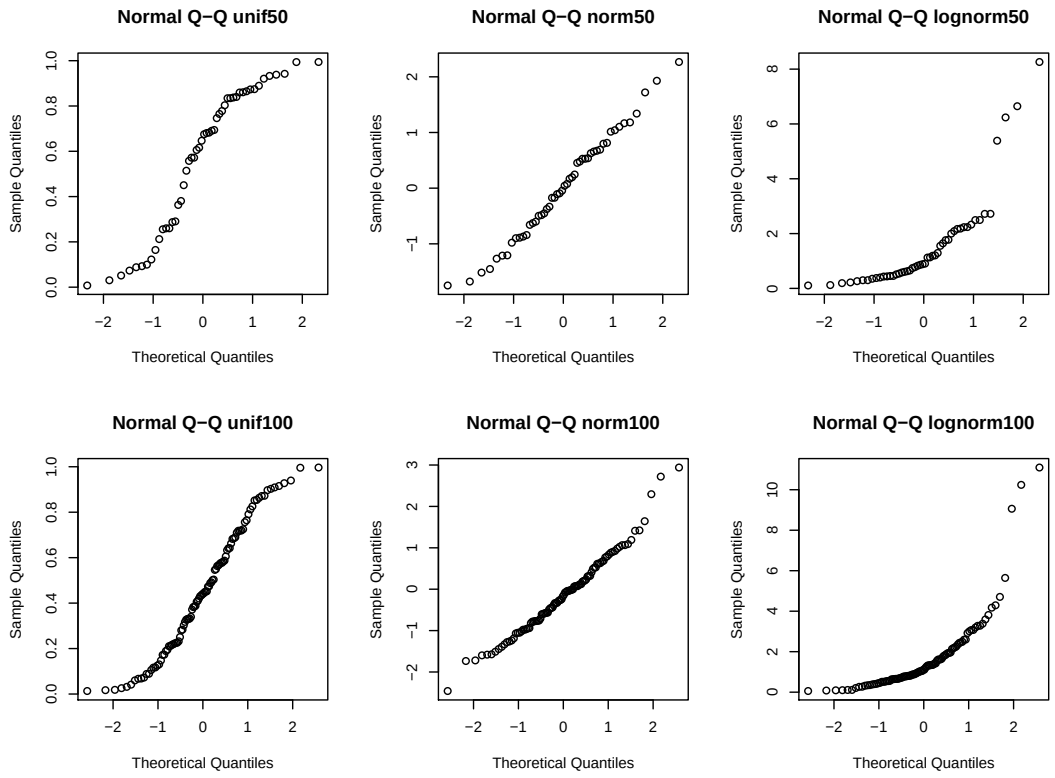
Beispiel 1.12:

Eingabe

```
oldpar <- par(mfrow = c(2, 3))
qqnorm(unif50, main = "Normal Q-Q unif50")
qqnorm(norm50, main = "Normal Q-Q norm50")
qqnorm(lognorm50, main = "Normal Q-Q lognorm50")
qqnorm(unif100, main = "Normal Q-Q unif100")
qqnorm(norm100, main = "Normal Q-Q norm100")
qqnorm(lognorm100, main = "Normal Q-Q lognorm100")
par(oldpar)
```



Aufgabe 1.23	
	Benutzen Sie P P-Plots anstelle von Verteilungsfunktionen, um die ^{2}- und Kolmogorov-Smirnov-Approximationen darzustellen.

Aufgabe 1.24	
	Benutzen Sie QQ-Plots anstelle von Verteilungsfunktionen. Konnen Sie in diesem Plot mit Hilfe der ^{2}- bzw. Kolmogorov-Smirnov-Stastistik Kon denzbereiche darstellen?

Um einen Eindruck uber die Fluktuation zu bekommen, mussen wir empirische Plots mit typischen Plots einer Modellverteilung vergleichen. Eine Plot-Matrix ist ein einfacher Weg dazu. Wir geben hier ein Beispiel fur den Normal-QQ-Plot, das wir gleich als Funktion implementieren:

Eingabe

```
qqnormx <- function(x, nrow = 5, ncol = 5, main = deparse(substitute(x))){
    oldpar <- par(mfrow = c(nrow, ncol))
    qqnorm(x, main = main)
    for (i in 1:(nrow*ncol-1)) qqnorm(rnorm(length(x)), main = "N(0, 1)")
    par(oldpar)
}
```

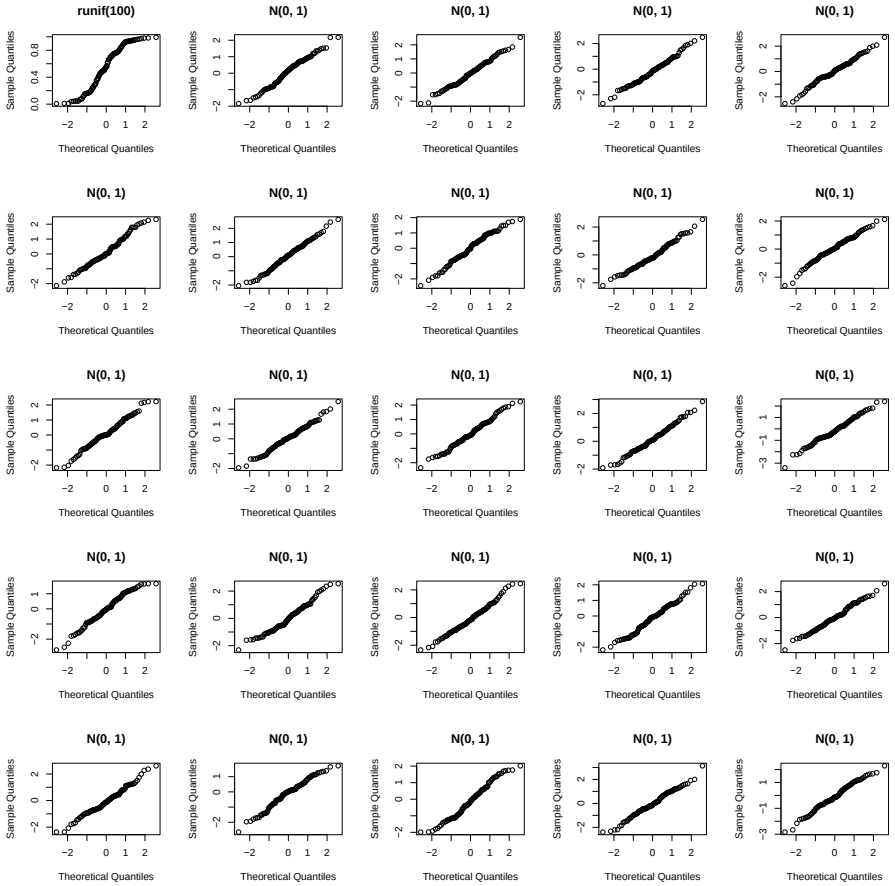
Wir haben in diesem Beispiel eine for -Schleife benutzt. Wie alle Programmiersprachen hat R Kontrollstrukuren, wie bedingte Anweisungen und Schleifen. Eine Ubersicht uber die Kontrollstrukturen in R ist im Anhang A.14 zu nden.

Abweichung von einer linearen Struktur ist als Fluktuation zu betrachten, wenn sie im Rahmen der simulierten Beispiele bleibt. Ist der zu untersuchende Datensatz extrem im Vergleich zu den simulierten Beispiele, so widerspricht das der Modellverteilung.

Beispiel 1.13:

Eingabe

```
qqnormx(runif(100))
```



Auf lange Sicht lohnt es sich, die Plot-Funktionen so zu modi zieren, dass sie auch Informationen uber die zu erwartende Fluktuation wieder geben. In Beispiel 1.9 haben wir fur die Verteilungsfunktion Monte-Carlo-Bander konstruiert. Wir konnen diese Idee auf den PP-Plot und den QQ-Plot ubertragen. Dazu ist es nur notwendig, die Bander jeweils in der fur den Plot geeigneten Skala darzustellen.

Aufgabe 1.25	
	Erzeugen Sie sich mit rnorm() Pseudozufallszahlen fur die Gau verteilung zum Stichprobenumfang $n = 10; 20; 50; 100$. Erzeugen Sie jeweils einen PP-Plot und einen QQ-Plot, wobei die theoretische Gau verteilung als Bezug dient.
	Fugen Sie Monte-Carlo-Bander aus der Einhullenden von 19 Simulationen hinzu. Sie mussen zunachst anstelle der uniformen Verteilung die Normalverteilung zur Erzeugung der Monte-Carlo-Bander benutzen. Sie mussen au erdem die Resultate im Koordinatensystem des QQ-Plots darstellen, d.h. die x-Achse reprasentiert die Quantile der Normalverteilung. Hinweis: inspizieren Sie dazu die Quelle von qqnorm() .
*	Die Bander sind zunachst Bander fur die Standard-Normalverteilung. Finden Sie Bander fur die vorliegenden Daten.

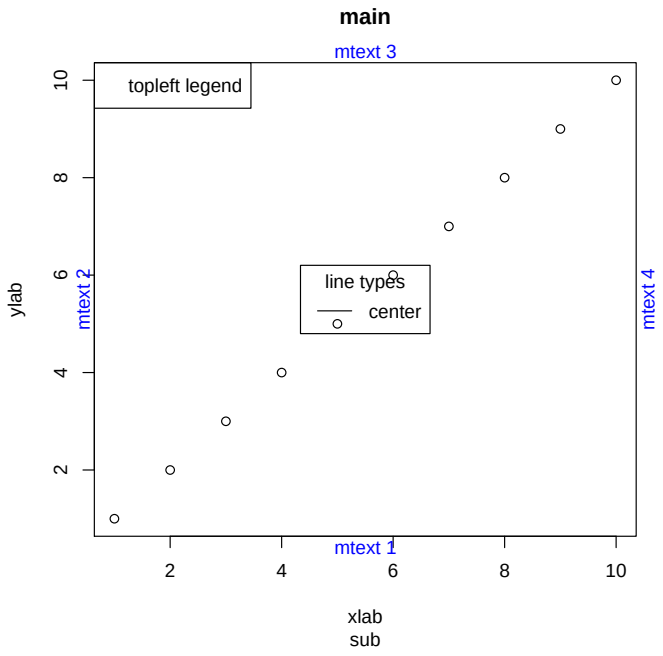
1.5.3. Erg anzung: Gra k-Aufbereitung. Bislang wurde die R-Gra k in rudimentarer Form benutzt. Fur ernsthafte Arbeit muss die Gra k so aufbereitet werden, dass ihre Bestandteile identi ziert und wiedererkennbar sind. Dazu gehoren Uberschriftungen, Achsenkennzeichnungen etc. R unterscheidet zwischen \high level"-Gra k und \low level". \High level"-Funktionen erzeugen eine neue Gra k. Sie bieten daruber hinaus Moglichkeiten, allgemeine Gra kparameter zu steuern.

Die \low level"-Funktionen fugen Elemente zu vorhandenen Gra ken hinzu oder modi zieren die Gra k im Detail. Die Funktion legend() zum Beispiel kann Legenden innerhalb der Graphik hinzu fugen.

Beispiel 1.14:

Eingabe

```
plot(1:10, xlab = "xlab", ylab = "ylab", main = "main", sub = "sub")
mtext("mtext 1", side = 1, col = "blue")
mtext("mtext 2", side  = 2, col = "blue")
mtext("mtext 3", side  = 3, col = "blue")
mtext("mtext 4", side  = 4, col = "blue")
legend("topleft", legend = "topleft legend")
legend("center", legend = "center" , lty =  1:4, title = "line types")
```



Aufgabe 1.26	
	Inspizieren Sie mit help(plot) die Steuerungsmoglichkeiten der plot-Funktion. Einige Detail-Information zu den Parametern erhalten Sie erst in help(plot.default) . Korrigieren Sie Ihren letzten Plot so, dass er eine korrekte Uberschrift tragt.

Weitere Hinweise: [R D07a] Ch. 12.

1.5.4. Erg anzung: Funktionen. R-Kommandos konnen zu Funktionen zusammengefasst werden. Funktionen konnen parametrisiert sein. Funktionen erlauben eine exible Wiederverwendbarkeit.

Beispiel fur eine Funktion

Beispiel 1.15:

Eingabe

```
ppdemo <- function (x, samps = 19) {
    # samps: nr of simulations

    y <- (1:length(x))/length(x)
    plot(sort(x), y, xlab = substitute(x), ylab = expression(F[n]),
        main = "Verteilungsfunktion mit Monte-Carlo-Band (unif.)",
        type = "s")
    mtext(paste(samps, "Monte-Carlo-Stichproben"), side = 3)
    samples <- matrix(runif(length(x)* samps), nrow = length(x), ncol = samps)
    samples <- apply(samples, 2, sort)
    envelope <- t(apply(samples, 1, range))
    lines(envelope[, 1], y, type = "s", col = "red");
    lines(envelope[, 2], y, type = "s", col = "red")
}
```

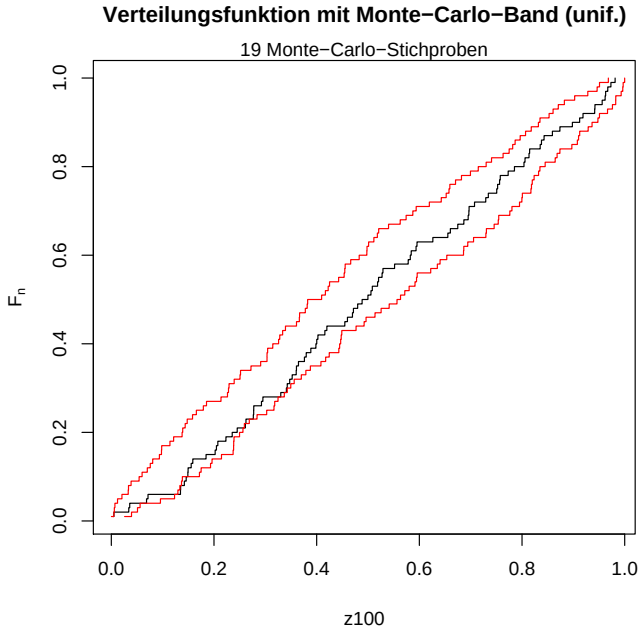
Wir haben bei ppdemo() die Funktion mtext() benutzt, die Randbeschriftungen erlaubt.

Funktionen werden in der Form `Name(Aktuelle Parameterliste)` aufgerufen.

Beispiel 1.16:

Eingabe

```
z100 <- runif(100)
ppdemo(z100)
```



Wird nur der Name der Funktion eingegeben, so wird die De nition der Funktion zuruck gegeben, d.h. die Funktion wird aufgelistet. Beispiel:

Beispiel 1.17:

Eingabe

```
ppdemo
```

Ausgabe

```
function (x, samps  =  19) {
    # samps: nr of simulations

    y <- (1:length(x))/length(x)
    plot(sort(x), y, xlab = substitute(x), ylab = expression(F[n]),
        main = "Verteilungsfunktion mit Monte-Carlo-Band (unif.)",
        type = "s")
    mtext(paste(samps, "Monte-Carlo-Stichproben"), side = 3)
    samples <- matrix(runif(length(x)* samps), nrow = length(x), ncol = samps)
    samples <- apply(samples, 2, sort)
    envelope <- t(apply(samples, 1, range))
    lines(envelope[, 1], y, type = "s", col = "red");
    lines(envelope[, 2], y, type = "s", col = "red")
}
```

Aufgabe 1.27	
	Inspizieren Sierunif() mit qqplot() und plot() . Uberarbeiten Sie Ihre bisherigen Programmieraufgaben und schreiben Sie die wiederverwendbaren Teile als Funktionen.

Parameter bei Funktionen werden dem Wert nachubergeben. Jede Funktion erhalt eine Kopie der aktuellen Parameterwerte. Dies sorgt fur eine sichere Programmierumgebung. Auf der anderen Seite fuhrt dies zu eine Speicherbelastung und bringt einen Zeitverlust mit sich. In Situationen, wo der Parameterumfang gro ist oder die Zeit eine kritische Gro e ist, kann dieser Aufwand vermieden werden, indem direkt auf Variable zugegri en wird, die in der Umgebung der Funktion de niert sind. Entsprechende Techniken sind in [GI00] beschrieben.

Funktionen in R konnen auch geschachtelt sein, d. h. innerhalb einer Funktion konne auch wieder Funktionen de niert werden. Diese sind nur innerhalb der umgebenden Funktion sichtbar.

Funktionen konnen Objekte als Resultate haben. Ein Objekt wird explizit als Resultat ubergeben mitreturn(obj) . Das Resultat kann auch implizit ubergeben werden: wird das Ende einer Funktion erreicht, ohne dassreturn() aufgerufen wurden, so wird der Wert des letzten ausgewerteten Ausdruckubergeben.

Eingabe

```
circlearea <- function( r) r^2 * pi
circlearea(1:4)
```

Ausgabe

```
[1]  3.141593 12.566371 28.274334 50.265482
```

Resultate konnen auch bereit gestellt werde, so dass sie nur auf Anfrage ubergeben werden. Wir haben diese Technik beim Histogramm kennen gelernt. Der Aufruf hist(x) ubergibt kein Resultat, sondern hat nur den (gewunschten) Seitene ekt, ein Histogramm zu zeichnen. Benutzen wir hist() jedoch in einem Ausdruck, zum Beispiel in einer Zuweisung xhist <- hist(x) , so erhalten wir als Wert die Beschreibung des Histogramms. Um Resultate nur bei Bedarf zu ubergeben, wird anstelle return(obj) der Aufruf invisible(obj) benutzt.

Aufgabe 1.28	
	Schreiben Sie als Funktionen: Eine Funktion ehist , die ein Histogramm mit Erganzungen zeigt. Eine Funktion eecdf , die die empirische Verteilung zeigt. Eine Funkion eqqnorm, die einen QQ-Plot mit der Standard-Normalverteilung vergleicht. Eine Funkion eboxplot , die einen Box&Whisker-Plot zeigt. Eine zusammenfasende Funktioneplot , die eine Plot-Matrix mit diesen vier Plots zeigt. Ihre Funktionen sollten die Standardfunktionen so aufrufen (oder modi zieren, falls notwendig), dass die Plots eine angemessene Beschriftung erhalten.

Wahrend Anweisungen in R schrittweise ausgefuhrt werden und so die Resultate bei jedem Schritt inspiziert werden konnen, werden beim Aufruf einer Funktion alle Anweisungen in der Funktion als Einheit ausgefuhrt. Dies kann eine Fehlersuche schwierig machen. R bietet Moglichkeiten, die Inspektion gezielt auf Funktionen zu ermoglichen. Details dazu nden sich im Anhang A.13 \Debugging und Pro ling" auf Seite Seite A-21.

1.5.5. Erg anzung: Das Innere von R. Ein typischer Arbeitsabschnitt von R verarbeitet Kommandos in drei Teilschritten:

parse() analysiert einen Eingabetext und wandelt ihn in ein interne Darstellung als R-Ausdruck. R-Ausdrucke sind spezielle R-Objekte.
eval() interpretiert diesen Ausdruck und wertet ihn aus. Das Resultat ist wieder ein R-Objekt.
print() zeigt das resultierende Objekt.

Details sind zu erganzen:

1.5.5.1. *Parse.* Der erste Schritt besteht aus zwei Teilen: einem Leseprozess, der die Eingabe einscannt und in Bausteine (Tokens) zerlegt, und dem eigentlichen Parsing, das die Bausteine falls moglich zu einem syntaktisch korrekten Ausdruck zusammenfasst. Die Funktion parse() fasst beide Schritte zusammen. Dabei kann parse() sowohl auf lokalen Dateien arbeiten, als auch auf externen, durch eine URL-Referenz bezeichneten Dateien.

Als inverse Funktion steht substitute() zur Verfugung. Eine typische Anwendung ist es, aktuelle Parameter eines Funktionsaufrufs zu entschlusseln und informative Beschriftungen zu erzeugen.

1.5.5.2. *Eval.* Die Funktion eval() wertet einen R-Ausdruck aus. Dazu mussen die Referenzen im Ausdruck je nach den aktuellen Umgebungsbedingungen in entsprechende Werte ubersetzt werden. Da R ein interpretiertes System ist, konnen die Umgebungsbedingungen variieren; je nach Umgebung kann derselbe Ausdruck zu unterschiedlichen Resultaten fuhren.

Jede Funktion de niert eine eigene lokale Umgebung. Funktionen konnen geschachtelt sein und somit auch die Umgebungen. Die Umgebung kann auch dadurch verandert werden, dass Zusatzpakete fur R geladen werden. Die aktuelle unmittelbare Umgebung kann mit environment() erfragt werden. Mit search() erhalt man eine Liste der Umgebungen, die sukzessive durchsucht werden, um Referenzen aufzulosen. Mit ls() erhalt man eine Liste der Objekte in einer Umgebung.

Die Erweiterbarkeit von R bringt die Moglichkeit mit sich, dass Bezeichnungen kollidieren und damit die Ubersetzung von Referenzen in aktuelle Werte fraglich wird. Als Schutz dagegen bietet R 2.x die Moglichkeit, Bezeichnungen in (geschutzten) Namensraumen zusammen zu fassen. In den meisten Fallen ist dies transparent fur den Benutzer; die Au osung von Namen folgt der Suchreihenfolge, die durch die Kette der Umgebungen bestimmt ist. Um explizit auf Objekte eines bestimmten Namensraums zuzugreifen, kann dieser mit angegeben werden (z. B. base::pi als expliziter Name fur die Konstante pi im Namensraum base).

1.5.5.3. *Print.* Die Funktion print() ist als polymorphe Funktion implementiert. Um print() auszufuhren bestimmt R anhand der Klasse des zu druckenden Objekts eine geeignete Methode. Details folgen spater in Abschnitt 2.6.5 Seite 2-39.

1.5.5.4. *Ausführung von Dateien.* Die Funktion source() steht bereit, um eine Datei als Eingabe fur R zu benutzen. Dabei kann die Datei lokal sein, oder uber eine URL-Referenz bezeichnet sein. Konventionell wird fur die Namen von R-Kommandodateien die Endung .R benutzt.

Die Funktion Sweave() erlaubt es, Dokumentation und Kommandos miteinander zu verweben. Konventionell wird fur die Namen von Sweave()-Eingabedateien die Endung .Rnw benutzt. Details zum Format nden sich in der Sweave()-Dokumentation <http://www.ci.tuwien.ac.at/~leisch/Sweave/Sweave-manual-20060104.pdf>.

1.5.6. Erg anzung: Pakete. Funktionen, Beispiele, Datensatze etc. konnen in R zu Paketen zusammengefasst werden, die bestimmten Konventionen entsprechen. Die Konventionen unterscheiden sich bei verschiedenen Implementierungen. Als aktuelle Referenz sollten die Konventionen von R [R D08] benutzt werden, denen wir auch hier folgen. Eine Reihe von Paketen sind Standardbestandteil von R. Pakete fur spezielle Zwecke nden man im Internet z.B. uber http://www.cran.r-project.org/src/contrib/PACKAGES.html.

Nicht-Standard-Pakete mussen zunachst im R-System installiert werden. In der Regel gibt es dazu betriebssystem-spezi sche Kommandos. Komfortabler ist jedoch die Installation aus R mit der Funktionen install.packages() . Ist keine spezielle Quelle angegeben, so greift install.packages dabei auf eine vorbereitete Adresse (in der Regel die oben angegebene) zuruck. Sie konnen Pakete jedoch von jedem beliebigen Speicher laden. Insbesondere kann mit install.packages(hpackagei , repos = NULL) unter hpackageiein direkter Zugri spfad auf Ihrem Rechner angegeben werden.

Die Funktion update.packages() vergleicht installierte Versionen mit dem aktuellen Stand im Netz und frischt gegebenenfalls die installierte Version auf.

Installierte Pakete werden mit

```
library(pkgname)
```

geladen. Danach sind die im Paket definierten Objekte (Funktionen, Datensätze, ...) über den aktuellen Suchpfad ansprechbar und direkt verwendbar.

Pakete werden mit

```
detach(pkgname)
```

wieder frei gegeben, d.h. ihre Objekte erscheinen nicht mehr im Suchpfad.

Technisch sind Pakete Verzeichnisse, die den R-Konventionen folgen. Üblich liegen sie in gepackter Form als .tar.gz-Files vor. In der Regel wird man zunächst als Benutzer vorbereitete Binärpakete installieren. Nur selten muss man auf die Quellpakete anderer Entwickler zurückgreifen.

Bei der Organisation der eigenen Arbeit lohnt es sich, den R-Konventionen zu folgen und zusammen gehörende Teile als R-Pakete zu organisieren. Dann stellt R eine ganze Reihe von Werkzeugen zur Unterstützung bereit. Die Konventionen und die bereitgestellten Werkzeuge sind in [R D08] dokumentiert. Für Unix/Linux/Mac OS X-Benutzer sind die wichtigsten Werkzeuge als Kommandos verfügbar:

```
R CMD check <directory> # überprüft ein Verzeichnis
R CMD build <directory> # generiert ein R-Paket
```

Als Einstieg: Die Funktion package.skeleton() hilft bei der Konstruktion neuer Pakete. package.skeleton() erzeugt dabei aus einem vorbereiteten Paket eine Hilfsdatei, die die weiteren Schritte zur Erzeugung eines ladbaren Pakets beschreibt.

Pakete müssen eine Datei DESCRIPTION mit bestimmter Information enthalten. Die Details sind in [R D08] beschrieben, und ein Prototyp wird von package.skeleton() erzeugt. Weiteres ist optional.

Name	Art	Inhalt
DESCRIPTION	Datei	eine Herkunftsbeschreibung nach Formatkonventionen.
R	Verzeichnis	R code. Dateien in diesem Verzeichnis sollten mit source() gelesen werden können. Empfohlene Namensendung: .R .
data	Verzeichnis	Zusätzliche Daten. Dateien in diesem Verzeichnis sollten mit data() gelesen werden können. Empfohlene Namensendungen und Formate: .R für R-Code. Alternativ: .r .tab für Tabellen. Alternativ: .txt, .csv .RData für Ausgaben von save() . Alternativ: .rda. Das Verzeichnis sollte eine Datei 00Index mit einer Übersicht über die Datensätze enthalten.

(Fortsetzung)!

Name	Art	Inhalt
exec	Verzeichnis	Zusatzliche ausfuhrbare Dateien, z.B. Perl- oder Shell-Skripte.
inst	Verzeichnis	Wird (rekursiv) in das Zielverzeichnis kopiert. Dieses Verzeichnis kann insbesondere eine Datei CITATION enthalten, die in R mit einer Funktion citation() ausgewertet wird.
man	Verzeichnis	Dokumentation im R-Dokumentationsformat (siehe: [R D08] \Writing R extensions", zuganglich uber <http://www.cran.r-project.org/>). Empfohlene Namensendung: .Rd
src	Verzeichnis	Fortran, C und andere Quellen.
demo	Verzeichnis	ausfuhrbare Beispiele. Dieses Verzeichnung sollte in einer Datei 00Index eine Beschreibung enthalten.

Aufgabe 1.29	
	Installieren Sie die Funktionen der letzten Aufgaben als Paket. Das Paket sollte enthalten: Eine Funktion ehist , die ein Histogramm mit Erganzungen zeigt. Eine Funktion eecdf , die die empirische Verteilung zeigt. Eine Funkion eqqnorm, die einen QQ-Plot mit der Standard-Normalverteilung vergleicht Eine Funkion eboxplot , die einen Box&Whisker-Plot zeigt. Eine zusammenfassende Funktioneplot , die eine Plot-Matrix mit diesen vier Plots zeigt. Sie konnen das Paket mit package.skeleton() vorbereiten, wenn Sie die einzelne Funktionen de niert haben. Laden Sie dieses PaketUberprufen Sie, ob Sie das Paket auch nach Neustart wieder mit library() laden konnen. *Hinweis:* ist x ein Objekt, so erzeugt die Funktion prompt(x) ein Gerust, aus dem eine Dokumentation fur x entwickelt werden kann.

1.6. Statistische Zusammenfassung

Als Leitbeispiel diente in diesem Kapitel die statistische Analyse einer (univariaten) Stichprobe. Dabei haben wir eine in der Statistik zentrale Modellvorstellung benutzt: die Werte der Stichprobe werden als Zufallsvariable aufgefasst, die aus einer zugrundeliegenden theoretischen Verteilung entstammen. Ziel der statistischen Analyse ist der Schluss aus der empirischen Verteilung der Stichprobe auf die unbekannte zu Grunde liegende theoretische Verteilung. Dieser Schluss kann zwei Formen annehmen: wirdkonnen die empirische

Verteilung mit einer hypothetischen Verteilung vergleichen. Dies ist das Vorgehen der klassischen Statistik. Oder wir konnen versuchen, aus der empirischen Verteilung Merkmale der zu Grunde liegenden Verteilung zu extrahieren. Dies ist das Vorgehen der Datenanalyse.

Beide Wege sind eng miteinander verwandt. Das wesentliche Werkzeug für beide war hier die Untersuchung der empirischen Verteilungsfunktion.

1.7. Literatur und weitere Hinweise:

[R D08]R Development Core Team (2000-2005): Writing R extensions. Siehe:<http://www.r-project.org/manuals.html>.

[GS77] Gan ler, P; Stute, W.: Wahrscheinlichkeitstheorie. Heidelberg: Springer 1977.

[GI00] Gentleman, R.; Ihaka, R.: Lexical Scope and Statistical Computing. Journal of Computational and Graphical Statistics 9 (2000) 491{508.

KAPITEL 2

Regression

2.1. Allgemeines Regressionsmodell

Aus der Tradition experimenteller Wissenschaften stammt das Paradigma des (kontrollierten) Versuchs. Unter Versuchsbedingungen x wird ein Resultat y gemessen, zusammengesetzt aus einem systematischen Effekt $m(x)$ und einem Messfehler ε.

$$y = m(x) + \varepsilon.$$

Dies ist eine ganz spezielle Betrachtungsweise; es wird nicht unvoreingenommen das gemeinsame Verhalten von x und y untersucht, sondern eine Unsymmetrie hineingesteckt: x ist die "Ursache", y (oder eine Veränderung von y) der Effekt. Die "Ursache" x ist in dieser Vorstellung vorgegeben oder direkt kontrollierbar; y ist mittelbar (über die Versuchsbedingungen) beeinflusst. In dieser Vorstellung ist ε ein Messfehler, der durch geeignete Rahmenbedingungen des Versuchs möglichst klein gehalten wird und im Mittel verschwinden sollte: es sollte keinen systematischen Fehler geben.

Vom statistischen Standpunkt ist der wesentliche Unterschied der Rollen von x und y, dass für y das stochastische Verhalten mithilfe von ε modelliert wird, während x als "gegeben" angenommen wird und dafür keine Stochastik im Modell vorgesehen ist.

Um einen formal überschaubaren Rahmen zu bekommen, betrachten wir den Fall, dass x als Vektor von reellen Variablen repräsentiert werden kann, $x \in \mathbb{R}^p$, und dass die Messwerte eindimensionale reelle Werte sind, $y \in \mathbb{R}$. In einem stochastischen Modell kann die oben skizzierte Idee formal gefasst werden. Eine mögliche Formalisierung ist es, den Messfehler ε als Zufallsvariable zu modellieren. Nehmen wir ferner an, dass der Erwartungswert von ε existiert, so können wir die Annahme, dass der Messfehler im Mittel verschwindet, formalisieren als $E(\varepsilon) = 0$.

Um den systematischen Effekt m zu untersuchen, betrachten wir Messreihen. Der Index $i; i = 1; \ldots; n$, kennzeichnet den Messpunkt, und das Modell ist damit

$$\begin{aligned} y_i &= m(x_i) + \varepsilon_i \quad i = 1; \ldots; n \\ \text{mit} \quad & x_i \in \mathbb{R}^p \\ & E(\varepsilon_i) = 0: \end{aligned}$$

Das statistische Problem ist:
schätze die Funktion m aus den Messwerten y_i bei Messbedingungen x_i.

Zu diesem Problem der Kurvenschätzung oder "Regression" gibt es eine umfangreiche Literatur in der Statistik. Wir wollen uns hier auf das "computing" konzentrieren. Dazu betrachten wir zunächst eine sehr vereinfachte Version des Regressionsproblems, die lineare Regression. Wesentliche Aspekte lassen sich bereits an diesem Problem illustrieren.

Einer einheitlichen Sprechweise zuliebe nennen wir y_i die Response und die Komponenten von x_{ij} mit $j = 1; \ldots; p$ die Regressoren. Die Funktion m heißt die Modellfunktion.

2.2. Lineares Model

Wir beginnen mit dem Regressionsmodell - jetzt in Vektorschreibweise[1] -

$$\begin{aligned} Y &= m(X) + \varepsilon \\ &\quad Y \text{ mit Werten in } \mathbb{R}^n \\ &\quad X \in \mathbb{R}^{n \times p} \\ &\quad E(\varepsilon) = 0 \end{aligned} \tag{2.1}$$

und setzen zusätzlich voraus, dass m linear ist. Dann gibt es (mindestens) einen Vektor $\beta \in \mathbb{R}^p$, so dass

$$m(X) = X\beta$$

und das Regressionsproblem ist jetzt reduziert auf die Aufgabe, β aus der Information $(Y; X)$ zu schätzen.

Das so modifizierte Regressionsmodell

$$\begin{aligned} Y &= X\beta + \varepsilon \\ &\quad Y \text{ mit Werten in } \mathbb{R}^n \\ &\quad X \in \mathbb{R}^{n \times p} \\ &\quad \beta \in \mathbb{R}^p \\ &\quad E(\varepsilon) = 0 \end{aligned} \tag{2.2}$$

heißt **lineares Modell** oder auch **lineare Regression**. Die Matrix X, in der die Werte der Regressoren zusammengefasst ist, also die Information über die Versuchsbedingungen, heißt **Design-Matrix** des Modells.

Beispiel 2.1. *(Einfache lineare Regression)* Wird die Versuchsbedingung durch den Wert einer reellen Variablen beschrieben, von der der Versuchsausgang über eine lineare Modellfunktion

$$m(x) = a + b\,x;$$

abhängt, so können wir eine Versuchsserie mit Versuchsausgängen y_i bei Versuchsbedingungen x_i koordinatenweise schreiben als

$$y_i = a + b\,x_i + \varepsilon_i: \tag{2.3}$$

In Matrixschreibweise können wir die Versuchsserie zusammenfassen als lineares Modell

$$Y = \underbrace{\begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}}_{X} \begin{pmatrix} a \\ b \end{pmatrix} + \varepsilon: \tag{2.4}$$

Die einfache lineare Regression ist ein Beispiel für lineare Modelle. Die Einweg-Klassifikation ist das andere Basis-Beispiel.

Beispiel 2.2. *(Einweg-Klassifikation)* Zum Vergleich von k Behandlungen (insbesondere für den Spezialfall $k = 2$) benutzen wir Indikatorvariablen, die in einer Matrix zusammengefasst werden. Die Indikatorvariable für Behandlung i steht in Spalte i. In der Regel

[1] Wir wechseln Konventionen und Schreibweisen, wenn es hilfreich ist. Die Verwirrung gehört zu den Konventionen: in einigen Konventionen kennzeichnen Großbuchstaben Zufallsvariable, in anderen Funktionen, in wieder anderen Vektoren. Die Auflösung bleibt jeweils dem Leser überlassen.

haben wir wiederholte Beobachtungen $j = 1; \ldots; n_i$ unter Behandlung i, insgesamt also $n = \sum_{i=1}^{k} n_i$ Beobachtungen. Dem Modell

$$Y = \underbrace{\begin{pmatrix} 1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}}_{X} \begin{pmatrix} \mu \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_k \end{pmatrix} + \varepsilon: \tag{2.5}$$

entspricht in Koordinaten

$$y_{ij} = \mu + \alpha_i + \varepsilon_{ij}: \tag{2.6}$$

Dies ist das typische Modell, um die Hypothese \kein Unterschied" $\alpha_1 = \ldots = \alpha_k$ gegen die Alternative zu testen, dass sich die Behandlungen im Mittel unterscheiden.

Der selbe Zusammenhang kann auch dargestellt werden, wenn wir die Messwerte als Summe eines Grundwertes μ_0 und dazu eines Behandlungse ekts $\alpha_i^0 = \mu_i - \mu_0$ interpretieren. Dies entspricht in Koordinaten

$$y_{ij} = \mu_0 + \alpha_i^0 + \varepsilon_{ij}: \tag{2.7}$$

In Matrixschreibweise ist dies

$$Y = \underbrace{\begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 0 & 1 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 0 & 0 & \cdots & 1 \end{pmatrix}}_{X^0} \begin{pmatrix} \mu_0 \\ \alpha_1^0 \\ \alpha_2^0 \\ \vdots \\ \alpha_k^0 \end{pmatrix} + \varepsilon: \tag{2.8}$$

Beispiel 2.2 illustriert, dass die Darstellung eines Problems als lineares Modell nicht eindeutig ist. (2.5) und (2.8) sind gleichwertige Darstellungen und nur aus der Anwendung kann entschieden werden, welche den Vorrang hat.

Fur die mathematische Analyse ist die Design-Matrix X ein wesentliches Hilfsmittel. Fur die Datenanalyse konnen wir R zu Hilfe nehmen, um diese Matrix (implizit) f ur uns zu erstellen. R versteht eine spezielle Notation, die Wilkinson-Rogers-Notation , mit der Modelle beschrieben werden konnen. In dieser Notation schreiben wir

$$y \sim x:$$

Der Fehlerterm wird in diesem Modell nicht notiert.

Der konstante Term wird implizit angenommen. Fur die Einweg-Klassi kation erhalten wir also das Modell (2.8). Wenn wir keinen konstanten Term wollen (also bei der Regression die Regressionsgerade durch den Ursprung geht, bzw. bei der Einweg-Klassi kation das Modell (2.5) benutzt und kein Gesamtmittel vorgegeben sein soll), so haben wir in Koordinatenschreibweise

$$y_i = b\ x_i + \varepsilon_i:$$

In der Wilkinson-Rogers-Notation muss der konstante Term explizit auf Null gesetzt werden:

$$y \quad 0 + x:$$

Weitere Regressoren konnen mit dem Operator + gekennzeichnet werden. So entspricht $y \quad u + v$ in Koordinaten dem Modell

$$y_i = a + b\ u_i + c\ v_i + \varepsilon_i:$$

Wir kommen in den Abschnitten 2.2.4 und 2.3 noch auf diese Notation zuruck.

Es gibt eine umfangreiche Literatur uber lineare Modelle. Das Buch \The Theory of Linear Models" von Bent J rgensen [J r93] ist besonders zu empfehlen. Es deckt den mathematischen Hintergrund dieses Kapitels weitgehend ab und enthalt zahlreiche illustrierende Beispiele.

2.2.1. Faktoren. Mit Hilfe der Notation zur Design- und Modellbeschreibung kann die Ubersetzung zwischen einer fallorientierten Beschreibung eines Designs in eine Design-Matrix f ur ein lineares Modell in kanonischer Form automatisch geschehen. Bisweilen braucht die Ubersetzung etwas Nachhilfe. Betrachten Sie z.B. einen Datensatz

```
y <- c( 1.1, 1.2, 2.4, 2.3, 1.8, 1.9)
x <- c( 1, 1, 2, 2, 3, 3).
```

Der Vektor x kann als quantitativer Vektor f ur das Regressions-Modell

$$y_i = a + b\ x_i + \varepsilon_i$$

als Regressor gemeint sein, oder es kann in der Einweg-Klassi kation, dem Modell der Einweg-Varianzanalyse,

$$y_{ix} = + _x + \varepsilon_{ix}$$

die Kennzeichnung einer Behandlungsgruppe sein. Um beide Moglichkeiten zu unterscheiden, konnen Vektoren in R als Faktoren de niert werden. Vektoren, die keine Faktoren sind, werden als quantitative Variable behandelt wie im ersten Beispiel. Faktoren werden als Kennzeichner behandelt und in der Design-Matrix in entsprechende Indikatorvariable ubersetzt. So ergibt

$$y \quad x$$

das Regressionsmodell, jedoch

$$y \quad \text{factor(x)}$$

das Varianz-Modell fur das Einweg-Layout.

Durch einen Parameter ordered = TRUE kann beim Aufruf der Funktion factor() die erzeugte Variable als geordnet gekennzeichnet werden. Die erzeugte Variable wird dann bei den Auswertungen als ordinal skaliert behandelt.

$$y \quad \text{factor(x, ordered = TRUE)}$$

Ohne diese Kennzeichnung werden Faktoren als kategorial skaliert betrachtet.

Die Zahlenwerte von Faktoren brauchen keine aufsteigende Folge zu sein. Sie werden (auch fur ordinale Faktoren) als blo e Namen benutzt und durch eine laufende Nummer ersetzt. So ergibt

factor(c(2, 2, 5, 5, 4, 4))

einen Vektor mit drei Faktorwerten 1, 2, 3, die die Namen \2", \5" und \4" haben. Faktoren konnen auch durch Namen bezeichnet werden, z.B.

y factor (c("Beh1", "Beh1", "Beh2", "Beh2", "Beh3", "Beh3"))

Die unterschiedlichen Werte eines Faktors nennt man Stufen des Faktors. Sie konnen mit levels() erfragt werden, z.B.

levels(factor(c(2, 2, 5, 5, 4, 4)))

levels(factor(c("Beh1", "Beh1", "Beh2", "Beh2", "Beh3", "Beh3")))

2.2.2. Kleinste-Quadrate-Sch atzung. Eine erste Idee zur Sch atzung im linearen Regressionsmodell kann so gewonnen werden: Bei gegebenem X ist $E(Y) = X$, also $X^> E(Y) = X^> X$ und damit $(X^> X) X^> E(Y) =$. Dabei bedeutet $X^>$ die transponierte Matrix zu X und $(X^> X)$ die (generalisierte) Inverse von $(X^> X)$. Die Gleichung motiviert das folgende Sch atzverfahren:

$$b = (X^> X) X^> Y: \tag{2.9}$$

Setzt man aus 2.2 die Modellbeziehung $Y = X + "$ ein und benutzt, dass $E(") = 0$, so erh alt man

$$E(b) = E \quad X^> X \quad X^> (X + ") = ; \tag{2.10}$$

d.h. b ist ein erwartungstreuer Sch atzer fur . Ob und wieweit dieser Sch atzer neben dieser Konsistenz auch noch statistische Qualit aten hat, wird in Statistik-Vorlesungen diskutiert. Ein Satz zur Charakterisierung dieses Sch atzers ist dort als Gau -Markov-Theorem bekannt. Wir werden auf diesen Sch atzer h au g zur uckkommen und geben ihm deshalb einen Namen: Gau -Markov-Sch atzer . Im Fall eines linearen Modells, wie dem Regressionsmodell, hat dieser Sch atzer eine Reihe von Optimalit atseigenschaften. So minimiert dieser Sch atzer die mittlere quadratische Abweichung, ist also in diesem Modell ein Kleinste-Quadrate-Sch atzer .

Der Kleinste-Quadrate-Sch atzer fur lineare Modelle wird durch die Funktion lm() berechnet.

Zu Illustration erzeugen wir uns einen Beispiel-Datensatz.

Eingabe

```
x <- 1:100
err <- rnorm(100, mean = 0, sd = 10)
y <- 2.5*x + err
```

Den Kleinste-Quadrate-Sch atzer erhalten wir nun durch

Beispiel 2.1:

Eingabe

```
lm(y ~ x)
```

Ausgabe

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

Coefficients:
(Intercept)            x
      1.460        2.492
```

Aufgabe 2.1	
	Wir haben die Daten ohne konstanten Term erzeugt, dies aber bei der Schätzung nicht vorausgesetzt. Wiederholen Sie die Schätzung im Modell ohne konstanten Term. Vergleichen Sie die Resultate.

Der Schätzer b führt unmittelbar zu einer Schätzung $\hat{m}$ für die Funktion m in unserem ursprünglichen Modell:

$$\hat{m}(x) = x^\top b.$$

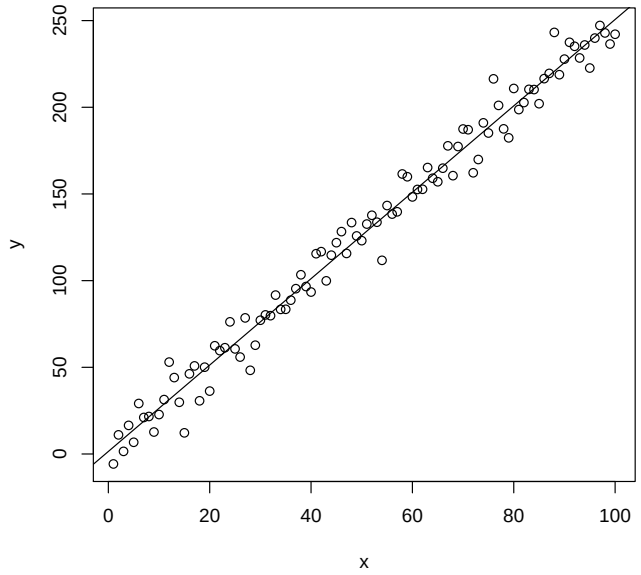
Die Auswertung an einem Punkt x ergibt Werte $\hat{y} := \hat{m}(x)$, den Fit an der Stelle x. Die Auswertung an den Messpunkten ergibt den Vektor der geschätzten Werte $\hat{Y} = X b$.

In unserem Beispiel ist dies eine Regressionsgerade. Mit plot() können wir die Datenpunkte zeichnen. Wenn wir das Resultat der Regression speichern, können wir mit abline() die Regressionsgerade hinzufügen.

Beispiel 2.2:

Eingabe

```
lmres <- lm(y ~ x)
plot(x, y)
abline(lmres)
```



abline() ist eine Funktion, die Geraden anhand von unterschiedlichen Parametrisierungen zeichnen kann. Weiter Information erhalten sie mit help(abline) .

Die Schatzgleichung (2.9) gibt uns an, wie der Fit an den Messpunkten berechnet wird.

$$\hat{Y} = X(X^{>}X)\ X^{>}\ Y: \tag{2.11}$$

Die Matrix

$$H := X(X^{>}X)\ X^{>} \tag{2.12}$$

nennt man Hut-Matrix [2]. Sie ist das wesentliche Werkzeug, um den Gau -Markov-Schatzer fur eine bestimmte Design-Matrix X zu untersuchen. Die Design-Matrix, und damit die Hutmatrix, h angt nur von den Versuchsbedingungen ab, nicht aber von dem Ausgang des Versuchs. Der Fit hingegen bezieht sich auf eine bestimmte Stichprobe, die in den in den beobachteten StichprobenwertenY reprasentiert ist.

Im linearen Modell ist ein Term " enthalten, der den Messfehler oder die Versuchsvariabilit at reprasentiert. Diesen stochastischen Fehler konnen wir nicht direkt beobachten - sonst konnten wir ihn subtrahieren und damit die Modellfunktion exakt bestimmen. Wir konnen nur mittelbar darauf schlie en.

Die Werte der ZufallsbeobachtungY unterscheidet sich in der Regel vom Fit $\hat{Y}$. Die Di erenz

$$R_X(Y) := Y - \hat{Y}$$

[2] sie setzt demY den Hut auf.

heißt Residuum. Das Residuum kann als Schätzer für den nicht-beobachtbaren Fehlerterm ε angesehen werden. Die Residuen sind nicht wirklich der Fehlerterm. Dies wäre nur der Fall, wenn die Schätzung exakt wäre. Für den allgemeinen Fall zeigt uns die Beziehung

$$\begin{aligned} R_X(Y) &= Y - \hat{Y} \\ &= (I - H)Y \\ &= (I - H)(X\beta + \varepsilon) \\ &= (I - H)\varepsilon; \end{aligned} \tag{2.13}$$

dass die Residuen Linearkombinationen der Fehler sind. Wir müssen aus diesen Linearkombinationen auf die Fehler zurück schließen.

Existiert die Varianz der Fehlerterme, so ist durch die Varianzmatrix der Fehlerterme $Var(\varepsilon) = \sigma^2 I$ die Varianz der Residuen bestimmt:

$$\begin{aligned} Var(R_X(Y)) &= Var((I - H)\varepsilon) \\ &= (I - H)\sigma^2 I (I - H)^\top = \sigma^2 (I - H). \end{aligned} \tag{2.14}$$

Bislang haben wir nur vorausgesetzt, dass kein systematischer Fehler vorliegt, modelliert als die Annahme

$$E(\varepsilon) = 0.$$

Wir sprechen von einem **einfachen linearen Modell**, wenn darüber hinaus gilt:

$(\varepsilon_i)_{i=1,\ldots,n}$ sind unabhängig

$Var(\varepsilon_i) = \sigma^2$ für ein σ^2 das nicht von i abhängt.

Im linearen Modell versuchen wir, den Parametervektor β zu schätzen. Die Varianzstruktur des Fehlervektors bringt dabei Störparameter mit sich, die die Schätzung verkomplizieren können. Im einfachen linearen Modell reduziert sich die Situation auf nur einen unbekannten Störparameter σ^2. Formeln wie eq:02-varerr vereinfachen sich, denn in diesem Fall ist $\Sigma = \sigma^2 I$ und der Parameter σ^2 kann aus der Formel herausgezogen werden. Wir können diesen Parameter aus den Residuen schätzen, denn die residuelle Varianz

$$s^2 := \frac{1}{n - Rk(X)} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \tag{2.15}$$

ist ein erwartungstreuer Schätzer für σ^2. Wir schreiben deshalb auch $\hat{\sigma}^2 := s^2$. (Das Wurzelziehen ist keine lineare Operation und erhält deshalb nicht den Erwartungswert. Die residuelle Standardabweichung $\sqrt{s^2}$ ist kein erwartungstreuer Schätzer für σ.) Wieder in die Schätzformel (2.9) eingesetzt liefert uns dies auch eine Schätzung für die Varianz/Covarianzmatrix des Schätzers für β, denn im einfachen Modell ist

$$Var(\hat{\beta}) = \sigma^2 (X^\top X)^{-1} \tag{2.16}$$

und kann durch $s^2 (X^\top X)^{-1}$ geschätzt werden.

Die Standard-Ausgabe in Beispiel 2.1 listet nur minimale Information über den Schätzer. Mehr Information über Schätzer, Residuen und daraus abgeleitete Kennzahlen erhalten wir, wenn wir eine zusammengefasste Darstellung anfordern.

Beispiel 2.3:

Eingabe

```
summary(lm( y ~ x))
```

Ausgabe

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

Residuals:
     Min       1Q   Median       3Q      Max
-26.6301  -4.8625   0.2448   6.7120  25.5667

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.4602     1.9487   0.749    0.455
x             2.4918     0.0335  74.380   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9.671 on 98 degrees of freedom
Multiple R-squared: 0.9826,        Adjusted R-squared: 0.9824
F-statistic:  5532 on 1 and 98 DF,  p-value: < 2.2e-16
```

Aufgabe 2.2	
	Analysieren Sie die in Beispiel 2.3 (Seite 2-8) gezeigten Ausgaben von lm() . Welche Terme konnen Sie interpretieren? Stellen Sie diese Interpretationen schriftlich zusammen. Fur welche Terme fehlt Ihnen noch Information? Erstellen Sie eine kommentierte Version der Ausgabe.

In Abschnitt 2.3 werden wir den theoretischen Hintergrund bereitstellen, der uns hilft, die noch o enen Terme zu interpretieren.

Der Aufruf der Funktion lm() liefert immer ein Resultat - wenn es den Daten angemessen ist, aber es gibt auch ein Resultat, wenn das lineare Modell gar nicht angemessen ist. Wir brauchen deshalb eine Diagnostik, die uns hilft, zu erkennen, ob das Modell verlässlich und brauchbar ist.

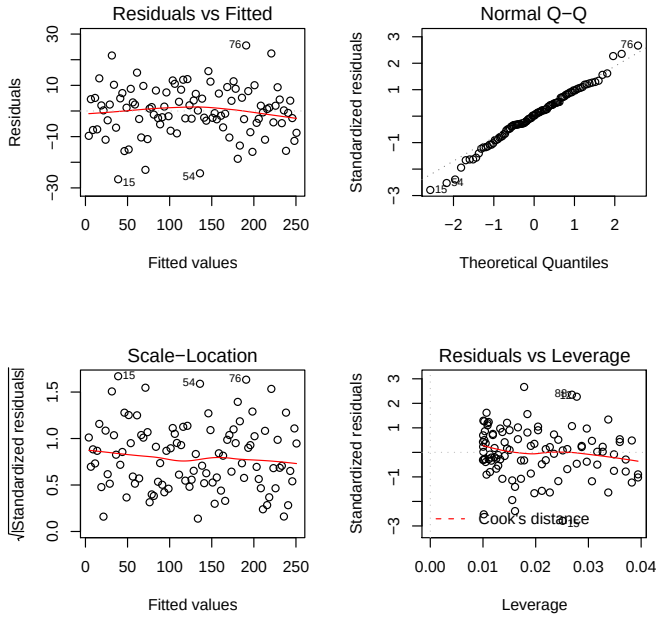
Aufgabe 2.3	
	Sei $yy < 2{:}5\ x + 0{:}01x^2 + \text{err}$. Welches Resultat erhalten Sie, wenn Sie eine Regression mit dem (falschen) Modell $y \sim x$ rechnen? Gibt es Hinweise darauf, dass dieses Modell nicht angemessen ist?

Die Funktion lm() führt nicht nur die Schätzung im linearen Modell durch, sondern liefert eine ganze Reihe von Diagnostiken, die helfen können zu beurteilen, ob die Modellvoraussetzungen vertretbar erscheinen. Eine Darstellung mit plot() zeigt vier Aspekte davon.

Beispiel 2.4:

Eingabe

```
plot(lm(y ~ x))
```



Der obere linke Plot zeigt die Residuen gegen den Fit. Die Verteilung der geschätzten Werte hängt vom Design ab.

Die Residuen sollten annähernd wie ein Scatterplot von unabhängigen Variablen aussehen. Die Verteilung der Residuen sollte nicht vom Fit abhängen. Sind hier systematische Strukturen zu erkennen, so ist das ein Warnzeichen dass das Modell oder die Modellvoraussetzungen nicht erfüllt sind.

Nach der vorausgegangen Diskussion können wir noch genauer sein: die Residuen sollten nach (2.13) Linearkombinationen von unabhängig identisch verteilten Variablen sein. Falls die Modellvoraussetzungen erfüllt sind, ist die Varianz durch (2.14) beschrieben.

In der eindimensionalen Situation würde ein Plot der Residuen gegen den Regressor ausreichen. Für p Regressoren wird die graphische Darstellung problematisch. Der Plot der Residuen gegen den Fit verallgemeinert sich auch auf höhere Dimensionen.

Verteilungsaussagen über die Schätzer und Residuen können wir machen, wenn wir mit Verteilungsaussagen über die Fehlerterme beginnen. Die kräftigsten Aussagen sind möglich, wenn die Fehlerterme unabhängig identisch normalverteilt sind. Der obere rechte Plot sollte annähernd wie der "normal probability plot" von normalverteilen Variablen aussehen, wobei das "annähernd" wiederum bedeutet: bis auf Transformation mit der Matrix $I - H$.

Die beiden übrigen Plots sind spezielle Diagnostiken für lineare Modelle (siehe help(plot.lm)).

Aufgabe 2.4	
	Inspizieren Sie das Resultat von Aufgabe 2.3 grafisch. Welche Hinweise gibt es jetzt, dass das lineare Modell nicht angemessen ist?

plot() stellt für lineare Modelle noch weitere diagnostische Plots bereit. Diese müssen explizit mit dem Parameter which angefordert werden.

---

help (lm)

`lm` *Fitting Linear Models*

Description.

`lm` is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although `aov` may provide a more convenient interface for these).

Usage.

```
lm(formula, data, subset, weights, na.action,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

Arguments.

`formula`	a symbolic description of the model to be fit. The details of model specification are given below.
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `lm` is called.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`weights`	an optional vector of weights to be used in the fitting process. Should be `NULL` or a numeric vector. If non-NULL, weighted least squares is used with weights `weights` (that is, minimizing `sum(w*e^2)`); otherwise ordinary least squares is used.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The \factory-fresh" default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
`method`	the method to be used; for fitting, currently only `method = "qr"` is supported; `method = "model.frame"` returns the model frame (the same as with `model = TRUE`, see below).
`model, x, y, qr`	logicals. If `TRUE` the corresponding components of the fit (the model frame, the model matrix, the response, the QR decomposition) are returned.

`singular.ok` logical. If `FALSE` (the default in S but not in R) a singular fit is an error.

`contrasts` an optional list. See the `contrasts.arg` of `model.matrix.default`.

`offset` this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length either one or equal to the number of cases. One or more `offset` terms can be included in the formula instead or as well, and if both are specified their sum is used. See `model.offset`.

`...` additional arguments to be passed to the low level regression fitting functions (see below).

Details.

Models for `lm` are specified symbolically. A typical model has the form `response ~ terms` where `response` is the (numeric) response vector and `terms` is a series of terms which specifies a linear predictor for `response`. A terms specification of the form `first + second` indicates all the terms in `first` together with all the terms in `second` with duplicates removed. A specification of the form `first:second` indicates the set of terms obtained by taking the interactions of all terms in `first` with all terms in `second`. The specification `first*second` indicates the *cross* of `first` and `second`. This is the same as `first + second + first:second`.

If the formula includes an `offset`, this is evaluated and subtracted from the response.

If `response` is a matrix a linear model is fitted separately by least-squares to each column of the matrix.

See `model.matrix` for some further details. The terms in the formula will be reordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a `terms` object as the formula (see `aov` and `demo(glm.vr)` for an example).

A formula has an implied intercept term. To remove this use either `y ~ x - 1` or `y ~ 0 + x`. See `formula` for more details of allowed formulae.

`lm` calls the lower level functions `lm.fit`, etc, see below, for the actual numerical computations. For programming only, you may consider doing likewise.

All of `weights`, `subset` and `offset` are evaluated in the same way as variables in `formula`, that is first in `data` and then in the environment of `formula`.

Value.

`lm` returns an object of `class "lm"` or for multiple responses of class `c("mlm", "lm")`.

The functions `summary` and `anova` are used to obtain and print a summary and analysis of variance table of the results. The generic accessor functions `coefficients`, `effects`, `fitted.values` and `residuals` extract various useful features of the value returned by `lm`.

An object of class `"lm"` is a list containing at least the following components:

`coefficients` a named vector of coefficients

`residuals` the residuals, that is response minus fitted values.

`fitted.values` the fitted mean values.

`rank` the numeric rank of the fitted linear model.

`weights` (only for weighted fits) the specified weights.

`df.residual` the residual degrees of freedom.

`call` the matched call.

`terms` the `terms` object used.

`contrasts`	(only where relevant) the contrasts used.
`xlevels`	(only where relevant) a record of the levels of the factors used in tting.
`offset`	the o set used (missing if none were used).
`y`	if requested, the response used.
`x`	if requested, the model matrix used.
`model`	if requested (the default), the model frame used.

In addition, non-null ts will have components `assign`, `effects` and (unless not requested) `qr` relating to the linear t, for use by extractor functions such as `summary` and `effects`.

Using time series.

Considerable care is needed when using `lm` with time series.

Unless `na.action = NULL`, the time series attributes are stripped from the variables before the regression is done. (This is necessary as omitting `NA`s would invalidate the time series attributes, and if `NA`s are omitted in the middle of the series the result would no longer be a regular time series.)

Even if the time series attributes are retained, they are not used to line up series, so that the time shift of a lagged or di erenced regressor would be ignored. It is good practice to prepare a `data` argument by `ts.intersect(..., dframe = TRUE)`, then apply a suitable `na.action` to that data frame and call `lm` with `na.action = NULL` so that residuals and tted values are time series.

Note.

O sets speci ed by `offset` will not be included in predictions by `predict.lm`, whereas those speci ed by an o set term in the formula will be.

Author(s).

The design was inspired by the S function of the same name described in Chambers (1992). The implementation of model formula by Ross Ihaka was based on Wilkinson & Rogers (1973).

References.

Chambers, J. M. (1992) *Linear models.* Chapter 4 of *Statistical Models in S* eds J. M. Chambers and T. J. Hastie, Wadsworth & Brooks/Cole.

Wilkinson, G. N. and Rogers, C. E. (1973) Symbolic descriptions of factorial models for analysis of variance. *Applied Statistics*, 22, 392{9.

See Also.

`summary.lm` for summaries and `anova.lm` for the ANOVA table; `aov` for a di erent interface.

The generic functions `coef`, `effects`, `residuals`, `fitted`, `vcov`.

`predict.lm` (via `predict`) for prediction, including con dence and prediction intervals; `confint` for con dence intervals of *parameters*.

`lm.influence` for regression diagnostics, and `glm` for generalized linear models.

The underlying low level functions, `lm.fit` for plain, and `lm.wfit` for weighted regression tting.

Examples.

```
## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## Page 9: Plant Weight Data.
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
group <- gl(2,10,20, labels=c("Ctl","Trt"))
weight <- c(ctl, trt)
anova(lm.D9 <- lm(weight ~ group))
summary(lm.D90 <- lm(weight ~ group - 1))# omitting intercept
summary(resid(lm.D9) - resid(lm.D90)) #- residuals almost identical

opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))
plot(lm.D9, las = 1)      # Residuals, Fitted, ...
par(opar)

## model frame :
stopifnot(identical(lm(weight ~ group, method = "model.frame"),
                    model.frame(lm.D9)))
```

Nicht erwahnt in der Help-Information: mit first-second werden Terme der ersten Gruppe ins Modell aufgenommen, die der zweiten Gruppe aber ausgeschlossen. Auch hier lichere Information zur Formel-Darstellung erhalt man mit help(formula) . Eine Zusammenfassung ist im Anhang A.53 (Seite A-35) zu nden.

Die Hut-Matrix ist eine Besonderheit linearer Modelle. Fit und Residuum jedoch sind allgemeine Konzepte, die bei allen Arten der Schatzung angewandt werden konnen. Die Anwender sind oft mit dem Fit (oder der Schatzung) zufrieden. Fur den ernsthaften Anwender und fur den Statistiker sind die Residuen oft wichtiger: sie weisen darauf hin, was vom Modell oder der Schatzung noch nicht erfasst ist.

2.2.3. Weitere Beispiele f ur lineare Modelle.
Die Matrix X hei t die Design-Matrix des Modells. Sie kann die Matrix sein, die mit den ursprunglichen Messbedingungen x_i als Zeilenvektoren gebildet wird. Aber sie ist nicht auf diesen Spezialfall beschankt. Unter der scheinbar so einfachen Modellklasse der linearen Modelle lassen sich viele wichtige Spezialfalle einordnen. Ein paar davon sind im folgenden zusammengestellt.

Einfache lineare Regression:

$$y_i = a + b\,x_i + \varepsilon_i \qquad \text{mit } x_i \in \mathbb{R};\ a, b \in \mathbb{R}$$

kann als lineares Modell mit

$$X = (1\ x)$$

geschrieben werden, wobei $1 = (1, \ldots, 1)^\top \in \mathbb{R}^n$.

Polynomiale Regression:

$$y_i = a + b_1 x_i + b_2 x_i^2 + \ldots + b_k x_i^k + \varepsilon_i \qquad \text{mit } x_i \in \mathbb{R};\ a, b_j \in \mathbb{R}$$

kann als lineares Modell mit

$$X = (1\ x\ x^2\ \ldots\ x^k)$$

geschrieben werden, wobei $x^j = (x_1^j, \ldots, x_n^j)^\top$.

Analog fur eine Vielzahl von Modellen, die durch andere Transformationen erreicht werden konnen.

Varianzanalyse: Einweg-Layout

Gemessen wird unterm Versuchsbedingungen, dabei n_j Messungen unter Versuchsbedingung $j; j = 1; \ldots; m$. Die Messung setze sich additiv zusammen aus einem Grunde ekt , einem fur die Bedingung j spezi schen Beitrag $_j$, und einem Messfehler nach

$$y_{ij} = + _j + "_{ij} \quad \text{mit} \quad ; \ _j \in \mathbb{R}; \ j = 1; \ldots; n_i:$$

Mit $n = \sum n_j$ und

$$X = (1 \ I_1 \ \ldots \ I_m);$$

wobei I_j die Vektoren der Indikatorvariablen fur die Zugehorigkeit zur Versuchsgruppe j sind, lasst sich dies als lineares Modell schreiben[3].

Covarianzanalyse

Analog zur Varianzanalyse werden Unterschiede zwischen Gruppen untersucht, aber zusatzliche (linear eingehende) Ein ussfaktoren werden korrigierend berucksichtigt. Unter Versuchsbedingung j bei Beobachtung i hangt die Messung zusatzlich von Ein ussfaktoren x_{ij} der Versuchseinheit ij ab.

$$y_{ij} = + _j + b \, x_{ij} + "_{ij} \quad \text{mit} \quad ; \ _j \in \mathbb{R}:$$

2.2.4. Modellformeln. R erlaubt es, Modelle auch dadurch zu spezi zieren, dass die Regeln angegeben werden, nach denen die Design-Matrix gebildet wird. Die Syntax, nach denen die Regeln notiert werden, ist sehr kurz in der Beschreibung von lm() angegeben. Wir diskutieren sie jetzt etwas ausfuhrlicher. Diese Modell-Spezi kation ist auch fur allgemeinere, nicht lineare Modelle moglich. Die Modell-Spezi kationen werden als Attribut mit dem Namen \formula" gespeichert. Sie konnen mit formula() manipuliert werden.

Beispiele

y 1 + x	entspricht $y_i = (1 \ x_i)(\ _1 \ _2)^> + "$
y x	Kurzschreibweise fur y 1 + x (ein konstanter Term wird implizit angenommen)
y 0 + x	entspricht $y_i = x_i + "$
log(y) x1 + x2	entspricht $\log(y_i) = (1 \ x_{i1} \ x_{i2})(\ _1 \ _2 \ _3)^> + "$ (ein konstanter Term wird implizit angenommen)
lm(y poly(x, 4), data = Experiment)	analysiert den Datensatz \Experiment" mit einem linearen Modell fur polynomiale Regression vom Grade 4 in x.

Wichtige Spezialfalle fur faktorielle Designs sind:

[3] Es ist Konvention, dass bei Varianzanalysen der letzte Index die Beobachtung zahlt, und Indizes in alphabetischer Folge vergeben werden. Konventionell werden also im Vergleich zu unserer Notation die Rollen von i und j vertauscht.

- `y ~ A` Einweg-Varianzanalyse mit Faktor A,
- `y ~ A + x` Covarianzanalyse mit Faktor A und Regressions-Covariable x,
- `y ~ A + B` Zwei-Faktor-Kreuz-Layout mit Faktoren A und B ohne Interaktion,
- `y ~ A * B` Zwei-Faktor-Kreuz-Layout mit Faktoren A und B und allen Interaktionen (Kombinationen der Stufen von A und B),
- `y ~ A/B` Zwei-Faktor hierarchisches Layout mit Faktor A und Subfaktor B.

Eine Übersicht über alle Operatoren zur Modellspezifikation ist im Anhang A.53 (Seite A-35) zu finden.

Aufgabe 2.5	
	Schreiben Sie die vier oben in Abschnitt 2.2.3 genannten Modelle als R-Modellformeln.
	Erzeugen Sie sich für jedes dieser Modelle ein Beispiel durch Simulation und wenden Sie `lm()` auf diese Beispiele an. Vergleichen Sie die durch `lm()` geschätzten Parameter mit den Parametern, die Sie in der Simulation benutzt haben.

Die Modellformel wird in einem Eintrag im Resultat von `lm()` gespeichert. Sie kann also aus dem Resultat zurück gewonnen werden. Anhand der Formel-Notation generiert R implizit eine Design-Matrix. Mit `model.matrix()` kann diese Design-Matrix inspiziert werden.

Aufgabe 2.6	
	Generieren Sie drei Vektoren mit je 10 $N(\mu_j; 1)$-verteilten Zufallsvariablen $\mu_j = j;\ j = 1; 3; 9$. Verketten Sie diese zu einem Vektor y. Generieren Sie sich einen Vektor x aus je 10 wiederholten Werten $j;\ j = 1; 3; 9$. Berechnen Sie die Gauß-Markov-Schätzer in den linearen Modellen `y ~ x` und `y ~ factor(x)`. Lassen Sie sich das Resultat jeweils als Tabelle mit `summary()` und als Grafik mit `plot()` anzeigen und vergleichen Sie die Resultate.

2.2.5. Gau -Markov-Sch atzer und Residuen. Wir werfen nun einen genaueren Blick auf den Gau -Markov-Schatzer. Kenntnisse aus der linearen Algebra, langes Nachdenken oder andere Quellen sagen uns:

Bemerkung 2.3.

(1) Die Design-Matrix X de niert eine Abbildung $\mathbb{R}^p \to \mathbb{R}^n$ mit $\mapsto X$.
Der Bild-Raum dieser Abbildung sei $\mathcal{M}_X$; $\mathcal{M}_X \subseteq \mathbb{R}^n$: $\mathcal{M}_X$ ist der von den Spaltenvektoren von X aufgespannte Vektorraum.
(2) Sind die Modell-Annahmen erfullt, so ist $E(Y) \in \mathcal{M}_X$.
(3) $\hat{Y} = \pi_{\mathcal{M}_X}(Y)$, wobei $\pi_{\mathcal{M}_X} : \mathbb{R}^n \to \mathcal{M}_X$ die (euklidische) Orthogonalprojektion ist.
(4) $\hat{b} = \arg\min |Y - \hat{Y}|^2$ wobei $\hat{Y} = X$.

Die Charakterisierung (3) des Gau -Markov-Schatzers als Orthogonalprojektion hilft fur das Verstandnis oft weiter: der Fit ist die Orthogonalprojektion des Beobachtungsvektors auf den Erwartungswertraum des Modells (und minimiert damit den quadratischen Abstand). Das Residuum ist das orthogonale Komplement.

In der Statistik ist die Charakterisierung als Orthogonalprojektion auch ein Ausgangspunkt, um den Schatzer systematisch zu analysieren. In einfachen Fallen helfen Kenntnisse aus der Wahrscheinlichkeitstheorie schon weiter, etwa zusammengefasst im folgenden Satz:

Theorem 2.4. *Sei* Z *eine Zufallsvariable mit Werten in* $\mathbb{R}^n$*, die nach* $N(0; \sigma^2 I_{n\times n})$ *verteilt ist und sei* $\mathbb{R}^n = L_0 \perp \ldots \perp L_r$ *eine Orthogonalzerlegung. Sei* $\pi_i = \pi_{L_i}$ *die Orthogonalprojektion auf* L_i; $i = 0; \ldots; r$.
Dann gilt

(i) $\pi_0(Z); \ldots; \pi_r(Z)$ *sind unabhängige Zufallsvariablen.*
(ii) $\frac{|\pi_i(Z)|^2}{\sigma^2} \sim \chi^2(\dim L_i)$ *für* $i = 0; \ldots; r$.

Beweis. ! Wahrscheinlichkeitstheorie. Siehe z.B. [J rgensen 1993, 2.5 Theorem 3]. □

Mit $\varepsilon = Y - X\beta$ konnen daraus theoretische Verteilungsaussagen uber Schatzer $\hat{b}$ und Residuen $Y - \hat{Y}$ abgeleitet werden.

Insbesondere erhalten wir fur einfache lineare Modelle aus der residuellen Varianz auch einen Schatzer fur die Varianz (bzw. Standardabweichung) jeder einzelnen Komponente $\hat{c}_k$. Die entsprechende t-Statistik und der p-Wert fur den Test der Hypothese $c_k = 0$ sind in der Ausgabe von summary() angegeben.

Aufgabe 2.7	
	Welche Verteilung hat $|R_X(Y)|^2 = |Y - \hat{Y}|^2$, wenn ε nach $N(0; \sigma^2 I)$ verteilt ist?

Auf den ersten Blick ist $|R_X(Y)|^2 = |Y - \hat{Y}|^2$ ein geeignetes Ma , um die Qualitat eines Modells zu beurteilen: kleine Werte sprechen fur den Fit, gro e Werte zeigen, dass der Fit schlecht ist. Dies ist jedoch mit Sorgfalt zu betrachten. Zum einen hangt diese Gro e von linearen Skalenfaktoren ab. Zum anderen muss die Dimensionen der jeweiligen Raume mit in Betracht gezogen werden. Was passiert, wenn weitere Regressoren ins Modell aufgenommen werden? Wir haben z.B. gesehen, dass \linear" auch die Moglichkeit gibt, nichtlineare

Beziehungen zu modellieren, zum Beispiel dadurch, dass geeignet transformierte Variable in die Design-Matrix mit aufgenommen werden. Die Charakterisierung (3) aus Bemerkung 2.3 sagt uns, dass e ektiv nur der von der Design-Matrix aufgespannte Raum relevant ist. Hier sind die Grenzen des Gau -Markov-Sch atzers im linearen Modell erkennbar: wenn viele transformierte Variablen aufgenommen werden, oder generell wenn der durch die Design-Matrix bestimmte Bildraum zu gro wird, gibt es eine Uberanpassung. Im Extrem ist $\hat{Y} = Y$. Damit werden alle Residuen zu null, aber die Sch atzung ist nicht brauchbar.

Wir benutzen $jR_X(Y)j^2=\dim(L_X)$, wobei L_X das orthogonale Komplement von $\mathcal{M}_X$ in $\mathbb{R}^n$ ist (also $\dim(L_X) = n \quad \dim(\mathcal{M}_X)$), um die Dimensionsabh angigkeit zu kompensieren.

Aufgabe 2.8	
	Modi zieren Sie die Plot-Ausgabe plot.lm() fur die linearen Modelle so, dass anstelle des Tukey-Anscombe-Plots die studentisierten Residuen gegen den Fit aufgetragen werden.
	Erganzen Sie den QQ-Plot durch Monte-Carlo-B ander fur unabhangige Gau 'sche Fehler. *Hinweis:* Sie konnen die B ander nicht direkt aus der Gau verteilung generieren - Sie brauchen die Residuenverteilung, nicht die Fehlerverteilung.

Aufgabe 2.9	
	Schreiben Sie eine Prozedur, die fur die einfache lineare Regression $y_i = a + bx_i + "_i$ mit $x_i \; 2 \; \mathbb{R}; a; b \; 2 \; \mathbb{R}$ den Gau -Markov-Sch atzer berechnet und vier Plots darstellt: Respons gegen Regressor, mit gesch atzter Geraden studentisierte Residuen gegen Fit Verteilungsfunktion der studentisierten Residuen im QQ-Plot mit B andern Histogramm der studentisierten Residuen

2.3. Streuungszerlegung und Varianzanalyse

Wenn ein einfaches lineares Modell mit gau verteilten Fehlern vorliegt, sind die t-Tests geeignet, eindimensionale Probleme (Tests oder Kon denzintervalle fur einzelne Parameter, punktweise Kon denzintervalle) zu l osen. Um simultane oder mehrdimensionale Probleme zu losen brauchen wir andere Werkzeuge. Anstelle der Di erenzen oder Mittelwerte, die den t-Tests zu Grunde liegen, benutzen wir Norm-Abst ande (bzw. quadratische Abst ande), die auch auf h ohere Dimensionen generalisieren.

Die Interpretation des Gau -Markov-Sch atzers als Orthogonalprojektion (Bem. 2.3 3) zeigt eine Moglichkeit, Modelle zu vergleichen: Fur $X; X^0$ Design-Matrizen mit $\mathcal{M}_{X^0} \quad \mathcal{M}_X$, betrachten wir die Zerlegung $\mathbb{R}^n = L_0 \quad \cdots \quad L_r$ mit $L_0 := \mathcal{M}_{X^0}$, und die orthogonale

Komplemente $L_1 := \mathcal{M}_X \ominus \mathcal{M}_{X_0}; L_2 := \mathbb{R}^n \ominus \mathcal{M}_X$. Wieder bezeichnet $\|\cdot\|$ jeweils die entsprechende Projektion.

$$F := \frac{\frac{1}{\dim(L_1)} \| \mathcal{M}_X Y - \mathcal{M}_{X_0} Y \|^2}{\frac{1}{\dim(L_2)} \| Y - \mathcal{M}_X Y \|^2}:$$

Diese Statistik, die F-Statistik (nach R.A. Fisher) ist die Basis für die Varianzanalyse, einer klassischen Strategie, Modelle zu vergleichen. Streuungszerlegung ist ein anderer Name für diesen Ansatz.

Die Idee wird auf Ketten von Modellen verallgemeinert. Ist $\mathcal{M}_0 \subset \cdots \subset \mathcal{M}_r = \mathbb{R}^n$, so liefert $L_0 := \mathcal{M}_0; L_i := \mathcal{M}_i \ominus \mathcal{M}_{i-1}$ für $i = 1, \ldots, r$ eine Orthogonalzerlegung. Mit den Bezeichnungen von oben ist dann

$$\frac{\frac{1}{\dim L_{i-1}} \| \mathcal{M}_i Y - \mathcal{M}_{i-1} Y \|^2}{\frac{1}{\dim L_i} \| Y - \mathcal{M}_i Y \|^2}$$

eine Teststatistik, die zum Test für das Modell $\mathcal{M}_{i-1}$ im Vergleich zum Obermodell $\mathcal{M}_i$ herangezogen wird.

Aufgabe 2.10	
	Welche Verteilung hat F, wenn $E(Y) \in \mathcal{M}_{X_0}$ gilt und ε nach $N(0; \sigma^2 I)$ verteilt ist?

Aufgabe 2.11	
	Geben Sie eine explizite Formel für die F-Statistik zur Varianzanalyse im Einweg-Layout $y_{ij} = \mu + \alpha_j + \varepsilon_{ij}$ im Vergleich zum homogenen Modell $y_{ij} = \mu + \varepsilon_{ij}$.

Die Varianzanalyse gibt eine andere Darstellung und Interpretation der linearen Modelle. Hier im Vergleich zu Beispiel 2.3 die Varianzanalyse-Darstellung:

Beispiel 2.5:

Eingabe

```
summary(aov(lmres))
```

Ausgabe

```
            Df Sum Sq Mean Sq F value    Pr(>F)
x            1 517386  517386  5532.4 < 2.2e-16 ***
Residuals   98   9165      94
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Aufgabe 2.12	
	Analysieren Sie die in Beispiel 2.3 (Seite 2-8) gezeigten Ausgaben von lm() . Welche Terme konnen Sie jetzt interpretieren? Stellen Sie diese Interpretationen schriftlich zusammen. Fur welche Terme fehlt Ihnen noch Information?

In der Ausgabe nden Sie noch einen Hinweis auf \R-squared". Der Term, der hier angegeben wird, ist ein Schatzer fur den Anteil von $Var(Y)$, der durch das Modell erklart wird:

$$R^2 = \frac{mss}{mss + rss}$$

mit $mss := \frac{1}{n}\sum (\hat{y}_i - \bar{\hat{y}})^2$ und $rss := \frac{1}{n}\sum (R_X(Y)_i - \overline{R_X(Y)})^2$. Die Bezeichnung R^2 kommt von der einfachen linearen Regression. Dort ist konventionell die Korrelation $Cor(X;Y)$ mit R bezeichnet, und $R^2 = Cor(X;Y)^2$. R^2 berucksichtigt nicht die Anzahl der geschatzten Parameter und kann deshalb zu optimistisch sein. Der Term \adjusted R-squared" hat eine Gewichtung, die die Freiheitsgrade berucksichtigt.

---

help (anova)

`anova` *Anova Tables*

Description.

Compute analysis of variance (or deviance) tables for one or more tted model objects.

Usage.

```
anova(object, ...)
```

Arguments.

`object`	an object containing the results returned by a model tting function (e.g., `lm` or `glm`).
`...`	additional objects of the same type.

Value.

This (generic) function returns an object of class `anova`. These objects represent analysis-of-variance and analysis-of-deviance tables. When given a single argument it produces a table which tests whether the model terms are signi cant.

When given a sequence of objects, `anova` tests the models against one another in the order speci ed.

The print method for `anova` objects prints tables in a \pretty" form.

Warning.

The comparison between two or more models will only be valid if they are tted to the same dataset. This may be a problem if there are missing values and R's default of `na.action = na.omit` is used.

References.

Chambers, J. M. and Hastie, T. J. (1992) *Statistical Models in S*, Wadsworth & Brooks/Cole.

See Also.

`coefficients`, `effects`, `fitted.values`, `residuals`, `summary`, `drop1`, `add1`.

Modelle für die Varianzanalyse können als Regeln angegeben werden. Dieselbe Syntax zur Modellbeschreibung wird benutzt wie schon bei der Regression. Wenn Terme auf der rechten Seite der Modellbeschreibung Faktoren sind, wird automatisch ein Varianzanalyse-Modell anstelle eines Regressionsmodells generiert.

Die Modellbeschreibung bestimmt die linearen Räume, in denen die Erwartungswerte liegen. Die Streuungszerlegungen sind dadurch jedoch nicht eindeutig bestimmt: die Angabe der Räume lässt evtl. noch verschiedene Orthogonalzerlegungen zu (z.B. abhängig von der Reihenfolge). Mehr noch: die Angabe der Faktoren bestimmt ein Erzeugendensystem der Räume. Die Faktoren brauchen nicht orthogonal zu sein, noch nicht einmal unabhängig.

Dies gilt für alle linearen Modelle. In der Regression ist Abhängigkeit eher die Ausnahme. Bei faktoriellen Designs taucht dieser Fall häufig auf. Die Einweg-Varianzanalyse in Koordinatendarstellung illustriert dieses Problem: mit

$$y_{ij} = \quad + \ _j + \ "_{ij} \quad \text{mit} \ ; \ _j \ 2 \ \mathbb{R}$$

ist für $n_j > 0$ die Zerlegung in und $_j$ nicht eindeutig. Der tieferliegende Grund ist: der globale Faktor de niert den vom Einheitsvektor 1 aufgespannten Raum, und dieser liegt in dem von den Gruppenindikatoren aufgespannten Raum.

Die Modellformel de niert eine Designmatrix X und damit einen Modellraum. Eine zusätzliche Matrix C wird benutzt, um die Matrix zu reduzieren und damit eine eindeutige Streuungszerlegung zu spezi zieren. Die e ektive Designmatrix ist dann $[1 \ X \ C]; C$ hei t Kontrastmatrix . Die Funktionen zur Varianzanalyse wie z.B. `lm()` oder `aov()` erlauben es, die Kontraste zu spezi zieren.

Die Funktion `anova()` operiert wie eine spezielle Formatierung der Ausgabe und wird analog `summary()` benutzt, also z.B. in der Form `anova(lm())` .

Aufgabe 2.13	
	Die Datei \micronuclei" enthält einen Datensatz aus einem Mutagenitätstest. Zellkulturen (je 50 Einheiten) wurden in einer Kontrollgruppe und unter 5 chemischen Behandlungen beobachtet. Der E ekt der Substanzen ist, die Chromosomen aufzubrechen und Mikronuklei zu bilden. Registriert wurde die Gröe der Mikronuklei (relativ zum Eltern-Nukleus). Lesen Sie die Datei \micronuclei" und berechnen Sie für jede Gruppe Mittelwert und Varianz. Hinweis: Sie können die Datei mit `data()` einlesen. Für Dateien mit Tabellenformat gibt es die spezielle Anweisung `read.table()` . Informieren Sie sich mit `help()` über beide Funktionen. Einige ausgewählte statistische Funktionen (z.B. Mittelwert) nden Sie in Tabelle A.22 im Anhang.
	Vergleichen Sie die Resultate. Sind Behandlungse ekte nachweisbar? Hinweise: Versuchen Sie zunächst, die Aufgabe als Einweg-Varianzanalyse zu formulieren. Den Datensatz müssen Sie zunächst z.B. mit Hilfe von `c()` auf eine geeignete Form bringen.

Aufgabe 2.14	
	Schreiben Sie eine Funktion oneway(), die als Argument eine Datentabelle nimmt und eine Einweg-Varianzanalyse als Test auf die Differenz zwischen den Spalten durchführt.
	Erganzen Sie oneway() durch die notwendigen diagnostischen Plots. Welche Diagnostiken sind notwendig?

Aufgabe 2.15	
	Das Industrieunternehmen Kiwi-Hopp[4] mochte einen neuen Hubschrauber auf den Markt bringen. Die Hubschrauber mussen also danach beurteilt werden, wie lange sie sich in der Luft halten, bis sie aus einer gegebenen Hohe (ca. 2m) den Boden erreichen.[5] Eine Konstruktionszeichnung ist unten (Abbildung 2.1, Seite 2-24) angegeben. Welche Faktoren konnten die Variabilitat der Flug(Sink)zeiten beeinflussen? Welche Faktoren konnten die mittlere Flugzeit beeinflussen?
	Fuhren Sie 30 Versuchsfluge mit einem Prototyp durch und messen Sie die Zeit in 1/100s. (Sie mussen vielleicht zusammenarbeiten, um die Messungen durchzufuhren.) Wurden Sie die gemessene Zeit als normalverteilt ansehen? Die Anforderung ist, dass die mittlere Flugdauer mindestens 2.4s erreicht. Erfullt der Prototyp diese Anforderung?
	Sie haben die Aufgabe, einen Entwurf fur die Produktion auszusuchen. Folgende Varianten stehen zur Diskussion: Rotorbreite 45mm Rotorbreite 35mm Rotorbreite 45mm mit Zusatzfalte als Stabilisierung Rotorbreite 35mm mit Zusatzfalte als Stabilisierung. Ihr Haushalt erlaubt ca. 40 Testfluge. (Wenn Sie mehr Testfluge benotigen, mussen Sie dies gut begrunden.) Bauen Sie 4 Prototypen und fuhren sie Testfluge durch, bei denen Sie die Zeit messen. Finden Sie diejenige Konstruktion, die die langste Flugdauer ergibt. Erstellen Sie einen Bericht. Der Bericht sollte folgende Details enthalten: eine Liste der erhobenen Daten und eine Beschreibung des experimentellen Vorgehens. geeignete Plots fur jede Konstruktion eine Varianzanalyse eine klare Zusammenfassung Ihrer Schlusse. (Fortsetzung)!

[4]Nach einer Idee von Alan Lee, Univ. Auckland, Neuseeland

[5]Kiwis konnen nicht fliegen.

Aufgabe 2.15	(Fortsetzung)
	Weitere Hinweise: Randomisieren Sie die Reihenfolge Ihrer Experimente. Reduzieren Sie die Variation, indem Sie gleichmässige Bedingungen für das Experiment schaffen (gleiche Höhe, gleiche Abwurftechnik etc.).
	Die Zusatzfaltung verursacht zusätzliche Arbeitskosten. Schätzen sie den Effekt ab, den diese Zusatzinvestition bringt.

Aufgabe 2.16	
	Benutzen Sie den Quantil-Quantil-Plot, um paarweise die Resultate des Helikopter-Experiments aus dem letzten Kapitel zu vergleichen. Formulieren Sie die Resultate.

Aufgabe 2.17	
	Inspizieren Sie die Implementierung von `qqnorm()`. Programmieren Sie eine analoge Funktion für den PP-Plot und wenden Sie diese auf die Helikopter-Daten an.

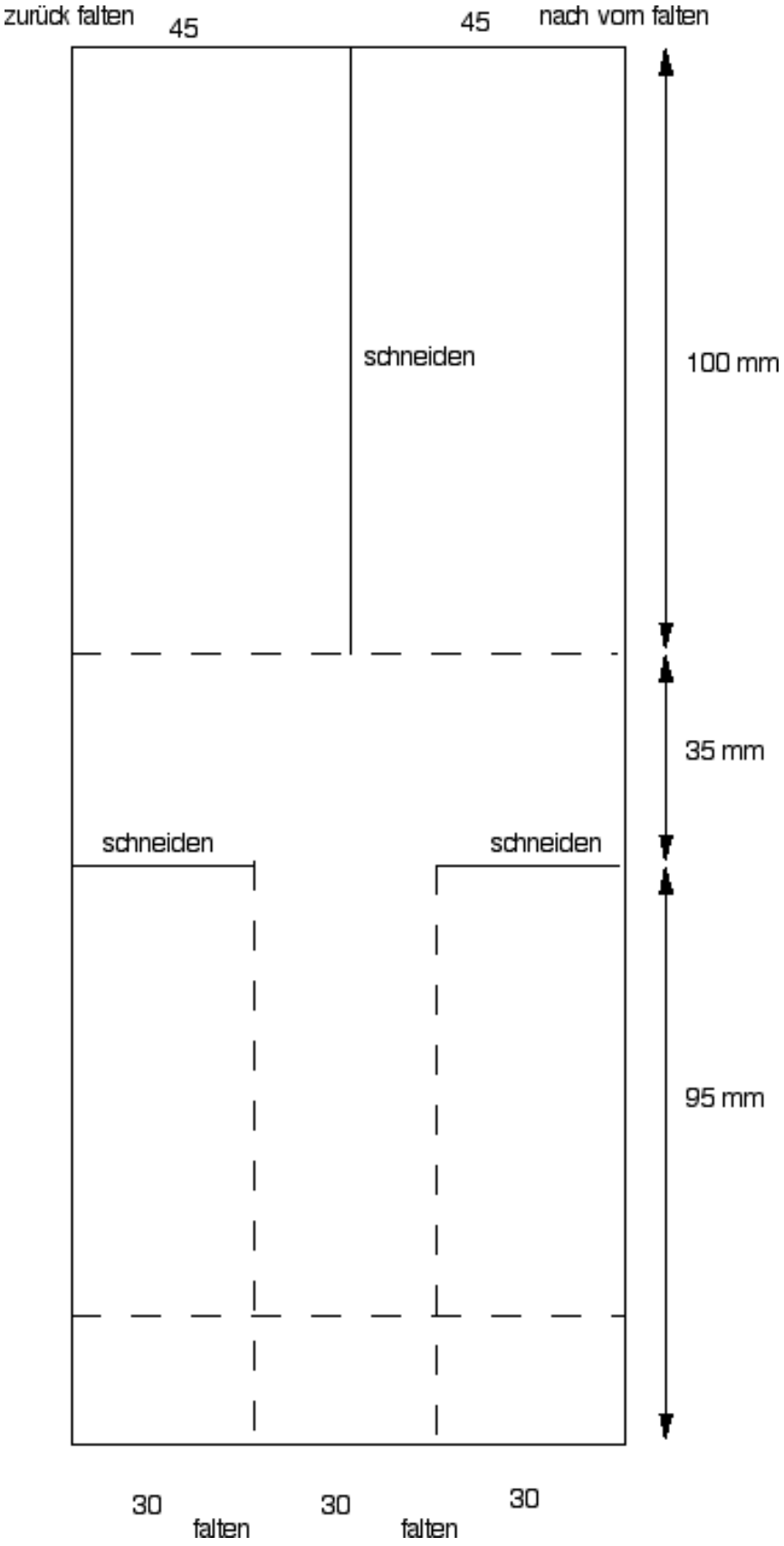


Abbildung 2.1. KiwiHopp

2.4. Simultane Sch atzung

2.4.1. Schee's Kon denz-B ander. Der Kleinste-Quadrate-Schatzer schatzt im Prinzip alle Komponenten des Parameter-Vektors simultan. Die Optimalitats-Aussagen des Gau -Markov-Theorems beziehen sich nur auf eindimensionale lineare Statistiken. Die Kon denzaussagen gelten jedoch multivariat. Es gilt: Der mithilfe der F -Verteilung gewonnene Kon denzbereich zum Kon denzniveau 1 hat die Form

$$f\beta 2 \mathbb{R}^k : (\sum_{j=1}^{k} (\beta_j \quad _j)^2 k x_j k^2 = k) = c^2 \quad F_1 \quad (k; n \quad k) g;$$

d.h. der Kon denzbereich ist eine Ellipse. Wir konnen die Ellipse auch als den Bereich de nieren, der durch alle Tangenten der Ellipse begrenzt wird. Diesubersetzt die (eine) quadratischen Bedingung an die Punkte im Kon denzbereich durch (unendlich viele) lineare Bedingungen. Diese geometrische Beziehung ist der Kernurfden folgenden Satz:

Theorem 2.5. *Sei* $\mathscr{L} \quad \mathbb{R}^k$ *ein linearer Unterraum der Dimension* d; $EY = X\beta$ *mit* $rk(X) = p < n$. *Dann ist*

$$P f ' t \quad 2 ' t \beta \quad (dF_{d;n} \quad)^{1=2} s(' t (X t X) \quad 1 ')^{1=2} 8 ' \ 2 \ \mathscr{L} g = (1 \quad):$$

Beweis. [Mil81 , 2.2, p. 48] □

Dies ist ein simultaner Kon denzbereich fur alle Linearkombinationen aus $\mathscr{L}$. Als Test ubersetzt ergibt dies einen simultanen Test fur alle linearen Hypothesen aus $\mathscr{L}$. Im Falle d = 1 reduziert sich dieser Schee-Test auf den ublichen F -Test. Ublicherweise ist es nicht moglich, am selben Datenmaterial mehrere Tests durchzufhren, ohne dadurch das Kon denzniveau zu verschlechtern. Der F -Test ist eine Ausnahme. Nach einem globalen F -Test konnen diese Linearkobinationen oder Kontraste einzeln getestet werden, ohne das Niveau zu verletzen.

Im Falle der einfachen linearen Regressionubersetzt sich das Kon denz-Ellipsoid im Parameterraum so in ein Hyperboloid als Kon denzbereich fur die Regressionsgeraden im Regressor/Respons-Raum.

Geht man zur Interpretation im Regressor/Respons-Raum, also dem Raum der Versuchsbedingungen und Beobachtungenuber, so ist man hau g nicht so sehr an einem Kon denzbereich fur die Regressionsgerade interessiert, sondern daran, einen Prognosebereich (Toleranzbereich) fur weitere Beobachtungen anzugeben. Fr diesen muss zur Streuung der Regression noch die Fehlerstreuung addiert werden. Der Toleranzbereich ist entsprechend gro er. Kon denzbereich fur die Regressionsgerade und Toleranzbereich fur Beobachtungen konnen mit der Funktion predict() berechnet werden. Die folgende Abbildung zeigt beide Bereiche. Die Funktion predict() ist eine generische Funktion. Fur lineare Modelle ruft sie predict.lm() auf. predict() erlaubt es, neue Stutzstellen als Parameternewdata vorzugeben, an denen anhand des gesch atzten Modells ein Fit berechnet wird. Die Variablen werden hier dem Namen nach zugeordnet. Deshalb muss newdata ein data.frame sein, dessen Komponenten-Namen den ursprunglichen Variablen entsprechen.

Wir bereiten einen Beispieldatensatz vor.

Eingabe

```
n <- 100
sigma <- 1
x <- (1:n)/n-0.5
err <- rnorm(n)
```

```
y <- 2.5 * x + sigma*err
lmxy <- lm(y ~ x)
```

Um bessere Kontrolleuber die Gra k zu bekommen, berechnen wir die Plot-Grenzen und Stutzpunkte vorab.

Eingabe

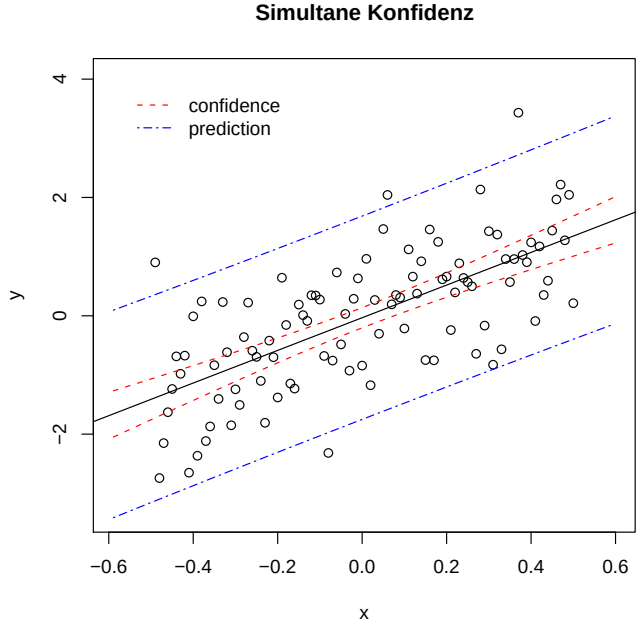
```
plotlim <- function(x){
      xlim <- range(x)
      # check implementation of plot. is this needed?
      del <- xlim[2]-xlim[1]
      if (del>0)
            xlim <- xlim+c(-0.1*del, 0.1*del)
      else xlim <- xlim+c(-0.1, 0.1)
      return(xlim)
}
xlim <- plotlim(x)
ylim <- plotlim(y)
#newx <- data.frame(x = seq(1.5*min(x), 1.5*max(x), 1/(2*n)))
newx <- data.frame(x = seq(xlim[1], xlim[2], 1/(2*n)))
```

Fur diese Daten berechnen wir nun Kon denzb ander und zeichnen sie.

Beispiel 2.6:

Eingabe

```
plot(x, y, xlim = xlim, ylim = ylim)
abline(lmxy)
pred.w.plim <- predict(lmxy, newdata = newx, interval = "prediction")
pred.w.clim <- predict(lmxy, newdata = newx, interval = "confidence")
matplot(newx$x,
      cbind(pred.w.clim[, -1], pred.w.plim[, -1]),
      lty = c(2, 2, 6, 6),
      col = c(2, 2, 4, 4),
      type = "l", add = TRUE)
title(main = "Simultane Konfidenz")
legend("topleft",
      lty = c(2, 6),
      legend = c("confidence", "prediction"),
      col = c(2, 4),
      inset = 0.05, bty = "n")
```



2.4.2. Tukey's Kon denz-Intervalle. Geometrisch ist das Kon denz-Ellipsoid also durch seine (unendlich vielen) Tangentialebenen gekennzeichnet. Übersetzt als Test werden hier unendlich viele lineare Tests simultan durchgeführt. In vielen Anwendungen ist es jedoch möglich, gezieltere Fragestellungen anzugehen, etwa im Zwei-Stichprobenfall nur die Hypothese $_1$ $_2$ = 0 : Diese reduzierten Fragestellungen können in linearen Modellen formuliert werden und zu schärferen Tests führen. Dies geschieht durch die Spezi zierung von Kontrasten und wird in R auch für die Varianzanalyse unterstützt.

2.4.2.1. *Fallstudie: Titrierplatte.* Eine typisches Werkzeug in der Biologie und Medizin sind Tritrierplatten, die z.B. bei Versuchen mit Zellkulturen eingesetzt werden. Die Platte enthält in einem rechteckigen Raster kleine Vertiefungen. Auf die Platte insgesamt können

Substanzen aufgebracht. Mit einer Multipipette konnen auch spaltenweise oder zeilenweise Substanzen aufgebracht werden (Abbildung 2.2).

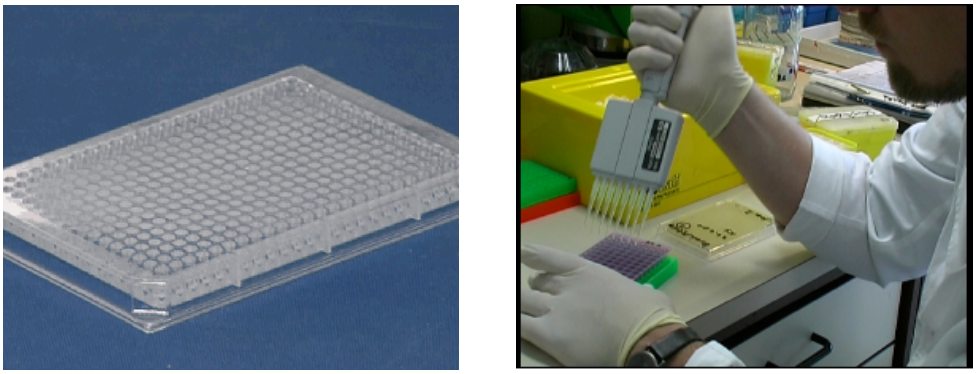


Abbildung 2.2. Titrierplatten. Mit Mulipipetten k onnen zeilenweise oder spaltenweise Substanzen aufgebracht werden.

Die Experimente werden oft in Serien durchgefhrt. Aus einer Serie benutzen wir als Beispiel nur die Daten einer Platte.

Eingabe

```
p35 <- read.delim("../data/p35.tab")
```

Fur die Analyse mit lm() mussen wir die Daten aus der Matrix-Form in eine lange Form uberfuhren, die die Behandlung in einer Spaltenvariablen auuhrt. Die Spalte H in diesem Versuch enthalt keine Behandlung, sondern dient nur zur Qualitatskontrolle zwischen denPlatten.

Eingabe

```
s35 <- stack(p35[,3:9])                                    # ignore column H
s35 <- data.frame(y=s35$values,
    Tmt=s35$ind,
    Lane=rep(1:12, length.out=dim(s35)[1]))    # rename
lmres <- lm(y ~ 0+ Tmt, data= s35)                         # we do not want an overall mean
```

Die Zusammenfassung als lineares Modell enthlt Tests fur die einzelnen Koe zienten.

Eingabe

```
summary(lmres)
```

Ausgabe

```
Call:
lm(formula = y ~ 0 + Tmt, data = s35)

Residuals:
      Min        1Q    Median        3Q       Max
-0.084833 -0.016354  0.009125  0.022729  0.073083

Coefficients:
     Estimate Std. Error t value Pr(>|t|)
TmtA  0.19383    0.01035   18.73   <2e-16 ***
TmtB  0.24892    0.01035   24.06   <2e-16 ***
TmtC  0.23783    0.01035   22.99   <2e-16 ***
TmtD  0.24117    0.01035   23.31   <2e-16 ***
```

```
TmtE  0.24392    0.01035   23.57   <2e-16 ***
TmtF  0.23558    0.01035   22.77   <2e-16 ***
TmtG  0.22367    0.01035   21.62   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.03584 on 77 degrees of freedom
Multiple R-squared: 0.9787,        Adjusted R-squared: 0.9768
F-statistic: 506.2 on 7 and 77 DF,  p-value: < 2.2e-16
```

Fur dieses Beispiel sind die Tests fur die einzelnen Koe zienten nicht angebracht. anova() listet eine Zusammnenfassung, die auf die Varianzanalyse zugeschnitten ist.

Eingabe

```
anova(lmres)
```

Ausgabe

```
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
Tmt        7 4.5513  0.6502  506.15 < 2.2e-16 ***
Residuals 77 0.0989  0.0013
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Wenn die Voraussetzungen des Gau -linearen Modells gegeben sind, so ist der Behandlungse ekt signi kant. Es stellt sich sofort die Frage, zwischen welchen der Behandlungen ein signi kanter Unterschied besteht, d.h. uns interessieren die Kontraste, die die Behandlungsunterschiede beschreiben. Ohne das Niveau zu verletzen kann diese post-hoc mit Tukey's Ansatz untersucht werden. Dazu brauchen wir die Funktion glht() fur den Test fur generalisierte lineare Hypothesen, die in der Bibliothek multcomp fur multiples Testen bereit gestellt. ist.

Beispiel 2.7:

Eingabe

```
library(multcomp)
lhtres<-glht(lmres,linfct=mcp(Tmt="Tukey"))
summary(lhtres)          # muliple tests
```

Ausgabe

```
         Simultaneous Tests for General Linear Hypotheses

Multiple Comparisons of Means: Tukey Contrasts


Fit: lm(formula = y ~ 0 + Tmt, data = s35)

Linear Hypotheses:
            Estimate Std. Error t value p value
B - A == 0  0.055083   0.014632   3.765 0.00577 **
C - A == 0  0.044000   0.014632   3.007 0.05277 .
D - A == 0  0.047333   0.014632   3.235 0.02834 *
E - A == 0  0.050083   0.014632   3.423 0.01651 *
F - A == 0  0.041750   0.014632   2.853 0.07793 .
G - A == 0  0.029833   0.014632   2.039 0.39861
C - B == 0 -0.011083   0.014632  -0.757 0.98819
D - B == 0 -0.007750   0.014632  -0.530 0.99832
E - B == 0 -0.005000   0.014632  -0.342 0.99986
F - B == 0 -0.013333   0.014632  -0.911 0.96971
G - B == 0 -0.025250   0.014632  -1.726 0.60126
D - C == 0  0.003333   0.014632   0.228 0.99999
E - C == 0  0.006083   0.014632   0.416 0.99958
F - C == 0 -0.002250   0.014632  -0.154 1.00000
G - C == 0 -0.014167   0.014632  -0.968 0.95930
E - D == 0  0.002750   0.014632   0.188 1.00000
F - D == 0 -0.005583   0.014632  -0.382 0.99974
G - D == 0 -0.017500   0.014632  -1.196 0.89361
F - E == 0 -0.008333   0.014632  -0.570 0.99748
G - E == 0 -0.020250   0.014632  -1.384 0.80861
G - F == 0 -0.011917   0.014632  -0.814 0.98279
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
(Adjusted p values reported)
```

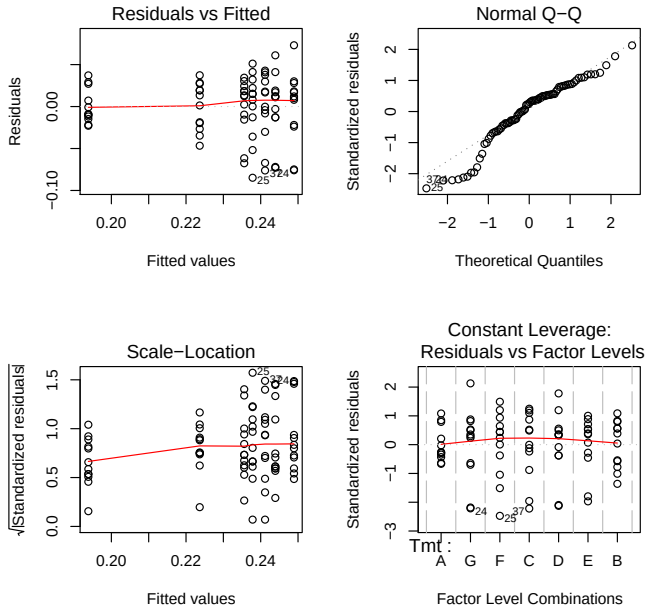
Unter den Voraussetzungen des Modells ist damit die Signi kanz der Unterschiede von A zu B und D gesichert.

Um die Voraussetzungen zu pufen, stehen uns die Residuen zur Verfgung, die wir mit plot() inspizieren konnen.

Eingabe

```
oldpar <- par(mfrow=c(2,2))
plot(lmres)
par(oldpar)
```

Beispiel 2.8:



Die Verteilung zeigt eine deutliche Abweichung von der Normalverteilung, insbesondere bei kleinen Werten. Wir konnen diese selektiv inspizieren.

Beispiel 2.9:

Eingabe

```
#diagnostic
library(MASS)
s35$studres <- studres(lmres)
s35[s35$studres < -1,]
```

Ausgabe

```
       y Tmt Lane   studres
13 0.174   B    1 -2.239390
24 0.173   B   12 -2.271296
25 0.153   C    1 -2.559766
33 0.202   C    9 -1.044865
36 0.186   C   12 -1.523409
37 0.165   D    1 -2.279286
48 0.174   D   12 -1.994858
49 0.171   E    1 -2.175828
60 0.172   E   12 -2.144169
61 0.168   F    1 -2.007887
72 0.174   F   12 -1.821449
73 0.177   G    1 -1.367608
84 0.189   G   12 -1.010381
```

Das Muster ist auffällig. Fast alle besonders kleinen Werte sind am Rand der Platte.

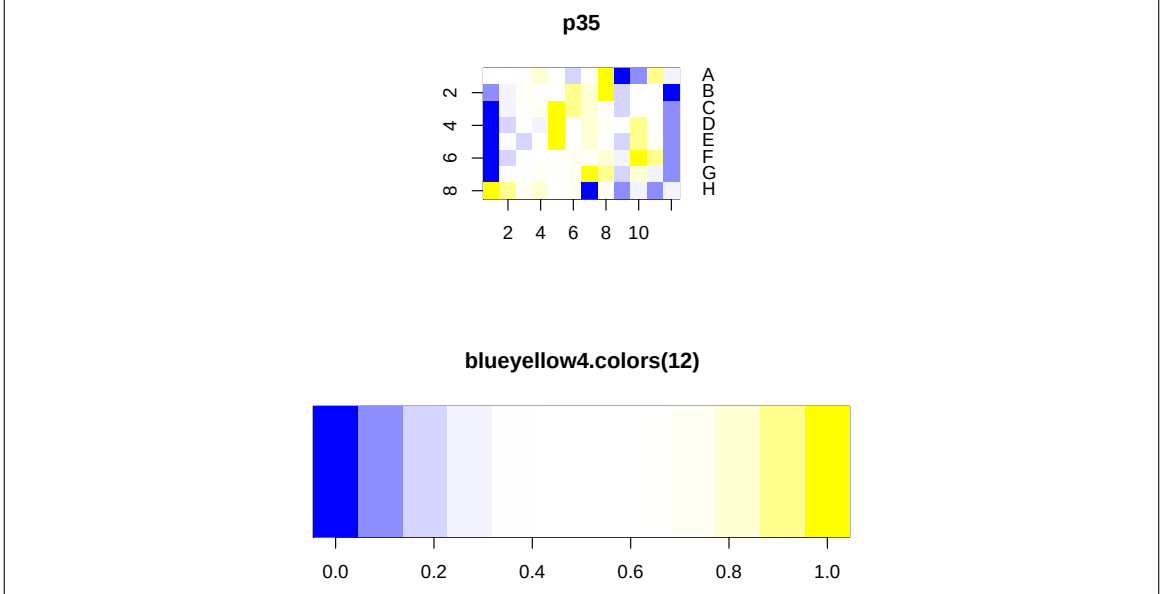
Dieses Muster hätten wir auch mit einer visuellen Inspektion erkennen können:

Beispiel 2.10:

Eingabe

```
# visualisation
# image, the easy way
a35 <- as.matrix(p35[3:10])
a35rk <- apply(a35,2, rank)
#image(a35rk)

# enhanced image, using bertin
library(bertin)
oldpar<- par(mfrow=c(2,1))
imagem(t(a35rk), col=blueyellow4.colors(12), main="p35")
colramp(blueyellow4.colors(12),12,horizontal=TRUE)
par(oldpar)
```



Bei unabhängigen Fehlern hätten wir eine zufällige Verteilung der Range in den Zeilen. Die Konzentrierung der extremen Werte in den extremen Spalten zeigt eine inhomogenität im Produktionsprozeß.

In diesem Beispiel können wir also berichten, dass anscheinend ein Unterschied zwischen der Behandlung A und speziellen anderen Behandlungen besteht. Die Beurteilung ist aber mit Vorbehalt zu betrachten: die Modellvoraussetzungen sind nicht erfüllt. Es gibt eine erkennbare Inhomogenität zwischen den Zeilen. Wichtiger ist also der Hinweis im Produktionsprozess nach Ursachen dieser Inhomogenität zu suchen.

2.5. Nichtparametrische Regression

2.5.0.2. *Transformationen.*

2.5.0.3. *Box-Cox-Transformationen.* Lage- und Skalenparameter konnen auch als Versuch verstanden werden, die Verteilung auf eine Referenzgestalt zu transformieren. Lage- und Skalenparameter erfassen nur lineare Transformationen.

Die Box -Cox -Transformationen

$$y^{( )} = \begin{cases} \frac{y \quad 1}{} & \text{fur } \quad 6= 0; \\ \log(y) & \text{fur } \quad = 0 \end{cases}$$

sind eine Familie, die so skaliert ist, dass die Logarithmus-Transformation stetig in Potenz-Transformationen eingebettet ist. Die Funktion boxcox() in library(MASS) kann benutzt werden, um zu wahlen.

Generalisierte lineare Modelle sind so erweitert, dass sie bestimmte Transformationen schon im Modell berucksichtigen konnen. Dazu nden Sie weitere Information in [VR02].

2.5.1. Zwischenspiel: verallgemeinerte lineare Modelle. Wir wollen schnell zur praktischen Arbeit kommen. An dieser Stelle sollte jedoch eine Ausblick nicht fehlen, wie wir uber die einschrankenden Annahmen des linearen Modells hinauskommen. Die linearen Modelle gehoren zu den am besten untersuchten Modellen. Theorie und Algorithmen hierfur sind weit entwickelt. Von daher ist es naheliegend, zu probieren, wieweit sich die Modellklasse so erweitern lasst, dass theoretische und algorithmische Erfahrungen noch nutzbar sind.

Wir notierten das lineare Modell als

$$\begin{aligned} Y &= m(X) + " \\ & \quad Y \text{ mit Werten in } \mathbb{R}^n \\ & \quad X \ 2 \ \mathbb{R}^{n \ p} \\ & \quad E(") = 0 \\ & \quad \text{mit } m(X) = X ; \qquad 2 \ \mathbb{R}^p: \end{aligned}$$

Eine wichtige Erweiterung ist, die Bedingung der Linearitat aufzuheben. Sie wird abgemildert mit einer Zwischenstufe. Wir setzen also nicht mehr voraus, dass m linear ist, sondern nur, dass es sich uber eine lineare Funktion faktorisieren lasst. Dies ergibt ein verallgemeinertes lineares Modell

$$\begin{aligned} Y &= m(X) + " \\ & \quad Y \text{ mit Werten in } \mathbb{R}^n \\ & \quad X \ 2 \ \mathbb{R}^{n \ p} \\ & \quad E(") = 0 \\ & \quad m(X) = \bar{m}( ) \text{ mit } \quad = X ; \qquad 2 \ \mathbb{R}^p: \end{aligned}$$

Die nachste naheliegende Verallgemeinerung ist, eine Transformation auf Y zu berucksichtigen. Zahlreiche weitere Abschwachungen sind diskutiert worden; eine kleine Anzahl hat sich als handhabbar erwiesen. Die verbliebenen Modelle werden als generalisierte lineare Modelle bezeichnet. Generalisierte lineare Modelle haben in R eine weitgehende Unterstutzung. In der Regel nden sich zu den hier diskutierten R-Funktionen fur lineare Modelle eine Entsprechung fur generalisierte lineare Modelle. Weitere Information mit help(glm) .

2.5.2. Lokale Regression. Wir machen nun einen gro en Sprung. Wir haben lineare Modelle diskutiert. Wir wissen, dass damit auch nichtlineare Funktionen modelliert werden konnen. Aber die Terme, die in die Funktion eingehen, mussen vorab spezi ziert werden. Zu viele Terme fuhren zu einer Uberanpassung. Die statistische Behandlung von Regressionsproblemen mit geringen Einschankungen an die Modellfunktion bleibt ein Problem.

Ein partieller L osungsansatz kommt aus der Analysis. Dort ist es eine Standard-Technik, Funktionen lokal zu approximieren. Das analoge Vorgehen in der Statistik ist, anstelle eines globalen Schatzverfahrens eine lokalisierte Variante zu wahlen. Wir nehmen immer noch an, dass

$$\begin{aligned} Y &= m(X) + \varepsilon \quad Y \in \mathbb{R}^n \\ & X \in \mathbb{R}^{n \times p} \\ & E(\varepsilon) = 0; \end{aligned}$$

aber wir nehmen Linearitat nur lokal an:

$$m(x) \approx x^0 \beta_{x_0} \quad \beta_{x_0} \in \mathbb{R}^p \text{ und } x \approx x_0:$$

Wenn wir praktisch arbeiten wollen, reicht abstrakte Asymptotik nicht. Wir m ussen das spezi zieren. Dies kann skalenspezi sch geschehen (z.B. $x \approx x_0$ wenn $|x - x_0| < 3$) oder designabhangig (z.B. $x \approx x_0$ wenn $\#\{i : |x - x_i| \leq |x - x_0|\} < n/3$). Die heute ublichen Implementierungen haben feinere Varianten, die hier noch nicht diskutiert werden konnen. Der Illustration halber kann die folgende Vergroberung reichen:

Lokalisierter Gau -Markov-Sch atzer:

Fur $x \in \mathbb{R}^p$, bestimme

$$\delta = \min_d \{d : (\#\{i : |x - x_i| \leq d\} \geq n \cdot f\}$$

wobei f ein gewahlter Anteil (z.B. 0.5) ist.

Bestimme den Gau -Markov-Schatzer $\hat{\beta}_x$, wobei nur diejenigen Beobachtungen einbezogen werden, fur die $|x - x_i| \leq \delta$.
Schatze

$$\hat{m}(x) = x^0 \hat{\beta}_x:$$

Diese Vergroberung ignoriert alle Messpunkte, die einen Abstand $> \delta$ haben. Feinere Methoden benutzen eine Gewichtung, um den Ein uss entfernter Messpunkte zunehmend zu reduzieren.

---

help (loess)

`loess` *Local Polynomial Regression Fitting*

Description.

Fit a polynomial surface determined by one or more numerical predictors, using local tting.

Usage.

```
loess(formula, data, weights, subset, na.action, model = FALSE,
      span = 0.75, enp.target, degree = 2,
      parametric = FALSE, drop.square = FALSE, normalize = TRUE,
      family = c("gaussian", "symmetric"),
      method = c("loess", "model.frame"),
      control = loess.control(...), ...)
```

Arguments.

`formula`	a formula specifying the numeric response and one to four numeric predictors (best speci ed via an interaction, but can also be speci ed additively).
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `loess` is called.
`weights`	optional weights for each case.
`subset`	an optional speci cation of a subset of the data to be used.
`na.action`	the action to be taken with missing values in the response or predictors. The default is given by `getOption("na.action")`.
`model`	should the model frame be returned?
`span`	the parameter which controls the degree of smoothing.
`enp.target`	an alternative way to specify `span`, as the approximate equivalent number of parameters to be used.
`degree`	the degree of the polynomials to be used, up to 2.
`parametric`	should any terms be tted globally rather than locally? Terms can be speci ed by name, number or as a logical vector of the same length as the number of predictors.
`drop.square`	for ts with more than one predictor and `degree=2`, should the quadratic term (and cross-terms) be dropped for particular predictors? Terms are speci ed in the same way as for `parametric`.
`normalize`	should the predictors be normalized to a common scale if there is more than one? The normalization used is to set the 10% trimmed standard deviation to one. Set to false for spatial coordinate predictors and others know to be a common scale.
`family`	if `"gaussian"` tting is by least-squares, and if `"symmetric"` a redescending M estimator is used with Tukey's biweight function.
`method`	t the model or just extract the model frame.
`control`	control parameters: see `loess.control`.
`...`	control parameters can also be supplied directly.

Details.

Fitting is done locally. That is, for the t at point x, the t is made using points in a neighbourhood ofx, weighted by their distance from x (with di erences in 'parametric' variables being ignored when computing the distance). The size of the neighbourhood is controlled by (set by `span` or `enp.target`). For < 1, the neighbourhood includes proportion of the points, and these have tricubic weighting (proportional to (1 (dist=maxdist)^3)^3. For > 1, all points are used, with the 'maximum distance' assumed to be $^{1=p}$ times the actual maximum distance for p explanatory variables.

For the default family, tting is by (weighted) least squares. For `family="symmetric"` a few iterations of an M-estimation procedure with Tukey's biweight are used. Be aware that as the initial value is the least-squares t, this need not be a very resistant t.

It can be important to tune the control list to achieve acceptable speed. See `loess.control` for details.

Value.

An object of class `"loess"`.

Note.

As this is based on the `cloess` package available at `netlib`, it is similar to but not identical to the `loess` function of S. In particular, conditioning is not implemented.

The memory usage of this implementation of `loess` is roughly quadratic in the number of points, with 1000 points taking about 10Mb.

Author(s).

B.D. Ripley, based on the `cloess` package of Cleveland, Grosse and Shyu avaliable at `http://www.netlib.org/a/`.

References.

W.S. Cleveland, E. Grosse and W.M. Shyu (1992) Local regression models. Chapter 8 of *Statistical Models in S* eds J.M. Chambers and T.J. Hastie, Wadsworth & Brooks/Cole.

See Also.

`loess.control`, `predict.loess`.

`lowess`, the ancestor of `loess` (with different defaults!).

Examples.

```
cars.lo <- loess(dist ~ speed, cars)
predict(cars.lo, data.frame(speed = seq(5, 30, 1)), se = TRUE)
# to allow extrapolation
cars.lo2 <- loess(dist ~ speed, cars,
  control = loess.control(surface = "direct"))
predict(cars.lo2, data.frame(speed = seq(5, 30, 1)), se = TRUE)
```

Während die lineare Regression durch die Modellannahmen verpflichtet ist, immer ein lineares (oder linear parametrisiertes) Bild zu geben, können bei einer lokalisierten Variante auch Nichtlinearitäten dargestellt werden. Die Untersuchung dieser Familie von Verfahren bildet ein eigenes Teilgebiet der Statistik, die nichtparametrische Regression.

Wir bereiten wieder ein Beispiel vor:

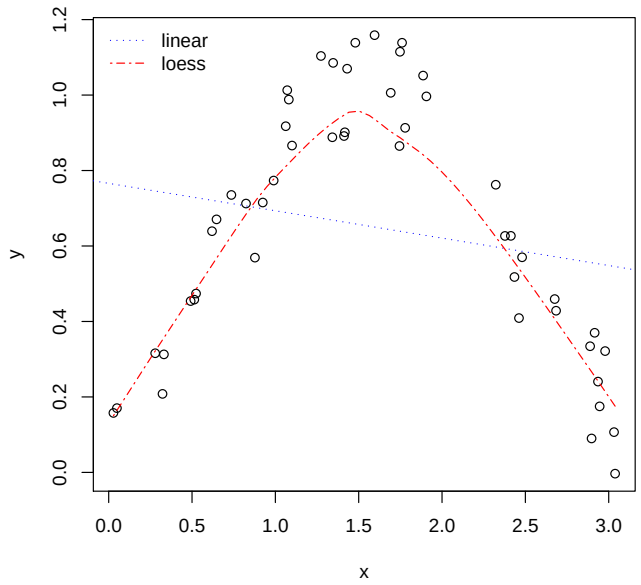
Eingabe

```
x <- runif(50) * pi
y <- sin(x)+rnorm(50)/10
```

Beispiel 2.11:

Eingabe

```
plot(x, y)
abline(lm(y ~ x), lty = 3, col = "blue")
lines(loess.smooth(x, y), lty = 6, col = "red")
legend("topleft",
       legend = c("linear", "loess"),
       lty = c(3, 6), col = c("blue", "red"), bty = "n")
```



2.6. Erg anzungen

2.6.1. Erg anzung: Diskretisierungen. Analog zum Vorgehen bei den Histogrammen konnen wir wieder diskretisieren. Im Hinblick auf die Regressoren haben wir dies beim Helikopter-Beispiel getan. Die Diskretisierung konnen wir auch bei der Respons vornehmen. Damit wird aus dem Regressionsproblem ein Kontingenztafel-Problem. Wir gehen hier nicht weiter auf diese Moglichkeit ein.

2.6.2. Erg anzung: Externe Daten. Daten, wie auch andere R-Objekte konnen mit save() in eine externe Datei geschrieben und mit load() wieder daraus gelesen werden. In diesen Dateien werden die Daten komprimiert; die Dateien sind zwischen verschiedenen R-Systemen austauschbar.

Ein- Ausgabe von Daten f ur R	
save()	Speichert Daten in eine externe Datei. Aufruf: save(h`Namen der zu speichernden Objektei`, file = h`Dateinamei`, ...)

(Fortsetzung)!

Ein- Ausgabe von Daten f ur R (Fortsetzung)	
load()	Ladt Daten aus einer externe Datei. Aufruf: load(file = hDateinamei, ...)

Hau g werden Daten in anderen Systemen vorbereitet.R stellt eine Reihe von Funktionen bereit, die Daten in unterschiedlichen Formaten einlesen konnen. Siehe dazu im Anhang Abschnittt A.17 auf Seite Seite A-29. Weitere Information ndet sich im Manual \Data Import/Export" ([R D07b]).

Die Funktion data() bundelt verschieden Zugri sroutinen, wenn die Zugri spfade und Datei-Namen den R-Konventionen folgen.

In der Regel mussen eingelesene Daten noch nachbearbeitet werden, um das Format an die Aufrufkonventionen der gewunschten R-Funktionen anzupassen.

So erwartet z. B. lm die Regressoren als getrennte Variable. Fur faktorielle Designs ist es hingegen ublich, die Resultate in einer Tafel zusammenzufassen, die die Faktor-Stufen als Zeilen- oder Spaltenlabels enthalt. Die Funktion stack() uberfuhrt Tafeln in Spalten.

2.6.3. Erg anzung: Software-Test. Alle vorbereiteten Algorithmen, wie hier die Algorithmen zu den linearen Modellen und deren Varianten, sollten mit derselben Vorsicht behandelt werden wie mathematische Veroentlichungen und Zitate. Selbst einfache Programme jedoch haben schnell eine semantische Komplexitat, die weit uber die mathematischer Beweise hinaus geht. Die ubliche Strategie des \Nachrechnens" oder des schrittweisen Nachvollziehens verbietet sich dadurch. Anstelle einer vollstandigen Uberprufung muss ein selektives Testen treten. Eine Teststrategie ist z.B. in [Sawitzki, 1994] beschrieben.

Die Uberprufung ist sowohl fur die Implementierung als auch fur den zu Grunde liegenden abstrakten Algorithmus notig.

Aufgabe 2.18	
	Fur diese Aufgabenserie sey $y_i = a + bx_i + \varepsilon_i$ mit ε_i iid $N(0; \sigma^2)$ und $x_i = i; i = 1; : : : ; 10$.
	Wahlen Sie eine Strategie, um lm() im Hinblick auf den Parameterraum $(a; b; \sigma^2)$ zu uberprufen. Gibt es eine naheliegende Zellzerlegung auf die einzelnen Parameter $a; b; \sigma^2$? Welche trivialen Falle gibt es? Welche (uniforme) Asymptotik? Wahlen Sie Testpunkte jeweils in der Mitte jeder Zelle und an den Randern. Fuhren Sie diese Test durch und fassen Sie die Resultate zusammen.
	Welche Symmetrien/Antisymmetrien gibt es? Uberprufen Sie diese Symmetrien.
	Welche Invarianten/welches Covariate Verhalten gibt es? Uberprufen Sie diese Invarianten/Covariaten.

Aufgabe 2.19	
	Fur diese Aufgabenserie sey $y_i = a + bx_i + \varepsilon_i$ mit ε_i iid $N(0; \sigma^2)$.
	Welche extremen Designs $x(x_i)$ gibt es? Uberprufen Sie das Verhalten von lm() bei vier extremalen Designs.
	Fuhren Sie die Aufgaben aus der letzten Gruppe aus, jetzt mit variablem Design. Fassen Sie Ihren Bericht zusammen.

Aufgabe 2.20	
	Fur diese Aufgabenserie sey $y_i = a + bx_i + \varepsilon_i$ mit ε_i iid $N(0; \sigma^2)$.
	Modi zieren Sie lm() so, dass eine gesicherte Funktion fur das einfache lineare Modell entsteht, die auch Abweichungen von den Modellannahmen untersucht.

2.6.4. R-Datentypen. R ist eine interpretierte Programmiersprache. Sie will es dem Anwender erlauben, De nitionen und Konkretisierungen exibel zu handhaben. Aus Geschwindigkeitsgrunden versucht R, Auswertungen so spat wie moglich durchzufuhren. Dies erfordert einige Einschrankungen an die Sprache, die R von anderen Programmiersprachen unterscheidet.

R kennt keine abstrakten Datentypen. Ein Datentyp ist durch seine Instanzen, die Variablen, de niert.

Der Datentyp einer Variablen ist dynamisch: derselbe Name in denselben Kontext kann zu unterschiedlichen Zeiten unterschiedliche Variablenwerte und Variablentypen kennzeichnen.

Dennoch hat zu jeder Zeit eine Variable einen bestimmten Typ. Das R-Typensystem versteht man jedoch am besten in seiner historischen Entwicklung und die entsprechenden Funktionen. In der ersten Stufe war der Typ beschrieben durch mode() (z.B. \numeric") und storage.mode() (z.B. \integer" oder \real").

Beide Funktionen sind weitgehend durch typeof() abgelost. Eine Zusammenfassung der Typen, die durch typeof() derzeit berichtet werden, ist in [R D07c] zu nden.

Komplexere Datentypen werden auf die in [R D07c] de nierten zur uckgefuhrt, indem die Variablen mit Attributen versehen werden. Dies geschieht mit der Funktion attr() , die auch benutzt werden kann, um Attribute zu inspizieren. So sind eine Matrix oder ein Array nur spezielle Vektoren, die sich dadurch auszeichnen, dass sie ein dim -Attribut haben. Das class -Attribut dient dazu, die Klasse explizit zu festzulegen.

Fur die wesentlichen Typen sind Inspektionsprozeduren und Umwandlungsprozeduren: is.$\langle$typi() pruft auf Typenzugehorigkeit, as.$\langle$typi() wandelt den Typ.

2.6.5. Klassen und polymorphe Funktionen. Im Zuge der Weiterentwicklung wurde eine Anleihe an objekt-orientierte Programmierung gemacht. Dafur wurde ein spezielles Attribut mit dem Namen class genutzt: der Name des Typs (oder der \Klasse") wird hier gespeichert. Multiple Klassenzugehorigkeit in einer Hierarchie von Klassen ist auch moglich. In diesem Fall enthalt class einen Vektor von Klassennamen. So hat zum Beispiel ein geordneter Faktor die Kennzeichnung class = c("ordered", factor") . Zur Verwaltung der Klassen stehen Funktionen class() , unclass() , inherits() zur Verfugung.

Die Klassenzuordnung basiert dabei auf Vertrauen. R überprüft nicht, ob die Datenstruktur mit der angegebenen Klasse konsistent ist.

Funktionen wie plot() , print() und viele weitere überprüfen die Typen- und Klassenzugehörigkeit ihrer Argument und verzweigen dann zu entsprechenden spezialisierten Funktionen. Dieses nennt man Polymorphismus. Wenn man eine polymorphe Funktion aufruft, erhält man zunächst nur den Hinweise, das eine Dispatch-Funktion UseMethod() aufgerufen wird. Beispiel:

Beispiel 2.12:

Eingabe

```
plot
```

Ausgabe

```
function (x, y, ...)
{
    if (is.function(x) && is.null(attr(x, "class"))) {
        if (missing(y))
            y <- NULL
        hasylab <- function(...) !all(is.na(pmatch(names(list(...)),
            "ylab")))
        if (hasylab(...))
            plot.function(x, y, ...)
        else plot.function(x, y, ylab = paste(deparse(substitute(x)),
            "(x)"), ...)
    }
    else UseMethod("plot")
}
<environment: namespace:graphics>
```

UseMethod() bestimmt die Klasse des ersten Argument, mit dem die Funktion aufgerufen wurde, sucht dann ein Spezialisierung für diese Klasse und ruft schliesslich die gefundene Funktion auf. Für polymorphe Funktionen findet man die entsprechenden Spezialisierungen mit Hilfe von methods() , z.B. methods(plot) .

2.6.6. Extraktor-Funktionen. Funktionen wie lm() liefern komplexe Datentypen mit umfangreicher Information. In einer rein objekt-orientierten Umgebung wurden Zugriffsmethoden mit den Daten gemeinsam verkapselt. In R ist Objekt-Orientierung in Ansätzen und auf verschiedene Weisen realisiert. Dies spiegelt zum Teil die Entwicklung wieder. Bei genügend verallgemeinerbaren Strukturen werden Zugriffsmethoden wie in Abschnitt 2.6.5 bereitgestellt. Für die Objekte wie die von lm() gelieferten gibt es eine Reihe von Extraktor-Funktionen, die auf Komponenten zugreifen und diese geeignet aufbereiten.

Extraktor-Funktionen für `lm`	
coef()	extrahiert geschätzte Koeffizienten
effects()	extrahiert sukzessiv orthogonale Komponenten

(Fortsetzung)!

Extraktor-Funktionen f ur `lm` (Fortsetzung)	
residuals()	Roh-Residuen
stdres()	(in library(MASS)) standardisierte Residuen
studres()	(in library(MASS)) extern studentisiere Residuen
fitted()	
vcov()	Varianz/Kovarianzmatrix der geschatzten Parameter
predict()	Kon denz- und Toleranzintervalle
confint()	Kon denz-Intervalle f ur Parameter
influence()	extrahiert Ein uss-Diagnostiken
model.matrix()	bildet die Design-Matrix

2.7. Statistische Zusammenfassung

Als Leitbeispiel diente in diesem Kapitel die statistische Analyse eines funktionellen Zusammenhangs. Die betrachteten Modelle sind nit in dem Sinne, dass ein endlichdimensionaler Funktionenraum den in Betracht gezogenen Zusammenhang zwischen Regressoren und Respons beschreibt. Die stochastische Komponente in diesen Modellen ist noch auf eine (eindimensionale) Zufallsverteilung beschankt. Die Dimensionsbegri e verdienen hier eine genauere Betrachtung. Wir haben zum einen die Regressor-Dimension. Dies ist die Dimension des Raumes der beobachteten oder abgeleiteten Parameter. Nicht alle Parameter sind identi zierbar oder schatzbar. Genauer gefasst ist die Dimension die Vektorraum-Dimension des gewahlten Modell-Raums. Die Modelle werden durch Parameter in diesem Raum beschrieben. Diese Parameterknnen unbekannt oder hypothetisch sein. In jedem Fall aber haben wir sie als deterministisch betrachtet. Zum anderen haben wir die stochastische Komponente, repasentiert durch den Fehler-Term. In diesem Kapitel sind wir von homogenen Fehlern ausgegangen. Damit bestimmt der Fehler-Term im Prinzip eine Dimension, die allerdings aus einem Raum von Verteilungen stammt. Fur den Spezialfall der einfachen Gau -linearen Modell sind die Verteilungen mit zwei Parametern prazisiert, dem Erwartungswert und der Varianz. Von dem Erwartungswert haben wir uns durch die Annahme befreit, dass das Modell im Mittel alle systematischen E ekte erfasst, also der Erwartungswert null ist. Die Varianz ist in unseren Problemen noch ein unbekannter Storparameter. Wir haben die dadurch entstehenden Problemen vermieden, indem wir uns auf Probleme beschankt haben, in denen diese Storparameter durch einen geschatzten Wert ersetzt und so eliminiert wird.

2.8. Literatur und weitere Hinweise:

[CH92] Chambers, J.M.; Hastie, T.J. (eds.) (1992): Statistical Models in S. NewYork: Chapman & Hall.

[J r93] J rgensen, B. (1993): The Theory of Linear Models. NewYork: Chapman & Hall.

[R D07c] R Development Core Team (2004): TheR language de nition.

[Saw94a] Sawitzki, G. (1994): Numerical Reliability of Data Analysis Systems. Computational Statistics & Data Analysis 18.2 (1994) 269-286.<http://www.statlab.uni-heidelberg.de/reports/>.

[Saw94b] Sawitzki, G. (1994):Report on the Numerical Reliability of Data Analysis Systems. Computational Statistics & Data Analysis/SSN 18.2 (1994) 289-301<http://www.statlab.uni-heidelberg.de/reports/>.

KAPITEL 3

Vergleich von Verteilungen

Wir beginnen mit der Konstruktion eines kleinen Werkzeugs, das uns Beispieldaten liefern wird. Basis ist ein kleiner Reaktionstester. Wir zeichen einen \zufälligen" Punkt, warten auf einen Maus-Klick, und registrieren die Position des Mauszeigers. Damit bei wiederholten Aufrufen das Bild stabil bleibt, fixieren wir das Koordinatensystem.

Beispiel 3.1:

Eingabe

```
plot(x = runif(1), y = runif(1),
     xlim = c(0, 1), ylim = c(0, 1),
     main = "Bitte auf den Punkt klicken",
     xlab = '', ylab = '',
     axes = FALSE, frame.plot = TRUE)
locator(1)
```

Ausgabe

```
$x
[1] 0.6956522

$y
[1] 0.1260563
```

Wir verpacken nun diesen Basistester. Wir merken uns die Koordinaten, versuchen, die Reaktionszeit des Benutzers zu messen, und liefern alle Resultate als Liste zurück.

Beispiel 3.2:

Eingabe

```
click1 <- function(){
    x <- runif(1);y <- runif(1)
    plot(x = x, y = y, xlim = c(0, 1), ylim = c(0, 1),
        main = "Bitte auf den Punkt klicken",
        xlab = '', ylab = '',
        axes = FALSE, frame.plot = TRUE)
    clicktime <- system.time(xyclick <- locator(1))
    list(timestamp = Sys.time(),
        x = x, y = y,
        xclick = xyclick$x, yclick = xyclick$y,
        tclick = clicktime[3])
}
```

Zur weiteren Verarbeitung konnen wir die Liste in einen data.frame integrieren und diesen data.frame schrittweise mit Hilfe von rbind erweitern.

Beispiel 3.3:

Eingabe

```
dx <- as.data.frame(click1())
dx <- rbind(dx, data.frame(click1()))
dx
```

Ausgabe

```
                  timestamp       x       y  xclick  yclick tclick
elapsed  2008-03-17 21:40:49 0.29683 0.43955 0.69565 0.12606  0.261
elapsed1 2008-03-17 21:40:50 0.29617 0.58226 0.69565 0.12606  0.262
```

Aufgabe 3.1	
	Definieren Sie eine Funktion click(runs), die zu vorgegebener Anzahl runs die Aufgabe von click1() wiederholt und das Resultat als data.frame ubergibt. Eine erste (zusatzliche) Messung sollte als Warmlaufen betrachtet werden und nicht in die Auswertung mit einbezogen werden. Wahlen Sie eine Anzahl runs. Begrunden Sie Ihre Wahl von runs. Fuhren Sie click(runs) durch und speichern Sie das Resultat mit Hilfe von write.table() in einer Datei. Stellen Sie die Verteilung der Komponente tclick() mit den Methoden aus Kapitel 1 (Verteilungsfunktion, Histogramm, Boxplot) dar.

3.1. Shift/Skalenfamilien

Ein Vergleich von Verteilungen kann eine sehr anspruchsvolle Aufgabe sein. Der mathematische Raum, in dem Verteilungen angesiedelt sind, ist nicht mehr ein Zahlenraum oder ein (endlichdimensionaler) Vektorraum. Der eigentliche Raum, in dem Verteilungen beheimatet sind, ist ein Raum von Ma en. In einfachen Fallen, etwa bei Verteilungen auf $\mathbb{R}$, konnen wir alles auf Verteilungsfunktionen reduzieren und sind damit immerhin bei einem Funktionenraum. Selbst hier kann ein Vergleich noch gro e Schwierigkeiten machen. Wir haben keine einfache Ordnungsrelation.

Aufgabe 3.2	
	Fuhren Sie Aufgabe 3.1 einmal mit der rechten und einmal mit der linken Hand durch. Vergleichen Sie die empirischen Verteilungen von tclick() . Die erhobenen Daten enthalten auch Informationuber die Positionen. De nieren Sie ein Ma dist fur die Abweichung. Begrunden Sie Ihre De nition. F uhren Sie auch fur dist einen rechts/links Vergleich durch.

Wir konzentrieren uns hier auf den Vergleich von nur zwei Verteilungen, etwa der von Messungen in zwei Behandlungsgruppen. Wie nehmen wieder einen einfachen Fall: die Beobachtungen seien jeweils unabhangig identisch verteilt (jetzt mit der f ur den Vergleich von Behandlungen ublichen Index-Notation).

Y_{ij} unabhangig identisch verteilt mit Verteilungsfunktion F_i

$i = 1; 2$ Behandlungen

$j = 1; : : : ; n_i$ Beobachtungen in Behandlungsgruppe i.

Wie vergleichen wir die Beobachtungen in den Gruppen $i = 1; 2$? Die (einfachen) linearen Modelle

$$Y_{ij} = + _i + "_{ij}$$

betrachten als Unterschied hau g nur eine Verschiebung $= _1 \quad _2$.

Bezeichnungen: Zu einer Verteilung mit Verteilungsfunktion F hei t die Familie mit

$$F_a(x) = F(x \quad a)$$

die Shift-Familie zu F. Die Verschiebung a hei t Shift- oder Lage-Parameter.

Die Behandlung kann aber, in Wahrscheinlichkeiten gesprochen, die Wahrscheinlichkeitsmassen auch in anderer Weise verschieben, als es ein additiver Term im Modell bewirken kann. Wir brauchen allgemeinere Vergleichsmoglichkeiten als die durch einen Shift de nierten.

Bezeichnung: Eine Verteilung mit Verteilungsfunktion F_1 hei t stochastisch kleiner als eine mit Verteilungsfunktion $F_2 (F_1 \quad F_2)$, wenn F_1 eher bei kleineren Werten liegt als F_2. Das bedeutet, dass F_1 eher ansteigt.

$$F_1(x) \quad F_2(x) \ 8x$$

und

$$F_1(x) > F_2(x) \text{ fur mindestens ein } x:$$

Fur Shift-Familien gilt: Ist $a < 0$, so ist $F_a \quad F$. Der Shift bewirkt eine Parallel-Verschiebung der Verteilungsfunktionen.

Ein typisches Resultat des Click-Experiments (Aufgabe 3.1 ist zum Beispiel in Abbildung 3.1 dargestellt. Die Zeiten für die rechte Seite sind stochastisch kleiner als die für die linke Seite. Die Verteilungen gehören jedoch nicht zu einer Shift-Familie, denn die Verteilungsfunktionen sind nicht parallel.

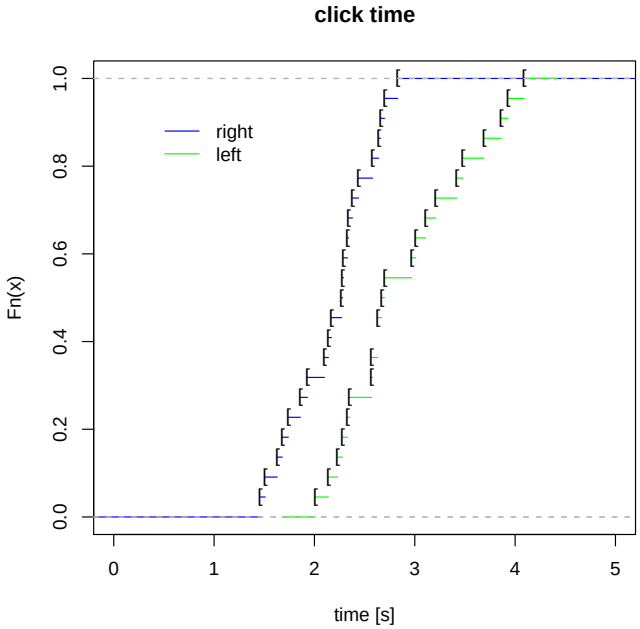


Abbildung 3.1. Verteilungsfunktion für die rechts/links-Klickzeit

Aufgabe 3.3	
	Wie sieht ein PP-Plot für F_1 gegen F_2 aus, wenn F_1 F_2? Wie sieht ein QQ-Plot für F_1 gegen F_2 aus, wenn F_1 F_2?

Leider ist die dadurch definierte stochastische Ordnung nur von beschränktem Wert. Sie definiert keine vollständige Ordnung. Für Shift-Familien ist sie ausreichend. Aber Gegenbeispiele kann man sich konstruieren, wenn man die Shift-Familien nur geringfügig erweitert.

Bezeichnungen: Zu einer Verteilung mit Verteilungsfunktion F heißt die Familie mit

$$F_{a;b}(x) = F\left(\frac{x - a}{b}\right)$$

die Skalen-Shiftfamile zu F.

Aufgabe 3.4	
	Die Skalen-Shiftfamilien zur $N(0;1)$-Verteilung sind die $N(\ ;\ ^2)$-Verteilungen. Welche $N(\ ;\ ^2)$-Verteilungen sind stochastisch kleiner als die $N(0;1)$-Verteilung? Welche sind stochastisch größer? Für welche ist die Ordnungsrelation unde niert?

Die aus der linearen Theorie kommende Einordnung nach Lage/Skalen und die stochastische Ordnung klaffen auseinander, und beide Aspekte müssen oft getrennt betrachtet werden. Viele statistische Methoden konzentrieren sich auf Aspekte, die durch Skalen-Shiftfamilien motiviert sind. Unterschiede jenseits dessen, was durch Skala und Shift beschrieben werden kann, bedürfen oft besonderer Aufmerksamkeit.

In Kapitel 2 haben wir eine typische Situation für lineare Modelle betrachtet. Im Prinzip haben wir es mit Skalen-Shiftfamilien zu tun. Der (stochastische) Skalenparameter in diesen Modellen ist jedoch nur ein Störparameter, der eliminiert werden kann. Dazu benutzten wir einen Schätzer für diesen Skalenparameter, die residuelle Varianz, die wir dann heraus gekürzt haben. Als eine Besonderheit bei Gauß-linearen Modellen erhalten wir hier unabhängige Schätzer für Erwartungswert und Varianz. Dadurch können wir im Falle der einfachen Gauß-linearen Modelle Statistiken gewinnen, die nicht mehr vom Skalenparameter abhängen.

Im allgemeinen Fall haben wir jedoch eine aufsteigende Leiter von Problemen:

Shift-Alternativen
Shift/Skalen-Alternativen
stochastische Ordnung
allgemeinere Alternativen

Test- und Schätzprobleme konzentrieren sich oft nur auf einen Aspekt des Problems, die Lage. Der Skalenparameter ist hier nur eine Störgröße, ein "nuisance parameter". Unterschiede im Shift-Parameter führen zu stochastisch monotonen Beziehungen. Unterschiede im Skalenparameter sind nicht so einfach einzuordnen und Test-Statistiken müssen erst von diesem, Störparameter bereinigt werden, wenn ausser dem Shift-Parameter auch der Skalenparameter variieren kann.

3.2. QQ-Plot, PP-Plot

Als Vergleichsdarstellung für Verteilungsfunktionen haben wir den PP-Plot und den QQ-Plot kennengelernt. So lange man innerhalb einer Skalen-Shiftfamilie bleibt, hat der QQ-Plot zumindest in einer Hinsicht einen Vorteil gegenüber dem PP-Plot:

Bemerkung 3.1. Sind $F_1; F_2$ Verteilungsfunktionen aus einer gemeinsamen Skalen-Shiftfamilie, so ist der QQ-Plot von F_1 gegen F_2 eine Gerade.

Insbesondere für die Gauß verteilungen ist der QQ-Plot gegen $N(0;1)$ ein wichtiges Hilfsmittel. Jede Gauß verteilung gibt in diesem Plot eine Gerade. Der QQ-Plot ist für diese Situation bereits als Funktion qqnorm() vorbereitet.

Für den Vergleich von zwei Stichproben mit gleichem Stichprobenumfang kann die entsprechende Funktion qqplot() genutzt werden: bezeichnen wir die empirischen Quantile mit $Y_{1;(i:n)}$ bzw. $Y_{2;(i:n)}$, so ist dieser Plot der Graph $Y_{1;(i:n)}; Y_{2;(i:n)}\ _{i=1:::n}$. Sind die Stichprobenumfange verschieden, so behilft sich R und generiert die Markierungspunkte durch lineare Interpolation, wobei der kleinere der beiden Stichprobenumfänge die Anzahl der Interpolationspunkte bestimmt.

Der PP-Plot hat keine dem QQ-Plot vergleichbare Aquivarianzeigenschaften. Wenn wir Skalen-Shiftparameter eliminieren wollen, müssen wir die Daten zunächst entsprechend transformieren. Die mathematische Theorie ist jedoch für den PP-Plot einfacher. Insbesondere gibt es auch hier einen entsprechenden Kolmogorov-Smirnov-Test (siehe Abschnitt 3.2.1).

Der Aquivarianz des QQ-Plots als Vorteil stehen auf der anderen Seite strukturelle Nachteile entgegen. In Bereichen niedriger Dichte bestimmen empirisch wenige Datenpunkte den Plot. Entsprechend hat er hier eine gro e Varianz. Gleichzeitig sind hier der Wahrscheinlichkeit nach benachbarte Quantile im Wertebereich weit entfernt: die gro e Varianz kombiniert sich ungunstig mit einer gro en Variabilit at, und der QQ-Plot zeigt entsprechend gro e Fluktuation. Fur die meisten Lehrbuch-Verteilungen bedeutet dies, dass der QQ-Plot in den Randbereichen kaum zu interpretieren ist. Der P P -Plot hat keine entsprechenden Skalende zite, aber auch nicht die Aquivarianzeigenschaft des QQ-Plots. Er wird deshalb in der Regel auf geeignet standardisierte Variable angewandt.

---

help (qqplot)

`qqnorm` *Quantile-Quantile Plots*

Description.

`qqnorm` is a generic function the default method of which produces a normal QQ plot of the values in `y`. `qqline` adds a line to a normal quantile-quantile plot which passes through the rst and third quartiles.

`qqplot` produces a QQ plot of two datasets.

Graphical parameters may be given as arguments to `qqnorm`, `qqplot` and `qqline`.

Usage.

```
qqnorm(y, ...)
## Default S3 method:
qqnorm(y, ylim, main = "Normal Q-Q Plot",
       xlab = "Theoretical Quantiles", ylab = "Sample Quantiles",
       plot.it = TRUE, datax = FALSE, ...)

qqline(y, datax = FALSE, ...)

qqplot(x, y, plot.it = TRUE, xlab = deparse(substitute(x)),
       ylab = deparse(substitute(y)), ...)
```

Arguments.

`x`	The rst sample for `qqplot`.
`y`	The second or only data sample.
`xlab, ylab, main`	plot labels. The `xlab` and `ylab` refer to the y and x axes respectively if `datax = TRUE`.
`plot.it`	logical. Should the result be plotted?
`datax`	logical. Should data values be on the x-axis?
`ylim, ...`	graphical parameters.

Value.

For `qqnorm` and `qqplot`, a list with components

`x`	The x coordinates of the points that were/would be plotted
`y`	The original `y` vector, i.e., the corresponding y coordinates *including* NAs.

References.

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also.

`ppoints`, used by `qqnorm` to generate approximations to expected order statistics for a normal distribution.

Examples.

```
y <- rt(200, df = 5)
qqnorm(y); qqline(y, col = 2)
qqplot(y, rt(300, df = 5))

qqnorm(precip, ylab = "Precipitation [in/yr] for 70 US cities")
```

Aufgabe 3.5	
	Benutzen Sie den Quantil-Quantil-Plot, um die Resultate des rechts/links click -Experiments zu vergleichen. Formulieren Sie die Resultate.
	Fassen Sie die rechts/linksclick -Daten zu einem Vektor zusammen. Vergleichen Sie den Quantil-Quantil-Plot mit dem von Monte-Carlo-Stichproben aus dem zusammengefassten Vektor. Erinnerung: Zufallsstichproben konnen Sie mit sample() ziehen. Mit par(mfrow = c(2, 2)) teilen Sie den Zeichenbereich so ein, dass Sie vier Plots gleichzeitig sehen konnen.
**	Benutzen Sie beisample() den Parameterwert replace = FALSE Wie mussen Sie jetztsample() anwenden, um den zusammengefassten Vektor in zwei Vektoren mit Monte-Carlo-Stichproben aufzuteilen? Welche Unterschiede zureplace = TRUE sind zu erwarten?

Aufgabe 3.6	
	Bestimmen Sie fur die tclick -Daten des rechts/links click -Experiments Skalen- und Shiftparameter so, dass die Verteilungen in den Gruppen nach Skalen-Shift-Transformation moglichst gut ubereinstimmen. Beschreiben Sie die Unterschiede anhand der Skalen-Shiftparameter. Verwenden Sie dazu eine Modellierung mit einem linearen Modell.
	Benutzen Sie die Funktion boxplot() , um Quartile und Flankenverhalten darzustellen. Vergleichen Sie die Information mit den Skalen-Shiftparametern. *Hinweis:* was entspricht dem Shift(Lage)parameter? Was entspricht dem Skalenparameter?

Wenn Darstellungen a n invariant sind, k onnen Skalen-Shiftparameter ignoriert werden. Wenn Darstellungen nicht a n invariant sind, ist es h au g hilfreich, zun achst Skalen-Shiftparameter geeignet zu sch atzen, die Verteilungen zu standardisieren, und dann die standardisierten Verteilungen zu untersuchen.

Das Problem, das wir uns damit potentiell einhandeln, ist, dass dann das stochastische Verhalten der Sch atzung fur die Skalen-Shiftparameter ber ucksichtigt werden muss. Der ubliche Ausweg ist es, vorsichtigerweise \konservative" Tests und robuste Sch atzer zu benutzen. Die folgende Transformation versucht, Skala und Lage an eine Standard-Normalverteilung anzupassen.

Eingabe

```
ScaleShiftStd <- function (x) {
    xq <- quantile(x[!is.na(x)], c(0.25, 0.75))
    y <- qnorm(c(0.25, 0.75))
    slope <- diff(y)/diff(xq)
    (x-median(x, na.rm = FALSE)) * slope
}
```

Um Verteilungen direkt miteinander vergleichen zu k onnen, greifen wir auf Techniken aus dem ersten Kapitel zur uck. Was dort uber den Vergleich zu einer theoretischen Verteilung gesagt worden ist, kann analog auf den Vergleich von zwei Verteilungen, z.B. aus zwei Behandlungsgruppen, ubertragen werden. Die statistischen Aussagen m ussen jedoch revidiert werden. Nun betrachten wir nicht mehr eine feste und eine zuf allige Verteilung, sondern wir vergleichen zwei zuf allige (empirische) Verteilungen.

Die fur den Einstichproben-Fall (eine Stichprobe im Vergleich zu einer hypothetischen Verteilung) benutzte Idee von Monte-Carlo-Bandern kann nicht unmittelbar ubertragen werden: wir wollen zwei Verteilungen miteinander vergleichen, aber wir haben keine ausgezeichnete Modellverteilung, aus der wir Referenzstichproben ziehen k onnen.

Wir k onnen jedoch die Idee modi zieren und bedingte Monte-Carlo-B ander konstruieren. Bedingt bedeutet hier: die Konstruktion h angt von beobachteten Stichprobenwerten ab. Wir nehmen an, dass wir zwei Stichproben $Y_{11}; : : : ; Y_{1n_1}$ und $Y_{21}; : : : ; Y_{2n_2}$ von insgesamt unabh angigen und innerhalb der Gruppen identisch nach F_1 bzw. F_2 verteilten Beobachtungen haben. Falls kein Unterschied zwischen den Verteilungen besteht, so ist $(Y_{11}; : : : ; Y_{1n_1}; Y_{21}; : : : ; Y_{2n_2})$ eine iid-Stichprobe aus einer gemeinsamen Verteilung $F = F_1 = F_2$ mit Stichprobenumfang $n = n_1 + n_2$. Bei einer iid-Stichprobe h atte jede Permutation der Indizes die gleiche Wahrscheinlichkeit.

Die motiviert das folgende Verfahren: wir permutieren das Tupel $(Y_{11}; : : : ; Y_{1n_1}; Y_{21}; : : : ; Y_{2n_2})$ und ordnen die ersten n_1 Werte (nach Permutation) der ersten Gruppe zu, die anderen der zweiten.

Die Permutationsgruppe ist schnell so gro , dass sie nicht mehr vollst andig ausgewertet werden kann. Anstelle dessen benutzen wir eine zuf allige Auswahl von Permutationen. Wir benutzen die so generierten Werte, um Monte-Carlo-B ander zu generieren.

Aufgabe 3.7	
	Modi zieren Sie die Funktionen fur P P-Plot und QQ-Plot so, dass Monte-Carlo-B ander fur den Vergleich von zwei Stichproben hinzugef ugt werden. (Fortsetzung)!

Aufgabe 3.7	(Fortsetzung)
	Hinweis: mit der Funktion sample() können Sie zufällige Permutationen generieren.

Bei großem Stichprobenumfang kann der Aufwand Permutationen zu generieren zu zeitaufwendig sein. Um Verwaltungsaufwand zu sparen, können wir die Permutation durch ein Ziehen aus denn Werten $(Y_{11}, \ldots, Y_{1n_1}, Y_{12}, \ldots, Y_{1n_2})$ mit Zurücklegen ersetzen. Diese approximative Lösung wird als Bootstrap-Approximation [1] bezeichnet.

Da es nur endlich viele Permutationen gibt, können wir bei kleinem Stichprobenumfang auch alle Permutationen durchgehen. Wir wählen die Bänder dann so, dass ein hinreichend großer Anteil (etwa mehr als 95 %) aller Kurven innerhalb der Bänder liegt. Permutationen, die sich nur innerhalb der Gruppen unterscheiden, ergeben dieselben Kurven. Diese Zusatzüberlegung zeigt, dass wir nicht allen! Permutationen überprüfen müssen, sondern nur die $\binom{n}{n_1}$ Auswahlen für die Zuteilung zu den Gruppen.

Aufgabe 3.8	
	Ergänzen Sie PP-Plot und QQ-Plot für die click -Experimente durch Permutations-Bänder, die 95 % der Permutationen abdecken.
	Erzeugen Sie neue Plots, in denen Sie die PP-Plots und QQ-Plots durch Monte-Carlo-Bänder aus den Permutationen ergänzen. Benutzen Sie die Einhüllende von 19 Monte-Carlo-Stichproben. *Hinweis:* benutzen Sie die Funktion sample() um eine Stichprobe vom Umfang n_1 aus $x = (Y_{11}, \ldots, Y_{1n_1}, Y_{12}, \ldots, Y_{1n_2})$ zu ziehen.
	Hinweis: Siehe help(sample) .

Aufgabe 3.9	
	Versuchen Sie, die Eigenschaften der Permutationsbänder, Monte-Carlo-Bänder und Bootstrap-Bänder zu vergleichen, wenn $F_1 = F_2$ gilt.

Wenn nicht die Verteilungen verglichen werden sollen, sondern nur einzelne festgelegte Kenngrößen, so können diese Strategien analog eingesetzt werden. Wenn wir uns z.B. auf die Shift-Alternative beschränken (d.h. F_1 und F_2 sind aus eine Shiftfamilie, d.h. $F_1(x) = F_2(x - a)$ für ein a), so können wir etwa den Mittelwert (oder den Median) als Kenngröße nehmen. Auf diese Kenngröße kann das obige Vorgehen analog angewandt werden, um zu entscheiden, ob die Hypothese, dass die Verteilungen sich nicht unterscheiden ($a = 0$), angesichts der Daten haltbar ist.

Aufgabe 3.10	
	Formulieren Sie die obigen Strategien für Intervalle für einzelne Teststatistiken (Beispiel: Mittelwert) anstelle für Bänder. (Fortsetzung)!

[1] Vorsicht: es gibt beliebig wilde Definitionen von Bootstrap. Versuchen Sie stets, das Vorgehen mathematisch genau zu formulieren, wenn von Bootstrap die Rede ist.

Aufgabe 3.10	(Fortsetzung)
	Hinweis: Konnen Sie anstelle der zwei Mittelwerte fur beide Gruppen eine eindimensionale zusammenfassende Statistik benutzen?

3.2.1. Kolmogorov Smirnov Tests. In Kapitel 1 haben wir den Kolmogorov-Smirnov-Test zum Vergleich einer Stichprobe $(X_i)_{i=1,\ldots,n}$ und der zugehorigen empirischen Verteilung F_n mit einer (festen, vorgegebenen) Verteilung F kennengelernt. Die kritische Testgro e ist dabei

$$\sup|F_n - F|.$$

Wir k onnen diesen Test etwas modi zieren, um zwei empirische Verteilungen zu vergleichen. Anstelle der Modellverteilung F tritt nun eine zweite empirische Verteilung G_m von Beobachtungen $(Y_j)_{j=1,\ldots,m}$ mit zu Grunde liegender (unbekannter) Verteilung G. Die kritische Testgro e ist dann

$$\sup|F_n - G_m|.$$

Der darauf basierende Test ist in der Literatur als 2-Stichproben-Kolmogorov-Smirnov-Test zu nden. Dieser Test korrespondiert zum P P-Plot und erlaubt es, Bander zum P P-Plot zu konstruieren.

Wir k onnen Bander auch durch Simulation bestimmen. Im Gegensatz zum 1-Stichproben-Test haben wir jetzt keine vorgegebene Verteilung, aus der wir simulieren konnen. Unter der Hypothese, dass die Verteilungen F und G sich nicht unterscheiden, verhalt sich jedoch bei unabhangigen Beobachtungen der gemeinsame Vektor $(X_1,\ldots,X_n,Y_1,\ldots,Y_m)$ wie ein Vektor von $n+m$ unabhangigen Zufallszahlen mit identischer Verteilung $F = G$. Bei gegebenen Daten kann diese Beziehung zur Simulation genutzt werden. Durch eine Permutation der Indizes erzeugt man aus dem Vektor $Z = (X_1,\ldots,X_n,Y_1,\ldots,Y_m)$ einen neuen Vektor Z' mit $Z'_i = Z_{(i)}$. Die ersten n Komponenten benutzen wir als simulierte Werte $(X'_i)_{i=1,\ldots,n}$, die ubrigen m Komponenten als simulierte Werte $(Y'_j)_{j=1,\ldots,m}$.

Aufgabe 3.11	
	Programmieren Sie diesen Algorithmus und erganzen Sie den P P-Plot durch simulierte P P-Plots fur eine kleine Anzahl (19?) von Permutation.
	Bestimmen Sie die Permutationsverteilung von $\sup\|F_n - G_m\|$ aus den Simulation und berechnen Sie diesen Wert fur die ursprunglichen Daten. Konnen Sie diesen Vergleich benutzen, um ein Testverfahren zu de nieren?
	Der implementierte Kolmogorov-Smirnov-Test beinhaltet eine Approximation fur den 2-Stichprobenfall. In unserer Simulation wissen wir, dass wir unter der Hypothese simulieren, die Hypothese also zutri t. Untersuchen Sie die Verteilung des nominellen Niveaus unter den simulierten Bedingungen.

3.3. Tests auf Shift

Wenn wir zusatzliche Verteilungsannahmen machen, konnen wir andere Entscheidungsverfahren wahlen. Fur diese Verfahren sind aber die Verteilungsannahmen kritisch. Diese Abhangigkeit von den Verteilungsannahmen kann gemildert oder vermieden werden, wenn

wir geeignete Verteilungsannahmen sicherstellen können. Der F-Test, den wir im letzten Kapitel kennengelernt haben, ist ein Beispiel für ein verteilungsabhängiges Verfahren. Für den Zwei-Stichprobenfall kann dieser Test modifiziert werden zum t-Test, der auch die Richtung des Unterschiedes widerspiegelt. (Das Quadrat der t-Statistik ist eine F-Statistik.)

---

help (t.test)

`t.test` *Student's t-Test*

Description.

Performs one and two sample t-tests on vectors of data.

Usage.

```
t.test(x, ...)

## Default S3 method:
t.test(x, y = NULL,
       alternative = c("two.sided", "less", "greater"),
       mu = 0, paired = FALSE, var.equal = FALSE,
       conf.level = 0.95, ...)

## S3 method for class 'formula':
t.test(formula, data, subset, na.action, ...)
```

Arguments.

`x`	a numeric vector of data values.
`y`	an optional numeric vector data values.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.
`mu`	a number indicating the true value of the mean (or difference in means if you are performing a two sample test).
`paired`	a logical indicating whether you want a paired t-test.
`var.equal`	a logical variable indicating whether to treat the two variances as being equal. If `TRUE` then the pooled variance is used to estimate the variance otherwise the Welch (or Satterthwaite) approximation to the degrees of freedom is used.
`conf.level`	confidence level of the interval.
`formula`	a formula of the form `lhs ~ rhs` where `lhs` is a numeric variable giving the data values and `rhs` a factor with two levels giving the corresponding groups.
`data`	an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used.
`na.action`	a function which indicates what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`.
`...`	further arguments to be passed to or from methods.

Details.

The formula interface is only applicable for the 2-sample tests.

If `paired` is `TRUE` then both `x` and `y` must be specified and they must be the same length. Missing values are removed (in pairs if `paired` is `TRUE`). If `var.equal` is `TRUE` then the pooled estimate of the variance is used. By default, if `var.equal` is `FALSE` then the variance is estimated separately for both groups and the Welch modification to the degrees of freedom is used.

If the input data are effectively constant (compared to the larger of the two means) an error is generated.

Value.

A list with class `"htest"` containing the following components:

`statistic`	the value of the t-statistic.
`parameter`	the degrees of freedom for the t-statistic.
`p.value`	the p-value for the test.
`conf.int`	a confidence interval for the mean appropriate to the specified alternative hypothesis.
`estimate`	the estimated mean or difference in means depending on whether it was a one-sample test or a two-sample test.
`null.value`	the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test.
`alternative`	a character string describing the alternative hypothesis.
`method`	a character string indicating what type of t-test was performed.
`data.name`	a character string giving the name(s) of the data.

See Also.

`prop.test`

Examples.

```
t.test(1:10,y=c(7:20))      # P = .00001855
t.test(1:10,y=c(7:20, 200)) # P = .1245    -- NOT significant anymore

## Classical example: Student's sleep data
plot(extra ~ group, data = sleep)
## Traditional interface
with(sleep, t.test(extra[group == 1], extra[group == 2]))
## Formula interface
t.test(extra ~ group, data = sleep)
```

In seiner einfachsten Form setzt der t-Test voraus, dass wir unabhängig identisch verteilte Stichproben aus Normalverteilungen haben. Tatsächlich reichen schwächere Voraussetzungen. Wenn wir die t-Test-Statistik als

$$t = \frac{c_1 \quad c_2}{\sqrt{(Var(c_1 \quad c_2))}} \tag{3.1}$$

schreiben, so sehen wir, dass t-verteilt ist, wenn $c_1 \quad c_2$ normalverteilt und $(Var(c_1 \quad c_2))$ ^{2} verteilt ist, und beide Term unabhängig sind. Der zentrale Grenzwertsatz garantiert, dass $c_1 \quad c_2$ unter milden Bedingungen zumindest asymptotisch normalverteilt ist. Analoges gilt oft für $(Var(c_1 \quad c_2))$. Gilt die Unabhängigkeit beider Terme, so ist t approximativ t-verteilt.

Aufgabe 3.12	
*	Bestimmen Sie in einer Simulation die Verteilung von $\overline{Y}$, $\widehat{Var}(Y)$ und der t-Statistik für Y aus der uniformen Verteilung $U[0;1]$ mit Stichprobenumfang $n = 1;\ldots;10$. Vergleichen Sie die Verteilungen aus der Simulation mit der entsprechenden Normal-, χ^2- bzw. t-Verteilung. Bestimmen Sie in einer Simulation die Verteilung von $\overline{Y}$, $\widehat{Var}(Y)$ und der t-Statistik für Y aus einer Mischung, die zu 90% aus einer $N(0;1)$- und zu 10% aus einer $N(0;10)$-Verteilung stammt, mit Stichprobenumfang $n = 1;\ldots;10$. Vergleichen Sie die Verteilungen aus der Simulation mit der entsprechenden Normal-, χ^2- bzw. t-Verteilung.

Der t-Test hat eine gewisse Robustheit, die ihm eine approximative Gültigkeit geben kann. Man kann sich jedoch ganz von der Normalverteilungs-Voraussetzung befreien. Wenn wir analog zum F-Test bzw. t-Test vorgehen, aber anstelle der Urdaten die Ränge benutzen, gewinnen wir Testverfahren, die verteilungsunabhängig sind (zumindest, solange keine Bindungen auftreten können). Der Wilcoxon-Test ist eine verteilungsunabhängige Variante des t-Tests. Theoretisch entspricht er genau dem t-Test, angewandt auf die (gemeinsam) rangtransformierten Daten. Wie der t-Test ist dieser Test nur darauf ausgelegt, die Nullhypothese (kein Unterschied) gegen eine Shift-Alternative zu testen. Für die praktische Anwendung können arithmetische Vereinfachungen ausgenutzt werden. Deshalb ist die Beziehung zwischen den üblichen Formeln für den t-Test und für den Wilcoxon-Test nicht einfach zu erkennen.

Um den Wilcoxon-Test anzuwenden, muss zum einen die Teststatistik berechnet werden. Zur Bestimmung kritischer Werte, mit denen die Teststatistik zu vergleichen ist, muss zum anderen die Verteilungsfunktion ausgewertet werden. Sind alle Beobachtungen paarweise verschieden, so hängt diese Funktion nur von n_1 und n_2 ab, und relativ einfache Algorithmen stehen zur Verfügung. Diese sind in der Funktion R standardmäßig verfügbar und werden von `wilcox.test()` benutzt. Gibt es Bindungen in den Daten, d.h. gibt es übereinstimmende Werte, so hängt die Verteilung vom speziellen Muster dieser Bindungen ab und die Berechnung ist aufwendiger. `wilcox.test()` greift in diesem Fall auf Approximationen zurück. Zur exakten (im Gegensatz zur approximativen) Auswertung stehen jedoch die entsprechenden Algorithmen ebenfalls zur Verfügung. Dazu benötigt man `libary(coin)`. Die exakte Variante des Wilcoxon-Tests findet sich dort etwa als `wilcox_test()`.

Auf den Rangen basierende verteilungsunabhängige Verfahren zu charakterisieren und mit den früher vorgestellten verteilungsunabhängigen Monte-Carlo-Verfahren und deren Varianten zu vergleichen ist ein klassischer Teil der Statistik. Literatur dazu findet man unter den Schlagworten \Rangtests" oder \verteilungsfreie Verfahren". Zusätzliche R-Funktionen finden sich in `libary(coin)` sowie in einigen speziellen Paketen.

Natürlich stellt sich die Frage nach dem Informationsverlust. Wenn wir uns auf die Daten beschränken und keine oder geringe Verteilungsannahmen machen, haben wir weniger Information als in einem Modell mit expliziten Verteilungsannahmen. Wenn wir die Daten auf die Ränge reduzieren, verschenken wir zusätzlich möglicherweise Information. Dieser Informationsverlust kann z.B. durch die asymptotische relative Effizienz gemessen werden. Dies ist (asymptotisch) der Stichprobenumfang eines optimalen Tests, der benötigt wird, eine vergleichbare Güte wie ein konkurrierender Test zu erreichen. Beim Wilcoxon-Test

unter Normalverteilung hat dies einen Wert von 94%. Gilt also die Normalverteilungsannahme, so benotigt der (optimale) t-Test nur 94% des Stichprobenumfangs, die der Wilcoxon-Test benotigt. 6% des Stichprobenumfangs sind die Kosten fur die Reduzierung auf Range. Gilt die Normalverteilungsannahme nicht, so kann der t-Test moglicherweise zusammenbrechen. Der Wilcoxon-Test bleibt ein valider Test auf die Shift-Alternative.

---

help (wilcox.test)

`wilcox.test` *Wilcoxon Rank Sum and Signed Rank Tests*

Description.

Performs one and two sample Wilcoxon tests on vectors of data; the latter is also known as 'Mann-Whitney' test.

Usage.

```
wilcox.test(x, ...)

## Default S3 method:
wilcox.test(x, y = NULL,
            alternative = c("two.sided", "less", "greater"),
            mu = 0, paired = FALSE, exact = NULL, correct = TRUE,
            conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'formula':
wilcox.test(formula, data, subset, na.action, ...)
```

Arguments.

`x`	numeric vector of data values. Non- nite (e.g. in nite or missing) values will be omitted.
`y`	an optional numeric vector of data values.
`alternative`	a character string specifying the alternative hypothesis, must be one of `"two.sided"` (default), `"greater"` or `"less"`. You can specify just the initial letter.
`mu`	a number specifying an optional location parameter.
`paired`	a logical indicating whether you want a paired test.
`exact`	a logical indicating whether an exact p-value should be computed.
`correct`	a logical indicating whether to apply continuity correction in the normal approximation for the p-value.
`conf.int`	a logical indicating whether a con dence interval should be computed.
`conf.level`	con dence level of the interval.
`formula`	a formula of the form `lhs ~ rhs` where `lhs` is a numeric variable giving the data values and `rhs` a factor with two levels giving the corresponding groups.
`data`	an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used.
`na.action`	a function which indicates what should happen when the data contain `NA`s. Defaults to `getOption("na.action")`.
`...`	further arguments to be passed to or from methods.

Details.

The formula interface is only applicable for the 2-sample tests.

If only `x` is given, or if both `x` and `y` are given and `paired` is `TRUE`, a Wilcoxon signed rank test of the null that the distribution of `x` (in the one sample case) or of `x-y` (in the paired two sample case) is symmetric about `mu` is performed.

Otherwise, if both `x` and `y` are given and `paired` is `FALSE`, a Wilcoxon rank sum test (equivalent to the Mann-Whitney test: see the Note) is carried out. In this case, the null hypothesis is that the distributions of `x` and `y` differ by a location shift of `mu` and the alternative is that they differ by some other location shift.

By default (if `exact` is not specified), an exact p-value is computed if the samples contain less than 50 finite values and there are no ties. Otherwise, a normal approximation is used.

Optionally (if argument `conf.int` is true), a nonparametric confidence interval and an estimator for the pseudomedian (one-sample case) or for the difference of the location parameters `x-y` is computed. (The pseudomedian of a distribution F is the median of the distribution of $(u + v)/2$, where u and v are independent, each with distribution F. If F is symmetric, then the pseudomedian and median coincide. See Hollander & Wolfe (1973), page 34.) If exact p-values are available, an exact confidence interval is obtained by the algorithm described in Bauer (1972), and the Hodges-Lehmann estimator is employed. Otherwise, the returned confidence interval and point estimate are based on normal approximations.

With small samples it may not be possible to achieve very high confidence interval coverages. If this happens a warning will be given and an interval with lower coverage will be substituted.

Value.

A list with class `"htest"` containing the following components:

`statistic`	the value of the test statistic with a name describing it.
`parameter`	the parameter(s) for the exact distribution of the test statistic.
`p.value`	the p-value for the test.
`null.value`	the location parameter `mu`.
`alternative`	a character string describing the alternative hypothesis.
`method`	the type of test applied.
`data.name`	a character string giving the names of the data.
`conf.int`	a confidence interval for the location parameter. (Only present if argument `conf.int = TRUE`.)
`estimate`	an estimate of the location parameter. (Only present if argument `conf.int = TRUE`.)

Warning.

This function can use large amounts of memory and stack (and even crash R if the stack limit is exceeded) if `exact = TRUE` and one sample is large (several thousands or more).

Note.

The literature is not unanimous about the definitions of the Wilcoxon rank sum and Mann-Whitney tests. The two most common definitions correspond to the sum of the ranks of the first sample with the minimum value subtracted or not: R subtracts and S-PLUS does not, giving a value which is larger by $m(m + 1)/2$ for a first sample of

size m. (It seems Wilcoxon's original paper used the unadjusted sum of the ranks but subsequent tables subtracted the minimum.)

R's value can also be computed as the number of all pairs `(x[i], y[j])` for which `y[j]` is not greater than `x[i]`, the most common de nition of the Mann-Whitney test.

References.

Myles Hollander & Douglas A. Wolfe (1973), *Nonparametric statistical inference*. New York: John Wiley & Sons. Pages 27{33 (one-sample), 68{75 (two-sample).
Or second edition (1999).

David F. Bauer (1972), Constructing con dence sets using rank statistics. *Journal of the American Statistical Association* 67, 687{690.

See Also.

`psignrank`, `pwilcox`.

`wilcox.exact` in exactRankTests covers much of the same ground, but also produces exact p-values in the presence of ties.

`kruskal.test` for testing homogeneity in location parameters in the case of two or more samples; `t.test` for an alternative under normality assumptions [or large samples]

Examples.

```
## One-sample test.
## Hollander & Wolfe (1973), 29f.
## Hamilton depression scale factor measurements in 9 patients with
##  mixed anxiety and depression, taken at the first (x) and second
##  (y) visit after initiation of a therapy (administration of a
##  tranquilizer).
x <- c(1.83,  0.50,  1.62,  2.48, 1.68, 1.88, 1.55, 3.06, 1.30)
y <- c(0.878, 0.647, 0.598, 2.05, 1.06, 1.29, 1.06, 3.14, 1.29)
wilcox.test(x, y, paired = TRUE, alternative = "greater")
wilcox.test(y - x, alternative = "less")    # The same.
wilcox.test(y - x, alternative = "less",
            exact = FALSE, correct = FALSE) # H&W large sample
                                            # approximation

## Two-sample test.
## Hollander & Wolfe (1973), 69f.
## Permeability constants of the human chorioamnion (a placental
##  membrane) at term (x) and between 12 to 26 weeks gestational
##  age (y).  The alternative of interest is greater permeability
##  of the human chorioamnion for the term pregnancy.
x <- c(0.80, 0.83, 1.89, 1.04, 1.45, 1.38, 1.91, 1.64, 0.73, 1.46)
y <- c(1.15, 0.88, 0.90, 0.74, 1.21)
wilcox.test(x, y, alternative = "g")        # greater
wilcox.test(x, y, alternative = "greater",
            exact = FALSE, correct = FALSE) # H&W large sample
                                            # approximation

wilcox.test(rnorm(10), rnorm(10, 2), conf.int = TRUE)
```

```
## Formula interface.
boxplot(Ozone ~ Month, data = airquality)
wilcox.test(Ozone ~ Month, data = airquality,
            subset = Month %in% c(5, 8))
```

Aufgabe 3.13	
	Benutzen Sie den Wilcoxon-Test, um die Resultate des rechts/links click -Experiments zu vergleichen.

Aufgabe 3.14	
***	Beim rechts/links click -Experiment sind mehrere Effekte vermischt. Einige Probleme: Die Antwortzeit beinhaltet Reaktionszeit, Zeit für die Grob-Bewegung der Maus, Zeit für die Fein-Adjustierung etc. Für rechts-links-Bewegungen reicht in der Regel ein Schwenken der Hand aus. Für vorwärts-rückwärts-Bewegungen ist in der Regel eine Arm-Bewegung nötig. Es ist nicht zu erwarten, dass beide vergleichbares statistisches Verhalten haben. Bei aufeinanderfolgenden Registrierungen kann es zum einen Trainings- zum anderen Ermüdungseffekte geben. Können Sie Experiment und Auswertung so modifizieren, dass Unterschiede in der Reaktionszeit untersucht werden können? Können Sie Experiment und Auswertung so modifizieren, dass Unterschiede in der Genauigkeit der Endposition untersucht werden können?
***	Untersuchen und dokumentieren Sie für sich rechts-links-Unterschiede in der Reaktionszeit und in der Genauigkeit. Formulieren Sie ihr Resultat als Bericht.

Aufgabe 3.15	
	Betrachten Sie als Verteilungsfamilien die Shift/Skalenfamilien von $N(0;1)$ und $t(3)$. Entwerfen Sie ein Szenario, um den Wilcoxon-Test mit dem t-Test jeweils innerhalb dieser Familien zu vergleichen. Führen Sie diesen Test in einer Simulation für Stichprobenumfänge $n_1 = n_2 = 10; 20; 50; 100$ durch und fassen Sie die Resultate zusammen. Führen Sie eine analoge Simulation für die Lognormal-Verteilungen durch.

3.4. Güte

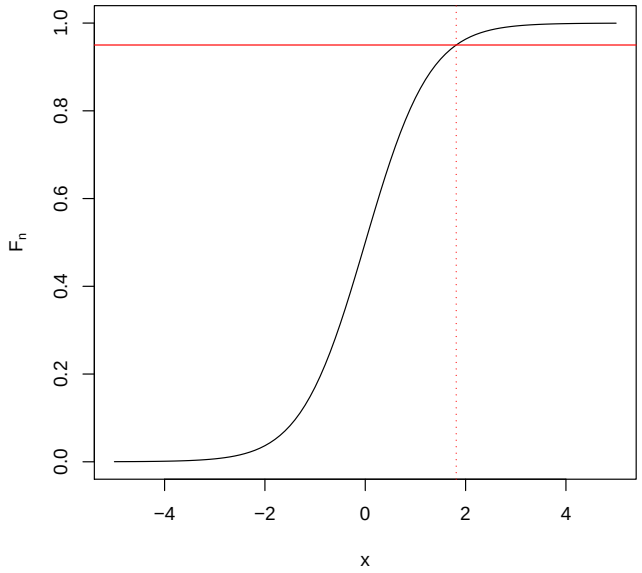
3.4.1. Theoretische Güte. Am Beispiel des t-Tests können wir illustrieren, wie ein Test aufgebaut ist. Der Test benutzt eine Teststatistik, hier die t-Test-Statistik zum

Vergleich zweier Gruppen (3.1). Wir kennen die Verteilung dieser Statistik: für den t-Test ist bei unabhängigen normalverteilten Fehlern und gleicher Varianz die Teststatistik $t(n_1 + n_2 - 2)$ verteilt. Zu gewähltem Niveau α können wir aus der Verteilungsfunktion können wir Grenzen ablesen, die bei dieser Verteilung nur mit einer Wahrscheinlichkeit α unter- bzw. überschritten werden. Benutzen wir beide Grenzen, so erhalten wir einen zweiseitigen Bereich mit der Irrtumswahrscheinlichkeit 2α.

Beispiel 3.4:

Eingabe

```
n1<- 6; n2 <- 6
df <- n1 + n2 -2
alpha <- 0.05
curve(pt(x,df=df),from=-5, to=5, ylab= expression(F[n]))
abline(h=1-alpha, col="red")          # cut at upper quantile
abline(v=qt(1-alpha, df=df), lty=3, col="red") # get critical value
```



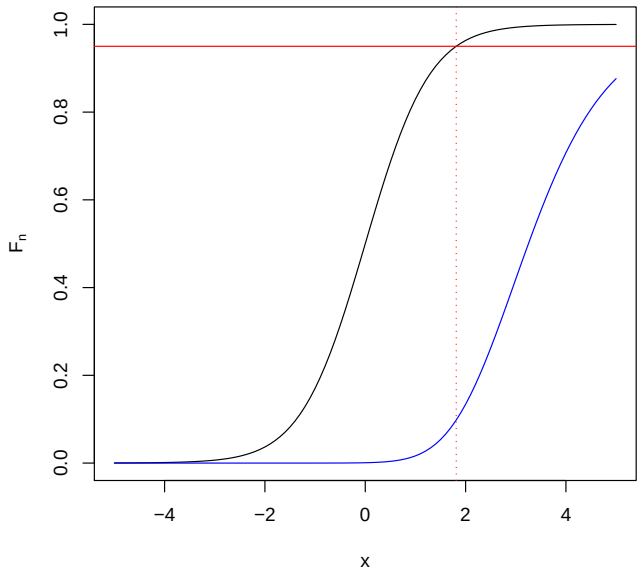
Wollen wir z.B. die Hypothese $mu_1 = mu_2$ gegen die Alternative $mu_1 > mu_2$ testen, so wählen wir als Verwerfungsbereich den Bereich über der oberen dieser Grenze. Wir wissen, dass wir bei Gültigkeit der Hypothese höchstens mit Wahrscheinlichkeit α zufällig eine Testgröße in diesem Bereich bekommen.

Für den t-Test wissen wir sogar mehr. Unter den Modellvoraussetzungen unabhängig normalverteilter Fehler und gleicher Varianz ist die t-Test-Statistik immer t verteilt. Auf der Hypothese ist sie t verteilt mit Nichzentralitätsparameter 0, folgt also der zentralen t-Verteilung. Auf der Alternative haben wir eine t-Verteilung mit Nichzentralitätsparameter $(\mu_1 - \mu_2)/\sigma\sqrt{n_1 n_2/(n_1 + n_2)}$. Damit kann für jede Alternative unter den Modellannahmen die Stärke des Tests abgelesen werden, d.h. die Wahrscheinlichkeit bei Vorliegen dieser Alternative die Hypothese zu verwerfen.

Beispiel 3.5:

Eingabe

```
n1<- 6; n2 <- 6
df <-  n1 + n2 -2
alpha <- 0.05
curve(pt(x,df=df),from=-5, to=5, ylab= expression(F[n]))
abline(h=1-alpha, col="red")          # cut at upper quantile
abline(v=qt(1-alpha, df=df), lty=3, col="red") # get critical value
n1 <- 5
n2 <- 5
n <- n1+n2
theta <- 2
ncp <- theta * sqrt(n1 * n2/(n1+n2))
curve(pt(x,df=df, ncp=ncp),add=TRUE, col="blue")
```



Die Gute des Tests konnen wir darstellen, indem wir die Verwerfungswahrscheinlichkeit in Abhangigkeit von ($_1$ $_2$) ^{1} auftragen.[2]

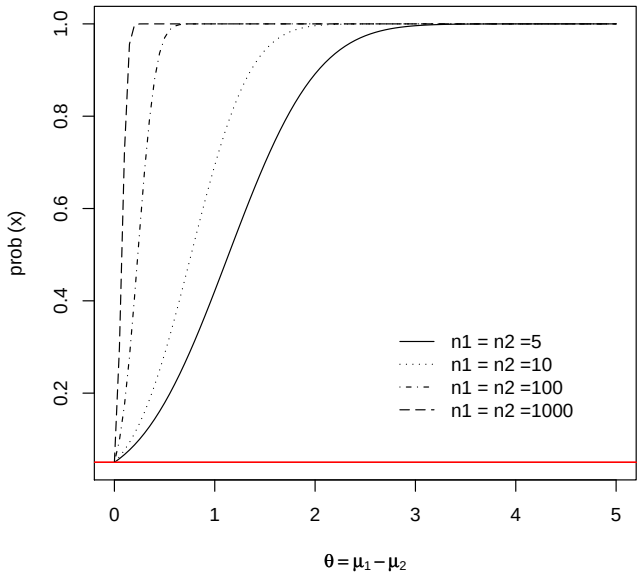
[2]Konventionell wird er die Gutefunktion nur auf der Alternative, d.h z.B. fur ($_1$ $_2$) > 0 betrachtet. Wir setzen sie hier auch auf der Hypothese, d. h. fur ($_1$ $_2$) > 0 fort.

Beispiel 3.6:

Eingabe

```
tpower <- function(n1, n2, alpha,...){
          df <-  n1 + n2 -2
          tlim <- qt(1-alpha,df=df)
          prob <- function(theta){
               pt(tlim, df = df,
                    ncp = theta * sqrt(n1 * n2/(n1+n2)),
                    lower.tail=FALSE)}
          curve(prob, 0, 5, xlab=expression(theta==mu[1]-mu[2]), ...)
          abline(h=alpha, col="red")
          }

          tpower(5, 5, 0.05)
          tpower(10, 10, 0.05, add =TRUE, lty = 3)
          tpower(100,100, 0.05, add =TRUE, lty = 4)
          tpower(1000, 1000, 0.05, add =TRUE, lty = 5)
          legend("bottomright",
               lty=c(1,3,4,5),
               legend=c("n1 = n2 =5", "n1 = n2 =10",
                    "n1 = n2 =100","n1 = n2 =1000"),
               inset=0.1, bty="n")
```



Der hier benutzte Zusammenhang kann auch benutzt werden, um zu bestimmen, wie gro der Stichprobenumfang sein muss, um auf der Hypothese höchstens mit einer Wahrscheinlichkeit falschlich zu verwerfen, bei Vorliegen einer spezi zierten Alternative jedoch mit einer gewahlten Wahrscheinlichkeit die Hypothese richtigerweise zu verwerfen. Dazu gibt es die vorbereitete Funktion power.t.test() .

Beispiel 3.7:

Eingabe

```
power.t.test(delta=2,
      power=0.8,
      sig.level=0.01,
      type="two.sample",
      alternative="one.sided")
```

Ausgabe

```
     Two-sample t test power calculation

              n = 6.553292
          delta = 2
             sd = 1
      sig.level = 0.01
          power = 0.8
    alternative = one.sided

NOTE: n is number in *each* group
```

3.4.2. Simulation der Gute. Sind die theoretischen Eigenschaften einer Test-Statistik bekannt, so ist dies der beste Weg, die Gute zu analyisieren. In einer Umgebung wie R haben wir die Moglichkeit, die Gute auch dann zu untersuchen, wenn theoretische Resultate nicht vorliegen oder nicht zuganglich sind. Zu festgelegten Alternativen konnen wir Zufallsstichproben generieren, Tests durchfuhren und den relative Anteil der Verwerfungen bestimmen. Generieren wir nsimul unabhangige Zufallsstichproben mit identischer Verteilung, so ist die Anzahl der Verwerfungen binomialverteilt und

$$\hat{p} = \frac{\#\,\text{Verwerfungen}}{\text{nsimul}}$$

ein Schatzer fur die Verwerfungswahrscheinlichkeit.

Als Beispiel untersuchen wir, wie sich der t-Test verhalt, wenn die Daten lognormal verteilt sind. Wir vergleichen zwei Gruppen jeweils mit Stichprobenumfang $n_1 = n_2 = 10$, zunachst auf der Hypothese:

Beispiel (Fortsetzung):

Beispiel 3.8:

Eingabe

```
nsimul <- 300
n1<- 10; n2 <- 10
alpha <- 0.01
x <- 0
for (i in 1:nsimul) {
          if (t.test(exp(rnorm(n1)),exp(rnorm(n2)),
          alternative="less",
          var.equal = TRUE)$p.value < alpha){
               x <- x+1}
           }
p <- x/nsimul
cat("estim p", p)
```

Ausgabe

```
estim p 0.006666667
```

Die Funktion prop.test() berechnet nicht nur diesen Schätzer, sondern auch einen Konfidenzbereich.

Beispiel 3.9:

Eingabe

```
prop.test(n=nsimul, x=x)
```

Ausgabe

```
        1-sample proportions test with continuity correction

data:  x out of nsimul, null probability 0.5
X-squared = 290.0833, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.001155561 0.026512753
sample estimates:
          p
0.006666667
```

Analog z. B. wenn für die Alternative $\log(x_2)$ nach $N(\ _2; 1)$ mit $\ _2 = 1$ verteilt ist:

Beispiel 3.10:

Eingabe

```
nsimul <- 300
n1<- 10; n2 <-10
alpha <- 0.01
x<-0
for (i in 1:nsimul) {
        if (t.test(exp(rnorm(n1)),exp(rnorm(n2, mean = 1)),
            alternative="less",
            var.equal = TRUE)$p.value < alpha){
                x <- x+1}
         }
p <- x/nsimul
cat("estim p", p)
```

Ausgabe

```
estim p 0.1866667
```

Eingabe

```
prop.test(n = nsimul, x = x)
```

Ausgabe

```
        1-sample proportions test with continuity correction

data:  x out of nsimul, null probability 0.5
X-squared = 116.5633, df = 1, p-value < 2.2e-16
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.1451407 0.2364119
sample estimates:
        p
0.1866667
```

In library(binom) finden sich eine Reihe von Werkzeugen für eine differenziertere Analyse der Binomialverteilung.

Die Konfidenzintervalle in diesem Beispiel zeige, dass hier ein Simulationsumfang von nsimul = 300 nur grobe Ergebnisse liefert. Für die Simulation wollen wir die Genauigkeit besser kontrollieren. Den Simulationsumfang können wir so wählen, dass eine gewünschte Genauigkeit erreicht werden kann. Eine genaue Planung können wir mit power.prop.test() machen. Für Simulationszwecke reicht oft schon eine Abschätzung.

Mit $\hat{p} := Z/n$ als Schätzer einer Wahrscheinlichkeit p haben wir $E(\hat{p}) = p$ und $Var(\hat{p}) = p(1-p)/n$. Ist p tatsächlich der interessierende Parameter, so sind Fehler relativ zum Zielparameter zu messen. Bei $p = 50\%$ ist ein Fehler von 1% anders zu bewerten, als bei $p = 99\%$. Der relative Fehler, der Variationskoeffizient, ist

$$\frac{\sqrt{Var(\hat{p})}}{E(\hat{p})} = \sqrt{\frac{1-p}{np}}.$$

Um einen Variationskoeffizienten von höchstens zu erhalten, brauchen wir eine Stichprobenumfang

$$n \quad \frac{1-p}{p^2}.$$

Sind n und p in einer Gro enordnung, bei der eine Normalapproximation gilt, so haben wir f ur ein Kon denzniveau 1 ein approximatives Kon denzintervall mit Grenzen

$$\hat{p} \quad {}_{1 = 2} \sqrt{\frac{\hat{p}(1 \quad \hat{p})}{n}}:$$

Soll das Kon denzintervall eine L ange von p nicht uberschreiten, so brauchen wir einen Stichprobenumfang

$$n \quad \frac{{}^{2}_{1 = 2}(1 \quad p)}{p^{2}}:$$

Wie ublich ist eine Wahl von zu tre en. F ur z. B. =1% mit ${}_{1 = 2} = 2{:}575829$ erhalten wir die Werte in Tabelle 3.26. Falls wir mit h oheren Quantilen arbeiten, werden wir versuchen, die Fehler relativ zu $1 \quad p$ zu beschr anken. Beispiele sind in Tabelle 3.26.

Tabelle 3.26. Erforderlicher Stichprobenumfang f ur zweiseitige Kon denzintervalle mit relativer L ange

p	$1 \quad p$	$n($ = 10%) = 0:1	= 0:01	$n($ = 1%) = 0:1	= 0:01
0.500	0.500	271	27 055	663	66 349
0.250	0.750	812	81 166	1990	199 047
0.100	0.900	2435	243 499	5971	597 141
0.010	0.990	26 785	2 678 488	65 685	6 568 547
0.001	0.999	270 283	27 028 379	662 826	662 822 617

Zu merken sind die groben Zahlen: um mit 90% Kon denz eine Wahrscheinlichkeit im Bereich von 50% 5% zu sch atzen, sind ca. 300 Simulationen notwendig. Um einen Wert bei 99% bis auf 0:1% genau zu sch atzen sind 30000 Simulationen n otig.

3.4.3. Quantilsch atzung durch Simulation.

Die andere Seite des Problems ist es, ein Quantil anhand einer Stichprobe zu sch atzen. Wir wissen bereits, dass f ur eine Zufallsvariable X mit stetiger Verteilungsfunktion F die Variable $F(X)$ eine uniforme Verteilung auf $[0; 1]$ hat. Fur die Quantilschatzung ben otigen wir die Verteilungsfunktion, ausgewertet an den Ordnungsstatistiken. Diese haben wir bereits in Kapitel ?? kennen gelernt. Dort hatten wir als Theorem ??.

Theorem 3.2. *Sind* $X_i; i = 1; : : : ; n$ *unabhängige Beobachtungen aus einer Verteilung mit stetiger Verteilungsfunktion* F *und ist* $X_{(k:n)}$ *die* k*: Ordnungsstatistik daraus, so ist*

$$F(X_{(k:n)}) \quad P_{beta}(\ ; k; n \quad k + 1):$$

Wir wiederholen:

Bemerkung 3.3. Im allgemeinen ist die beta-Verteilung schief. Der Erwartungswert der Beta$(k; n \quad k+1)$-Verteilung ist $k=(n+1)$. Um eine unverzerrte Sch atzung des Quantils x_p zu erhalten, benutzt man $X_{(k:n)}$ mit $k=(n+1) = p$. Die \plug in"-Approximation $k=n = p$ gibt eine verzerrte Sch atzung.

Das Theorem kann direkt angewendet werden, um eine obere oder untere Abschätzung für Quantile zu gewinnen. Insbesondere können wir versuchen, das Minimum der beobachteten Werte $X_{(1:n)}$ als untere Abschätzung für das p-Quantil. zu benutzen Das Konfidenzniveau ist

$$P(X_{(1)} \leq F_p) = P(F(X_{(k)}) \geq p) = I_p(1; n);$$

wobei I das unvollständige Beta-Integral ist. Für die speziellen Parameter $(1, n)$ vereinfacht sich die Beta-Dichte zu $n(1-p)^{n-1}$ und wir bekommen für das unvollständige Beta-Integral $I_p(1; n) = 1 - (1-p)^n$. Daraus folgt

$$P(X_{(1)} \leq F_p) = 1 - (1-p)^n$$

und wir können ein Konfidenzniveau $1-\alpha$ sicherstellen, wenn

$$n \geq \frac{\ln \alpha}{\ln(1-p)}.$$

Der beobachtete Höchstwert kann als obere Abschätzung für das p-Quantil verwendet werden. Aus Symmetriegründen erhalten wir einen Konfidenzniveau von $1-\alpha$, wenn

$$n \geq \frac{\ln \alpha}{\ln p}.$$

Beispiele sind in Tabelle 3.27 angegeben.

Tabelle 3.27. Benötigter Stichprobenumfang zur Abschätzung eines Quantils mit Konfidenzniveau $1-\alpha$

p		n			
$X_{(1)} \leq F_p$	$X_{(n)} \geq F_p$	$\alpha = 10\%$	$\alpha = 5\%$	$\alpha = 1\%$	$\alpha = 0{,}5\%$
0.500	0.500	4	5	7	8
0.250	0.750	9	11	17	19
0.100	0.900	22	29	44	51
0.010	0.990	230	299	459	528
0.001	0.999	2302	2995	4603	5296

Zu merken sind wieder die groben Zahlen: um eine einseitige Abschätzung für ein 1% (99%)-Quantil einer stetigen Verteilungsfunktion mit einer Konfidenz von 99% zu erhalten, werden beinahe 500 Simulationen benötigt.

Wir können einseitige Schranken zu Intervallen verknüpfen. Das entsprechende Resultat zur Berechnung der Wahrscheinlichkeit von Intervallen ist in Korollar ??:

Korollar 3.4. *Mit der k_1-ten und $k_1 + k_2$-ten Ordnungsstatistik ist das Intervall $(X_{(k_1:n)}; X_{(k_1+k_2:n)})$ ein Konfidenzintervall für das p-Quantil mit der Überdeckungswahrscheinlichkeit*

$$I_p(k_1; n - k_1 + 1) - I_p(k_1 + k_2; n - k_1 - k_2 + 1):$$

Die Simulationsumfänge zur Abschätzung von Quantilen sind drastisch geringer als diejenigen, die zur vergleichbaren Schätzung von Wahrscheinlichkeiten benötigt werden. Im Nachhinein ist dies nicht verwunderlich: die Frage, ob eine Beobachtung über einem bestimmten Quantil liegt, ist einfacher, als die Aufgabe, den p-Wert zu schätzen. In Abschnitt ?? werden wir sehen, dass der notwendige Stichprobenumfang noch einmal drastisch verringert werden kann, wenn die Fragestellung auf ein Testproblem reduziert wird.

Ohne weitere Verteilungsannahmen gibt dies eine erste Möglichkeit, den Umfang einer Simulation festzulegen. In speziellen Situationen können geschickte Einfälle eine bedeutende Reduzierung des Stichprobenumfangs erlauben. Zunächst aber sind die obigen Abschätzungen die Grundlage für Simulationen.

3.5. Qualitative Eigenschaften von Verteilungen

3.6. Erg anzungen

3.7. Statistische Zusammenfassung

Als Leitbeispiel diente in diesem Kapitel der Vergleich von Stichproben. In einfachen Fallen unterscheiden sich Stichproben nur um eine Verschiebung des Mittelwerts. In diesem Fall konnen die Probleme auf die Ansatze aus Kapitel 2 reduziert werden. In diesem reduzierten Fall stimmen die um den Mittelwert zentrierten Verteilungen uberein. Fur den allgemeineren Fall, den wir jetzt untersucht haben, gilt diese Vereinfachung nicht. Ein wichtiges Beispiel ist etwa die Untersuchung von Therapie-Studien. Hat eine Behandlung einen homogenen E ekt, so konnen wir diesen mit den Mitteln von Kapitel 2 untersuchen. Haufig aber gibt es unter einer Behandlung eine Aufspaltung in \Responder" und \Nicht-Responder". Dies geht uber die in Kapitel 2 skizzierten Modelle hinaus, und die allgemeineren Ansatze aus diesem Kapitel 3 werden notig.

Wir haben uns hier auf den Vergleich von zwei Stichproben beschrankt. Die Praxis fuhrt oft auf andere Probleme. So ist ein typischer Fall, dass eine neue Behandlung mit einer bekannten Referenz-Behandlung verglichen werden soll, wobei fur die neue Behandlung nur eine Stichprobeninformation, fur die Referenz-Behandlung aber umfassendere Vorinformation bereit steht. Oder eine Referenz-Behandlung soll mit einer Serie von Alternativ-Behandlungen verglichen werden. Diese Probleme gehen uber den Rahmen unserer Einfuhrung hinaus. Hier kann nur auf weiterfuhrende Literatur, z.B [Mil81] verwiesen werden.

3.8. Literatur und weitere Hinweise:

[VR02] Venables, W.N.; Ripley, B.D. (2002): Modern Applied Statistics with S. Heidelberg: Springer

[VR00] Venables, W.N.; Ripley, B.D. (2000): S Programming. Heidelberg: Springer

[Mil81] Miller, R. G. (1981): Simultaneous Statistical Inference. Heidelberg: Springer

KAPITEL 4

Dimensionen 1, 2, 3, . . . , 1

4.1. Erg anzungen

In diesem Kapitel beginnen wir Erganzungen zuR, um uns dann auf statistische Fragen zu konzentrieren. Fur werfen einen Blick auf die graphischen Mglichkeiten, die uns zur Verfugung stehen. Die Basis-Graphik vonR ist an einem Plotter-Modell orientiert. Die Graphik folgt den Moglichkeiten, die das Zeichnen mit einem Stift bietet. Neben den ein- und zweidimensionalen Mglichkeiten, die wir bis jetzt kennengelernt haben, gibt es Mglichkeiten, eine Funktion darzustellen, die uber einem Raster de niert sind. Dazu stehen im wesentlichen drei Funktionen zur Verfugung.

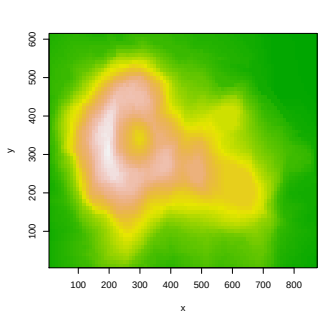
3d-Graphik	
image()	Gibt die Werte einer Variablen z in Graustufen oder Farbcodierung wieder.
contour()	Gibt die Kontouren einer Variablen z.
persp()	Gibt einen perspektivischen Plot einer Variablen z.

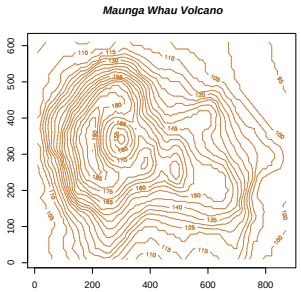
image() und contour() konnen auch benutzt werden, um andere Plots zuberlagern.

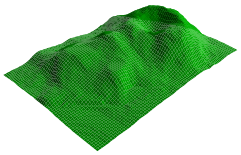
Beispiel 4.1:

Eingabe

```
#oldpar <- par(mfrow=c(1,3))
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))
image(x, y, volcano, col = terrain.colors(100), axes = FALSE)
axis(1, at = seq(100, 800, by = 100))
axis(2, at = seq(100, 600, by = 100))
box()
title(main = "Maunga Whau Volcano", font.main = 4)
contour(x, y, volcano, levels = seq(90, 200, by = 5),
        col = "peru", main = "Maunga Whau Volcano", font.main = 4)
z <- 2 * volcano          # Exaggerate the relief
x <- 10 * (1:nrow(z))     # 10 meter spacing (S to N)
y <- 10 * (1:ncol(z))     # 10 meter spacing (E to W)
## Don't draw the grid lines :   border = NA
#par(bg = "slategray")
persp(x, y, z, theta = 135, phi = 30, col = "green3", scale = FALSE,
      ltheta = -120, shade = 0.75, border = NA, box = FALSE)
```







image() contour() persp()

Die Basis-Graphik ist einfach zu handhaben, aber limitiert. Ein neues Gra ksystem arbeitet konzeptuell mit Objekten und einem Kamera-Modell. Die Gra k-Objekte k onnen kombiniert und bearbeitet werden. Die Darstellung erfolgt in einem getrennten Schritt. Einfache 2d-Gra ken konnen hier nachbearbeitet werden. F ur eine 3d-Darstellung konnen wie bei einer Kamera Abstand, Betrachtungsebene und Brennweite bestimmt werden. Das objektorientierte Graphiksystem besteht aus einer Bibliothek grid mit den notwendigen elementaren Operationen, und einer darauf aufbauende Bibliothek lattice , die die aus der Basis-Graphik bekannten Darstellungen neu implementiert und durch weitere erg nzt.

Lattice-Objekte werden mit print() ausgegeben.

Beispiel 4.2:

Eingabe

```
library(lattice)
## volcano   ## 87 x 61 matrix
print(wireframe(volcano, shade = TRUE,
                aspect = c(61/87, 0.4),
                light.source = c(10,0,10)))
```

Basis-Graphik und Lattice-Graphik sind getrennte Graphik-Systeme. Leider benutzen sie auch für vergleichbare Funktionen unterschiedliche Bezeichnungen, und vergleichbare Displays haben unterschiedliche Darstellungen. Eine kleine Übersetzungshilfe ist in Tabelle 4.4 angegeben. Einige Hilfsfunktionen, um beide Graphik-Systeme in Kombination zu nutzen, werden durch die Bibliothek gridBase bereitgestellt. Eine ausführliche Einführung in beide Graphik-System ist [Mur06].

Für Visualisierungen im weiten Spektrum von wissenschaftlichen Visualisierungen bis hin zu aufwendigen Spielen wird verbreitet OpenGL benutzt. Dessen Funktionen stehen auch in R durch die Bibliothek rgl zur Verfügung. Es gibt jedoch einen deutlichen Unterschied zwischen den üblichen Anforderungen an Graphik, und den speziellen Erfordernissen statistischer Graphik. Wenn es um die Darstellungen von Funktionen geht, ist statistische Graphik noch vergleichbar mit den Anforderungen der üblichen Analysis. Der kleine Unterschied ist, dass Funktionen in der Statistik häufig stückweise konstant oder nur stückweise stetig sind, während z.B. in der Analysis stetige oder sogar in di erenzierbare Funktionen die Regel sind. Bei der Darstellung von Daten ändert sich die Situation drastisch. Statistische Daten sind üblicherweise diskret. Glattheitseigenschaften, die die Visualisierung analytischer Daten einfacher machen, fehlen bei statistischen Daten. Deshalb sind spezielle angepasste Visualisierungen nötig.