Computerpraktikum im GP II
Lineare Regression

Daniel Brete

16. Januar 2004

Dieses Skript besteht aus 2 Teilen. Die Aufgaben finden Sie am Ende von Teil 1. Teil 2 ist eine Einführung in *Mathematica*. Beweise und Ergänzungen, die für die Bearbeitung der Aufgaben nicht unbedingt erforderlich sind, sind mit * gekennzeichnet.

Dank

Die Idee und ersten Versuche zu dieser Übung sind gemeinsam mit Jens Koesling entstanden, ohne den ich dieses Projekt nie angefangen hätte. Großer Dank gebührt Michael Karcher, der uns viele mathematische Fragen erklärt hat und uns bei Schwierigkeiten mit Mathematica immer wieder aus der Patsche geholfen hat. Tobias Burnus hat sich stets Zeit genommen, um uns bei unzähligen Computerproblemen zu helfen.Tristan Faber schließlich hat mich, nach dem er diese Übung betreut hat, davon überzeugt, das Skript völlig neu zu gliedern und zu überarbeiten.

Inhaltsverzeichnis

1 Warum Sie diesen Versuch machen

Das Ziel dieser Praktikumsaufgabe ist es, Sie soweit mit den mathematischen Grundlagen der linearen Regression und der Software *Mathematica* vertraut zu machen, dass Sie diese Technik anstelle grafischer Auswertungen bei den folgenden Praktikumsversuchen sinnvoll einsetzen können.

Das Skript zu dieser Aufgabe ist wesentlich umfangreicher, als Sie es von anderen Versuchen gewohnt sind. Das liegt zum einen daran, dass wir uns bemüht haben die theoretischen Grundlagen ausführlich darzustellen und nicht nur summarisch zu wiederholen, zum anderen erhalten Sie eine Anleitung zu *Mathematica*, die die Lösung der ersten Teilaufgabe Schritt für Schritt vorführt. Mehr Text bedeutet für Sie in diesem Fall also hoffentlich nicht mehr sondern weniger Arbeit.

Eine Bitte: Dieses Skript wurde vollständig überarbeitet und ist daher sicher nicht fehlerfrei. Helfen Sie nachfolgenden Praktikanten; machen Sie uns auf Fehler und Unklarheiten aufmerksam!

2 Grundlagen

In diesem Abschnitt werden einige Begriffe aus der Einführung in die Fehlerrechung und Statistik im Grundpraktikum I wiederholt und vertieft.

2.1 Der Messprozess als Zufallsexperiment

Messen wir dieselbe Größe mehrfach, erhalten wir im allgemeinen verschiedene Messwerte $(x_1, x_2, \ldots)$. Mathematisch betrachtet man die Messung daher als Zufallsexperiment. Das Ergebnis X der Messung heißt *Zufallsgröße*. Jedes Messergebnis x_i ist eine Realisation der Zufallsgröße. Geht man von einer diskreten Zufallsgröße aus, wird bei jeder Messung genau einer der Werte $x_i \in \{x_1, \ldots, x_n\}$ angenommen. (z.B. Würfelexperiment) Bei einer Messung beobachtet man dann mit der Wahrscheinlichkeit p_i den Wert x_i.

2.2 kontinuierliche Verteilung (Wahrscheinlichkeitsdichtefunktion)

In der Praxis kommen meist kontinuierliche Größen vor. Hier kann man keine Wahrscheinlichkeit für einen diskreten Wert mehr angeben. Man verwendet stattdessen eine *Wahrscheinlichkeitsdichtefunktion* $P(x)$. Die Wahrscheinlichkeit einen Wert im Intervall $[x_a, x_b]$ zu messen, ist dann das bestimmte Integral über die Wahrscheinlichkeitsdichtefunktion. Ähnlich wie man bei der Integration über die Dichte die Masse erhält, erhält man bei der Integration über die W-Dichte eine Wahrscheinlichkeit. [Bar, chap. 3.1.4]

$$p(x \in [x_a, x_b]) = \int_{x_a}^{x_b} P(x)\mathrm{d}x \tag{1}$$

Genau wie bei diskreten Verteilungen die Summe über die Wahrscheinlichkeiten aller möglichen Ereignisse 1 ist, ist das Integral über den gesamten Wertebereich der Zufallsgröße 1. Man sagt die Verteilung ist *normiert*.

$$\int_{-\infty}^{+\infty} P(x)\mathrm{d}x = 1 \tag{2}$$

2.3 Gaußverteilung

Die für die Fehlerrechung wichtigste Verteilung ist die Gaußverteilung. Sie wird durch die Standardabweichung σ, die die Breite der Verteilung beschreibt, und den Erwartungswert μ, den Schwerpunkt der Verteilung, beschrieben.

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3}$$

Im folgenden werden wir stets davon ausgehen, dass unsere Messgrößen gaußverteilt sind. Warum das meist so ist, steht z.B. in [Bar, S. 49 ff].

2.4 Erwartungswert

Naiv möchten wir bei einer Messung den exakten "wahren" Wert x_w der Größe X erfahren, was auch abgesehen von der Frage ob es so etwas überhaupt gibt, auf Grund von unvermeidlichen Ungenauigkeiten des Messprozesses prinzipiell unmöglich ist. Als Ziel der Messung betrachten wir deshalb den Erwartungswert $\mu = \langle X \rangle$ der Messgröße. Im Fall einer diskreten Vertielung entspricht das Integral der mit den Eintrittswahrscheinlichkeiten gewichteten Summe über

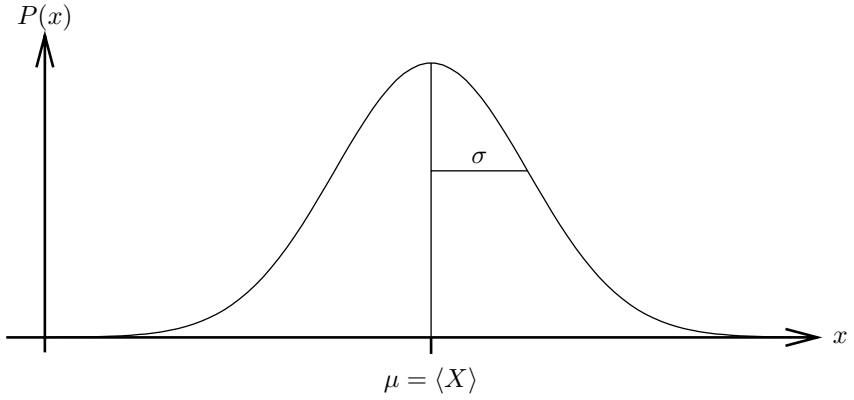


Abbildung 1: Gaußverteilung

alle möglichen Werte. Wir werden der Anschaulichkeit halber in diesem Skript dennoch häufiger den Begriff des "wahren" Wertes x_w verwenden.

$$\mu = \langle X \rangle = \int_{-\infty}^{\infty} xP(x)dx \tag{4}$$

Der Erwartungswert kann auch für eine Funktion f der Zufallsgröße X definiert werden, die dann selbst als Zufallsgröße Faufgefasst werden kann. Für $\langle f(X)\rangle$ wird häufig kurz $\langle f\rangle$ geschrieben.

$$\langle F\rangle = \langle f(X)\rangle = \langle f\rangle = \int_{-\infty}^{\infty} f(x)xP(x)\mathrm{d}x \tag{5}$$

Linearität des Erwartungswerts: Sind f und g Funktionen derselben Zufallsgröße X, dann gilt für Ihre Erwartungswerte:

$$\begin{aligned}\langle f+g\rangle &= \int_{-\infty}^{\infty} \Big(f(x)+g(x)\Big)P(x)dx \\ &= \int_{-\infty}^{\infty} f(x)P(x)dx + \int_{-\infty}^{\infty} g(x))P(x)dx = \langle f\rangle + \langle g\rangle\end{aligned} \tag{6}$$

[Bar, S. 22]

$$\begin{aligned}
\sigma^2 \overset{(7)}{=} \langle (X-\mu)^2\rangle &= \int_{-\infty}^{\infty} (x-\mu)^2 P(x)\mathrm{d}x \\
= \langle X^2 - 2\mu X + \mu^2\rangle &= \int_{-\infty}^{\infty} (x^2 - 2\mu x + \mu^2)P(x)\mathrm{d}x \\
= \langle X^2\rangle - \langle 2\mu X\rangle + \langle \mu^2\rangle &= \int_{-\infty}^{\infty} x^2P(x)\mathrm{d}x - \int_{-\infty}^{\infty} 2\mu x P(x)\mathrm{d}x + \int_{-\infty}^{\infty} \mu^2 P(x)\mathrm{d}x \\
= \langle X^2\rangle - 2\mu\langle X\rangle + \mu^2 &= \int_{-\infty}^{\infty} x^2P(x)\mathrm{d}x - 2\mu \underbrace{\int_{-\infty}^{\infty} xP(x)\mathrm{d}x}_{\overset{(4)}{=}\mu} + \mu^2 \underbrace{\int_{-\infty}^{\infty} P(x)\mathrm{d}x}_{\overset{(2)}{=}1} \\
= \langle X^2\rangle - 2\mu^2 + \mu^2 &= \int_{-\infty}^{\infty} x^2P(x)\mathrm{d}x - 2\mu^2 + \mu^2 \\
= \langle X^2\rangle - \langle X\rangle^2 &= \int_{-\infty}^{\infty} x^2P(x)\mathrm{d}x - \mu^2
\end{aligned}$$

Abbildung 2: *Beweis der Varianzformel (8)

2.5 Varianz, Standardabweichung, physikalische Fehler

In der Statistik ist die Varianz σ^2 *das* Maß für die Güte einer Messung der Größe X. Sie ist die mittlere quadratische Abweichung vom als bekannt vorausgesetzten Erwartungswert μ oder "wahren Wert" der Messung. Für kontinuierliche Verteilungen ist sie wie folgt definiert:

$$\sigma^2 := \langle (X-\mu)^2\rangle = \langle (X - \langle X\rangle)^2\rangle \overset{(5)}{=} \int_{-\infty}^{\infty} (x-\mu)^2 P(x)\mathrm{d}x \tag{7}$$

Für die Varianz gilt folgende vielverwendete Beziehung:

$$\sigma^2 = \langle X^2\rangle - \langle X\rangle^2 \tag{8}$$

In Worten: Die Varianz ist die Differenz aus dem Erwartungswert der Quadrate und dem Quadrat des Erwartungswerts.

Im Beweis (Abb. 2) wird die Linearität des Erwartungswerts verwendet.

Die *Standardabweichung* $\sigma := \sqrt{\sigma^2}$ hat dieselbe Dimension wie der Erwartungswert bzw. die Messgröße. Aus diesem Grund wird in der Physik als

Fehler Δx meist die Standardabweichung oder ein Schätzwert dafür angegeben. Warum man in der Statistik lieber die Varianz verwendet, werden wir in Abschnitt 3.3 erfahren.

2.6 Gaußsches Fehlerfortpflanzungsgesetz

Ist die gesuchte Größe $z = z(x_1, \ldots, x_n)$ eine Funktion mehrerer Messwerte $(x_1, \ldots, x_n)$ mit den Fehlern $(\sigma_1, \ldots, \sigma_n)$ und sind die Messwerte gaußverteilt und statistisch unabhängig gilt für den Fehler von z in erster Ordnung:

$$\sigma_z = \sqrt{\left(\frac{\partial z}{\partial x_1}\right)^2 \sigma_1^2 + \cdots + \left(\frac{\partial z}{\partial x_n}\right)^2 \sigma_n^2} = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial z}{\partial x_i}\right)^2 \sigma_i^2} \tag{9}$$

[Bar, S. 55ff], [Bev, S. 41f]

3 Paramerterschätzung

Wir werden am einfachen Beispiel des Mittelwerts neue Konzepte einführen und diese dann in den folgenden Kapiteln auf die Ausgleichsgerade übertragen.

3.1 Schätzfunktion

Stellen wir uns vor, wir haben dieselbe Größe X n-mal gemessen, dann können wir unsere Messreihe, die aus den Elementen x_i besteht, als Messwertevektor $\vec{x}$ darstellen:

$$\vec{x} = (x_1, \ldots, x_n) \tag{10}$$

Die Aufgabe besteht nun darin, aus den Messwerten $\vec{x} = (x_1, \ldots, x_n)$ einen *Schätzwert* $\hat{x}$ für den Erwartungswert $\langle X \rangle$ der Zufallsvariable X zu finden. Dieser Schätzwert muss offensichtlich aus den Messwerten $\vec{x} = (x_1, \ldots, x_n)$ berechnet werden. Er ist also eine Funktion der Messwerte. Diese Funktion heißt *Schätzfunktion*.

$$\hat{x} = \hat{x}(\vec{x}) = \hat{x}(x_1, \ldots, x_n) \tag{11}$$

Die Wahl dieser Funktion ist zunächst einmal willkürlich. Natürlich drängt sich in unserem Beispiel der gewöhnliche Mittelwert auf. In komplizierteren Fällen ist jedoch nicht immer klar, wie eine geeignete Schätzfunktion für das Problem aussieht.

Es gibt also für eine Aufgabenstellung prinzipiell beliebig viele verschiedene Schätzfunktionen.

Welche Eigenschaften charakterisieren nun eine *gute* Schätzfunktion? Wichtig sind die folgenden drei Eigenschaften, die jedoch auch bei häufig verwendeten Methoden nicht immer erfüllt werden können.

Konsistenz (asymptotische Erwartungstreue) Asympthotisch (für unendlich lange Messreihen) sollte der Schätzwert $\hat{x}(\vec{x})$ mit dem Erwartungswert des zu schätzenden Parameters übereinstimmen:

$$\lim_{n \to \infty} \Big(\hat{x}(x_1, \ldots, x_n)\Big) = \langle X \rangle \tag{12}$$

Erwartungstreue Die Konsistenz macht nur eine Aussage für große Stichproben. Wir wünschen uns aber auch, dass im Mittel über viele kleine Stichproben der Mittelwert der Schätzwerte mit dem "wahren" Wert des Parameters übereinstimmt. Das heißt, der Erwartungswert der Schätzfunktion $\langle \hat{x} \rangle$ soll gleich dem Erwartungswert $\langle X \rangle$ des zu schätzenden Parameters sein.

$$\langle \hat{x} \rangle = \langle X \rangle \tag{13}$$

Abweichungen des *Erwartungswerts der Schätzfunktion* vom Erwartungswert des Parameters der Ausgangsverteilung nennt man *Bias*.

Wirksamkeit (*efficiency*) Eine Schätzfunktion heißt wirksamer als eine andere, wenn ihre Varianz kleiner als die der anderen ist. Auf Fragen der Wirksamkeit können wir in diesem Skript nicht weiter eingehen.

3.2 Maximum-Likelihood-Methode

3.2.1 am Beispiel einfacher Mittelwert

Eine systematische Methode eine Schätzfunktion zu finden, ist die *Maximum-Likelihood-Methode* (ML-Methode). Bei der Herleitung dieses Prinzips geht man zunächst davon aus, dass die Wahrscheinlichkeitsdichtefunktion $P(x)$, und damit auch der *unbekannte zu schätzende Erwartungswert* $\langle X \rangle = \mu$ *bekannt sei* und fragt nach der Wahrscheinlichkeit, mit der diese Ursprungsverteilung zu den beobachteten Messwerten $\vec{x} = (x_1, \ldots, x_n)$ führt.

Die Wahrscheinlichkeit erst den Wert x_1, dann x_2 usw. in dieser Reihenfolge zu messen, ist das Produkt der Einzelwahrscheinlichkeiten $P(x)$. Auch für kontinuierliche Wahrscheinlichkeitsverteilungen gilt diese Regel. Als Ergebnis

erhält man eine Wahrscheinlichkeitsdichtefunktion $L(\vec{x})$, die die Wahrscheinlichkeit, den beobachteten Satz von Messwerten zu erhalten, beschreibt; die *Likelihood-Funktion*. Gehen wir davon aus, dass unsere Messwerte einer Gaußverteilung $P(x)$ folgen, erhält man:

$$\begin{aligned} L(\vec{x}) = L(x_1, \ldots, x_n) &= P(x_1)P(x_2)\cdots P(x_n) \\ &= \prod_i P(x_i) \\ &= \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \end{aligned} \tag{14}$$

Leider kennen wir den Erwartungswert $\langle X \rangle = \mu$ nicht. (Sonst brauchten wir X nicht zu messen!) Wir wollen deshalb jetzt μ so wählen, dass die Wahrscheinlichkeit die beobachten Messwerte zu erhalten maximal wird. (Maximum-Likelihood) *Dieser Wert von μ ist unser Schätzwert $\hat{x}$ für X!*

Suchen wir also das Maximum von L. Notwendige Bedingung für ein Maximum ist eine Nullstelle der ersten Ableitung. Wenn L ein Maximum aufweist, wird auch $\ln(L)$ maximal. Dieses Maximum ist leichter zu bestimmen, da das Produkt von Exponentialfunktionen so zu einer Summe wird, die sich leichter ableiten lässt.

$$\begin{aligned} \ln\left(L(\vec{x})\right) &= -n \ln\left(\sqrt{2\pi\sigma^2}\right) - \frac{1}{2\sigma^2}\sum_i (x_i - \mu)^2 \\ 0 = \left.\frac{\partial \ln L}{\partial \mu}\right|_{\mu=\hat{x}} &= \frac{1}{\sigma^2}\sum_i (x_i - \hat{x}) = \frac{1}{\sigma^2}\left(\left(\sum_i x_i\right) - n\hat{x}\right) \\ \hat{x} &= \frac{1}{n}\sum_{i=1}^{n} x_i (=: \bar{x}) \end{aligned} \tag{15}$$

Es ist nicht überraschend, dass die ML-Schätzfunktion $\hat{x}$ in unserem Beispiel das arithmetische Mittel $\bar{x}$ ist. Man müsste jetzt eigentlich zeigen, dass diese Schätzfunktion konsistent, erwartungstreu und wirksam ist. Darauf können wir jedoch an dieser Stelle leider nicht weiter eingehen.

Kommen wir nun zu einer Fragestellung mit etwas weniger offensichtlichem Ergebnis:

3.2.2 Der gewichtete Mittelwert

Angenommen eine Größe X wird mit *verschiedenen Methoden, mit unterschiedlichen Fehlern* σ_i gemessen. Wir werden naiv wieder eine Art Mittelwert bilden wollen. Aber wie sind die einzelnen Werte zu gewichten? Wir setzen wieder gaußverteilte Messwerte voraus. In der Sprache der Statistik entstammt jetzt jeder Messwert x_i einer anderen Verteilung P_i – jede mit eigenem σ_i aber gemeinsamem μ. Wenden wir nun die oben eingeführte ML-Methode an, um einen Schätzwert für den Erwartungswert von X zu erhalten:

$$\begin{aligned} L(\vec{x}) = L(x_1, \ldots, x_n) &= \prod_i P_i(x_i) = \prod_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{\frac{-(x_i-\mu)^2}{2\sigma_i^2}} \\ \ln\left(L(x_1, \ldots, x_n)\right) &= \sum_i -\ln\left(\sqrt{2\pi\sigma_i^2}\right) - \sum_i \frac{(x_i-\mu)^2}{2\sigma_i^2} \\ 0 = \left.\frac{\partial \ln L}{\partial \mu}\right|_{\mu=\hat{x}} &= \sum_i \left(\frac{x_i - \hat{x}}{\sigma_i^2}\right) = \sum_i \frac{x_i}{\sigma_i^2} - \hat{x}\sum_i \frac{1}{\sigma_i^2} \\ \hat{x} &= \frac{\sum_{i=1}^{n} \frac{x_i}{\sigma_i^2}}{\sum_{i=1}^{n} \frac{1}{\sigma_i^2}} (=: \bar{x}) \end{aligned} \tag{16}$$

Bei der Mittelwertbildung sind also statistisch unabhängige, gaußverteilte Messwerte mit $w_i = \frac{1}{\sigma_i^2}$ zu gewichten. Es lässt sich zeigen, dass diese ML-Schätzfunktion die wirksamste Schätzfunktion für diesen Fall ist.

3.2.3 Schätzung des Fehlers einer Messung aus der Streuung

Gehen wir noch einen Schritt weiter, nehmen wir an, wir haben eine Messung mehrfach mit *derselben Methode* wiederholt und kennen den Fehler der Messwerte nicht, dann sollte es möglich sein, aus der Abweichung der Messwerte vom Mittelwert *den Fehler einer einzelnen Messung* σ zu schätzen.

Wird eine Messung mehrfach mit derselben Methode wiederholt, dann haben alle Messwerte x_i denselben Fehler σ, denn sie stammen aus derselben Ausgangsverteilung $P(x)$. Diese soll auch in diesem Abschnitt gaußförmig sein.

Die Aufgabe besteht nun darin, die beiden Parameter σ^2 und μ gleichzeitig zu schätzen. Für den Schätzwert von μ bei konstantem σ^2 haben wir bereits in (15) das arithmetische Mittel gefunden:

$$\hat{x} = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{17}$$

Der Schätzwert für μ ist also von den Fehlern unabhängig, solange sie nur für alle Messwerte gleich groß sind.

Um auch eine Schätzung für σ zu erhalten, wenden wir wieder das ML-Verfahren an.

$$P(x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{18}$$

$$L(\vec{x}) = \prod_i \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \right) \tag{19}$$

$$\ln L = -n \ln\left(\sqrt{2\pi\sigma^2}\right) - \frac{1}{2\sigma^2} \sum_i (x_i - \mu)^2 \tag{20}$$

Es sind zwei Parameter zu bestimmen; wir erhalten also ein System aus zwei Gleichungen. Diese Gleichungen heißen *Normalgleichungen*.

$$0 = \left.\frac{\partial \ln L}{\partial \mu}\right|_{\substack{\mu=\hat{x} \\ \sigma^2=\widehat{\sigma^2}}} \Leftrightarrow 0 = \sum_i (x_i - \hat{x}) \Leftrightarrow \hat{x} = \frac{1}{n}\sum_i x_i =: \bar{x} \tag{21a}$$

$$0 = \left.\frac{\partial \ln L}{\partial (\sigma^2)}\right|_{\substack{\mu=\hat{x} \\ \sigma^2=\widehat{\sigma^2}}} = -\frac{n}{2\widehat{\sigma^2}} + \frac{1}{2\left(\widehat{\sigma^2}\right)^2} \sum_i (x_i - \hat{x})^2 \tag{21b}$$

Die erste Normalgleichung (21a) ist unabhängig von σ^2 und liefert das bekannte Ergebnis. Die zweite Normalgleichung (21b) liefert nach einsetzen von $\hat{x} = \bar{x}$ aus (21a) eine Schätzfunktion für σ^2:

$$\widehat{\sigma^2} = \frac{1}{n} \sum_i (x_i - \bar{x})^2 \tag{22}$$

Über ihre Konsistenz und Erwartungstreue haben wir keine Aussage gemacht!

Die Wahl von ML zur Gewinnung einer Schätzfunktion ist willkürlich. Wir hätten genausogut direkt die naheliegende Formel (22) wählen können. Prinzipiell liefern verschiedene Methoden jedoch unterschiedliche Schätzfunktionen. Welche davon gut sind, muss im Einzelfall untersucht werden. *22 ist nicht gut!* Wir werden diese Funktion deshalb im nächsten Abschnitt untersuchen und eine bessere Schätzfunktion (25) für unser Problem finden.

3.3 Das Problem der Erwartungstreue Bessels Korrektur am Beispiel von 3.2.3

Die in (22) hergeleitete Schätzfunktion für die Varianz lässt sich analog zu Abb. 2 umformen:

$$\begin{aligned} \widehat{(\sigma^2)} &= \frac{1}{n}\sum_i \big(x_i - \bar{x}(x_1,\ldots,x_n)\big)^2 = \frac{1}{n}\sum_i (x_i^2 - 2\bar{x}x_i + \bar{x}^2) \\ &= \frac{1}{n}\left(\sum_i x_i^2\right) - 2\bar{x}\underbrace{\frac{1}{n}\sum_i x_i}_{=\bar{x}} + \frac{1}{n}\sum_i \bar{x}^2 = \frac{1}{n}\sum_i \left(x_i^2 - \bar{x}^2\right) \end{aligned} \tag{23}$$

In dieser Form lässt sich zeigen, *dass diese Schätzfunktion nicht erwartungstreu ist!* Wir betrachten den Erwartungswert von $\widehat{(\sigma^2)}$. Dazu benötigen wir den Erwartungswert des Mittelwerts $\langle\bar{x}\rangle$.

Wir fassen den Mittelwert $\bar{x}(x_1,\ldots,x_n)$ als Funktion der Zufallsgrößen X_i auf.

Da jeder Messwert x_i eine Realisation derselben Zufallsgröße X ist, hängt der Mittelwert $\bar{x}(X_1,\ldots,X_n)$, nur von der einen Zufallsgröße $X = X_1 = \cdots = X_n$ ab: $\bar{x}(X,\ldots,X)$.

$\bar{x}$ ist eine erwartungstreue Schätzfunktion für X, deshalb gilt: $\langle X\rangle = \langle\bar{x}\rangle$; jedoch gilt dies nicht für die Erwartungswerte der Quadrate: $\langle X^2\rangle \neq \langle\bar{X}^2\rangle$!

$$\begin{aligned} \langle\widehat{\sigma^2}\rangle &= \left\langle \frac{1}{n}\left(\sum_i X^2\right) - \bar{X}^2 \right\rangle = \langle X^2\rangle - \langle\overline{X}^2\rangle \\ &= \langle X^2\rangle - \langle\overline{X}^2\rangle - \underbrace{\left(\langle X\rangle^2 - \langle\overline{X}\rangle^2\right)}_{=0} = \left(\langle X^2\rangle - \langle X\rangle^2\right) - \left(\langle\overline{X}^2\rangle - \langle\overline{X}\rangle^2\right) \\ &\overset{(8)}{=} \sigma_X^2 - \sigma_{\overline{X}}^2 \overset{(26)}{=} \sigma_X^2 - \frac{1}{n}\sigma_X^2 \\ &= \left(\frac{n-1}{n}\right)\sigma_X^2 \neq \sigma_X^2 \end{aligned} \tag{24}$$

Bessels Korrektur Offensichtlich ist die Schätzfunktion tatsächlich nicht erwartungstreu, allerdings konsistent, denn für große n geht der Faktor $\frac{n-1}{n}$ gegen 1. Ebenso offensichtlich ist, wie man aus dieser Funktion eine erwartungstreue Schätzfunktion erhält: Man multipliziert einfach mit dem Kehrwert des

störenden Faktors. Diese Korrektur heißt *Bessels Korrektur*, und die korrigierte Schätzfunktion wird üblicherweise s^2 genannt[1]:

$$s^2 = \frac{n}{n-1}\widehat{\sigma^2} = \frac{1}{n-1}\sum_i \left(x_i - \overline{x}\right)^2 \tag{25}$$

Dies ist also die Antwort auf die in (3.2.3) *gestellte Frage, wie der Fehler einer Messung aus der Streuung geschätzt werden kann!*

Freiheitsgrade Wie kann man dieses Ergebnis anschaulich deuten? Um den Fehler eines Messwertes aus der Streuung zu bestimmen, benötigt man zusätzliche Messungen. Man kann sinnvoll aus einer Messung einen Mittelwert (als Schätzwert für den Erwartungswert der Messung) bilden. Aus einem Punkt kann man jedoch keinen Schätzwert für die Streuung bestimmen. (22) liefert in diesem Fall stets den sinnlosen Wert 0. (25) hingegen ist sinnvoller Weise für $n = 1$ nicht definiert.

Man kann diese Überlegung verallgemeinern: Eine Gerade geht immer durch zwei Punkte und wird durch zwei Parameter (Steigung b und Achsenabschnitt a) vollständig bestimmt. Wurden nur zwei Punkte gemessen, ist eine Schätzung der Streuung aus der Abweichung von der Geraden nicht möglich.

Allgemein wird durch die Bestimmung eines Parameters ein Messwert verbraucht. Werden also m Parameter geschätzt, bleiben für die Schätzung der Fehler noch $(n - m)$ Werte übrig. Dies ist die Zahl der *Freiheitsgrade*.

Es gibt eine weitere Überlegung, die verständlich macht, warum Bessels Korrekturfaktor > 1 ist. s^2 ist ein Schätzwert für die Varianz σ^2, den mittleren quadratischen Abstand eines Messwerts vom Erwartungswert. Im allgemeinen werden die Messwerte $(x_1, \ldots, x_n)$ näher an *ihrem* Mittelwert $\bar{x}(x_1, \ldots, x_n)$, als am Erwartungswert der unterliegenden Verteilung liegen. (22) liefert also einen zu kleinen Wert.

Standardabweichung Für die Standardabweichung σ gibt es keine solche erwartungstreue Schätzfunktion! Glücklicherweise tritt in praktisch allen Rechnungen nur das Quadrat σ^2 auf, für das die erwartungstreue Schätzfunktion s^2 verwendet werden kann. Lediglich wenn $\Delta x = \sqrt{s^2}$ als Schätzung für den Fehler angegeben wird, tritt das Problem auf. Wenn man mit diesem Wert weiter rechnet, wird er aber wieder quadriert und die Welt ist in Ordnung. Man gibt nicht die Varianz direkt an, damit Wert und Fehler dieselbe Dimension haben.

[1] Auf Taschenrechnern wird $\sqrt{s^2}$ häufig mit σ_{n-1} bezeichnet.

4 Der Fehler der geschätzten Parameter am Beispiel Mittelwert

Bisher haben wir nur Messwerte und ihre Fehler betrachtet und daraus möglichst gute Schätzwerte für den "wahren Wert" unserer Messgröße ermittelt. In diesem Abschnitt untersuchen wir die Frage, wie man aus den Fehlern der einzelnen Messwerte einen Fehler für diesen *Schätzwert* erhalten kann.

4.1 Bekannte Messfehler

4.1.1 Fehler gleich groß

Beispiel: Ein Praktikumsbetreuer klatscht zweimal vor der von ihm betreuten Gruppe von 6 Studenten, ausgestattet mit je einer Stoppuhr mit $^1/_{100}$ s Auflösung, für diese nicht sichtbar mit einer Knallpatsche. Die Gruppe hat die Aufgabe, die Zeit zwischen den Ereignissen möglichst genau zu bestimmen. Der Betreuer gibt an, die Reaktionszeitdifferenz zwischen Start und Stopp solle mit $\Delta t = 0.1\,\mathrm{s}$ als Fehler angenommen werden.

Wurde dieselbe Größe, wie im Beispiel mehrfach unabhängig mit derselben Methode gemessen und ist die Varianz für alle Werte gleich und bekannt, geben wir als Ergebnis der Messung den ungewichteten Mittelwert nach (15) an. Wir erwarten, dass der Fehler dieses Mittelwertes kleiner ist, als der eines einzelnen Wertes. Wie groß der Fehler ist, erfahren wir, in dem wir das Gaußsche Fehlerfortpflanzungsgesetz (9) auf die Mittelwertbildung (15) anwenden:

$$\bar{x} = \frac{1}{n}\sum_i x_i \Rightarrow \sigma_{\bar{X}} = \frac{1}{\sqrt{n}}\sigma \tag{26}$$

Das Ergebnis ist auch als $\sqrt{n}$-Regel bekannt. Wie sind diese Fehler nun zu interpretieren? Sicher handelt es sich nicht um die Standardabweichung σ der Ursprungsverteilung. Aber in der Theorie der Fehlerrrechnung interpretieren wir die Fehler als Standardabweichungen von Zufallsgrößen. Auch der oben berechnete Fehler des Mittelwertes $\sigma_{\bar{x}}$ ist eine Standardabweichung, und zwar die einer Verteilungsfunktion der Mittelwerte.

Wiederholen wir das Experiment, erhalten wir viele solche Mittelwerte die aus einer Gaußverteilung mit dem Erwartungswert $\mu_x = \mu_{\bar{x}}$ aber mit der Standardabweichung $\sigma_{\bar{x}}$ stammen. (Abb. 3)

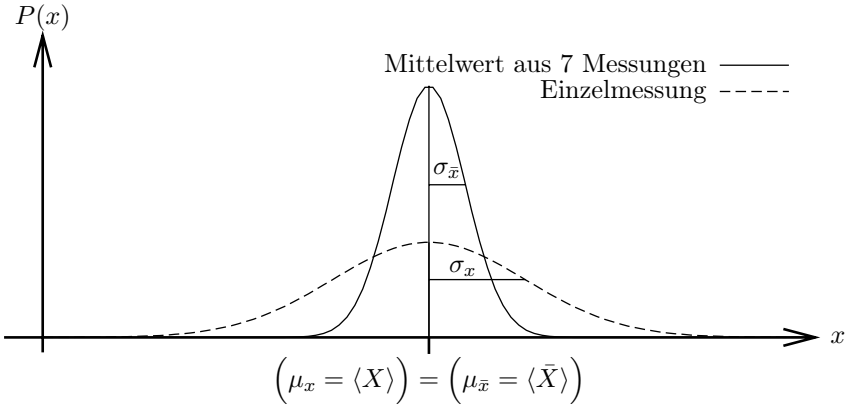


Abbildung 3: Wahrscheinlichkeitsdichtefunktionen für Einzelmessung und Mittelwert

4.1.2 Fehler unterschiedlich groß

Beispiel: Die Studenten 1,2,3 und 4 sind ausgeschlafen, 5 und 6 sind müde. Der Betreuer gibt an, für unausgeschlafene Praktikanten sei ein Fehler von $\Delta t = 0.2s$ zu verwenden. Wir setzen also $\sigma_1 = \sigma_2 = \sigma_3 = \sigma_4 = 0.1\,\mathrm{s}$ und $\sigma_5 = \sigma_6 = 0.2\,\mathrm{s}$.

Wurde dieselbe Größe mit unterschiedlichen Methoden i mit unterschiedlichen bekannten Fehlern σ_i gemessen, verwenden wir den gewichteten Mittelwert nach (16) und erhalten durch Anwendung des Fehlerfortpflanzungsgesetzes:

$$\bar{x} = \frac{\sum_{i=1}^{n} \frac{x_i}{\sigma_i^2}}{\sum_{i=1}^{n} \frac{1}{\sigma_i^2}} \quad \Rightarrow \quad \sigma_{\bar{x}} = \frac{1}{\sqrt{\sum_i \frac{1}{\sigma_i^2}}} \tag{27}$$

4.2 Unbekannte Fehler - alle Fehler gleich groß

Bisher haben wir stets Situationen betrachtet, in denen der Fehler eines Messwerts unabhängig von der Messung bekannt war. (z.B.: vom Betreuer, aus einer Bedienungsanleitung, durch eine Schätzung des Ablesefehlers, ...). Der Fehler der Messung war also prinzipiell *a priori*, also bevor die Messung durchgeführt wurde, bekannt.

Nun gehen wir davon aus, dass wir aus *derselben* Messreihe sowohl unser Messergebnis (Einen Schätzwert für den Erwartungswert) bestimmen wollen, als auch eine Schätzwert für den Fehler dieses Ergebnisses.

Gehen wir wieder davon aus, dass alle Messwerte denselben Fehler haben, d.h., dass sie aus derselben Wahrscheinlichkeitsverteilung mit der *unbekannten* Varianz σ^2 stammen, dann ist s^2 (25) ein Schätzwert für die Varianz eines *einzelnen* Wertes x_i. Wir können diesen Schätzwert nun in Gleichung (26) für den Fehler des Mittelwertes einsetzen und erhalten so:

$$\sigma_{\overline{X}} \approx \sqrt{\frac{s^2}{n}} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})}{n(n-1)}} \tag{28}$$

4.3 *Bekannte Fehler / aus der Streung geschätzt Fehler

Betrachtet man eine gaußverteilte Messgröße x mit bekannter Standardabweichung σ. x_w sei ihrer wahrer Wert, x_i der Messwert. Man kann diese Angabe auf zwei Arten interpretieren.

Zum einen ist diese Standardabweichung definitionsgemäß die Wurzel aus dem mittleren quadratischen Fehler.

Zum anderen kann man die Standardabweichung als Konfidenzintervall auffassen. Denn auf Grund der zugrundeliegenden Gaußverteilung, liegt der Messwert x_i mit einer Wahrscheinlichkeit von 68% (1σ-Wahrscheinlichkeit) in dem Intervall $]x_\mathrm{w} - \sigma; x_\mathrm{w} + \sigma[$. Dann gilt natürlich auch: „Der wahre Wert x_w liegt mit der selben Wahrscheinlichkeit im Intervall $]x_i - \sigma; x_i + \sigma[$ um den Messwert x_i. Genau so sind Konfidenzintervalle definiert. Dabei kann die Wahrscheinlichkeit, die jetzt statistische Sicherheit oder Vertrauensnivau heißt, beliebig vorgegeben werden. Die Intervallgröße ändert sich dann entsprechend.

Wenn die Standardabweichung nun nicht a priori bekannt ist, sondern geschätzt wird, erhält man zwar einen Schätzwert für die Wurzel aus dem mittleren quadratischen Fehler, die zweite Interpretationsmöglichkeit als Konfidenzintervall geht allerdings verloren, weil die Grenzen des Intervalls selbst mit einer Unsicherheit behaftet sind.

Dennoch ist es auch in diesem Fall möglich, wieder Konfidenzintervalle anzugeben. Diese sind allerdings für kleine n wesentlich größer, als durch die Schätzwerte der Standardabweichung suggeriert wird. Lediglich für große n stimmen diese Konfidenzintervalle mit den geschätzten Standardabweichungen überein.

Wir können hier nicht auf die Berechnung der Konfidenzintervalle aus den Schätzwerten der Standardabweichung eingehen. Mehr dazu findet man in [Mand, S. 114]. [Bar, S. 134ff] Stichwort: Students t Verteilung.

5 Lineare Regression

In diesem Abschnitt kommen wir endlich zu der Frage, wie man die Lage einer Ausgleichsgerade durch die Messpunkte bestimmen kann. Um den Leser nicht zu ermüden, betrachten wir von Anfang an den Fall, dass die einzelnen Messwerte unterschiedliche Fehler haben.

Wir haben Paare von Punkten (x_i, y_i) gemessen. Wir nehmen an, dass der Zusammenhang zwischen x und y durch eine Gerade, d.h., durch eine Funktion vom Typ

$$Y(X) = A + BX \tag{29}$$

beschrieben werden kann. Diese Funktion heißt *Modellfunktion*. Wir können unsere Messdaten durch zwei Vektoren $\vec{x} = (x_1, \ldots, x_n)$ und $\vec{y} = (y_1, \ldots, y_n)$ darstellen. A und B sind Zufallsgrößen und unsere Aufgabe besteht darin, möglichst gute Schätzwerte $\hat{a}$ und $\hat{b}$ für Ihre Erwartungswerte $\langle A\rangle$ und $\langle B\rangle$ (anschaulich ihre wahren Werte a_w, b_w) zu bestimmen. Wir machen folgende Voraussetzungen:

- die x_i seien vom Experimentator frei wählbar und ihre Fehler seien gegenüber denen der y_i vernachlässigbar klein. In diesem Fall besteht also kein Unterschied zwischen den Messwerten x_i und ihren Erwartungswerten $\langle X_i\rangle$ bzw. ihren wahren Werten $x_{\mathrm{w}i}$.
- Jeder einzelne Messwert y_i stammt aus einer *eigenen* Gaußverteilung mit dem Erwartungswert $\mu_i = \langle Y_i\rangle$ und der Standardabweichung σ_i. Dies wird in Abb. 4 verdeutlicht.

In Abb. 5 ist der Unterschied zwischen Mess-, Schätz-, und Erwartungswert dargestellt.

5.1 Schätzung der Parameter A und B der Geraden

Wenden wir nun die Maximum-Likelihoodfunktion auf dieses Problem an: Für jede Stelle x_i gibt es nun einen "wahren Wert" $y_{i\mathrm{w}} = y_\mathrm{w}(x_i) = \langle Y(x_i)\rangle = \langle Y_i\rangle$. Die Wahrscheinlichkeit an der Stelle x_i die Messung y_i zu machen, wird durch die folgende Gaußfunktion beschrieben:

$$P_i(y_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(y_i - \langle Y_i\rangle)^2}{2\sigma_i^2}\right) \tag{30}$$

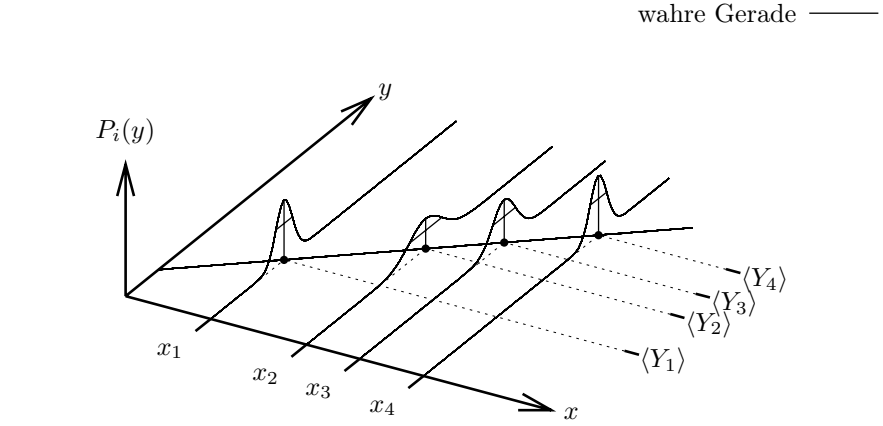


Abbildung 4: Wahrscheinlichkeitsdichte bei der linearen Regression

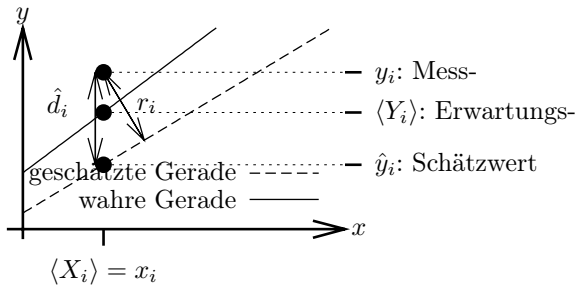


Abbildung 5: Unterschied zwischen Mess-, Schätz-, und Erwartungswert bei der linearen Regression

Die Likelihoodfunktion, also die Wahrscheinlichkeitsdichte, wenn man an den Stellen $\vec{x}$ misst die Werte $\vec{y}$ zu beobachten, ergibt sich wieder als Produkt der Einzelwahrscheinlichkeiten:

$$L(\vec{y}) = L(y_1, \ldots, y_n) = \prod_i P_i = \prod_i \left(\frac{1}{\sqrt{2\pi\sigma_i^2}} \right) \exp\left(-\frac{(y_i - \langle Y_i \rangle)^2}{2\sigma_i^2} \right) \tag{31}$$

Die $\langle Y_i \rangle$ können wir nun mit Hilfe des Modells (29) durch die Erwartungswerte für die Parameter $\langle A \rangle$ und $\langle B \rangle$ ausdrücken:

$$\langle Y_i \rangle = \langle A \rangle + \langle B \rangle x_i \tag{32}$$

Setzen wir dies in (31) ein, erhalten wir:

$$\begin{aligned} L(\vec{y}) = \prod_i P_i &= \prod_i \left(\frac{1}{\sqrt{2\pi\sigma_i^2}} \right) \exp\left(-\frac{(y_i - \langle A \rangle - \langle B \rangle x_i)^2}{2\sigma_i^2} \right) \\ &= \left(\prod_i \left(\frac{1}{\sqrt{2\pi\sigma_i^2}} \right) \right) \exp\left(\sum_i \left(-\frac{(y_i - \langle A \rangle - \langle B \rangle x_i)^2}{2\sigma_i^2} \right) \right) \end{aligned} \tag{33}$$

Um die Werte für A und B zu finden, die unsere Messwerte am wahrscheinlichsten machen, müssen wir nun die Likelihoodfunktion maximieren. Mit denselben Überlegungen wie in 3.2 bestimmen wir deshalb die Nullstellen der Ableitung von $\ln L$.

$$0 = \left. \frac{\partial \ln L}{\partial(\langle A \rangle)} \right|_{\substack{\langle A \rangle = \hat{a} \\ \langle B \rangle = \hat{b}}} \quad \bigwedge \quad 0 = \left. \frac{\partial \ln L}{\partial(\langle B \rangle)} \right|_{\substack{\langle A \rangle = \hat{a} \\ \langle B \rangle = \hat{b}}} \tag{34}$$

Da (33) verhältnismäßig lang ist, ist es sicherlich gut, sich auf den Term zu beschränken, der für die Ableitung von Bedeutung ist. Das ist allein die Summe im Exponenten. Diese Summe nennen wir nun χ^2 (chiquadrat).[2] Um L zu maximieren, müssen wir diese Summe minimieren.

$$\chi^2 := \sum_i \left(\frac{y_i - \hat{y}(x_i)}{\sigma_i} \right)^2 = \sum_i \left(\frac{1}{\sigma_i} (y_i - \hat{a} - \hat{b} x_i) \right)^2 \tag{35}$$

Wir wollen also die beste Ausgleichsgerade dadurch finden, dass wir Werte für die Parameter $\hat{a}$ und $\hat{b}$ finden, die die gewichtete Summe der Quadrate der Abweichungen χ^2 minimieren. Wir wollen also die Ausgleichsgerade finden, die die kleinste Summe der Quadrate hervorruft. Daher der Name *Methode der kleinsten Quadrate* oder auf Englisch *least-squares fit* für das Verfahren.

[2] Diese Größe wird häufig auch *WSSR* („**W**eighted **S**um of **S**quared **R**esiduals“) genannt. Eine analoge Größe findet sich auch schon bei der Herleitung des Mittelwertes in (14).

Minimierung von χ^2: Um die Werte der Parameter $\hat{a}$ und $\hat{b}$, die χ^2 minimieren, zu finden, bilden wir die partiellen Ableitungen von χ^2 nach den Parametern und setzen diese gleich Null. Die so erhaltenen Gleichungen heißen *Normalgleichungen*.

$$\frac{\partial}{\partial \hat{a}} \chi^2 = \frac{\partial}{\partial \hat{a}} \sum_i \left(\frac{1}{\sigma_i^2} (y_i - \hat{a} - \hat{b} x_i)^2 \right) = -2 \sum_i \left(\frac{1}{\sigma_i^2} (y_i - \hat{a} - \hat{b} x_i) \right) = 0 \tag{36a}$$

$$\frac{\partial}{\partial \hat{b}} \chi^2 = \frac{\partial}{\partial \hat{b}} \sum_i \left(\frac{1}{\sigma_i^2} (y_i - \hat{a} - \hat{b} x_i)^2 \right) = -2 \sum_i \left(\frac{x_i}{{\sigma_i}^2} (y_i - \hat{a} - \hat{b} x_i) \right) = 0 \tag{36b}$$

Die Normalgleichungen bilden ein lineares Gleichungssystem in zwei Variablen $\hat{a}$ und $\hat{b}$, das leicht gelöst werden kann:

$$\left(\sum_i \frac{1}{\sigma_i^2} \right) \hat{a} + \left(\sum_i \frac{x_i}{\sigma_i^2} \right) \hat{b} = \left(\sum_i \frac{y_i}{\sigma_i^2} \right) \tag{37a}$$

$$\left(\sum_i \frac{x_i}{\sigma_i^2} \right) \hat{a} + \left(\sum_i \frac{x_i^2}{\sigma_i^2} \right) \hat{b} = \left(\sum_i \frac{x_i y_i}{\sigma_i^2} \right) \tag{37b}$$

Um die Lösung übersichtlicher darstellen zu können, führen wir als Abkürzung die Determinante S der Koeffizientenmatrix ein.

$$S = \sum_i \frac{1}{\sigma_i^2} \sum_i \frac{x_i^2}{\sigma_i^2} - \left(\sum_i \frac{x_i}{\sigma_i^2} \right)^2$$

$$\hat{a} = \frac{1}{S} \left(\sum_i \frac{x_i^2}{\sigma_i^2} \sum_i \frac{y_i}{\sigma_i^2} - \sum_i \frac{x_i}{\sigma_i^2} \sum_i \frac{x_i y_i}{\sigma_i^2} \right) \tag{38a}$$

$$\hat{b} = \frac{1}{S} \left(\sum_i \frac{1}{\sigma_i^2} \sum_i \frac{x_i y_i}{\sigma_i^2} - \sum_i \frac{x_i}{\sigma_i^2} \sum_i \frac{y_i}{\sigma_i^2} \right) \tag{38b}$$

Dies sind also die ML-Schätzfunktionen $\hat{a}(\vec{x}, \vec{y})$ und $\hat{b}(\vec{x}, \vec{y})$.

5.2 Berechnung der Fehler der Parameter

5.2.1 a priori-Fehler

Gehen wir wie in Abschnitt 4.1.1 davon aus, dass die Fehler σ_i der Messwerte y_i unabhängig von der Messung bekannt sind, (aus der Bedienungsanleitung etc.) dann erhalten wir mit dem Gaußschen Fehlerfortpflanzungsgesetz die Fehler der Parameter $\hat{a}$ und $\hat{b}$.

$$\begin{aligned}
\sigma_a^2 &= \sum \left(\frac{\partial \hat{a}}{\partial y_i}\right)^2 \sigma_i^2 \\
&= \sum_{i=1}^{N} \frac{\sigma_i^2}{S^2} \left[\frac{1}{\sigma_i^4}\left(\sum_j \frac{x_j^2}{\sigma_j^2}\right)^2 - \frac{2x_i}{\sigma_i^4}\sum_j \frac{x_j^2}{\sigma_j^2}\sum_j \frac{x_j}{\sigma_j^2} + \frac{x_i^2}{\sigma_i^4}\left(\sum_j \frac{x_j}{\sigma_j^2}\right)^2\right] \\
&= \frac{1}{S^2}\left(\sum \frac{x_j^2}{\sigma_j^2}\right)\left[\sum \frac{1}{\sigma_i^2}\sum \frac{x_j^2}{\sigma_j^2} - \left(\sum \frac{x_j}{\sigma_j^2}\right)^2\right] \\
&= \frac{1}{S}\sum_j \frac{x_j^2}{\sigma_j^2} \qquad (39a)
\end{aligned}$$

$$\begin{aligned}
\sigma_b^2 &= \sum \left(\frac{\partial \hat{b}}{\partial y_i}\right)^2 \sigma_i^2 \\
&= \sum_{i=1}^{N} \frac{\sigma_i^2}{S^2} \left[\frac{x_i^2}{\sigma_i^4}\left(\sum_j \frac{1}{\sigma_j^2}\right)^2 - \frac{2x_i}{\sigma_i^4}\sum_j \frac{1}{\sigma_j^2}\sum_j \frac{x_j}{\sigma_j^2} + \frac{1}{\sigma_i^4}\left(\sum_j \frac{x_j}{\sigma_j^2}\right)^2\right] \\
&= \frac{1}{S^2}\left(\sum \frac{x_j^2}{\sigma_j^2}\right)\left[\sum \frac{1}{\sigma_i^2}\sum \frac{1}{\sigma_j^2} - \left(\sum \frac{x_j}{\sigma_j^2}\right)^2\right] \\
&= \frac{1}{S}\sum_j \frac{1}{\sigma_j^2} \qquad (39b)
\end{aligned}$$

Für die Fehler $\Delta\hat{a}$ und $\Delta\hat{b}$ der Parameter $\hat{a}$ und $\hat{b}$ gilt nun

$$\Delta\hat{a} = \sqrt{\sigma_a^2} \qquad \text{und} \qquad \Delta\hat{b} = \sqrt{\sigma_b^2} \tag{40}$$

Im Spezialfall gleicher Fehler ($\sigma_i = \sigma$) in den gemessenen Werten y_i, vereinfachen sich die Ausdrücke (38) und (39) zu

$$S' = \frac{S}{\sigma^4} = n\sum x_i^2 - \left(\sum x_i\right)^2$$

$$\hat{a} = \frac{1}{S'}\left(\sum_i x_i^2 \sum_i y_i - \sum_i x_i \sum_i x_i y_i\right) \qquad \sigma_a^2 = \frac{\sigma^2}{S'}\sum x_i^2 \tag{41a}$$

$$\hat{b} = \frac{1}{S'}\left(\sum_i x_i y_i - \sum_i y_i\right) \qquad \sigma_b^2 = N\frac{\sigma^2}{S'} \tag{41b}$$

5.2.2 Fehler aus der Streuung

In diesem Abschnitt wollen wir analog zu 4.2 aus *derselben Messreihe* sowohl die Parameter A und B, als auch ihre Fehler schätzen.

Absolute Fehler gleich

Die y-Werte sollen alle denselben unbekannten Fehler $\sigma_i = \sigma$ aufweisen. Wir möchten nun diesen gemeinsamen Fehler schätzen, um daraus Fehler für Steigung und Achsabschnitt berechnen zu können. In (41) haben wir gezeigt, dass für konstante Fehler die Parameter A und B unabhängig vom Fehler geschätzt werden können. Dies ermöglicht es, an jeder Stelle x_i einen zugehörigen Funktionswert $\hat{y}_i$ zu schätzen. Die Abstände von Schätzwert $\hat{y}$ und Messwert y_i heißen *Residuen* $\hat{d}_i$. Das Dach auf dem d rührt daher, dass es sich nicht um die Abweichung von den wahren Werten y_{iw}, sondern um Schätzwerte für diesen Abstand handelt. (Abb. 5)

$$\hat{d}_i = y_i - \hat{y}_i = y_i - \hat{a} - \hat{b}x \tag{42}$$

Als Schätzwert für die Varianz σ^2 der y-Werte wählen wir nun in naheliegender Weise:

$$\widehat{\sigma^2} = \frac{1}{n}\sum \hat{d}_i^{\,2} \tag{43}$$

Die Anwendung des ML-Prinzips hätte auch in diesem Fall wieder auf die Methode der kleinsten Quadrate und damit auf das obige Ergebnis geführt.

Der so gewonnene Schätzwert ist nur konsistent, aber nicht erwartungstreu, wir wenden daher Bessels Korrektur analog zu 3.3 an und definiert die erwartungstreue Schätzfunktion s^2 für die Varianz eines einzelnen Messpunktes y_i. Da 2 Parameter a und b geschätzt wurden, ist die Zahl der Freiheitsgrade

hier $(n-2)$.

$$s^2 = \frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n-2}\sum_{i=1}^{n}\left(y_i - \hat{a} - \hat{b}x\right)^2 \tag{44}$$

Aus der so geschätzen Varianz der Messwerte kann man in der Näherung $\sigma^2 \approx s^2$ die Fehler von $\hat{a}$ und $\hat{b}$ nach (39) schätzen. Bei dieser Schätzung tritt wieder das Problem auf, dass der Fehler mit dem s behaftet ist in diese Überlegung nicht eingeht. Deshalb gilt für die so geschätzten $\Delta\hat{a}$ und $\Delta\hat{b}$ das in 4.3 gesagte. Bei wiederholter Durchführung der Messreihe fluktuieren die Varianzen s^2 bei kleinem n (Zahl der Punkte einer Messreihe) sehr stark.

Absolute Fehler verschieden – Verhältnis bekannt (gewichtete Streuung)

Beispiel: Exponentialfunktion Wir untersuchen eine Messreihe bei der der Zusammenhang zwischen x und z durch eine Exponentialfunktion $Z(x) = \exp(A + Bx)$ beschrieben wird. Wir wissen, dass alle Punkte z_i denselben unbekannten Fehler σ aufweisen. Wir möchten diesen Fehler und damit auch die Fehler der Parameter a und b aus derselben Messreihe schätzen.

Um die Daten auswerten zu können, müssen wir die z_i logarithmieren und den unbekannten Fehler der z_i nach dem Fehlerfortpflanzungsgesetz umrechnen. Wir erhalten So einen neuen Datensatz mit $y_i = ln(z_i)$ und $\sigma_i \overset{(9)}{=} \frac{1}{y_i}\sigma$. Die Fehler der y_i können also in der Form $\sigma_i = f_i\sigma$ dargestellt werden. In diesem Beispiel ist $f_i = \frac{1}{y_i}$.

Im Beispiel wurde eine Situation dargestellt, in der die Fehler der einzelnen Punkte zwar nicht bekannt sind, wir jedoch ihre relative Größe untereinander kennen. Im Praktikum ist zwar vermutlich die Exponentialfunktion der einzige (und nicht seltene) Fall in dem dies vorkommt, da andere linearisierbare Zusammenhänge auf andere f_i führen, wollen in diesem Abschnitt allgemein den Fall untersuchen, in dem sich die unbekannten Fehler σ_i als Produkt aus einem bekannten individuellen Faktor f_i und dem unbekannten gemeinsamen Anteil σ darstellen lässt.

$$\sigma_i = f_i\sigma \tag{45}$$

Die Parameter $\hat{a}$ und $\hat{b}$ können wir weiterhin problemlos nach (38) schätzen. Dazu setzen wir (45) in (38) ein. Dabei kürzt sich der unbekannte gemeinsame Faktor σ heraus! In (38) treten lediglich die f_i an die Stelle der σ_i. Die Gewichte der Messwerte sind also $w_i = \frac{1}{f_i^2}$:

$$S' = \frac{S}{\sigma^4} = \sum_i \frac{1}{f_i^2}\sum_i \frac{x_i^2}{f_i^2} - \left(\sum_i \frac{x_i}{f_i^2}\right)^2$$

$$\hat{a} = \frac{1}{S'}\left(\sum_i \frac{x_i^2}{f_i^2}\sum_i \frac{y_i}{f_i^2} - \sum_i \frac{x_i}{f_i^2}\sum_i \frac{x_i y_i}{f_i^2}\right) \tag{46a}$$

$$\hat{b} = \frac{1}{S'}\left(\sum_i \frac{1}{f_i^2}\sum_i \frac{x_i y_i}{f_i^2} - \sum_i \frac{x_i}{f_i^2}\sum_i \frac{y_i}{f_i^2}\right) \tag{46b}$$

Wir schätzen nun analog zu (44) das gemeinsame σ^2. Dabei müssen wir beachten, dass es nicht mehr nur einen Erwartungswert $\langle Y\rangle$, sondern zu jedem Messwert eine Erwartungswert $\langle Y_i\rangle = \langle y(x_i)\rangle$ gibt. So erhalten wir die folgende erwartungstreue Schätzung:

$$s^2 = \frac{\chi^2}{n-2} = \frac{1}{n-2}\sum_{i=1}^{n}\frac{1}{f_i^2}\left(y_i - \hat{a} - \hat{b}x_i\right)^2 \tag{47}$$

Nun können wir durch Einsetzen von $\sigma_i^2 \simeq f_i^2 s^2$ in (39a) und (39b) Schätzungen für die Fehler der Parameter $\hat{a}$ und $\hat{b}$ berechnen.

$$\Delta\hat{a} = \sqrt{\frac{1}{S'}\sum_{i=1}^{n}\frac{x_i^2}{f_i^2}s^2} \qquad \text{und} \qquad \Delta\hat{b} = \sqrt{\frac{1}{S'}\sum_{i=1}^{n}\frac{1}{f_i^2}s^2} \tag{48}$$

Geschafft! Sie sind fast am Ziel!
Noch ein paar Hinweise zu den so aus der Streuung berechneten Fehlern:

- $\Delta\hat{a}$ und $\Delta\hat{b}$ sind für kleine n nicht verlässlich. Hier müssten eigentlich Konfidenzintervalle betrachtet werden. (Siehe **4.3**). Ein Beispiel, in dem Konfidenzintervalle mit der geschätzten Standardabweichung verglichen werden, findet man in [Mand, S. 141-147]

- Die f_i sind nur bis auf einen gemeinsamen Faktor bestimmt. $\frac{\chi^2}{n-2}$ ist von diesem Faktor ebenfalls abhängig. Die daraus berechneten $\Delta\hat{a}$ und $\Delta\hat{b}$ sind davon unabhängig.

- Der Schätzwert für die Varianz eines Messpunktes ist $s_i^2 = f_i^2 s^2 = f_i^2\frac{\chi^2}{n-2}$, auch wenn die Größe $\frac{\chi^2}{n-2}$ in *Mathematica* auch im Fall unterschiedlicher

Gewichte als `EstimatedVariance` bezeichnet wird. Die Bezeichnung ist dennoch nicht falsch, in unserem Beispiel mit der Exponentialfunktion ist $s^2 = \frac{\chi^2}{n-2}$ tatsächlich ein Schätzwert für die Varianz der nicht logarithmierten Werte.

6 Mehr zur linearen Regression

6.1 Die Bedeutung von χ^2 bei *bekannten* Fehlern

Die in Abschnitt 5.2.2 hergeleiteten Relationen geben uns jetzt ein Mittel in die Hand, die Güte unserer linearen Approximation zu beurteilen, *falls* wir die Fehler der Messwerte *doch* kennen.

Wählt man $f_i = \sigma_i$, dann ist σ definitionsgemäß gleich 1. Der Erwartungswert für s^2 ist (da s^2 erwartungstreu ist) dann ebenfalls gleich 1. Weicht $s^2 = \frac{\chi^2}{n-m}$ [3] stark von 1 ab, können wir mit einer gewissen Wahrscheinlichkeit schließen, dass die Eingangsfehler σ_i schlecht geschätzt wurden. Um dieses Werkzeug systematisch einsetzen zu können, müsste man sich Gedanken machen, mit welcher Wahrscheinlichkeit $\frac{\chi^2}{n-m}$ nicht mehr als um einen bestimmten Betrag vom Erwartungswert 1 abweicht. Darauf können wir hier nicht weiter eingehen.

Im Praktikum werden Sie in der Regel $\frac{\chi^2}{n-m} \ll 1$ beobachten, weil die Eingangsfehler auch systematische Anteile enthalten und nach oben – also zu groß – geschätzt wurden. Erhalten Sie allerdings $\frac{\chi^2}{n-m} > 1$, sollten Sie sich Gedanken machen, ob nicht ungenauer gemessen wurde, als man hinterher angegeben hat oder Fehlerquellen übersehen wurden. Bei einer ausreichenden Anzahl von Messpunkten kann man auch eine Fehlerschätzung nach Abschnitt 5.2.2 in Erwägung ziehen. Dies entspricht etwa dem Vorgehen beim Zeichnen einer Grenzgerade, bei der man sich an der Lage der Punkte orientiert.

6.2 Falsches Modell

Wir haben bisher immer vorausgesetzt, dass der wahre Zusammenhang tatsächlich ein Gerade ist. Wenn dies nicht der Fall ist, sind die Schätzwerte für die Parameter *und* die Fehler bedeutungslos. Es kann leicht passieren, dass Achsabschnitt und Steigung mit sehr kleinem Fehler geschätzt werden, aber

[3] In diesem Kontext wird s^2 in der Literatur praktisch ausschließlich als „reduced chi squared“ bezeichnet. Um dieser Konvention zu entsprechen, schreiben wir an dieser Stelle $\frac{\chi^2}{n-m}$, dabei ist $(n-m)$ die Zahl der Freiheitsgrade.

eine graphische Darstellung offenbart, dass die Werte überhaupt keinen linearen Verlauf zeigen. Dass die Fehler trotzdem klein sind, liegt daran, dass in ihre Berechnung die tatsächliche Abweichung der Messwerte von der Schätzgeraden überhaupt nicht eingeht! Verfahren, die diese Abweichungen berücksichtigen, liefern in einem solchen Fall zwar größere Fehler (siehe später). Aber die damit berechneten Werte sind genauso sinnlos. Deshalb ist die graphische Darstellung auch bei rechnerischer Auswertung so wichtig. Sie werden sich in Aufgabe 2 mit diesem Problem beschäftigen.

6.3 Das Bestimmtheitsmaß R^2

Die Residuen ($\hat{d}_i = y_i - \hat{y}_i$) beschreiben, wie gut jeder einzelne Punkt durch das Modell vorhergesagt wird. Wir möchten jedoch eine Größe finden, die beschreibt, wie gut das Modell insgesamt passt. Hierzu definieren wir zunächst die Streuung der Messwerte y_i als ihre quadratische Abweichung vom ihrem Mittelwert $\bar{y}$. Die Summe über diese Abweichungen nennt man SQT („**S**um of S**q**uares **T**otal“):

$$SQT = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{49}$$

Um nun ein Modell zu beurteilen, können wir die Streuung, die dieses Modell erklärt, untersuchen. Die zugehörige Summe heißt SQE („**S**um of S**q**uares **E**xplained“) und wird aus den Abständen zwischen den vom Modell vorhergesagten Werten $\hat{y}_i$ und dem Mittelwert aller Messwerte $\bar{y}$ gebildet:

$$SQE = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 \tag{50}$$

Als Maß für die Güte des Modells definieren wir das Bestimmtheitsmaß R^2 als den Anteil der Streuung, den das Modell erklärt. Also bewegt sich R^2 zwischen 0 und 1.

$$R^2 = \frac{SQE}{SQT} \tag{51}$$

Für diese Größen gilt die folgende *Streuungszerlegung* (52), deren Gültigkeit nicht ohne weiteres einsichtig ist. Dabei stellt sich heraus, dass die Streuung der Messwerte einfach die Summe der durch das Modell erklärten Streuung und der Residualstreuung (die Summe der Quadrate der Residuen $\hat{d}_i$ SQR („**S**um of S**q**uared **R**esiduals“)) ist:

$$\begin{gathered} SQT = SQE + SQR \\ \sum_{i=1}^{n} (y_i - \bar{y})^2 = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \end{gathered} \tag{52}$$

Daraus folgt eine praktischere Formel für die Berechnung von R^2:

$$R^2 = \frac{SQE}{SQT} = \frac{SQT - SQR}{SQT} = 1 - \frac{SQR}{SQT} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Liegen alle beobachteten Punkte exakt auf einer Geraden, so sind die Residuen alle gleich Null und ebenso die Residualstreuung. In diesem Fall ist also die Gesamtstreuung gleich der erklärten Streuung, d.h. die gesamte Variation von Y lässt sich durch die Variation von X zusammen mit der postulierten linearen Beziehung erklären. In diesem Fall ist R^2 also 1. Je grösser nun die Residualstreuung ist, desto schlechter beschreibt das Modell die Daten, d.h. desto weniger wird die in den Daten vorhandene Streuung durch das Modell erklärt.

Haben die Messwerte unterschiedliche Fehler, treten an die Stelle der Mittelwerte gewichtete Mittelwerte. Auch die einzelnen Summanden müssen gewichtet werden.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}\frac{(y_i - \hat{y_i})^2}{\sigma_i^2}}{\sum_{i=1}^{n}\frac{(y_i - \bar{y})^2}{\sigma_i^2}} = 1 - \frac{\sum_{i=1}^{n}\frac{(y_i - \hat{a} - \hat{b}x_i)^2}{\sigma_i^2}}{\sum_{i=1}^{n}\frac{(y_i - \bar{y})^2}{\sigma_i^2}} \text{ mit } \bar{y} = \frac{\sum_{i=1}^{n}\frac{y_i}{\sigma_i^2}}{\sum_{i=1}^{n}\frac{1}{\sigma_i^2}} \qquad (53)$$

[Fahr, S. 158ff] Grenzen innerhalb derer ein Wert für R^2 akzeptabel ist, lassen sich nicht so ohne weiteres angeben. Man könnte jedoch eine Grenze angeben, so dass mit einer vorgegebenen Wahrscheinlichkeit R^2 kleiner als diese Grenze ist, falls es sich um einen linearen Zusammenhang handelt. Diese Grenze ist natürlich von der Zahl n der Messwerte abhängig. Das heisst *nicht*, dass wenn R^2 kleiner als diese Grenze ist, der wahre Zusammenhang mit der angegebenen Wahrscheinlichkeit linear ist!

6.4 *Fehler in den x-Werten

Sind nur die x-Werte fehlerbehaftet, bzw. die y-Fehler gegenüber den x-Fehlern vernachlässigbar, vertauschen Sie einfach abhängige und unabhängige Variable.

Die Umrechnung der so erhaltenen Parameter ist unproblematisch. Auch der Fehler der Steigung kann problemlos nach dem Fehlerfortpflanzungsgesetz berechnet werden. Bei der Umrechnung des Achsenabschnitts tritt das selbe Problem wie in **6.6** auf, da die Fehler von Achsabschnitt und Steigung nicht unabhängig sind. Glücklicherweise wird im Praktikum häufig nur die Steigung benötigt.

6.5 *x- und y-Werte fehlerbehaftet

Diese Aufgabe lässt sich nicht ganz so einfach lösen. Das Prinzip der kleinsten Quadrate kann hier nicht so ohne weiteres angewendet werden, denn es verrät nicht,

welche quadratischen Abstände minimiert werden sollen. Der kürzeste Abstand r_i vom Messpunkt zur Ausgleichsgeraden ist es im Allgemeinen jedenfalls nicht (Abb. 5), denn dann wäre die Lage der Geraden von der Achsenskalierung abhängig. Glücklicherweise hilft auch hier das ML-Prinzip weiter. [Bar, S. 109]

6.6 *Interpolation und Kalibrierkurven

Gelegentlich führt man zunächst eine Messreihe durch, aus der man eine Ausgleichsgerade bestimmt, um dann später die Grösse y zu messen indem man x misst und y mit Hilfe dieser *Kalibrierkurve* ausrechnet – oder umgekehrt. Das funktioniert indem man einfach den Messwert x in die Geradengleichung der Kalibrierkurve einsetzt. Leider ist die Berechnung des Fehlers des so geschätzten y-Wertes nicht ganz so unproblematisch, weil Steigung und Achsenabschnitt nicht statistisch unabhängig sind. Wenn man einfach das Gaußsche Fehlerfortpflanzungsgesetz anwendet, um den Fehler dieses Schätzwertes zu bestimmen, erhält man deshalb einen zu kleinen Wert. Mit dem Größtfehler liegt man zwar auf der sicheren Seite, aber er ist wie der Name schon sagt unangemessen groß. Auch hier lässt uns die Mathematik nicht im Stich. Durch eine einfache Koordinatentransformation ist es möglich, eine neue Geradengleichung mit statistisch unabhängigen Parametern zu erhalten. [Bar, S. 103]

6.7 *Ausblick: Matrixdarstellung und nichtlineare Modelle

Wir haben gesehen, dass die Normalgleichungen ein lineares Gleichungssystem darstellen. Man kann dieses Gleichungssystem in der Sprache der linearen Algebra als Matrixgleichung schreiben. Die übrigen Rechnungen werden dann viel kürzer.([Bar] S. 103) Insbesondere wenn kompliziertere lineare Modelle als eine einfache Gerade verwendet werden, ist diese Darstellung vorzuziehen, auch weil sich solche Gleichungssysteme besonders gut programmieren lassen.

Auch auf nicht lineare Modelle kann das ML-Prinzip bzw. die Methode der kleinsten Quadrate angewendet werden. In diesem Fall sind die Normalgleichungen leider nicht linear und man ist auf numerische Lösungen angewiesen. Eine Anwendung eines solchen nichtlinearen Modells auf die simultane Bestimmung der Halbwertzeiten von ^{108}Ag und ^{110}Ag, die Teil des Versuchs *radioaktiver Zerfall* ist, findet sich in [Bev].

7 Hinweise zu Software

Warum müssen Sie in der Praktikumsaufgabe selbst eine Funktion zu Linearen Regression schreiben, wenn es schon Programme gibt, die dies beherrschen?

Vielleicht bietet selbst Ihr Taschenrechner diese Funktion an.

Leider berechnen die meisten Programme, selbst wenn sie die Eingabe von Fehlern für die Ermittlung der Gewichte gestatten, nur Standardfehler aus der Streuung. Dies ist bei der Funktion `fit` aus *gnuplot* oder `Regress` mit der Option `Weights` aus *Mathematica*, sowie den entsprechenden Funktionen aus *Excel* der Fall.

Wenn Sie diese Funktionen verwenden möchten, müssen Sie bei *gnuplot* die Faktoren f_i in der dritten Spalte angeben. Bei `Regress` aus *Mathematica* hingegen sind die Gewichte $w_i = 1/f_i^2$ gefragt.

Bei den meisten Praktikumsversuchen ist die Zahl der Messwerte so klein, dass die Fehlerrechnung aus der Streuung nach 5.2.2 nicht sinnvoll ist. Zu dem enthalten die Eingangsfehler oft auch systematische Anteile, die im Rahmen des Praktikums nicht gesondert berücksichtigt werden, aber doch zumindest teilweise in die Schätzung des Fehlers eingehen sollten.

Bei unkritischer Anwendung dieser Funktionen in dem Glauben, sie würden Eingangsfehler nach 5.2.1 berücksichtigen, bleibt der Fehler oft unbemerkt, da die Parameter $\hat{a}$ und $\hat{b}$ ja in beiden Fällen identisch sind. Lediglich die Schätzungen für $\Delta\hat{a}$ und $\Delta\hat{b}$ sind oft unrealistisch und tendenziell zu klein.

Wichtig ist, nicht zu vergessen, dass $\frac{\chi^2}{n-2}$ nur dann die in 6.1 beschriebene diagnostische Bedeutung hat, wenn Daten mit Fehlern angegeben wurden. (Rechnung nach 5.2.1). Wurden die Fehler hingegen nach 5.2.2 aus der Streuung geschätzt, hat $\frac{\chi^2}{n-2}$ nicht diese diagnostische Bedeutung!

Umrechnung von Fehlern aus der Streuung in apriori-Fehler

Sie können die oben erwähnten Programme dennoch verwenden, um Schätzungen der Fehler nach (39a) Δa und Δb aus den apriori-Fehlern σ_i der Messwerte y_i zu bestimmen, indem Sie $f_i = \sigma_i$ verwenden und die von den Programmen aus der Streuung nach (48) berechneten Fehler $\tilde{\Delta}a$ und $\tilde{\Delta}b$ wie folgt umrechnen.

$$\Delta a = \sqrt{\frac{\tilde{\Delta}a^2}{\frac{\chi^2}{n-2}}} \qquad \text{und} \qquad \Delta b = \sqrt{\frac{\tilde{\Delta}b^2}{\frac{\chi^2}{n-2}}} \tag{54}$$

Dabei ist $\frac{\chi^2}{n-2}$ der von den Programmen nach (47) berechnete Wert. Sie können diese Beziehung verwenden, um ihre Lösung an der von *Mathematica* mit der Funktion `Regress` berechneten zu prüfen.

8 Aufgaben

0. Machen Sie sich etwas mit *Mathematica* vertraut, in dem Sie ein wenig mit den in der *Einführung in Mathematica* beschriebenen Funktionen spielen. (Diese Aufgabe brauchen Sie nicht abzugeben!)

1. (a) Schreiben Sie die in der *Einführung in Mathematica* beschriebene Funktion zur linearen Regression ergänzt um
   - die Ausgabe der Zahl der Messwerte
   - die Berechnungen und Ausgabe des Bestimmtheitsmaßes R^2
   - die Berechnungen und Ausgabe der Größe $\frac{\chi^2}{n-2}$

   (b) Wenden Sie diese Funktion, wie in der *Einführung in Mathematica* gezeigt, auf die Beispieldatensätze `linear.txt` und `exp.txt` an und stellen Sie das Ergebnis mit Ausgleichs- und Grenzgeraden graphisch dar.

2. Wenden Sie die Funktion aus Aufgabe 1a *ohne zu logarithmieren* (!) auf den Datensatz `exp.txt` an und und stellen Sie das Ergebnis mit Ausgleichs- und Grenzgeraden grafisch dar.

   Dies ist ein Beispiel für eine fehlerhafte Anwendung! Betrachten Sie die Lage der Grenzgraden. Entspricht das Ergebnis ihrer Erwartung? Wo hätten Sie die Grenzgeraden gezeichnet? Wie erklären Sie das Ergebnis? Vielleicht bringen Sie das Bestimmtheitsmaß R^2 (6.3) oder $\frac{\chi^2}{n-2}$ (6.1) auf die richtige Spur. **Vergleichen** Sie dazu die Werte für diese beiden Parameter mit der korrekten Anwendung in Aufgabe 1a. (Hier müssen Sie ein wenig Text schreiben!)

3. (a) Vervierfachen Sie die Fehler in `linear.txt`. Erzeugen Sie dazu mit den Funktionen `Column` und `Transpose` eine neue Matrix `linear4` in der Sie die dritte Spalte aus `linear.txt` mit 4 multipliziert haben. Alternativ können sie auch die Funktion `Replace` zum Erzeugen der neuen Matrix verwenden.

   (b) Schreiben Sie analog zu Aufgabe 1a eine Funktion, die Fehler nicht aus den Eingangsfehlern, sondern aus der gewichteten Streuung nach **5.2.2** berechnet.

   (c) Wenden Sie diese auf auf die Datensätze `linear.txt` und `linear4` an. Wenden Sie die Funktion aus Aufgabe 1a auf beide Datensätze an und stellen Sie die Ergebnisse mit Ausgleichs- und Grenzgeraden grafisch dar.

(d) Vergleichen Sie die vier Ergebnisse (Grenzgeraden, Fehler der Parameter, $\frac{\chi^2}{n-2}$ und R^2). Was ist der Unterschied zwischen den beiden Verfahren? (Hier müssen Sie ein wenig Text schreiben!)

(e) **Zusatzaufgabe:**

- Vergleichen Sie die Ergebnisse mit dem durch Anwendung der Funktion `Regress` aus dem Statistikpaket von *Mathematica* mit der Option `Weights` auf `linear.txt` erhaltenen.
- Vergleichen Sie das Ergebnis mit Hilfe von (54) mit dem von Aufgabe 1.

4. (a) Leiten Sie die Gleichungen für Parameter und Fehler so wie das reduzierte χ^2 für das Problem einer **Ursprungs**gerade analog zur Darstellung in 5 her. (Geht am besten mit Papier und Bleistift!)

   (b) Schreiben Sie die Funktion aus Aufgabe 1a für diesen Fall und testen Sie sie mit dem Datensatz `linear.txt`. (Die Funktion `Regress` aus Mathematica verwendet in diesem Fall eine abweichende Definition für R^2, lassen Sie sich also nicht verunsichern, falls Sie auch das Ergebnis dieser Aufgabe mit dem der Funktion `Regress` vergleichen!)

5. **Zusatzaufgabe** (alternativ zu 1-4) Lösen Sie die vorstehenden Aufgaben nicht wie hier beschrieben, sondern verwenden Sie die Matrixdarstellung nach [Bar, S. 111ff] und die Funktionen von *Mathematica* zur linearen Algebra.

Hinweise

- Wenn Sie noch wenig Computererfahrung haben, sollten Sie die Aufgaben an den Computern des Fachbereichs bearbeiten.
- Falls Sie die Aufgaben zu Hause bearbeiten, sorgen Sie bitte dafür, dass Sie stets eine aktuelle Version in der Uni verfügbar haben, damit sie kleine Korrekturen sofort ausführen können.
- Erzeugen Sie *ein* Notebook, das die Lösung aller Aufgaben enthält, da Sie in einigen Aufgaben auf Ergebnisse der 1. Aufgabe zurückgreifen müssen.
- Verwenden Sie die Funktionen `Copy, Paste` und `Find` und `Replace` ausgiebig. Kleinere Formelteile oder Sonderzeichen können auch gut mit der mittleren Maustaste kopiert werden.

- Schicken Sie Ihre Lösung als e-Mail-Attachment an Ihren Betreuer, und benennen Sie es dabei wie folgt:

  cp*nachname*_*vo*_*xx*.nb

  Dabei steht *nachname* für Ihren Nachnahmen, *vo* für die ersten beiden Buchstaben ihres Vornamens. Mit *xx* wird die Versionsnummer bezeichnet: Bei der ersten Abgabe verwenden Sie `01`, bei der zweiten Abgabe (also der ersten Berichtigung) dann `02` und so weiter. **Bei Berichtigungen geben sie bitte den vom Tutor erhaltenen korrigierten Ausdruck mit ab!** Bitte schreiben Sie zusätzlich Ihren Namen in einen Kommentar am Anfang des Notebooks. Die Lösung von Aufgabe 4 (a) können sie natürlich auch auf Papier abgeben.
- Beachten Sie, dass Mathematica keine Variablennamen mit Quadraten wie `R^2` oder `s^2` akzeptiert.
- **Bewertung** Um 5 Punkte zu erlangen, ist die Bearbeitung der Zusatzaufgaben erforderlich. 4 Punkte können Sie jedoch auch ohne Zusatzaufgaben erreichen.

Literatur

[Bar] Barlow, Roger J. *Statistics: a guide to the use of statistical methods in the physical sciences*. Wiley, 1989; Nachdr. 1999.

[Bev] Bevington, Philip R.; Robinson, D. Keith. *Data reduction and error analysis for the physical sciences*. McGraw-Hill, 2. Aufl. 1992.

[Fahr] Fahrmeir, Ludwig [u.a.] *Statistik – der Weg zur Datenanalyse*. Springer, 1997.

[Mand] Mandel, John. *The statistical analysis of experimental Data*. New York: Interscience, 1964; korr. Nachdr. New York: Dover, 1984.

[Squi] Squires, G. L.. *Messergebnisse und ihre Auswertung: Eine Anleitung zum praktischen naturwissenschaftlichen Arbeiten*. Walter de Gruyter, 1971.

Als Einführung in die Fehlerrechnung und Datenanalyse für Physiker sind vor allem [Bar] und [Bev] gut geeignet. Am Ende einiger Abschnitte in diesem Skript finden Sie Hinweise, wo das jeweilige Thema ausführlicher dargestellt wird.

Computerpraktikum im GP II
Einführung in *Mathematica*

Daniel Brete Michael Karcher Jens Koesling

Was ist *Mathematica*

Mathematica ist ein Computeralgebrasytem, d. h., dass *Mathematica* z.B. Integrale symbolisch lösen kann. Dies ist vermutlich die meist genutzte Funktion von *Mathematica* in den ersten Semestern. Für den Praktikumsversuch nutzen wir nicht diese Eigenschaft, sondern die Möglichkeit relativ einfache Rechnungen, die man auch mit Papier und Taschenrechner durchführen könnte zu automatisieren. Diese Aufgabe lässt sich mit praktisch jeder Programmiersprache bearbeiten.

Wir haben uns für *Mathematica* entschieden, weil Sie diese Software im weiteren Verlauf Ihres Studiums noch häufiger verwenden werden. Zum Beispiel für die Bearbeitung von Übungsaufgaben zu den Theorievorlesungen. Einige Dozenten stellen auch bereits in den Vorlesungen *Physik I* oder *II* Aufgaben, die mit *Mathematica* gelöst werden sollen. In diesem Fall haben sie schon erste Erfahrungen gesammelt. Vorteile bei unseren kleinen Rechenaufgaben im Praktikum gegenüber Tabellenkalkulationen sind, dass man nicht an ein starres Tabellenraster gebunden ist, Formeln und Ergebnisse gleichzeitig sichtbar sind und die Schreibweise von Funktionen der gewohnten Darstellung näher kommt.

Mathematica kann weit mehr als wir hier ausnutzen. Neben den verblüffenden Funktionen zum symbolischen Rechnen gibt es Funktionen für numerische Berechnungen, die Konstruktion komplizierter Grafiken, das Einbinden externer C−Programme und für die statistische Datenanalyse. Ausserdem gibt es einige Textverarbeitungsfunktionen und Möglichkeiten zum Formelsatz, mit denen dieses Heft erstellt wurde.

Sie können mit *Mathematica* ein komplettes Praktikumsprotokoll von den physikalischen Grundlagen bis zur Fehlerrechnung erstellen. Zumindest aber sollten Sie nach dieser Einführung in der Lage sein, grafische Auswertungen statt mit Bleistift und Millimeterpapier mit *Mathematica* auszufühen

Hinweise zur Bedienung

Starten

Geben Sie in einem Terminalfenster `mathematica` ein. Geschickt ist es, vor dem Starten in das Verzeichnis zu wechseln, in dem sich die zu bearbeitenden Dateien befinden oder erstellt werden sollen. Sie ersparen sich so das wiederholte Auswählen von Verzeichnissen in einer etwas unübersichtlichen Dialogbox. Sie können auch beim Programmaufruf eine Datei mit angeben, z.B.: »`mathematica meinedatei.nb`«. *Mathematica*−Dateien heissen Notebooks und haben die Endung »`.nb`«.

Dateien öffnen, schliessen, Programm beenden

Diese Funktionen erreichen Sie über das Menü **File**, dass sich mit der Maus in gewohnter Weise bedienen lässt. Das Beenden des Programms gelingt nur mit dem Befehl **Quit** aus diesem Menü. Ein Doppelklick auf das Fenster−schliessen−Symbolin der Titelleiste des Programms schliesst zwar das aktuelle Fenster, öffnet aber ein neues, falls es sich um das einzige *Mathematica*−Fensterhandelt.

Einige Unzulänglichkeiten

In *Mathematica* funktionieren leider viele Tasten nicht so, wie man es gewohnt ist. Auf vielen Systemen löscht DEL nicht vorwärts sondern rückwärts. **Der numerische Zehnerblock kann nicht benutzt werden.** Bei gedrückter Nummlocktaste funktioniert kaum noch etwas. Glücklicherweise ist der Spuck beendet, wenn Sie sie wieder ausschalten.

Drucken

Drucken können Sie über die Druckfunktion im Menu **File**. Achten Sie unbedingt darauf, dass sie die Box **Include Mathematica Fonts** einschalten; die Sonderzeichen werden sonst nicht richtig gedruckt. Auch die Wahl des richtigen Papierformats (A4) ist entscheidend. Um auf dem Drucker im Rechenerraum auszudrucken, geben Sie im Feld **Print to** »`lpr -Ptal`« ein. Oder erzeugen Sie eine Postscriptdatei, in dem Sie den Punkt **File** markieren und einen Dateinamen eingeben. Vorsicht, Sie können gleichzeitig sowohl in eine Datei, als auch auf den Drucker drucken. Achten Sie also darauf, dass nur die gewünschte Option markiert ist. Gegenwärtig sind die von *Mathematica* erzeugten Postscript−Dateien nicht ganz DSC−Standard konform. Dies kann dazu führen, dass die Sonderzeichen nicht richtig wiedergegeben werden, wenn Sie versuchen die Dateien mit den pstools (psbook, psnup usw.) nachzubearbeiten.

■ Hilfe

Die Hilfe rufen Sie über den Menupunkt **Help → Help Browser ...** auf.
Den gesuchten Begriff oder Teile davon (z.B. `ParametricPlot3D` oder auch nur `Paramet`) im Feld **Go To:** eingeben und mit ↵ bestätigen.

Der Help Browser besitzt mehrere Ebenen: Bestimmte Texte oder Erklärungen stehen in verschiedenen Ebenen. Mathematica sucht in diesen Ebenen immer nur abwärts. So kommt es, dass man manchmal bestimmte Stichwörter nicht findet. In diesem Fall muss man mittels **Master Index** auf die oberste Ebene zurück gehen.

Meist ruft man aus einem von zwei Gründen die Hilfe auf. Entweder sucht man eine Befehlsbeschreibung, man weiß also schon (oder vermutet es zumindest), welcher Befehl das leistet, was man erreichen möchte; oder man will etwas bestimmtes erreichen, weiss aber nicht mit welchem Befehl.
Im ersten Fall sucht man einfach den Befehlsnamen. Die Befehlserklärungen sind meist sehr illustrativ.
Im zweiten Fall sollte man nach Begriffen suchen, die in Texten zu eben diesem Thema vorkommen könnten. Da die Hilfe auch das *Mathematica*−Buch enthält, kann einem auch hier meist geholfen werden.

■ Eingeben von Befehlen

Klicken Sie auf die weisse Arbeitsfläche oder erzeugen Sie ein neues leeres Notbook mit **File→New** und beginnen Sie zu schreiben. Eine Zelle (Eingabe) in Mathematica kann mehrere Zeilen umfassen. Der Zeilenwechsel erfolgt einfach mit ENTER. Nach der letzten Anweisung drücken Sie SHIFT ENTER, um alle Befehle in einer Zelle auf einmal auszuführen. Dieses SHIFT ENTER ruft den Kernel, den mathematischen Kern Mathematicas, auf und lässt diesen die eingegebene Zelle interpretieren und ausrechnen. Dieser Vorgang heisst Evaluieren (Evaluation). Im folgenden sind die fettgedruckten Zeilen die **`Eingabe,`** und die dünn gedruckten Zeilen die zugehörige `Ausgabe` von *Mathematica*. Probieren Sie das folgende Beispiel aus. In *Mathematica* können Sie für die Multiplikation anstelle des Sterns »*« auch ein Leerzeichen verwenden.

```
2 5
```

```
10
```

Möchten Sie zwischen zwei Zellen (Durch eckige Klammern am rechten Rand markiert.) eine neue Zelle einfügen, klicken Sie zwischen die Zellen.

Es erscheint eine horizontale Linie. Wenn Sie zu schreiben beginnen, öffnet sich die neue Zelle.

Man kann auch das ganze Notebook auf einmal auswerten. Hierzu muss man in der Befehlsleiste **Kernel→Evaluation→Evaluate Notebook** auswählen.

Mathematica speichert alle Evaluierungen (mit allen Daten, Variablen und Funktionen), die es während einer Sitzung durchführt. So kommt es, dass *Mathematica* Funktionen, die man schon vor langem gelöscht hat, immer noch kennt und deren Funktionsnamen deshalb nicht freigibt.
In einer solchen Situation hilft es, den Kernel zu beenden und dann das Notebook erneut insgesamt zu evaluieren. Dann kann man dort, wo man ausgesetzt hatte, weiter arbeiten. Den Kernel beendet man über **Kernel→-Quit Kernel**.

■ Grundlagen der Syntax von Mathematica

Sämtliche in *Mathematica* vordefinierten Funktionen beginnen mit einem Großbuchstaben. Die Parameterlisten für Funktionen werden in **eckigen** Klammern angegeben. Die runden Klammern dienen nur zum Klammern von mathematischen Ausdrücken. Daher heisst es `3 (2 + 4)` aber `Sin[2 Pi]`, da nur im ersten Fall die Klammern die Funktion der Gruppierung von Termen haben, im zweiten Fall aber die Klammern angeben, dass Argumente an eine Funktion übergeben werden. Da alle vordefinierten Namen (die von Funktionen wie `Sin`, `Cos` oder `Sqrt` oder von Variablen wie `Pi`) mit einem Großbuchstaben beginnen, wird empfohlen, alle selbst definierten Namen mit einem Kleinbuchstaben anfangen zu lassen.

−Ein einfacher Ausdruck: `4/(2+3)`
−Ein Ausdruck, in dem eine Funktion benutzt wird: `3*Sin[2*(2+x)]`

■ Kommentare

Will man, dass Mathematica eine Zelle nicht auswertet, da sie nur Fließtext oder einen Kommentar enthält, so muß Mathematica gesagt werden, dass es sich nur um eine Text−Zellehandelt. Dazu muss die Zelle am rechten Rand markiert werden, dann wählt man aus dem Menu **Format→Style→Text.** Markiert man hingegen nur den Inhalt der Zelle oder Teile davon, bewirken die verschiedenen Styles nur unterschiedliche Formatierungen, ohne die Art der Auswertung der Zelle zu verändern.

Es gibt eine große Menge weiterer Style− und allgemeiner Einstellungen für Zellen, die sich unter **Format→Style**, bzw. **Format→...** befinden.

■ Ausdrücke

Ausdrücke, die *Mathematica* zur Bearbeitung gegeben werden, werden so weit wie möglich ausgewertet:

```
3 (2 + 4)
```

```
18
```

```
3 (2 + 5 a + 3)
```

```
3 (5 + 5 a)
```

Wie man sieht, werden die Zahlen so weit wie möglich zusammengefasst, aber sobald Variablen auftreten, arbeitet *Mathematica* sehr vorsichtig. Um *Mathematica* zu sagen, dass ein Ausdruck vereinfacht werden soll, verwendet man die Funktion `Simplify`:

```
Simplify[3 (2 + 5 a + 3)]
```

```
15 (1 + a)
```

Hier hat *Mathematica* jetzt von selbst den Faktor 5 ausgeklammert.

Die Funktion **`FullSimplify`** tut dasselbe, versucht jedoch bei komplizierten Ausdrücken noch mehr Umformungen und liefert so manchmal ein besseres Ergebniss.
Will man einen Ausdruck ausmultiplizieren, muss man `Expand` verwenden:

```
Expand[3 (2 + 5 a + 3)]
```

```
15 + 15 a
```

Möchte man dagegen aus einem ausmultiplizierten Term wieder ein Produkt gewinnen, kann man `Factor` benutzen, obwohl in den meisten Fällen `Simplify` auch schon ausreicht.

```
Factor[x^3 + 3 x^2 y + 3 x y^2 + y^3]
```

$$(x + y)^3$$

■ Variablen

Variablen sind vom Prinzip her sehr einfach. Man kann mit dem Gleichheitszeichen Variablennamen, die, wie schon oben empfohlen, mit einem Kleinbuchstaben beginnen sollten, einen Wert zuweisen, ohne dass eine vorherige Deklaration erforderlich ist. Sobald der Name später wieder auftaucht, setzt *Mathematica* dort den zugewiesenen Wert ein. Mit der folgenden Zeile wird `n` auf `3` gesetzt:

```
n = 3
```

```
3
```

Jetzt kann man `n` wie eine Zahl verwenden, und wenn man `n` danach einen neuen Wert zuweist, gilt ab dann dieser Wert:

```
n * 2
```

```
6
```

```
2^n
```

```
8
```

```
n = 4
```

```
4
```

```
n * 2
```

```
8
```

Wie man sieht, kommt nach der Zuweisung `n=4` bei dem gleichen Ausdruck `2*n` ein anderer Wert heraus, da sich `n` selbst geändert hat. Diese Funktion ist manchmal nützlich, sollte aber vorsichtig benutzt werden, denn es ist in *Mathematica* vorgesehen, dass man jederzeit jede Formel ändern und neuberechnen kann. Das geht natürlich nur solange gut, wie sich die Variablen seit der ersten Berechnung nicht geändert haben. Daher lautet eine Empfehlung: Variablen sollten nur einmal gesetzt werden, und zwar natürlich vor jeder Anwendung dieser Variablen.

■ Unterdrücken der Ausgabe

Häufig möchte man das Ergebnis einer Berechnung nicht auf dem Bildschirm ausgeben. Um dies zu erreichen, kann man die Anweisung deren Ausgabe unterdrückt werden soll mit einem Semikolon abschliessen. Wir machen davon in diesem Text gelegentlich Gebrauch um Platz zu sparen. Wenn Sie die Beispiele nachvollziehen wollen, können sie die Semikolons weglassen, um die Ausgabe zu sehen.

```
5 + 3;
```

ist ohne Ausgabe sinnlos. Aber wenn Sie einer Variablen einen Wert zuweisen, kennen Sie die Ausgabe schon:

```
h = 7;
```

Ausserdem können mit dem Semikolon mehrere Anweisungen in einer Zeile eingegeben werden.

```
h = 7; l = 9;
```

■ **Sonderzeichen**

Will man nun griechische Buchstaben in einem Ausdruck, Umlaute in einem Kommentar oder mathematische Symbole wie ∈ oder ∀ in einem Text verwenden, muss man diese Zeichen gesondert einfügen, wenn sie sich nicht auf der Tastatur befinden. Dies kann man über Paletten, die man nach Belieben im Menu unter **File→Palette →...** aufrufen kann. Für viele dieser Zeichen existiert aber auch ein Tastatur Shortcut. Diese Shortcuts werden in der Form esc Shortcut esc eingegeben.

So sind, z.B.
esc D esc = Δ
esc d esc = δ
esc [[esc = ⟦
esc a " esc = ä

■ **Listen −Vektoren und Matrizen**

Die grundlegende Datenstruktur in Mathematica ist die Liste. Eine Liste ist eine Aufzählung oder eine Menge von Elementen. Hierbei kommt es, im Gegensatz zu mathematischen Mengen, jedoch auch auf die Reihenfolge der Elemente an. Die Elemente einer Liste können so gut wie alles sein, was Mathematica kennt. Also ist auch eine Liste einer Zahl, einer Funktion und einer weiteren Liste eine Liste.

Um Elemente zu einer Liste zusammenzufassen, gibt es den Befehl `List`.

```
List[1, a, ArcTan, {c, d}, "String"]
```

```
{1, a, ArcTan, {c, d}, String}
```

Abkürzend kann man aber auch einfach die Elemente in ein Paar gemeinsamer geschweifter Klammern setzen.

```
{1, a, ArcTan, {c, d}, "String"}
```

```
{1, a, ArcTan, {c, d}, String}
```


Die meisten naiven Funktionen in Mathematica sind dergestalt, dass sie auf eine Liste angewendet auf jedes Listenelement einzeln wirken. Dies ist eine Kurzschreibweise für die Funktion `Map`.

```
Sin[{0, Pi / 2, Pi}]
```

```
{0, 1, 0}
```

kurz für:

```
Map[Sin, {0, Pi / 2, Pi}]
```

```
{0, 1, 0}
```

Listen sind eine Möglichkeit, Vektoren und Matrizen darzustellen. Vektoren fasst man als Listen von Zahlen (`Real`) auf und Matrizen als Listen von Vektoren. Das Arbeiten mit Vektoren und Matrizen ist in *Mathematica* recht bequem möglich, da die einfachen Rechenoperatoren Vektoren und Matrizen elementweise verknüpfen:

```
{2, 3, 4} + {1, 5, 8}
```

```
{3, 8, 12}
```

Ist nur einer von zwei Operanden ein Vektor, der andere dagegen ein einfacher Ausdruck, so wird der Ausdruck mit jedem einzelnen Element des Vektors verknüpft. Das mag bei der Addition zwar reichlich sinnlos erscheinen, ist aber das Verhalten, das man bei einer Multiplikation erwartet:

```
{2, 3, 4} + 1
```

```
{3, 4, 5}
```

```
{2, 3, 4} * 2
```

```
{4, 6, 8}
```

Man kann natürlich auch zwei Vektoren miteinander multiplizieren, aber das führt wie bei der Addition, bei der eine Zahl zu einem Vektor addiert wird, nicht zu mathematisch sinnvollen Operationen:

```
{2, 3, 4} * {1, 1, 2}
```

```
{2, 3, 8}
```

Zum Bestimmen des Skalarproduktes verwendet man statt des Sternes einen Punkt. Dieser steht im allgemeinen für Matrixprodukt. Das Skalarprodukt zweier Vektoren ist auch nichts weiter als das Matrixprodukt eines Zeilenvektors mit einem Spaltenvektor. Da

Mathematica zwischen Zeilen− und Spaltenvektoren sowieso nicht unterscheidet, ist das Skalarprodukt tatsächlich ein Matrixprodukt

```
{2, 3, 4}.{1, 1, 1}
```

```
9
```

Eine Matrix wird als Vektor von Vektoren gleicher Länge eingegeben:

```
{{1, 2}, {3, 4}}
```

```
{{1, 2}, {3, 4}}
```

Die Ausgabe erscheint normalerweise schöner, wenn man die Matrix in der sogenannten `MatrixForm` darstellt, indem man die Funktion `MatrixForm` auf die Matrix anwendet. *Mathematica* macht die Ausgabe in der Spezialform dadurch kenntlich, dass an die Ausgabennummer `//MatrixForm` angehängt wird. Das kommt von einer ganz allgemeinen Methode, Funktionen zu verketten, die erst etwas später erklärt wird:

```
MatrixForm[{{1, 2}, {2, 3}}]
```

$$\begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$$

```
MatrixForm[{{1, 2}, {2, 3}} + {{1, 0}, {0, 1}}]
```

$$\begin{pmatrix} 2 & 2 \\ 2 & 4 \end{pmatrix}$$

```
MatrixForm[{{1, 2}, {2, 3}} * {{1, 0}, {0, 1}}]
```

$$\begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix}$$

```
MatrixForm[{{1, 2}, {2, 3}}.{{1, 0}, {0, 1}}]
```

$$\begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$$

Man beachte vor allem den Unterschied zwischen der elementweisen Multiplikation mit dem Stern und der Matrixmultiplikation mit dem Punkt.

- **Zugreifen auf Elemente einer Liste:**

Wir definieren zu Demonstrationszwecken einen Vektor `u` und eine Matrix `v`.

```
u = {6, 9}
```

```
{6, 9}
```



```
MatrixForm[v = {{1, 2}, {3, 4}}]
```

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

Häufig kommt es vor, dass man auf ein ganz bestimmtes Element eines Vektors, einer Matrix oder allgemein einer Liste zugreifen will. Diesen Zugriff ermöglicht die Funktion `Part`. Ihre Kurzformen sind viel einfacher. Alle der folgenden Formen sind Kurzschreibweisen für den mit `Part` realisierten Zugriff auf die Elemente einer Liste.

Zugreifen auf das erste Element des Vektors `u` geschieht so:

```
Part[u, 1]
u[[1]]
```

```
6
```

```
6
```

und der Zugriff auf das Element in der ersten Zeile und zweiten Spalte der Matrix v so:

```
Part[Part[v, 1], 2];
Part[v, 1, 2];
v[[1]][[2]];
v⟦1, 2⟧;
v[[1, 2]];
v⟦1, 2⟧;
v⟦1,2⟧
```

```
2
```

- **Die Listeneingabehilfe**

Die Eingabe längerer Listen z.B. Messwertetabellen mit geschweiften Klammern ist umständlich und unübersichtlich. Wenn sie solche langen Tabellen nicht aus einer Datei einlesen möchten (siehe später), können Sie die Listeneingabehilfe verwenden. Wählen Sie hierzu aus dem Menu **Input→Create Table/Matrix/Palette**. In dem sich öffnenden Dialogfeld wählen Sie, ob Sie die Liste als Matrix oder Tabelle formatieren möchten und die Zahl der Zeilen und Spalten. In der so eingegügten Vorlage können Sie mit TAB zum nächsten freien Feld springen.

■ Funktionen

Funktionen werden genauso wie Variablen mit dem einfachen Gleichheitszeichen definiert. Dabei steht auf der linken Seite des Gleichheitszeichens ein sogenanntes Muster. Über Muster kann man sehr viel sagen, was aber den Rahmen dieses Skripts bei weitem sprengt. Daher fassen wir hier kurz zusammen, was von vornehmlicher Bedeutung ist: Schreibt man einfach `f[x]=x^2,` scheint es auf den ersten Blick auch zu funktionieren:

```
f[x] = x^2
```

x^2

```
x + f[x]
```

$x + x^2$

Aber sobald man versucht, ein anderes Argument als `x` zu verwenden, tut *Mathematica* plötzlich so, als ob es keine Ahnung hätte, wie `f` definiert ist:

```
y + f[y]
```

```
y + f[y]
```

Hier wurde die Funktion nicht eingesetzt. Man kann auch noch `Simplify` darauf anwenden: Es klappt einfach nicht. Der Fehler liegt in der Definition der Funktion. Wie schon erwähnt, steht auf der linken Seite des Gleichheitszeichens ein Muster. In diesem Fall ist es ein sehr einfaches Muster, nämlich die Funktion `f` mit dem Argument `x`. Daher wird »Die Funktion `f` mit dem Argument `x`« auch durch die rechte Seite ersetzt. Allerdings nicht »Die Funktion `f` mit dem Argument `y`«, da diese Eingabe nicht auf das Muster passt.
Was man benötigt, ist ein Muster, das allgemeiner gefasst ist. Dazu verwendet man den Unterstrich, häufig auch »Blank« genannt, da er für einen beliebigen Ausdruck steht, wie eine Blankostelle, die noch ausgefüllt werden muss. Daher der nächste Versuch mit einer konstanten Funktion:

```
constZwei[_] = 2
```

```
2
```

```
constZwei[5] + constZwei[x] + constZwei[x^2]
```

```
6
```

Hier hat *Mathematica* festgestellt, dass das Muster »Die Funktion `constZwei` mit einem beliebigen Argument« in der unteren Zeile dreimal vorkommt und für jedes Vorkommen `2` eingesetzt, so dass sich

das Gesamtergebnis `6` ergeben hat. Das ist aber auch noch nicht ganz das, was man eigentlich braucht, da man zwar jetzt ein schön allgemeines Muster hat, aber leider nicht mehr an das Argument herankommt.
Daher gibt es in *Mathematica* die Möglichkeit, dem Blank einen Namen zu geben, indem man ihn vor den Unterstrich schreibt. Dieser Name kann dann auf der rechten Seite wie eine normale Variable verwendet werden: Die Funktion `quadrat` wird jetzt mit dem Muster »Die Funktion `quadrat` mit einem beliebigen Argument, dass im weiteren `x` genannt wird,« definiert.

```
quadrat[x_] = x^2
```

x^2

```
quadrat[3]
```

```
9
```

Man sollte darauf achten, dass, wenn auf der rechten Seite eine Variable vorkommt, die bereits einen Wert zugewiesen bekommen hat, dieser Wert verwendet wird, egal, ob es vielleicht einen Parameter gibt, der den gleichen Namen hat:

```
n = 15
```

```
15
```

```
negiereFalsch[n_] = -n
```

```
-15
```

```
negiereRichtig[nn_] = -nn
```

```
-nn
```

```
{negiereFalsch[8], negiereRichtig[8]}
```

```
{-15, -8}
```

Die Funktion `negiereFalsch` ist ein Beispiel für dieses Problem. Man sieht es allerdings hier schon in der Antwort von *Mathematica*, denn nicht jede Zahl kann negiert −15 ergeben. Leider bemerkt man bei schwierigen Funktionen dieses Problem nicht immer sofort. Daher die Warnung: Variablen sollten nur mit Bedacht und aussagekräftigeren Namen als `n` benutzt werden.

Funktionen können mit = oder := definiert werden. Üblicherweise wird := verwendet. Dann tritt auch das oben beschriebene Problem nicht auf.

```
negieredochnichtfalsch[n_] := -n
```

```
negieredochnichtfalsch[27]
```

```
-27
```

Bei Verwendung von = wird die rechte Seite erst so weit wie möglich ausgewertet und dann als Funktionsdefinition gespeichert. Verwendet man hingegen := erfolgt die Auswertung erst beim Aufruf der Funktion, nachdem die Parameter eingesetzt wurden. Bei vielen einfachen Funktionen ist das egal. Der Unterschied wird aber bei Konstruktionen wie im folgenden Beispiel, das wir aus der Hilfe entnommen haben, deutlich.

```
ex[x_] := Expand[(1 + x)^2]
```

```
iex[x_] = Expand[(1 + x)^2]
```

```
1 + 2 x + x^2
```

```
ex[y + 2]
iex[y + 2]
```

```
9 + 6 y + y^2
```

```
1 + 2 (2 + y) + (2 + y)^2
```

Es gibt drei Möglichkeiten, eine Funktion auf einen Ausdruck anzuwenden. Bisher haben wir nur die Funktionsnotation kennengelernt. Darüberhinaus gibt es noch die Präfix- und die Postfixnotation. Die Syntax lautet `Funktion@Argument`, bzw. `Argument//Funktion`.

Jeweils das gleiche leisten also

```
Sqrt[4];
Sqrt@4;
4 // Sqrt
```

```
2
```

■ Symbolisch Rechnen

Was *Mathematica* wirklich besser macht als einen Taschenrechner ist nicht nur die Möglichkeit, mit selbstdefinierten Funktionen oder Vektoren zu rechnen, sondern vor allem die Möglichkeit, symbolisch zu rechnen, also die mathematische Terme zu bearbeiten. Zum Beispiel kann *Mathematica* viele Gleichungen exakt lösen, in dem es die Umformungen, die nötig sind, um nach den Variablen aufzulösen selbst vornimmt:

```
Solve[3 * x + x^2 == 10]
```

```
{{x → -5}, {x → 2}}
```


Für den Vergleich zweier Terme verwendet man das doppelte Gleichheitszeichen, das im Gegensatz zum einfachen Gleichheitszeichen keine Zuweisung, sondern einen Vergleich vornimmt. Die Funktion `Solve` dient dazu, die Lösungen einer Gleichung zu suchen. Solange nur eine Variable darin vorkommt, wird automatisch nach dieser Variablen aufgelöst. Dabei darf der Variablen vorher kein Wert zugewiesen worden sein, denn dieser Wert würde eingesetzt, bevor die Gleichung gelöst wird, und damit kann *Mathematica* nur noch feststellen, ob der aktuelle Wert der Variablen die Gleichung löst oder nicht:

```
n = 15
```

```
15
```

```
Solve[2 * n == 30]
```

```
{{}}
```

```
Solve[2 * n == 29]
```

```
{}
```

Die Ausgabe sieht etwas seltsam aus. In dem ersten Fall erhält man eine Liste, die die leere Liste enthält, im zweiten Fall einfach die leere Liste. Man erklärt, dass die einfache leere Liste bei `Solve` stets dann herauskommt, wenn eine Gleichung unlösbar ist, da es dann eben keine Lösung gibt, die in der Liste stehen könnte, und die leere Liste in einer Liste bei allgemeingültigen Gleichungen, da es eine Lösung gibt, bei der aber keine weiteren Aussagen gemacht werden können. Diese Aussage "man muss nichts weiter vorraussetzen" wird durch die innere leere Liste ausgedrückt, und ist in der Liste aller Lösungen das einzige Ergebnis.

Man kann symbolisch nicht nur Gleichungen lösen, sondern auch differenzieren und integrieren:

```
D[2 x + x^2, x]
```

```
2 + 2 x
```

```
Integrate[2 + 2 x, x]
```

```
2 x + x^2
```

Im ersten Fall wurde der Term `2x+x^2` nach `x` differenziert, im zweiten Fall wurde über x integriert. Die Integrationskonstante lässt *Mathematica* weg, man muss sie selbst addieren, wenn man eine Integrationskonstante braucht.

■ Numerisch rechnen

Die Anweisung

```
Sin[Pi / 8]
```

$\text{Sin}\left[\frac{\pi}{8}\right]$

funktioniert nicht so, wie Sie es vermutlich erwartet haben. *Mathematica* wertet Ausdrücke normalerweise nur symbolisch aus. Wenn das nicht funktioniert wird der Ausdruck unverändert zurückgegeben. Nur wenn der Ausdruck Gleitkommazahlen enthält, erfolgt automatisch eine numerische Auswertung. Ganze Zahlen können als Gleitkommazahlen geschrieben werden, in dem man einen Punkt anhängt.

```
{Sin[Pi], Sin[Pi / 8], Sin[0.125 Pi], Sin[1], Sin[1.]}
```

$\left\{0, \text{Sin}\left[\frac{\pi}{8}\right], 0.382683, \text{Sin}[1], 0.841471\right\}$

Mit der Funktion `N` kann man eine numerische Auswertung erzwingen. Sie wird oft in der übersichtlicheren Postfixnotation verwendet.

```
N[Sin[Pi / 8]]
Sin[Pi / 8] // N
```

```
0.382683

0.382683
```

■ Wichtige Befehle

■ ReadList[ ]

`ReadList["file",{typ1,typ2,...}]` liest Daten aus `file` ein und erstellt eine Liste mit Elementen der Form `{typ1,typ2,...}`.

```
datenliste = ReadList["sinusn.txt", {Real, Real}]
```

```
{{0., 0.}, {1., 0.0174524}, {2., 0.0348995},
 {3., 0.052336}, {4., -0.756802}}
```

`Real`, `Integer`, `Complex` sind hierbei grundlegende Zahlentypen, die Mathematica kennt. Der Punkt hinter einer Zahl in der Ausgabe bedeutet, dass es sich um eine Gleitkommazahl vom Type `Real` und nicht um eine Ganzzahl vom Typ `Integer` handelt.

■ TableForm[ ]

Um diese Daten übersichtlicher darzustellen, kann man sich des Befehles `TableForm` bedienen.

```
TableForm[datenliste]
```

```
0.      0.
1.      0.0174524
2.      0.0348995
3.      0.052336
4.      -0.756802
```

In Postfixnotation wird auch die Befehlseingabe übersichtlicher:

```
datenliste // TableForm
```

```
0.      0.
1.      0.0174524
2.      0.0348995
3.      0.052336
4.      -0.756802
```

Der Tabelle kann man noch Spaltenüberschriften verpassen, indem man beim Funktionsaufruf von `TableForm` die Option `TableHeadings` anspricht. Die leere geschweifte Klammer steht dabei für die hier nicht benutzte Zeilenbeschriftung.

```
TableForm[datenliste,
 TableHeadings → {{}, {"n", "sin(n)"}}]
```

```
n       sin(n)
0.      0.
1.      0.0174524
2.      0.0348995
3.      0.052336
4.      -0.756802
```

Wie man schnell bemerkt, ist eine Postfixnotation unter Verwendung von mehreren Argumenten nicht möglich. Hier muss man sich mit einem Trick behelfen. Auch wenn das zuerst umständlicher aussieht, als in der Funktionsnotation, zahlt es sich bei längeren Ausdrücken durch eine größere Übersichtlichkeit wieder aus. Dazu definiert man eine Hilfsfunktion, die sich auf ein Argument beschränkt. Die primitive Variante sieht dann so aus:

```
tableWithHeads[table_] = TableForm[table,
   TableHeadings → {{}, {"n", "sin(n)"}}];
datenliste // tableWithHeads;
```

Das ergibt zwar das gewünschte Ergebnis, allerdings ist es etwas aufwändig, für die einmalige Benutzung eine eigene Funktion zu definieren. Daher kennt *Mathematica* das Konzept der »pure functions«, das man vielleicht mit »anonyme Funktionen« übersetzen könnte. Eine pure function hat keinen Namen, sondern wird dort definiert, wo sie benutzt wird.

Die ausführliche Variante benutzt dazu den *Mathematica*-Befehl `Function`, der eine solche Funktion erzeugt. Der erste Parameter von `Function` ist der Name der Variablen, die als Parameter der zu definierenden anonymen Funktion dient. Möchte man in seiner Funktion mehrere Parameter benutzen wollen, so gibt man einfach statt einer Variablen eine Liste von Variaben als ersten Parameter von `Function` an. Der zweite Parameter von `Function` ist die Definition der anonymen Funktion. Also das, was auf der rechten Seite des Gleichheitszeichen in der Funktionsdefinition steht. Das Beispiel sieht jetzt so aus:

```
datenliste // Function[table, TableForm[table,
    TableHeadings → {{}, {"n", "sin(n)"}}]];
```

Die benannten Parameter, hier also `table`, machen die ganze Sache zwar etwas übersichtlicher, aber auch länger. Daher gibt es auch noch eine Syntax, bei der nicht nur die Funktion keinen Namen mehr hat, sondern auch die Parameternamen entfallen. Den ersten (und häufig einzigen) Parameter bezeichnet man mit `#`, die weiteren Parameter mit `#2`, `#3` und so weiter. Der Konsistenz wegen kann der erste Parameter auch mit `#1` angesprochen werden. Um diese Syntax zu benutzen, lässt man das erste Argument von `Function` einfach ganz weg:

```
datenliste // Function[
   TableForm[#, TableHeadings → {{}, {"n", "sin(n)"}}]];
```

Und schliesslich gibt es noch die Möglichkeit, auch das Wort `Function` durch den Postfixoperator `&` zu ersetzen, der hinter den Funktionsrumpf zu schreiben ist, wodurch man kürzere und daher übersichtlichere Ausdrücke erhält, sofern man die Möglichkeiten nicht überstrapaziert:

```
datenliste //
 TableForm[#, TableHeadings → {{}, {"n", "sin(n)"}}] &
```

n	sin(n)
0.	0.
1.	0.0174524
2.	0.0348995
3.	0.052336
4.	−0.756802

- **Transpose[ ]**

`Transpose[list]` bildet das Transponierte eines Vektors oder einer Matrix.

```
(v = {{1, 2, 3}}) // MatrixForm
(M = {{11, 12, 13}, {21, 22, 23}, {31, 32, 33}}) //
 MatrixForm
```

$$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

$$\begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix}$$

```
Transpose[v] // MatrixForm
Transpose[M] // MatrixForm
```

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$\begin{pmatrix} 11 & 21 & 31 \\ 12 & 22 & 32 \\ 13 & 23 & 33 \end{pmatrix}$$

- **Replace[ ], ReplaceAll[ ] oder /.**

`ReplaceAll` benutzt Ersetzungsregeln der Form `x→2`, um Ausdrücke umzuformen, bzw. zu lösen. Die Syntax ist `ReplaceAll[Ausdruck,Ersetzungsregel(n)]` oder äquivalent dazu auch `Ausdruck /. Ersetzungsregel(n)`.

```
ReplaceAll[x + x^2, x → 2]
x + x^2 /. x → 2
```

```
6

6
```

Mehrere Ersetzungsregeln werden in Form einer Liste von Ersetzungsregeln angegeben.

```
Cos[a] + b^2 /. {a → 0, b → 1}
```

```
2
```

Im Gegensatz hierzu ersetzt der Befehl `Replace` nur ganze Ausdrücke.

```
Replace[1 + x, 1 + x → 1 + a]
Replace[1 + x, x → a]
```

```
1 + a
```

```
1 + x
```

Das Interessante an `Replace` ist nun, dass man hiermit Ersetzungen bis zu einem vorher bestimmten Level eines Ausdrucks vornehmen kann. Der Level bezeichnet hier die Mathematica−interne Schachtelung der Ausdrücke. Denn Level kann man immer mit der Funktion `FullForm` ansehen, der Ausdrücke vollständig in Funktionsnotation, so wie sie in *Mathematica* intern dargestellt werden und auch eingegeben werden könnten, ausgibt.

```
FullForm[a + b - c * d]
```

```
Plus[a, b, Times[-1, c, d]]
```

```
FullForm[x^2]
```

```
Power[x, 2]
```

```
Replace[x^2, x → a, 1]
```

```
a^2
```

Der Level von x in x^2 ist 1,
wie man aus `FullForm[x^2]` zu ersehen ist :

$$\underbrace{\texttt{Power}\left[\underbrace{\texttt{x}}_{\text{Level 1}}, \underbrace{\texttt{2}}_{\text{Level 1}}\right]}_{\text{Level 0}}$$

```
FullForm[Sin[x]^2]
```

```
Power[Sin[x], 2]
```

```
Replace[Sin[x]^2, x → a, 1]
```

```
Sin[x]^2
```

`Replace` hatte hier keinen Effekt, weil sich x erst auf dem zweiten Level befindet.
Betrachten wir einmal, wie sich das Ersetzen in Listen verhält.

```
{FullForm[{x}] , Replace[{x}, x → a, 2]}
```

```
{List[x], {a}}
```



```
{FullForm[{{x}}] , Replace[{{x}}, x → a, 2]}
```

```
{List[List[x]], {{a}}}
```

```
{FullForm[{{{x}}}] , Replace[{{{x}}}, x → a, 2]}
```

```
{List[List[List[x]]], {{{x}}}}
```

Die dritte Ersetzung hatte wiederum keinen Effekt, da sich das x in der dritten Liste auf der dritten Ebene befindet.

- **Funktionen aus Packages**

Für das folgende Beispiel werden die Befehle `Column`, `DiplayTogether` und `ErrorListPlot` benötigt, die nicht zum Standardsprachumfang gehören, sondern in Zusatzpaketen enthalten sind, die mit der folgenden Eingabe geladen werden. Bitte beachten Sie, dass es sich bei dem Hochkomma »'« nicht um ein Apostroph »’« sondern um einen Akzent grave handelt, der sich auf englischen Tastaturen auf der Taste links neben der »1« befindet.

```
<< Statistics`DataManipulation`
<< Graphics`Graphics`
```

`Column[liste,n]` gibt die n. Spalte einer Liste als Vektor aus.

`ErrorListPlot`[{ ..., {x_i, y_i, Δy_i}, ...}]
zeichnet die Meßwerte (x, y) mit y – Fehlern Δy.

Mit `PlotStyle`→{`Anweisungen`} kann man grafische Eigenheiten, wie Farbe und Strichbreite eines Plots bestimmen. Hier gibt es eine große Anzahl verschiedener Möglichkeiten, die in der Mathematica−Hilfe beschrieben werden. Dies gilt ebenso für den Befehl `PointSize`.

■ Anwendungsbeispiel: Lineare Regression

Dieses Beispiel sollten Sie Schritt für Schritt nachvollziehen. Am Ende werden Sie eine Funktion erhalten, die Sie für die Auswertung Ihrer Praktikumsversuche weiter verwenden können.

- **Laden der Mathematika−Standardpakete**

```
<< Statistics`DataManipulation`
<< Graphics`Graphics`
```

Mit dem nächsten Befehl werden die Messwerte aus einer Textdatei in eine Liste gelesen. In der Datei stehen die Werte durch Leerzeichen getrennt in drei Spalten.

```
linear = ReadList["linear.txt", {Real, Real, Real}]
```

```
{{0., 2.8, 1.}, {2., 5.2, 1.5},
 {4., 6.8, 1.2}, {6., 9.6, 0.9}, {8., 11.2, 1.4},
 {10., 14., 1.4}, {12., 15.6, 1.5}, {14., 17., 1.2},
 {16., 19.8, 1.3}, {18., 21.4, 1.5}}
```

Die inneren Listen der so erzeugten Liste `linear` entsprechen den Zeilen der Datei. Der Zugriff auf die einzelnen Werte wird einfacher, wenn wir Listen erzeugen, die einer Spalte der ursprünglichen Datei entsprechen. Diese Aufgabe lässt sich mit dem Befehl `Column` lösen:

```
x = Column[linear, 1]
y = Column[linear, 2]
Δy = Column[linear, 3]
```

```
{0., 2., 4., 6., 8., 10., 12., 14., 16., 18.}
```

```
{2.8, 5.2, 6.8, 9.6, 11.2, 14., 15.6, 17., 19.8, 21.4}
```

```
{1., 1.5, 1.2, 0.9, 1.4, 1.4, 1.5, 1.2, 1.3, 1.5}
```

In den Formeln für die lineare Regression treten Summen über alle Datenpunkte auf. Um diese auzuwerten, brauchen wir die Zahl der Datenpunkte, d.h., die Zahl der Elemente einer Liste. `Length` liefert den gewünschten Wert.

```
n = Length[x]
```

```
10
```

Jetzt können wir die Formeln aus dem Skript verwenden, um Steigung, Achsenabschnitt und die zugehörigen Fehler der Regressionsgeraden zu ermitteln. Das Summenzeichen wurde aus einer Palette eingefügt, die sich beim Starten von Mathematica automatisch öffnet. Ist dies nicht der Fall, wählen Sie aus dem Menu **File→Palettes→BasicInput**.

$$S = \sum_{i=1}^{n} \frac{1}{(\Delta y[[i]])^2} * \sum_{i=1}^{n} \frac{(x[[i]])^2}{(\Delta y[[i]])^2} - \left(\sum_{i=1}^{n} \frac{x[[i]]}{(\Delta y[[i]])^2} \right)^2$$

```
1395.94
```


$$a0 = \frac{1}{S} * \left(\sum_{i=1}^{n} \frac{(x[[i]])^2}{(\Delta y[[i]])^2} * \sum_{i=1}^{n} \frac{y[[i]]}{(\Delta y[[i]])^2} - \sum_{i=1}^{n} \frac{x[[i]]}{(\Delta y[[i]])^2} * \sum_{i=1}^{n} \frac{x[[i]] * y[[i]]}{(\Delta y[[i]])^2} \right)$$

```
3.01566
```

$$b0 = \frac{1}{S} * \left(\sum_{i=1}^{n} \frac{1}{(\Delta y[[i]])^2} * \sum_{i=1}^{n} \frac{x[[i]] * y[[i]]}{(\Delta y[[i]])^2} - \sum_{i=1}^{n} \frac{x[[i]]}{(\Delta y[[i]])^2} * \sum_{i=1}^{n} \frac{y[[i]]}{(\Delta y[[i]])^2} \right)$$

```
1.03634
```

$$\Delta a0 = \text{Sqrt}\left[\frac{1}{S} * \sum_{i=1}^{n} \frac{(x[[i]])^2}{(\Delta y[[i]])^2} \right]$$

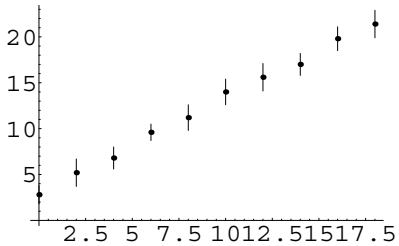
```
0.675303
```

$$\Delta b0 = \text{Sqrt}\left[\frac{1}{S} * \sum_{i=1}^{n} \frac{1}{(\Delta y[[i]])^2} \right]$$

```
0.0685982
```

Nun sind alle Paramter berechnet und wir können das Ergebnis grafisch ausgeben. Die Messpunkte werden mit dem Befehl `ErrorListPlot` ausgegeben:

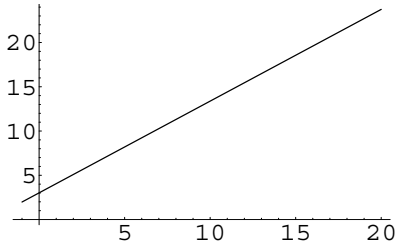
```
ErrorListPlot[linear]
```




```
- Graphics -
```

Die Ausgleichsgerade kann mit dem Befehl `Plot` gezeichnet werden. Das erste Argrument dieser Funktion ist die zu plottende Funktion, das zweite eine Liste, die Laufvariable und die Grenzen der x−Achse beinhaltet. x kann nicht als Laufvariable verwendet werden, weil wir schon eine Liste gleichen Namens definiert haben. Probieren Sie es aus!

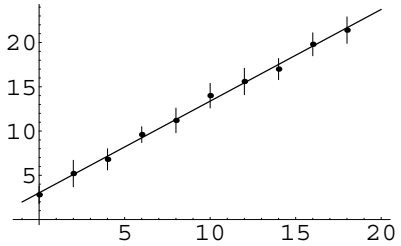
```
Plot[ a0 + b0 t, {t, -1, 20}]
```




```
- Graphics -
```

Jetzt müssen wir nur noch beide Graphen zusammen anzeigen lassen. Hierzu verwenden wir den Befehl `DisplayTogether`, der als Argrument eine Liste von Grafik−Befehlen erhält und auch dafür sorgt, dass alle Graphen gleich skaliert werden.

```
DisplayTogether[
 {ErrorListPlot[linear], Plot[ a0 + b0 t, {t, -1, 20}]}]
```




```
- Graphics -
```


■ Lineare Regession als wiederverwendbare Funktion

Die Aufgabe, Lineare Regression auf einen Satz Werte anzuwenden, haben wir im vorigen Abschnitt gelöst. Wenn wir das Ganze nun auf eine andere Datei anwenden wollen, müssen wir die Schritte alle einzeln wiederholen. Oder besser das Notebook unter einem anderen Namen speichern, den Dateinamen der Messwerte ändern und das Notebook neu auswerten. Sie könnten so verfahren und diesen Abschnitt überspringen, wenn Sie nicht weiter in Mathematica eindringen wollen. Sinnvoller ist es aber, eine Funktion zu definieren, der die Werte übergeben werden und die dann alle Berechnungen durchführt.

Funktionen, die irgendwo aufgerufen werden, dürfen keine eventuell schon vorhandenen Variablen überschreiben. Deshalb ist die Verwendung von lokalen Variablen unerlässlich. Lokale Variablen sind Variablen, die nur innerhalb eines gewissen Bereiches −hier einer Funktion −gelten.

Im Gegensatz zu anderen Programmiersprachen sind innerhalb von Funktionen definierte neue Variable in Mathematica grundsätzlich global. Dies ist eine gefährliche Fehlerquelle!

Um Variablen als lokal zu deklarieren wird der Befehl `Module` verwendet. Erstes Argrument ist eine Liste der lokalen Variabeln, das zweite die Anweisungen, die ausgeführt werden sollen. Die einzelnen Anweisungen werden mit Semikolons getrennt. Das Semikolon unterdrückt gleichzeitig die Ausgabe des jeweiligen Befehles. Vergessen Sie das Semikolon, erhalten Sie eine auf den ersten Blick nicht unbedingt nachvollziehbare Fehlermeldung. Probieren Sie es aus, damit sie später nicht verzweifeln!
Beginnen wir mit einem einfachen Beispiel, die Übergabe von Parametern haben wir ja schon kennen gelernt:

```
EinfachesBeispiel[a_, b_] := Module[{summe, mittel},
       summe = a + b;
  mittel = summe / 2
 ]
```

Mit dieser Funktion kann man tatsächlich den Mittelwert zweier Zahlen berechnen:

```
EinfachesBeispiel[3, 5]
```

```
4
```

Die lokalen Variablen existieren nach diesem Aufruf nicht mehr. (Hier platzsparend als Liste ausgegeben):

```
{summe, mittel, a, b}
```

```
{summe, mittel, a, b}
```

Auch eine vor dem Funktionsaufruf definierte globale Variable wird nicht überschrieben:

```
summe = 12
```

```
12
```

```
EinfachesBeispiel[2, 6]
```

```
4
```

```
summe
```

```
12
```

Schreiten wir also zur Tat und bauen wir unser Beispiel zu einer Funktion um. Als Parameter übergeben wir unserer Funktion die aus der Datei eingelesene Liste, dann folgt die Deklaration der lokalen Variabeln, die Summenindizes sind dabei automatisch lokal. Die restlichen Zeilen können wir unverändert weiter verwenden. Dabei helfen die Funktionen Kopieren und Einfügen. Lediglich ein paar Semikolons müssen ergänzt werden. Nicht mehr benötigte Teile des Notebooks können Sie löschen, in dem Sie die Zellen am rechten Rand mit der Maus markieren und DEL drücken. Durch dieses Löschen vergisst *Mathematica* Ihre Eingaben jedoch noch nicht. Das können Sie nur durch Beenden des Kernels und erneutes Auswerten des Notebooks erreichen. Deshalb sollten Sie dies stets tun bevor Sie eine mit *Mathematica* erstellte Lösung zum Abgeben ausdrucken.


```
LinReg[daten_] :=
 Module[{x, y, Δy, S, Δa0, Δb0, a0, b0, n},
  n = Length[daten];
  x = Column[daten, 1];
  y = Column[daten, 2]; Δy = Column[daten, 3];
  S = Sum[1/(Δy[[i]])^2 * Sum[(x[[i]])^2/(Δy[[i]])^2, {i, 1, n}] -
      (Sum[x[[i]]/(Δy[[i]])^2, {i, 1, n}])^2;
  a0 = 1/S * (Sum[(x[[i]])^2/(Δy[[i]])^2, {i, 1, n}] *
      Sum[y[[i]]/(Δy[[i]])^2, {i, 1, n}] -
      Sum[x[[i]]/(Δy[[i]])^2, {i, 1, n}] *
      Sum[x[[i]] * y[[i]]/(Δy[[i]])^2, {i, 1, n}]);
  b0 = 1/S * (Sum[1/(Δy[[i]])^2, {i, 1, n}] *
      Sum[x[[i]] * y[[i]]/(Δy[[i]])^2, {i, 1, n}] -
      Sum[x[[i]]/(Δy[[i]])^2, {i, 1, n}] *
      Sum[y[[i]]/(Δy[[i]])^2, {i, 1, n}]);
  Δa0 = Sqrt[1/S * Sum[(x[[i]])^2/(Δy[[i]])^2, {i, 1, n}]];
  Δb0 = Sqrt[1/S * Sum[1/(Δy[[i]])^2, {i, 1, n}]]
 ]
```

Wenden wir die Funktion nun auf unseren Datensatz an, erhalten wir:

```
LinReg[linear]
```

```
0.0685982
```

Das ist nicht ganz das, was wir uns vorgestellt haben. Es wird nur das Ergebnis der letzten Anweisung ausgegeben. Um mehrere Variablen in übersichtlicher Form auszugeben, unterdrücken wir zunächst auch die Ausgabe der letzten Anweisung mit einem Semikolon und erweitern die Prozedur um die folgenden Zeilen:

```
Print[
  "Berechnung der Fehler aus den Eingangsfehlern"];
Print[TableForm[{{"", "Wert", "Fehler"}, {"Steigung b",
     b0, Δb0}, {"Achsenabschnitt a", a0, Δa0}}]];
```

Der Befehl Print erzeugt auch dann eine Ausgabe, wenn er mit einem Semikolon abgeschlossen wird.

Als Ausgabe erhalten wir dann:

```
LinReg[linear]
```

```
Berechnung der Fehler aus den Eingangsfehlern

                      Wert        Fehler
Steigung b            1.03634     0.0685982
Achsenabschnitt a     3.01566     0.675303
```

Um auf die von unserer Funktion berechneten Werte in folgenden Rechnungen einfach zugreifen zu können, lassen wir die Funktion mit einer zusätzlichen Zeile eine Liste von Ersetzungsregeln zurückgeben. Beachten sie, dass die Anweisung, deren Ausgabe der zurückgegebene Funktionswert sein soll, nicht mit einem Semikolon abgeschlossen werden darf und folglich als letzte stehen muss.

```
{b → b0, Δb → Δb0, a → a0, Δa → Δa0}
```

■ Anwendung der Funktion

Um die Funktion anzuwenden und mit den berechneten Werten weiterzurechnen, wird die von der Funktion zurückgegebene Liste in einer Variabeln gespeichert.


```
regerg = LinReg[linear]
```

```
Berechnung der Fehler aus den Eingangsfehlern

                      Wert        Fehler
Steigung b            1.03634     0.0685982
Achsenabschnitt a     3.01566     0.675303

{b → 1.03634, Δb → 0.0685982, a → 3.01566, Δa → 0.675303}
```

Auf einzelne der berechneten Werte können wir jetzt zugreifen, indem wir ausnutzen, dass `regerg` eine Liste von Ersetzungsregeln ist, und daher a, Δa, b und Δb durch Anwendung der Ersetzregeln mit `/. regerg` durch ihre Werte ersetzt werden.

```
a /. regerg
```

```
3.01566
```

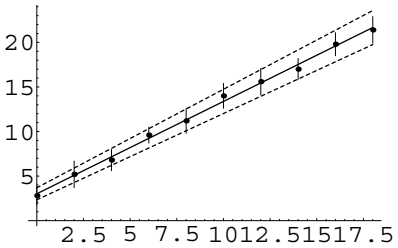
oder etwas komplizierter:

```
a + b /. regerg
```

```
4.052
```

Ausgleichs−und Grenzgeraden können dann wie folgt erzeugt werden:

```
DisplayTogether[ErrorListPlot[linear],
  Plot[
  Evaluate[{b t + a, (b + Δb) t + a + Δa, (b - Δb) t + a - Δa} /.
    regerg], {t, 0, 18},
   PlotStyle → {Dashing[{.0}], Dashing[{0.01}],
    Dashing[{.01}]} ] ]
```




```
- Graphics -
```

Der Befehl `Evaluate` ist an dieser Stelle notwendig, um zu erzwingen, dass erst die Ersetzung vorgenommen wird und dann die Stützstellen der Kurven berechnet werden. Lässt man ihn weg, erhält man eine Fehlermeldung, weil Mathematica bei Abarbeitung des Plotbefehls

zunächst die zu plottende Funktion kompiliert, um die Rechnung zu beschleunigen. Dies ist mit Ersetzungsanweisungen nicht möglich.
Der Parameter `PlotStyle` dient dazu, die Geraden unterscheidbar zu machen. Details finden Sie in der Hilfe.

■ Linearisierte Exponentialfunktionen

Lesen wir zunächst eine Datei mit Messwerten, die einem exponentiellen Gesetz folgen, ein. Schön wäre es, wenn die Werte dabei gleich übersichtlich ausgegeben werden könnten. Dies gelingt mit dem schon zuvor gebrauchten Befehl `TableForm` – diesmal als nachgestellter Operator.

```
exponential = ReadList["exp.txt", {Real, Real, Real}] //
  TableForm[#, TableHeadings → {{}, {"x", "y", "Δy"}}] &
```

x	y	Δy
0.	82.	1.
2.	56.	1.
4.	37.	1.
6.	26.	1.
8.	18.	1.
10.	11.	1.
12.	9.	1.
14.	5.	1.
16.	3.	1.

Versucht man weiter zu arbeiten, stellt man schnell fest, dass man auf die einzelnen Listenelemente nicht mehr richtig zugreifen kann. Dies liegt daran, dass die Variable `exponential` nun die formatierte Tabelle enthält. Durch richtiges Klammern gelingt es, der Variablen die unformatierte Liste zuzuweisen und dennoch eine formatierte Ausgabe zu erzeugen:

```
(exponential =
    ReadList["exp.txt", {Real, Real, Real}] ) //
  TableForm[#, TableHeadings → {{}, {"x", "y", "Δy"}}] &;
```

Der Funktion `LinReg` werden nun nicht die Werte direkt sondern die logarithmierten Werte mit entsprechend umgerechneten Fehlern übergeben. Das dritte Argument der `Replace`–Anweisung gibt dabei an, dass nur bis zur zweiten Schachtelungsebene der Liste ersetzt werden soll.


```
(linearisiert = Replace[exponential,
    {xi_, yi_, Δyi_} → {xi, Log[yi], (1 / yi) * Δyi}, 2]) //
 TableForm[#, TableHeadings →
    {{}, {"x", "ln(y)", "Δln(y) = 1/y Δy"}}] &
```

x	ln(y)	$\Delta\ln(y)=\frac{1}{y}\Delta y$
0.	4.40672	0.0121951
2.	4.02535	0.0178571
4.	3.61092	0.027027
6.	3.2581	0.0384615
8.	2.89037	0.0555556
10.	2.3979	0.0909091
12.	2.19722	0.111111
14.	1.60944	0.2
16.	1.09861	0.333333

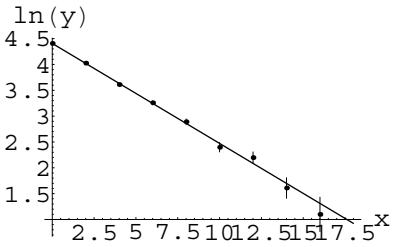
```
experg = LinReg[linearisiert]
```

```
Berechnung der Fehler aus den Eingangsfehlern

                        Wert           Fehler
Steigung b              -0.193466      0.00365475
Achsenabschnitt a       4.40686        0.010881

{b → -0.193466, Δb → 0.00365475,
 a → 4.40686, Δa → 0.010881}
```

```
DisplayTogether[
 ErrorListPlot[linearisiert,
  AxesLabel → {"x", "ln(y)"}],
 Plot[Evaluate[b t + a /. experg], {t, 0, 18}]
]
```




```
- Graphics -
```

Ergebnis

Wenn Sie an diesem Punkt angekommen sind, sollten Sie ein Notebook vor sich haben, dass etwa wie folgt aussieht:

```
<< Statistics`DataManipulation`
<< Graphics`Graphics`

LinReg[daten_] :=
 Module[{x, y, Δy, S, Δa0, Δb0, a0, b0, n},
  n = Length[daten];
  x = Column[daten, 1];
  y = Column[daten, 2]; Δy = Column[daten, 3];

  S = Sum[1/(Δy[[i]])^2, {i, 1, n}] * Sum[(x[[i]])^2/(Δy[[i]])^2, {i, 1, n}] -
      (Sum[x[[i]]/(Δy[[i]])^2, {i, 1, n}])^2;

  a0 = 1/S * (Sum[(x[[i]])^2/(Δy[[i]])^2, {i, 1, n}] * Sum[y[[i]]/(Δy[[i]])^2, {i, 1, n}] -
       Sum[x[[i]]/(Δy[[i]])^2, {i, 1, n}] * Sum[x[[i]] * y[[i]]/(Δy[[i]])^2, {i, 1, n}]);

  b0 = 1/S * (Sum[1/(Δy[[i]])^2, {i, 1, n}] * Sum[x[[i]] * y[[i]]/(Δy[[i]])^2, {i, 1, n}] -
       Sum[x[[i]]/(Δy[[i]])^2, {i, 1, n}] * Sum[y[[i]]/(Δy[[i]])^2, {i, 1, n}]);

  Δa0 = Sqrt[1/S * Sum[(x[[i]])^2/(Δy[[i]])^2, {i, 1, n}]];

  Δb0 = Sqrt[1/S * Sum[1/(Δy[[i]])^2, {i, 1, n}]];

  Print[
   "Berechnung der Fehler aus den Eingangsfehlern"];
  Print[TableForm[{{"", "Wert", "Fehler"},
     {"Steigung b", b0, Δb0},
     {"Achsenabschnitt a", a0, Δa0}}]];
  {b → b0, Δb → Δb0, a → a0, Δa → Δa0}
 ]
```


■ Auswertung lineare Funktion

```
linear = ReadList["linear.txt", {Real, Real, Real}]
```

```
{{0., 2.8, 1.}, {2., 5.2, 1.5},
 {4., 6.8, 1.2}, {6., 9.6, 0.9}, {8., 11.2, 1.4},
 {10., 14., 1.4}, {12., 15.6, 1.5}, {14., 17., 1.2},
 {16., 19.8, 1.3}, {18., 21.4, 1.5}}
```

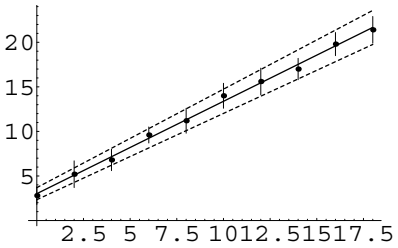
```
regerg = LinReg[linear]
```

```
Berechnung der Fehler aus den Eingangsfehlern

                          Wert        Fehler
Steigung b                1.03634     0.0685982
Achsenabschnitt a         3.01566     0.675303

{b → 1.03634, Δb → 0.0685982, a → 3.01566, Δa → 0.675303}
```

```
DisplayTogether[ErrorListPlot[linear],
  Plot[
  Evaluate[{b t + a, (b + Δb) t + a + Δa, (b - Δb) t + a - Δa} /.
    regerg], {t, 0, 18},
   PlotStyle → {Dashing[{.0}], Dashing[{0.01}],
    Dashing[{.01}]} ] ]
```




```
- Graphics -
```

■ **Auswertung Exponentialfunktion**

```
(exponential =
   ReadList["exp.txt", {Real, Real, Real}] ) //
 TableForm[#, TableHeadings → {{}, {"x", "y", "Δy"}}] &
```

x	y	Δy
0.	82.	1.
2.	56.	1.
4.	37.	1.
6.	26.	1.
8.	18.	1.
10.	11.	1.
12.	9.	1.
14.	5.	1.
16.	3.	1.

```
(linearisiert = Replace[exponential,
    {xi_, yi_, Δyi_} → {xi, Log[yi], (1 / yi) * Δyi}, 2]) //
 TableForm[#, TableHeadings →
    {{}, {"x", "ln(y)", "Δln(y) = 1/y Δy"}}] &
```

x	ln(y)	$\Delta\ln(y) = \frac{1}{y}\Delta y$
0.	4.40672	0.0121951
2.	4.02535	0.0178571
4.	3.61092	0.027027
6.	3.2581	0.0384615
8.	2.89037	0.0555556
10.	2.3979	0.0909091
12.	2.19722	0.111111
14.	1.60944	0.2
16.	1.09861	0.333333

```
experg = LinReg[linearisiert]
```

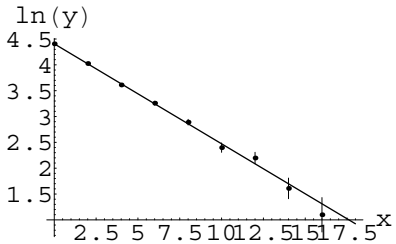
```
Berechnung der Fehler aus den Eingangsfehlern
```

	Wert	Fehler
Steigung b	-0.193466	0.00365475
Achsenabschnitt a	4.40686	0.010881

```
{b → -0.193466, Δb → 0.00365475,
 a → 4.40686, Δa → 0.010881}
```



```
DisplayTogether[
 ErrorListPlot[linearisiert,
  AxesLabel → {"x", "ln(y)"}],
 Plot[Evaluate[b t + a /. experg], {t, 0, 18}]
]
```




```
- Graphics -
```

Literatur:

Die beiden folgenden Titel sind bei der Auskunft in der Fachbereichsbibliothek erhältlich. Sie können im Rahmen den Kurzausleihe für einen Tag (z.B. in den Rechnerraum), über Nacht oder übers Wochenende ausgeliehen werden.

Wolfram, Stephen: *Mathematica book*. 4 Aufl. 1999
Gaylord, Richard J. et al: *Programming with Mathematica*. 2. Aufl 1996

Das *Mathematica book* enthält eine gut lesbare Einführung, die vor allem das Lösen kleinerer mathematischer Probleme ohne eigentliche Programmierung erklärt. Ausserdem enthält es eine ausführliche Beschreibung der Befehle von *Mathematica* und der meisten Standard−Packages. Das Buch ist auch im Volltext über die Hilfe−Funktion zugänglich.

Programming with Mathematica richtet sich an völlige Anfänger sowohl im Bezug auf Mathematica als auch im Hinblick auf allgemeine Programmiertechniken. Es führt gleichzeitig Schritt für Schritt in Mathematica und in elementare Programmiertechniken ein.