Züchtungslehre

Peter von Rohr

2017-09-20

Contents

Vorwort

Züchtungslehre ist der Titel der Vorlesung 751-6305-00L, welche an der ETH Zürich im Herbstsemester für Studierende des Masterstudiengangs Agrarwissenschaften angeboten wird. Dieses Dokument enthält alle relevanten Unterlagen zu dieser Vorlesung.

Im Kontext dieser Vorlesung verstehen wir das Fachgebiet der Züchtungslehre als die Methoden und die Techniken, welche in der Züchtung von landwirtschaftlichen Nutztieren verwendet werden. Unter **Züchtung** verstehen wir hier die gezielte Auswahl (Selektion) und die gezielte Anpaarung von Elterntieren. Aus diesen Paarungen entsteht dann eine Nachkommengeneration, welche dem Idealbild eines Nutztiers besser entspricht als die Eltern. Das Idealbild wird über das Zuchtziel festgelegt.

Voraussetzungen

Da bei der Paarung von Elterntieren von jedem Elternteil eine zufällige Auswahl von Genvarianten der Eltern an die Nachkommen weitergegeben wird, sind die in der Züchtung von Nutztieren ablaufenden Prozesse nicht deterministisch kontrollierbar. Das heisst, es gibt bei der Weitergabe von Genvarianten der Eltern an ihre Nachkommen immer einen gewissen unkontrollierbaren oder zufälligen Anteil im gesamten Prozess.

Sobald Zufallsprozesse ablaufen, kommen die Werkzeuge der Wahrscheinlichkeitsrechnung und der Statistik zur Anwendung. Für unsere Anwendung der Züchtung bedeutet das, dass wir den genetischen Wert eines Tieres anhand eines statistischen Modells charakterisieren.

Somit sind neben der statistischen Modellierung von Daten auch die Grundlagen der linearen Algebra und grundlegende Kenntnisse der quantitativen Genetik als Voraussetzung für diese Vorlesung in Züchtungslehre zu bezeichnen.

Die Voraussetzungen werden aber alle zu Beginn der Vorlesung erklärt. Somit braucht es keine Vorkenntnisse um diese Vorlesung erfolgreich zu absolvieren.

Lernziele

Für die Verwendung des hier präsentierten Stoffs schlagen wir die folgenden Lernziele vor.

Die Studierenden …

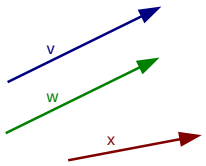
Chapter 1

Einführung in Lineare Algebra

Aus der linearen Algebra brauchen wir für diese Vorlesung nur das Rechnen mit Vektoren und Matrizen. Der Grund, weshalb Vektoren und Matrizen wichtig sind, ist dass sie uns das Aufstellen und das Lösen von Gleichungssystemen wesentlich erleichtern. Lineare Gleichungssysteme sind in der traditionellen Zuchtwertschätzung, d.h. überall dort, wo die genomische Selektion noch nicht angewendet wird, sehr wichtig.

1.1 Was ist ein Vektor

Ein Vektor ist durch seine **Länge** und seine **Richtung** eindeutig bestimmt. Das heisst, sind zwei Vektoren v und w gleich lang und haben die gleich Richtung, dann sind die beiden Vektoren gleich und somit gilt $v = w$. Haben zwei Vektoren v und x nicht die gleiche Richtung oder sind nicht gleich lang, dann sind die Vektoren nicht gleich und somit gilt $v \neq x$ und somit auch $w \neq x$.

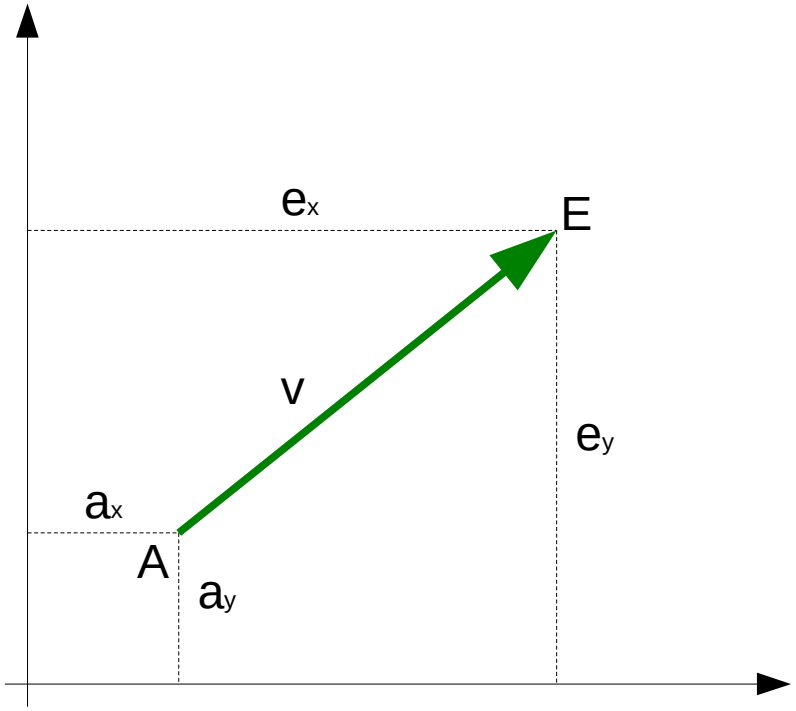


1.1.1 Koordinaten

Durch die Einführung eines Koordinatensystems wird jedem Punkt im n-dimensionalen Raum ein n-Tupel von Zahlen -die so genannten Koordinaten - zugewiesen. Zum Beispiel bekommt jeder Punkt in einer Ebene (zweidimensionaler Raum) ein Paar von Koordinaten zugewiesen.

Die Koordinaten eines Vektors errechnen sich aus der Differenz der Koordinaten des Endpunktes des Vektors minus die Koordinaten des Ausgangspunktes des Vektors. Betrachten wir als Beispiel im nachfolgenden Bild den Vektor v mit Anfangspunkt A und Endpunkt E, dann errechnen sich die Koordinaten als Differenzen zwischen den Koordinaten von E minus die Koordinaten von A, das bedeutet

$$v = \begin{bmatrix} e_x - a_x \\ e_y - a_y \end{bmatrix}$$



1.1.2 Geometrie

In der elementaren Geometrie werden Vektoren häufig durch Pfeile dargestellt. Parallele Pfeile mit gleicher Länge werden Repräsentanten des gleichen Vektors bezeichnet. Repräsentanten, welche ihren Anfangspunkt im Ursprung des Koordinatensystems haben, sind speziell und werden als Ortsvektoren des jeweiligen Endpunkts bezeichnet. Die Koordinaten des Endpunktes und die Koordinaten des Vektors sind identisch.

1.2 Operationen mit Vektoren

Vektoren als solches werden erst dann richtig nützlich, wenn man sie mit Operationen verknüpfen kann. Konkret heisst das, die Einführung von Operationen zwischen Vektoren erlaubt es uns mit ihnen zu rechnen.

1.2.1 Addition und Subtraktion

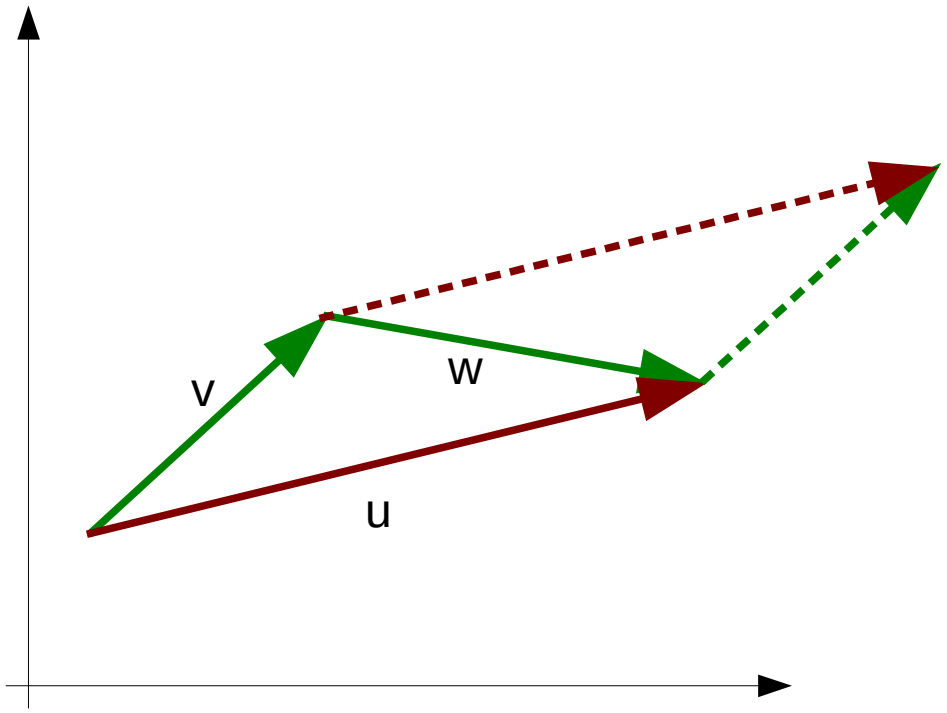
Die Addition zweier Vektoren v und w hat als Resultat wieder einen Vektor, sagen wir er heisse u. Es gilt somit, dass

$$v + w = u$$

Die Koordinaten des Summenvektors berechnen sich als die Summen der Koordinaten der Summandenvektoren. Angenommen die Koordinaten der Summandenvektoren v und w werden mit v_x, v_y, w_x und w_y bezeichnet, dann entsprechen die Koordinaten von u

$$u = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} v_x + w_x \\ v_y + w_y \end{bmatrix}$$

Graphisch entspricht die Addition zweier Vektoren v und w der Verkettung der beiden Pfeile.



Die Reihenfolge der beiden Summanden ist nicht wichtig. Es gilt also, dass $v + w = u = w + v$. Dies ist auch aus der obigen Abbildung ersichtlich.

Die Subtraktion kann aus der Addition abgeleitet werden. Subtrahiert man auf beiden Seiten der Additionsgleichung einer der Vektoren, dann folgt

$$v = u - w$$

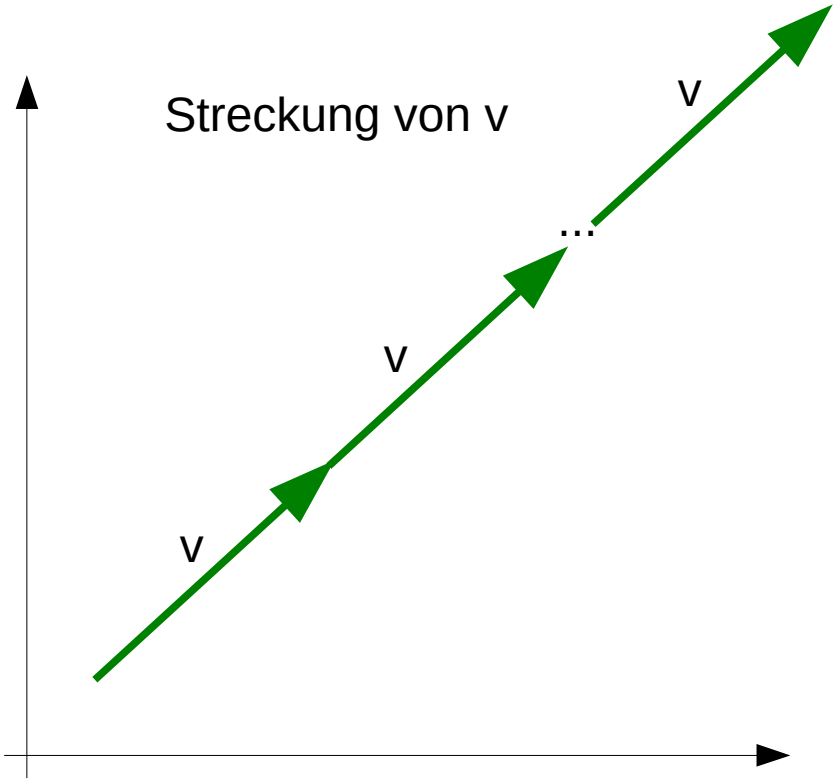
und

$$w = u - v$$

1.2.2 Multiplikation mit einem Skalar

Die Multiplikation eines Vektors mit einer skalaren Grösse, d.h. mit einer Zahl führt zu einer Streckung oder einer Stauchung des Vektors. Die Koordinaten des Resultatvektors entsprechen den Koordinaten des Ursprungsvektors, welche mit dem skalaren Faktor multipliziert werden.

$$u = \lambda * v = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \lambda * v_x \\ \lambda * v_y \end{bmatrix}$$



Je nach Wert von λ verändert v die Länge und die Richtung. Folgende Tabelle gibt eine Übersicht zu den Änderungen.

Faktor	Richtung	Länge
$\lambda < -1$	entgegengesetzt	länger
$\lambda = -1$	entgegengesetzt	gleich
$-1 < \lambda < 0$	entgegengesetzt	kürzer
$\lambda = 0$	unbestimmt	kürzer
$0 < \lambda < 1$	gleich	kürzer
$\lambda = 1$	gleich	gleich
$\lambda > 1$	gleich	länger

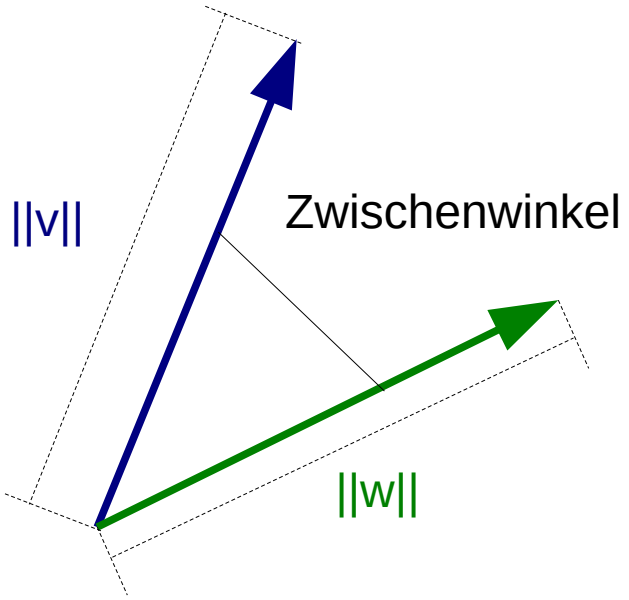
1.2.3 Skalarprodukt

Das Skalarprodukt zweier Vektoren entspricht einer Zahl. Diese berechnet sich aus den Produkten der einzelnen Koordinaten, welche dann addiert werden. Konkret bedeutet das

$$v \cdot w = v_x * w_x + v_y * w_y$$

Wird das Skalarprodukt zweier Vektoren um das Produkt der Längen der beiden Vektoren skaliert, so resultiert der Kosinus des Zwischenwinkels der beiden Vektoren. Somit liefert das Skalarprodukt einen

Vergleich bezüglich der Richtungen von zwei Vektoren.



In einer Formel geschrieben, heisst das

$$cos(\alpha) = \frac{v \cdot w}{||v|| * ||w||}$$

Aufgrund der Eigenschaften der Winkelfunktion *cos* können wir sagen, dass Vektoren, deren Skalarprodukt 0 ist, senkrecht zueinander stehen. Vektoren deren Skalarprodukt gleich 1 ist, haben einen Zwischenwinkel von 0 und verlaufen somit parallel zueinander.

1.3 Was ist eine Matrix

Werden mehrere Vektoren "nebeneinander" gestellt, resultiert ein neues Objekt, welches wir als **Matrix** bezeichnen. Stellen wir als Beispiel die Vektoren v und w nebeneinander, dann erhalten wir die Matrix M

$$M = \begin{bmatrix} v & w \end{bmatrix} = \begin{bmatrix} v_x & w_x \\ v_y & w_y \end{bmatrix}$$

Eine Matrix kann auch als Schema von $m * n$ Zahlen definiert werden, welche in m Zeilen und n Kolonnen angeordnet sind. Man spricht dann auch von einer $m * n$ - Matrix. Matrizen werden im allgemeinen mit Grossbuchstaben bezeichnet. Die Elemente einer bestimmten Matrix A werden mit a_{ij} bezeichnet, wobei i dem Zeilenindex und j dem Kolonnenindex entsprechen.

Als Beispiel betrachten wir die $2 * 3$-Matrix A

$$A = \begin{bmatrix} 2 & 3 & 0 \\ -1 & 4 & 7 \end{bmatrix}$$

Das Element a_{12} ist das Element aus der ersten Zeile und der zweiten Kolonne von A also ist $a_{12} = 3$.

1.3.1 Eigenschaften von Matrizen

Die Anzahl Zeilen und Kolonnen einer Matrix werden auch als **Dimension** bezeichnet. Als Beispiel ist die Dimension der Matrix M, welche wir oben aus den Vektoren v und w zusammengesetzt haben ist $2 * 2$.

1.3.2 Matrixoperationen

Addition und **Subtraktion** von Matrizen sind analog zu den entsprechenden Operationen zwischen Vektoren definiert. Die Addition und Subtraktion von Matrizen wird Element-weise durchgeführt. Nehmen wir an, wir haben die Matrizen A und B, dann ist deren Summen S wieder eine Matrix,

$$S = A + B = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$$

Für die Subtraktion können wir schreiben

$$A = S - B = \begin{bmatrix} s_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix} - \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} s_{11} - b_{11} & s_{12} - b_{12} \\ s_{21} - b_{21} & s_{22} - b_{22} \end{bmatrix}$$

Die Addition und die Subtraktion ist nur zwischen Matrizen mit gleicher Dimension möglich.

Die **Matrixmultiplikation** zwischen den Matrizen A und B lässt sich als eine Serie von Skalarprodukten zwischen den Zeilen von A und den Kolonnen von B auffassen. Die nachfolgende Formel gibt eine formale Definition der Matrixmultiplikation. Das Produkt der Matrizen $A \cdot B$ ist wieder eine Matrix. Der Term $(A \cdot B)_{ij}$ steht für das Element auf Zeile i und Kolonne j der Produktmatrix. Dieses Element $(A \cdot B)_{ij}$ entspricht

$$(A \cdot B)_{ij} = \sum_{k=1}^{n} a_{ik} * b_{kj}$$

Die Summation in der oben gezeigten Formel läuft über die Kolonnen von Matrix A und über die Zeilen von Matrix B.

Die Matrixmultiplikation ist nur möglich, falls die Dimensionen der beiden Matrixfaktoren kompatibel sind. Die Anzahl Kolonnen des ersten Faktors A muss der Anzahl Zeilen des zweiten Faktors B entsprechen, nur dann kann das Produkt $A \cdot B$ berechnet werden. Es können also nur Matrizen der Dimension $m * n$ mit Matrizen der Dimension $n*p$ multipliziert werden. Im allgemeinen können die Faktoren auch nicht vertauscht werden, d.h. im allgemeinen Fall gilt, dass

$$A \cdot B \neq B \cdot A$$

Werden Addition und Multiplikation kombiniert, gilt für entsprechend kompatible Matrizen A, B und C das Kommutativgesetz.

$$(A + B) \cdot C = A \cdot C + B \cdot C$$

Wichtig ist aber auch hier die Einhaltung der Reihenfolge der Faktoren, das heisst, da Matrix C von links zur Summe $(A + B)$ multipliziert wird, muss sie auch zu jedem Summanden von links multipliziert werden.

1.3.3 Spezielle Matrizen

Die **Nullmatrix** O ist die Matrix, deren Elemente o_{ij} alle gleich 0 sind.

$$O = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Diese Matrix ist das Neutralelement der Addition und der Subtraktion. Somit gilt, dass

$$A + O = A - O = A$$

für alle Matrizen A. Das Produkt einer beliebigen Matrix A mit der Nullmatrix O ist wieder gleich der Nullmatrix.

$$A \cdot O = O$$

Eine quadratische Matrix R heisst **obere Dreiecksmatrix** oder **Rechtsmatrix** falls $(R)_{ij} = 0$ für alle $i > j$. Als Beispiel ist

$$R = \begin{bmatrix} 1 & 3 & -7 \\ 0 & 2 & 5 \\ 0 & 0 & 9 \end{bmatrix}$$

eine obere Dreiecksmatrix.

Eine quadratische Matrix L heisst **untere Dreiecksmatrix** oder **Linksmatrix** falls $(L)_{ij} = 0$ für alle $i < j$. Die folgende Matrix L ist ein Beispiel für eine untere Dreiecksmatrix.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 2 & 0 \\ -2 & 4 & 9 \end{bmatrix}$$

Die **Einheitsmatrix** I ist eine quadratische Matrix, deren Diagonolaelemente alle gleich 1 sind und deren Off-Diagonalelemente alle gleich 0 sind.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Die Einheitsmatrix I ist das Neutralelement der Matrixmultiplikation. Wir können also schreiben

$$A \cdot I = A$$

Die **Transponierte** A^T einer Matrix A entsteht durch vertauschen von Zeilen und Kolonnen. Somit hat die Matrix A^T einer $m * n$-Matrix die Dimensionen $n * m$. Die Elemente in Matrix A werden einfach an der Hauptdiagonalen gespiegelt.

$$(A)_{ij} = (A^T)_{ji}$$

Folgende Regeln im Bezug auf transponierte Matrizen gelten:

- die Transponierte von A^T ist wieder die Matrix A: $(A^T)^T = A$
- Summe: $(A + B)^T = A^T + B^T$
- Produkt: $(A \cdot B)^T = B^T \cdot A^T$

- Einheitsmatrix: $I^T = I$

Bei einer symmetrischen Matrix A, bei welcher gilt, dass $(A)_{ij} = (A)_{ji}$, ist die transponierte Matrix A^T gleich der ursprünglichen Matrix A.

Die **Inverse** A^{-1} einer Matrix A ist definiert, als diejenige Matrix für welche gilt, dass

$$A \cdot A^{-1} = I$$

Falls die inverse Matrix A^{-1} existiert, dann wird die Matrix A als invertierbar bezeichnet. Die Inverse ist eindeutig, das heisst, es gibt zu jeder Matrix A nur eine inverse Matrix. Falls wir eine Matrix B finden für welche gilt, dass

$$A \cdot B = I$$

dann wissen wir sofort, dass $B = A^{-1}$.

Folgende Regeln gelten im Bezug auf inverse Matrizen.

- Inverse der Inversen: $(A^{-1})^{-1} = A$
- Produkt: $(A \cdot B)^{-1} = B^{-1} \cdot A^{-1}$
- Transponierte: $(A^T)^{-1} = (A^{-1})^T$
- Einheitsmatrix: $I^{-1} = I$

1.4 Gleichungssysteme

Lineare Gleichungssysteme spielen in der Züchtung, namentlich bei der Zuchtwertschätzung eine wichtige Rolle. In einem Gleichungssystem werden Beziehungen zwischen bekannten und unbekannten Grössen verwendet um Lösungen für die unbekannten Grössen berechnen zu können. Bei der Klasse der linearen Gleichungssysteme beschränkt man sich auf lineare Beziehungen zwischen bekannten und unbekannten Grössen. Für unser Anwendungsbeispiel der Zuchtwertschätzung werden wir Beziehungen zwischen unbekannten Umwelteffekten und unbekannten Zuchtwerten den phänotypischen Leistungen für alle Tiere in einer Population gleichsetzen. Daraus resultieren sehr grosse Gleichungssystem, d.h. die Anzahl Gleichungen kann sehr gross sein.

Als einführendes Beispiel schauen wir uns folgendes Gleichungssystem mit zwei Gleichungen und zwei Unbekannten x_1 und x_2 an.

$$\begin{aligned} x_1 + 2x_2 &= 5 \\ 2x_1 + 3x_2 &= 8 \end{aligned} \tag{1.1}$$

Wir wollen nun Zahlen für die Unbekannten x_1 und x_2 finden, so dass beide Gleichungen erfüllt sind. Die Zahlen $x_1 = 1$ und $x_2 = 2$ erfüllen beide Gleichungen. Sie stellen somit die **Lösung** für unser Gleichungssystem dar. Im allgemeinen besteht ein Gleichungssystem aus m Gleichungen mit n unbekannten. Für unser erstes Beispiel ist $m = 2$ und $n = 2$.

Aufgrund des folgenden Beispiels erkennen wir, dass nicht jedes Gleichungssystem eine Lösung hat.

$$\begin{aligned} x_1 + x_2 &= 4 \\ 2x_1 + 2x_2 &= 5 \end{aligned} \tag{1.2}$$

Da wir keine Zahlen x_1 und x_2 finden können, deren Summe gleich 4 ist und deren doppelte Summe gleichzeitig gleich 5 ist. Aus der ersten Gleichung folgt, dass $2x_1 + 2x_2 = 8$ und das ist ein Widerspruch zur zweiten Gleichung.

Als drittes Beispiel betrachten wir ein Gleichungssystem mit $m = 2$ Gleichungen und $n = 3$ Unbekannten.

$$\begin{aligned} x_1 - x_2 + x_3 &= 2 \\ 2x_1 + x_2 - x_3 &= 4 \end{aligned} \tag{1.3}$$

Es gilt $x_1 = 2$ und $x_2 = x_3$. Somit haben wir unendlich viele Lösungen, nämlich $x_1 = 2$, $x_2 = \alpha$ und $x_3 = \alpha$ für jede reelle Zahl α.

Für die gezeigten Beispiele von Gleichungssystem hatten wir gesehen, dass diese entweder eine Lösung, keine Lösung oder unendlich viele Lösungen haben können. Damit wir das Finden von Lösungen für Gleichungssysteme verallgemeinern können, ist der Begriff der **Lösungsmenge** wichtig. Die Lösungsmenge eines Gleichungssystems ist definiert als die Menge aller Lösungen des Gleichungssystems.

1.5 Bestimmung der Lösungsmenge - das Gaussverfahren

Das Ziel dieses Abschnitts ist es ein allgemein gültiges Verfahren zu entwickeln, welches uns für ein lineares Gleichungssystem die Lösungsmenge bestimmt. Die grundlegende Idee wird sein, ein existierendes Gleichungssystem mit bestimmten Operationen so umzuformen, dass die Bestimmung der Lösungsmenge einfach ist. Dieses Verfahren heisst **Gauss'sches Eliminationsverfahren**.

Für die Herleitung des Gaussverfahrens müssen wir zuerst noch den Begriff der **Äquivalenz** zwischen Gleichungssystemen einführen. Zwei Gleichungssystem A und B sind dann äquivalent zueinander, falls die beiden Gleichungssysteme A und B die gleichen Lösungsmengen haben. Mit zwei Operationen lässt sich aus einem bestehenden Gleichungssystem A ein äquivalentes Gleichungssystem B erzeugen.

1. Vertauschen der Reihenfolge der Gleichungen
2. Addition (oder Subtraktion) eines vielfachen einer Gleichung aus dem Gleichungssystem A zur einer anderen Gleichung des Gleichungssystems A.

1.5.1 Vertauschen der Reihenfolge

Als Beispiel sei das folgende lineare Gleichungssystem gegeben.

$$\begin{aligned} x_1 + 2x_2 &= 5 \\ 2x_1 + 3x_2 &= 8 \end{aligned} \tag{1.4}$$

Dieses Gleichungssystem ist äquivalent zum folgenden Gleichungssystem

$$\begin{aligned} 2x_1 + 3x_2 &= 8 \\ x_1 + 2x_2 &= 5 \end{aligned} \tag{1.5}$$

Die Gleichungssysteme sind äquivalent, da beide die Lösungsmenge $L = \{x_1 = 1, x_2 = 2\}$ haben.

1.5.2 Addition eines Vielfachen der einen zur anderen Gleichung

Gegeben sei das folgende Gleichungssystem

$$\begin{aligned} x_1 + 2x_2 &= 5 \\ 2x_1 + 3x_2 &= 8 \end{aligned} \tag{1.6}$$

Lassen wir die erste Gleichung unverändert und ersetzen die zweite Gleichung durch eine neue Gleichung, welche wir erhalten durch Subtraktion des Zweifachen der ersten Gleichung von der zweiten Gleichung, erhalten wir folgendes äquivalentes Gleichungssystem.

$$\begin{aligned} x_1 + 2x_2 &= 5 \\ -x_2 &= -2 \end{aligned} \tag{1.7}$$

Aufgrund der zweiten Gleichung im Gleichungssystem (1.7) sehen wir sofort, dass $x_2 = 2$ ein Teil der Lösungsmenge des Gleichungssystems sein muss. Setzen wir $x_2 = 2$ in der ersten Gleichung von (1.7) ein, dann folgt $x_1 = 1$ und die Lösungsmenge $L = \{x_1 = 1, x_2 = 2\}$ ist komplett. Wir haben also die Lösungsmenge des Gleichungssystems (1.7) gefunden. Wir können das Gleichungssystem (1.6) aus (1.7) herleiten, indem wir die erste Gleichung unverändert lassen und die zweite Gleichung durch eine neue Gleichung ersetzen, welche wir als Summe der zweiten Gleichung aus (1.7) plus das doppelte der ersten Gleichung aus (1.7) berechnen. Das heisst aber, dass die Lösungsmenge $L = \{x_1 = 1, x_2 = 2\}$, welche wir für (1.7) gefunden hatten, auch die Lösungsmenge für (1.6) ist. Somit sind die Gleichungssystem (1.6) und (1.7) äquivalent.

1.5.3 Verfahren zur Bestimmung der Lösungsmenge

Bei der Herleitung eines Verfahrens zur Bestimmung der Lösungsmenge eines Gleichungssystems, stellt sich die Frage, wann ist ein lineares Gleichungssystem einfach zu lösen? Ein Hinweis für eine mögliche Antwort liefert uns der Vergleich der äquivalenten Gleichungssysteme (1.6) und (1.7) im Bezug auf die Bestimmung der Lösungsmenge. Beim Gleichungssystem (1.7) konnten wir in der zweiten Gleichung den Lösungswert für die Variable x_2 sofort ablesen. Die Lösung für x_1 erhielten wir dann einfach durch Einsetzen des Wertes für x_2. Diese Vorgehensweise wie beim Gleichunssystem (1.7) ist beim Gleichungssystem (1.6) nicht möglich.

Die Einfachheit der Bestimmung der Lösungsmenge für das Gleichungssystem (1.7) basiert auf seiner speziellen Struktur. Der Fachterminus für diese Struktur lautet **Dreiecksgestalt**. Wir können aus unseren Beobachtungen also ableiten, dass Gleichungssystem mit einer Dreiecksgestalt einfach zu lösen sind.

Das **Gaussverfahren** zur Bestimmung der Lösungsmenge eines linearen Gleichungssystems besteht nun darin, die besprochenen Operationen zur Transformation von äquivalenten Gleichungssystem geschickt einzusetzen, damit ein gegebenes Gleichungssystem mit beliebiger Struktur in eine Dreiecksgestalt zu verwandeln. Dann können die Lösungswerte der unbekannten Variablen vom der letzten Gleichung her bestimmt und durch Rückwärtseinsetzen zur Berechnung der weiteren Variablen verwendet werden.

Wir wollen das Gaussverfahren an einem allgemeinen Gleichungssystem mit $m = 3$ Gleichungen und $n = 3$ Unbekannten demonstrieren. Gegeben ist also das allgemeine lineare Gleichungssystem

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned} \tag{1.8}$$

Zur Veranschaulichung der Schritte im Gaussverfahren wandeln wir das Gleichungssystem (1.8) in das folgende tabellarische Eliminationsschema um.

a_{11}	a_{12}	a_{13}	b_1
a_{21}	a_{22}	a_{23}	b_2
a_{31}	a_{32}	a_{33}	b_3

Wir nehmen an, dass $a_{11} \neq 0$. Trifft diese Annahme nicht zu wird die Reihenfolge der Gleichungen im Gleichungssystem so geändert, dass $a_{11} \neq 0$ ist. Nun bilden wir ein äquivalentes Gleichungssystem, indem wir von der zweiten Zeile des Ausgangsschemas das a_{21}/a_{11}-fache der ersten Zeile subtrahieren und von der dritten das a_{31}/a_{11}-fache der ersten Zeile. Dadurch stehen im neuen Gleichungssystem an der ersten Stelle der zweiten und der dritten Zeile eine Null. Die anderen Koeffizienten werden mit dem oberen Index (2) bezeichnet.

a_{11}	a_{12}	a_{13}	b_1
0	$a_{22}^{(2)}$	$a_{23}^{(2)}$	$b_2^{(2)}$
0	$a_{32}^{(2)}$	$a_{33}^{(2)}$	$b_3^{(2)}$

Im zweiten Schritt des Verfahrens wird das gleiche wie im Schritt 1 wiederholt mit dem Untersystem, welches durch die oberen Indices (2) bezeichnet ist. Dabei wird als erstes wieder angenommen, dass $a_{22}^{(2)} \neq 0$ gilt. Von der dritten Gleichung wird das $a_{32}^{(2)}/a_{22}^{(2)}$-fache der zweiten Gleichung abgezogen. Daraus entsteht das folgende dritte Eliminationsschema, welches schon die gewünschte Dreiecksgestalt aufweist.

a_{11}	a_{12}	a_{13}	b_1
0	$a_{22}^{(2)}$	$a_{23}^{(2)}$	$b_2^{(2)}$
0	0	$a_{33}^{(3)}$	$b_3^{(3)}$

Aus diesem Schema in Dreiecksgestalt können wir die Lösung für x_3 bestimmen, da aufgrund der letzten Zeile gilt, dass

$$a_{33}^{(3)} * x_3 = b_3^{(3)}$$

Somit ist

$$x_3 = b_3^{(3)}/a_{33}^{(3)}$$

Das ist die Lösung für die unbekannte Variable x_3, da der Ausdruck $b_3^{(3)}/a_{33}^{(3)}$ nicht mehr von einer der Unbekannten x_1, x_2 oder x_3 abhängig ist. Setzen wir die Läsung für x_3 in die zweite Zeile des Schemas in Dreiecksgestalt ein, dann folgt

$$x_2 = \frac{b_2^{(2)} - a_{23}^{(2)} * b_3^{(3)}/a_{33}^{(3)}}{a_{22}^{(2)}}$$

Wir haben jetzt Lösungen für die Unbekannten x_2 und x_3. Diese setzen wir in die erste Gleichung des Dreicks-Schemas ein und erhalten

$$x_1 = \frac{b_1 - a_{12} * ((b_2^{(2)} - a_{23}^{(2)} * b_3^{(3)}/a_{33}^{(3)})/a_{22}^{(2)}) - a_{13} * (b_3^{(3)}/a_{33}^{(3)})}{a_{11}}$$

1.6 Matrix- und Vektorschreibweise

Bisher haben wir Gleichungssysteme explizit als Liste von Gleichungen notiert. Für das allgemeine lineare Gleichungssystem kennen wir das bereits schon aus der Beschreibung des Gaussverfahrens.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3 \end{aligned} \tag{1.9}$$

Für kleine Gleichungssystem wie (1.9) ist diese Notation befriedigend. Für praktische Anwendungen, wie z. Bsp. die Zuchtwertschätzung, wo die Anzahl Gleichungen in einem Gleichungssystem in der Grössenordnung von 10^6 liegt, ist die Notation, wie sie in (1.9) verwendet wurde, ungeeignet. Somit brauchen wir für grosse Gleichungssystem eine effizientere Notation. Eine mögliche solche Notation ist die Schreibweise mit Matrizen und Vektoren.

Wir definieren die Matrix A und die Vektoren x und b, wie folgt.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Die Matrix A und Vektoren x und b lassen sich nun zur Matrix-Vektorschreibweise des Gleichungssystems (1.9) kombinieren.

$$A \cdot x = b \tag{1.10}$$

Mit $\cdot$ ist die im Kapitel zu den Matrizen eingeführte Matrixmultiplikation gemeint. Aus der in (1.10) ist die Information zur Anzahl Gleichungen und zur Anzahl Unbekannten im Gleichungssystem verloren gegangen. Diese Information wird erst durch die Definitionen von A, x und b sichtbar. Die Matrix A wird als Koeffizienten-Matrix, der Vektor x als Vektor der Unbekannten und der Vektor b als rechte Handseite bezeichnet.

Was hier wie ein Nachteil der Matrix-Vektorschreibweise erscheinen mag, ist effektiv der grosse Vorteil dieser Notation. Denn ob wir über ein kleines Gleichungssystem wie in (1.9) oder über ein System mit 10^6 Gleichungen sprechen wollen, wir können immer die Notation in (1.10) verwenden.

Des weiteren können wir auch alle Eigenschaften der Matrizenrechnung zur Lösung des Gleichungssystems verwenden. Als Beispiel können wir unter Verwendung der Inversen A^{-1} der Koeffzienten-Matrix, die Lösung für x berechnen als

$$x = A^{-1} \cdot b$$

Die Inverse A^{-1} der Koeffizienten-Matrix ist zunächst unbekannt. Wir werden aber zu einem späteren Zeitpunkt noch sehen, dass uns auch bei der Berechnng von A^{-1} das Gaussverfahren sehr nützlich sein kann.

Chapter 2

Einführung in R

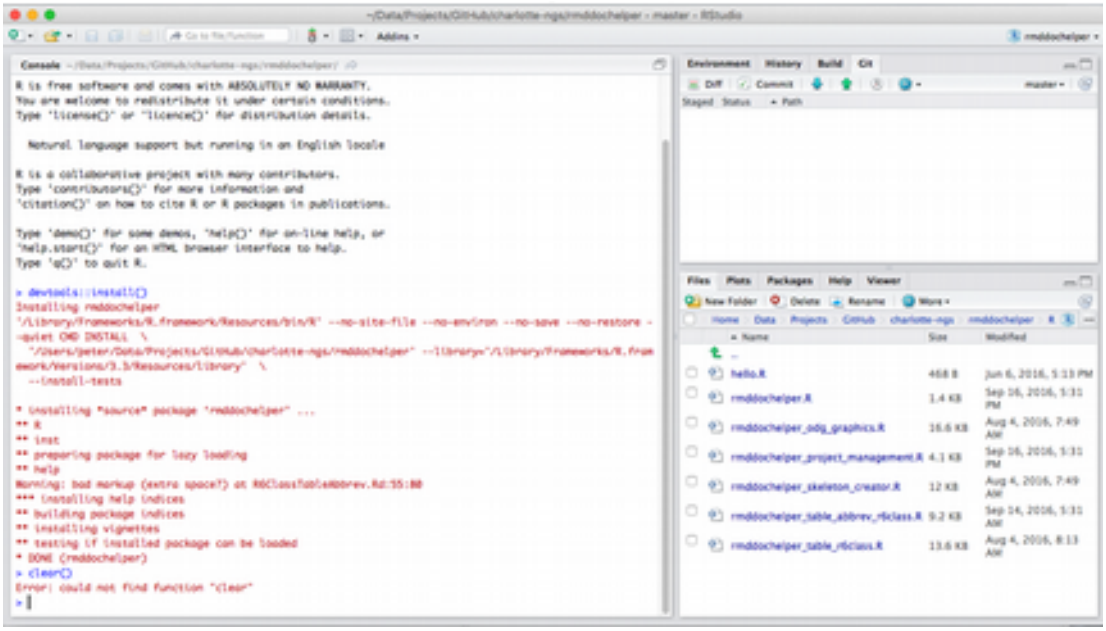
R (https://www.r-project.org/) ist ein sehr populäres System im Bereich der Datenanalyse. Ursprünglich wurde R von den Statistikern Ross Ihaka und Robert Gentlemen kreiert. Da R über die eigene Programmiersprache erweiterbar ist, wird es aktuell in sehr verschiedenen Gebieten eingesetzt. Das System wird als Open Source vertrieben und kann gratis für die gängigen Betriebssystem heruntergeladen werden. Eine Vielzahl von Dokumentationen zu R ist unter https://cran.r-project.org/manuals.html erhältlich. Unter https://cran.r-project.org/other-docs.html sind auch Anleitungen in Deutsch und vielen weiteren Sprachen erhältlich.

Dieses Dokument basiert auf einigen der oben genannten Quellen und versucht die wichtigsten Punkte von R für diese Vorlesung zusammenzufassen.

2.1 Voraussetzung für die Verwendung von R

R muss zuerst heruntergeladen und dann installiert werden. Unter den sogenannten `Comprehensive R Archive Network` (CRAN)-Mirror Webseiten ist R als ausführbares Programm für die Betriebssystem `Windows`, `Mac OSX` und `Linux` erhältlich.

Zusätzlich zu R empfielt es sich, die Entwicklungsumgebung RStudio (https://www.rstudio.com/) zu verwenden.



2.2 Interaktiver Modus

R kann wie ein Taschenrechner verwendet werden. Man spricht dann auch vom so genannten **interaktiven** Modus, d.h. die/der BenutzerIn gibt einen Befehl ein und bekommt eine Antwort zurück. In Rstudio werden die Befehle in das Fenster, welches mit dem Tab namens `Console` überschrieben ist, eingegeben. Beispiele für mögliche Eingaben im interaktiven Modus sind

```
43-15
```

```
## [1] 28
```

```
76/8
```

```
## [1] 9.5
```

R befolgt die in der Arithmetik üblichen Rechenregeln, so gilt beispielsweise, dass Klammern vor Punkt vor Strich

```
65 - 18 / 3
```

```
## [1] 59
```

```
(65 - 18) / 3
```

```
## [1] 15.66667
```

2.2.1 Arithmetische Operationen

Die folgende Tabelle enthält die Liste mit den in R verwendeten arithmetischen Operationen

Operator	Operation
$+$	Addition
$-$	Subtraktion
$*$	Multiplikation
$/$	Division
^	Potenz
e	Zehnerpotenz

2.2.2 Logische Operationen

Abgesehen von arithmetischen Rechenoperationen lassen sich auch logische Operationen durchführen. Mithilfe dieser Operationen lassen sich Vergleiche machen oder Beziehungen überprüfen. Das Resultat eines solchen Vergleichs ist immer entweder `TRUE` (wahr) oder `FALSE` (falsch). (Im Abschnitt zu den Datentypen werden wir sehen, dass es für die Vergleichsresultaten einen speziellen Datentyp - den `BOOLEAN` Datentyp - gibt.) Die folgende Liste zeigt die Vergleichsoperatoren in R.

Operator	Operation
$==$	ist gleich
$!=$	ist ungleich
$>$	ist grösser
$<=$	ist kleiner gleich
&	und
|	oder

Die folgenden Beispiele zeigen, wie logische Vergleiche ausgewertet werden

```
3 == 3
```

```
## [1] TRUE
```

```
3 == 4
```

```
## [1] FALSE
```

```
3 != 4
```

```
## [1] TRUE
```

```
3 <= 4 & 3 == 3
```

```
## [1] TRUE
```

```
3 <= 4 | 3 == 4
```

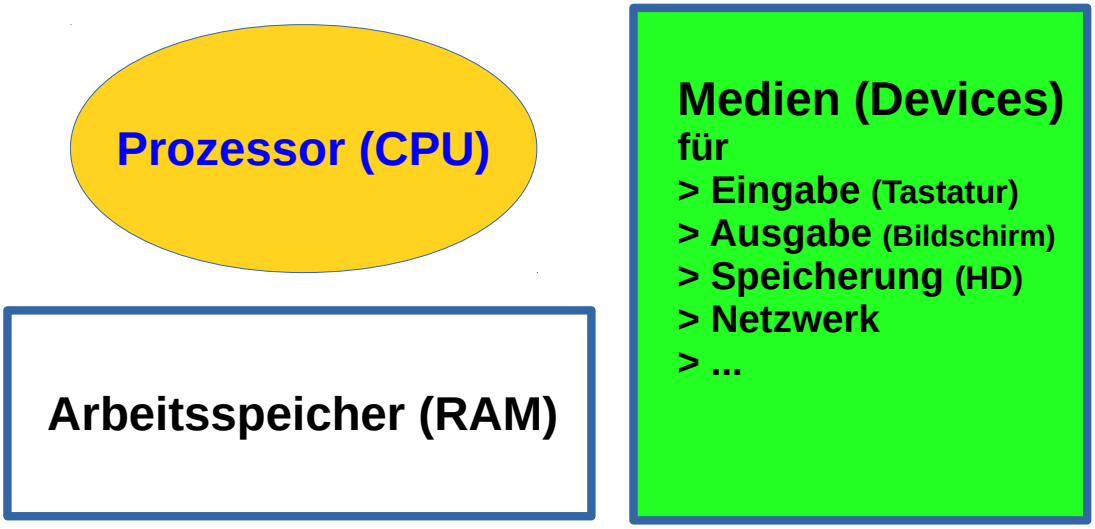
```
## [1] TRUE
```

```
(3 <= 4 | 3 == 3) & 3 == 4
```

```
## [1] FALSE
```

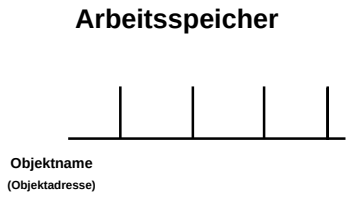
2.3 Variablen - Objekte

Die Begriffe `Variablen` und `Objekte` werden hier als Synonyme verwendet. Das Rechnen im Taschenrechner-Modus ist sicher sehr nützlich, aber wir wollen auch in der Lage sein, bestimmte Grössen an speziellen Orten im Speicher abzulegen. Dazu verwenden wir Variablen. Damit wir besser verstehen, was bei der Verwendung bei Variablen passiert, schauen wir uns das so genannte `Von Neumann`-Modell eines Computers an.



Von Neumann Computer-Architektur

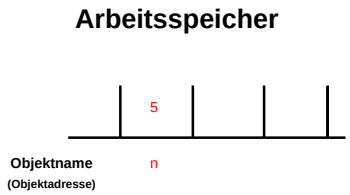
Schauen wir uns den Arbeitsspeicher etwas genauer an, dann können wir uns den als eine Art Setzkasten, in dem man verschiedene Objekte ablegen kann, vorstellen.



Wenn wir in der Console von Rstudio als Beispiel der Variablen `n` den Wert 5 zuweisen, dann wird das mit folgender Anweisung gemacht

```
n <- 5
```

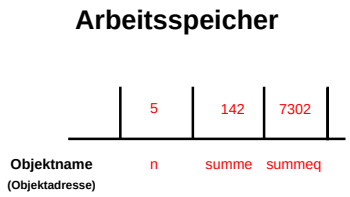
Das folgende Diagramm zeigt, wie sich durch diese Anweisung der Arbeitsspeicher verändert.



Berechnung und Zuweisung zu Variablen können kombiniert werden.

```
summe <- 15 + 9 + 8 + 34 + 76
summeq <- 15^2 + 9^2 + 8^2 + 34^2 + 76^2
```

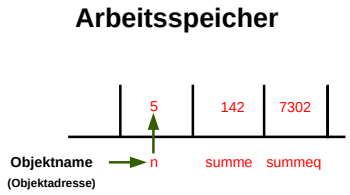
Die obige Berechnung und Zuweisung ist im nachfolgenden Diagramm dargestellt.



Der Zugriff auf die Werte, welche unter einer Variablen abgelegt sind, erfolgt durch die Eingabe des Variablennamens an der Console.

```
n
```

```
## [1] 5
```



Bezüglich der Namengebung von Variablen müssen einige Regeln eingehalten werden. Variablennamen können Buchstaben, Zahlen und Zeichen wie "-""," "_" oder "." enthalten. Sie sollen aber nicht mit einer Zahl beginnen. Es wird empfohlen Namen von schon existierenden Funktionen nicht als Variablen zu verwenden. Allgemeine Hinweise zu Style, Namen und Notationen sind unter http://r-pkgs.had.co.nz/style.html erhältlich.

Mit Variablen kann wie mit Zahlen gerechnet werden. Für die Berechnung werden die im Arbeitsspeicher unter dem entsprechenden Variablennamen abgelegten Werte verwendet. Wollen wir als Beispiel aus den Werten, welche wir unter den Variablennamen `summe` und `summeq` abgelegt haben, den Mittelwert und die Standardabweichung berechnen, dann sieht das wie folgt aus.

```
m <- summe / n
s <- sqrt((summeq - summe^2/n)/(n-1))
```

Bei einer normalen Zuweisung wird kein Output generiert. Für die Anzeige der Resultate muss der Variablenname eingeben oder die Funktion `print()` verwendet werden.

```
m
```

```
## [1] 28.4
```

```
print(s)
```

```
## [1] 28.58846
```

2.4 Datentypen

Bis anhin hatten wir in R mit Zahlen gerechnet. Abgesehen von Zahlen gibt es noch die folgenden Datentypen in R.

Datentyp	Beschreibung
numeric	reelle Zahlen
integer	ganze Zahlen
complex	Quadratwurzel aus negativen Zahlen
character	Buchstaben, Zeichen
factor	Datentyp für lineare Modelle

2.4.1 Wichtige Punkte zu Datentypen

R kennt keine strenge Prüfung von Datentypen, d.h. R erlaubt es der gleichen Variablen einmal eine Zahl und dann ein Character zuzuweisen. Falls nötig und möglich macht R eine automatische Umwandlung zwischen Datentypen. Diese Umwandlung wird als **coersion** bezeichnet.

Hilfreiche Funktionen im Zusammenhang mit Datentypen sind:

- `class()` gibt den Typ eines Objekts zurück
- `is.<data.type>()` prüft, ob Objekt vom Datentype `<data.type>` ist. Als Beispiel überprüft `is.integer(5)`, ob 5 eine ganze Zahl ist.
- `as.<data.type>()` kann für explizite Umwandlungen verwendet werden. Zum Beispiel wandelt `as.character(12)` die Zahl 12 in den String “12” um.

2.4.2 Vektoren

Prinzipiell behandelt R jede Variable oder jedes Objekt als einen Vektor. Das ist für den Gebrauch in dieser Vorlesung nicht von grosser Bedeutung. Vektoren als Sequenzen von Komponenten des gleichen Datentyps sind für uns viel wichtiger. Diese werden mit der Funktion `vector()` erzeugt und mit der Funktion `c()` erweitert.

```
vecNum <- vector(mode = "numeric", length = 2)
vecNum[1] <- 5
vecNum[2] <- -4
print(vecNum)
```

```
## [1]  5 -4
```

Analog dazu kann eine Vektor auch direkt ohne den Umweg über die Funktion `vector()` direkt erzeugt werden.

```
vecNum <- c(5,-4)
print(vecNum)
```

```
## [1]  5 -4
```

Vektoren können mit der Funktion `c()` auch erweitert werden.

```
vecNum <- c(vecNum, 43,-2)
print(vecNum)
```

```
## [1]  5 -4 43 -2
```

```
vecChar <- c("aa", "ba")
vecChar <- c(vecChar, vecNum)
print(vecChar)
```

```
## [1] "aa" "ba" "5"  "-4" "43" "-2"
```

Das zweite Beispiel, in welchem wir den Vektor `vecChar` um den Vektor `vecNum` erweiterten, zeigt, dass bei der Erweiterung eines Character-Vektors um einen Zahlen-Vektor, die Zahlen automatisch in Strings umgewandelt werden. Umgekehrt, ist es nicht möglich einen Vektor von Zahlen um einen Character-Vektor zu erweitern, da Character nicht eindeutig in Zahlen verwandelt werden können.

Eine wichtige Eigenschaft eines Vektors ist seine **Länge**. Hier hat Länge nicht eine geometrische Bedeutung, sondern hier ist Länge die Anzahl Komponenten im Vektor gemeint. Die Funktion `length()` ermittelt die Anzahl Elemente in einem Vektor.

```
length(vecChar)
```

```
## [1] 6
```

Logische Vergleiche können direkt auf Vektoren angewendet werden. Als Resultat erhalten wir einen Vektor mit Boolschen Komponenten der gleichen Länge, wie der ursprüngliche Vektor. Als Beispiel können wir für alle Komponenten eines numerischen Vektors testen, ob die Komponenten grösser als ein Bestimmter Wert sind.

```
vecNum > 5
```

```
## [1] FALSE FALSE  TRUE FALSE
```

Die **arithmetischen Operationen** mit Vektoren werden alle komponenten-weise ausgeführt.

```
x <- c(3,5,13,-2)
y <- c(2,6,-3,19)
x+y
```

```
## [1]  5 11 10 17
```

```
x-y
```

```
## [1]   1  -1  16 -21
```

```
x*y
```

```
## [1]   6  30 -39 -38
```

```
x/y
```

```
## [1]  1.5000000  0.8333333 -4.3333333 -0.1052632
```

Das Skalarprodukt zweier Vektoren berechnen wir mit der Funktion `crossprod()`

```
crossprod(x,y)
```

```
##      [,1]
## [1,]  -41
```

Der **Zugriff auf ein bestimmtes Elementes** i eines Vektors x geschieht mit dem Ausdruck $x[i]$.

```
x[2]
```

```
## [1] 5
```

Setzen wir den Index i in $x[i]$ auf einen Wert $i < 0$, dann erhalten wir den Vektor, in welchem das Element i fehlt.

```
x[-3]
```

```
## [1]  3  5 -2
```

Mit einem Bereich von Indices kann ein Teil des Vektors angesprochen werden.

```
x[2:4]
```

```
## [1]  5 13 -2
```

2.4.3 Matrizen

Matrizen sind ihrem mathematischen Vorbild nachempfunden und somit in Zeilen und Kolonnen organisiert. Alle Komponenten einer Matrix müssen vom gleichen Datentyp sein. Eine Matrix in R wird mit der Funktion `matrix()` erstellt.

```
matA <- matrix(c(5,3,4,-6,3,76), nrow = 2, ncol = 3, byrow = TRUE)
print(matA)
```

```
##      [,1] [,2] [,3]
## [1,]    5    3    4
## [2,]   -6    3   76
```

Falls die Matrix zeilenweise aufgefüllt werden soll, dann ist die Option `byrow = TRUE` wichtig. Andernfalls wird die Matrix kolonnenweise aufgefüllt.

Eine grundlegende Eigenschaft einer Matrix ist ihre Dimension. Sie entspricht der Anzahl Zeilen und Kolonnen der Matrix und wird mit der Funktion `dim()` bestimmt.

```
dim(matA)
```

```
## [1] 2 3
```

Der **Zugriff auf Elemente** einer Matrix erfolgt analog zum Zugriff bei den Vektoren. Aber für die Spezifikation eines einzelenen Elementes braucht es bei den Matrizen zwei Indices.

```
matA[2,1]
```

```
## [1] -6
```

Es ist aber auch möglich eine ganze Zeile oder eine ganze Spalte zu selektieren. Das Resultat ist in beiden Fällen ein Vektor.

```
matA[1,]
```

```
## [1] 5 3 4
```

Was etwas verwirrend aussieht ist, dass beim Zugriff auf eine Spalte, scheinbar auch eine Zeile ausgegeben wird, das hängt aber damit zusammen, dass das Resultat ein Vektor ist und Vektoren werden immer auf einer Zeile ausgegeben.

```
matA[,2]
```

```
## [1] 3 3
```

Bestehende Matrizen können auch **erweitert** werden. Dies kann auf zwei Arten passieren. Entweder stapeln wir Matrizen aufeinander oder wir stellen sie nebeneinander. Die beiden Fälle sind in den nachfolgenden Statements gezeigt.

```
matB <- matrix(c(3,-1,90,1,1,4), nrow = 2, ncol = 3, byrow = TRUE)
rbind(matA, matB)
```

```
##      [,1] [,2] [,3]
## [1,]    5    3    4
## [2,]   -6    3   76
## [3,]    3   -1   90
## [4,]    1    1    4
```

Alternativ können wir die Matrizen auch in Kolonnenrichtung erweitern.

```
cbind(matA, matB)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    5    3    4    3   -1   90
## [2,]   -6    3   76    1    1    4
```

Die aus der lineare Algebra bekannten **Operationen** können wir auch hier in R anwenden. Als erstes können wir eine gegebene Matrix **transponieren**.

```
t(matA)
```

```
##      [,1] [,2]
## [1,]    5   -6
## [2,]    3    3
## [3,]    4   76
```

Wir können also einfach überprüfen, dass die Transponierte Matrix der Transponierten wieder der ursprünglichen Matrix entspricht.

```
t(t(matA))
```

```
##      [,1] [,2] [,3]
## [1,]    5    3    4
## [2,]   -6    3   76
```

Die **arithmetischen** Rechenoperationen $+$, $-$, $*$ und $/$ werden alle Element-weise ausgeführt.

```
matA + matB
```

```
##      [,1] [,2] [,3]
## [1,]    8    2   94
## [2,]   -5    4   80
```

```
matA - matB
```

```
##      [,1] [,2] [,3]
## [1,]    2    4  -86
## [2,]   -7    2   72
```

```
matA * matB
```

```
##      [,1] [,2] [,3]
## [1,]   15   -3  360
## [2,]   -6    3  304
```

```
matA / matB
```

```
##           [,1] [,2]        [,3]
## [1,]  1.666667   -3  0.04444444
## [2,] -6.000000    3 19.00000000
```

Die **Matrixmultiplikation**, welche wir aus der linearen Algebra kennen, muss entweder mit dem speziellen Operator `%*%` oder mit der Funktion `crossprod()` berechnet werden.

```
matA %*% t(matB)
```

```
##      [,1] [,2]
## [1,]  372   24
## [2,] 6819  301
```

```
crossprod(matA, matB)
```

```
##      [,1] [,2] [,3]
## [1,]    9  -11  426
## [2,]   12    0  282
## [3,]   88   72  664
```

Wir sehen hier dass die beiden Matrixmultiplikationen `matA %*% t(matB)` und `crossprod(matA, matB)` nicht das gleiche Resultat ergeben. Aus der Linearen Algebra wissen wir auch, dass die Matrizen `matA` und `matB` so nicht kompatibel sind für die Matrixmultiplikation. Die Funktion `crossprod()` transponiert automatisch die erste Matrix und somit wird die folgende Berechnung durchgeführt.

```
t(matA) %*% matB
```

```
##      [,1] [,2] [,3]
## [1,]    9  -11  426
## [2,]   12    0  282
## [3,]   88   72  664
```

Die **Inverse** einer Matrix erhalten wir als Resultat der Funktion `solve()`.

```
matC <- matA %*% t(matB)
(matCinv <- solve(matC))
```

```
##              [,1]          [,2]
## [1,] -0.005823853  0.0004643603
## [2,]  0.131936383 -0.0071975853
```

Als Kontrolle berechnen wir das Produkt der Inversen und der ursprünglichen Matrix und müssen als Resultat die Einheitsmatrix bekommen.

```
matCinv %*% matC
```

```
##              [,1]          [,2]
## [1,] 1.000000e+00 -2.775558e-17
## [2,] 7.105427e-15  1.000000e+00
```

2.4.4 Listen

In Listen kann man Sammlungen von Objekten, welche nicht den gleichen Datentyp haben, abspeichern. Die Definition einer Liste erfolgt über die Angabe von Schlüssel-Werte-Paaren. Was das bedeutet, wird im folgend Beispiel gezeigt.

```
lstA <- list(nZahlen = c(5,-2,7),
             sNamen = c("Fred","Mary"),
             lBool = c(FALSE,TRUE))
```

Der **Zugriff** auf die Elemente mit Indices oder die Angabe von Schlüsselnamen gibt als Resultat wieder eine Liste zurück mit dem entsprechenden Schlüssel-Werte-Paar, welches selektiert wurde.

```
lstA[2]
```

```
## $sNamen
## [1] "Fred" "Mary"
```

Ein Element kann aber auch mit einem Namen selektiert werden.

```
lstA["nZahlen"]
```

```
## $nZahlen
## [1]  5 -2  7
```

Die Liste aller Namen erhalten wir mit der Funktion `names()`.

```
names(lstA)
```

```
## [1] "nZahlen" "sNamen"  "lBool"
```

Wollen wir die Elemente einer Liste so selektieren, dass ein Vektor als Resultat zurückkommt, dann müssen wir doppelte Klammern oder den `$`-Operator verwenden.

```
lstA[[3]]
```

```
## [1] FALSE  TRUE
```

```
lstA[["lBool"]]
```

```
## [1] FALSE  TRUE
```

```
lstA$lBool
```

```
## [1] FALSE  TRUE
```

2.4.5 Dataframes

Dataframes sind spezielle Listen in R. Diese erhalten wir als Resultat, wenn wir Daten von Dateien einlesen (siehe nächster Abschnitt). Technisch gesehen sind Dataframes eine Mischung aus Listen und Matrizen. Dies wird klar, wenn wir uns den möglichen Zugriff auf Elemente eines Dataframes anschauen. Zuerst müssen wir aber ein Dataframe erzeugen. Dies geschieht an dieser Stelle mit der Funktion `data.frame()`. Die Option `stringsAsFactors = FALSE` muss angegeben werden, da sonst alle Strings in Faktoren umgewandelt werden.

```
dfA <- data.frame(nZahl = c(-2,15),
                  sZeichen = c("Alice","Bob"),
                  bWahr = c(FALSE,FALSE),
                  stringsAsFactors = FALSE)
```

Der Zugriff auf die einzelnen Elemente kann nun wie bei einer Matrix über die Angaben von Zeilen- und Kolonnenindices sein, oder wie bei einer Liste mit der Angabe eines Schlüsselnamens.

```
dfA[2,1]
```

```
## [1] 15
```

Mit der Angabe eines Schlüsselnamens erhalten wir alle Werte zum entsprechenden Schlüssel.

```
dfA$sZeichen
```

```
## [1] "Alice" "Bob"
```

2.5 Einlesen von Daten

Eine Verwendungsart von R ist die statistische Analyse von Daten. Zu diesem Zweck müssen wir die Daten zuerst einlesen. Erst dann können wir sie analysieren. Die wichtigste Funktion um Daten in R einzulesen, heisst `read.table()`. Mit dieser Funktion können Daten aus Files, welche Tabellen-artig organisiert sind, eingelesen werden. Haben die einzulesenden Daten ein spezifischers Format, so wie zum Beispiel `Comma Separated Values` (CSV), so gibt es spezialisierte Funktionen, wie `read.csv2()`, welche Daten im CSV-Format einlesen können. Ein Beipiel dafür sei nachfolgend gezeigt. Die Option `file = "csv/br_gew.csv"` gibt an, wo im Dateisystem das File mit den Daten zu finden ist.

```
dfBrGew <- read.csv2(file = "csv/br_gew.csv")
dim(dfBrGew)
```

```
## [1] 10  2
```

Die Funktion `dim()` gibt die Dimension der eingelesenen Daten. Dies ist eine gute Kontrolle, ob der Einleseprozess auch wirklich funktioniert hat. Als Resultat gibt die Funktion `read.csv2()` ein Dataframe zurück. Die Funktion `head()` liefert die ersten paar Zeilen des eingelesenen Dataframes.

```
head(dfBrGew)
```

```
##   Brustumfang Gewicht
## 1         176     471
## 2         177     463
## 3         178     481
## 4         179     470
## 5         179     496
## 6         180     491
```

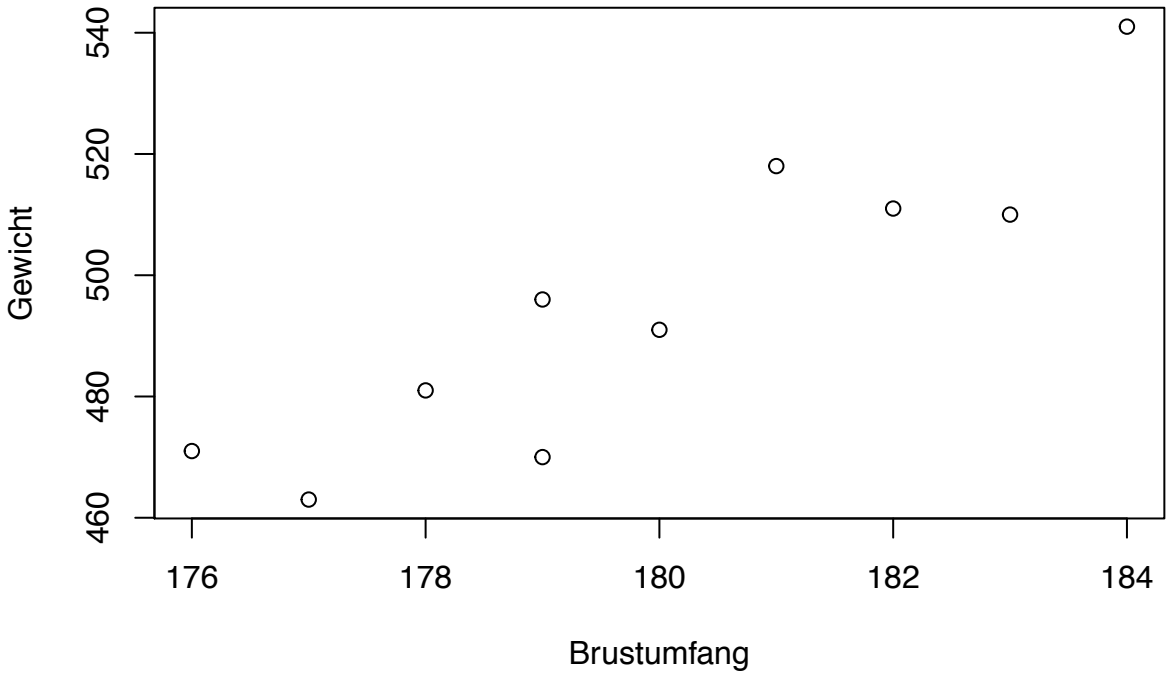
Die umgekehrte Vorgang des Lesens von Daten, das **Schreiben von Daten** in Dateien, kann je nachdem auch wichtig sein. Dazu gibt es die zu `read.table()` analogen Funktionen namens `write.table()`. Sollen die Daten im CSV-Format geschrieben werden dann können wir das mit `write.csv2()` tun. Für den Output von Daten, welche nicht Tabellen-artig organisiert sind, kann auch die Funktion `cat()` verwendet werden.

2.6 Plots und Diagramme

Sobald die Daten in R eingelesen sind, ist ein erster Schritt häufig eine Beschreibung der Daten mithilfe von graphischen Hilfsmitteln. R hat eine grosse Auswahl an Möglichkeiten für die graphische Darstellung von Daten. Hier sind nur die einfachsten erwähnt.

Die Funktion `plot()` kann verwendet werden, um einfache zwei-dimensionale Darstellungen zu erzeugen. Wenn wir als Beispiel das Gewicht gegen den Brustumfang aus dem Dataframe `dfBrGew` darstellen wollen, dann geschieht das mit folgendem Statement.

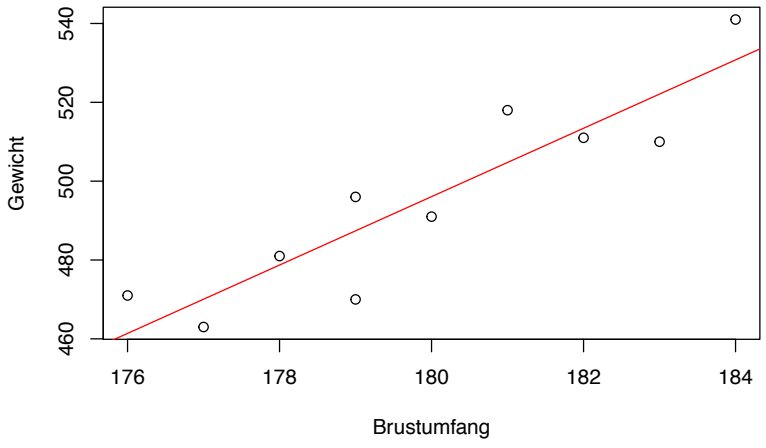
```
plot(dfBrGew$Brustumfang, dfBrGew$Gewicht, xlab = "Brustumfang", ylab = "Gewicht")
```



2.7 Einfaches lineares Modell

Zur Überprüfung eines statistischen Zusammenhangs zwischen den Variablen `Gewicht` und `Brustumfang` können wir ein einfaches lineares Modell anpassen. Wie gut das angepasste lineare Modell zu den Daten passt, können wir visuell durch den Vergleich der Daten zur gefundenen Regressionsgeraden beurteilen. Die Modellanpassung und die Erstellung des Plots werden mit folgenden Statements erzeugt.

```
lmBrGew <- lm(Gewicht ~ Brustumfang, data = dfBrGew)
plot(dfBrGew$Brustumfang, dfBrGew$Gewicht, xlab = "Brustumfang", ylab = "Gewicht")
abline(coef = coefficients(lmBrGew), col = "red")
```



2.8 Erweiterungen

Die Popularität von R ist auch bedingt durch, dass das System als solches fast beliebig erweitert werden kann. Grundsätzlich sind zwei Arten der Erweiterung denkbar.

1. Packages
2. Benutzer-definierte Funktionen

2.8.1 Packages

Unter einem Package versteht man eine Sammlung von Funktionen, welche von Autoren zur Verfügung gestellt werden. Als BenutzerIn von R können diese Packages mit der Funktion `install.packages()` verwendet werden. Wenn wir als Beispiel das Package namens `pedigreemm` verwenden wollen, dann können wir dieses mit folgendem Befehl installieren.

```
install.packages(pkgs = "pedigreemm")
```

Nach einer erfolgreichen Installation von `pedigreemm` können die Funktionen in `pedigreemm` verwendet werden.

2.8.2 Eigene Funktionen

Müssen gewisse Befehle wiederholt und häufig ausgeführt werden, dann empfielt es sich die Befehle in eine BenutzerIn-definierte Funktion zu verpacken. Nach der Definition der Funktion, kann diese genau wie jede andere R-Funktion aufgerufen werden.

Nehmen wir beispielsweise an, wir möchten Temperaturwerte von der Celsius- auf die Fahrenheit-Skala umrechnen, dann können wir das mit einer Funktion machen.

```
celcius_in_fahrenheit <- function(pnCelsius){
  nResultFahrenheit <- 32 + 9/5 * pnCelsius
  return(nResultFahrenheit)
}
```

Die Umrechnung für einen bestimmten Wert geschieht jetzt über folgenden Aufruf.

```
celcius_in_fahrenheit(pnCelsius = 0)
```

```
## [1] 32
```

```
celcius_in_fahrenheit(pnCelsius = 7)
```

```
## [1] 44.6
```

```
celcius_in_fahrenheit(pnCelsius = 25)
```

```
## [1] 77
```

Abkürzungen

Abbreviation	Meaning
CRAN	Comprehensive R Archive Network
CSV	Comma Separated Values

Bibliography