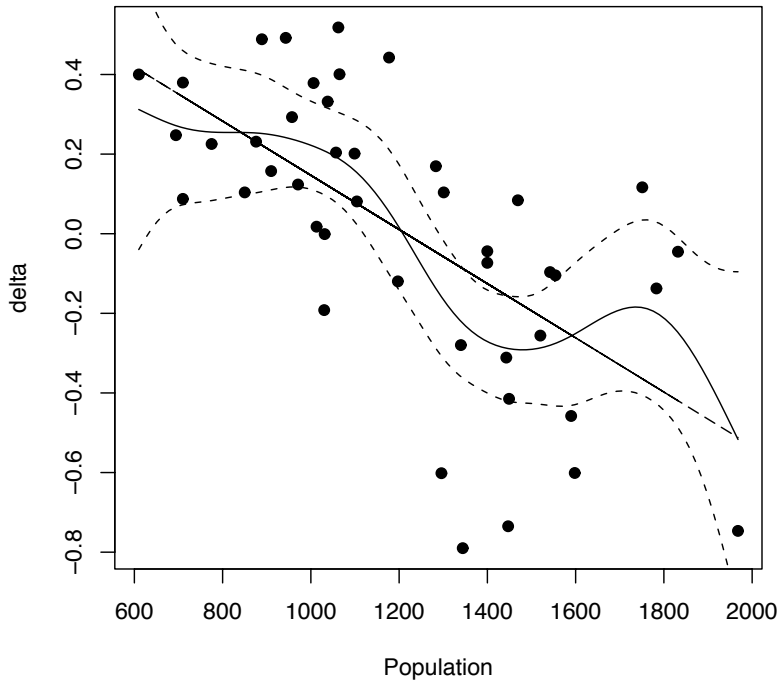


Abbildung 9.3.: Beispiel eines GAM-Fits: Abhängigkeit des Populationswachstums (delta) von der Populationsgröße des Vorjahres, in diesem Fall von wilden Schafen auf der schottischen Insel Soay (St. Kilder Archipel). Dargestellt ist ein GAM-Fit und, zum Vergleich, eine lineare Regression. Gestrichelte Linien sind $\pm$ 2 Standardabweichungen $\approx$ 95% CV-Interval. Daten von Mick Crawley, Imperial College, Silwood Park, UK.

Teil IV.

Multivariate Verfahren

10. Multivariate Daten und ihre Analyse: Einleitung und Überblick

Multivariate Daten enthalten mehr als nur eine Variable von unterschiedlichen Beobachtungspunkten. Typische Beispiele hierfür sind z. B. Vegetationstabellen oder ähnliche Sammlungen von Arten in unterschiedlichen Untersuchungsflächen. Ebenso gehören auch chemisch-physikalische Messwerte oder Landschaftsstrukturen in mehreren Untersuchungsflächen dazu. Die Daten werden dazu in Matrizen zusammengefasst. Die n Untersuchungsobjekte (z. B. die Untersuchungsflächen) sind in einer Matrix zeilenweise angegeben, die p untersuchten Variablen (z. B. unterschiedliche Umweltparameter, Arten, Reaktionen auf Management, usw.) bilden die Spalten. Im Gegensatz zu Verfahren wie der multiplen Regression zeichnen sich multivariate Analysen dadurch aus, dass auch die Ergebnisvariablen (die Abhängigen) aus mehreren Variablen bestehen können, also Matrix-Form aufweisen. Das Rechnen mit Matrizen erfolgt z. T. mit ganz anderen Operatoren als dies bei der skalaren Algebra (dem Rechnen mit einzelnen Zahlenwerten) der Fall ist. Auf die Besonderheiten der Matrixalgebra soll hier jedoch nicht weiter eingegangen werden (siehe auch Abschnitt 6.6 auf Seite 117).

Ebenso wie die parametrischen Statistikverfahren eine Normalverteilung voraussetzen, ist dies auch für daraus abgeleitete multivariate Verfahren notwendig (multinormale Verteilung). Dies bedeutet nicht nur, dass die einzelnen Variablen normal verteilt sein sollten (und ggf. transformiert werden müssen), sondern dass auch zwischen den Variablen überprüft werden muss, ob es dabei Ausreißer gibt, bzw. einzelne Variablen das Ergebnis überproportional beeinflussen.

Über eine Normalisierung hinaus müssen multivariate Daten meist auch standardisiert werden. Bei der Zentrierung werden von allen Variablenwerten die Mittelwerte der jeweiligen Variable abgezogen, so dass der Mittelwert Null ergibt. Bei der Standardisierung werden die Daten darüber hinaus noch durch die Standardabweichung der jeweiligen Variable geteilt, so dass die Standardabweichung aller Variablen eins beträgt (siehe Abschnitt 3.3 auf Seite 37).

10.1. Ordinationstechniken

Der Begriff „Ordination“ leitet sich aus dem lateinischen *ordinatio* (deutsch: Ordnung) ab. Der Sinn einer Ordination ist es, eine Reihe von Objekten oder Variablen einer Liste (oder Matrix) in einer sinnvollen Weise zu ordnen. Die daraus resultierende Grafik (Ordinationsdiagramm) zeigt diese Objekte bzw. Variablen geordnet entlang zweier Achsen. Verkürzt gesagt werden Arten, Aufnahmeflächen, Umweltparameter etc. mit einer Vielzahl von Variablen (und damit multivariaten Dimensionen) auf wenige Dimensionen (meist bis zu vier) eingeschränkt, wobei die erklärbare Variabilität für diese wenigen Dimensionen maximiert wird. Komplexe Datensätze können also vereinfacht abgebildet werden.

Die gebräuchlichsten Ordinationsverfahren lassen sich grob in direkte oder indirekte Gradientenanalyse mit linearer oder unimodaler Beziehung gliedern (Tabelle 10.1).

Lineare Gradientenanalysen gehen von einer linearen Reaktion der betrachteten Variable auf einen zu Grunde liegenden Gradienten aus (Abb. 10.1a). **Unimodale** Gradientenanalysen gehen von einer nicht-linearen Reaktion der betrachteten Variable auf einen zu Grunde liegenden Gradienten aus (Abb. 10.1b).

In der aktuellen ökologischen Diskussion wird davon ausgegangen, dass Arten unimodal auf Umweltgradienten reagieren. Ist ein Gradient (z.B. Nährstoff oder pH-Wert) zu gering,

Tabelle 10.1.: übersicht über die gebräuchlichsten multivariaten Ordinationsverfahren.

	Indirekt	Direkt
Linear	Hauptkomponentenanalyse (*Principal Component Analysis*, PCA)	Redundanzanalyse (*Redundancy Analysis* RDA)
Unimodal	Korrespondenzanalysen (*[Detrended] Correspondence Analysis* [D]CA)	Kanonische Korrespondenanalyse (*Canonical Correspondence Analysis*, CCA)

kann eine Pflanze nicht (Pessimum) oder nur in geringer Abundanz wachsen; diese steigt mit ansteigendem Gradient bis zu einem Optimum an und sinkt dann wieder bis zum Maximum ab.

Bei der **direkten** Gradientanananalyse werden die Aufnahmen nach einem gemessenen Gradienten (z. B. pH-Wert oder Nährstoffverfügbarkeit) geordnet. Gegenüber diesem Gradienten können die Art-Reaktionen aufgetragen werden. Im einfachsten Fall kann so z.B. aus den nach Art-Abundanzen gewichteten Mittelwerten der pH-Werte ein Wert für die Lage im Koordinatensystem (ein so genannter*score*) für die Aufnahmewerte errechnet werden. Aus diesen *scores* können wiederum gewichtete Mittelwerte für jede Art errechnet werden. Sowohl die Aufnahmen als auch die Arten lassen sich nach den *scores* sortiert in eine Reihenfolge (z.B. der pH-Abhängigkeit) ordnen.

Die **indirekte** Gradientanananalyse ermittelt einen hypothetischen zu Grunde liegenden Gradienten durch Maximierung der Varianzen in den multivariaten Dimensionen. Dies kann iterativ geschehen, indem willkürlich Werte eines Gradienten zugewiesen werden und daraus die *scores* für Aufnahmen berechnet werden. Diese *scores* werden zur Berechnung der *scores* von Arten genutzt und daraus können neue Aufnahme-*scores* errechnet werden. Wenn sich die Werte der hypothetischen Gradienten aus Arten und Aufnahmen angenähert haben ist das Ergebnis erreicht. Mit Hilfe von Matrix-Algebra können die Ergebnisse aber auch schneller berechnet werden. Die Interpretation dieses Gradienten muss durch den Anwender erfolgen.

Allen Ordinationsmethoden ist gemeinsam, dass auf der ersten Achse (erster [hypothetischer] Gradient) der Anteil an erklärbarer Varianz maximiert wird. Auf den folgenden Achsen wird jeweils die verbleibende Varianz maximiert. Wichtig dabei ist, dass alle Achsen orthogo-

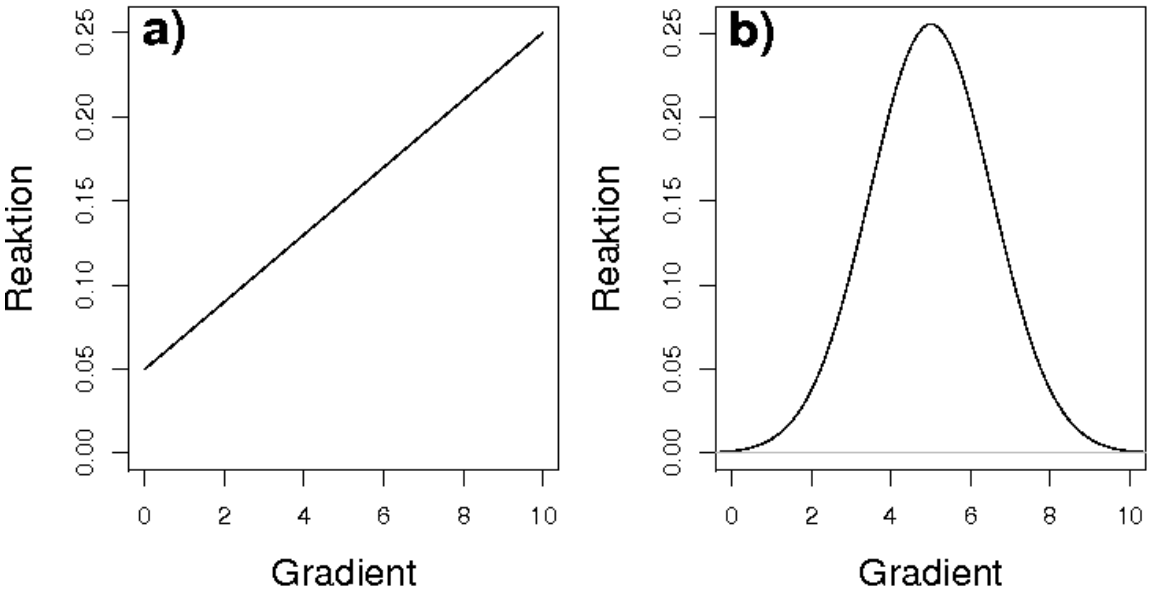


Abbildung 10.1.: Mögliches Reaktionsverhalten der betrachteten Variablen auf einen zugrundeliegenden Umweltgradienten: a) linear; b) unimodal.

nal (rechtwinkelig) zueinander sind, d.h. die Achsen sind unkorreliert. Der Anteil der Varianz, der durch eine Achse erklärt werden kann, kann aus dem **Eigenwert** (engl: *eigenvalue*) jeder Achse abgelesen werden. Die Summe der Eigenwerte ist die Gesamtvarianz. Der **Eigenvektor** besteht aus den Gewichtungen (oder **Ladungen**) mit denen die einzelnen Variablen zum Eigenwert beitragen. Die Quadratsummen eines jeden Eigenvektors ergibt immer 1.

10.2. Transformationen multivariater Daten

Artdaten werden häufig log-transformiert (`log(artmatrix+1)`), weil zum einen nur selten sehr abundante Arten auftreten und zum anderen diese Arten in ihrem Einfluss gemindert werden sollen. So ist z.B. in einem Orchideen-Buchenwald nicht die dominante Buche die wichtigste Art, sondern die seltenen Orchideen.

Mit dem Befehl `scale` können Matrizen spaltenweise zentriert und standardisiert werden: (etwa: `scale(artmatrix)`). Es wird eine Zentrierung (Mittelwert 0 und Standardisierung (Standardabweichung 1) durchgeführt (siehe auf Abschnitt 3.3 auf Seite 37). Durch Angabe weiterer Optionen sind auch Abweichung von den Standarisierungen möglich: `scale(artmatrix, centre=TRUE, scale=FALSE)`: Es wird nur eine Zentrierung (Mittelwert = 0) durchgeführt, die Standardabweichung bleibt unberührt.
Weitere wichtige Standardisierungen sind mit dem Befehl `decostand` (*package* **vegan**) möglich.

11. Indirekte Ordination

11.1. Hauptkomponentenanalyse

11.1.1. Beschreibung

Hauptkomponentenanalysen (*Principal Component Analysis*, PCA) kann man als Erweiterung der Regression auffassen. Dabei handelt es sich um eine Methode, die eine hypothetische Variable konstruiert, die die gesamte Summe der Quadratischen Abweichungen einer Ausgleichsgerade durch die Datenpunkte minimiert. Während allerdings bei einer (multiplen) Regression die unabhängigen Variablen vorgegeben werden, sind diese in einer PCA hypothetischer Natur und müssen aus den Daten abgeleitet werden.

Die PCA kann zur Dimensionsreduktion Ordnung von einzelnen Matrizen genutzt werden. Dabei werden die Variablen durch lineare Kombinationen zu den so genannten Hauptkomponenten zusammengefasst. Dies kann man sich bildlich so vorstellen, dass der Ursprung eines Koordinatensystems in den Mittelpunkt der Datenpunkte gelegt wird (Zentrieren). Danach wird das Koordinatensystem so lang rotiert, biss die Varianz auf der 1. Achse maximiert wurde. Daraus ergibt sich, dass insbesondere zwei Datentransformationen wichtig sind: (1) Das Zentrieren der Elemente (z. B. Arten, Umweltparameter), dadurch dass der Mittelwert eines Objektes (Aufnahme) von jedem Element abgezogen. Der Mittelwert eines Objektes wird dadurch 0. (2) Darüber hinaus kann eine Standardisierung (dividieren durch die Standardabweichung) erfolgen; die daraus folgende Standardabweichung ist 1.

Wenn alle Daten der Matrix in der gleichen Größenordnung und der gleiche physikalischen Einheit gemessen werden, ist die Zentrierung der Elemente der Normalfall. Dies entspricht einer PCA über die *Kovarianz*matrix. Allerdings können Elemente mit hoher Varianz (meist mit hoher Abundanz) das Ergebnis der PCA dominieren. Diese Methode ist der Normalfall bei Abundanzwerten von Arten.

Wenn die Daten in unterschiedlichen physikalischen Einheiten gemessen wurden, oder sich um mehrere Größenordnungen unterscheiden oder der Einfluss von Elementen mit hoher Varianz vermindert werden soll, wird eine PCA nach vorhergehender Standardisierung durchgeführt. Dies entspricht einer PCA über die *Korrelations*matrix und wird meistens bei Hauptkomponentenanalysen über Umweltvariablen durchgeführt (siehe Abschnitt 4.2 auf Seite 52 zur Unterscheidung und Berechnung von Kovarianz und Korrelation).

11.1.2. Beispiel

Ein Klassiker unter den Beispielen sind die so genannten *dune meadow* Daten von Jongman et al. (1995). Die Daten sind in R im Paket **vegan** vorhanden und eine Hauptkomponentananalyse sollte insbesondere mit der Funktion `prcomp` berechnet werden. Eine mathematisch weniger gute Variante, die allerdings zu S kompatibel ist, ist die Funktion `princomp`. Beachte dass alle etwas unterschiedliche Grundeinstellungen und Ausgabeformate besitzen.

```
> library(vegan)
> data(dune)
> names(dune)

 [1] "Belper" "Empnig" "Junbuf" "Junart" "Airpra" "Elepal" "Rumace"
 [8] "Viclat" "Brarut" "Ranfla" "Cirarv" "Hyprad" "Leoaut" "Potpal"
[15] "Poapra" "Calcus" "Tripra" "Trirep" "Antodo" "Salrep" "Achmil"
```

```
[22] "Poatri" "Chealb" "Elyrep" "Sagpro" "Plalan" "Agrsto" "Lolper"
[29] "Alogen" "Brohor"
```

```
> dune.pca <- prcomp(dune, scale=T)
```

Der Paramter *scale* bewirkt eine Standardisierung der Daten auf die Varianz 1, d.h. es wird eine Hauptkomponentenanalyse über die Korrelationsmatrix durchgeführt.
Das Ergebnis der Hauptkomponentenanalyse für die ersten sieben Hauptkomponenten (im folgenden auch HKs genannt) und die ersten 23 Arten sieht folgendermaßen aus (um nicht in Zahlen zu ertrinken, lassen wir uns die Ergebnisse nur auf drei Nicht-Nullstellen hinter dem Komma anzeigen):

```
> options(digits = 3)
> dune.pca

Standard deviations:
 [1] 2.65e+00 2.24e+00 1.89e+00 1.63e+00 1.46e+00 1.33e+00 1.22e+00
 [8] 1.15e+00 1.05e+00 8.99e-01 8.63e-01 8.35e-01 7.58e-01 5.98e-01
[15] 4.72e-01 4.69e-01 3.88e-01 3.63e-01 2.52e-01 1.55e-16

Rotation:
            PC1      PC2      PC3      PC4     PC5      PC6      PC7
Belper -0.20157 -0.12758 -0.03886 -0.32245 -0.0715  0.08590  0.18762
Empnig  0.00663  0.30681 -0.29141  0.04176 -0.0897  0.09343  0.07702
Junbuf  0.03087 -0.16878 -0.16085  0.39463  0.0598 -0.20101  0.03979
Junart  0.23881 -0.05436  0.10429  0.00541  0.1201  0.15446 -0.17938
Airpra -0.00162  0.33442 -0.27428  0.05642 -0.2116  0.09642 -0.09771
Elepal  0.29360  0.00647  0.22769 -0.09486 -0.0464  0.08615  0.03933
Rumace -0.17725 -0.00437  0.20522  0.40696  0.0558  0.24649  0.11379
Viclat -0.11239  0.10603  0.02872 -0.23818  0.3884 -0.35240  0.04043
Brarut  0.02569  0.14195  0.14116  0.09047  0.5133  0.15998  0.20582
Ranfla  0.30624 -0.00788  0.16894 -0.02894 -0.0569 -0.04235  0.09006
Cirarv -0.01665 -0.14938 -0.19344 -0.18919  0.0584  0.41828  0.32137
Hyprad -0.01752  0.35277 -0.28814  0.00427 -0.0352 -0.00437 -0.04949
Leoaut -0.17091  0.23208 -0.13556 -0.10787  0.1612 -0.17337  0.25103
Potpal  0.16148  0.03996  0.18702 -0.10601 -0.1918 -0.14085  0.21369
Poapra -0.24759 -0.21141 -0.04597 -0.14798  0.1861 -0.03391 -0.22959
Calcus  0.24492  0.03798  0.22235 -0.11309 -0.1389  0.01857  0.16720
Tripra -0.17973  0.04566  0.26082  0.32890  0.0648  0.23252  0.12855
Trirep -0.16740 -0.01958  0.11346 -0.02227 -0.0804 -0.32261  0.42213
Antodo -0.20229  0.25299  0.02503  0.14902 -0.2573  0.15613  0.02664
Salrep  0.11589  0.23795 -0.03469 -0.10080  0.1615  0.11866  0.04579
Achmil -0.27736  0.01591  0.17154 -0.04917 -0.2980 -0.01825 -0.00628
Poatri -0.14763 -0.32057 -0.09890  0.19519 -0.1388 -0.02864  0.10394
Chealb  0.05462 -0.11909 -0.13518  0.23244 -0.1214 -0.35487  0.11108
...
```

Dabei sind die Standardabweichungen die Wurzeln der Eigenwerte (das Maß der Variation) und in der Matrix `rotation` sind die Eigenvektoren aufgeführt, die die Ladungen der einzelnen Elemente enthalten. Diese entsprechen der Korrelation der Arten mit den HKs.

Die Eigenwerte für die einzelnen Hauptkomponenten lassen sich also einfach errechnen:

```
> dune.pca$sdev^2

 [1] 7.03e+00 5.00e+00 3.55e+00 2.64e+00 2.14e+00 1.76e+00 1.48e+00
 [8] 1.32e+00 1.11e+00 8.09e-01 7.45e-01 6.97e-01 5.75e-01 3.58e-01
[15] 2.23e-01 2.20e-01 1.51e-01 1.32e-01 6.35e-02 2.41e-32
```

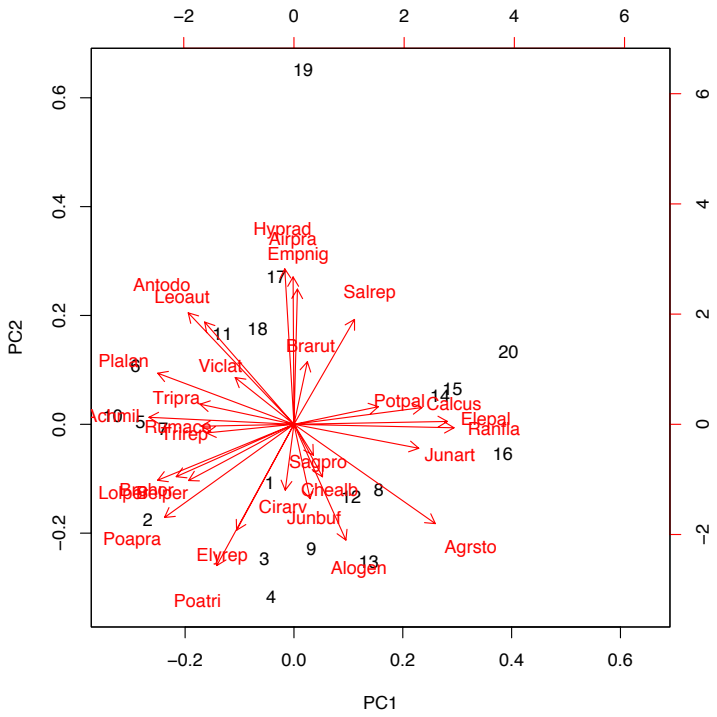


Abbildung 11.1.: *Biplot* über die Ausgabe der Funktion `prcomp`.

Das Ergebnis der Hauptkomponentenanalyse (auch als *scores* bezeichnet) wird in der Matrix `x` abgelegt (hier nur für die ersten acht HKs):

```
> dune.pca$x
```

```
       PC1      PC2      PC3      PC4      PC5       PC6      PC7      PC8
2   -3.187 -1.7465 -0.4008 -2.2958 -1.8566 -7.68e-01   0.4429 -1.3370
13   1.632 -2.5284 -2.0416  2.6111 -1.1030 -2.65e+00   0.6977 -1.3745
4   -0.497 -3.1715 -2.9214 -2.1252  0.5304  3.12e+00   2.0184  1.0607
16   4.554 -0.5481  1.6007 -0.0356 -0.1152  1.03e+00 -0.2483 -0.3256
6   -3.451  1.0827  3.1738  2.9998  1.1638  1.23e+00   0.9312  0.5374
1   -0.528 -1.0801 -0.0702 -0.7783 -0.5150  2.82e-01 -3.1319  1.0988
8    1.850 -1.2115 -0.1549 -0.1469  0.8911 -7.25e-02 -0.6649 -0.3715
5   -3.344  0.0461  1.5171  1.2150 -1.2204  1.57e+00   0.2157 -0.7181
17  -0.382  2.7143 -0.6823  0.4108 -2.1584  2.20e-01 -1.8047  0.6657
15   3.455  0.6482  1.7278 -0.5273 -0.2287 -8.05e-05   0.1557  1.2633
10  -3.950  0.1650  1.1943 -1.9036 -1.0920 -1.27e+00   0.8826 -0.6754
11  -1.558  1.6736 -0.3260 -1.4511  3.3126 -2.05e+00 -0.2625  1.3369
9    0.384 -2.2931 -1.1941  1.5006  0.6567  2.33e-01 -1.2316  0.2417
18  -0.790  1.7544  0.4726 -1.4537  2.7235 -7.79e-01   0.2365 -1.1894
3   -0.644 -2.4730 -1.0262 -0.7122  0.3316 -2.12e-03 -1.2772 -0.1563
20   4.674  1.3404  1.7036 -0.8215  0.6851  1.15e+00 -0.0734 -2.6786
14   3.186  0.5196  2.1602 -1.1119 -2.1709 -1.45e+00   1.6918  2.1126
19   0.198  6.5140 -4.4011  0.4691 -0.8149  6.98e-01   0.4837 -0.2331
12   1.248 -1.3288 -1.7519  2.6736  1.0451 -6.51e-01   1.2535  0.6919
7   -2.851 -0.0772  1.4205  1.4832 -0.0647  1.55e-01 -0.3152  0.0505
```

Mit Hilfe dieser Daten kann ein so genannter *biplot* erzeugt werden, um das Ergebnis der PCA zu visualisieren (Abb. 11.1).

```
> biplot(dune.pca)
```

Die stärksten Gradienten auf der ersten Achse werden durch *Eleocharis palustris* und *Ranunculus flammula* sowie *Achillea millefolium* und *Rumex acetosella* abgebildet. Eine Reduktion auf diese Arten dürfte einen

Großteil der Gesamtvariabilität des Datensatzes widerspiegeln. Eine senkrechte Projektion der Aufnahmeflächen auf die Art-Pfeile ermöglicht drüber hinaus auch eine Abschätzung der Bedeutung der Arten in den Aufnahmeflächen. Die größte Deckung mit *Ranunculus flammula* dürfte Aufnahme 20 haben, gefolgt von 16. Weiterhin ist zu erkennen, dass z.B. die Aufnahmen 5, 6, 7 & 10 einander recht ähnlich sind und sich stark von z. B. 16 oder 20 unterscheiden. Eine Interpretation der hypothetischen Gradienten könnte für Achse 1 ein Feuchtegradient und für Achse 2 ein Nährstoffgradient sein.

11.1.3. Stärken und Schwächen

Stärken der Methode sind die einfache Anwendbarkeit und Interpretation. Da die Methode aber quasi auf Kovarianz- bzw. Korrelationmatrizen beruht, ist sie **nur** für Daten gültig, die miteinander linear in Beziehung stehen.

Das der PCA zu Grunde liegende Ähnlichkeitsmaß entspricht der Euklidischen Distanz. Das führt dazu, dass die Objekte an den Achsenenden i.d.R. nur wenige Elemente gemein haben, dafür aber viele Null-Werte. Daher ist ihre Euklidische Distanz vergleichsweise gering, was zu verzerrten Abständen bzw. Rangfolgen der Objekte an den Gradientenenden führt. Ist ein weitergehender Gradient nur gering ausgebildet kann es zu einem „Hufeiseneffekt“ (engl. *horse-shoe effect*) kommen, der sehr stark ausgebildet ist und bei dem sich die Extrema wieder zusammen neigen. kommen. Dadurch wird fälschlich ein weiterer Gradient errechnet, der jedoch nicht vorhanden ist. Geeignete Transformationen können einen solchen Effekt vermindern.

11.2. Korrespondenzanalyse

11.2.1. Beschreibung

Korrespondenzanalysen sind eine Erweiterung zur Analyse von Kontingenztabellen („X^2-Tabellen“). In diesen Tabellen wird der Zustand eines Deskriptors (Objekte in Zeilen) mit denen eines anderen Deskriptors (Variablen in Spalten) verglichen. Für die Korrespondenzanalyse wird diese so weit verallgemeinert, dass Kontingenztabellen in wenige zusammenfassende Variablen reduziert werden, die eine mangelnde Unabhängigkeit zwischen Zeilen und Spalten repräsentieren. Für ökologische Zusammenhänge bedeutet dies, dass in wenigen Dimensionen Arten und Aufnahmen, die nicht unabhängig voneinander sind (sich also ähneln), gemeinsam dargestellt werden können. Die Variabilität der Daten wird so auf wenige Dimensionen reduziert. Korrespondenzanalysen können generell dann durchgeführt werden, wenn *alle Daten in einer Tabelle die gleiche physikalische Dimension aufweisen und nur größer oder gleich Null sind.*

In der ökologischen Theorie liegt der Korrespondenzanalyse eine unimodale Reaktion der Arten auf einen Umweltgradienten zu Grunde. Für jede Art kann das Optimum (wiedergegeben durch einen *score*) der Art entlang eines Gradienten als das gewichtete Mittel (nach Abundanz/Häufigkeit gemittelt) über die Aufnahmen berechnet werden. Die entsprechenden Werte (*scores*) für die Aufnahmen errechnen sich als gewichtete Mittel über die *scores* der Arten auf iterative Weise. Einfacher ist dies jedoch über eine Matrix-Erweiterung der X^2-Statistik. Ähnlich wie bei der PCA wird der größte Teil der Variabilität auf der 1. Achse erklärt, dann in abnehmender Rangfolge auf den nächsten, jeweils senkrecht zu den vorhergehenden stehenden Achsen. Das inhärent zu Grunde liegende Distanzmaß ist hier allerdings die X^2-Distanz (anstelle der Eukild'schen Distanz der PCA).

11.2.2. Beispiel

Zum besseren Vergleich sollen auch hier die Daten des *dune meadow*-Datensatzes als Korrespondenzanalyse berechnet werden. Eine der einfachsten der zur Verfügung stehenden Funktionen für eine Korrespon-

denzanalyse ist die Funktion `corresp` (*package* **MASS**). Weitere sind z.B. `cca` und `decorana` (*package* **vegan**), `cca` (*package* **ade4**) oder `CAIV` (*package* **CoCoAn**).

```
> library(MASS)
> dune.ca <- corresp(dune, nf=19)
```

Der Paramter *nf=19* bewirkt dass alle 19 möglichen Dimensionen dieser Korrespondenzanalyse berechnet werden.
Das Ergebnis läßt sich einfach durch Aufruf des erzeugten Objektes *dune.ca* anzeigen. Zur besseren Übersicht soll es aber in die Bestandteile zerlegt werden:

```
> dune.ca
```

```
First canonical correlation(s): 0.732 0.633 0.510 ...
```

Die Güte der Achsen wird hier als Kanonische Korrelation widergegeben. Durch quadrieren erhält man die Eigenwerte. Die Erklärbare Variabilität und die kumulative erklärbare Variabilität kann einfach berechnet werden:
Eigenwerte:

```
> dune.ca$cor^2
```

```
 [1] 0.536 0.400 0.260 0.176 0.145 0.108 0.092 0.081 0.073 0.056
[11] 0.048 0.041 0.035 0.021 0.015 0.009 0.008 0.007 0.003
```

Daraus können wir schnell die erklärbare Variabilität berechnen:

```
> dune.ca$cor^2/sum(dune.ca$cor^2)
```

```
 [1] 0.253 0.189 0.123 0.083 0.068 0.051 0.044 0.038 0.035 0.027
[11] 0.023 0.020 0.017 0.010 0.007 0.004 0.004 0.003 0.002
```

Die kumulative erklärbare Variabilität erhalten wir folgendermaßen:

```
> cumsum(dune.ca$cor^2/sum(dune.ca$cor^2))
```

```
 [1] 0.253 0.443 0.565 0.649 0.717 0.768 0.812 0.850 0.885 0.911
[11] 0.934 0.954 0.970 0.980 0.987 0.991 0.995 0.998 1.000
```

Die ersten fünf Achsen mit Eigenwerten `sum(dune.ca$cor^2)/19=0.11` erklären mehr Variabilität als im Durchschnitt zu erwarten wäre; die ersten drei Achsen erklären mehr als die Hälfte, die ersten vier fast zwei Drittel der Variabilität in den Daten.

Wenden wir uns aber wieder der ursprünglichen Ausgabe zu. Dort sind weiterhin die Koordinaten (*scores*) der Objekte (Aufnahmen) und Variablen (Arten) zu erkennen. Diese sind in `dune.ca` in den Matrizen `$rscore` bzw. `$cscore` zu finden. Von den bei den Matrizen werden jedoch nur die ersten Zeilen und Spalten wiedergegeben:

```
 Row scores:
      [,1]      [,2]   [,3]    [,4]    [,5]    [,6]     [,7]   [,8]
2  -0.6327 -0.695836 -0.097 -1.1870 -0.9769 -0.0658 -1.18231 -0.571
13  0.4245 -0.844019  1.590  1.2488 -0.2075 -0.8757 -1.14433 -1.312
4  -0.0565 -0.763978  0.918 -1.1759 -0.3840  0.1399 -0.53616  2.551
16  2.0023  0.109063 -0.334  0.3376 -0.5010  0.7616  0.52454 -0.565
...

 Column scores:
          [,1]    [,2]    [,3]    [,4]    [,5]    [,6]    [,7]
Belper -0.68319 -0.5612 -0.2990 -1.6786 -0.1539 -0.2225 -2.3121
Empnig -0.94283  5.1602  3.8398 -0.4218 -0.1932  0.4896  0.0225
Junbuf  0.11141 -1.0762  1.9726  2.5707  0.7053 -0.7357 -0.3300
Junart  1.74260  0.1575 -0.1829  0.0132  0.7606  2.3820  1.1815
...
```

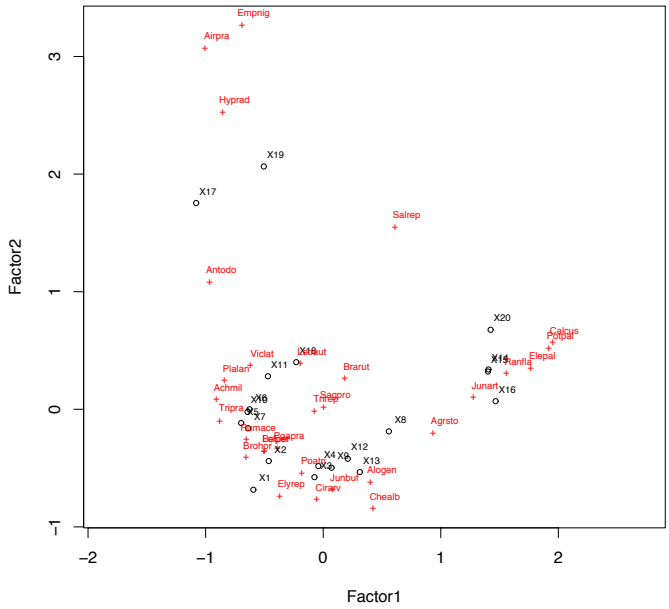


Abbildung 11.2.: Ordinationsplot über die Ausgabe der Funktion `corresp`.

Analog zur grafischen Ausgabe der Hauptkomponentenanalyse können wir auch das Ergebnis der Korrespondenzanalyse mit der Funktion `biplot` visualisieren:

```
> biplot(dune.ca)
```

Aus dem Ordigramm können wir sehen, dass sich die Arten *Empetrum nigrum, Hypochaeris radicata* und *Aira praecox* als Ausreißer relativ ähnlich verhalten und besonders mit den Aufnahmen 17 und 19 assoziiert sind. Das andere Ende des entsprechenden Gradienten auf der 2. Achse wird von *Elymus repens, Cirsium arvense* und *Chenopodium album* sowie den Aufnahmen 1, 3, 4, 9, 12, 13 gebildet. Hier stehen also Arten nährstoffarmer Sandrasen denen nährstoffreicher Ruderalstandorte gegenüber. Die erste Achse hat im unteren Bereich mit *Plantago lanceolata* und *Achillea millefolium* Arten der mäßig frischen bis trockenen Bereiche mit einer Reihe von Aufnahmen und im oberen Bereich die Feuchtezeiger *Ranunculus flammula, Eleocharis palustris* und *Potentilla palustris*. Im Gegensatz zur Hauptkoordinatenanalyse wird bei diesem Ergebnis implizit eine unimodale Reaktion der Pflanzen auf einen Umweltgradienten angenommen.

11.3. „Detrended“ Korrespondenz-Analyse

11.3.1. Beschreibung

Die Korrespondenzanalyse (CA) hat zwei große Nachteile: 1. die Enden der Achsen sind oft im Vergleich zu deren Zentren gestaucht und 2. zeigt die zweite Achse häufig eine systematische, meist quadratische Beziehung mit der 1. Achse. Dies führt zum so genannten *Bogeneffekt* (engl. *arch effect*), der weniger stark ausgeprägt ist als der Hufeiseneffekt der Hauptkomponentenanalyse. Dieser Bogeneffekt ist ein mathematisches Artefakt ohne reale Beziehung zu den Daten. Aus diesem Grund wurde die „Detrended“ Korrespondenz-Analyse (DCA) mit dem Programm DECORANA entwickelt, die die Achsen entzerrt und so den Bogen ausgleicht.

Beim *detrending* wird die 1. Achse in eine Reihe von Segmenten (meist 10-46, empfohlen 26) untergliedert. Für jedes Segment wird der *score* der 2. Achse durch den Mittelwert aller *scores* dieser Achse des entsprechenden Segmentes geteilt (so dass der neue Mittelwert der 2. Achse jedes Segmentes 0 ergibt). Dies wird für alle nachfolgenden Achsen wiederholt. Um

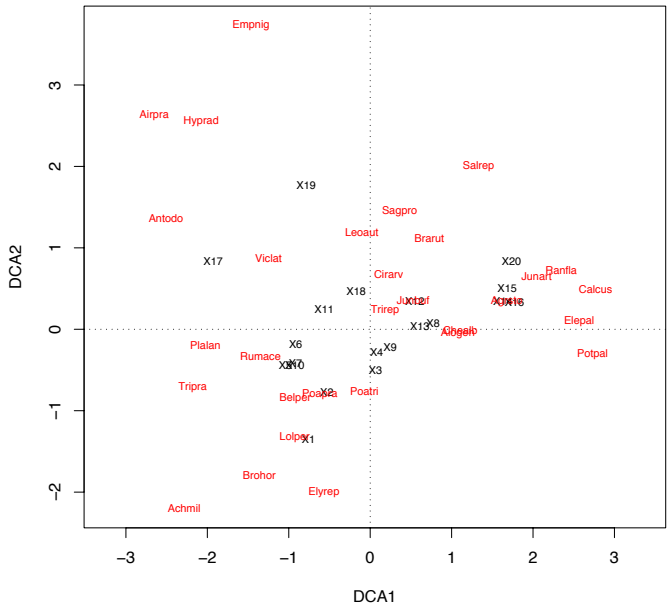


Abbildung 11.3.: *Biplot* über die Ausgabe der Funktion `decorana`.

die Achsenenden im Vergleich zu deren Zentren zu entzerren, werden Segmente mit einer geringen Varianz zwischen den Aufnahmeflächen verlängert und Segmente mit hoher Varianz innerhalb der Aufnahmeflächen gestaucht. Dies führt zu einer Vereinheitlichung der Breite der unimodalen Arten-Reaktionskurven.

Nach der Neuskalierung der Achsen haben sie eine interessante Eigenschaft gewonnen, denn sie sind in Einheiten durchschnittlicher Standardabweichung (*standard deviation*, SD) skaliert. Bei unimodaler Reaktion einer Art entlang eines einheitlichen Gradienten erscheint eine Art, erreicht ihr Optimum und verschwindet wieder über eine Entfernung von ca. 4 SD. Entsprechend erfolgt ein (fast) kompletter Artenaustausch über eine Entfernung von 4 SD; ein Austausch des halben Arteninventars über ca. 1-1.4 SD. Daher kann die Länge der 1. DCA-Achse als Maß für den ökologischen Gradienten im Sinne eines Artenaustausches gesehen werden.

Aufgrund der Neuskalierung der Arten und der Entfernung von ökologisch evtl. doch wichtigen Gradienten entlang der 1. Achse wird eine DCA zur reinen Ordinationszwecken meist nicht mehr empfohlen; eine Alternative für komplexe Gradienten ist die Multidimensionale Skalierung (MDS). DCA-Ordinations-Diagramme sollten immer mit Vorsicht interpretiert werden. Als Methode für die Ermittlung der Gradientenlänge ist die DCA geeignet.

11.3.2. Beispiel

Auch für diese Methode soll der *dune meadow*-Datensatz zur besseren Vergleichbarkeit genutzt werden. Die entsprechende Funktion ist `decorana` (*package* **vegan**). Alle multivariaten Funktionen des Paketes vegan haben den Vorteil, dass sie sich mit den Funktionen `plot` bzw. `ordiplot` leicht ausgeben lassen (Abb. 11.3).

```
> dune.dca <- decorana(dune)
> dune.dca

Call:
decorana(veg = dune)
```

```
Detrended correspondence analysis with 26 segments.
Rescaling of axes with 4 iterations.

                 DCA1  DCA2   DCA3   DCA4
Eigenvalues     0.512 0.304 0.1213 0.1427
Decorana values 0.536 0.287 0.0814 0.0481
Axis lengths    3.700 3.117 1.3006 1.4788
```

```
> plot(dune.dca, cex = 0.7)
```

Der Ordinationsplot gibt den ersten Gradienten (Feuchte) gut wieder, aber im Vergleich zur CA fällt die Interpretation eines Nährstoff- bzw. Ruderalisierungs-Gradienten schwer.
Die Eigenwerte einer DCA sind mit großer Vorsicht zu genießen, da sie ja entweder auf den Berechnungen vor dem *detrending* beruhen, und damit nicht dem Ordinationsergebnis entsprechen, oder nach dem *detrending* ermittelt wurden, und somit nicht den realen Daten entsprechen. Sie werden nur zur Kompatibilität mit anderen Programmen ausgegeben. Das wichtigste Ergebnis der DCA ist die Länge des Gradienten: Ist dieser größer als 4, findet ein vollkommener Artenaustausch statt, die unimodale Reaktion der Arten wird vollständig erfasst und damit auch die gesamte Länge des ökologischen Gradienten. Ist dieser kleiner als 2 wird nur ein Teil des Gradienten erfasst, z. B. der aufsteigende oder absteigende Teil der unimodalen Reaktionskurve. Damit kann die Länge des Gradienten genutzt werden, um zu entscheiden, on lineare oder unimodale Modelle für eine Ordination genutzt werden sollen: Bei Gradientenlängen > 4 sind unimodale, bei < 2 lineare Modell dem Gradienten angemessen. Dazwischen sind beide Methoden gleich (un)angemessen.
Nachbemerkung: Da nun die DCA eingeführt wurde, kann auch viel problemloser als im zuvor genannten Beispiel eine CA gerechnet und ausgedruckt werden: die Funktion `decorana` ermöglicht eine Auswahl der Analysemethode über den Parameter `ira` (`ira=0` $\rightarrow$ DCA, `ira=1` $\rightarrow$ CA) und damit alle Möglichkeiten der einfachen Zusammenfassung oder graphischen Ausgabe des **vegan**-Paketes:

```
> dune.ca2 <- decorana(dune, ira = 1)
> summary(dune.ca2)

Call:
decorana(veg = dune, ira = 1)

Orthogonal correspondence analysis.

              RA1 RA2  RA3   RA4
Eigenvalues 0.536 0.4 0.26 0.176

Species scores:

            RA1      RA2      RA3      RA4 Totals
Belper -1.00296 -0.72465 -0.34750 -1.84907     13
Empnig -1.38412  6.66257  4.46314 -0.46462      2
Junbuf  0.16356 -1.38950  2.29281  2.83193     13
Junart  2.55824  0.20336 -0.21255  0.01452     18
Airpra -2.01391  6.26108  3.05059  0.51028      5
Elepal  3.53683  0.70547 -1.30748 -0.00782     25
Rumace -1.30918 -0.52099 -1.36205  3.04663     18
Viclat -1.24109  0.75809 -1.05027 -2.62698      4
Brarut  0.36540  0.54044 -0.37869  0.16806     49
Ranfla  3.12583  0.62661 -0.67877  0.12335     14
Cirarv -0.11323 -1.55938  2.09322 -3.08793      2
Hyprad -1.71260  5.16036  2.59856 -0.45984      9
Leoaut -0.39233  0.79366  0.09065 -0.34242     54
Potpal  3.84378  1.06448 -2.69575 -0.05714      4
Poapra -0.78041 -0.67354 -0.04594 -0.94112     48
Calcus  3.91413  1.15819 -1.95995 -0.25977     10
```

```
Tripra -1.76692 -0.19987 -2.69480  3.36766      9
Trirep -0.15372 -0.04148 -0.46962  0.06951     47
Antodo -1.93855  2.21178 -0.39194  1.20745     21
Salrep  1.22389  3.16105  0.11335 -1.59433     11
Achmil -1.82192  0.17271 -1.33714 -0.02338     16
Poatri -0.36465 -1.10214  0.53333  0.46961     63
Chealb  0.85111 -1.72282  3.62650  3.27926      1
Elyrep -0.74342 -1.51345  0.59832 -1.48714     26
Sagpro  0.00730  0.03506  2.54424  0.17591     20
Plalan -1.68554  0.50796 -1.78022  0.97466     26
Agrsto  1.87243 -0.42153  0.64227  0.06379     48
Lolper -1.00807 -0.73388 -0.49760 -1.24670     58
Alogen  0.80384 -1.26223  1.93862  0.91055     36
Brohor -1.31867 -0.82938 -0.69975 -1.30439     15

Site scores:

          RA1       RA2       RA3       RA4 Totals
X2  -0.680002 -0.568315 -0.057518 -0.548515     42
X13  0.456200 -0.689357  0.942135  0.577082     33
X4  -0.060695 -0.623976  0.543804 -0.543419     45
X16  2.152063  0.089075 -0.197957  0.156019     33
X6  -0.920382 -0.000436 -0.827830  0.738970     48
X1  -0.872389 -0.884258 -0.085786 -0.973560     18
X8   0.819733 -0.242448  0.211187 -0.049793     40
X5  -1.024208 -0.150767 -0.566419  0.401378     43
X17 -1.585814  2.264340  0.242077  0.347150     15
X15  2.057001  0.414831 -0.572090  0.019520     23
X10 -0.944590 -0.028836 -0.491639 -0.314482     43
X11 -0.690273  0.362664 -0.102904 -0.506892     32
X9   0.104183 -0.642314  0.512398  0.185263     42
X18 -0.335779  0.516869 -0.393964 -0.520947     27
X3  -0.109066 -0.745584  0.407676 -0.314888     40
X20  2.089822  0.873001 -0.394529 -0.255625     31
X14  2.063575  0.437049 -0.828588 -0.039606     24
X19 -0.741898  2.665997  1.159497 -0.081755     31
X12  0.306934 -0.543646  0.974089  0.792530     35
X7  -0.936680 -0.208023 -0.514406  0.418075     40
```

```
> plot(dune.ca2)
```

Die Ergebnisse entsprechen weitgehend denen der Funktion ca, allerdings erfolgt eine andere Skalierung der *scores*, nämlich nach der so genannten Hill's Skalierung. Diese ist ein Quasi-Standard in der Ökologie und errechnet sich aus den Objekt *scores* geteilt durch $\sqrt{(1-\lambda)/\lambda}$ bzw. aus den Variablen *scores* geteilt durch $\sqrt{\lambda \cdot (1-\lambda)}$. λ entspricht dabei dem Eigenwert der jeweiligen Achse. Eine solche Standardisierung ist notwendig, weil sich die Achsen in ihrer Bedeutung unterscheiden. Eine wünschenswerte Eigenschaft zur Standardisierung, die durch Hills *scaling* erreicht wird, ist dass die durchschnittliche Breite der Reaktionskurve einer Art für jede Achse gleich ist. Die Funktion `decorana` skaliert die Aufnahmen, die Funktion `ca` sowohl Aufnahmen als auch Arten.

12. Kanonische Ordinationen (Direkte Ordinationen)

Kanonische oder direkte Ordinationsmethoden führen eine Ordination (PCA oder CA) nicht nur über eine Matrix durch, sondern haben zwei Matrizen zur Grundlage. Als Spezialfall der kanonischen Ordination kann die multiple Regression gelten, bei der aus einer Matrix von abhängigen Variablen ein Vektor aus abhängigen Variablen berechnet werden soll. Für jede der unabhängigen Variablen gibt es einen Koeffizienten, so dass man diese wiederum in einem Vektor zusammenfassen kann. Grob vereinfacht kann man so eine multiple Regression als eine kanonische Ordination mit nur einer Dimension (Gradienten) betrachten. Handelt es sich bei den Abhängigen nun nicht um einen Vektor, sondern auch um eine Matrix (also eine Reihe von „Abhängigen“, kann man dies als Fall für kanonische Ordinationsmethoden betrachten. Die für den Ökologen normale Sichtweise ist die, dass eine Ordination von Art-Matrizes durchgeführt wird, deren maximaler Gradient aber linear durch eine weitere Matrix von Umweltvariablen eingeschränkt. Es steht also nicht mehr die uneingeschränkt maximale Variabilität für die Ermittlung der Gradienten zur Verfügung, sondern diese Variabilität wird im Raum durch die gemessenen Umweltvariablen beschränkt. Die Artdaten werden so durch die lineare Kombination der Umweltdaten erklärt.

12.1. Redundanz-Analyse

12.1.1. Beschreibung

Die Redundanz-Analyse (RDA) ist eine Erweiterung sowohl der multiplen Regression als auch der Hauptkomponenten-Analyse (Redundanz ist synonym mit „erklärbarer Varianz“). Wie bei einer multiplen Regression gibt es unabhängige (Prädiktor-)Variablen und abhängige Variablen. Die Ordination über eine Y-Matrix der Abhängigen ist eingeschränkt, da die resultierenden Ordinations-Vektoren lineare Kombinationen der Unabhängigen sind. Bei der Redundanz-Analyse wird für jede abhängige Variable eine multiple Regression über die Matrix der Unabhängigen durchgeführt und so Vorhersagen für die abhängigen erhalten. So erhält man eine Matrix mit vorhergesagten Y-Werten. Über diese Matrix wird nun eine Hauptkomponenten-Analyse der Kovarianzmatrix durchgeführt.

In der Theorie unterliegt eine RDA den gleichen Schwächen wie eine PCA. Da die Reaktion der Abhängigen jedoch durch eine Matrix von unabhängigen Variablen eingeschränkt wird, tritt der Hufeiseneffekt allerdings meist nicht zu Tage.

Die RDA sollte angewandt werden, wenn es eine lineare Beziehung der Variablen untereinander gibt und abhängige Variablen (Art-Abundanzen) mit unabhängigen Variablen (Umweltvariablen) in Beziehung gesetzt werden sollen.

12.1.2. Beispiel

Wir wollen einen Datensatz über flechtenreiche Weide aus Skandinavien (Väre et al. 1995) analysieren. Die Arten (`varespec`) umfassen sowohl höhere Pflanzen als auch Flechten. Die Umweltvariablen sind meist selbsterklärend; `baresoil` ist der Anteil an Boden ohne Vegetationsbedeckung, `humpdepth` ist die Mächtigkeit der Humusauflage. Unsere Analyse sieht folgendermaßen aus (hier stellen wir einmal den

Gesamtoutput dar:[1]:

```
> data(varespec)
> data(varechem)
> vare.rda <- rda(varespec, varechem)
> summary(vare.rda)

Call:

Partitioning of variance:

Total         1826
Constrained   1460
Unconstrained  366

Eigenvalues, and their contribution to the variance

             RDA1    RDA2    RDA3  RDA4   RDA5   RDA6   RDA7   RDA8  RDA9 RDA10 RDA11 RDA12
lambda    820.104 399.285 102.562 47.63 26.838 24.048 19.064 10.167 4.429 2.272 1.535 0.926
accounted   0.449   0.668   0.724  0.75  0.765  0.778  0.789  0.794 0.796 0.798 0.799 0.799
          RDA13 RDA14     PC1    PC2    PC3    PC4    PC5    PC6   PC7   PC8   PC9
lambda    0.716 0.312 186.192 88.464 38.188 18.402 12.839 10.552 5.519 4.521 1.092
accounted 0.799 0.800   0.102  0.150  0.171  0.181  0.188  0.194 0.197 0.200 0.200

Scaling 2 for species and site scores
-- Species are scaled proportional to eigenvalues
-- Sites are unscaled: weighted dispersion equal on all dimensions


Species scores

             RDA1      RDA2      RDA3      RDA4      RDA5      RDA6
Cal.vul -0.151656  0.324463  8.02e-02  1.077511 -0.875270 -0.089505
Emp.nig -0.160118 -0.295235  6.17e-01 -0.250897  0.681201 -0.252877
Led.pal -0.114893 -0.030597  4.40e-02 -0.035158  0.039086  0.009801
Vac.myr -0.733427 -0.577896 -2.23e-01  0.035301  0.079772  0.096078
Vac.vit -0.108712 -0.649317  1.13e+00 -0.609606  0.658570 -0.320296
Pin.syl  0.024789 -0.011152  1.13e-02  0.012330  0.007748  0.009476
Des.fle -0.098909 -0.078437 -6.59e-02  0.005557  0.034969 -0.034367
Bet.pub -0.002854  0.001411  5.60e-03 -0.004637  0.001005  0.001663
Vac.uli -0.004139  0.274204 -1.01e-01 -0.043156  0.109086 -0.241733
Dip.mon -0.012995  0.039677  2.10e-02 -0.020340  0.036649 -0.055026
Dic.sp  -0.391859 -0.259263 -1.81e-01 -0.747383 -0.438809  0.345530
Dic.fus -0.858631  0.048389  6.35e-01  1.659621  0.754859  0.489101
Dic.pol -0.035344 -0.000421  5.26e-02 -0.110438 -0.041912  0.045303
Hyl.spl -0.350288 -0.429344 -3.85e-01  0.047726 -0.001644 -0.038355
Ple.sch -3.692157 -4.156900 -1.98e+00  0.229243  0.190887 -0.263967
Pol.pil  0.000424  0.004799 -3.77e-03 -0.002923  0.003819  0.009023
Pol.jun -0.082708 -0.080187  1.73e-02 -0.072109  0.148604 -0.007148
Pol.com -0.006939 -0.004644  5.12e-03 -0.007904  0.002777 -0.001361
Poh.nut  0.006505 -0.008582  1.25e-02 -0.009229 -0.002445  0.004808
Pti.cil -0.106878  0.046798  2.49e-01 -0.208508  0.040672  0.046419
Bar.lyc -0.031292  0.017893  6.56e-02 -0.060380  0.002663  0.015874
Cla.arb -0.279383  2.486282 -2.20e-01  0.455194 -0.067269 -1.200452
Cla.ran  0.868568  3.911566 -2.12e+00 -0.156539  0.496217  0.388777
Cla.ste  8.712547 -2.152912 -5.82e-01  0.263792  0.098363 -0.119456
Cla.unc -0.093608 -0.093978  4.77e-01  0.018243 -0.340317  0.541468
```

[1]Im folgenden verzichten wir darauf. Bei den übrigen vorgestellten Funktion aus **vegan** können aber diese Zusatzinformationen mittels des `summary`-Befehls abgerufen werden.

```
Cla.coc  0.009413  0.005643  2.09e-03  0.008153  0.005331  0.001917
Cla.cor -0.011489 -0.014913  2.02e-02 -0.011515  0.030053 -0.011306
Cla.gra -0.009579  0.009724  1.13e-02 -0.009684 -0.002963 -0.001376
Cla.fim  0.003092  0.002948  1.46e-02  0.007925  0.007273 -0.008674
Cla.cri -0.006426 -0.013822  5.78e-02  0.021121  0.007990 -0.026541
Cla.chl  0.012566 -0.004340  6.77e-03 -0.009483 -0.001092 -0.001549
Cla.bot -0.004429  0.000588  6.00e-03 -0.003831 -0.002378  0.000817
Cla.ama -0.000917  0.002246 -1.28e-05 -0.001058  0.001696 -0.000815
Cla.sp   0.006653 -0.001748  3.77e-03  0.003479 -0.003723 -0.002204
Cet.eri  0.004940  0.002320  6.00e-03 -0.011463 -0.035826  0.017448
Cet.isl  0.011271 -0.004466  1.53e-02 -0.013047  0.000706  0.006039
Cet.niv  0.050842  0.116982 -7.24e-02 -0.029325 -0.323682 -0.168165
Nep.arc -0.041409 -0.035714 -2.52e-02 -0.025048  0.090842 -0.004609
Ste.sp  -0.074157  0.316097 -2.01e-01 -0.094486  0.099957  0.297752
Pel.aph -0.002906 -0.003706  1.49e-03 -0.001651  0.008005 -0.000861
Ich.eri -0.001075  0.004612 -1.66e-03  0.000312  0.001252  0.003792
Cla.cer  0.000779  0.000307 -7.03e-04 -0.000147 -0.001187 -0.000234
Cla.def -0.031628 -0.014931  1.02e-01  0.002146 -0.008116  0.016785
Cla.phy  0.017950 -0.008041  4.36e-04 -0.001491  0.002900  0.001971


Site scores (weighted sums of species scores)


       RDA1   RDA2   RDA3    RDA4   RDA5   RDA6
18 -1.0820  2.829  2.265 -2.4621  1.928 -4.891
15 -2.8875 -1.638 -1.153  2.4191  0.146 -1.476
24 -2.7058 -2.075 -0.158 -5.3987 -6.513  4.299
27 -3.4840 -4.572 -4.158 -0.8985  4.810 -2.735
23 -2.1085 -0.252  3.235 -3.1031  3.315 -2.425
19 -0.0859 -1.757  0.872 -1.1314 -1.077 -2.130
22 -2.7564 -1.357  3.386 10.3001  5.654  5.574
16 -2.2420  0.434  1.450  6.8677  2.683  5.940
28 -4.3080 -6.195 -6.924  0.3404 -1.174 -1.925
13 -0.3225  3.006 -0.197  4.9519 -9.029 -3.629
14 -1.7984  0.895  5.737  3.1785 -6.401  2.463
20 -1.8249  0.471  2.601 -2.6654 -1.323  1.550
25 -2.6508 -1.764  1.523 -0.6635 -1.226  2.065
7  -1.0124  6.192 -2.311 -1.4594  1.366 -6.193
5  -0.7275  6.818 -5.980 -2.6180  2.088  6.341
6  -0.0363  5.356 -1.520 -0.4347  0.750 -8.160
3   4.1900  0.973 -2.587  0.0279  0.186  2.250
4   1.4373  1.808 -0.282  0.5497 -5.776 -1.019
2   5.9669 -1.031 -1.916 -0.4118  2.199  3.324
9   6.5112 -3.713  1.713 -0.2282  0.687 -1.189
12  4.5293 -1.770  0.431 -1.0630  2.382 -0.270
10  6.6891 -2.861  0.967  0.2044  0.985 -0.223
11  1.2203  0.456 -3.714 -0.7908  0.711  1.090
21 -0.5111 -0.255  6.721 -5.5111  2.631  1.369


Site constraints (linear combinations of constraining variables)

     RDA1   RDA2    RDA3    RDA4   RDA5    RDA6
18 -2.903  2.900  2.9700 -1.6903  3.548 -4.2398
15  0.212 -1.774  0.7305 -0.0971  0.113 -3.5595
24 -2.137 -1.164 -1.2920 -6.0664 -6.036  2.7987
27 -4.190 -3.909 -3.8627 -0.6262  1.364 -2.3742
23 -1.972 -1.258  3.3549 -0.6227  2.577 -1.4180
```

```
19 -0.783 -2.879  2.5464 -2.1714 -0.871 -1.9391
22 -2.532 -0.018  1.4792  6.5995  2.714  1.8842
16 -1.531  2.485  1.1659  3.1493  0.908  1.9966
28 -4.201 -6.144 -6.0141  1.7260 -0.590  0.3341
13 -0.847  2.092 -0.0431  7.1488 -7.367 -1.5646
14  0.185 -0.598  3.6178  1.7865 -0.794  3.3712
20 -1.327 -0.370  5.0867 -2.3813 -2.715  3.4924
25 -1.869 -1.844 -0.8828 -0.9162  3.753  0.0744
7   1.248  6.720 -2.3326 -1.9183  1.288 -5.2986
5  -1.542  5.814 -4.2502 -1.3461  1.703  7.2880
6   0.390  2.151 -0.9125  1.1885  1.691 -1.7612
3   3.022  1.792 -3.7574 -1.3724  0.421 -1.5472
4   0.454  2.272 -1.4606 -0.4744 -6.743 -3.3667
2   5.808 -0.458 -1.7796  0.3974  0.629  3.9649
9   6.701 -2.670  3.0241  0.4716 -0.610 -0.5504
12  1.828 -0.646  1.4584  3.5223  1.565 -0.2831
10  5.757 -3.634 -1.2768 -1.5152  1.112 -0.7829
11  2.218 -0.166 -2.1132 -0.5360  2.032  2.2743
21 -1.988  1.305  4.5437 -4.2557  0.306  1.2065


Biplot scores for constraining variables

              RDA1    RDA2    RDA3     RDA4     RDA5     RDA6
N         -0.3745  0.4001 -0.3203  0.12100 -0.02109  0.24811
P         -0.0789 -0.6774 -0.1843 -0.21289 -0.34655 -0.00648
K         -0.2015 -0.5461 -0.0711  0.24851 -0.53040 -0.10011
Ca        -0.2354 -0.5952  0.0444 -0.12641 -0.10692  0.17946
Mg        -0.2580 -0.5033  0.0562 -0.25027 -0.34716 -0.03753
S          0.1104 -0.4698  0.0775  0.05345 -0.72593 -0.09419
Al         0.6228  0.3206 -0.2191  0.05557 -0.53897 -0.21954
Fe         0.5385  0.3625 -0.3625 -0.00509 -0.20495 -0.17628
Mn        -0.5318 -0.5560 -0.1046  0.11700  0.07688 -0.06308
Zn        -0.1893 -0.4609 -0.0839 -0.21412 -0.55727  0.36851
Mo         0.0825  0.1513  0.1257 -0.05456 -0.71692 -0.16311
Baresoil -0.5845  0.0252  0.7051 -0.17972 -0.12150 -0.06767
Humdepth -0.5282 -0.4989  0.3939  0.10590 -0.00588 -0.22862
pH         0.4853  0.2092 -0.4076 -0.28236  0.09884  0.16873
```

Von der Gesamtvarianz kann 1459.9/1825.7*100=80% durch die Umweltvariablen erklärt werden.
An der Varianzerklärung haben insbesondere die ersten zwei bis drei Achsen (mit insgesamt 67-72%) den höchsten Anteil.
Die *scores* für Arten und Aufnahmen sind ähnlich zu interpretieren wie bei der Hauptkomponentenanalyse. Allerdings können sie anders skaliert sein. In diesem Fall erfolgt die Skalierung mit einem Fokus auf die Arten. Wie skaliert wird, ist aber von Software zu Software und von Lehrbuch zu Lehrbuch unterschiedlich.

Bei der Redundanzanalyse entsprechen die *site constraints* dem Ergebnis der vorgeschalteten Regression; es handelt sich also um lineare Kombinationen der Umweltdaten. Die *site scores* entsprechen fast dem Ergebnis einer „normalen“ Hauptkomponentenanalyse. Da im R-*package* **vegan** die Ergebnisse einer Redundanzanalyse in einem Objekt für die Ergebnisse einer Kanonischen Korrespondenzanalyse abgelegt werden, erfolgt auch die Benennung der Elemente entsprechend. Hier muss darauf geachtet werden, dass es sich in Wirklichkeit jedoch immer um die analogen Begriffe der linearen Ordination handeln muss. Eine genauere Beschreibung erfolgt daher im Kapitel „Kanonische Korrespondenzanalyse“.

Die *scores* für die Umweltvariablen zeigen, dass auf der ersten Achse Stickstoff der wichtigste Parameter ist und negativ mit der Achse korreliert. Der wichtigste Parameter der zweiten Achse ist Phosphor und er verhält sich negativ.

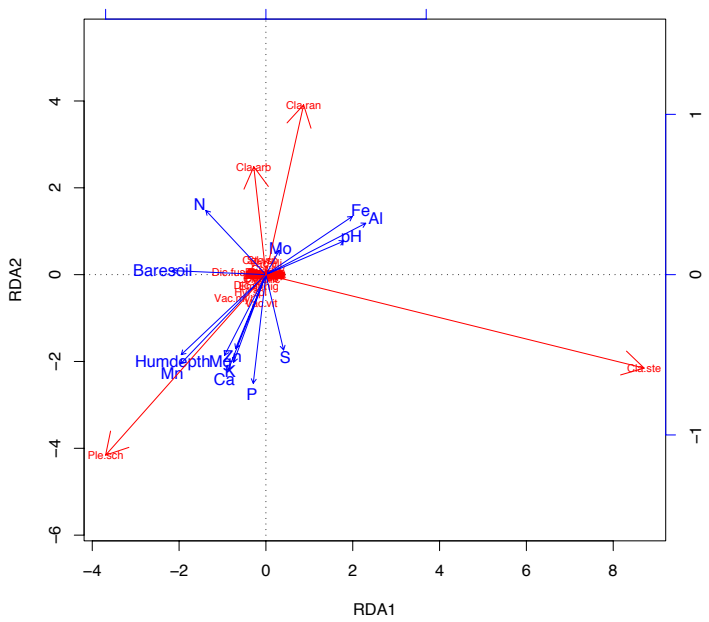


Abbildung 12.1.: *Plot* über die Ausgabe der Funktion `rda`.

Die Ausgabe des *biplot* erfolgt mit dem `plot`-Befehl. Allerdings müssen, da es sich um eine lineare Ordination handelt, die Arten als Pfeile dargestellt werden. Dies erfolgt in einem zweiten Schritt für diejenigen Arten, deren *scores* groß genug (hier > 1) sind (Abb. 12.1).

```
> plot(vare.rda, display = c("sp", "bp"))
> scor <- scores(vare.rda, display = "sp")
> arrows(0, 0, scor[abs(scor[, 2]) > 1, 1], scor[abs(scor[, 2]) > 1, 2],
+     col = "red")
```

Während die meisten Arten in diesem Beispiel um den Ursprung versammelt sind (sich also mehr oder weniger indifferent gegenüber den Umweltparametern verhalten), kann man erkennen, dass *Pleurocium schreberi* (ein Moos) positiv mit dem Gehalt an Mangan, Kalzium, Phosphor und der Humusauflage korreliert. Entgegengesetzt verhalten sich zwei Cladonien (Rentierflechten), während eine dritte negativ mit vegetationsfreiem Boden korreliert ist.

12.1.3. Variablenauswahl

Bei der zuvor durchgeführten Redundanzanalyse haben wir alle Variablen in die Analyse eingegeben. Dies ist jedoch nicht sinnvoll, weil

- dadurch der Ordinationsraum zu groß werden kann und eine Bogeneffekt auftreten kann,
- viele Daten miteinander korrelieren können (Multikollinearität) und somit die Analyse schwächen,
- die wichtigen Variablen so ggf. nicht erkannt werden können.

Als Ausweg gibt es mehrere Möglichkeiten:

1. Eine Auswahl der Umweltvariablen erfolgt an Hand bestimmter Hypothesen bzw. zuvor erworbenen Wissens.

2. Die Auswahl erfolg durch Dimensionsreduktion mit Hilfe einer Hauptkomponentenanalyse. Es werden jeweils nur die wichtigsten Variablen (mit der höchsten Ladung) der wichtigsten Hauptkomponenten in das System eingegeben.

3. Es erfolgt eine automatisierte Variablenauswahl über das Akaike-Informations-Kriterium (AIC). Dies kann automatisiert mit folgenden Funktionen von Jari Oksanen (Finnland, Autor von **vegan**) durchgeführt werden:

```
> "deviance.capscale" <- function(object, ...) NA
> "deviance.cca" <- function(object, ...) object$CA$tot.chi * object$grand.tot
> "deviance.rda" <- function(object, ...) object$CA$tot.chi * (nrow(object$CA$Xbar) -
+     1)
> "extractAIC.cca" <- function(fit, scale = 0, k = 2, ...) {
+     n <- nrow(fit$CA$Xbar)
+     edf <- 1
+     if (!is.null(fit$CCA$rank))
+         edf <- edf + fit$CCA$rank
+     if (!is.null(fit$pCCA$rank))
+         edf <- edf + fit$pCCA$rank
+     RSS <- deviance(fit)
+     dev <- if (scale > 0)
+         RSS/scale - n
+     else n * log(RSS/n)
+     c(edf, dev + k * edf)
+ }
```

Die schrittweise Auswahl erfolgt dann analog zur Funktion `step` bei linearen Modellen (Regression, Varianzanalysen, GLM etc.).

4. kann eine schrittweise Reduktion bzw. Addition von Variablen erfolgen und mit Hilfe der Funktion `anova.cca` wird getestet, ob das neue Modell sich signifikant vom vorhergehenden unterscheidet.

Die Signifikanzprüfung unseres Modells über einen Permutationstest zeigt, dass das Ergebnis signifikant ist:

```
> anova(vare.rda)

Permutation test for rda under reduced model

Model: rda(X = varespec, Y = varechem)
         Df  Var    F N.Perm Pr(>F)
Model    14 1460 2.57    100   0.01 **
Residual  9  366
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

Eine ausführlichere Darstellung dieses Tests und des Punktes 4 der Variablenauswahl erfolgt jedoch erst später.

12.2. Kanonische Korrespondenzanalyse (CCA)

12.2.1. Beschreibung

Die Kanonische Korrespondenzanalyse ist die Erweiterung der Korrespondenzanalyse um eine weitere Matrix mit Umweltdaten. Die Berechnung erfolgt analog einer Redundanzanalyse (RDA), allerdings mit modifizierter Algebra, um eine X^2-Distanz statt der Euklidischen Distanz zu erhalten.

Der „Bogeneffekt“ der Korrespondenzanalyse tritt normalerweise bei der Kanonischen Korrespondenzanalyse nicht mehr auf, da die Arten und Aufnahmen ja lineare Kombinationen der bereitgestellten Matrix der Umweltvariablen sind. Wenn aber die Anzahl der Umweltvariablen zu hoch wird, erfolgt kaum noch eine Einschränkung im Ordinationsraum und es kann dann doch zu einem Bogeneffekt kommen.

12.2.2. Beispiel

In diesem Beispiel analysieren wir einen Datensatz über Nagetiere in fragmentierten Gebüschen in Kalifornien (`bolger4.spe`) (Bolger et al. 1997). Die Arten *Rattus rattus* (Hausratte) und *Mus musculus* (Hausmaus) sind in Kalifornien gebietsfremd, alle anderen heimisch. Es soll untersucht werden, auf welche Umweltfaktoren die Arten am stärksten reagieren. An Umweltparametern (`bolger4.env`) werden nur drei in die Analyse eingegeben (um Multikollinearität, d.h. eine Vielzahl hoch korrelierender Variablen zu vermeiden): Die Flächengröße der Fragmente, die Entfernung zum nächsten Canyon (als Besiedelungsquelle) und das Alter der Fragmente. Die entsprechende Berechnung wird mit der Funktion `cca` (*package* **vegan**) durchgeführt und sieht folgendermaßen aus:

```
> bolger4 <- read.table("bolger4.txt", header = T)
> bolger4.spe <- bolger4[, 5:13]
> bolger4.env <- bolger4[, 2:4]
> rownames(bolger4.spe) <- rownames(bolger4.env) <- bolger4[, 1]
> bolger4.cca <- cca(bolger4.spe, bolger4.env)
```

Dies ergibt folgendes Ergebnis (was im Aufbau mit dem der Redundanzanalyse vergleichbar ist):

```
> bolger4.cca

Call:
cca(X = bolger4.spe, Y = bolger4.env)

              Inertia Rank
Total           1.702
Constrained     0.717    3
Unconstrained   0.985    8
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
  CCA1   CCA2   CCA3
0.5954 0.0827 0.0391

Eigenvalues for unconstrained axes:
     CA1      CA2      CA3      CA4      CA5      CA6      CA7      CA8
0.460023 0.260034 0.172009 0.058869 0.021712 0.009201 0.002469 0.000584
```

Die Gesamtvarianz (*Intertia, mean squared contingency coefficient*) beträgt 1.70, die zu 42.1% (0.72/1.71*100) durch die Umweltvariablen (*contraining variables*) erklärt werden kann. Der Rest (0.98/1.71*100=57.9%) kann nicht durch die Umweltvariablen erklärt werden.

Bei drei Umweltvariablen sind auch nur die ersten drei Ordinationsachsen durch Umweltvariablen eingeschränkt. Davon kann die 1. Achse fast 35% erklären. Die erste nicht eingeschränkte Achse (Achse 4) kann nur 27% erklären.

Die *scores* für Arten und Aufnahmen sind ähnlich zu interpretieren wie bei der Korrespondenzanalyse. Allerdings können sie anders skaliert sein. In diesem Fall erfolgt die Skalierung mit einem Fokus auf die Arten. Wie skaliert wird, ist aber von Software zu Software und von Lehrbuch zu Lehrbuch unterschiedlich.

Die *site constraints* unterscheiden sich von den *site scores* dadurch, dass sie keine gewichteten Mittel der Art scores sind, sondern sich aus den linearen Kombinationen der Umweltvariablen im Zuge der zuvor durchgeführten multiplen Regression errechnen. Es handelt

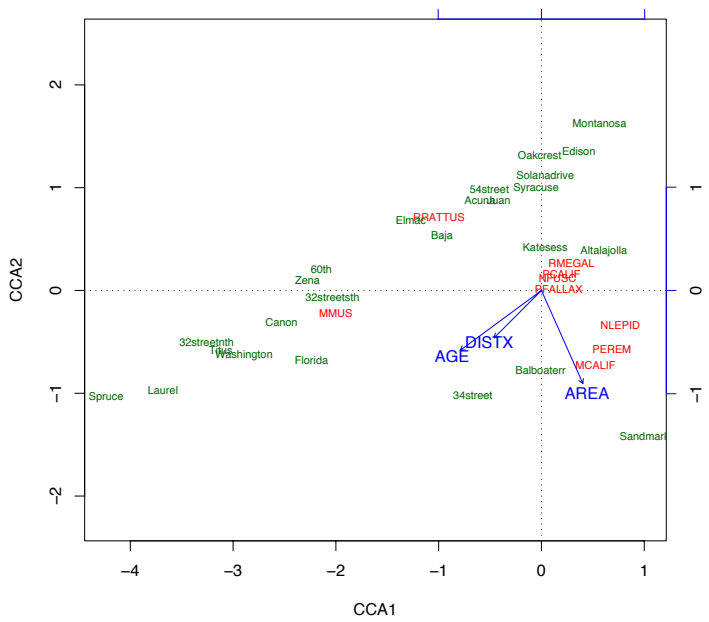


Abbildung 12.2.: *Plot* über die Ausgabe der Funktion `cca`.

sich hierbei um die „vorhergesagten“ Werte für die Aufnahmen aus der Regression. In der Literatur herrscht Uneinigkeit darüber, welches die besseren Werte seien, um sie graphisch darzustellen: Die Einen (z.B. Palmer 1993) sagen, dass die *LC scores* (*linear combinations, site constraints*) die besseren Werte seien, da diese sich auf die Umweltvariablen beziehen und eine Weiterentwicklung gegenüber den *WA scores* (*weighted averages, site scores*) sind. Bei Benutzung der *WA scores* sei das Ergebnis kaum besser als eine normale Korrespondenzanalyse. Andere (z. B. McCune 1997) haben allerdings festgestellt, dass Umweltvariablen mit starkem „Rauschen“ (und alle Umweltvariablen rauschen) eine starke Auswirkung auf die *LC scores* haben, während *WA scores* davon kaum berührt werden.

Schlussendlich sind noch die berechneten *scores* für die Umweltvariablen angegeben. In diesem Fall sind die Umweltvariablen kontinuierlich. Wenn diese aber kategorial sind, werden bei den *biplot scores* für kategoriale Daten die Kontraste der Kategorien angegeben und es gibt noch eine Ausgabe für *centroid scores*. Kategoriale Daten werden nämlich nicht als Pfeile (Gradienten) aufgefasst, sondern sind die Lagemittel der kategorialen Daten. Im Gegensatz zu den Gradienten nehmen diese also nicht in einer Richtung zu, in der Gegenrichtung ab und verhalten sich neutral senkrecht hierzu, sondern sie nehmen gleichmäßig mit der Entfernung von den Lagemitteln (Zentroiden) ab.

Wir können der Tabelle entnehmen, dass die Flächengröße mit dem Gradienten der ersten Achse zunimmt, während insbesondere das Alter der Fragmente abnimmt. Auf der zweiten Achse ist besonders die Fläche wichtig, die negativ mit der Achse korreliert. Die dritte Achse lassen wir bei der Betrachtung aus, da sie kaum noch zur Varianzerklärung beiträgt. Der *plot* zeigt folgendes Bild (Abb. 12.2):

```
> plot(bolger4.cca, display = c("sp", "lc", "bp"))
```

Die Graphik (Abb. 12.2) zeigt, dass insbesondere die Hausmaus (*M. musculus*) mit älteren Habitatfragmenten assoziiert ist, die weiter entfernt von den Canyons liegen. Die Hausratte als zweite gebietsfremde Art ist in den kleinen Fragmenten häufiger. Drei heimische Arten bevorzugen große Fragmente, während die vier anderen heimischen Arten jüngere Fragmente in der Nähe der Canyons bevorzugen.

Aufschlussreich bei jeder Form der Kanonischen Ordination ist der Vergleich mit der nicht eingeschränkten Variante der angewandten Methode (hier Korrespondenzanalyse).

12.3. Partielle kanonische Ordination

Unter Umständen ist die Auswirkung bestimmter (oft dominanter) Umweltgradienten bekannt und für eine Analyse daher nicht von weiterem Interesse. Vielmehr soll untersucht werden, wie sich die Art-Umweltbeziehungen verhalten, wenn der Einfluss dieses Faktors als bekannt vorausgesetzt wird. Ein Beispiel ist die Analyse des Flechtenrasens von Oksanen. Allerdings nehmen wir diesmal an, dass der Einfluss des pH-Wertes bekannt ist, und uns nicht weiter interessiert. Die Analyse soll daher unter der Annahme erfolgen, dass der Einfluss des pH-Wertes als bekannt berechnet wurde und die Restvarianz durch z. B. die Gehalte an Kalzium, Aluminium und Stickstoff erklärt werden soll. In diesem Beispiel sind die Kalzium-, Aluminium- und Stickstoff-Gehalte Variablen und der pH-Wert eine Kovariable, auch konditionale (*conditional*) Variable genannt. Es können auch mehrere Kovariablen angegeben werden. Technisch wird dies so durchgeführt, dass erstmal eine Regression der Kovariablen gegen alle Artvariablen durchgeführt wird und dann die eigentlichen Umweltvariablen gegen die Residuen dieser ersten Analyse gerechnet werden. Die daraus resultierende Matrix wird an den entsprechenden Ordinations-Algorithmus (PCA, CA) weitergegeben.

Zum Vergleich soll eine Berechnung ohne und eine Berechnung mit Kovariablen durchgeführt werden:

```
> data(varespec)
> data(varechem)
> vare.cca <- cca(varespec ~ Ca + Al + N, varechem)
> vare.cca

Call:
cca(formula = varespec ~ Ca + Al + N, data = varechem)

              Inertia Rank
Total           2.083
Constrained     0.528    3
Unconstrained   1.555   20
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
  CCA1   CCA2   CCA3
0.3521 0.1305 0.0457

Eigenvalues for unconstrained axes:
   CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8
0.3196 0.2167 0.1996 0.1795 0.1665 0.1136 0.0852 0.0785
(Showed only 8 of all 20 unconstrained eigenvalues)
```

Und jetzt die Analyse mit pH als Kovariable:

```
> vare.pcca <- cca(varespec ~ Ca + Al + N + Condition(pH), varechem)
> vare.pcca

Call:
cca(formula = varespec ~ Ca + Al + N + Condition(pH), data = varechem)

              Inertia Rank
Total           2.083
Conditional     0.146    1
Constrained     0.476    3
Unconstrained   1.461   19
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
```

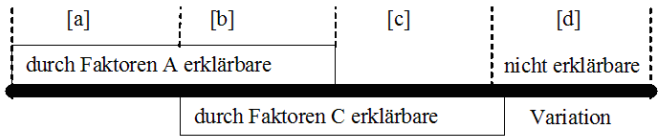


Abbildung 12.3.: Aufteilung der Variation einer multivariaten Art-Reaktions-Matrix Y zwischen Bodenfaktoren (Matrix B) und Altersfaktoren (Matrix A). Die Länge des horizontalen Balkens entspricht 100% der Variation in Y (nach Borcard et al. 1992, verändert).

```
  CCA1   CCA2   CCA3
0.2784 0.1350 0.0629

Eigenvalues for unconstrained axes:
   CA1    CA2    CA3    CA4    CA5    CA6    CA7    CA8
0.3054 0.1996 0.1810 0.1703 0.1642 0.1131 0.0839 0.0781
(Showed only 8 of all 19 unconstrained eigenvalues)
```

Der Vergleich der Inertia (Variabilität) zeigt, dass bei der 1. Analyse der durch die Variablen erklärte Anteil bei 0.5283/2.0832=0.2536 (25,36%) liegt. Bei der zweiten Variable ist der Anteil geringer (0.4763/2.0832=0.2286), weil diejenige Variabilität, die auch schon durch den pH-Wert erklärt werden kann, abgezogen wird. Hier kommt dann der Anteil *Conditional* hinzu, der durch den pH-Wert erklärt werden kann (0.1458/2.0832=0.07). Dem entsprechend liegen auch die Eigenwerte der Ordinationsachsen bei der partiellen Analyse immer unter denen der ersten Analyse.
Das Ergebnis der partiellen CCA (mit dem Befehl *summary*) entspricht von seinem Aufbau her ansonsten dem einer normalen direkten Ordination.

Die Betrachtung von Kovariablen ermöglicht uns auch, ANOVA-ähnliche Analysen verschiedener Block-Designs durchzuführen (siehe Beispiel im Kapitel 13.3.1 auf Seite 210).

12.4. Partitionierung der Variation einer multivariaten Reaktions-Matrix

Borcard et al. (1992) haben eine Methode entwickelt, um die relativen Anteile von Umweltfaktoren und von räumlichen Faktoren zu trennen und ihren Einfluss auf die Pflanzenverteilung zu quantifizieren (Partitionierung oder Aufteilung der Variation, variation partitioning). Die Methoden wurden auch von Øklaand & Eilertsen (1994) sowie Legendre and Legendre (1998, S. 769-779) ausführlich beschrieben. Dabei wir der Anteil erklärbarer Variabilität zwischen zwei Gruppen A (bzw. Matrix Y) und C (bzw. Matrix Z), z. B. Boden und Wasser, physikalisch-chemische und geographische, Nutzungs- und natürliche Faktoren aufgeteilt. Dieses ist in Abb. 12.3 verdeutlich: Dabei entspricht der Anteil [a] der nur auf einer Faktoren(gruppe) „A“ beruhenden Variation; [b] ist die gemeinsam durch Faktoren A und C zu erklärende Variation und [c] die nur durch Faktoren „C“ zu erklärende Variation. Anteil [d] ist auf Grund der Daten nicht zu erklären (nicht berücksichtigter Gradient oder Rauschen/Zufall).

In die vorliegende Analyse sind nur die signifikanten Umweltfaktoren eingegangen (s. o.). Der Anteil der Variation errechnet sich aus der Summe der kanonischen Eigenwerte geteilt durch die Summe aller Eigenwerte ($(\sum$ canonical eigenvalues)/total inertia) nach folgenden Analysen (vgl. Legendre and Legendre 1998, S. 771-772):

1. [a+b] wird durch eine CCA der Artmatrix mit erklärender Matrix A errechnet.

2. [b+c] wird durch eine CCA der Artmatrix mit erklärender Matrix C errechnet.

3. [a+b+c] ist das Ergebnis einer CCA der Artmatrix mit allen signifikanten Faktoren aus A und C gemeinsam als erklärender Matrix.

4. [a] wird durch eine partielle CCA der Artmatrix mit erklärender Matrix A und der Matrix C als Kovariablen ermittelt oder aus [a+b+c] − [b+c] errechnet.

5. [c] wird durch eine partielle CCA der Artmatrix mit erklärender Matrix C und der Matrix A als Kovariablen ermittelt oder aus [a+b+c] − [a+b] errechnet.

6. [b] kann auf folgende Weisen berechnet werden: [b] = [a+b] − [a] oder [b] = [b+c] − [c] oder [b] = [a+b+c] − [a] − [c] oder [b] = [a+b] + [b+c] − [a+b+c]

7. [d] = 1 − [a+b+c]

Økland (1999) hat diese Methode kritisch untersucht und gibt einige Empfehlungen. Unter anderem ist auf folgende Punkt zu achten:

- Die Gesamtvarianz zwischen unterschiedlichen Datengruppen ist schwerlich vergleichbar.
- Die Ordinationsmethode zur Bestimmung der Variationsaufteilung richtet sich nach der Gradientenlänge der 1. Ordinationsachse der DCA (= β-Diversität).
- Bei der Analyse sollten daher nur die relativen Anteile der erklärbaren Variation berücksichtigt werden.

Bei der Berechnung von geographischen Faktoren ist es (leider) üblich, nicht nur die Rechts- und Hochwerte und ihre Interaktion zu berücksichtigen, sondern eine so genannte „Trend-Oberfläche“ (*trend surface*) dritten Grades zu berechnen. Dazu werden die Rechts- und Hochwerte (bzw. Längen- und Breitengrade) quadriert und zur dritten Potenz erhoben und die entsprechenden Interaktionsterme berechnet. Die Formel für eine solche Trendoberfläche sieht folgendermaßen aus:

$$\text{Abhängige} = x + y + x \cdot y + x^2 + y^2 + x^2 \cdot y + x \cdot y^2 + x^2 \cdot y^2 + x^3 + y^3,$$

wobei x dem Rechtswert und y dem Hochwert entsprechen.

Werden für die beiden Faktoren der Variablen (A bzw. Y) und der Kovariablen (C bzw. Z) keine Vektoren (Einzelvariablen) benutzt, sondern tatsächlich Matrizen, ist es sinnvoll, für jede Matrix getrennt zuerst eine Datenreduktion (über eine PCA oder eine schrittweise Selektion) durchzuführen.

12.4.1. Beispiel

In unserem Beispiel soll untersucht werden, welchen Effekt die Düngergabe (Dosis) bzw. die Deckung von Gerste auf die Mächtigkeit von Ackerkräutern hat (Pyšek & Lepš 1991). Die Düngergabe ist dabei in drei Stufen verschlüsselt: 0 für ungedüngt, 1 für 70 kg N/ha und 2 für 140 kg N/ha.
Das Ergebnis der wichtigsten Parameter unserer Analyse sieht folgendermaßen aus:

```
> fertil <- load("fertil.r")
> fertil.cca <- cca(fertil.spe, fertil.env)
> fertil.cca

Call:
cca(X = fertil.spe, Y = fertil.env)

              Inertia Rank
Total           5.304
```

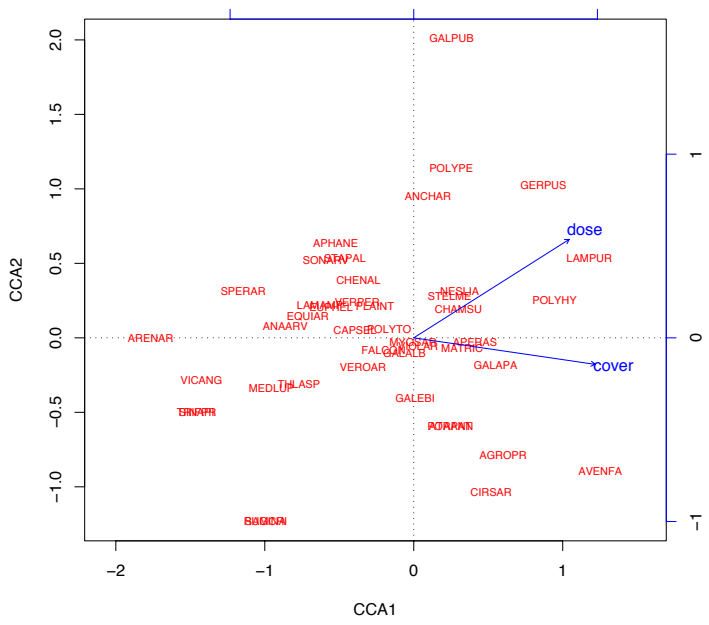


Abbildung 12.4.: Biplot der CCA des Düngeexperiments. Pfeile geben die Einflussrichtung der manipulierten Behandlungen an, Artenkürzel die Lage der Arten bezüglich der Behandlungen.

```
Constrained     0.278    2
Unconstrained   5.027   42
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
  CCA1   CCA2
0.2003 0.0773

Eigenvalues for unconstrained axes:
  CA1   CA2   CA3   CA4   CA5   CA6   CA7   CA8
0.446 0.335 0.279 0.257 0.242 0.226 0.207 0.196
(Showed only 8 of all 42 unconstrained eigenvalues)
```

Mit den beiden Variablen können wir nur gut 5% (0.278/5.304)) der Variation in der Artengemeinschaft überhaupt erklären. Nichtsdestotrotz kann dies signifikant sein. Deshalb wird das Ergebnis mit einer ANOVA getestet. Das Ergebnis ist höchst signifikant:

```
> anova(fertil.cca, step = 999)

Permutation test for cca under reduced model

Model: cca(X = fertil.spe, Y = fertil.env)
         Df Chisq    F N.Perm Pr(>F)
Model     2  0.28 1.16    999 <0.001 ***
Residual 42  5.03
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

Ein Plot zeigt folgendes Ergebnis (Abb. 12.4):

```
> plot(fertil.cca, display = c("bp", "sp"))
```

Die meisten Arten sind auf der linken Seite des Ordinationsdiagrammes, den Gradienten Dosis und Deckung entgegengesetzt. Einer der Gründe hierfür ist, dass natürlich dort, wo die Gerste eine hohe Deckung hat, die Wildkräuter nicht mehr wachsen können. Aber uns interessiert jetzt, welche dieser beiden Variablen wichtiger für die Gesellschaftskomposition ist. Hierzu werden zwei Analysen durchgeführt, bei der einmal die Dosis Variable und Deckung Kovariable sind, das zweite Mal ist es anders herum.

```
> fertil.cca1 <- cca(fertil.spe ~ dose + Condition(cover), fertil.env)
> fertil.cca1

Call:
cca(formula = fertil.spe ~ dose + Condition(cover), data = fertil.env)

              Inertia Rank
Total           5.304
Conditional     0.198    1
Constrained     0.080    1
Unconstrained   5.027   42
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
CCA1
0.08

Eigenvalues for unconstrained axes:
  CA1   CA2   CA3   CA4   CA5   CA6   CA7   CA8
0.446 0.335 0.279 0.257 0.242 0.226 0.207 0.196
(Showed only 8 of all 42 unconstrained eigenvalues)
```

Durch die Düngerdosis alleine können also nur $0.080/5.304 = 1.5\%$ erklärt werden. Dies sind aber 28.8% ($=0.080/(5.304 - 5.027)$) der erklärbaren Variabilität.

```
> fertil.cca2 <- cca(fertil.spe ~ cover + Condition(dose), fertil.env)
> fertil.cca2

Call:
cca(formula = fertil.spe ~ cover + Condition(dose), data = fertil.env)

              Inertia Rank
Total           5.304
Conditional     0.166    1
Constrained     0.112    1
Unconstrained   5.027   42
Inertia is mean squared contingency coefficient

Eigenvalues for constrained axes:
 CCA1
0.112

Eigenvalues for unconstrained axes:
  CA1   CA2   CA3   CA4   CA5   CA6   CA7   CA8
0.446 0.335 0.279 0.257 0.242 0.226 0.207 0.196
(Showed only 8 of all 42 unconstrained eigenvalues)
```

Durch die Deckung der Gerste kann ca. 2.2% ($= 0.112/5.304$) der Variabilität in der Ackerkrautgesellschaft erklärt werden. Dies sind aber 40.3% ($= 0.112/(5.304-5.027)$) der erklärbaren Variabilität. Daraus folgt, dass die erklärbare Variabilität aufgeteilt werden kann in 28.8% erklärt durch die Düngerdosis, 40.3% erklärt durch die Gerstendeckung und $100\%-(28.8\% + 40.3\%) = 69.1\%$ durch Dünger und Gerstendeckung gemeinsam (Abb. 12.5).

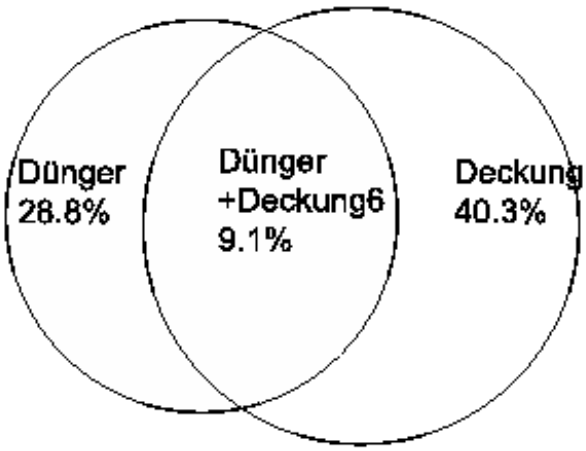


Abbildung 12.5.: Aufteilung der Erklärung der Variation der Ackerkrautgesellschaft durch Düngung, Gerstendeckung sowie ihren gemeinsamen Beitrag.

12.5. Weitere Ordinationstechniken

Weitere häufig verwendete Ordinationstechniken sind u. a. die Kanonische Korrelations-Analyse und Multidimensionale Skalierung (inkl. Haupt-Koordinaten-Analyse).

Die **Kanonische Korrelationsanalyse** (CCorA) verhält sich zur Kanonischen Analysen (RDA, CCA) wie die Korrelation zur Regression. Mit Hilfe der CCorA können Achsen mit maximaler linearer Korrelation zueinander gefunden werden. Soll aber eine Matrix $\mathbf{Y}$ mit Hilfe einer Matrix $\mathbf{X}$ erklärt werden, sind RDA und CCA die Methoden der Wahl.

Bei der **Multidimensionale Skalierung** (Multidimensional Scaling, MDS) werden Objekte mit Hilfe von Ähnlichkeits- oder Unähnlichkeits- (Distanz-) Maßen zueinander in Beziehung gesetzt. Eine der häufig angewendeten Verfahren von MDS ist die Haupt-Koordinaten-Analyse (PCoA). Allerdings werden statt Assoziationsmaße (Kovarianz- bzw. Korrelationsmatrizen) Distanzmaße genutzt. Diese werden auf die gleiche Art weiter analysiert, wie dies auch bei der PCA der Fall ist. Tatsächlich entspricht eine PCoA mit dem Distanzmaß „Euklidische Distanz“ rechnerisch einer PCA. Durch weiter ausgeklügelte Verfahren der MDS lässt sich iterativ der *fit* zwischen den Unähnlichkeiten und Distanzen der Objekte optimieren. Die nicht-metrische Multidimensionale Skalierung (nMDS) verwendet als Distanzmaße die Rangfolgen der Distanzen (häufig Bray-Curtis).

Die ***Analysis of Similarity*** (ANOSIM) als eine Art Varianzanalyse über die Distanzmaße verschiedener Gruppen anzusehen. Die Teststatistik berechnet sich aus der Rangfolge der Bray-Curtis-Distanzen zwischen verschiedenen Gruppen und innerhalb dieser Gruppen. Das Testen der Nullhypothese erfolgt wieder über Permutationstests.

12.6. Multivariate Distanzmaße und Unähnlichkeitsmaße

Während einige multivariate Verfahren die Varianz aus Korrelations- oder Kovarianz-Matrizen berechnen, ist für andere Verfahren notwendig, die (Un-)Ähnlichkeit zwischen den Objekten zu berechnen. Prinzipiell kann man zwischen symmetrischen und asymmetrischen Maßen unterscheiden. Bei den symmetrischen Maßen geht man davon aus, dass das Vorkommen einer Art genauso viel aussagt, wie das Nicht-Vorkommen. Beim Vergleich von Artenlisten werden also auch „Doppel-Nullen“ (also zwei Aufnahmeflächen, wo in beiden eine Art nicht vorkommt) berücksichtigt. Nun kann man aber auch gut argumentieren, dass das Nicht-Auftreten einer Art nicht damit zusammenhängt, dass zwei Untersuchungsflächen gleichartig sind. Vielmehr kann die eine einen Faktor (z. B. Wasser) im Überfluss, die andere als Man-

gelfaktor haben, die zu untersuchende Art mag es aber lieber im mittleren Bereich. Daher werden bei asymmetrischen Distanzmaßen die „Doppel-Null-Vorkommen“ ausgelassen.

Von der Vielzahl unterschiedlicher Distanzmaße soll nur auf einige wenige, häufig genutzte eingegangen werden. Um die Maße zu veranschaulichen, wird folgende Art-Abundanz/Aufnahme-Matrix zu Grunde gelegt, für die beispielhaft Ergebnisse präsentiert werden. Dabei sind Aufnahme 1 und 3 einander qualitativ ähnlich; Aufnahme 2 unterscheidet sich hingegen deutlich von beiden.

	Art 1	Art 2	Art 3
Nr. 1	0	1	1
Nr. 2	1	0	0
Nr. 3	0	4	4

```
> (mat <- matrix(c(0,1,1, 1,0,0, 0,4,4), nrow=3, byrow=T))
```

12.6.1. Euklidische Distanz

$$\sqrt{\sum_{j=1}^{p}(y_{1j}-y_{2j})^2}$$

Die Euklidische Distanz ist eines der am meisten benutzten Distanzmaße und beruht auf dem Satz des Pythagoras. Es gibt keine definierte Obergrenze an Unähnlichkeit. Dies kann aber umgangen werden, wenn mit standardisierten Variablen gerechnet wird. Eines der bekanntesten Probleme der Euklidischen Distanz sind die „Doppel-Nullen“. Die Distanz von Aufnahme Nr. 1 zu Aufnahme Nr. 2 in unserem Beispiel ist die geringste, weil sie die ähnlichsten Abundanzen haben. Wenn zwei Aufnahmen eine Reihe von Arten mit „gemeinsamen“ Abundanzen von Null haben, sind diese sich rechnerisch sehr ähnlich, obwohl sie ökologisch kaum etwas miteinander zu tun haben. Diese Problem ist insbesondere bei der Hauptkomponenten-Analyse von Belang.

Im R-*package* **vegan** kann man die verschiedenen Distanzmaße in der Funktion `vegdist` abrufen. Dies sei hier nur exemplarisch gemacht, die Hilfe gibt ausführliche Details zur Berechnung. Wem die 13 Maße von Jari Oksanens `vegdist` nicht genug sind, der hat zwei Optionen: entweder selber implementieren in der ziemlich genialen `designdist`-Funktion oder im *package* **simba** stöbern: die Funktion `com.sim` stellt weitere 56 Distanzmaße für binäre (Vorkommen/Nichtvorkommen) Datensätze zur Verfügung. Für nicht-binäre Daten ruft es auch `vegdist` auf! Schließlich gibt es noch die **ade4**-Funktionen `binary.dist`, `dist.quant` und `dist.prop`, die ebenfalls Distanzmaße anbieten.

```
> require(vegan)
> vegdist(mat, method="euclid")
```

```
         1        2
2 1.732051
3 4.242641 5.744563
```

Um das Ergebnis als vollständige Matrix zu erhalten, nicht nur als untere Dreiecksmatrix:

```
> as.matrix(vegdist(mat, method="euclid"))
```

```
         1        2        3
1 0.000000 1.732051 4.242641
2 1.732051 0.000000 5.744563
3 4.242641 5.744563 0.000000
```

Übersetzt in eine normale Tabelle bedeutet dies:

	Art 1	Art 2	Art 3
Nr. 1	0.000	1.732	4.243
Nr. 2	1.732	0.000	5.746
Nr. 3	4.243	5.745	0.000

12.6.2. Manhattan

$$\sum_{j=1}^{p} |y_{1j} - y_{2j}|$$

Bei diesem Maß, das der Euklidischen Distanz verwandt ist, handelt es sich um die Summe der absoluten Differenz der Elemente jeder Variablen zwischen zwei Objekten. Sie wird durch Variablen mit großen Werten stark beeinflusst. Der Name rührt aus der Vorstellung her, dass die Distanzen wie im schachbrettartigen Straßenmuster von Manhatten hergeleitet werden.

	Art 1	Art 2	Art 3
Nr. 1	0	3	6
Nr. 2	3	0	9
Nr. 3	6	9	0

12.6.3. Canberra

$$\frac{1}{NZ} \sum_{j=1}^{p} \frac{|y_{1j} - y_{2j}|}{(y_{1j} + y_{2j})}$$

NZ: Anzahl der Nicht-Null (*non zero*) Werte.
Der Canberra Koeffizient ist eine Variante der Manhatten-Distanz, schließt „Doppel-Null“ Zellen aber explizit aus. Daraus ergibt sich eine sinnvolle Distanz. In der hier präsentierten Version ist der Koeffizient auf Werte von 0 bis 1 skaliert.

	Art 1	Art 2	Art 3
Nr. 1	0	1	0.6
Nr. 2	1	0	1
Nr. 3	0.6	1	0

12.6.4. Bray-Curtis

$$\sum_{j=1}^{p} \frac{\sum_{j=1}^{p} |y_{1j} - y_{2j}|}{\sum_{j=1}^{p} (y_{1j} + y_{2j})}$$

Auch dieser Index ist eine Modifikation der Manhatten-Distanz mit einer Spanne von 0 bis 1. Im Gegensatz zum Canberra-Index tragen Unterschiede zwischen häufigen Arten genauso zum Index bei wie Unterschiede zwischen seltenen Arten. Dies kann gewünscht sein, wenn normalisierte Daten genutzt werden. Das heißt, dass insbesondere Arten mit hohen Werten (Abundanzen) diesen Index bestimmen, da diese Arten mit großer Wahrscheinlichkeit auch größere Unterscheide zwischen den Objekten aufweisen. In unserem Beispiel sind die Werte identisch mit denen des Canberra-Index.

	Art 1	Art 2	Art 3
Nr. 1	0	1	0.6
Nr. 2	1	0	1
Nr. 3	0.6	1	0

12.6.5. X^2 (Chi-Quadrat)

$$\sqrt{\sum_{j=1}^{p} \frac{y_{1j} / \sum_{j=1}^{p} (y_{1j} - y_{2j})^2 / \sum_{j=1}^{p} y_{2j}}{\sum_{i=1}^{n} y_i}}$$

Die Chi-Quadrat-Distanz errechnet sich aus der Summe der Abweichungsquadrate geteilt durch die relative Häufigkeit (Wahrscheinlichkeit) jeder Zeile in der Matrix. Die Chi-Quadrat-Distanz wird implizit bei Korrespondenz-Analysen zu Grunde gelegt. Sie ist streng genommen

nur zulässig bei Zähldaten, wie Art-Abundanzen. Wie unser Beispiel zeigt, tritt das Problem der falschen Behandlung von Aufnahmen ohne gemeinsame Arten bei Chi-Quadrat-Distanzen nicht auf.

	Art 1	Art 2	Art 3
Nr. 1	0	3.477	0
Nr. 2	3.477	0	3.477
Nr. 3	0	3.477	0

12.6.6. Jaccard

Beim Jaccard-Index handelt es sich um einen Ähnlichkeits-Index (S), der aber in der Funktion `vegdist` einfach als $1-S$ in ein Distanzmaß überführt wird. Der Jaccard Ähnlichkeitsindex ist im Gegensatz zu den vorgenannten Maßen ein qualitatives (binäres) Maß, da die Häufigkeit der Arten keine Rolle spielt. Er ist einer der am meisten verwendeten und berechnet sich als

$$S = \frac{a}{a+b+c}$$

wobei a die Anzahl der gemeinsam in zwei Aufnahmeflächen auftretenden Arten ist, während b und c die Anzahlen der nur in der einen oder anderen Aufnahmenflächen auftretenden Arten sind.

	Art 1	Art 2	Art 3
Nr. 1	0	1	0.75
Nr. 2	1	0	1
Nr. 3	0	1	0

12.7. Matrix-Vergleiche

Ähnlich wie bei bivariaten Vergleichen, ist häufig auch bei multivariaten Vergleichen von Interesse, ob Matrizen miteinander korrelieren. Dies kann z.B. der Vergleich von genetischen und räumlichen Distanzen einer Gruppe, aber auch der Vergleich räumlicher Distanzen zweier Gruppen im selben Gebiet sein.

Der **Manteltest** testet die Nullhypothese, dass zwei zu vergleichende Distanz-Matrizen nicht miteinander korrelieren. Als Teststatistik koennen mehrere Koeffizienten in Frage kommen. Der gebräuchlichste ist der z_M-Koeffizient. z_M berechnet sich aus der Summe der Kreuzprodukte der (*unstandardisierten*) Werte von zwei Matrizen, wobei die Diagonale (mit trivialen Ergebnissen 0 oder 1) ausgeschlossen wird. Andere Teststatistiken sind einfach Korrelationskoeffizienten (meist Pearson) bzw. die Steigung der Regression zwischen den Matrizen. Die Summe der Kreuzprodukte zweier Ähnlichkeitsmatrizen mit *standardisierten* Matrizen geteilt durch die Anzahl der Elemente der halben (dreieckigen) Ähnlichkeitsmatrix minus 1 (also $n(n-1)/2-1$) liefert den r_M-Koeffizienten. Er entspricht (ebenso wie der Regressionkoeffizient über standardisierte Matrizen) dem Pearsons-Korrelationskoeffizienten.

Der Test auf Signifikanz muss mit Hilfe eines Permutationsverfahren durchgeführt werden, da zum einen die Zahl der Freiheitsgrade *nicht* $n(n-1)/2-1$ entspricht, da nur n Element zu Grunde liegen und zum anderen die Distanzmaße nicht unabhängig voneinander sind. So ist beim Vergleich von Aufnahme Nr. 1 & 2 mit Nr. 1 & 3 die Hälfte der zu Grunde liegenden Elemente identisch.

Typische Fragen hierfür sind z. B. „Sind die Art-Daten mit den Umweltdaten korreliert?“ oder „Spiegelt sich die geographische Distanz in der Zusammensetzung der Arten wieder?“.

Als Beispiel rechnen wir wieder mit den bekannten Datensätzen der finnischen Flechtenrasen von Oksanen:

```
> data(varespec)
> data(varechem)
> veg.dist <- vegdist(varespec)
> env.dist <- vegdist(scale(varechem), "euclid")
> res <- mantel(veg.dist, env.dist)
> res

Call:
mantel(xdis = veg.dist, ydis = env.dist)

Mantel statistic based on Pearson's product-moment correlation

Mantel statistic r: 0.305
      Significance: <0.001

Empirical upper confidence limits of r:
  90%   95% 97.5%   99%
0.116 0.145 0.170 0.229

Based on 1000 permutations
```

Wir sehen also, dass der Korrelationskoeffizient der beiden Matrizen 0.3047 ist und bei 1000 Permutationen keine Permutation gleich groß oder größer waren als die zu testende Korrelation. Das durch die Permutation nachgewiesene Vertrauensintervall ist weitaus geringer als der gefundene Korrelationskoeffizient. Auch dies spricht für die „Signifikanz“ des Ergebnisses.

Der **Prokrustes-Test**: In der griechischen Mythologie hat der Gastwirt Prokrustes Gästen, die zu klein für seine Betten waren, die Beine lang gezogen und zu großen Gästen wurden die Beine abgeschnitten, so dass sie in die Betten passten. Ähnlich verfährt auch der Prokrustes-Test. Er vergleicht keine Distanz- oder Ähnlichkeitsmatrizen, sondern zwei rechteckige Rohmatrizen. Haben die beiden Matrizen ungleich viele Variablen, wird die kleine Matrize durch Erweiterung mit Null-Werten auf die Größe der Größeren gebracht. Durch Rotation werden die zu vergleichenden Matrizen so gefitted, dass die Summe der quadratischen Abweichungen zwischen korrespondierenden Punkten der Matrizen minimiert wird. Es handelt sich also streng genommen um eine Ordinationstechnik. Während die zugehörige Teststatistik normalerweise asymmetrisch ist ($m_{12}^2 \neq m_{21}^2$), kann man durch Standardisierung der Matrizen die Teststatistik symmetrisch machen ($m_{12}^2 = m_{21}^2$). Über den Prokrustes Randomisations-Test kann die Signifikanz abgeschätzt werden. Diese Methoden werden meist genutzt, um Ordinationsergebnisse zu vergleichen.

Im folgenden Beispiel werden die Ergebnisse der Ordination von Aufnahmeflächen mit RDA und CCA der altbekannten Flechtenrasen verglichen.

```
> var.rda <- rda(varespec, varechem)
> var.cca <- cca(varespec, varechem)
> proc <- procrustes(scores(var.rda, display = "sites"), scores(var.cca,
+     display = "sites"))
> summary(proc)

Call:
procrustes(X = scores(var.rda, display = "sites"), Y = scores(var.cca,
+     display = "sites"))

Number of objects: 24    Number of dimensions: 2

Procrustes sum of squares:
 19.96
Procrustes root mean squared error:
```

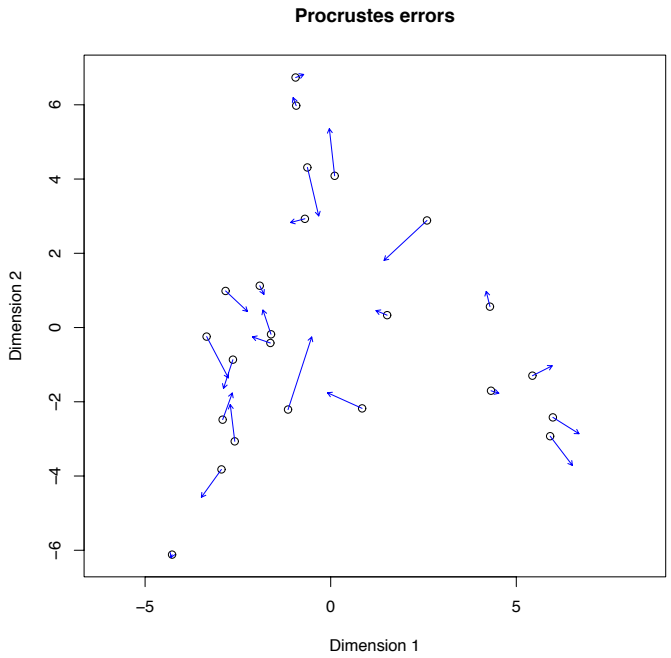


Abbildung 12.6.: Visualisierung des Procrustes-Tests. X-Werte stellen den Ausgangspunkt der Pfeile dar, Y-Werte das Ziel. Die beiden Koordinatensysteme sind ebenfalls basiert auf der Ordination der X-Werte (gestrichelt) bzw. der Y-Werte (durchgezogene Linien).

```
 0.9121
Quantiles of Procrustes errors:
   Min     1Q Median     3Q    Max
 0.086  0.381  0.785  1.005  2.058
```

Insgesamt beträgt die geringste zu ermittelnde Quadratsummenabweichung 19.96. Dies sagt uns allerdings nicht, ob das Ergebnis tatsächlich signifikant ist. Hierzu müssen wir einen Permutationstest (`protest`) durchführen.

```
> protest(scores(var.rda, display = "sites"), scores(var.cca, display =
+                         "sites"))

Call:
protest(X = scores(var.rda, display = "sites"), Y = scores(var.cca,
+     display = "sites"))

Correlation in a symmetric Procrustes rotation:  0.979
Significance:  <0.001
Based on 1000 permutations.
```

Und wie wir sehen, sind die Ergebnisse der beiden unterschiedlichen Ordinationsmethoden hochgradig korreliert (0.979) mit einer Irrtumswahrscheinlichkeit von $p<0.001$.

Das Ergebnis der Prokrustes-Rotation lässt sich auch visualisieren (Abb. 12.6):

```
> plot(proc)
```

Die Pfeile zeigen an, wie sich die entsprechenden Punkte von Matrix 1 (RDA) zu Matrix 2 (CCA) verschieben. Insgesamt sehen wir, dass die Pfeile nur sehr kurz sind (also sind sich die Ergebnisse ähnlich). Allerdings ist keine klare Richtung zu sehen (es handelt sich also nicht um eine bloße „Verschiebung“).

13. Das Design von Experimenten

13.1. Grundzüge des Designs von Experimenten

Die folgende Aufstellung ist ein kurzer Abriss des gesunden Menschenverstandes beim experimentellen Design. Wir wählen diese wenig prosaische Darstellung, damit die Chance, dass diese Merksätze im Gedächtnis bleiben, möglichst hoch ist.

i Ziel, Hypothesen und die Sinnhaftigkeit des Ansatzes müssen klar sein, bevor man sich Gedanken über die konkrete Ausarbeitung des Versuchsdesigns macht.

ii Für ein vernünftiges Design sollte man über die Art der zu erhebenden Daten und ihre Analyse nachgedacht haben.

iii Experimentelle Einheiten auf Ähnlichkeit auszuwählen ist nur dann erlaubt, wenn nachher die Behandlungen zufällig diesen Einheiten zugewiesen werden. Bei deskriptiven Studien (z.B. vegetationskundliche Erhebungen) ist sonst die Annahme der Unabhängigkeit der Versuchseinheiten verletzt.

1 Die Bible des experimentellen Designs ist Hurlbert (1984) (siehe Tabelle 13.1). Dieser Artikel ist ein MUSS!

2 Daten werden entweder deskriptiv oder experimentell gewonnen. In jedem Fall sind Vorstudien (Pilotexperimente) sinnvoll, um Methodik und Relevanz abzuschätzen, bevor man ein gigantisches Experiment aufbaut.

3 Die drei zentralen Konzepte beim experimentelle Design sind **Unabhängigkeit**, **Durchmischung** und **Replikation**.

4 Randomisierung ist die häufigste und einfachste Art und Weise Unabhängigkeit zu erreichen. Trotzdem muss man das Ergebnis der Randomisierung immer überprüfen. Sechs Würfelwürfe können zu der Sequenz 6 5 4 3 2 1 führen, was zwar zufällig, aber

Tabelle 13.1.: Mögliche Ursachen für Verwirrung in einem Experiment, und wie man sie minimiert (nach Hurlbert 1984).

	Verwirrungsursache	Merkmal des Designs, die diese Verwirrung reduzieren
1.	zeitliche Veränderung	Kontrollbehandlung
2.	Behandlungsartefakte	Kontrollbehandlung
3.	Voreingenommenheit des Forschers	zufällige Zuweisung der Behandlung zu den Versuchseinheiten Randomisierung bei allen möglichen Prozessen „Blindprozeduren“ (für sehr subjektive Messungen)
4.	Forscher-induzierte Variabilität (zufälliger Fehler)	Replikation
5.	Initiale oder inhärente Variabilität zwischen Versuchseinheiten	Replikation; Durchmischen der Einheiten; Begleitbeobachtungen
6.	Nichtdämonische Einflüsse (Zufall)	Replikation; Durchmischung
7.	Dämonische Eingriffe	Ewige Wachsamkeit, Geisteraustreibung, Menschenopfer

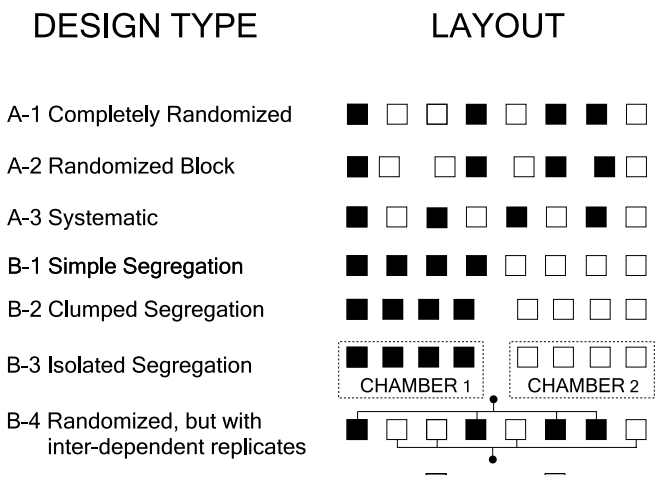


Abbildung 13.1.: Schema akzeptabler Designs (A), wie man Replikate (Kästchen) zweier Behandlungen (weiss und grau) durchmischt, und verschiedene Designs, bei denen das Durchmischungsprinzip verletzt wurde (B). (Nach Hurlbert 1984.)

in einem experimentellen Design **nicht akzeptierbar** ist: so ein Gradient kann jede Datenaufnahme ruinieren. Die Lösung heist nochmaliges Randomisieren.

5 Ob man die Versuchseinheiten systematisch oder randomisiert anordnet ist fallabhängig. Kein System ist *per se* besser, auch wenn Sokal and Rohlf (1995) das behaupten (siehe Abb. 13.1).

6 Eine Kontrolle ist unerlässlich.

7 Eine Kenntnis des Zustandes der Versuchseinheit *vor* dem Versuch erhöht die statistische Teststärke.

8 Ein Erfassen relevanter Umweltparameter (Kovariaten) kann die Teststärke ebenfalls dramatisch verbessern.

9 Die Kontrolle soll nur dem Behandlungseffekt gegenüberstehen. Wenn man also z.B. den Effekt von Steinen im Boden auf das Pflanzenwachstum erforschen will, und dafür die Steine aus dem Boden aussiebt, während die Kontrolle die Steine behält, so muss die Kontrolle doch genauso gesiebt werden, nur dass die Steine wieder zurückkommen. Sonst hätte die Bodenlockerung und -durchlüftung möglicherweise positive Effekte, die aber nichts mit den Steinen zu tun hätten.

10 Das Nämliche gilt für Behandlungen an sich, die, wenn irgend möglich, mit einer Behandlungsartefaktkontrolle verglichen werden sollte. Wenn man z.B. Motten durch kleine Stoffsäckchen von Blättern fernhält, so reduziert man gleichzeitig die Windgeschwindigkeit an der Blattoberfläche. Eine Behandlungsartefaktkontrolle wäre hier ein offener Stoffsack, der die Motten ans Blatt lässt, aber den Wind nicht. Das gibt dann eine Abschätzung über den Windeffekt, der hier mit dem Mottenfraß verquickt ist.

A Von allen statistischen Erwägungen abgesehen sollte ökologisches Verständnis das Design leiten. So hängt z.B. der sinnvolle Mindestabstand zweier Versuchsflächen von den Zielorganismen und Behandlungen ab. Klonale Pflanzen transportieren Nährstoffe über mehrere Meter, Bestäuber interferieren mit plots Dutzende Meter entfernt. Als Daumenregel sollte der Abstand zwischen zwei Flächen das 2-3-fache der Plotgröße betragen (also mindestens 2 m zwischen plots von 1×1 m).

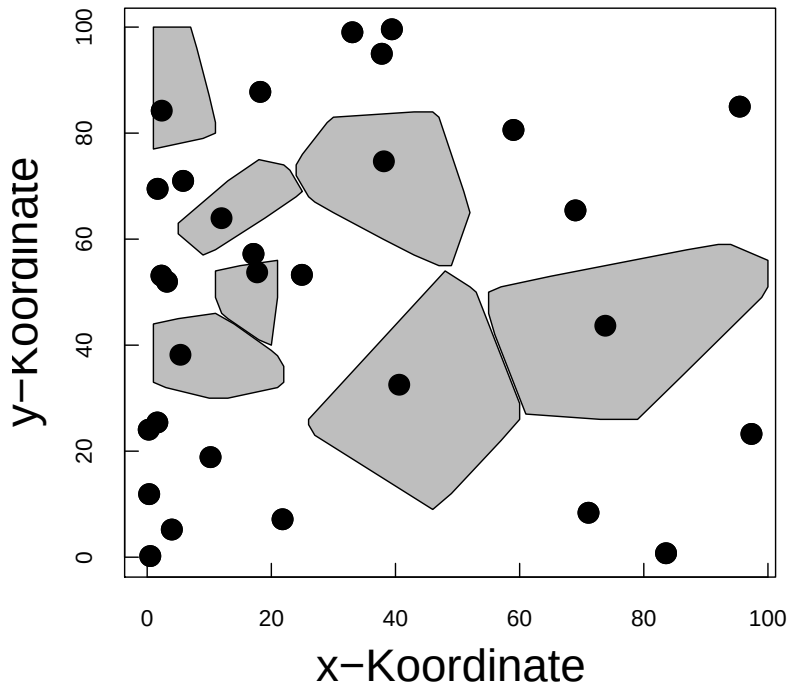


Abbildung 13.2.: Wenn wir x und y-Koordinaten zufällig wählen, und dann den nächsten Baum als Untersuchungsobjekt benutzen, so ist dies nicht zufällig. Auf Bäume, die weiter auseinanderstehen, beziehen sich mehr (x,y)-Paare (graue Fläche um Baumpunkt), als auf solche die enger stehen. (Verändert nach Crawley 1993.)

B Finanzielle, räumliche, logistische oder zeitliche Beschränkungen werden immer das „ideale“ Design zu Kompromissen zwingen. Erwähne diese Beschränkungen und ihren möglichen Effekt auf die Ergebnisse beim Zusammenschreiben des Versuchs.

Mit Ausnahme der klassischen Test sind viele statistischen Verfahren selbst bei Fachleuten Gegenstand der Diskussion (Quinn and Keough 2002). D.h. es gibt keine “wahre” oder “beste” Analyse. Sei ehrlich und benutze Deinen gesunden Menschenverstand. Trage den Annahmen der unterschiedlichen Tests Rechnung, und welche Du davon ohne großen Schaden verletzen kannst (z.B. Normalverteilung), und welche absolut essentiell sind (z.B. Varianzhomogenität). Frage schamlos Profis oder Kollegen um Hilfe: sie haben genau die gleichen Probleme!

13.2. Randomisierung im Feld

Im Feld lassen einen die obigen Erwägungen bisweilen schnell im Stich. Wer 200 zufällige Pflanzen ausgraben soll, der wird kaum 10000 Pflanzen mit kleinen Nummern versehen und dann eine Zufallszahlentabelle bemühen. Stattdessen wird ein Taschenmesser über die Schulter geworfen oder alle 5 Minuten bei einem Spaziergang der Kescher geschwenkt. Natürlich sind dies nicht zufällige Stichproben. Sie nehmen an, dass unsere Willkür unvoreingenommen ist, und somit die gleiche Zufälligkeit besitzt, wie eine wirklich zufällige Stichprobe. Zumeist wird dies in Ordnung sein, aber wir müssen uns der Problematik immer bewusst sein!

Ein beinah klassisches Beispiel, wo diese Willkür schief gehen kann ist in Abbildung 13.2 dargestellt. Wenn die Abnahme der Baumdichte einen ökologischen Grund hat, so werden wir

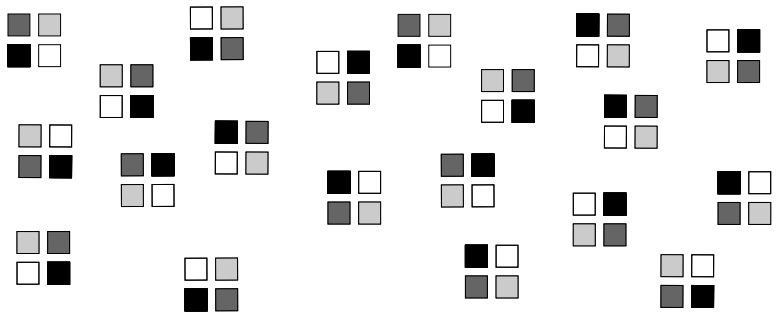


Abbildung 13.3.: Ein 20-fach repliziertes, randomisiertes Blockdesign.

mit der beschriebenen Art der Baumauswahl möglicherweise diesem Gradienten aufsitzen, und unsere Daten werden von ihm überlagert, so dass die zu untersuchenden Effekte verschwinden.

13.3. Häufige Experimentelle Designs

13.3.1. Vollständig randomisiertes Blockdesign (fully randomised block design)

1. **Definition** Die Behandlungsflächen aller Behandlungskombinationen werden in einem Block zusammengefasst. Dieser Block wird dann repliziert. Die Behandlungen werden den Flächen innerhalb des Blocks zufällig zugewiesen.

2. **Beispiel** Wir wollen den Effekt von Nematoden auf Konkurrenz untersuchen. Untersucht werden soll, ob die Konkurrenz von *Festuca* auf *Artemisia* abhängig ist von Nematodenfraß. Wir haben dabei folgende Behandlungen: *Festuca rubra* in Konkurrenz mit *Artemisia maritima* (F+); *Artemisia maritima* in Monokultur (F−); unbehandelter Boden mit Nematoden (N+); Nematicide-behandleter Boden ohne Nematoden (N−). Damit erhält man in einem faktoriellen Experiment vier Behandlungskombinationen (F+N+, F+N−, F−N+, F−N−), die in einem Block zusammengefasst, der, sagen wir, 20-fach repliziert wird. Damit sieht in einem Vollständig Randomisierten Blockdesign das Ergebnis etwa so aus wie Abbildung 13.3.

3. **Details** Dieses Design ist das häufigste, wichtigste und intellektuell befriedigenste Versuchsdesign. Es hat zwei Kernelemente: (1) Randomisiere was immer Du kannst. Und (2) Tue alle Behandlungskombinationen in einen Block, der dann repliziert wird. Die statistische Analyse ist unkompliziert, da keine Abhängigkeiten der Behandlungen oder Blöcke vorliegen. Die gängige Auswertungsmethode ist (bei normalverteilten Daten) die ANOVA. Da wir nicht wirklich am Unterschied zwischen Blöcken interessiert sind, wird die Blocknummer als Zufallsvariable mit ins Modell hereingenommen.

4. **Stärken & Schwächen** Das Randomisieren kann recht viel Zeit in Anspruch nehmen (obwohl wir dies auch schon im Vorraus und im Sessel machen können). Wenn wir um die Existenz eines Gradienten im Untersuchungsgebieten wissen, der unser Experiment beeinflussen kann, so müssten wir dies sowohl bei der räumlichen Verteilung der Blöcke als auch der der Behandlungsflächen innerhalb der Blöcke berücksichtigen. Dies ist bei einem randomisierten Design natürlich nicht möglich, sondern Einflüsse dieser Art müssen über eine Erhöhung der Stichprobenzahl kompensiert werden. D.h. wir müssen mehr Replikate anlegen, als ohne diesen Gradienten notwendig wäre.

5. **Literatur** Crawley (2002); Hurlbert (1984); Mead (1988); Potvin (2001); Underwood (1997)

6. Rechenbeispiel Wir bleiben bei dem Beispiel mit den Nematoden und der Konkurrenz von *Festuca* auf *Artemisia*. Unser Datensatz ist etwas löchriger, da für die Monokulturen nur 11 Replikate benutzt wurden und für die Konkurrenz 16. Zudem sind die Pflanzen in zwei Töpfen gestorben. Aber dies nur am Rande. Zunächst lesen wir die Daten ein, kodieren die Faktoren als Faktoren und lassen uns die Mittelwerte für die Behandlungskombinationen ausgeben. Dann rechnen wir eine ANOVA, in der wir dem Modell sagen, dass wir geblockt haben.

```
> nema <- read.table("nematode.txt", header = T)
> nema$comp <- as.factor(nema$comp)
> nema$block <- as.factor(nema$block)
> nema$nematodes <- as.factor(nema$nematodes)
> attach(nema)
> names(nema)

[1] "pot"       "comp"      "nematodes" "block"     "Artemisia"

> tapply(Artemisia, list(comp, nematodes), mean, na.rm = T)

          0         1
0 15.216455 14.339500
1  2.069063  2.116063

> fm <- aov(Artemisia ~ nematodes * comp + Error(block))
> summary(fm)

Error: block
          Df  Sum Sq Mean Sq F value Pr(>F)
nematodes  1   0.018   0.018  0.0017 0.9678
comp       1  25.197  25.197  2.4368 0.1425
Residuals 13 134.422  10.340

Error: Within
               Df  Sum Sq Mean Sq  F value  Pr(>F)
nematodes       1   20.74   20.74   3.2313 0.08169 .
comp            1 1896.81 1896.81 295.5305 < 2e-16 ***
nematodes:comp  1    2.08    2.08   0.3236 0.57344
Residuals      32  205.39    6.42
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Durch den Aufruf des Modells (`fm`) erhalten wir eine wichtige Zusatzinformation: Im Stratum 1 ist einer der 3 Effekte nicht berechenbar. Dies ist die Interaktion der beiden Behandlungen. Sie fehlt in obigem *output*. Der Grund für ihr Fehlen liegt darin, dass das Design unausgewogen (*unbalanced*) ist und somit nicht für alle Blocks genug Daten zur Berechnung des Interaktionseffekt bereitstehen (nur Blocks 2, 3, 4, 6 und 8: `table(block)`).
Der obere Teil des outputs bezieht sich auf das obere Stratum: die Blöcke. Innerhalb jedes Blocks wird nur ein Wert für die jeweilige Behandlung berechnet, also durch Mitteln über die andere Behandlung. Damit stehen insgesamt 16 Werte je Behandlung zur Verfügung (einer je Block). Die Aussage der nicht-signifikanten Behandlungen hier ist also: Wenn wir nur die Mittelwerte je Block vergleichen, so sehen wir keinen Unterschied zwischen den Behandlungen. Im zweiten Stratum hingegen werden alle Daten benutzt, entsprechend haben wir mehr Freiheitsgrade für die Residuen. Beachte, dass die Freiheitsgrade der Residuen aus dem 1. Stratum von denen im 2. Stratum abgezogen werden. Die Aussage hier ist dann: Wenn wir den Blockeffekt aus jedem einzelnen Messwert herausrechnen, dann finden wir einen signifikanten Effekt der Konkurrenz, und einen ganz leichten Nematodeneffekt.
Der *output* für das erste Stratum beinhaltet nur dann die Behandlungen selbst, wenn das Design unausgewogen ist, d.h. Behandlungskombinationen in einem oder mehreren Blöcken fehlen. Grund ist, dass natürlich bei fehlenden Kombinationen der Blockeffekt nicht sauber geschätzt werden kann. Fehlt beispielsweise ein Konkurrenztopf, so liegt der Mittelwert ja höher (konkurrenzfreie *Artemisia*

sind schwerer). Ist das Design ausgewogen (*balanced*), so kann der Blockeffekt unabhängig von den Behandlungseffekten berechnet und vom 2. Stratum abgezogen werden.

7. Multivariates Rechenbeispiel Post (1986) hat ein Experiment durchgeführt, um u. a. den Effekt des Pflüge-Zeitpunktes auf die Ackerkraut-Vegetation eines Gerstenackers zu untersuchen. Die drei Zeitpunkte im Frühjahr 1983 sind mit 1-3 angegeben und wurden randomisiert auf vier Blöcke aufgeteilt (vgl. Abbildung):

I	II	III	IV
1	3	2	3
2	1	3	2
3	2	1	1

Dieses Design ist in unserem Beispiel in `plough.env` gespeichert. Die Erhebungen ergaben folgende Deckungen der Ackerkäuter (`plough.spe`):

	Chealb	Polper	Polavi	Spearv	Matrec	Gnauli	Capbur	Stemed	Solnig	Viorvi	Rorpal	Galpar	Apespi
SAMPLE01	147.4	9.5	4.5	118.5	153.3	2.5	5.0	29.0	5.5	9.5	150.3	1.5	4.0
SAMPLE03	88.5	9.5	9.0	61.0	0.0	56.8	8.5	15.9	15.0	15.9	186.5	0.5	1.5
SAMPLE02	38.0	8.5	2.5	67.5	184.9	106.0	7.5	20.0	38.4	0.5	138.5	3.0	0.0
SAMPLE06	119.5	5.0	1.0	15.9	20.0	119.5	6.5	16.5	37.5	3.0	90.3	3.0	0.0
SAMPLE04	290.9	12.5	1.0	55.5	17.9	54.3	3.5	25.0	20.5	1.5	106.0	17.5	0.0
SAMPLE05	43.5	5.0	2.5	29.0	20.9	84.5	12.5	24.3	29.3	2.5	50.0	16.5	0.0
SAMPLE08	32.5	7.5	9.5	132.0	67.5	95.0	28.4	0.0	25.9	8.0	120.0	11.5	0.0
SAMPLE07	127.0	26.8	4.5	194.0	147.9	25.5	1.5	28.4	3.0	0.5	126.0	8.0	1.0
SAMPLE09	38.0	22.5	4.0	130.3	82.8	228.0	2.5	21.5	30.0	5.0	306.4	13.4	0.0
SAMPLE12	122.9	25.5	12.8	125.0	384.4	199.5	1.5	38.4	30.9	8.0	481.4	77.5	2.5
SAMPLE11	88.5	50.5	4.5	45.0	342.5	153.0	2.0	39.0	25.0	5.5	259.9	17.0	0.5
SAMPLE10	180.8	125.0	9.5	288.4	350.9	58.0	4.0	53.5	20.5	4.5	539.3	23.4	1.0

Die Frage ist, ob der Zeitpunkt des Pflügens (*treatment*) einen Einfluss auf die Deckung der Krautvegetation hat.

Zur Beantwortung wird eine multivariate Methode genutzt, da es mehrere Abhängige (Krautarten) gibt. Theoretisch lässt sich die Frage auch beantworten, indem für jede Krautart einzeln ein univariater Test genutzt wird. Dann ergibt sich jedoch das Problem der „multiplen Tests“mit jeweils einer Irrtumswahrscheinlichkeit auf der 5%-Ebene. Bei z. B. 20 Arten kann also das Ergebnis auf Grund des Typ I Fehlers (falsche Verwerfung der Null-Hypothese) bei einer Art „falsch signifikant“ sein. Dies kann durch geeignete Korrekturen überwunden werden, allerdings sind diese Verfahren (z. B. „Bonferroni-Korrektur“) schwach. Eine Möglichkeit zur Analyse einer derartigen Fragestellung wäre eine multivariate Varianzanalyse (MANOVA). Da aber die Anzahl der Arten (13) größer ist als die Anzahl der Probeflächen (12), kann eine MANOVA nicht durchgeführt werden.

Wenn uns darüber hinaus die Reaktion der gesamten Gemeinschaft interessiert, ist es auch nicht unbedingt notwendig, jede Art einzeln zu betrachten. Wenn wir diejenigen Arten im multivariaten Test betrachten, die die stärkste Reaktion zeigen, dürften dies in der Regel auch diejenigen Arten sein, die bei univariaten Tests signifikante Ergebnisse erhalten. Univariate Test würden uns aber keine weiteren Erkenntnisse bringen.

Vortest: Allen multivariaten Ordinationsmethoden ist gemeinsam, dass durch unterschiedliche mathematische Methoden ein hypothetischer, zu Grunde liegender Gradient (1. Achse) gesucht wird, der den höchsten Anteil an Varianz erklären kann. Senkrecht dazu wird ein zweiter Gradient (2. Achse) gesucht, der den höchsten Anteil an Restvarianz erklären kann usw.

Multivariate Tests, genauer gesagt Ordinationsverfahren, lassen sich traditionell in zwei Gruppen einteilen: 1. Verfahren, bei denen ein linearer Gradient zu Grunde liegt und 2. Verfahren, bei denen ein unimodaler Gradient zu Grunde liegt. Dies kann mit einer DCA (*Detrended Correspondence Analysis*) getestet werden.

Vor den weiteren Berechnungen werden die Häufigkeiten der Arten logarithmiert, um den Einfluss der hohen Häufigkeiten zu verringern.

Eine DCA in R mit Hilfe des Paketes **vegan** liefert folgendes Ergebnis:

```
> library(vegan)
> plough.spe <- read.csv("plough.spe.csv", head = T, sep = " ")
```

```
> plough.env <- read.csv("plough.env.csv", head = T, sep = " ")
> plough.env$Block <- as.factor(plough.env$Block)
> plough.env$Treatment <- as.factor(plough.env$Treatment)
> decorana(log(plough.spe + 1))

Call:
decorana(veg = log(plough.spe + 1))

Detrended correspondence analysis with 26 segments.
Rescaling of axes with 4 iterations.

                   DCA1    DCA2     DCA3     DCA4
Eigenvalues     0.02695 0.02382 0.010776 0.012649
Decorana values 0.02968 0.01546 0.004736 0.002045
Axis lengths    0.54792 0.57198 0.436047 0.432539
```

Das für uns wichtigste Ergebnis in diesem Zusammenhang ist die Länge der 1. DCA-Achse. Diese beträgt 0.548. Auf die anderen Werte soll bei vergleichbaren Verfahren weiter unten eingegangen werden.

Ist die Länge der ersten Achse einer DCA größer als 4, kann man von einer unimodalen Reaktion auf den zu Grunde liegenden Gradienten ausgehen; ist die Länge der ersten Achse kleiner 2, ist die Reaktion linear. Dazwischen sind sowohl lineare als auch unimodale Modelle gleich (un)geeignet. Allerdings weisen ter Braak and Smilauer (1998) darauf hin, dass Korrespondenz-Analysen je nach Kontext sowohl ein unimodales als auch ein lineares Gesicht haben können. Dabei sind häufig Methoden, die auf Euklidischen Distanzen beruhen (lineare Modelle PCA und RDA) schwächer als solche, die auf Chi-Quadrat-Distanzen beruhen (unimodale Modelle wie CA und CCA).

Das entsprechende lineare Modell zur Untersuchung von Art-Matrizen in Abhängigkeit von Umweltvariablen ist die Redundanz-Analyse (*redundancy analysis*, RDA; Redundanz ist Synonym zu erklärbarer Varianz). Hier bei handelt es sich (stark vereinfacht), um eine Hauptkomponenten-Analyse mit zu Grunde liegenden multiplen Regressionen.

Haupt-Analyse: Zum Berechnen der entsprechenden RDA werden die Artdaten mit den Behandlungen (Zeitpunkte des Pflügens) in Abhängigkeit gebracht unter Beachtung der Blöcke. Dazu müssen die Blöcke als Kovariablen (*conditional variables*) in das Model gegeben werden. Dies führt dazu, das der Einfluss der Blöcke als bekannt voraus gesetzt wird. Somit wird nur untersucht, welchen Einfluss die unterschiedlichen Behandlungen haben, wenn der Block-Effekt heraus gerechnet wird. Die Behandlung besteht aus drei Stufen. Da sich die dritte Behandlungsstufe automatisch aus den beiden ersten ergibt, sind auch nur die ersten beiden Achsen durch die Umweltvariablen eingeschränkt (*constrained*). Die nachfolgenden sind nicht eingeschränkt und entsprechen damit einer herkömmlichen Hauptkomponenten-Analyse (*principal component analysis*, PCA).

Das Ergebnis einer RDA in R sieht folgendermaßen aus:

```
> plough.rda <- rda(log(plough.spe + 1) ~ Treatment + Condition(Block),
+     data = plough.env)
> plough.rda

Call:
rda(formula = log(plough.spe + 1) ~ Treatment + Condition(Block),
+     data = plough.env)

              Inertia Rank
Total          11.315
Conditional     5.213    3
Constrained     2.529    2
Unconstrained   3.574    6
Inertia is variance
```

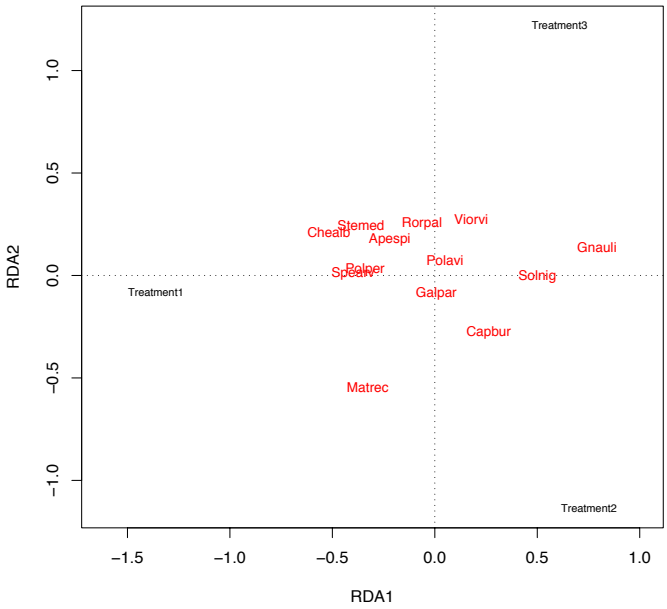


Abbildung 13.4.: Ergebnis der RDA als Ordinationsplot.

```
Eigenvalues for constrained axes:
  RDA1   RDA2
1.8360 0.6926

Eigenvalues for unconstrained axes:
   PC1    PC2    PC3    PC4    PC5    PC6
1.6459 0.7674 0.5594 0.3276 0.1645 0.1089
```

Die *inertia* entspricht hier der Varianz einer Matrix und entspricht den X^2-Werten bzw. errechnet sich aus der Summe aller Eigenwerte. Die einzelnen Eigenwerte geben an, wie hoch der Anteil an der ursprünglichen Varianz ist, die durch eine Achse erklärt wird.

Die Gesamtvariabilität (inkl. Kovariablen) liegt bei 11.3, die der vier Kovariablen (vier Blöcke, von denen nur drei gezählt werden, da sich dann der vierte automatisch ergibt) bei 5.213. Die durch die Umweltvariablen erklärte Varianz beträgt 2.5 und die restliche Varianz 3.57. Die erklärbare Varianz ist der Anteil einer RDA-Achse an der Varianz, die nicht durch die Kovariablen erklärt wird ($11.315 - 5.213 = 6.103$), also kann durch die 1. RDA-Achse (`treatment 1`) ein Anteil von 1.836/6.103=0.3 erklärt werden. Der Anteil an Varianz, der nicht durch Umweltparameter zu erklären ist (3. Achse der Ordination = 1. Achse der PCA) beträgt $1.646/6.103 = 0.27$. Unser Modell kann also auf der 1. Achse 30% und auf den ersten beiden Achsen 41.4% der Varianz erklären. Dies ist für multivariate ökologische Daten sehr gut. Der Zeitpunkt des Pflügens scheint also einen gewissen Einfluss auf die Häufigkeiten der Ackerkräuter zu haben. Dies lässt sich auch grafisch darstellen (siehe Abbildung 13.4):

```
> plough.rda, display = "lc")
> spe <- scores(plough.rda)$species
> text(spe, rownames(spe), col = "red", cex = 0.8)
```

Während z. B. *Spergularia arvensis* und *Chenopodium album* durch frühes Pflügen (`treatment 1`) gefördert werden, ist *Capsella bursa-pastoris* am häufigsten bei etwas späterem Pflügen (`treatment 2`).

Inwieweit das Gesamtergebnis der Ordination signifikant ist, kann (anders als bei einer normalen Varianzanalyse) nicht direkt abgeschätzt werden, sondern muss durch einen Permutationstest überprüft

werden. Dabei wird als Test-Statistik ein pseudo-F-Wert zu Grunde gelegt. Dieser berechnet sich aus dem Verhältnis von eingeschränkter zu nicht-eingeschränkter Inertia (als Chi-Quadrate/Anzahl Variablen); die Chi-Quadrate sind hier also anolog zu Quadratsummen einer Varianzanalyse zu behandeln. Bei der Permutation werden die Residuen der eingeschränkten CCA gemischt und die entsprechenden pseudo-F-Werte errechnet. Aus der Anzahl der Pseudo-F-Werte aus der Permutation, die größer als das Pseudo-F sind, wird eine Irrtumswahrscheinlichkeit abgeleitet. In unserem Beispiel ist es wichtig, dass die Permutationen innerhalb der Blöcke durchgeführt werden. Denn uns interessiert nicht die Irrtumwahrscheinlichkeit der Behandlung an sich, sondern der Behandlung im Block.

Das Ergebnis nach 999 Permutationen in R sieht folgendermaßen aus:

```
> attach(plough.env)
> anova(plough.rda, strata = Block, step = 999)

Permutation test for rda under direct model
Permutations stratified within `Block'

Model: rda(formula = log(plough.spe + 1) ~ Treatment + Condition(Block),
+                         data = plough.env)
         Df    Var      F N.Perm  Pr(>F)
Model     2   2.53 2.1226 999.00 0.02903 *
Residual  6   3.57
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In unserem Beispiel errechnet sich Pseudo-F also als (2.53/2)/(3.57/6)=2.12; bei den 999 Permutation waren nur 22 Pseudo-F-Werte größer als 2.12. Der Zeitpunkt des Pflügens (innerhalb eines Blockes) hat also einen signifikanten Einfluss auf die Zusammensetzung der Gesellschaft (genauer auf die Häufigkeit der Ackerkräuter) und kann 41.44% der Varianz erklären (s. o.).

Weitere Exploration: Da das Ergebnis signifikant ist, können wir uns die ausführliche Ausgabe in R anschauen:

```
> summary(plough.rda)

Call:
rda(formula = log(plough.spe + 1) ~ Treatment + Condition(Block),
+     data = plough.env)

Partitioning of variance:

Total           11.315
Conditioned out  5.213
Constrained      2.529
Unconstrained    3.574

Eigenvalues, and their contribution to the variance
after removing the contribution of conditiniong variables

            RDA1   RDA2    PC1    PC2    PC3    PC4    PC5    PC6
lambda    1.8360 0.6926 1.6459 0.7674 0.5594 0.3276 0.1645 0.1089
accounted 0.3009 0.4144 0.2697 0.3955 0.4871 0.5408 0.5678 0.5856

Scaling 2 for species and site scores
-- Species are scaled proportional to eigenvalues
-- Sites are unscaled: weighted dispersion equal on all dimensions


Species scores
```

```
            RDA1      RDA2      PC1      PC2      PC3      PC4
Chealb -0.516786  0.212846 -0.06278  0.02149  0.05959  0.06272
Polper -0.340168  0.027264  0.02827  0.27839 -0.05575  0.05881
Polavi  0.049588  0.073881 -0.25831 -0.13227 -0.04758 -0.16755
Spearv -0.398656  0.009766 -0.02050 -0.06297 -0.22231 -0.15735
Matrec -0.327070 -0.544915  0.98840 -0.46152 -0.01362  0.10324
Gnauli  0.791441  0.139490  0.12101  0.32811 -0.36462  0.03469
Capbur  0.262828 -0.279547 -0.35254 -0.23891 -0.19651  0.08632
Stemed -0.359970  0.244514  0.43519  0.38773  0.24379 -0.20328
Solnig  0.500187 -0.002953  0.08121 -0.01151 -0.35185  0.09345
Viorvi  0.177679  0.273330 -0.45303 -0.28520  0.08710 -0.04787
Rorpal -0.062715  0.253345  0.03758  0.03353 -0.16194  0.01443
Galpar  0.007556 -0.088885  0.17011 -0.11122 -0.25607 -0.41169
Apespi -0.219788  0.173628 -0.05906 -0.18484  0.19715 -0.14645


Site scores (weighted sums of species scores)

            RDA1     RDA2      PC1       PC2     PC3      PC4
SAMPLE01 -1.8645 -0.92311  0.30913 -1.775877  1.0755 -0.08588
SAMPLE03  1.1283  3.33114 -2.00004  1.169402  0.1820 -0.30293
SAMPLE02  0.7362 -2.40804  1.69091  0.606475 -1.2575  0.38880
SAMPLE06  0.4588  0.21311  0.41464 -0.593298  0.5577  2.04375
SAMPLE04 -0.8925  0.48829 -0.24250  0.590098 -1.1687 -0.30331
SAMPLE05  0.4337 -0.70141 -0.17214  0.003199  0.6109 -1.74044
SAMPLE08  1.3663 -1.14688 -1.45142 -1.487648 -0.8460  0.54418
SAMPLE07 -1.8639  0.11886  0.46448  0.903577  1.2921 -0.25234
SAMPLE09  0.4976  1.02802  0.98694  0.584070 -0.4461 -0.29183
SAMPLE12  0.3501  0.30852  0.59846 -1.160174 -0.2937 -1.44899
SAMPLE11  0.4733 -0.29268 -0.06736  0.877973  1.4925  0.80746
SAMPLE10 -0.8234 -0.01584 -0.53110  0.282201 -1.1989  0.64152


Site constraints (linear combinations of constraining variables)

            RDA1     RDA2
SAMPLE01 -1.3611 -0.08295
SAMPLE03  0.6087  1.22020
SAMPLE02  0.7524 -1.13725
SAMPLE06  0.6087  1.22020
SAMPLE04 -1.3611 -0.08295
SAMPLE05  0.7524 -1.13725
SAMPLE08  0.7524 -1.13725
SAMPLE07 -1.3611 -0.08295
SAMPLE09  0.6087  1.22020
SAMPLE12  0.6087  1.22020
SAMPLE11  0.7524 -1.13725
SAMPLE10 -1.3611 -0.08295


Biplot scores for constraining variables

             RDA1    RDA2
Treatment2 0.5518 -0.8340
Treatment3 0.4464  0.8948


Centroids for factor constraints
```

```
              RDA1     RDA2
Treatment1 -1.3611 -0.08295
Treatment2  0.7524 -1.13725
Treatment3  0.6087  1.22020
```

Der obere Teil entspricht dem schon besprochenen Ergebnis. Die Eigenwerte entsprechen noch der einfachen Ausgabe, ihr Anteil an der Varianz wird wie oben angegeben berechet, allerdings kumulativ. Dabei wird das Ergebnis der eingeschränkten Analyse getrennt von der nicht-eingeschränkten Analyse betrachtet.

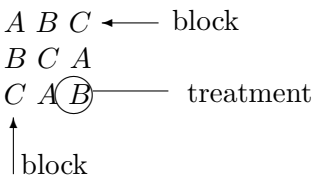
Die *Scores* geben die Koordinaten der jeweiligen Objekte in einem Euklidschen Koordinatensystem wieder. Sie sind bei den Arten entsprechend ihrer Eigenwerte (d. h. ihres Beitrages zur Varianzerklärung) skaliert, um eine optimierte Darstellung zu gewährleisten (die der von CANOCO entspricht). Die *Scores* der Aufnahmeflächen können auf zwei verschiedene Weisen berechnet werden: Erstens *linear constraints* (LC Scores) und zweitens *weighted averages* (WA Scores). LC Scores errechnen sich aus den zu Grunde liegenden multiplen Regressionen einer durch Umweltvariablen eingeschränkten Analyse (RDA, CCA). WA Scores sind die gewichteten Mittel der *Species Scores* und werden normalerweise in einer nicht-eingeschränkten Korrespondenz-Analyse (*Correspondence Analysis*, CA) errechnet. Palmer (1993) schlägt auf Grund von Simulationen LC Scores zur Nutzung in Ordinations-Diagrammen vor, um so auch den Vorteil einer CCA gegenüber einer CA ausnutzen zu können. Allerdings hat McCune et al. (2002) gezeigt, dass Umweltvariablen mit Hintergrundrauschen die LC Scores unbenutzbar machen, während die WA Scores kaum beeinflusst wurden. Letztgenannte Option ist Standard in R. Im Programm CANOCO werden Aufnahmeflächen mit ihren LC Sores dargestellt (dort SamE genannt), in PC-ORD sind es die WA Scores.

Die *Biplot Scores* geben das Ende eines kontinuierlichen Umweltvektors ("Pfeilspitzen") im Ordinationsdiagramm wieder, während Centroid Scores die Lagemittel von kategorialen Daten im Ordinationsraum sind. Biplot Scores werden auch für Kategoriale Daten errechnet, da der entsprechende Algorithmus die Kodierung nicht erkennt. In unserem Beispiel sind die Biplot Scores daher überflüssig. Über die unterschiedlichen Skalierungen, mit denen aus den jeweiligen Eigenwerten die Scores berechnet werden, gibt es noch keine einheitliche Auffassung. Diese kann von Programm zu Programm unterschiedlich sein und sich auch innerhalb eines Programms von Paket zu Paket unterscheiden (z. B. R).

Literatur zur multivariaten Statistik:
Legendre and Legendre (1998): DAS Standardwerk zu fast allen Fragen der multivariaten Statistik; z.T. recht ausführlich und mathematisch.
Jongman et al. (1995): Klassiker mit einer guten Übersicht über alle wichtigen multivariaten Verfahren; ter Braak ist Programmierer von CANOCO.
McCune et al. (2002): Gilt als neues Standardwerk; McCune ist Programmierer von PC-Ord.
Lepš and Smilauer (2003): Wurde aus dem Script zur *Summer School on Multivariate Analysis of Ecological Data* der Autoren entwickelt. Es ist leicht verständlich sein und enthält viele Beispiele; Smilaur ist Co-Programmierer von CANOCO 4.x.
Gauch (1982): Sehr einfache Einführung in einfache Ordinationsverfahren mit wenig mathematischem Hintergrund. Für den Anfang zu empfehlen, wird aber schnell unzureichend.
Digby and Kempton (1987): Einfache Einführung in einfache Ordinationsverfahren mit starker mathematischer Herleitung. Für den Anfang zu empfehlen, wird aber schnell unzureichend.

13.3.2. Latin Square Design

1. Definition Beim *latin square* Design erfolgt die Blockbildung in zwei Richtungen, statt in einer wie beim vollständig randomisierten Blockdesign. Dabei entsteht ein regelmäßiges quadratisches Behandlungsmuster, die *lattice*. Die Reihen und Spalten der lattice können dabei beispielsweise einem oder zwei ökologischen Gradienten folgen (etwa Hangneigung und Himmelsrichtung). Jede Behandlungsvariante findet sich aber in jeder Reihe oder Spalte nur genau einmal:



2. Beispiel Wir interessieren uns für die Unterschiede in Phytophagenbefall von vier Pflanzentypen. Dazu beproben wir diese 4 Arten in 4 Habitaten, und zwar einmal pro Jahreszeit. Damit sind `Jahreszeit` und `Fläche` die Gradienten entlang derer das *latin square* aufgespannt wird (Blockfaktoren). Der dritte interessante Faktor ist `Pflanzentyp`. Wir erhalten

Jahreszeit	Habitat			
	A	B	C	D
	B	A	D	C
	C	D	A	B
	D	C	B	A

3. Details Der Merksatz für LS-Designs ist: „Block, was Du kannst, randomisier, was Du nicht blocken kannst." Um die Vorteile des LSD nutzen zu können, brauchen wir Symmetrie: die Anzahl der Level jeder Behandlung muss gleich sein (Habitat = Jahreszeiten = Arten = 4). Wenn dies nicht der Fall ist, scheidet das LSD aus. Randomisierung kann durch das Vertauschen ganzer Reihen und Spalten erfolgen. LSD findet vor allem bei sogenannten *cross-over experiments* Anwendung. Dabei wird eine Behandlungseinheit mehrfach genutzt. Wir haben also etwa 6 Laborratten, denen 6 verschiedene Medikamente verabreicht werden, und jedes wird mit jedem Medikament behandelt, aber zu unterschiedlichen Zeiten. Dabei muss man annehmen können, dass es keine Übertragungseffekt (*carry over effect*) gibt, also ein Medikament *vor* einem anderen genauso wirkt wie *danach*. Eine Erweiterung ist das Graeco-Latin Square. Dabei werden zwei LSD übereinandergelegt. Dies klingt komplizierter als es ist: Ein Beispiel ist ein Kartenspiel, bei dem die Farben (Kreuz, Pik, Herz, Karo) mit vier Bildern (Bube, Dame, König, Ass) gekreuzt werden. Abbildung 13.5 zeigt ein *Graeco-Latin square* der 10. Ordnung.

4. Stärken & Schwächen Hauptproblem ist natürlich die Symmetrie. Für Versuche mit wenigen Behandlungsleveln ist das LSD aber ein hervorragendes Design, besonders entlang von Umweltgradienten. Ein anderer Nachteil ist, dass wir mit dem LSD nicht auf die Interaktion der beiden Faktoren testen können. Bei *cross-over experiments* muss man darauf achten, dass genug Zeit zwischen den Test an einer Behandlungseinheit liegt, so dass diese sich wieder erholen kann.

6. Beispielrechnung Wir bleiben bei dem obigen Beispiel des Blattfrasses an vier verschiedenen Pflanzentypen in vier Habitaten zu den vier Jahreszeiten. Die Antwortvariable `Frass` ist die Anzahl gefressener Blätter (was auch immer die ökologische Sinnhaftigkeit dieser Messung sein mag). Damit handelt es sich um Poisson-verteilte Daten und wir müssen unser GLM entsprechend instruieren. Da wir uns für alle Faktoren interessieren (also sowohl die blockbildenden Saison und Habitat, als auch für Art), haben wir keine Zufallseffekte. Da in diesem Fall das LSD nicht repliziert wurde, können wir auch keine Interaktionen zwischen den Effekten berechnen.

```
> LSD <- read.table("LSD.txt", header = T)
> summary(glm(Frass ~ Saison + Habitat + Art, family = poisson,
+     data = LSD))

Call:
glm(formula = Frass ~ Saison + Habitat + Art, family = poisson,
    data = LSD)

Deviance Residuals:
```

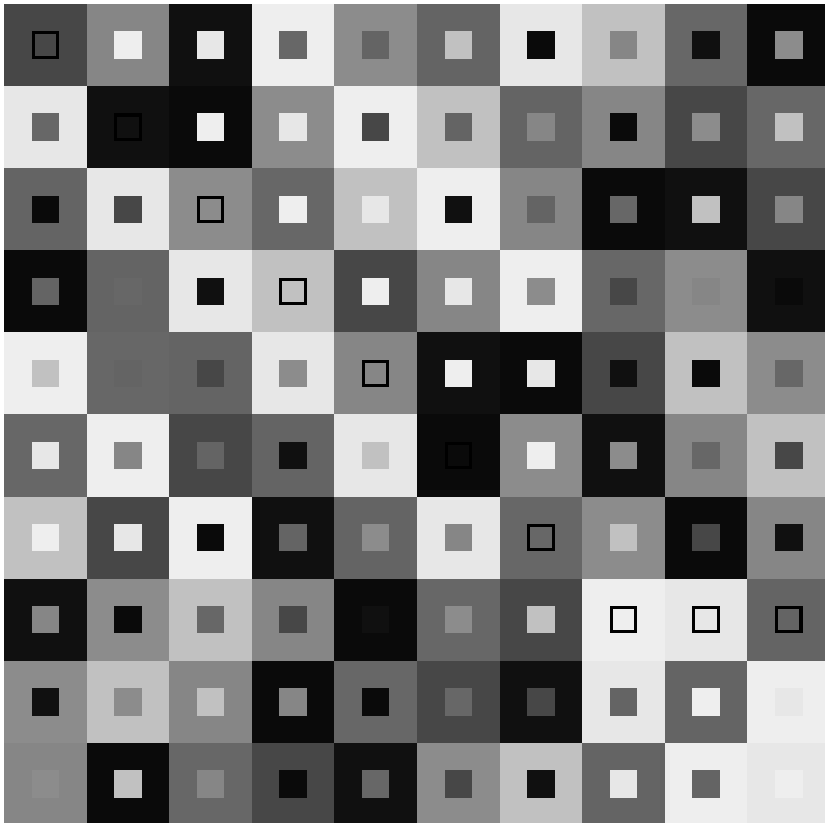


Abbildung 13.5.: Ein Beispiel für ein *Graeco-Latin square* 10. Ordnung. Die größeren Quadrate stellen das *latin square* dar, die kleineren, inneren das *greek square*. Jede der 100 möglichen Kombinationen tritt genau einmal auf. In 10 Fällen sind inneres und äußeres Quadrat gleichfarbig.

```
    Min       1Q   Median       3Q      Max
-1.4840  -0.9206  -0.1015   0.4606   1.6375

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.90453    0.63899   1.416   0.1569
Saison      -0.28725    0.18419  -1.560   0.1189
HabitatB     0.01664    0.52993   0.031   0.9750
HabitatC     0.34221    0.54135   0.632   0.5273
HabitatD     0.17358    0.55002   0.316   0.7523
ArtGras     -0.14400    0.61534  -0.234   0.8150
ArtKraut     1.04622    0.49469   2.115   0.0344 *
ArtMoos     -0.69452    0.71340  -0.974   0.3303
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 27.064  on 15  degrees of freedom
Residual deviance: 12.382  on  8  degrees of freedom
AIC: 60.434

Number of Fisher Scoring iterations: 6
```

Oder, um eine ANOVA-artige Tabelle zu erhalten:

```
> anova(glm(Frass ~ Saison + Habitat + Art, family = poisson,
+     data = LSD), test = "Chisq")

Analysis of Deviance Table

Model: poisson, link: log

Response: Frass

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                       15    27.0637
Saison   1   2.1779        14    24.8858    0.1400
Habitat  3   0.1334        11    24.7523    0.9875
Art      3  12.3705         8    12.3818    0.0062
```

Wir sehen, dass für die Residuen nur wenige Freiheitsgrade zur Verfügung stehen, und entsprechend die statistische Auflösungskraft entsprechend gering sein wird. Als Lösung bietet sich einfach die Replikation des gesamten Designs an, was auch die Analyse der Interaktionen möglich macht.

13.3.3. Design für deskriptive Studien (descriptive sampling design)

Sie dienen der Beschreibung eines Aspektes der Welt, und können keine Kausalitäten belegen. Weil ihre Ergebnisse leider gelegentlich immer noch als Beschreibung von Ursache-Wirkungszusammenhängen gewertet werden, gelten sie als wissenschaftlich schwach. Tatsächlich sind sie aber oft ein erster Schritt zum Verständnis von Mustern, die dann durch weitere Hypothesen kausal untersucht werden können.

Da keine experimentelle Manipulation erfolgt, können diese auch nicht randomisiert werden. Stattdessen werden hier Daten für eine rein deskriptiv-korrelative Analyse erhoben. Ein Beispiel:

Wir wollen die Artenvielfalt einer Stadt quantifizieren. Jetzt gibt es zwei grundsätzliche Ansätze: (1) Wir legen einen (oder mehrere) Transekte zufällig/regelmäßig durch unser Untersuchungsgebiet und erfassen die Artenvielfalt entlang dieses Transektes. (2) Wir teilen die Stadt in verschiedene Habitattypen (Stratum) (z.B. Parkplatz, Wald, Park) ein und beproben diese selektiv.

Beide Ansätze haben ihre Probleme. Der Erste wird *simple random sampling* genannt, wenn die Stichproben zufällig bestimmt werden, sonst *systematic sampling*. Er ist einfach durchzuführen und leicht zu analysieren, führt aber zu unterschiedlichen Stichprobengrößen in den verschiedenen Habitattypen. Damit wird sich auch die Varianz in diesen Strata unterscheiden, ein Verstoß gegen eine Grundannahme von ANOVA.

Der zweite, *stratified random sampling*, erfordert mehr Vorbereitung, aber ermöglicht es, ausreichend viele Stichproben in jedem Stratum zu erfassen. Im Prinzip werden in der Vorbereitung die Prozentanteile der verschiedenen Strata bestimmt (etwa Fläche je Habitattyp) und die Gesamtstichprobenzahl N dann gleichmäßig auf die Strata verteilt. Innerhalb der Strata wir dann ein einfaches randomisiertes sampling durchgeführt. Damit fallen auf jedes der k Strata dann N/k Stichproben. Somit hat jeder Punkt innerhalb eines Stratum die gleiche Chance ausgewählt zu werden, und diese Wahrscheinlichkeit ist abhängig von der Größe des Stratum. Diese gleichmäßige Erfassung führt zu einer konsistenteren Abschätzung der Varianz innerhalb der Strata, und sie ist damit der ANOVA zugänglich. Allerdings kann dies bedeuten, dass große Strata mit extrem wenigen Stichproben erfasst werden, und diese dann nicht mehr repräsentativ sind.

Im Prinzip kann jedes Stratum in weitere Substrata usw. untergliedert werden. Dies wird etwa bei der Erfassung von Schulqualitäten gemacht: Je Schultyp, je Bundesland, je Gemein-

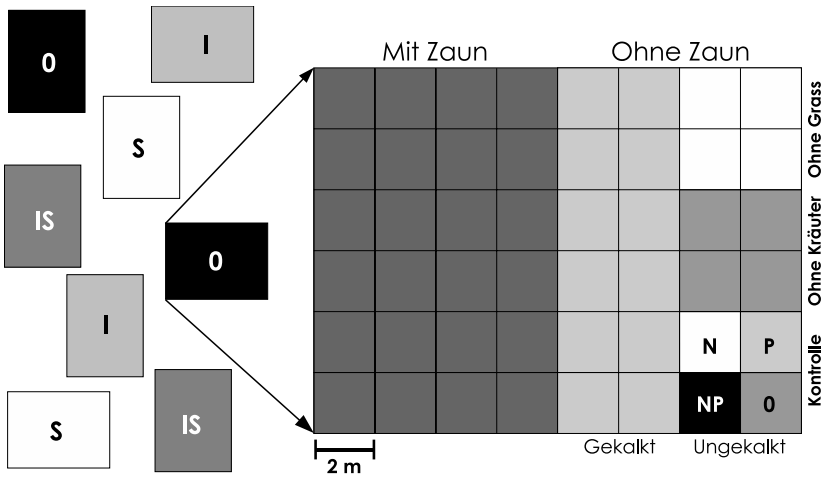


Abbildung 13.6.: Beispiel eines *split-split-split-split-plot* Designs. Auf der Ebene des gesamten 12 × 16 m großen plots 2 Behandlungen mit je zwei Leveln wurden faktoriell angewandt: Schneckengift (0/1) und Insektengift (0/1). Jeder *plot* wurde dann wiederholt geteilt: Zunächst wird eine Hälfte eingezäunt/nicht eingezäunt; die eine Hälfte darin wurde gekalkt (oder nicht). In diesem *split-split-plot* wurden dann drei Artenzusammensetzungen realisiert (ohne Grass, ohne Kräuter, Kontrolle), und in jedem dieser *subplots* wurde faktoriell mit Stickstoff und Phosphor gedüngt. Damit gibt es hier 384 *plots* von vier Quadratmeter Größe. Der Erdenker dieses Experiments, M.C. Crawley, Imperial College, GB, ist berüchtigt für diese Art „effizienter" Resourcennutzung, wenngleich dieses Beispiel von ihm frei erfunden wurde. (Verändert nach Crawley 2002)

de, je Schule, je Jahrgang, je Klasse werden i Mädchen und j Jungen getestet. Dies fasst man dann unter dem Begriff *multistage sampling*.

Nur der Vollständigkeit halber seien noch *quota* und *purposive sampling* erwähnt. Dabei haben die Stichpunkte *nicht* die gleiche Wahrscheinlichkeit ausgewählt zu werden. Die Auswahl der Punkte wird dem Erfasser überlassen, um z.B. möglichst gezielt einen bestimmten Personenkreis (jung-dynamische Alleinerziehende) zu beproben. Diese Willkür verbietet eigentlich jede weitere Statistik. Praktisch jede Umfrage in Fussgängerzonen ist von dieser Art, während die Telefonumfrage zumeist ein *simple random sampling* ist.
Literatur: `http://www.cems.uwe.ac.uk/~pwhite/SURVEY1/node26.html`

13.3.4. Split-plot design

1. Definition Die verschiedenen Level einer Behandlung werden innerhalb jedes Levels einer anderen Behandlung appliziert.

2. Beispiel Wir wollen den Effekt von Düngung und Beweidung untersuchen. Dazu errichten wir Ausschlusskäfige (*exclosures*), innerhalb derer wir dann die beiden Düngelevel mit/ohne applizieren. Dass dies ein *split-plot* Design ist erkennen wir leicht daran, dass die beiden Behandlungen auf zwei unterschiedlichen Flächengrößen appliziert werden: Der Weidegängerausschluss auf der gesamten Fläche des *exclosures*, aber die der Düngung nur auf der Hälfte dieser Fläche.

3. Details *Split-plot* Designs werden zumeist aus praktischen Gründen benutzt. Im obigen Beispiel brauchen wir dann bei 10-facher Replikation eben nur 10 *exclosures*, und nicht

20, wie bei einem nicht-gesplitteten Design. Ein illustratives, wenngleich extremes, Beispiel ist in Abbildung 13.6 dargestellt.

4. Stärken & Schwächen Die praktischen Vorteile werden mit zwei Nachteilen erkauft: (1) Wird die Statistik etwas komplizierter, da wir die *split-plot*-Struktur mit einbeziehen müssen. Dies führt dazu, dass im obigen Beispiel zunächst der *exclosure*-Effekt analysiert wird, und dann *exclosure* und Düngung zusammen. (2) Wir wenden unsere Behandlungen auf unterschiedlichen Flächengrößen an. Wenn aber nun z.B. die Wirkung von Konkurrenz auf eine Einzelpflanze nur auf 1/2 m^2 untersucht wird, Düngung aber auf 2 m^2, dann unterscheidet sich entsprechend auch die Einflusssphäre der einzelnen Pflanze: Bei der Düngung können viel mehr Arten und Individuen an der Nutzung des Düngers teilnehmen, als bei der Konkurrenz. Praktisch wird dies oft wenig ausmachen, aber dieses Problem sollten wir trotzdem im Hinterkopf haben.

5. Literatur Crawley (2002); Quinn and Keough (2002)

6. Beispielrechnung Anstelle der Reproduktion des gesamten *split-split-...-plot* Designs von Crawley werden wir uns nur die obersten *level* anschauen. Das (erfundene) Experiment sieht wie folgt aus (siehe Abb. 13.6): Eine Behandlung mit Insektizid wird mit einer mit Molluskizid gekreuzt und zweifach repliziert. Dies sind die Behandlugen auf der gesamten Fläche eines *plots* (linke Hälfte der Abbildung). Jetzt wird in jedem *plot* eine Hälfte durch einen Zaun gegen Kaninchen umgeben. Jeder dieser sub-plots wird wiederum in zwei Hälften geteilt, die gekalkt werden (oder nicht). Somit haben wir einen *split-split-plot*-Versuch. Die Daten geben natürlich noch viel mehr her, aber das soll uns hier nicht interessieren. Wir laden also erst die Daten, mitteln dann die Werte über die ignorierten, niedrigeren level und generieren so einen neuen Datensatz, den wir dann analysieren.[1] Beachte vor allem wie die „splits“ in der Formel kodiert werden!

```
> crawley <- read.table("splitplot.txt", header = T)
> attach(crawley)
> splitplot <- aggregate(Biomass, list(Block, Insect, Mollusc,
+     Rabbit, Lime), mean)
> colnames(splitplot) <- colnames(crawley)[c(1:5, 8)]
> fm <- aov(Biomass ~ Insect * Mollusc * Rabbit * Lime +
+     Error(Block/Rabbit/Lime), data = splitplot)
> summary(fm)

Error: Block
               Df Sum Sq Mean Sq F value   Pr(>F)
Insect          1 34.551  34.551 34.2712 0.004248 **
Mollusc         1  0.729   0.729  0.7229 0.443088
Insect:Mollusc  1  0.921   0.921  0.9139 0.393209
Residuals       4  4.033   1.008
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Error: Block:Rabbit
                     Df Sum Sq Mean Sq   F value    Pr(>F)
Rabbit                1 32.399  32.399 4563.5924 2.877e-07 ***
Insect:Rabbit         1  0.033   0.033    4.6985   0.09607 .
Mollusc:Rabbit        1  0.001   0.001    0.1600   0.70963
Insect:Mollusc:Rabbit 1  0.021   0.021    2.9078   0.16335
Residuals             4  0.028   0.007
```

[1] Wir können auch einfach das unten definiert Modell mit Crawleys vollem Datensatz rechnen. R gibt uns dann zusätzlich zum aufgeführten *output* auch noch eine Angabe über die Varianz, die in den restlichen Strata steckt. Unser reduziertes Modell liefert die identischen Aussagen wie Crawleys vollständiges (siehe Crawley 2002, S.354 ff). Allerdings sind unsere SS und MS-Werte andere, da wir ja weniger Daten insgesamt betrachten.

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

Error: Block:Rabbit:Lime
                          Df Sum Sq Mean Sq   F value    Pr(>F)
Lime                       1 7.2198  7.2198 1918.2643 8.148e-11 ***
Insect:Lime                1 0.0028  0.0028    0.7556   0.41001
Mollusc:Lime               1 0.0102  0.0102    2.7005   0.13894
Rabbit:Lime                1 0.0122  0.0122    3.2284   0.11010
Insect:Mollusc:Lime        1 0.0043  0.0043    1.1425   0.31631
Insect:Rabbit:Lime         1 0.0003  0.0003    0.0794   0.78529
Mollusc:Rabbit:Lime        1 0.0075  0.0075    2.0043   0.19458
Insect:Mollusc:Rabbit:Lime 1 0.0389  0.0389   10.3353   0.01233 *
Residuals                  8 0.0301  0.0038
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Es gibt auch mindestens zwei spezielle *packages* um sogenannte gemischte Modelle zu analysieren, **nlme** und **lme4**. Die Benutzung von `lme` (*linear mixed effects*) ist immer wieder eine Herausforderung, aber in diesem Fall ist es einfach. Kernunterschied zum `aov` ist, dass wir die Zufallseffekte auch als solche Deklarieren und das *splitting* und *nesting* entsprechend kodieren:

```
> require(nlme)

Loading required package: nlme
[1] TRUE

> fme <- lme(Biomass ~ Insect * Mollusc * Rabbit * Lime, random = ~1 |
+     Block/Rabbit/Lime, data = splitplot)
> anova(fme)

                          numDF denDF  F-value p-value
(Intercept)                   1     8  922.328  <.0001
Insect                        1     4   34.271  0.0042
Mollusc                       1     4    0.723  0.4431
Rabbit                        1     4 4563.592  <.0001
Lime                          1     8 1918.264  <.0001
Insect:Mollusc                1     4    0.914  0.3932
Insect:Rabbit                 1     4    4.698  0.0961
Mollusc:Rabbit                1     4    0.160  0.7096
Insect:Lime                   1     8    0.756  0.4100
Mollusc:Lime                  1     8    2.701  0.1389
Rabbit:Lime                   1     8    3.228  0.1101
Insect:Mollusc:Rabbit         1     4    2.908  0.1634
Insect:Mollusc:Lime           1     8    1.143  0.3163
Insect:Rabbit:Lime            1     8    0.079  0.7853
Mollusc:Rabbit:Lime           1     8    2.004  0.1946
Insect:Mollusc:Rabbit:Lime    1     8   10.335  0.0123
```

Wir sehen, dass die Werte alle identisch sind, wenngleich anders geordnet. Bei gemischten Modellen wird neben den Freiheitsgraden des Effekts (`numDF` = *numerator degrees of freedom* = Zählerfreiheitsgrade) auch die angegeben, gegen die getestet wird (`denDF` = *denominator degrees of freedom* = Nennerfreiheitsgrade). Wenn die Nennerfreiheitsgrade nicht stimmen, ist das Modell falsch kodiert!

In einem gemischten Modell werden für jedes Level eines Zufallseffekts ein eigener Achsenabschnitt gefittet. Wenn wir diesen angezeigt haben wollen, dann können wir dies über `summary(fme, verbose=T)` erreichen. Der *output* ist recht voluminös und deshalb hier weggelassen.

Zur Interpretation müssen wir uns durch die einzelnen Strata kämpfen. Auf der obersten Ebene, der des gesamten *plots*, haben wir nur Molluskizid und Insektizid als Behandlungen. Ihre Effekte werden

zuerst dargestellt. Es gibt 8 Versuchseinheiten, jede Behandlung und die Interaktion haben einen Freiheitsgrad, so dass $8-1-1-1=5$ Freiheitgrade übrig bleiben (somit haben die Residuen $5-1$ Freiheitsgrade). In diesem obersten Stratum ist der Effekt des Insektizids signifikant.
Im nächsten Stratum kommt der Effekt des Kaninchenausschlusses hinzu. Zunächst können wir diesen Effekt direkt bewerten (und er ist signifikant). Sodann können wir die Interaktionen mit *allen* Faktorenkombinationen des vorigen Stratums analysieren. Für die Analyse stehen uns jetzt je *plot* zwei Werte zur Verfügung (also 16). Davon ziehen wir für die Berechnung der Freiheitsgrade zunächst die Summe aller Freiheitsgrade der vorigen Strata ab ($16-7=9$). Die Verbleibenden werden auf die Effekte verteilt (jeweils einer), und es bleiben $5-1$ für die Residuen dieses Stratums übrig.
Noch ein Wort zur Interpretation: In diesem zweiten Stratum tauchen die Behandlungen des ersten Stratums nicht als Haupteffekte auf. Wollten wir etwa den Effekt des Molluskizids bestimmen, so müssten wir ja über die Kaninchenflächen mitteln, um Pseudoreplikation zu vermeiden. Genau dies wird aber im ersten Stratum gemacht.
Im dritten Stratum kommt wie im zweiten der neue Faktor Kalkung (`Lime`) dazu, nebst allen Interaktionen mit den Effekten des Stratums darüber. Die Freiheitsgrade berechnen sich wie gehabt: $32-7-8=17$ für dieses Stratum, davon 8 für die Effekte, bleiben $9-1=8$ für die Residuen. Die Signifikanz der 4-Wege-Interaktion muss hier genauso interpretiert werden wie in einem konventionellen Design: Der Effekt jeder Behandlung ist abhängig von dem Level jeder anderen Behandlung.
An dieser Stelle muss sich ein wenig Modellkritik anschließen. Die maximale Interaktion ist mit $P<0.05$ signifikant. Andererseits führen wir ja mehrfache Behandlungstests durch, da die Effekte der höheren Ebene immer auf der tieferen wiederum in Interaktionen mitgetestet werden. Es bietet sich deshalb an, denjenigen p-Wert, den wir als signifikant anerkennen, von 0.05 herabzusetzen, je nach Anzahl der „*splits*“, z.B. auf 0.01 oder 0.005 (hierfür gibt es keine feste Regel). Wenn wir entsprechend das Modell vereinfachen (durch manuelles Löschen nicht-signifikanter Interaktionen in den verschiedenen Leveln), so bleiben nur unsere Haupteffekte (`Insect`, `Rabbit`, `Lime`) auf ihrer jeweiligen Flächengröße=Stratum signifikant.

13.3.5. Nested design

1. Definition Innerhalb einer Behandlungseinheit (*experimental unit*) werden mehrere Messungen durchgeführt, entweder parallel (mehrere Objekte parallel gemessen) oder zeitlich sequentiell (*repeated measurements*).

2. Beispiel Wir wollen untersuchen, ob der Ausschluss von Weidegängern sich auf das Geschlechterverhältnis einer Weidenart auswirkt. Dazu errichten wir n *exclosures* und Kontrollen, warten 10 Jahre und erfassen dann auf jeder dieser $2n$ Flächen das Geschlechterverhältnis, indem wir z.B. 10 Quadrate zufällig platzieren, und das Geschlecht der darin vorkommenden Weidenpflanzen notieren. Das *nesting* entsteht dadurch, dass die 10 Quadrate innerhalb der Behandlungseinheit liegen, und somit keine Replikate sondern Unterproben (*subsamples*) sind. Das gleiche gilt, wenn wir statt 10 Jahre zu warten jedes Jahr in einem Quadrat (oder der Gesamtfläche) das Geschlechterverhältnis erfassen. Diese Datenpunkte sind natürlich nicht unabhängig, sondern genested.

3. Details Die Auswertung genesteter Designs ist nicht ohne. Insbesondere wenn wir zeitliches *nesting* praktizieren, muss der Faktor `Zeit` sich ja nicht notwendigerweise linear auf die Antwortvariable auswirken. Entsprechend müssen wir die Struktur dieser Korrelation mit in das Modell einspeisen.

4. Stärken & Schwächen *Nesting* muss berücksichtigt werden. Dies ist statistisch möglich, so dass wir unsere 10 Messungen innerhalb einer Behandlungseinheit nicht vor der Analyse zu mitteln brauchen. Die Nutzung dieser *subsamples* oder *repeats* ist sicherlich die große Stärke des *nesting*. Gleichzeitig ist die Analyse nicht mehr so selbstverständlich.

5. Literatur Quinn and Keough (2002); Pinheiro and Bates (2000)

6. Beispielrechnung Schauen wir uns das oben erwähnte Beispiel an. Ein 10 × 30 m großer Ausschlusskäfig verhindert die Beweidung eines Stücks arktischer Tundra durch Rentiere. Daneben liegt eine ebenso große Kontrollfläche. Dieser Arrangement (`block`) wird 6 Mal repliziert. Nach 6 Jahren bestimmen wir das Geschlecht von 10 zufällig ausgewählten Weidenpflanzen je Behandlungseinheit. Damit ist jede Pflanze ein *subsample* oder Pseudoreplikat, also genested innerhalb der Behandlung `exclosure`. Dem GLM müssen wir darob mitteilen, dass `block` ein Zufallseffekt (*random effect*) ist, der uns nicht wirklich interessiert, und dass alle Messungen der `subsamples` im Behandlungseffekt `excl` genested sind (Struktur steht hinter dem senkrechten Strich, beginnend mit der größten Einheit). Der Syntax im entsprechenden gemischten verallgemeinerten linearen Modell in R sieht so aus:

```
> willowsex <- read.table("willowsex.txt", header = T)
> require(MASS)
> summary(glmmPQL(fixed = female ~ excl, random = ~1 | block/excl,
+     family = binomial, data = willowsex))

iteration 1
Linear mixed-effects model fit by maximum likelihood
 Data: willowsex
  AIC BIC logLik
   NA  NA     NA

Random effects:
 Formula: ~1 | block
         (Intercept)
StdDev: 9.199434e-05

 Formula: ~1 | excl %in% block
         (Intercept)  Residual
StdDev: 3.397999e-05 0.9999998

Variance function:
 Structure: fixed weights
 Formula: ~invwt
Fixed effects: female ~ excl
                 Value Std.Error  DF    t-value p-value
(Intercept)  0.2006707 0.2616895 108  0.7668274  0.4449
excl        -0.9698038 0.3831561   5 -2.5310931  0.0525
 Correlation:
     (Intr)
excl -0.683

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-1.1055418 -0.6807455 -0.6807455  0.9045342  1.4689772

Number of Observations: 120
Number of Groups:
          block excl %in% block
              6              12
```

Die Interpretation ist etwas hässlich. Das Ergebnis, an dem wir interessiert sind, nämlich ob durch die *exclosure* das Geschlechterverhältnis in den Weiden verschoben wird, erfahren wir mittendrin, unter der Überschrift `Fixed effects: female` ∼ `excl`: Der Effekt `excl` ist in der Tat signifikant. Hier ist auch zu sehen, ob die Modellstruktur korrekt implementiert ist. Für den Effect `excl` stehen 5 Freiheitsgrade zur Verfügung, also Anzahl *exclosure* −1. Das ist korrekt. Ebenfalls interessant in diesem Zusammenhang ist die ganz unten aufgeführte Anzahl der Gruppen: 6 für `block`, 12 für `excl within block`, ebenfalls korrekt.

Kommen wir jetzt zum oberen Teil des *outputs*. Zunächst wird der Modellfit quantifiziert: `AIC` und `BIC` stehen für *Akaike* und *Bayes' Information Criterion*, `logLik` für *log-likelihood*. Diese Angaben wären für Modellselektion von Interesse, sind aber für die benutzte *quasilikelihood* nicht definiert, also: `NA`. Danach werden die Zufallseffekte gefittet, und zwar für jedes Stratum einzeln. An den Standardabweichungen sehen wir, dass mehr Variabilität zwischen den *exclosures* im Block besteht, als zwischen den Blöcken an sich. Da der *exclosure*-Effekt signifikant ist, ist dies nicht verwunderlich. Wenn wir die erhaltenen Parameter für den Anteil weiblicher Pflanzen zurückrechnen (sie sind ja bei binomialverteilten Daten standardmäßig logit-transformiert), so erhalten wir für die Kontrollen einen Wert von $\frac{e^{0.2}}{1+e^{0.2}} \approx 0.55$ und für die *exclosure* einen Wert von $\frac{e^{0.2-0.97}}{1+e^{0.2-0.97}} \approx 0.32$. Gar nicht schlecht für wahre Werte von 0.6 und 0.4, respektive.
Es sei noch erwähnt, dass wir zwar mittels `anova.lme(.)` auch eine ANOVA-artige Tabelle erzwingen können, diese aber inkorrekt ist, und sei es nur, weil statt des X^2- unveränderlich ein F-Test benutzt wird.

13.4. Abschätzung des benötigten Stichprobenumfangs

Nur für sehr einfache Versuche kann man mit der Kenntnis der zu erwartenden Streuung schon im Vorfeld eines Versuches den benötigten Stichprobenumfang errechnen.

Für kompliziertere Designs, nicht-normalverteilte Daten oder unbalancierte Versuchsdesigns hilft nur eine Simulation der Ergebnisse. Die Grundlage hier sind eine bekannte Verteilung der Residuen (d.h. die Streuung der Werte muss bekannt sein oder abgeschätzt werden könne), sowie eine Festlegung der Größe der Effekte, die wir mit dem Experiment detektieren wollen. Wenn wir beispielsweise den Einfluss vom Rapsglanzkäfer auf den Ertrag von Raps bestimmen, so müssen wir definieren, dass etwa 10% weniger Ertrag als mindestens nachzuweisender Effekt gelten mag. Dann können wir Daten mit einer Rapsglanzkäferbehandlung erfinden, die eine Ertragseinbuße um 10% simuliert. Diese Daten können wir dann analysieren und den Stichprobenumfang so lange erhöhen, bis die 10% auch tatsächlich signifikant sind.

Das Vorgehen sei hier für einen anderen, etwas komplizierten Versuch einmal illustriert. Es soll untersucht werden, wie sich die Zugabe von labilem Kohlenstoff (in der Form von Zucker) auf den Wuchs von 20 verschiedenen Pflanzenarten auswirkt. Diese Arten rekrutieren sich aus zwei Wuchsformen (Kräuter und Gräser) sowie aus zwei Lebenserwartungen (ein- und zweijährig). Der Zucker wird in vier Stufen zugegeben. Die Arten 1 bis 10 sind einjährig, 11 bis 20 zweijährig. Alle geraden Arten sind Gräser, die ungeraden Kräuter:

```
> Zucker <- c(0, 5, 50, 500)
> Arten <- 1:20
> Reps <- 1:20
> sim.data <- expand.grid(Arten = Arten, Zucker = Zucker, Reps = Reps)
> Annuell <- ifelse(Arten < 11, 1, 0)
> Gras <- ifelse(Arten%%2 == 0, 1, 0)
> sim.data <- cbind(sim.data, Annuell = Annuell, Gras = Gras)
> rm(Zucker, Arten, Reps, Annuell, Gras)
```

Jede Art erhält jetzt eine mittlere Biomasse für den unbehandelten Zustand. Die Daten werden zufällig aus einer Verteilung gezogen, zumeist aus einer Normalverteilung mittels `rnorm`. Hier benutzen wir aber eine uniforme Verteilung (mittels `runif`[2]) und ziehen Werte aus dem Intervall 10 bis 100.

```
> set.seed(1)
> null.biomasse <- rep(floor(runif(20, 10, 100)), nrow(sim.data)/20)
```

Wir nehmen an, dass der Zucker das Wachstum reduziert, und zwar logarithmisch. Einjährige sollen stärker reagieren als zweijährige Pflanzen und Gräser stärker als Kräuter.

[2]Im `rnorm`-Befehl müssen wir dann bei der Option `sd` den Wert für die Standardabweichung der Residuen eingeben. Dieser muss entsprechend aus Voruntersuchungen oder Literatur abgeschätzt werden.

```
> attach(sim.data)
> effekt.zuck <- 0.25 * log10(Zucker + 1)
> effekt.ann <- ifelse(Annuell, 0.3, 0.2)
> effekt.gras <- ifelse(Gras, 0.5, 0.2)
> effekt.total <- effekt.zuck * (effekt.ann + effekt.gras)
> detach(sim.data)
> biomasse <- null.biomasse * (1 - effekt.total)
> sim.data2 <- cbind(sim.data, Biomasse = round(biomasse, 2))
> rm(null.biomasse, biomasse)
> attach(sim.data2)
> table(list(Arten, Zucker))[1:5, ]

   .2
.1   0  5 50 500
  1 20 20 20  20
  2 20 20 20  20
  3 20 20 20  20
  4 20 20 20  20
  5 20 20 20  20


> tapply(Biomasse, list(Arten, Zucker), mean)[1:5, ]

   0     5    50   500
1 33 29.79 25.96 21.86
2 43 36.31 28.31 19.78
3 61 55.07 47.98 40.41
4 91 76.84 59.92 41.86
5 28 25.28 22.02 18.55
```

Mit `table` können wir durchzählen lassen, wieviele Werte es für eine Faktorenkombination gibt. Wenn wir als Faktorenkombination `Arten` und `Gras` eingeben, so sehen wir, dass manche Arten 80, andere 0 Werte haben. Dies ist logisch, da die mit 80 Werten Gräser sind, die anderen Kräuter.

Der `tapply`-Befehl fasst dann die Biomassewerte zusammen, hier durch mitteln. Es kann aber jede beliebige Funktion übergeben werden (also auch `sd`, `median` oder selbstdefinierte). (Für beide Funktionen lassen wir uns hier aus Platzgründen nur die ersten 5 Zeilen ausgeben.)

Jetzt haben wir unseren Datensatz zusammen. In einer Analyse dieser Daten können wir nun unterschiedlich viele Replikate mit in die Auswertung nehmen. Wenn wir mehr Replikate brauchen, dann müssen wir bei der Generierung oben statt 20 etwa 50 angeben. Es sei hier nur kurz darauf hingewiesen, dass diese Daten ausführlich im Abschnitt genestete ANOVA behandelt werden. Hier sei nur für einen Faktor exemplarisch das Vorgehen zur Bestimmung der Stichprobenzahl dargelegt.

```
> summary(aov(Biomasse ~ Zucker + Annuell + Gras, data =
+     sim.data2[sim.data2$Reps < 5, ]))

             Df Sum Sq Mean Sq F value    Pr(>F)
Zucker        1  18966   18966 42.2354 3.150e-10 ***
Annuell       1    503     503  1.1190    0.2909
Gras          1    920     920  2.0489    0.1533
Residuals   316 141899     449
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> p.wert <- 1:3
> for (i in 2:21) p.wert[i - 1] <- summary(aov(Biomasse ~ Zucker +
+     Annuell + Gras, data = sim.data2[sim.data2$Reps < i, ]))[[1]][3,
+     5]
> plot(p.wert, type = "b", pch = 16, lwd = 3, xlab = "Anzahl Replikate")
> abline(h = 0.05)
```

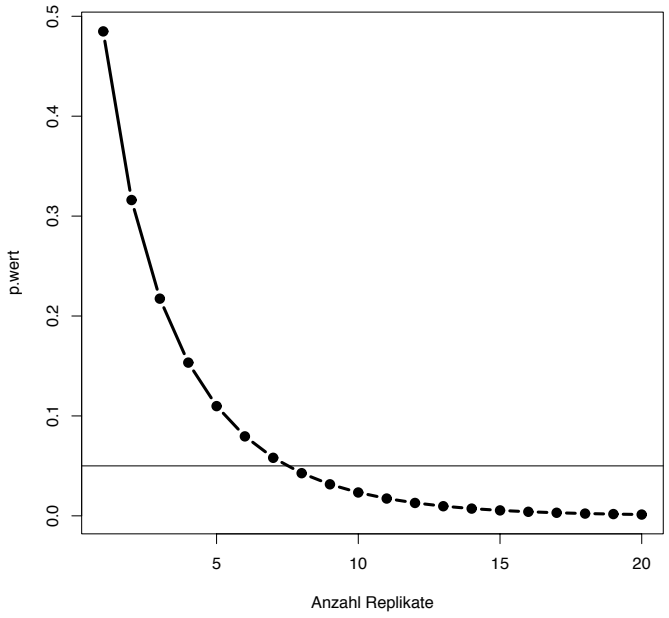


Abbildung 13.7.: Abnahme des Signifikanzwertes mit zunehmender Replikation für simulierte Daten.

Die `for`-Schleife extrahiert aus dem Modell den p-Wert für 1 bis 20 Replikate. Das Ergebnis des `summary`-Befehls ist eine Liste mit einem Element, auf das wir mit der doppelten eckigen Klammer zugreifen (`[[1]]`). Innerhalb des Listeneintrags hat dann der p-Wert von `Gras` die Position dritte Zeile, fünfte Spalte (`[2,5]`).

Es zeigt sich (Abb. 13.7), das der p-Wert für `Gras` mit der Replikation abnimmt. Der kritische Wert von 0.05 wird dabei erst mit 9 Replikaten unterschritten. Beachte dass dies erst eine Simulation ist. Bei einem erneuten Durchlauf werden sich andere Werte ergeben, und wir sollten deshalb dieses *Procedere* mindestens 100 Mal wiederholen.

Begrenzter Arbeitsaufwand: mehr Replikate oder mehr subsamples?

Ein häufiges Problem ist die Abwägung zwischen der Genauigkeit, mit der ich eine Probe untersuche, und dem Stichprobenumfang. Stellen wir uns beispielsweise vor, dass wir den Mykorrhizierungsgrad von Wurzeln in Abhängigkeit einer Behandlung (etwa mit und ohne Dünger) untersuchen wollen. Eine sehr sorgfältige Untersuchung einer Probe beinhaltet das Betrachten von mehreren Hundert Wurzelsegmenten. Für jedes Segment wird dann erfasst, ob Mykorrhiza vorhanden ist, oder nicht. Ein Segment zu färben und zu betrachten dauert, sagen wir, eine Minute. Dann brauche ich pro Probe 300 Minuten (also 5 Stunden), um einen einzigen Wert zu erarbeiten: den Mykorrhizierungsgrad einer Probe. Die Genauigkeit beträgt $1/300 \cdot 100\% = 0.33\%$. Wenn wir nur 100 Segmente betrachten, dann sinkt die Genauigkeit auf 1%, aber wir haben auch „nur“ anderthalb Stunden damit zugebracht. (Oftmals ist der Zusammenhang zwischen Genauigkeit und Zeitaufwand nicht so linear wie hier. Beispielsweise muss jede Probe gewaschen, sortiert, gefärbt, präpariert, ausgerichtet usw. werden, unabhängig davon, wieviele Segmente ich betrachte. Dann macht es keinen so großen Unterschied mehr, ob ich 100 oder 300 Segmente betrachte.)

Wenn ich einen schwachen Effekt meiner Behandlung (etwa des Düngens) auf den Mykorrhizierungsgrad nachweisen will, dann kann ich die Genauigkeit je Probe erhöhen, oder meinen Probenumfang vergrößern. Was ist hier effizienter? Das hängt offensichtlich von zwei Parametern ab: 1. der Variabilität der Mykorrhizierung der Segmente; und 2. der Streuung des Mykorrhizierungsgrades meiner Proben (also der Residuen der Behandlungen).

Dieses Beispiel sei hier einmal vorgerechnet. Zunächst konstruieren wir einen Datensatz mit vielen Replikaten und Segmenten. Dann fügen wir Werte zur Mykorrhizierung ein. Die Variabilität zwischen den Segmenten wird durch den Parameter `s` gesteuert. Den absoluten Wert (hier 60 bzw 30% für ohne und mit Düngung) wird durch den Parameter `m` angegeben (hier 0.6 und 0.3). `mykor2` ist dann die Tabelle mit der maximalen Replikation und Anzahl betrachteter Segmente.
Im nächsten Schritt (erledigt durch die Funktion `mykor.fun`) lassen wir uns für eine Kombination von Replikation und Segmentzahl (`rs`) das Signifikanzniveau des Effektes Düngung (`NPK`) ausgeben. Dazu werden die Anzahl mykorrhizierter Segmente gezählt und in die logistische Regression hineingenommen.
Schließlich lassen wir diese Funktion das Signifikanzniveau für eine Vielzahl von Replikationen und Segmentzahlen berechnen und stellen diese dann graphisch dar (Abb. 13.8).

```
> NPK <- c(0, 1)
> rep <- 1:20
> seg <- 1:500
> mykor <- expand.grid(seg = seg, rep = rep, NPK = NPK)
> mykor.var <- function(m, s) {
+     rbinom(10000, 1, rnorm(1, mean = m, sd = s))
+ }
> col <- c(mykor.var(0.6, 0.5), mykor.var(0.3, 0.5))
> mykor2 <- cbind(mykor, col = col)
> rm(col, seg, rep, NPK)
> mykor.fun <- function(rs) {
+     r <- rs[1]
+     s <- rs[2]
+     mykor.ex <- mykor2[(mykor2$rep < (r + 1) & mykor2$seg <
+         (s + 1)), ]
+     neu <- aggregate(mykor.ex$col, by = list(NPK = mykor.ex$NPK,
+         rep = mykor.ex$rep), sum)
+     neu.mykor <- cbind(neu, s)
+     colnames(neu.mykor) <- c("NPK", "rep", "col", "total")
+     summary(glm(cbind(col, total) ~ NPK, binomial, data =
+         neu.mykor))$coefficients[2, 4]
+ }
> rs.vec <- expand.grid(r = 1:20, s = seq(1, 100, by = 10))
> res3 <- apply(rs.vec, 1, mykor.fun)
> res.mat <- matrix(res3, nrow = 20)
> colnames(res.mat) <- seq(1, 100, by = 10)
> rownames(res.mat) <- 1:20
> filled.contour(y = seq(1, 100, by = 10), x = 1:20, res.mat,
+     xlab = "Replikation", ylab = "Anzahl Segmente", col = grey(20:1/20),
+     key.title = title(main = "p-Wert"))
```

Abbildung 13.8 kann uns nun als Grundlage bei der Allokation unsere Ressourcen dienen. Wir können jedem Segment bzw. jedem Replikat Kosten zuweisen: ein Segment zu bearbeiten kostet 5 Minuten, ein Replikat hingegen 1 Stunde. Eine Beschränkung auf 5 Replikate erfordert dann aber mindestens 40 Segmente, also $40 \cdot 5 + 5 \cdot 60 = 500$ Minuten. Bei 10 Replikaten (und nur 22 Segmenten) sieht die Rechnung so aus: $22 \cdot 10 + 10 \cdot 60 = 820$ Minuten. Beachte, dass in diesem Beispiel ab etwa 10 Replikaten keinen Sinn macht, mehr als sagen wir 25 Segmente zu beproben.

Dabei müssen wir folgendes berücksichtigen: 1. können wir mit diesem Mindestmaß an Replikation nur den recht großen Mykorrhizierungsunterschied von 50% (0.3 vs. 0.6) nachweisen. Wenn wir einen kleineren Unterschied detektieren wollen verändert sich auch die Abbildung. 2. legen wir eine Streuung der Mykorrhizierung zugrunde, die höher oder niedriger ist, als wir in Wirklichkeit finden. 3. Mit 5 Replikaten haben wir auch nur Aussagen für 5 Proben, egal wieviel Tausend Segmente wir auch zählen, und egal wie signifikant der gefundene Unterschied auch immer sein mag. Mehr Replikate erhöhen die Belastbarkeit der Ergebnisse, mehr *subsamples* die Genauigkeit der Abschätzung jedes Replikats. 4. Extrem kleine Unterschiede

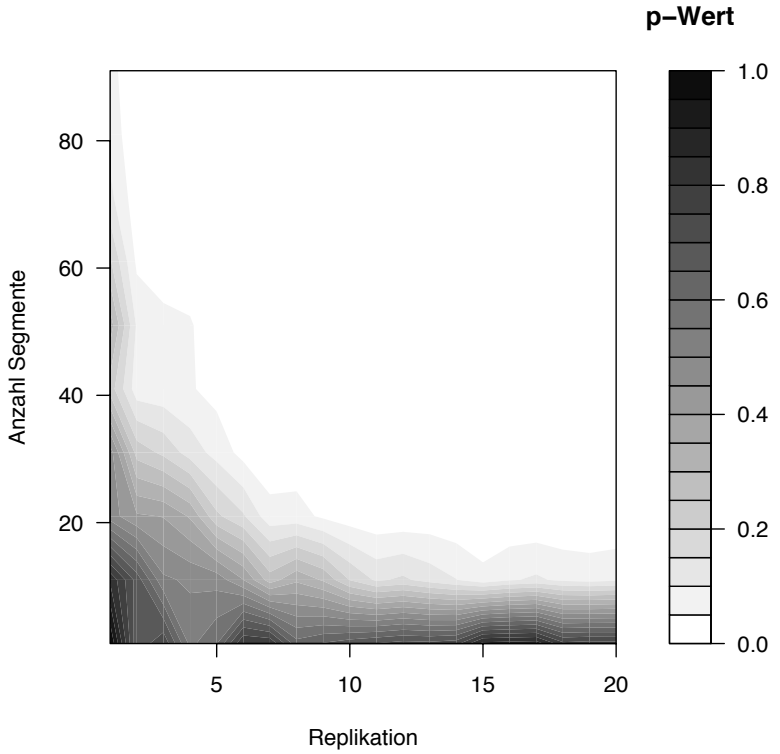


Abbildung 13.8.: Verhältnis zwischen Anzahl *subsamples* (Segmente) und Replikation und dem zu erwartenden Signifikanzniveau. Die weiße Fläche repräsentiert signifikante Unterschiede zwischen den beiden Behandlungsstufen.

zwischen Proben (die also eine hohe Anzahl an *subsamples* erfordern) sind unter Umständen von geringer biologischer Relevanz. Wenn also die Düngung zu einer Reduzierung der Mykorrhizierung von 50 auf 47% führt, so können wir dies getrost als irrelevant bezeichnen. Der Aufwand (hier vor allem an Segmentzählungen), um diese kleine Differenz nachzuweisen, wäre enorm, die ökologische Aussage hingegen minimal.

A. R

A.1. Was R? Wieso R? Wo R?

R ist die freie Version der Implementierung der statistischen Programmiersprache S. S wurde erfunden, um statistische Probleme zu bearbeiten und zu visualisieren. Inzwischen hat sich S aufgetrennt in die kommerzielle Software S-plus (`www.insightful.com`), und die freie R (`www.r-project.org`). Beide sind hinsichtlich ihres Syntax recht kompatibel, aber die von den jeweiligen Programmen bereitgestellten Befehle sind es nicht immer. Für den Benutzer hat dies Vor- und Nachteile: Was man in R lernt, kann man auch direkt in S-plus anwenden (und umgekehrt). Andererseits gibt es für S-plus zusätzliche Programmpackete („*modules*“), die es so in R nicht gibt (und umgekehrt). Alle Hauptverfahren und -funktionen stehen sowohl in R als auch in S-plus zur Verfügung.

Das spektakuläre an R ist seine Vielfältigkeit. Nahezu alles machen zu können hat allerdings einen Preis: den der Unhandlichkeit. Weil R eine Kommandozeilensoftware ist, bei der also der Benutzer an einem sogenannten *prompt* (bei R das Zeichen >) dem Programm mitteilt, was es tun soll. Wenn man also eine hässliche Abbildung produzieren will, so ist dies extrem einfach. Eine wunderschöne Abbildung hingegen bedeutet einigen Aufwand, da man R dann genau mitteilen muss, welche Linien wie dick, welche Beschriftung in welcher Schriftart, usw. Es ist möglich, aber umständlich. Andererseits gilt hier ebenfalls, dass alle einfachen Funktionen über recht kurze und übersichtliche Befehle zugänglich sind. Auch will hier noch erwähnt sein, das R für allen gängigen Betriebssystemen verfügbar ist, und dass der benutzte Code voll kompatibel ist.

Für den Nutzer statistischer Funktionen ist R ziemlich komfortabel. Die allermeisten Auswertungsroutinen sind als Funktion implementiert: für eine ANOVA reicht ein Befehl (`aov`). Wer aufwändigere Analysen machen will oder gar eigene Indizes und Funktionen programmieren will, der wird R schnell schätzen lernen, denn der Übergang von der reinen Funktionennutzung zum Funktionenerstellen ist fließend. Auch sind die Konstruktionen anderer Programmiersprachen vollständig in R verfügbar, von der `for`-Schleife über vektorisiertes Programmieren bis zur Objektorientierung. Wie Python ist R eine „interpretierte“ Programmiersprache, d.h. die übergebenen Befehle werden an einem Prompt ausgeführt. R-Programme können nicht kompiliert werden und ist deshalb nicht als Anwendung (.exe) transportierbar (wohl aber als Code für alle Plattformen).

R besteht aus einem Kernpaket (*base*), einem Schwung empfohlener zusätzlicher *packages* (die in der Grundinstallation enthalten sind), sowie einer Unzahl zusätzlicher *packages* (insgesamt über 100). Diese enthalten Funktionen für bestimmte Anwendungen. All dies ist zu beziehen über CRAN (Comprehensive R Archive Network): `http://cran.r-project.org`. Dort kann man dann auf das benutzte Betriebssystem klicken (z.B. Windows), und dann weiter nach „base“. Dort gibt es dann eine Datei wie etwa „R-2.6.2-win32.exe“, was R für Windows, Version 2.6.2 bedeutet. Diese lädt man sich in einen neuen Ordner auf der Festplatte (etwa C:\Programme \R) und doppelklickt die Datei, woraufhin sich der „Installationszauberer“ meldet und beim Installieren hilft. Fertig. Zusätzliche *packages* kann man am besten über R selbst installieren, so man einen schnellen Zugang zum Internet hat, oder man kann sie sich ebenfalls herunterladen und von einem lokalen Verzeichnis aus installieren. Dafür klickt man sich statt nach „base“ nach „contrib“, und lädt von dort die gewünschten .zip-Dateien in einen Ordner der Festplatte (z.B. ebenfalls C:\Programme\R).

Ebenfalls spektakulär ist die Arbeit von R-Enthusiaten, die nicht nur *packages* programmie-

ren, sondern auch Einführungen in R und R-Statistikskripte umsonst zur Verfügung stellen. Diese sind unter `www.r-project.org` unter `Documentation` contributed zu finden (vornehmlich in Englisch).

Bei der Installation kreiert R auch ein *icon* auf dem *desktop*. Damit wird R gestartet und meldet sich schlussendlich mit dem *prompt*: > Jetzt kann es losgehen!

A.2. Ein Editor für R: John Fox' R-commander, Tinn-R und Xemacs mit ESS

Wer längere Datenmanipulationen vornehmen will, oder (was jedem zu raten ist) die benutzten Befehle zu Dokumentations- und Erinnerungszwecken als Textdatei aufheben will, der sollte einen Editor zur Eingabe benutzen. Dafür steht beispielsweise der Editor (früher Notepad) von Windows (unter Zubehör) zur Verfügung. Dort tippt man die Befehle ein und kopiert sie dann nach R. MS WORD, WORD PERFECT, OPEN OFFICE WRITER oder ähnliche Textverarbeitungsprogramme sind dafür **NICHT GEEIGNET**! Sie haben automatische Ersetzungsfunktionen (etwa gerade Anführungszeichen durch typographische), die dazu führen können, dass das, was mein bei Word eintippt, nicht das ist, was man danach nach R kopiert!

Eleganter sind natürlich spezielle R-Editoren, wie etwas Tinn-R oder XEmacs. Beide sind kostenlos und werden in einem ESS (*Emacs Speaks Statistics*) genannten Packet für R angeboten (siehe `http://software.biostat.washington.edu/wikis/front/EmacsSpeaksStatistics/`). Damit steht eine Rechtschreibkontrolle für die R-Befehle in Form eines *syntax highlighting* zur Verfügung, ebenso wie kleinere Makros und Kopierknöpfe. Für Jemand, der mit Emacs/XEmacs schon LaTeX-Documente verfasst ist dies eine auf der Hand liegende Lösung. Wir nutzen vor allem Tinn-R: es ist klein, besitzt Syntax-Hervorhebung und ist recht zuverlässig. Nachteil: Tinn-R gibt es bislang nur für Windows. Unter Linux gibt es allerdings einen angeblich noch besseren Editor (Kite) und natürlich auch (X)Emacs.

R besitzt einen kleinen *script editor*, in den wir unsere Befehle eingeben, ausführen lassen und abspeichern können. John Fox hat darüberhinaus ein GUI (Graphic User Interface) für R entwickelt. Es führt eine Reihe einfacher Analysen auf Knopfdruck durch, die es in R-*code* übersetzt. So kann man mit dem R-COMMANDER über die grafische Benutzeroberfläche Daten einlesen, Histogramme von auszuwählenden Variablen machen, Regressionen und ANOVAs rechnen. Kompliziertere Verfahren stehen nicht zur Verfügung.

Ein großer Vorteil ist, dass das Erlernen von R deutlich vereinfacht wird. Wir benutzen zwar die grafische Oberfläche, aber der R-COMMANDER bildet unsere Knopfdrücke als R-*code* ab, so dass wir diesen Syntax danach modifizieren können. Wenn wir beispielsweise einen *scatterplot* machen, nachher aber die Symbole, Beschriftungen usw. ändern wollen, so müssen wir nur die entsprechenden Befehle ergänzen/abwandeln, nicht aber den Befehl aus dem Kopf kennen.

Für etwas erfahrenere R-Nutzer hat der R-COMMANDER allerdings kaum einen erkennbaren Wert, zumal er als Editor Syntax-unterstützenden Editoren wie Xemacs oder WinEdt unterlegen ist.

Beziehen kann man den R-COMMANDER über John Fox' *website*: `http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/`, oder noch ein einfacher in R selber über `install packages from CRAN` in den *pull-down* Menus. Das *package* trägt den Namen **Rcmdr**. Mittels `library(Rcmdr)` (oder über das *pull-down* `packages` → `load package`, dann `Rcmdr` auswählen) wird der R-COMMANDER installiert.

Die Benutzung des R-COMMANDERS ist etwas komfortabler, wenn wir statt der R-Grundeinstellung `multiple windows` (MDI) die Option `single window` (SDI) benutzen. Dies kann entweder in R über `edit` → `GUI preferences` und dann in der obersten Zeile `SDI` klicken erfolgen, oder schon beim Aufruf von R vom *desktop icon*, indem wir dort (nach Klick mit der rechten Maustaste unter Eigenschaften) in der Zeile `Ziel` folgende Zeichen hintanfügen:

`--SDI` Jetzt erscheint nur das R-Ausgabe und Grafikfenster, nicht aber der R-Hintergrund, der sonst den R-COMMANDER immer überdeckt.

A.3. Kurze Einführung

Den besten Eindruck erhält man anhand von Beispielen. Dafür arbeiten wir einfach die *sample session* von Venables et al. (2003) durch. Die in `Schreibmaschinenschrift` angegebenen Befehle werden hinter den prompt von R eingegeben, der normal gestellte Text dahinter erklärt ihre Funktion.

`help.start()` Hiermit wird im Standardwebbrowser eine on-line Hilfedatei geöffnet, die alle R-Funktionen der installierten Pakete enthält. Sehr nützlich zum Hin- und Herspringen. Die Alternative ist ein einfaches `?Befehl`, was die gleichen Informationen in einem neuen R-Fenster öffnet.

`x <- rnorm(50)`

`y <- rnorm(x)` Erzeugt 2 Vektoren mit jeweils 50 Zufallszahlen und weißt sie den Variablen x und y zu (`<-` bedeutet Zuweisung. Auch die Benutzung des `=` anstelle des `<-` ist möglich.). (Warum `rnorm(x)` und nicht `rnorm(50)`? Beides geht. Wenn das Argument länger ist als nur eine Zahl, nimmt R die Länge des im Argument angegebenen Vektors. Siehe `?rnorm`.)

`plot(x, y)` Macht einen Scatterplot der angegebenen Variablen. Das Ausgabefenster erscheint automatisch.

`ls()` Listet alle Objekte in Rs Arbeitsspeicher (*workspace*) auf.

`rm(x,y)` Löscht (*remove*) die im Argument angegebenen Objekte. (Aufräumen des Arbeitsspeichers)

`x <- 1:20` Macht eine Variable x mit Werten von 1 bis 20.

`w <- 1 + sqrt(x)/2` Macht eine Variable w aus den Werten von x gemäß der angegebenen Formel .

`dummy <- data.frame(x=x, y= x + rnorm(x)*w)` Erzeugt einen `data.frame`, also ein tabellenartiges Gebilde, mit dem Namen „dummy“, und mit den Einträgen `x` und `y`. Durch das `x=` und `y=` wird den neuen Daten ein Wert zugewiesen.

`dummy` Gibt eben jene Tabelle aus.

`fm <- lm(y ~ x, data=dummy)` Rechnet eine Regression (`lm` = lineares Modell), wobei `y` die abhängige und `x` die erklärende Variable ist. Das Ergebnis wird dann der Variablen `fm` zugewiesen (`fm` = fitted model). (Da im Augenblick nur ein Datensatz im Arbeitsspeicher ist, ist die Angabe `data=` hier eigentlich überflüssig.

`fm` Hiermit kann man sich das Regressionsmodell anzeigen lassen. Diese Informationen sind nur spärlich: y-Achsenabschnitt und Geradensteigung.

`summary(fm)` Hiermit erhält man dann die erwarteten Daten!

`fm1 <- lm(y ~ x, data=dummy, weight=1/w^2)` Beispiel für eine gewichtete Regression (`y` gewichtet mit der Variablen `w`, die hier die Standardabweichung darstellen soll).

`summary(fm1)` Die Zusammenfassung der Daten dieser statistische Analyse.

`attach(dummy)` Damit werden die Variablennamen der Tabelle „dummy“ zugreifbar.

`lrf <- lowess(x, y)` Führt eine nicht-parametrische lokale Regression durch.

`plot(x, y)` Scatterplot wie oben.

`lines(x, lrf$y)` Fügt eine Linie hinzu. Hierbei werden die angegebenen x,y-Wertepaare gradlinig verbunden (Sowohl `x` als auch `lrf$y` ist ein Vektor!). Der Ausdruck `lrf$y` greift auf die vorhergesagten y-Werte im nicht-parametrischen Regressionsmodell zu.

`abline(0, 1, lty=3)` Fügt eine Gerade mit dem y-Achsenabschnitt 0 und der Steigung 1 hinzu. `lty` beschreibt die Art der Strichelung.

`abline(coef(fm1), col = "red")` Fügt eine Gerade hinzu, deren Koeffizienten der gewichteten Regression entnommen werden (mittels `coef(fm1)`). `col` definiert die Linienfarbe.

`detach()` Macht die Spalten der Tabelle wieder „unsichtbar“ für R.

```
plot(fitted(fm), resid(fm),
 xlab="Vorhergesagte Werte",
 ylab="Residuen",
 main="Residuen gegen vorhergesagte Werte")
```
Plotted die Residuen der Regression (= gemessene − vorhergesagte Werte) gegen die vorhergesagten. Diagnostischer *plot* für die überprüfung auf Heteroskedatizität. `xlab`, `ylab` und `main` setzen die Achsen- und Abbildungsbeschriftung.

`qqnorm(resid(fm), main="Residuen Rang Plot")` Quantilen-Quantilen-plot (*q-q-plot* oder *normal scores plot*) zur überprüfung der Verteilung auf Ausreißer. Hier ausnahmsweise nicht sehr nützlich.

`rm(fm, fm1, lrf, x, dummy)` Wieder aufräumen.

Es folgen Daten von Michaelson und Morley, die in einem klassischen Experiment gezeigt haben, dass die Lichtgeschwindigkeit konstant ist.

`file.show(file="C:/Programme/R/rw1062/library/base/data/morley.tab")` Datentabelle wird in neuem Fenster angezeigt.

`mm <- read.table(file="C:/Programme/R/rw1062/library/base/data/morley.tab")` Liest Daten ein und steckt sie in die Variable `mm`. Alternative: Da diese Daten Teil des Grunddatenschatzes von R sind, kann man sie auch über `data(morley)` einlesen. Dann: `mm <- morley`

`mm` Zeigt Daten an. Sie bestehen aus 5 Experimenten (`Expt`), die jeweils 20 mal wiederholt wurden (`Run`). Die Lichtgeschwindigkeit (−295000 km/s) wurde gemessen (`Speed`).

`attach(mm)` Macht die Spalten von `mm` als Variablen verfügbar.

`Run <- factor(Run)` Kategorisiert die Werte von `Run`.

`Expt <- factor(Expt)` Kategorisiert die Werte von `Expt`.

`plot(Expt, Speed, main="Lichtgeschwindigkeit", xlab="Experiment Nr.")` Wenn die erklärende Variable ein Faktor ist (also kategorial), so ist die Grundeinstellung (*default*) für `plot` der *Box-and-Whiskers plot*.

`fm <- aov(Speed  Run + Expt, data=mm)` Führt eine ANOVA durch, mit `Run` und `Expt` als Faktoren, in einem randomisierten Blockdesign.

`summary(fm)` Gibt Ergebnisse der ANOVA aus.

`fm0 <- update(fm, . ~ . - Run)` Wiederholt die Analyse, die zu `fm` geführt hat, nimmt aber den Faktor `Run` wegen Nichtsignifikanz aus dem Modell, unter Beibehaltung der anderen Modellterme („`. ~ .`“).

`anova(fm0, fm)` Vergleicht die Modelle `fm0` und `fm`. Da diese sich nicht signifikant unterscheiden, ist das mit weniger Termen zu bevorzugen (`fm0`).

`detach()` Nimmt die Variablennamen wieder aus dem Suchpfad.

`rm(fm, fm0)` Räumt auf.

Wir kommen jetzt zu ein paar graphischen Möglichkeiten: Konturen und *image plots*.

`x <- seq(-pi, pi, len=50)` Macht einen Vektor mit 50 Elementen im Interval $[-\pi, \pi]$.

`y <- x` `y` sei identisch zu `x`.

`f <- outer(x, y, function(x, y) cos(y)/(1 + xˆ2))` `f` ist eine quadratische Matrix, produziert durch `outer()`, deren Spalten und Reihen durch `x` und `y` indiziert werden. Die Werte der Matrix werden durch eine Funktion von x und y bestimmt: $f(x, y) = cos(y)/(1 + x^2)$.

`oldpar <- par(no.readonly = TRUE)` Speichert die aktuellen Grundeinstellungen für Graphiken unter dem Namen `oldpar`.

`par(pty="s")` Macht die Graphikoberfläche quadratisch.

`contour(x, y, f)` Plottet `f`.

`contour(x, y, f, nlevels=15, add=TRUE)` Fügt weitere Linien hinzu.

`fa <-(f-t(f))/2` Sei `fa` der asymmetrische Teil von `f`. (`t()` transponiert `f`.)

`contour(x, y, fa, nlevels=15)` Plottet `fa`.

`par(oldpar)` Setzt die alten Parameter für die Graphikoberfläche.

`image(x, y, f)` Macht „*high density image plots*“ von `f` ...

`image(x, y, fa)` ... und `fa`.

`objects(); rm(x, y, f, fa)` Räumt auf.

R arbeitet auch mit komplexen Zahlen:

`th <- seq(-pi, pi, len=100)`

`z <- exp(1i*th)` `1i` codiert die komplexe Zahl $i = \sqrt{-1}$.

`par(pty="s")`

`plot(z, type="l")` Bei komplexen Argumenten stellt `plot` den imaginären gegen den realen Teil dar. Dies sollte ein Kreis sein. `type` codiert Linien, Punkte, Säulen, usw.

Wenn wir jetzt Zufallspunkte innerhalb eines Einheitskreises auswählen wollen, so könnten wir den imaginären und den realen Teil zufällig wählen:

`w <- rnorm(100) + rnorm(100)*1i` Jetzt liegen natürlich Werte außerhalb des Kreises.

`w <- ifelse(Mod(w) > 1, 1/w, w)` Hiermit werden diejenigen Werte, deren Länge (`Mod()`) größer 1 ist, auf den Wert `1/w` plaziert.

`plot(w, xlim=c(-1,1), ylim=c(-1,1), pch="+", xlab="x", ylab="y")` Plottet `w` in einem Graph zwischen -1 und 1 (für x- und y-Achse), die Werte werden als „+“ dargestellt, und die Achsen werden entsprechend benannt.

`lines(z)` Plottet den Einheitskreis. – Jetzt sind zwar alle Werte in dem Einheitskreis, aber ihre Verteilung ist nicht gleichmäßig. Ein Ausweg ist statt normalverteilter Zufallszahlen uniform verteilte zu benutzten:

```
w <- sqrt(runif(100))*exp(2*pi*runif(100)*1i)
plot(w, xlim=c(-1,1), ylim=c(-1,1), pch="+", xlab="x", ylab="y")
```

`lines(z)` Jetzt sehen die Punkte viel gleichmäßiger verteilt aus.

`rm(th, w, z)` Räumt auf.

`q()` So verlässt man R. Im Dialog wird der Benutzer gefragt, ob der „*workspace*“ gespeichert werden soll. Damit würden alle Variablen, *packages* usw. beim nächsten Gebrauch zur Verfügung stünden. Generell ist dies keine gute Wahl, da man doch sehr schnell vergisst, welche Variablen da noch im Hintergrund lauern.

A.4. Daten vorbereiten für R

A.4.1. Allgemeines zu Datenstrukturen

Wenn wir Daten erheben und kleinere oder größere Mengen Zettel aus dem Feld zum Computer tragen wird es Zeit, sich über die Art und Weise, wie wir diese Daten organisieren wollen. Im einfachsten und weitaus häufigsten Fall werden wir unsere Daten in einem Tabellenblatt auflisten können.

Abbildung A.1.: Beispiel für ein Standarddatenblatt, in dem Libellenfänge in verschiedenen Orten, mit mehreren Fallen je Ort, dargestellt sind. Dies wird dann etwa unter C:\Data\libellen.txt abgelegt.

```
Ort Falle Art Anzahl
Leipzig 1 Aeschna.affinis 4
Leipzig 1 Libellula.quadrimaculata 6
⋮
Leipzig 2 Aeschna.affinis 3
⋮
Halle 1 Libellula.quadrimaculata 2
⋮
Zwickau 17 Zonophora.batesi 1
```

Wir haben also eine Tabelle mit Spalten und Zeilen, wobei üblicherweise jede Messung („Fall“, *case*) eine eigene Zeile bekommt, und die Spalten entsprechend die gemessenen Variablen oder Datenkodes enthalten. Ein Beispiel ist etwa in Abbildung A.1 dargestellt.

Im Prinzip können wir alle vorstellbaren Daten so ablegen. Bei umfangreichen (z.B geographischen) Datensätzen wird häufig eine sog. relationale Datenbank aufgebaut, bei der jedem Punkt ein ganzer Wust von verschiedenen Datenthemen angehängt wird (etwa Bodenart, Niederschlag, Höhe über NN, Nutzungsform, Luftqualität, Grundwasserabstand usw.). Jetzt kann jedes Datenthema einzeln bearbeitet werden, oder alle Punkte mit bestimmter Bodenart oder ... . Die fragespezifische Zusammenstellung dieser Daten erfolgt dann über eine Datenbankabfrage.

A.4.2. Daten und R

R kann eine Vielzahl an Datenformaten einlesen (es gibt dafür eine eigene Gebrauchsanweisung in der R-Hilfe: „R Data Import/Export“). Hier sollen uns vor allem zwei Themen kurz beschäftigen: (1) Wie kriege ich Daten *am einfachsten* in R importiert? und (2) Wie bekomme ich Daten von Excel nach R?

Der wichtigste Befehl für das Einlesen von Daten ist `read.table`. Er sieht z.B. so aus:
`> daten.name <- read.table("Daten.txt", sep="\t", dec=",", header=TRUE)` Diesen Befehl gibt es in verschiedenen Abwandlungen: `read.delim`, `read.csv` usw (siehe, in R: `apropos("read")`).

Zunächst muss man also R sagen, wo die Daten zu finden ist. Dann kann man noch einen Rattenschwanz an Optionen mitteilen, in diesem Fall, dass die Spalten durch Tabulatoren getrennt sind (Grundeinstellung: beliebig viele Leerzeichen; würde also auch hier funktionieren, solange keine Daten fehlen); dass die Kommazahlen durch ein Komma, nicht einen Punkt zu identifizieren sind (Grundeinstellung ist der Punkt); dass die erste Zeile der Datei die Namen der Spalten enthält.

Eine Reihe abgewandelter Befehle (etwa: `read.mtb`) im *package* **foreign** lässt es zu, Daten direkt aus anderen Statistikprogrammen zu importieren (z.B. SAS, SPSS, Minitab, Stata).[1]

Nun nutzen wir meist ein Programm wie MS Excel, um unsere Daten zu organisieren, ordnen, Werte zu mitteln, oder gar Abbildungen zu machen. Wie bekommen wir also die Daten von dort nach R? Natürlich könnten wir das zu Fuß machen: wir markieren die Daten und kopieren sie in einen einfachen Editor (unter Windows: Start → Programme → Zubehör

[1] Hier gibt es allerdings keine Importfunktion für Excel. Die verschiedentlich durch die R-Hilfe geisternden Excel-Importfunktionen haben bei uns nicht zwingend funktioniert.

→ Editor. Zu diesem Zweck **nicht** MS Word benutzen, da hier manche Zeichen ersetzt werden. Aus „"“ wird dann etwa „„“, ein Zeichen, dass es im ASCII- oder ANSI-Code nicht gibt!). Dort speichern wir dann die Daten als Textdatei (*.txt) ab. Jetzt können wir sie mit

```
> libelle <- read.table("c:/temp/libelle.txt", header = T)
```

einlesen.

Eine Stufe besser ist das Abspeichern der Daten als Textdatei in Excel selber (Datei → Speichern unter → Dateityp Text (Tabstopp-getrennt)(*.txt) auswählen). Wenn wir diese Datei dann im Editor öffnen, erleben wir bisweilen eine überraschung: Aus ungeklärten Gründen fügt Excel eine bis mehrere Zeilen unten an die Daten an. Diese müssen wir jetzt im Editor löschen. R versucht sonst diese Zeilen einzulesen, und findet heraus, dass es dort nur eine Spalte je Zeile gibt, woraufhin es eine Fehlermeldung gibt wie „`Line 23 did not have 5 elements.`“.

Ein direkter Zugriff auf Excel-Daten ist etwas schwieriger und lohnt sich vor allem bei sehr großen Datenmengen oder mehreren Tabellenblättern in einer Excel-Datei. Der im folgenden vorgestellte Weg geht über eine Datenbankabfrage und ist somit auch auf relationale Access oder Oracle Datenbanken übertragbar (siehe Dokumentation im von R mitgelieferten *manual* „R Data Import/Export“).

Zunächst müssen wir (1) ein *package* laden, (2) einen „Kommunikationskanal“ öffnen, (3) über diesen Kanal das Tabellenverzeichnis der Datenbank abfragen, bevor wir schließlich (4) auf die entsprechende Tabelle der Datenbank zugreifen können:

```
> library(RODBC)
> channel <- odbcConnectExcel("beispiel.xls")
> sqlTables(channel)
> gesuchte.Daten <- sqlQuery(channel, " select * from [Beispiel.Tabelle$]")
> odbcClose(channel)
```

Während des Zugriffs ist die Excel-Datei mit R verbunden und kann in Excel bearbeitet werden. Um diese Daten zu updaten ist eine Wiederholung des `sqlQuery`-Befehls nötig. Wird das Tabellenblatt in Excel geschlossen während die Verbindung besteht, so kann es nicht wieder in Excel zur Bearbeitung geöffnet werden. Dafür muss erst der Kanal wieder geschlossen werden (siehe Befehl `odbcClose(channel)` im obigen Beispiel).

Literaturverzeichnis

Anscombe, F. J. (1973). Graphs in statistical analysis. *American Statistician*, 27:17–21.

Baron, J. and Li, Y. (2003). *Notes on the Use of R for Psychology Experiments and Questionaires*. `http://www.r-project.org/other-docs.html`.

Berger, J. (1980). *Bayesian Statistics*. Springer, New York.

Berry, D. A. (1987). Logarithmic transformation in anova. *Biometrics*, 43:439–456.

Bolger, D. T., Alberts, A. C., Sauvajot, R. M., Potenza, P., McCalvin, C., Tran, D., Mazzoni, S., and Soulé, M. E. (1997). Response of rodents to habitat fragmentation on coastal southern california. *Ecological Applications*, 7:552–563.

Borcard, D., Legendre, P., and Drapeau, P. (1992). Partialling out the spatial component of ecological variation. *Ecology*, 73:1045–4055.

Bowerman, B. L. and O'Connell, R. T. (1990). *Linear Statistical Models: an Applied Approach*. Duxbury Press, California.

Box, G. E. P. and Cox, D. R. (1964). The analysis of transformations (with discussion). *Journal of the Royal Statistical Society B*, 26:211–252.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Burnham, K. P. and Anderson, D. R. (2002). *Model Selection and Multi-Model Inference: a Practical Information-Theoretical Approach*, volume 2nd. Springer, Berlin.

Chaterjee, S. and Price, B. (1991). *Regression Analysis by Example*. John Wiley & Sons, New York.

Chevan, A. and Sutherland, M. (1991). Hierarchical partitioning. *American Statistician*, 45:90–96.

Crawley, M. (1993). *GLIM for Ecologists*. Blackwell, Oxford. NOT filed.

Crawley, M. (2002). *Statistical Computing: An Introduction to Data Analysis using S-Plus*. Wiley, New York.

Dalgaard, P. (2002). *Introductory Statistics with R*. Springer, Berlin.

Day, R. W. and Quinn, G. P. (1989). Comparisons of trearments after an analysis of variance in ecology. *Ecological Monographs*, 59:433–4636.

Digby, P. G. N. and Kempton, R. A. (1987). *Multivariate Analysis of Ecological Communities*. Chapman & Hall, London.

Doden, K. (1991). Einführung in die Statistik für Naturwissenschaftler: Skript zur Vorlesung.

Dormann, C., Albon, S., and Woodin, S. (2002). No evidence for adaptation of two *Polygonum viviparum* genotypes to length of growing season: abundance, biomass and germination. *Polar Biology*, 25:884–890.

Dormann, C. F. and Roxburgh, S. H. (2005). Experimental evidence rejects classical modelling approach to coexistence in plant communities. *Proceedings of the Royal Society of London Series B*, 272:1279–1285.

Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*. Chapman & Hall, London.

Evans, M., Hastings, N., and Peacock, B. (2000). *Statistical Distributions*. John Wiley, Hoboken, N.J., 3rd edition.

Faraway, J. J. (2005). *Linear Models with R*. Chapman & Hall/CRC, Boca Raton.

Faraway, J. J. (2006). *Extending Linear Models with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. Chapman & Hall/CRC, Boca Raton.

Fielding, A. and Bell, J. (1997). A review of methods for the assessment of prediction errors in conservation presence/absence models. *Environmental Conservation*, 24:38–49.

Fisher, R. A. (1956). *Statistical Methods and Scientific Inference*. Oliver & Boyd, Edinburgh.

Flack, V. F. and Chang, P. C. (1987). Frequency of selecting noise variables in subset regression analysis: a simulation study. *American Statistician*, 41:84–86.

Ford, E. D. (2000). *Scientific Method for Ecological Research*. Cambridge Univ. Press, Cambridge.

Fox, J. (2002). *An R and S-Plus Companion to Applied Regression.* Sage, Thousand Oaks, 312 edition.

Gauch, H. G. (1982). *Multivariate Analysis in Community Ecology.* Cambridge University Press, Cambridge.

Gelman, A., Carlin, J. B., Stern, H. S., and Rubin, D. B. (1995). *Bayesian Data Analysis.* Chapman & Hall, New York.

Harrell, Frank E., J. (2001). *Regression Modeling Strategies - with Applications to Linear Models, Logistic Regression, and Survival Analysis.* Springer Series in Statistics. Springer, New York.

Hastie, T., Tibshirani, R., and Friedman, J. H. (2008). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, Berlin, 2nd edition.

Hastie, T. J. and Tibshirani, R. J. (1990). *Generalized additive models.* Chapman & Hall, London.

Hilborn, R. and Mangel, M. (1997). *The Ecological Detective: Confronting Models with Data*, volume 28 of *Monographs in Population Biology.* Princeton University Press, Princeton, N.Y.

Hurlbert, S. (1984). Pseudoreplication and the design of ecological experiments. *Ecological Monographs*, 54:187–211.

Inc., S. I. (1999). *SAS/STAT User's Guide, Release 8.1.* SAS Institute Inc., Cary, N.C.

James, F. C. and McCulloch, C. E. (1990). Multivariate analysis in ecology and systematics: Panacea or pandora's box? *Annual Review of Ecology and Systematics*, 21:129–166.

Johnson, D. H. (1999). The insignificance of statistical significance testing. *Journal of Wildlife Management*, 63:763–772.

Jongman, R. H. G., ter Braak, C. J. F., and van Tongeren, O. F. R. (1995). *Data Analysis in community and landscape ecology.* Cambridge University Press, Cambridge.

Legendre, P. and Legendre, L. (1998). *Numerical Ecology.* Elsevier Science, 2nd edition.

Lepš, J. and Smilauer, P. (2003). *Multivariate Analysis of Ecological Data using CANOCO.* Cambridge University Press, Cambridge.

Mac Nally, R. (2000). Regression and model-building in conservation biology, biogeography and ecology: The distinction between - and reconciliation of - 'predictive' and 'explanatory' models. *Biodiversity and Conservation*, 9:655–671.

Martz, H. and Waller, R. (1982). *Bayesian Reliability Analysis.* John Wiley & Sons, New York.

McCullough, P. and Nelder, J. A. (1989). *Generalized Linear Models.* Chapman & Hall, London, 2nd edition.

McCune, B., Grace, J. B., and Urban, D. L. (2002). *Analysis of Ecological Communities.* MjM Software.

Mead, R. (1988). *The Design of Experiments: Statistical Principles for Practical Application.* Cambridge University Press, Cambridge.

Menard, S. (2000). Coefficients of determination for multiple logistic regression. *American Statistician*, 54:17–24.

Oakes, M. (1986). *Statistical Inference: a Commentary for the Social and Behavioural Sciences.* Oxford Univ. Press, Oxford.

Palmer, M. W. (1993). Putting things in even better order: the advantages of canonical correspondence analysis. *Ecology*, 74:2215–2230.

Paruelo, J. M. and Lauenroth, W. K. (1996). Relative abundance of plant functional types in grassland and shrubland of north america. *Ecological Applications*, 6:1212–1224.

Pinheiro, J. and Bates, D. (2000). *Mixed-Effect Models in S and S-plus.* Springer, New York.

Post, B. J. (1986). Factors of influence on the development of an arable weed vegetation. *Proc. EWRS Symposium 1986, Economic Weed Control*, pages 317–325.

Potthoff, R. F. and Roy, S. N. (1964). A generalized multivariate analysis of variance model usefull especially for growth curve problems. *Biometrika*, 51:3131–326.

Potvin, C. (2001). Anova: experimental layout and analysis. In Scheiner, S. and Gurevitch, J., editors, *Design and Analysis of Ecological Experiments*, pages 63–76. Oxford University Press, Oxford, 2nd edition.

Quinn, G. P. and Keough, M. J. (2002). *Experimental Design and Data Analysis for Biologists.* Cambridge University Press, Cambridge.

Reineking, B. and Schröder, B. (2004a). Gütemaße für Habitatmodelle. In Dormann, C. F., Blaschke, T., Lausch, A., Schröder, B., and Söndgerath, D., editors, *Habitatmodelle - Methodik, Anwendung, Nutzen*, volume 9/2004, pages 27–38. UFZ-Bericht, Leipzig.

Reineking, B. and Schröder, B. (2004b). Variablenselektion. In Dormann, C. F., Blaschke, T., Lausch, A., Schröder, B., and Söndgerath, D., editors, *Habitatmodelle - Methodik, Anwendung, Nutzen*, volume 9/2004, pages 39–46. UFZ-Bericht, Leipzig.

Shaver, J. P. (1993). What statistical significance testing is, and what it is not. *Journal of Experimental Education*, 61:293–316.

Simberloff, D. (1983). Competition theory, hypothesis-testing, and other community ecological buzzwords. *American Naturalist*, 122:626–635.

Sokal, R. and Rohlf, F. (1995). *Biometry*. Freeman, New York, 3rd edition.

ter Braak, C. J. F. and Smilauer, P. (1998). *CANOCO reference manual and user's guide to canoco for windows: software for canonical community ordination (version 4)*. Microcomputer Power, Ithaca, New York.

Toms, J. D. and Lesperance, M. L. (2003). Piecewise regression: a tool for identifying ecological thresholds. *Ecology*, 84:2034–2041.

Underwood, A. (1997). *Experiments in Ecology: Their Logical Design and Interpretation using Analysis of Variance*. Cambridge University Press, Cambridge.

Väre, H., Ohtonen, R., and Oksanen, J. (1995). Effects of reindeer grazing on understorey vegetation in dry *Pinus sylvestris* forests. *Journal of Vegetation Science*, 6:523–530.

Venables, W. N. (2000). Exegeses on linear models. *S-PLUS User's Conference, Washington D.C.*, 1998.

Venables, W. N. and Ripley, B. D. (1994). *Modern Applied Statistics with S-plus*. Springer, New York.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, 4th edition.

Venables, W. N., Smith, D. M., and the R Development Core Team (2003). *In Introduction to R (Version 1.6.2)*. `http://cran.r-project.org/doc/manuals/R-intro.pdf`.

Verzani, J. (2002). *simpleR - Using R for Introductory Statistics*. `http://www.math.csi.cuni.edu/Statistics/R/simpleR`, version 0.4.

Wade, P. (2000). Bayesian methods in conservation biology. *Conservation Biology*, 14:1308–1316.

Werner, J. (1997). *Lineare Statistik*. BeltzPVU, Weinheim, 2nd edition.

Wood, S. N. (2006). *Generalized Additive Models*. Texts in Statistical Science. Chapman & Hall/CRC, New York.

Zar, J. H. (1996). *Biostatistical Analysis*. Prentice Hall, Upper Saddle River, New Jersey, 3rd edition.

Zuur, A. F., Ieno, E. N., and Smith, G. M. (2007). *Analysing Ecological Data*. Springer, Berlin.

Index