Technical Applications & Data Analytics (WS 2021)

Game Design & Management (B. A.)

© Benjamin Gross

Dieses Dokument enthält das Kursmaterial für die Veranstaltung Technical Applications & Data Analytics im Bachelorstudiengang Game Design & Management. Es behandelt grundlegende Konzepte der Datenanalyse und -verarbeitung, technische Applikationen wie Microsoft Excel (VBA) und R, verschiedene Datenquellen und deren Zusammenhänge, Speicherung von Daten in Datenbanken und Data Warehouses, sowie wichtige Data Mining Methoden einschließlich Korrelationsanalyse, lineare Regressionsanalyse, Basket-Analyse, Cluster-Analyse K-Means und neuronale Netze. Darüber hinaus werden ethische Aspekte der Datenanalyse und Datenschutzfragen diskutiert.

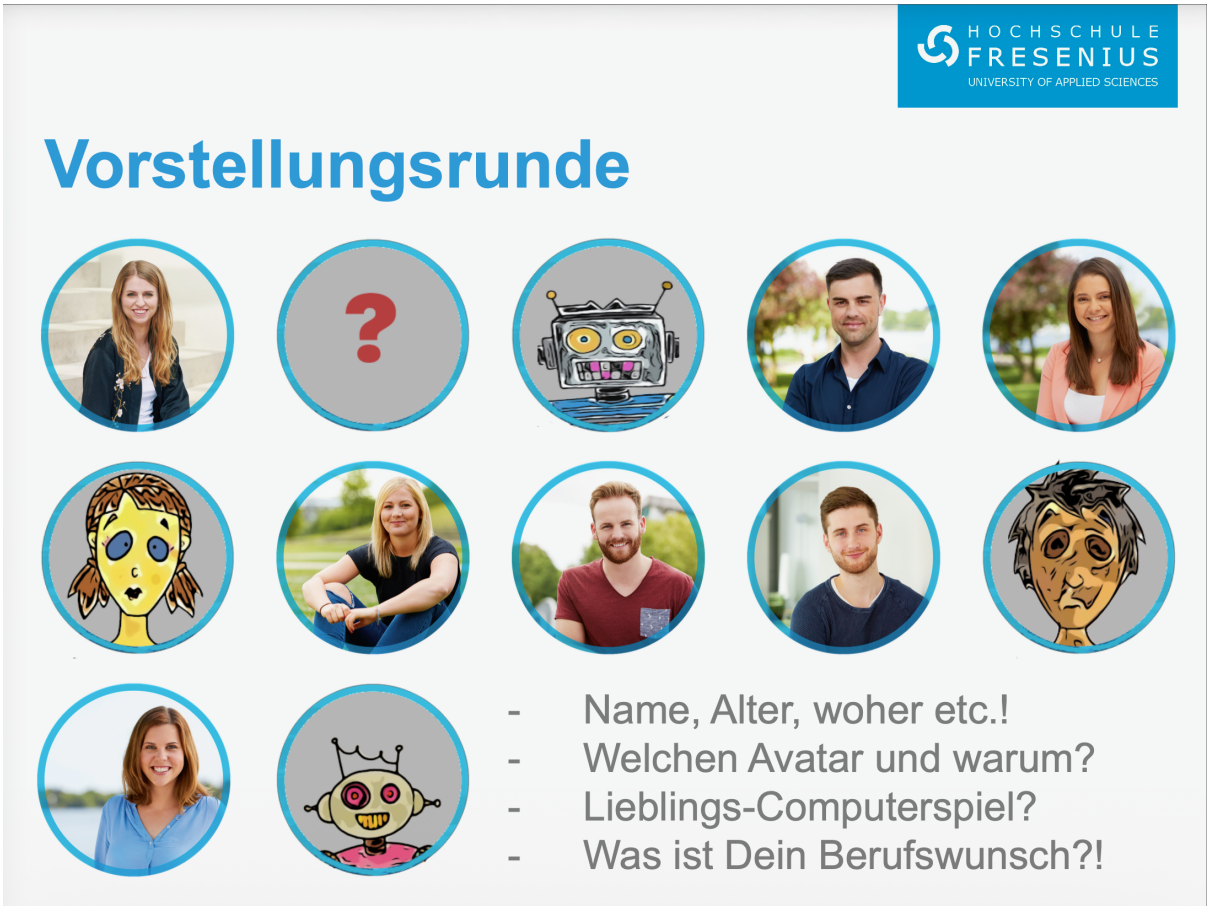
Inhaltsverzeichnis

1 Technical Applications & Data Analytics

Dies ist der Kursinhalt für die Veranstaltung *Technical Applications & Data Analytics* im Bachelorstudiengang *Game Design & Management* an der *Hochschule Fresenius - University of Applied Sciences*.

1.1 Vorstellungsrunde

Abbildung 1: Vorstellung



1.2 Einleitung

1.2.1 Lernziele

oder auch: Wie erreiche ich in Zukunft die „Daten-Weltherrschaft“?!

- Datenarten verarbeiten, erstellen und analysieren
- Vor- und Nachteile von Methoden und Applikationen einschätzen können
- verschiedene Applikation beurteilen und einordnen können
- Open Source Applikationen bedienen können
- theoretische Analyse der Algorithmen und Interpretation der Ergebnisse

1.2.2 Gegenwartsbezug

oder auch: Was hat das mit mir und meiner Zukunft zu tun?!

- Für die Wirtschaft bestehst Du mittlerweile fast ausschließlich aus einer Flut an Daten
- Ethik vs. Wirtschaft
- Vorteile der Technik nutzen (z.B. Medizin)
- Was kann ein datenschutzkonformer Ansatz Gutes bewirken?
- Die Datenverarbeitung wird immer automatisierter, neue Jobs werden geschaffen

1.2.3 Der “Fragebogen”

oder auch: Was sind heute die wirklich wichtigen Fragen?!

Als Excel-Makro Code-Beispiel (VBA) auf Ilias verfügbar! *Die Excel-Makro-Datei (xlsm) wird regelmäßig aktualisiert, da sie sich in der Entwicklung befindet – daher regelmäßig den aktuellen Stand downloaden!*

Der „Fragebogen“ oder auch: Was sind heute die wirklich wichtigen Fragen?!

Abb.: Microsoft Excel „Fragebogen“

1.2.4 Gliederung

Abbildung 2: Gliederung

1.2.5 Grundbegriffe des Technical Applications & Data Management

- Geschichte der Datenanalyse und Datenverarbeitung (incl. Hardware)
- Bedeutung von Daten und deren Digitalisierung
- Ziele der Datenanalyse und Stand der Technik der Datenanalyse
- Einführung in die Begriffe BI, Reporting und Big Data

Definition Statistik: Wissenschaft von Sammlung, Analyse und Interpretation und Kommunikation von Daten mithilfe mathematischer Verfahren zur Entscheidungsfindung.

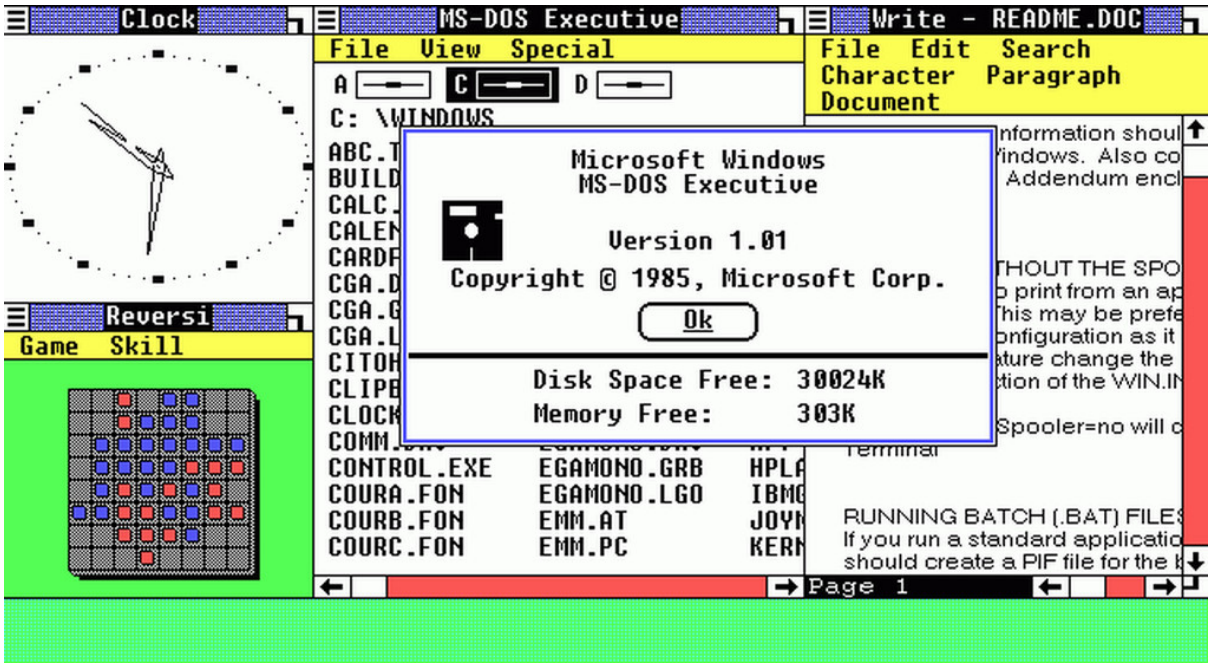
Definition Datenanalyse: Der Prozess des Erhebens von Daten, ihrer Auswertung und Interpretation.

Definition Datenverarbeitung: Der Prozess des Datensammelns und deren Umwandlung in nützliche Informationen. Es werden mit angemessenen Techniken (ohne die gewünschten Daten zu beeinträchtigen) Rohdaten in ein besser lesbares Format (Graphen, Dokumente etc.) umwandelt. Die Daten erhalten so die nötige Form und den erforderlichen Kontext, damit sie von Computern interpretiert und von Mitarbeitern in der gesamten Organisation genutzt werden können (Romeijn, 2016; Tukey, 1962)

1.2.5.1 1.a) Geschichte der Datenanalyse und Datenverarbeitung (incl. Hardware)

- 1934 – 1941: ZUSE Z1, Z2 und Z3 von Konrad Zuse in Relaistechnik (elektromagnetischer Schalter)
- 1945: Idee des Programms (Arbeitsanweisung für den Computer) von John von Neumann
- 1954: Entwicklung der ersten Programmiersprache FORTRAN (Formula Translation; eine Mischung aus wenigen englischen Wörtern, mathematischen Symbolen & Gestaltungsvorschriften)
- 1980: Entwicklung von Mikroprozessoren auf Siliziumbasis
- 1985: Personal Computer von IBM mit Betriebssystem Windows (Abb. 1.a.1)
- 1983: TCP/IP-Protokoll im ARPANET (Vorläufer Internet, aktuell IPv4/6)
- 1989: HTML Entwurf (Hyper Text Markup Language)

Abbildung 3: Abb. 1.a.1: Microsoft Windows 1.01



1.2.5.1.1 Computer Hardwareentwicklung

- Generation 0 (bis 1945)
- Mechanik für Addition
- Generation 1 (1945 – 1955)
- Röhren in Maschinensprache fest programmiert
- Generation 2 (1955 - 1965)

- Transistoren mit Stapelverarbeitung
- Generation 3 (1965 - 1980)
- integrierte Schaltungen (IC), IBM System/360
- Generation 4 (1980 - 1990)
- Personal Computer (PC), IBM PC Modell 5150
- Generation 5 (1990 - ?)
- Laptops, PDAs, Mobiltelefon, etc.

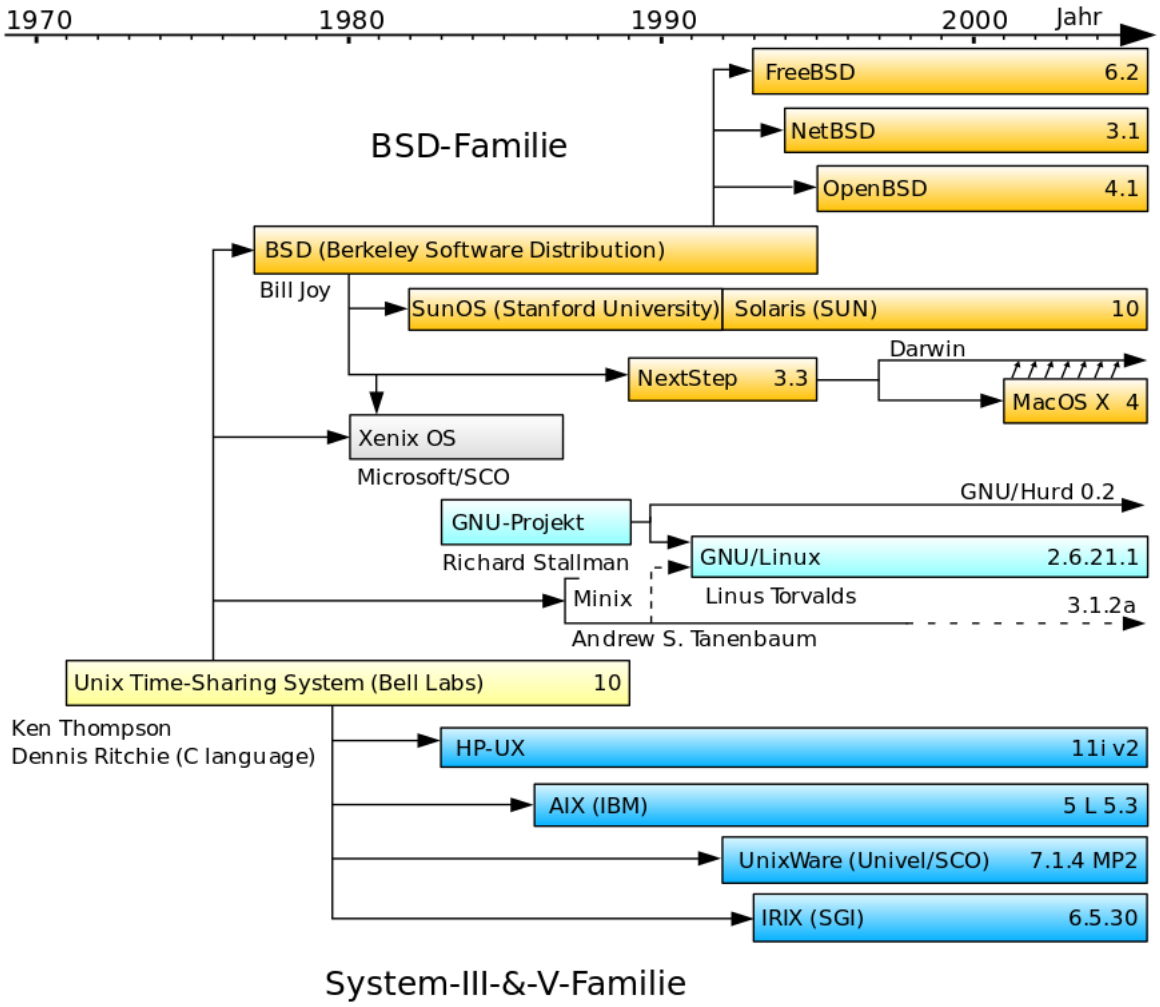
Abbildung 4: Abb. 1.a.2: IBM System/360

1.2.5.1.2 Betriebssysteme und Erscheinungsjahr

- **Unix (1969)**
- **MS-DOS (1981)**
- **Windows (1985)**

- **Linux (1991)**
  - Distributionen (https://en.wikipedia.org/wiki/List_of_Linux_distributions)
    - Debian (1993)
    - Red Hat Linux (1994)
    - Ubuntu (2004)
- **MacOS (2001)**

Abbildung 5: Abb. 1.a.2: Betriebssystem Unix Timeline

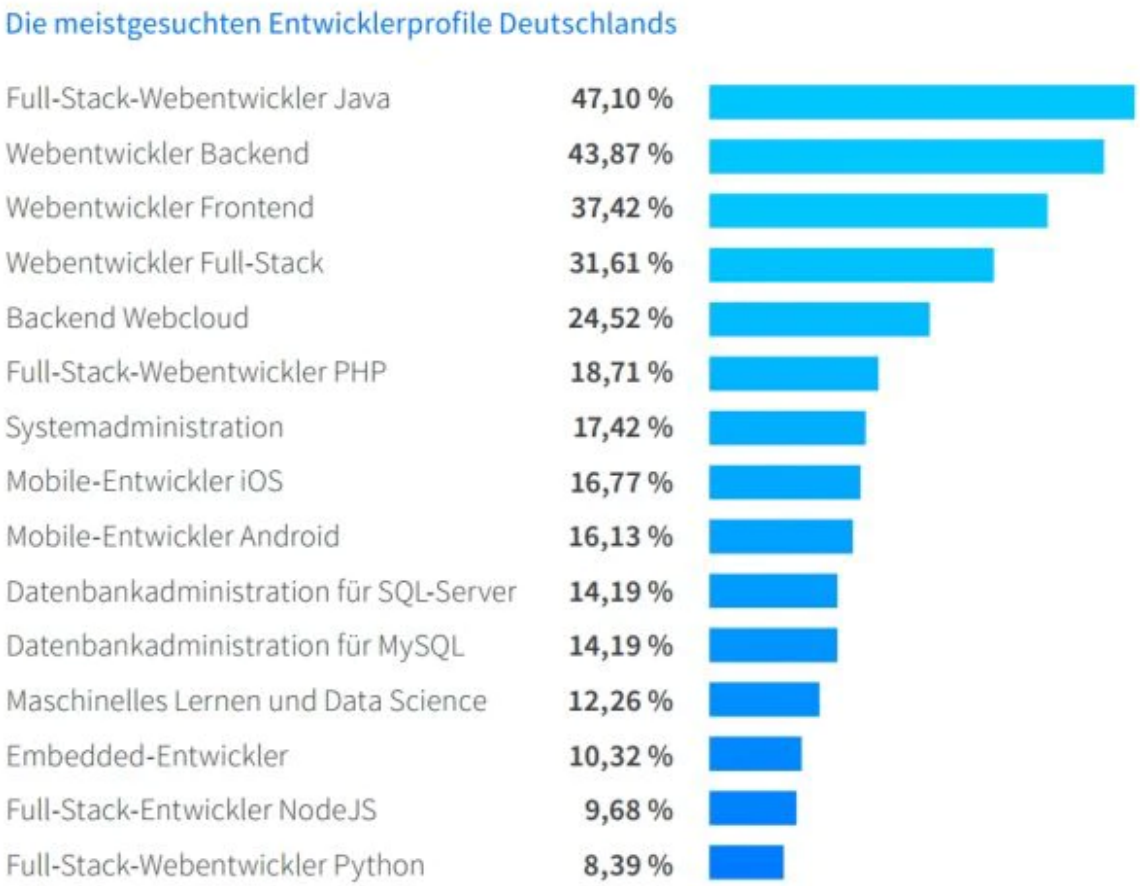


1.2.5.1.3 Programmiersprachen und Erscheinungsjahr

Abbildung 6: Abb. 1.a.3: Programmiersprachen im Ranking

- Assembler (~ Maschinensprache - aber kein ByteCode, seit 1950)
- Fortran (1957)
- Basic (1964)
- Pascal (1971)
- C (1972) / C++ (1985)
- Python (1991)
- Visual Basic (1991) / **VBA (1995)**
- **R (1993)**
- Java (1995) / JavaScript (1995)
- PHP (1995)
- C# (2001)
- Swift (2014), Objective-C (1984)
- Low Code: z.B. Scratch (2007), **Power Fx / Power Apps (2021)**

Abbildung 7: Abb. 1.a.4: Jobs als Programmierer im Ranking



1.2.5.1.4 Programmierparadigmen / Programmierstile

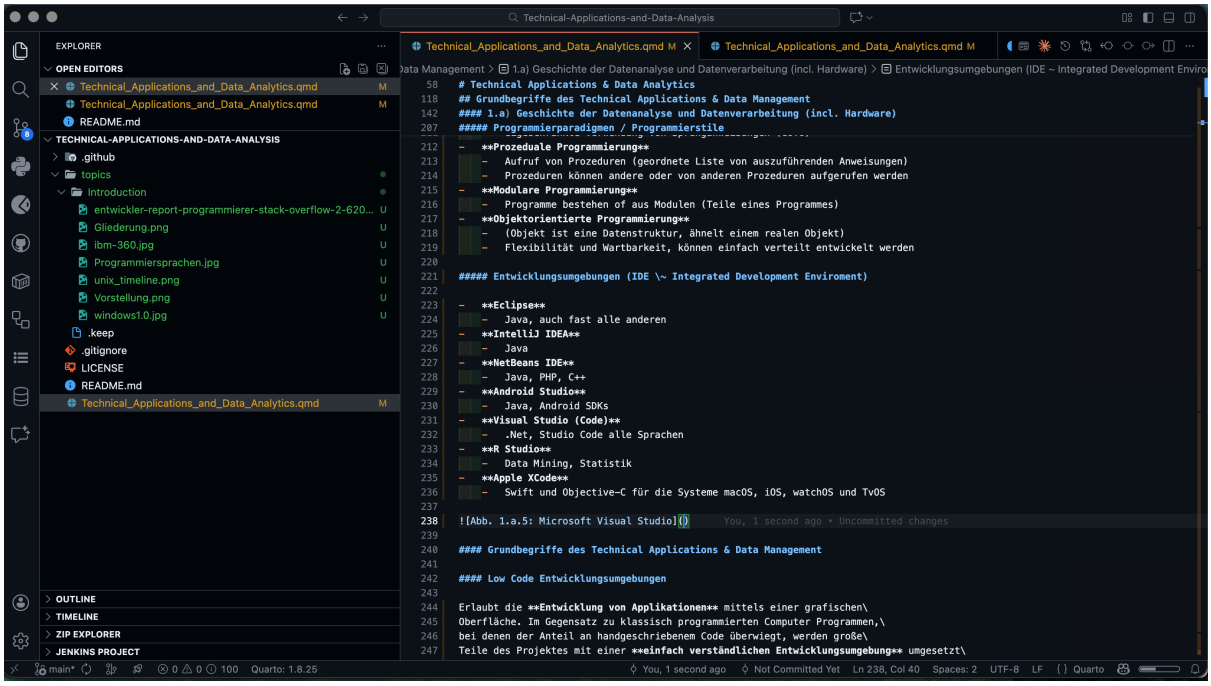
- **Strukturierte Programmierung**
  - baumartige Zerlegung
  - eigeschränkte Verwendung von Sprunganweisungen (GOTO)
- **Prozeduale Programmierung**
  - Aufruf von Prozeduren (geordnete Liste von auszuführenden Anweisungen)
  - Prozeduren können andere oder von anderen Prozeduren aufgerufen werden
- **Modulare Programmierung**
  - Programme bestehen of aus Modulen (Teile eines Programmes)
- **Objektorientierte Programmierung**
  - (Objekt ist eine Datenstruktur, ähnelt einem realen Objekt)
  - Flexibilität und Wartbarkeit, können einfach verteilt entwickelt werden

1.2.5.1.5 Entwicklungsumgebungen (IDE ~ Integrated Development Enviroment)

- **Eclipse**
  - Java, auch fast alle anderen

- **IntelliJ IDEA**
  - Java
- **NetBeans IDE**
  - Java, PHP, C++
- **Android Studio**
  - Java, Android SDKs
- **Visual Studio (Code)**
  - .Net, Studio Code alle Sprachen
- **R Studio**
  - Data Mining, Statistik
- **Apple XCode**
  - Swift und Objective-C für die Systeme macOS, iOS, watchOS und TvOS

Abbildung 8: Abb. 1.a.5: Microsoft Visual Studio

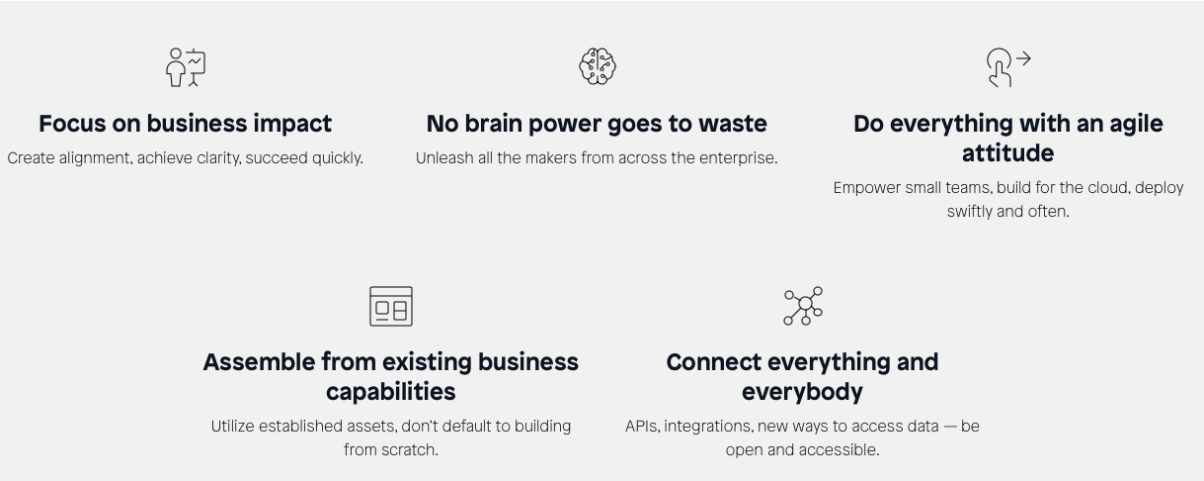


1.2.5.1.6 Low Code Entwicklungsumgebungen

Erlaubt die **Entwicklung von Applikationen** mittels einer grafischen Oberfläche. Im Gegensatz zu klassisch programmierten Computer Programmen, bei denen der Anteil an handgeschriebenem Code überwiegt, werden große Teile des Projektes mit einer **einfach verständlichen Entwicklungsumgebung** umgesetzt (vgl. *Rapidminer* oder *Microsoft Power Apps*).

Für Kinder und Jugendliche gibt es z.B. mit *Scratch* und *Game Builder Garage* für die *Nintendo Switch* Umgebungen zum Erlernen von Basisfähigkeiten der (Spiele-) Programmierung.

Abbildung 9: Abb. 1.a.6: Säulen der Low Code Entwicklung



1.2.5.1.7 Projektmanagement

Laut der **DIN 69901-5:2009** ist ein Projekt ein Vorhaben, das im Wesentlichen durch Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist (im Gegensatz zur Linienproduktion).

Ziel ist nach dieser Definition, dass Projekte **richtig geplant** und **gesteuert** werden, dass die **Risiken begrenzt**, **Chancen genutzt** und **Projektziele qualitativ**, **termingerecht** und **im Kostenrahmen erreicht** werden.

Abbildung 10: Abb. 1.a.7: Projektmanagement Prozess

1.2.5.1.8 Agiles Projektmanagement

SCRUM ist ein Framework, das für die Produktentwicklung in Teamarbeit verwendet werden kann.

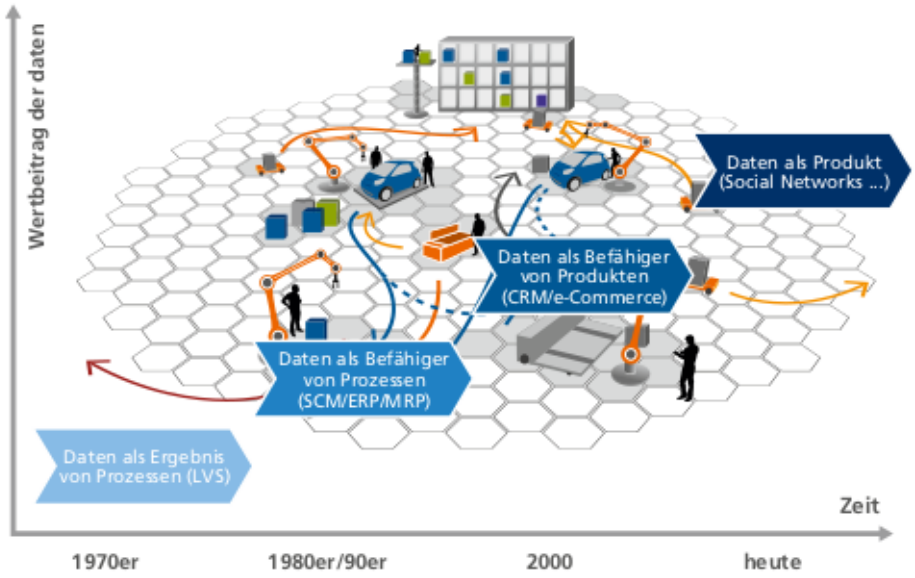
Ursprünglich wurde diese Methodik nur bei IT-Projekten eingesetzt, doch mittlerweile kann sie auch in branchenfremden Bereichen zum Einsatz kommen.

Abbildung 11: Abb. 1.a.8: SCRUM Manifesto

Daten wurden seit den 1970er Jahren unterstützend zur **Entscheidungsfindung** und in Produktionsprozessen eingesetzt. Im Gegensatz dazu sind Daten heutzutage oft auch das finale Produkt.

Glossar: LVS = Lagerverwaltungssystem **SCM** = Supply-Chain-Management **ERP** = Enterprise-Resource-Planning **MRP** = Material Requirements Planning **CRM** = Customer-Relationship-Management

Abbildung 12: Abb. 1.a.6: Rolle der Daten der Digitalisierung



1.2.6 1.b) Bedeutung von Daten und deren Digitalisierung

Die **Informationslogistik** befasst sich mit Informationsflüssen in Organisationsstrukturen. Das Ziel der Informationslogistik ist die Versorgung aller erforderlichen Akteure in einer Organisation

oder zwischen Organisationen mit den relevanten Daten zum richtigen Zeitpunkt.

Durch die ständig wachsende **Menge an Daten** aus verschiedensten Lebensbereichen können Produkte und Dienstleistungen digital erweitert und miteinander verknüpft werden. Dieser Entwicklung entsprechend findet auch auf wirtschaftlicher Ebene eine Kehrtwende, von der klassischen Produkt- und Servicezentrierung hin zu einer Fokussierung auf den individuellen Konsumenten und eine auf dessen Bedürfnisse maßgeschneiderte Lösung statt.

Die **Schnittstellen** zwischen den einzelnen Akteuren, vor allem diejenigen zum Kunden, müssen neu definiert werden um den Bedürfnissen nach individuellen Produkten und Dienstleistungen gerecht zu werden. Daten werden hierzu **analysiert** und der Erkenntnisgewinn fließt in die Geschäftsprozesse ein.

1.2.6.1 5 Formen der Datenanalyse

Die nun folgenden **5 Formen der Datenanalyse** unterscheiden sich in ihren Zielen, Ihren Methoden und den daraus gewonnenen Erkenntnissen.

1.2.6.1.1 a) Die deskriptive Datenanalyse

Die deskriptive Datenanalyse, auch als **beschreibende Datenauswertung** bezeichnet, konzentriert sich auf die **Daten aus der Vergangenheit**. Sie ordnet und strukturiert empirische Daten. Durch die Datenauswertung soll die Fragestellung beantwortet werden: “Was ist passiert?”.

Beispielsweise liefert sie Informationen wie den Umsatz im letzten Quartal oder die Art und die Anzahl von Serviceanfragen. Um diese Ergebnisse zu liefern, kann die deskriptive Analyse Daten aus verschiedenen Quellen extrahieren und die **Informationen aggregieren**, **ordnen** und **strukturieren**. Die deskriptive Analyse liefert aber keine Antworten auf Fragen wie: “Warum ist etwas geschehen?”. Oft werden deskriptive Datenanalysen mit anderen Analysemethoden kombiniert.

1.2.6.1.2 b) Die explorative Datenanalyse

Ziel der explorativen Datenanalyse ist es, **Zusammenhänge in Daten** zu finden und Hypothesen zu generieren. Vor der explorativen Analyse existiert nur ein begrenztes Wissen über die Zusammenhänge der Daten und Variablen. Typischer Anwendungsbereich für die explorative Datenanalyse ist das Data Mining. Durch das Aufdecken von Zusammenhängen mit Hilfe der explorativen Datenanalyse lassen sich Rückschlüsse auf die Ursachen der Vorgänge ziehen.

1.2.6.1.3 c) Die diagnostische Datenanalyse

Die diagnostische Datenanalyse beschäftigt sich ganz konkret mit der Fragestellung: “Warum ist etwas geschehen?”. Indem sie **historische** und andere Daten vergleicht, **Muster identifiziert** und **Zusammenhänge** aufdeckt, findet sie Ursachen oder gegenseitige Wechselwirkungen. Mit Hilfe der diagnostischen Datenanalyse können Unternehmen konkrete Problemstellungen lösen, da die Ursachen aufgezeigt werden.

1.2.6.1.4 d) Die prädiktive Datenanalyse

Die prädiktive Datenanalyse, auch als Vorhersageanalyse bzw. Predictive Analytics bezeichnet, gestattet den Blick in die Zukunft. Es wird die Fragestellung beantwortet: "Was wird passieren?".

Um die richtigen Vorhersagen zu treffen, nutzt die prädiktive Datenanalyse die Ergebnisse der zuvor beschriebenen **deskriptiven**, **explorativen** oder **diagnostischen Analysemethoden** sowie **Algorithmen** und **Methoden** der **Künstlichen Intelligenz (KI)** und des **Maschinellen Lernens (ML)**.

Durch das Finden von **Zusammenhängen**, **Ursachen** und **zeitlichen Tendenzen** werden zukünftige Trends vorhersagbar. Die Vorhersagewahrscheinlichkeit und -genauigkeit hängen maßgeblich von der Qualität der Daten, den gefundenen Mustern, Zusammenhängen und Trends sowie von der Intelligenz der Algorithmen ab. Beispielsweise lassen sich zukünftige Umsätze vorhersagen oder das Kundenverhalten prognostizieren.

1.2.6.1.5 e) Die präskriptive Datenanalyse

Die präskriptive Datenanalyse ist die komplexeste und aufwendigste Analysekategorie. Sie liefert Unternehmen aber einen immensen Mehrwert, indem Sie die Fragestellung beantwortet: "Mit welchen Maßnahmen lassen sich Probleme beseitigen, zukünftige Entwicklungen positiv beeinflussen oder die gesetzten Ziele erreichen?".

Präskriptive Datenauswertungen basieren auf **historischen** und **aktuellen Daten** interner und externer Datenquellen. Sie nutzen Ergebnisse zuvor beschriebener Analysekategorien. Die Vorhersagen werden kontinuierlich aktualisiert. Zum Einsatz kommen ML- und KI-Algorithmen, neuronale Netzwerke, Simulationen und Business Regeln.

Digitalisierung bezeichnetdas Umwandeln von analogen Werten in ein digitales Format bzw. digitale Repräsentation), welche zu verarbeiten sind.

Hierzu werden **Daten** IT-technisch gespeichert, intechnisch gespeichert, in Form von Bits & Bytes. Diese werden in verschiedenen Stellenwertesystemen und Zeichensätzen dargestellt.

Abbildung 13: Abb. 1.b.2: ASCII Zeichensatz

0	NUL	0	0	0	0	0	0	0	32	<space>	0	1	0	0	0	0	0	64	@	1	0	0	0	0	0	0	96	`	1	1	0	0	0	0	0
1	SOH	0	0	0	0	0	0	1	33	!	0	1	0	0	0	0	1	65	A	1	0	0	0	0	0	1	97	a	1	1	0	0	0	0	1
2	STX	0	0	0	0	0	1	0	34	"	0	1	0	0	0	1	0	66	B	1	0	0	0	0	1	0	98	b	1	1	0	0	0	1	0
3	ETX	0	0	0	0	0	1	1	35	#	0	1	0	0	0	1	1	67	C	1	0	0	0	0	1	1	99	c	1	1	0	0	0	1	1
4	EOT	0	0	0	0	1	0	0	36	$	0	1	0	0	1	0	0	68	D	1	0	0	0	1	0	0	100	d	1	1	0	0	1	0	0
5	ENQ	0	0	0	0	1	0	1	37	%	0	1	0	0	1	0	1	69	E	1	0	0	0	1	0	1	101	e	1	1	0	0	1	0	1
6	ACK	0	0	0	0	1	1	0	38	&	0	1	0	0	1	1	0	70	F	1	0	0	0	1	1	0	102	f	1	1	0	0	1	1	0
7	BEL	0	0	0	0	1	1	1	39	'	0	1	0	0	1	1	1	71	G	1	0	0	0	1	1	1	103	g	1	1	0	0	1	1	1
8	BS	0	0	0	1	0	0	0	40	(	0	1	0	1	0	0	0	72	H	1	0	0	1	0	0	0	104	h	1	1	0	1	0	0	0
9	HT	0	0	0	1	0	0	1	41	)	0	1	0	1	0	0	1	73	I	1	0	0	1	0	0	1	105	i	1	1	0	1	0	0	1
10	LF	0	0	0	1	0	1	0	42	*	0	1	0	1	0	1	0	74	J	1	0	0	1	0	1	0	106	j	1	1	0	1	0	1	0
11	VT	0	0	0	1	0	1	1	43	+	0	1	0	1	0	1	1	75	K	1	0	0	1	0	1	1	107	k	1	1	0	1	0	1	1
12	FF	0	0	0	1	1	0	0	44	,	0	1	0	1	1	0	0	76	L	1	0	0	1	1	0	0	108	l	1	1	0	1	1	0	0
13	CR	0	0	0	1	1	0	1	45	-	0	1	0	1	1	0	1	77	M	1	0	0	1	1	0	1	109	m	1	1	0	1	1	0	1
14	SO	0	0	0	1	1	1	0	46	.	0	1	0	1	1	1	0	78	N	1	0	0	1	1	1	0	110	n	1	1	0	1	1	1	0
15	SI	0	0	0	1	1	1	1	47	/	0	1	0	1	1	1	1	79	O	1	0	0	1	1	1	1	111	o	1	1	0	1	1	1	1
16	DLE	0	0	1	0	0	0	0	48	0	0	1	1	0	0	0	0	80	P	1	0	1	0	0	0	0	112	p	1	1	1	0	0	0	0
17	DC1	0	0	1	0	0	0	1	49	1	0	1	1	0	0	0	1	81	Q	1	0	1	0	0	0	1	113	q	1	1	1	0	0	0	1
18	DC2	0	0	1	0	0	1	0	50	2	0	1	1	0	0	1	0	82	R	1	0	1	0	0	1	0	114	r	1	1	1	0	0	1	0
19	DC3	0	0	1	0	0	1	1	51	3	0	1	1	0	0	1	1	83	S	1	0	1	0	0	1	1	115	s	1	1	1	0	0	1	1
20	DC4	0	0	1	0	1	0	0	52	4	0	1	1	0	1	0	0	84	T	1	0	1	0	1	0	0	116	t	1	1	1	0	1	0	0
21	NAK	0	0	1	0	1	0	1	53	5	0	1	1	0	1	0	1	85	U	1	0	1	0	1	0	1	117	u	1	1	1	0	1	0	1
22	SYN	0	0	1	0	1	1	0	54	6	0	1	1	0	1	1	0	86	V	1	0	1	0	1	1	0	118	v	1	1	1	0	1	1	0
23	ETB	0	0	1	0	1	1	1	55	7	0	1	1	0	1	1	1	87	W	1	0	1	0	1	1	1	119	w	1	1	1	0	1	1	1
24	CAN	0	0	1	1	0	0	0	56	8	0	1	1	1	0	0	0	88	X	1	0	1	1	0	0	0	120	x	1	1	1	1	0	0	0
25	EM	0	0	1	1	0	0	1	57	9	0	1	1	1	0	0	1	89	Y	1	0	1	1	0	0	1	121	y	1	1	1	1	0	0	1
26	SUB	0	0	1	1	0	1	0	58	:	0	1	1	1	0	1	0	90	Z	1	0	1	1	0	1	0	122	z	1	1	1	1	0	1	0
27	ESC	0	0	1	1	0	1	1	59	;	0	1	1	1	0	1	1	91	[	1	0	1	1	0	1	1	123	{	1	1	1	1	0	1	1
28	FS	0	0	1	1	1	0	0	60	<	0	1	1	1	1	0	0	92	\	1	0	1	1	1	0	0	124	|	1	1	1	1	1	0	0
29	GS	0	0	1	1	1	0	1	61	=	0	1	1	1	1	0	1	93	]	1	0	1	1	1	0	1	125	}	1	1	1	1	1	0	1
30	RS	0	0	1	1	1	1	0	62	>	0	1	1	1	1	1	0	94	^	1	0	1	1	1	1	0	126	~	1	1	1	1	1	1	0
31	US	0	0	1	1	1	1	1	63	?	0	1	1	1	1	1	1	95	_	1	0	1	1	1	1	1	127	DEL	1	1	1	1	1	1	1

1.2.6.2 Speichereinheiten

Die kleinste Einheit ist *1 Bit*. "Bit" steht für "Binary Digit". In der Informatik wird im Binärsystem gerechnet, was bedeutet, dass es nur zwei Zustände gibt – *an / aus* bzw. *0 / 1*. Der Computer rechnet mit diesen Zahlen, dem binären Dualsystem. Daher wird der Faktor *1024* genutzt, denn im Binärsystem kommt die Zahl *1000* nicht vor, sondern nur *1024* (*1, 2, 4, 16, 32, 64, 128, 256, 512, 1024 = 10* Bit.

Abbildung 14: Abb. 1.b.3: Tabelle der Speichergrößen

byte	b	8 bits	1 byte
kilobyte	Kb	1024 bytes	1 024 bytes
megabyte	MB	1024 KB	1 048 576 bytes
gigabyte	GB	1024 MB	1 073 741 824 bytes
terabyte	TB	1024 GB	1 099 511 627 776 bytes
Petabyte	PB	1024 TB	1 125 899 906 842 624 bytes
Exabyte	EB	1024 PB	1 152 921 504 606 846 976 bytes
Zetabyte	ZB	1024 EB	1 180 591 620 717 411 303 424 bytes
Yottabyte	YB	1024 ZB	1 208 925 819 614 629 174 706 176 bytes
Brontobyte	BB	1024 YB	1 237 940 039 285 380 274 899 124 224 bytes
Geopbyte	GB	1024 BB	1 267 650 600 228 229 401 496 703 205 376 bytes

1.2.6.3 Stellenwertsysteme

Dualsystem / Binärsystem: Basis *2*, Ziffern *0* und *1*

Oktalsystem: Basis *8* (= 23), Ziffern *0* bis *7*

Dezimalsystem: Basis *10*, Ziffern *0* bis *9*

Hexadezimalsystem: Basis *16 (*24*)*, Ziffern *0* bis *9* und *A* bis *F*

Abbildung 15: Abb. 1.b.4: Stellenwertsysteme

Hex.	Dualsystem	Dez.
0	0 0 0 0	00
1	0 0 0 1	01
2	0 0 1 0	02
3	0 0 1 1	03
4	0 1 0 0	04
5	0 1 0 1	05
6	0 1 1 0	06
7	0 1 1 1	07
8	1 0 0 0	08
9	1 0 0 1	09
A	1 0 1 0	10
B	1 0 1 1	11
C	1 1 0 0	12
D	1 1 0 1	13
E	1 1 1 0	14
F	1 1 1 1	15

1.2.6.4 Zeichensätze

ASCII: Früher Standardzeichensatz, *7* Bit, Kodierung von *128* Zeichen möglich

ANSI: *8* Bit Zeichensatz, Kodierung von *256* Zeichen möglich. Die Zeichen *0* bis *127* sind der ASCII-Zeichensatz und die Werte zwischen *128* und 255 sind Sonderzeichen.

ISO-8859: Set von Zeichensätzen, die auf *8* Bit basieren und *256* Zeichen abbilden. Die ersten *128* Zeichen (0 - 127) sind immer identisch mit dem ASCII-Zeichensatz. Die zweiten *128* Stellen enthalten spezielle Zeichen. Z.B. ISO-8859-1: schriftspezifische Zeichen für westeurop. / amer. Sprachen

Unicode: Universeller Zeichensatz zur Darstellung aller Zeichen aller Sprachen. *16* Bit gleich *65536* Zeichen (erste Byte auf null gesetzt -> ASCII / ANSI)

UTF-8: „8-Bit Unicode Transformation Format“, unterstützt bis zu *4* Byte, mit denen sich *1.114.112* Zeichen abbilden lassen.

1.2.6.5 Textbasierte Dateiformate

CSV (comma-separated values) Dateiformat: Eine Textdatei zum Austausch einfach strukturierter Daten, die Trennung der Daten erfolgt mit dem „,“ Zeichen, Zeichenkodierung 7-Bit-ASCII-Code. Beispiel: alter,geschlecht,note_mathe,note_annahme,kopf,angst,interesse,praxis 44,3,1,1,60,1,1,30 22,1,2,4,50,3,3,50

XML: e**X**tensible **M**arkup **L**anguage, wie bei HTML sind Daten von Markierungen umgeben. Es sind Struktur und Bedeutungen zu vergeben. Bsp.:

```
<?xml version="1.0"?>
<CAT>
  <SUBCAT>
    <NAME>Izzy</NAME>
    <BREED>Siamese</BREED>
    <AGE>6</AGE>
    <ALTERED>yes</ALTERED>
    <DECLAWED>no</DECLAWED>
    <LICENSE>Izz138bod</LICENSE>
    <OWNER>Colin Wilcox</OWNER>
  </SUBCAT>
</CAT>
```

1.2.6.6 (Text-)Editoren

Um Textdateien jeder Art (aber auch Code wie HTML oder C#) einfach auszulesen und zu bearbeiten gibt es sogenannte Text-Editoren. Hier eine Auswahl an kostenlosen Programmen für Windows, Mac und iPadOS.

Brackets: Für *Mac (und Windows)*, einfach zu bedienen und etliche neue

Funktionen können durch Erweiterungen hinzuzugefügt werden
https://github.com/adobe/brackets/releases/download/release-1.14.2/Brackets.Release.1.14.2.dmg

NotePad++: Für *Windows*, mächtige Software mit vielen Funktionen und

Erweiterungen. Wird häufig im Produktiveinsatz verwendet.

https://github.com/notepad-plus-plus/notepad-plus-plus/releases/download/v8.1.9/npp.8.1.9.Installer.x64.exe

1.2.6.6.1 Kodex:

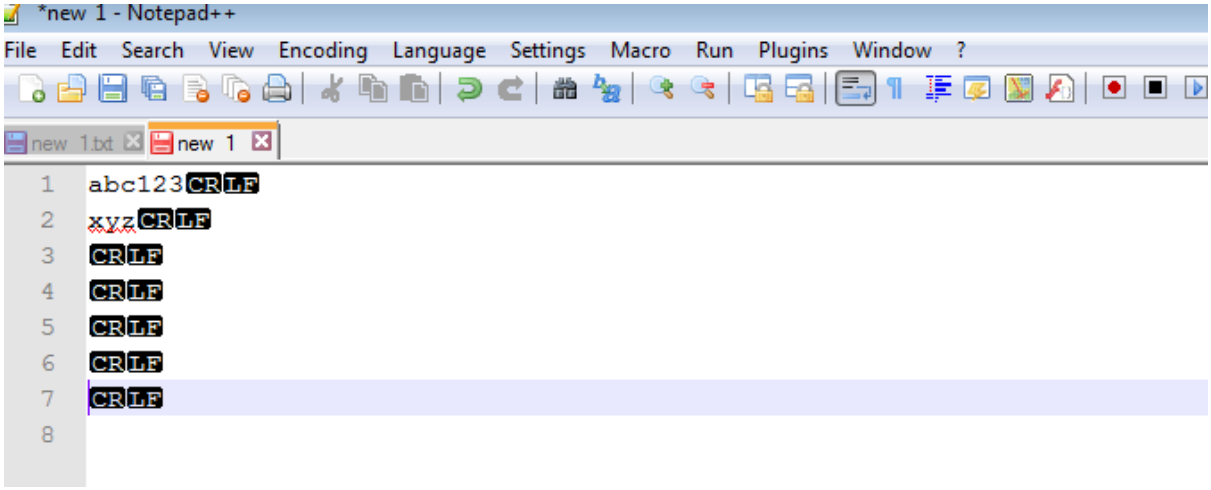
Für *iPadOS*, simpel und leicht zu bedienen Download über den Apple AppStore

1.2.6.7 Line Feed (LF) vs. Carriage Return + Line Feed (CRLF)

LF: Unter Linux / Unix und Mac wird mittels *LF* eine neue Zeile eingeleitet (auch mit *Newline* oder $\backslash n$ bezeichnet).

CRLF: Unter Windows-Systemen wird mit *CR* eingeleitet (auch mit *Carriage Return* oder $\backslash r$ bezeichnet) der Cursor an den Anfang der Zeile gesetzt und mit *LF* die neue Zeile eingeleitet.

Abbildung 16: Abb. 1.b.5: NotePad++ - CRLF

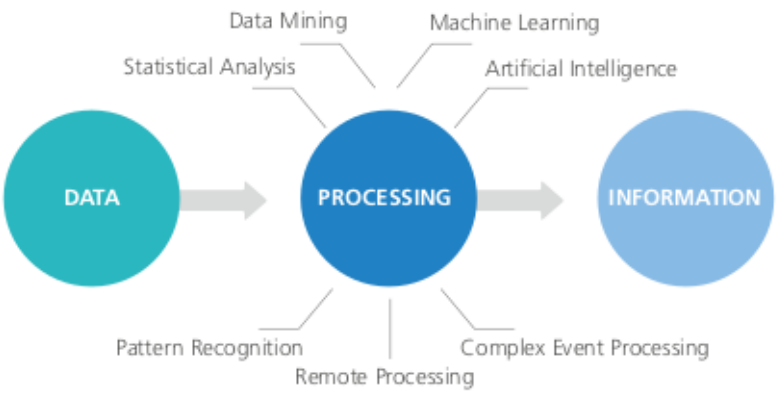


1.2.6.8 Grundbegriffe des Technical Applications & Data Management

Weiterhin muss je nach Anwendungsfall der eigentliche **Ort der Datenanalyse** ausgewählt werden. Je nach Anforderung ist es sinnvoll Daten zentral zu verarbeiten, wofür sie erst an einen zentralen Ort transportiert werden müssen. Dies ist erstens aufwändig und zweitens übersteigt die Menge an Daten womöglich die Systemkapazitäten.

Der Ansatz, die Daten remote, sprich dort zu verarbeiten, wo sie entstehen, bietet die Chance den Prozess effizienter zu gestalten, und zusätzlich nicht nur statische Datenbestände, sondern auch kontinuierlich Ströme von Daten in Echtzeit auszuwerten.

Abbildung 17: Abb. 1.b.7: Methoden zur Extraktion von Informationen



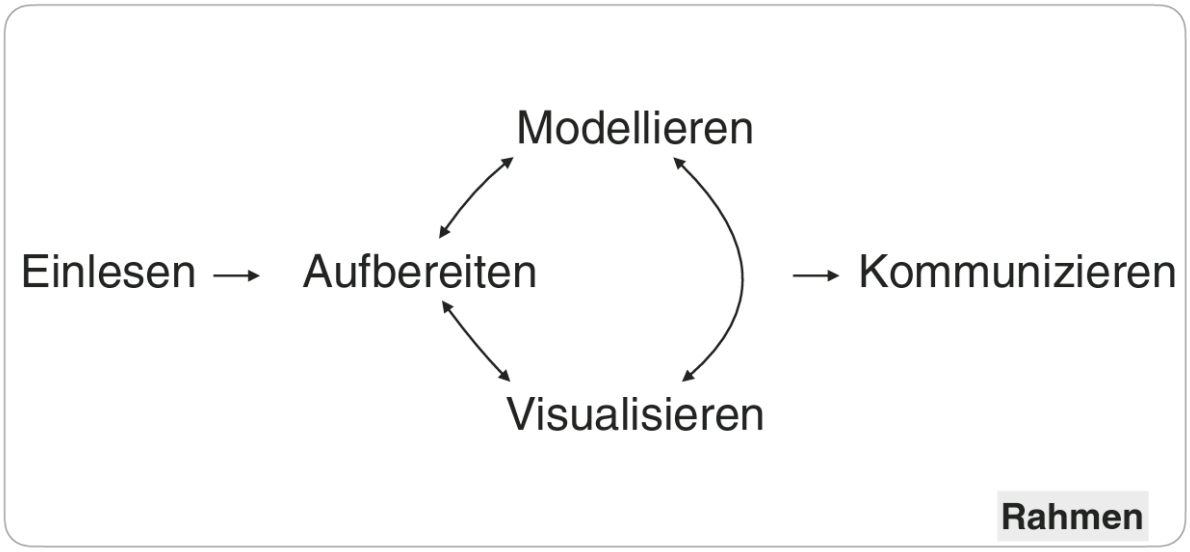
Datenanalyse ist praktisch betrachtet

das **Einlesen**, **Aufbereiten**, **Modellieren**, **Visualisieren** und **Kommunizieren** von Daten.

Die Schritte *Aufbereiten*, *Visualisieren* und *Modellieren* folgen keinem starren Ablauf.

Die Datenanalyse ist in einen Rahmen eingebettet. Dieser beinhaltet *philosophische* und *technische* Grundlagen

Abbildung 18: Abb. 1.b.8: Der Rahmen der Datenanalyse



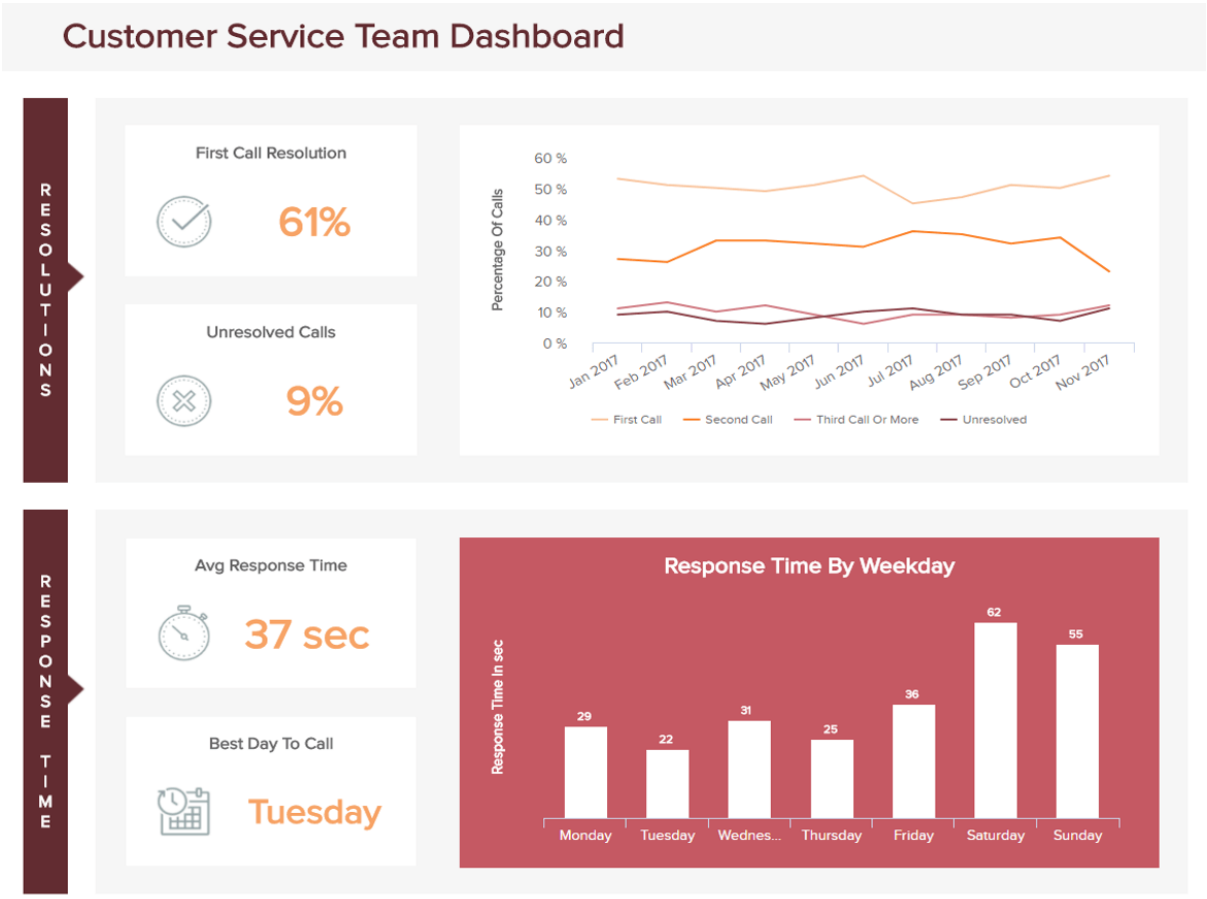
1.2.7 1.c) Ziele der Datenanalyse und Stand der Technik

Ziel der **Datenanalyse** ist es, aus den vorhandenen Daten durch Anwendung verschiedener Methoden und Analyseverfahren die jeweils benötigten Erkenntnisse oder Informationen zu gewinnen.

Durch Technologien wie **KI (Künstliche Intelligenz)** und **Machine Learning**, lassen sich automatisiert große Datenmengen analysieren und zur Verbesserung von Geschäftsprozessen heranziehen.

Große Rechenzentren können diese Analysen binnen kurzer Zeit durchführen.

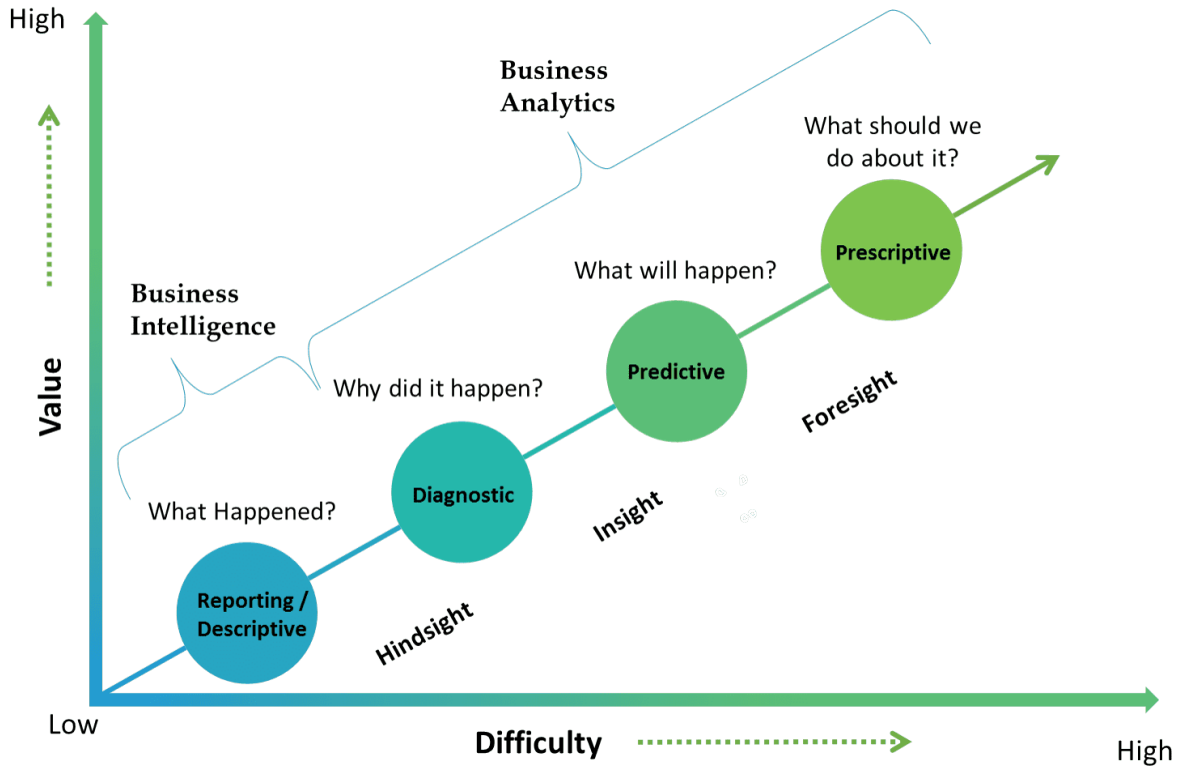
Abbildung 19: Abb. 1.c.1: Der Rahmen der Datenanalyse



1.2.8 1.d) Einführung in die Begriffe BI, Reporting und Big Data

Es sind grundsätzlich **zwei verschiedene Blickwinkel** der Daten- auswertung zu Unterscheiden: 1. **Business Intelligence** „Was ist in der Vergangenheit bis heute geschehen und warum ist es geschehen¿‘. Die Business Intelligence findet Muster und Trends, zielt aber nicht darauf ab, die zukünftigen Entwicklungen vorherzusagen. 2. **Business Analytics** „Warum ist etwas geschehen und was bedeutet das für die Zukunft?”. Es werden die verantwortlichen Faktoren und kausalen Zusammen- hänge für die Geschehnisse und Abläufe ermittelt. Diese nutzt Business Analytics, um Vorhersagen über zukünftige Entwicklungen zu machen.

Abbildung 20: Abb. 1.d.1: Formen der Datenanalyse

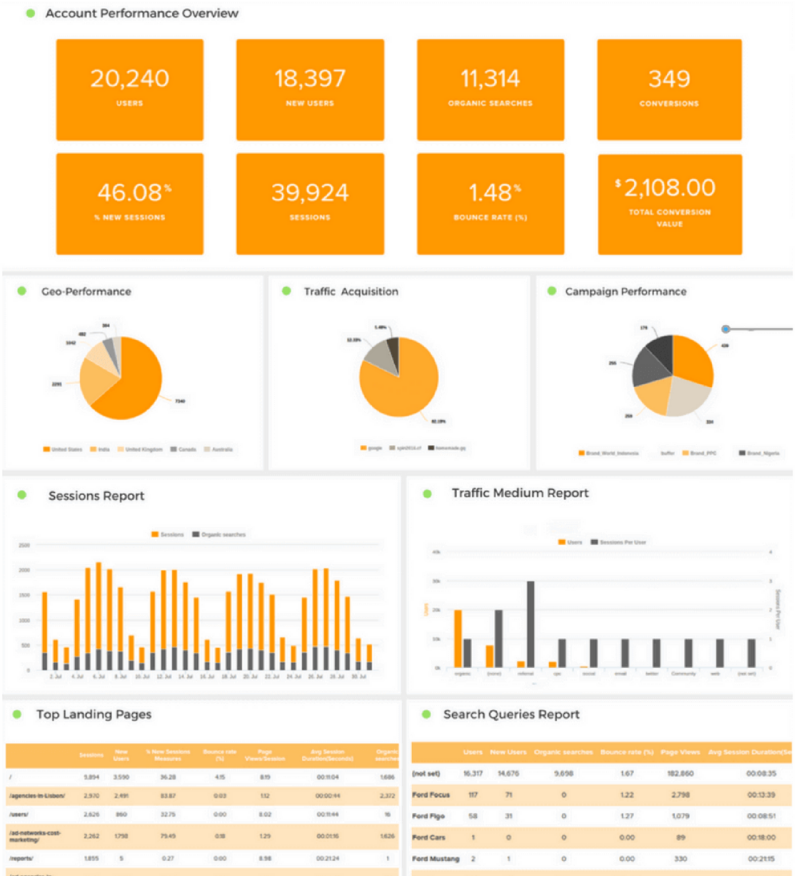


1.2.8.1 Reporting

Der Begriff Reporting beschreibt das **systematische Erstellen** von **Berichten** auf Basis relevanter Unternehmensdaten
und Informationen.

Wichtigstes Ziel des Reportings ist die Unterstützung der Führungs- und Entscheidungsprozesse in den verschiedenen Unternehmensbereichen.

Abbildung 21: Abb. 1.d.2: Dashboard einer Reporting Software



1.2.9 1.d) Einführung in die Begriffe BI, Reporting und Big Data

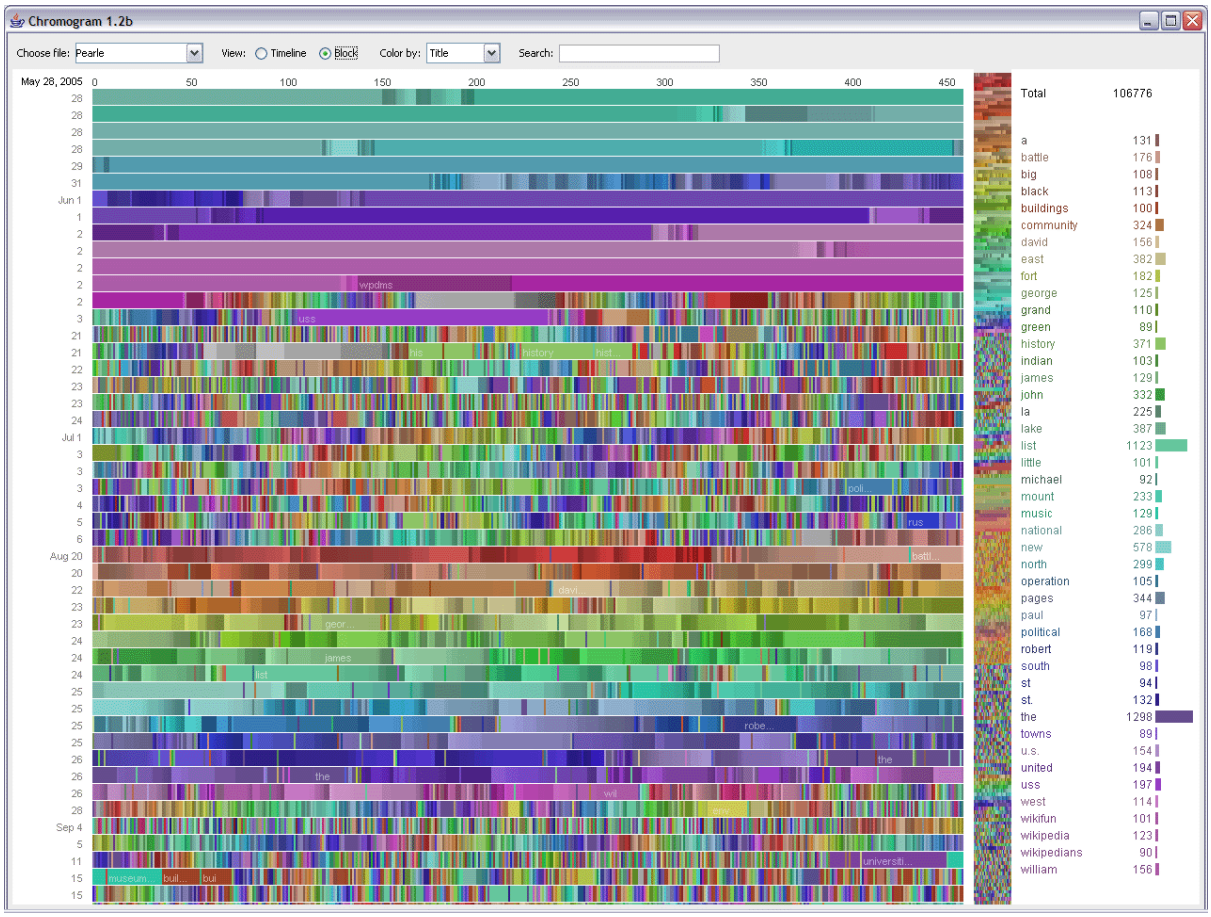
1.2.9.1 Big Data

Big Data steht für die **große Mengen** an **Daten** – Daten in der Größenordnung von Zettabytes, die von Computern, Mobilgeräten und elektronischen Sensoren produziert werden.

Anhand dieser Daten können Unternehmen **Entscheidungen** treffen, Prozesse und Richtlinien verbessern und kundenorientierte Produkte, Services und Erfahrungen entwickeln.

Big Data werden nicht nur wegen ihres Umfangs als „groß“ bezeichnet, sondern auch wegen der **Vielfalt** und **Komplexität** ihrer **Beschaffenheit**.

Abbildung 22: Abb. 1.d.3: Aktivität eines Wikipedia-Bots



1.3 Technical Applications

- Einführung in die technische Datenerfassung
- Verschiedene Quellen von Daten und deren Zusammenhänge
- Speicherung der Daten, Datenbanken, Datawarehouses
- Grundlagen der Technischen Applikationen
- Einführung in verschiedene Anwendungen und deren Darstellung mit Vor- und Nachteilen

1.3.1 2.a) Einführung in die technische Datenerfassung

Die **Anforderungen** von *Datenanalyse*, *Data Mining*, *maschinellem Lernen* und auch der *klassischen Statistik* gleichen sich. Die geteilten **Wissensgebiete** zur Anwendung dieser Verfahren sind:

- Philosophische Grundlagen
- Mathematisch-statistische Anwendungen
- Computerwissenschaftliche Anwendungen
- Fach- und Branchenkenntnis

Abbildung 23: Abb. 2.a.1: Buch – Einführung in die Datenverarbeitung

Einführung in die Datenverarbeitung

für

Berufsgrundbildungsjahr
Grundstufe der Berufsausbildung
Berufsfachschule
Berufsaufbauschule
Berufsfortbildung

von

Dr. Manfred Zschenderlein †

Dipl.-Kfm. Heinz Paelzer
EDV-Systemberater, Koblenz

Dipl.-Hdl. Birgid Fuchs
Berufsbildende Schule Wirtschaft Koblenz

ISBN 3-8045-
3883-5

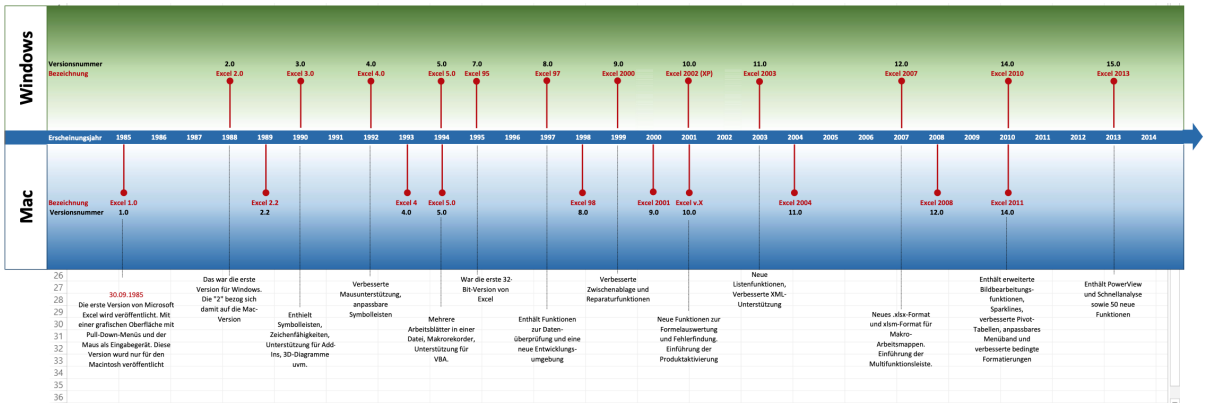
11.[2.] Auflage, 1988
(2., unveränderter Nachdruck der
11. Auflage, 1986)

Winklers Verlag
Gebrüder Grimm
Darmstadt

1.3.1.1 *Microsoft Excel*

Die einfachste Form Daten am PC zu betrachten, bearbeiten und darzustellen ist heutzutage *Microsoft Excel* oder eine vergleichbare Tabellenkalkulation. Excel ist in Version 1.0 erstmals 1985 für Macintosh erschienen. Im November 1987 in Version 2.0 ist es auch für Windows (Windows 2.x / 386) erschienen.

Abbildung 24: Abb. 2.a.2: Zeitschiene *Microsoft Excel*



1.3.1.2 Grundbegriffe

Daten (die Einzahl Datum ist ungewöhnlich) kann man als Informationen, die eine Funktion haben, beschreiben.1 Somit sind, von für den Menschen nützliche Daten, beliebige Daten ohne Zweck und Systematik ausgenommen (Sauer, 2019). Der Zweck muss demnach zumindest Fachpersonal bekannt und gängigen Methoden der **Datenerfassung** und **Datenverarbeitung** zugänglich sein.

Daten bestehen somit aus **Merkmalen** (Variablen). Hinzu kommen **Beobachtungseinheiten** (Fälle, Beobachtungen). Eine **Tabelle** mit ihren rechtwinkligen Achsen der Zeilen und Spalten hält im Kopf die **Variablennamen** und in den einzelnen Zeilen, die jeweils dem Merkmal zugordneten **Objekte**.

Merkmal 1 | **Merkmal 2** | **Merkmal 3** Objekt 1 M1 | Objekt 1 M2 | Objekt 1 M3 Objekt 2 M1 | Objekt 2 M2 | Objekt 2 M3 Objekt 3 M1 | Objekt 3 M2 | Objekt 3 M3

Abb. 2.a.3: Tabelle als Datensatz

1.3.1.3 Verschiedene Quellen von Daten und deren Zusammenhänge

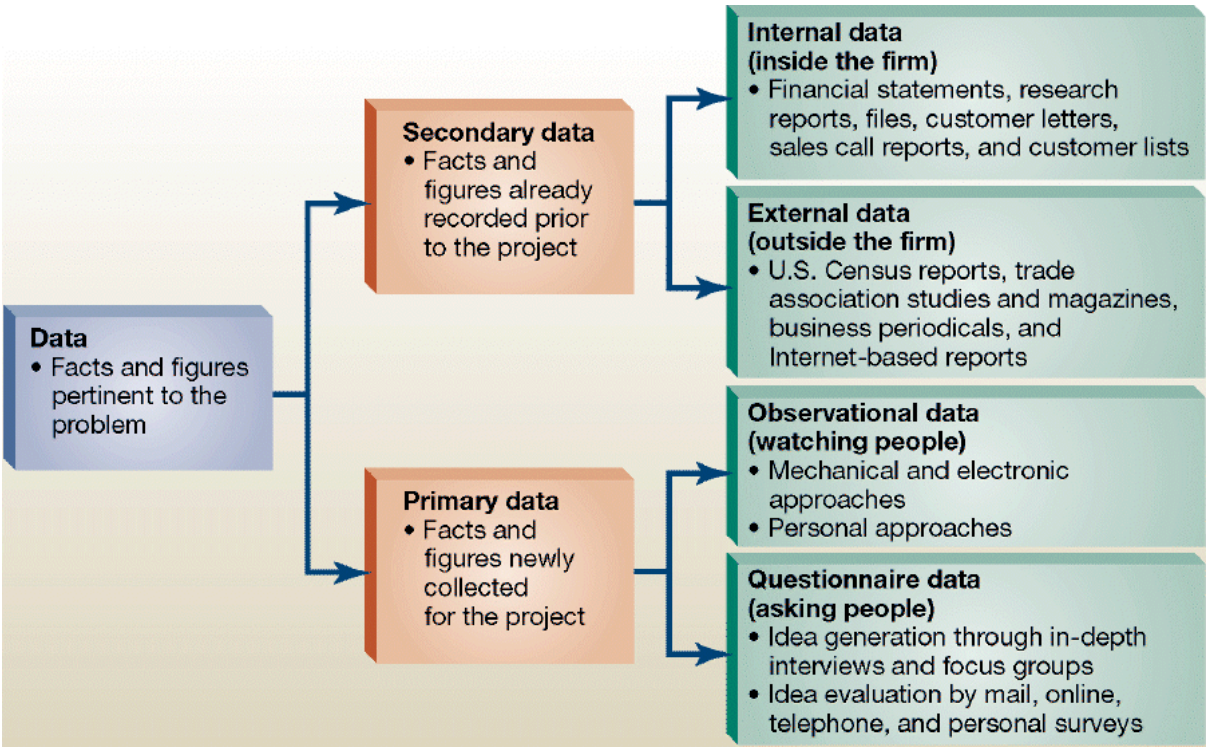
Die Herkunft von Daten wird differenziert betrachtet.

1.3.1.3.1 Primärdaten

- Beobachtung
- Befragung
- Experimentell
- Sekundärdaten
  - Interne Quellen
  - Externe Quellen

1.3.2 2.b) Verschiedene Quellen von Daten und deren Zusammenhänge

Abbildung 25: Abb. 2.b.1: Primär- und Sekundärdaten



1.3.2.1 Speicherung von Daten in Dateien

Typischer Weise werden Daten in ***CSV*** und ***XML*** **Dateien** gespeichert.

Einfache **Text-Dateien** (*.txt*) oder **Textverarbeitungsdateien** mit Endungen (Suffix) wie *.docx* (Microsoft Word) bzw. .xlsx (Microsoft Excel) und offene Formate (*.odt*) von Open-Source Office Suiten, eignen sich aufgrund der evtl. fehlenden Bibliotheken bzw. der fehlenden Struktur in den technischen Applikationen eher weniger.

Ebenso denkbar ist die Speicherung in einer **Binärdatei**. Hier sind die Daten in Bitmustern gespeichert und nur mit entsprechenden Import-Bibliotheken der technischen Applikationen lesbar.

1.3.3 2.c) Speicherung der Daten, Datenbanken, Datawarehouses

1.3.3.1 Data Warehouse

Ein **Data Warehouse** ist eine Art **Datenmanagementsystem**, mit dem BI- Aktivitäten (Business Intelligence), insbesondere Analysen, aktiviert und unterstützt werden. Data Warehouses dienen ausschließlich zur Durchführung von Abfragen und Analysen und enthalten häufig große Mengen an Verlaufsdaten. Die Daten in einem Data Warehouse stammen üblicherweise aus einer Vielzahl von Quellen, z. B. aus Anwendungsprotokolldateien und Transaktionsanwendungen.

Ein Data Warehouse zentralisiert und konsolidiert **große Datenmengen** aus mehreren Quellen. Die Analysefunktionen ermöglichen es Unternehmen, **wertvolle Geschäftsinformationen** aus ihren Daten abzuleiten, um die **Entscheidungsfindung** zu verbessern. Im Laufe der Zeit wird ein Verlaufsdatensatz erstellt, der für Daten- und Geschäftsanalysten von unschätzbarem Wert

sein kann. Aufgrund dieser Funktionen kann ein Data Warehouse als die „*einzige Quelle wahrer Daten*“ eines Unternehmens betrachtet werden.

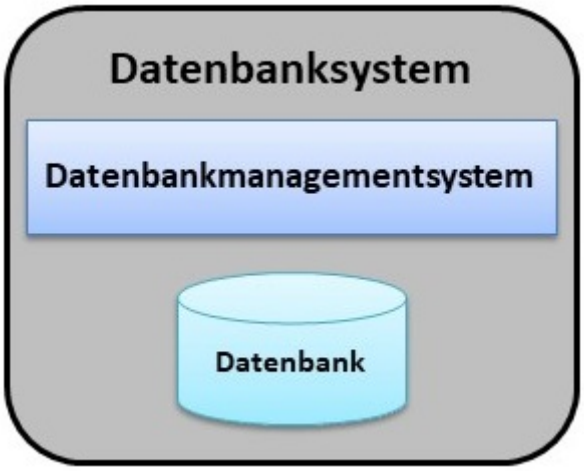
1.3.3.2 Speicherung von Daten in Computerspielen

Eine Datenbank ist ein elektronisches Verwaltungssystem, das besonders mit großen Datenmengen effizient, widerspruchsfrei, dauerhaft umgehen muss und logische Zusammenhänge digital abbilden kann.

Es können Datenbestände aus verschiedenen Teilmengen zusammengestellt und bedarfsgerecht für Anwendungsprogramme und deren Benutzern angezeigt werden.

Heutzutage kann eine Datenbank viele Informationen beinhalten. Komplexe Datenbanken werden oft in Warenwirtschaftssystemen (abgekürzt WWS oder WaWi) verwendet oder auch zur Verarbeitung von Sozial- und Gesundheitsdaten (Sozialversicherungsträger, Sozialhilfeträger, Versorgungsbehörden, Haus- und Fachärzte).

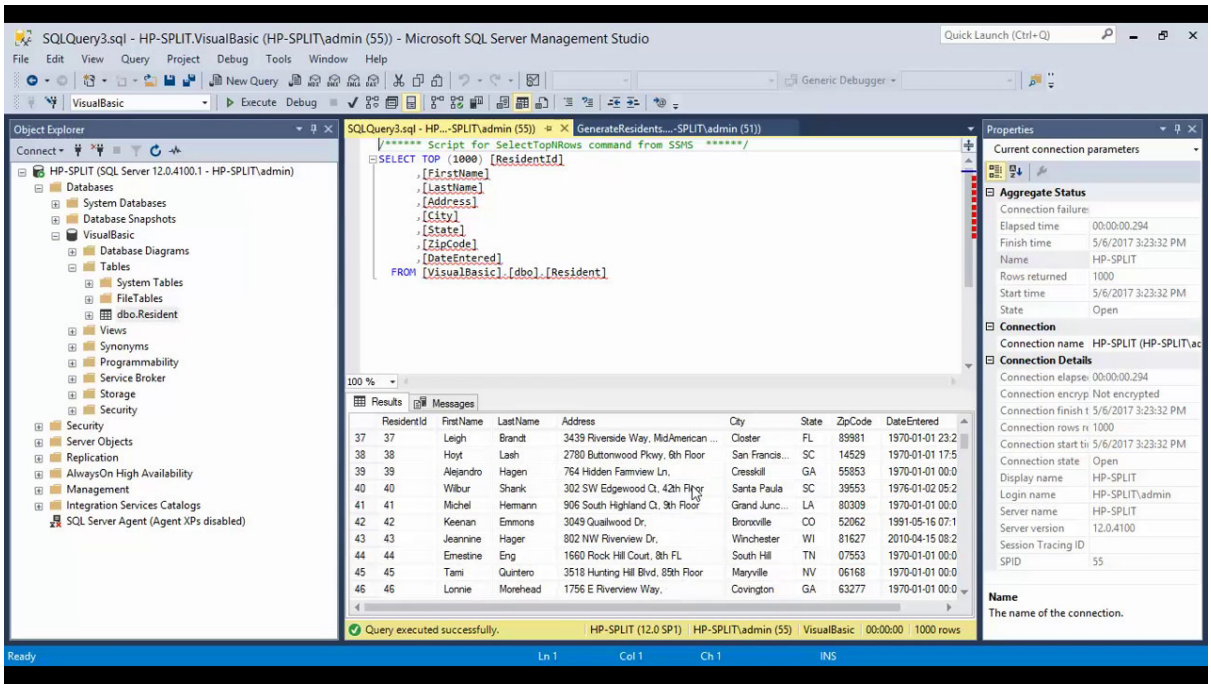
Abbildung 26: Abb. 2.c.1: Datenbanksysteme



Spiele verwalten eine Menge Daten. Zum einen die Werte und Eigenschaften von In-Game Gegenständen, sowie die erhobenen Daten von Spieler- Interaktionen und Verhalten (z.B Highscores).

Da Spiele oftmals im Hinblick auf **Performance** konzipiert werden, müssen diese Daten effizient und konsistent gespeichert werden. Hierzu eigenen sich **Datenbanksysteme** um große Datenmengen zu speichern.

Abbildung 27: Abb. 2.c.2: *Microsoft SQL Studio*



1.3.3.3 Structured Query Language – SQL

Datenbank-Sprache(n):

- MySQL
- Microsoft SQL Server
- SQLite
- uvm. -> **Datenbank-Management-Systeme**!

Datenbank-Definition- und Manipulationssprache:

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- Data Control Language (DCL)

1.3.3.4 Data Definition Language

```
# Erstellen einer Tabelle
CREATE TABLE loot_table(
id integer PRIMARY KEY,
name
text,
probability float);

# Löschen einer Tabelle
- DROP TABLE

# Editieren der Tabellenstruktur
- ALTER TABLE
```

1.3.3.5 Data Manipulation Language

```
# Daten in Tabelle einfügen
INSERT INTO loot_table(1, 'Stuff you don't need', 0.999);
INSERT INTO loot_table(2, 'Stuff you need', 0.0001);
INSERT INTO loot_table(3, 'Stuff you might need', 0.0099);

# Daten in Tabelle verändern
UPDATE loot_table SET probability = 0 WHERE id <> 1

# Daten in Tabelle löschen
DELETE FROM loot_table WHERE probability < 0.1

# Daten der Tabelle ausgeben
SELECT \* FROM loot_table
```

1.3.3.6 Verbund von Daten

```
# Verbund von Daten aus zwei Tabellen
JOIN

SELECT
l_t.id, l_t.name, price
FROM loot_table l_t
LEFT JOIN marketvalue_table mv_t
ON mv_t.name = l_t.name
```

id	name	price
1	Stuff you don't need	1
2	Stuff you need	100
3	Stuff you might need	25

1.3.3.7 Mengenoperationen

```
# Mengenoperationen auf Tabellen
UNION
SELECT \* loot_table
UNION ALL
SELECT id, name, probability

INTERSECT

SELECT \* loot_table
INTERSECT
SELECT id, name, probability WHERE id IN (1, 2)

MINUS
```

```
SELECT \* loot_table
MINUS (EXCEPT!)
SELECT id, name, probability WHERE id IN (1, 2)
```

Fragen an Lukas der als Gast-Dozent zu einer Vorlesung kam (er ist Softwareentwickler im Gaming-Bereich für Datenbanksysteme)

- Welche Arten der Speicherung von Spieldaten (Spielstände, Konfiguration / Einstellungen, Werte von Spielobjekten z.B. Fußballspieler) gibt es grundsätzlich und welche Vor- und Nachteile (Komprimierungsgrad, Geschwindigkeit, Datensicherheit) hat das jeweilige?
- Wie groß ist das Team?
- Welche Aufgaben hast Lukas im Einzelnen?
- Ungefähre Auflage / Verkäufe des Spiels?
- Verwendete Programmiersprache(n) und Entwicklungsumgebungen
- Warum kam keine Game-Engine wie z.B. Unity 3D zu Einsatz?

SQL-Workshop mit Lukas (Datei *SQL_mit_Lukas.R* auf Ilias)

```
# Mit LEFT JOIN zwei Tabellen vereinen, mit UNION Einträge hinzufügen
# R Code in **Farbe**, SQL Code in **Farbe** koelsch <- read.csv("Koelsch.csv")
kirche <- read.csv("Kirche.csv")
print(koelsch)
print(kirche)

koelsch_kirche <- sqldf('
    SELECT kirche.Jahr, Koelsch AS "Kölschkonsum", Austritte AS "Kirchenaustritte"
    FROM koelsch
    LEFT JOIN kirche ON kirche.Jahr = koelsch.Jahr
    WHERE kirche.Jahr = 2020
    UNION ALL
    SELECT kirche.Jahr, Koelsch AS "Kölschkonsum", Austritte AS "Kirchenaustritte"
    FROM koelsch
    LEFT JOIN kirche ON kirche.Jahr = koelsch.Jahr
    WHERE kirche.Jahr <> 2020
')
```

1.3.3.8 Grundlagen

Für die Durchführung der in diesem Kurs behandelten praktischen Anwendungsfälle wird spezielle **Data-Analyse-Software** (*R*) eingesetzt, deren Bedienung und Funktionalität erweiterte Kenntnisse voraussetzt.

Darüber hinaus sind Kenntnisse in der **Tabellenkalkulation** (*Excel*) notwendig. Mit einem Tabellenkalkulationsprogramm können Daten einer ersten Analyse unterzogen werden; es lassen sich Abhängigkeiten zwischen Attributen entdecken, oder die Ergebnisse eines Daten-Analyse-Modells können analysiert beziehungsweise nachgearbeitet werden.

Ein **Text-Editor** (*Notepad++ (Windows), Brackets (MacOS), Kodex (iPadOS)*), der zudem die Arbeit mit Makros ermöglicht, ist ein weiteres nützliches Werkzeug in der Vor- sowie Nachbereitung einer Datenanalyse.

Nicht zuletzt sei darauf verwiesen, dass hin und wieder die eigene Entwicklung kleiner Programme (Funktionen für Berechnungen) für die Datenvorverarbeitung sowie die Analyseauswertung erforderlich sein kann. Dazu kann eine beliebige **Programmiersprache** (z.B. *R* oder *VBA*) herangezogen werden.

Vor- und Nachteile von R

- ist kostenlos da Open Source
- viele Erweiterungen verfügbar
- eigene R-Pakete (mit eigenen Daten oder Funktionen lassen sich mittels Github erstellen
- gute Dokumentation und viele Anleitungen (Community) im Netz
- für viele Anwendungszwecke geeignet (z.B. um SQL zu lernen)
- für alle Plattformen verfügbar (Windows, Mac, Linux, iPadOS)
- viele Konnektoren zu anderen Sprachen (z.B. zu Datenbanken)
- umfangreicher Funktionsumfang -> lange Einarbeitung notwendig
- setzt fortgeschrittene Statistik-Kenntnisse voraus
- nur eines unter vielen Statistik-Programmen (im Hinblick auf Bewerbungen)

1.4 Einführung und Erarbeitung der „Data Management“

- Grundlagen der Algorithmen
- Data Management Methoden
- Prozesse des Data Mining
- Techniken des Data Mining
- Visuelle und Maschinelle Verfahren des Data Mining
- Einführung und Übersicht der Algorithmen des Data Mining
- Interpretation der Verschiedenen Algorithmen und Reflektion der Ergebnisse des Data Mining
- Grundlagen des Web und Text Mining

1.4.1 3.a) Grundlagen der Algorithmen

1.4.1.1 Definition Algorithmus

*Ein **Algorithmus** ist eine formale Handlungsvorschrift zur Lösung von Instanzen einer bestimmten Problemklasse.* (Sauer, 2019)

Die Bezeichnung ist abgeleitet aus dem Namen des persischen Gelehrten Muhammad ibn Musa al-Chwarizmi.

1.4.1.2 Informelle Beispiele

In der Alltagssprache werden oft informelle Beispiele für Algorithmen angeführt, z. B.:

- Kochrezept
- Bauanleitung
- Schriftliches Rechnen
- Weg aus dem Labyrynth
- Zeichnen eines Kreises

1.4.2 3.b) Data Management Methoden

1.4.2.1 Definition Data Management

Data Management *befasst sich mit den Themen und Disziplinen deren Ziel es ist, Daten bereitzustellen und zu verwalten.*

Lean Data Management ist die Verbesserung dieser Disziplinen durch Vermeidung von Verschwendungen. **Verschwendungen** im Datenmanagement treten hauptsächlich auf, wenn Prozesse ineffizient laufen oder fehlerhafte Ergebnisse liefern, weil die Daten auf denen sie basieren, schlecht strukturiert oder gar falsch sind. Auswirkungen sind **steigende Prozesskosten** und Umsatzverluste.

1.4.2.2 Methoden des Data Managements

Vereinfachen Sie den Zugriff auf traditionelle und neue Daten. Mehr Daten bedeuten im Allgemeinen bessere Prädiktoren.

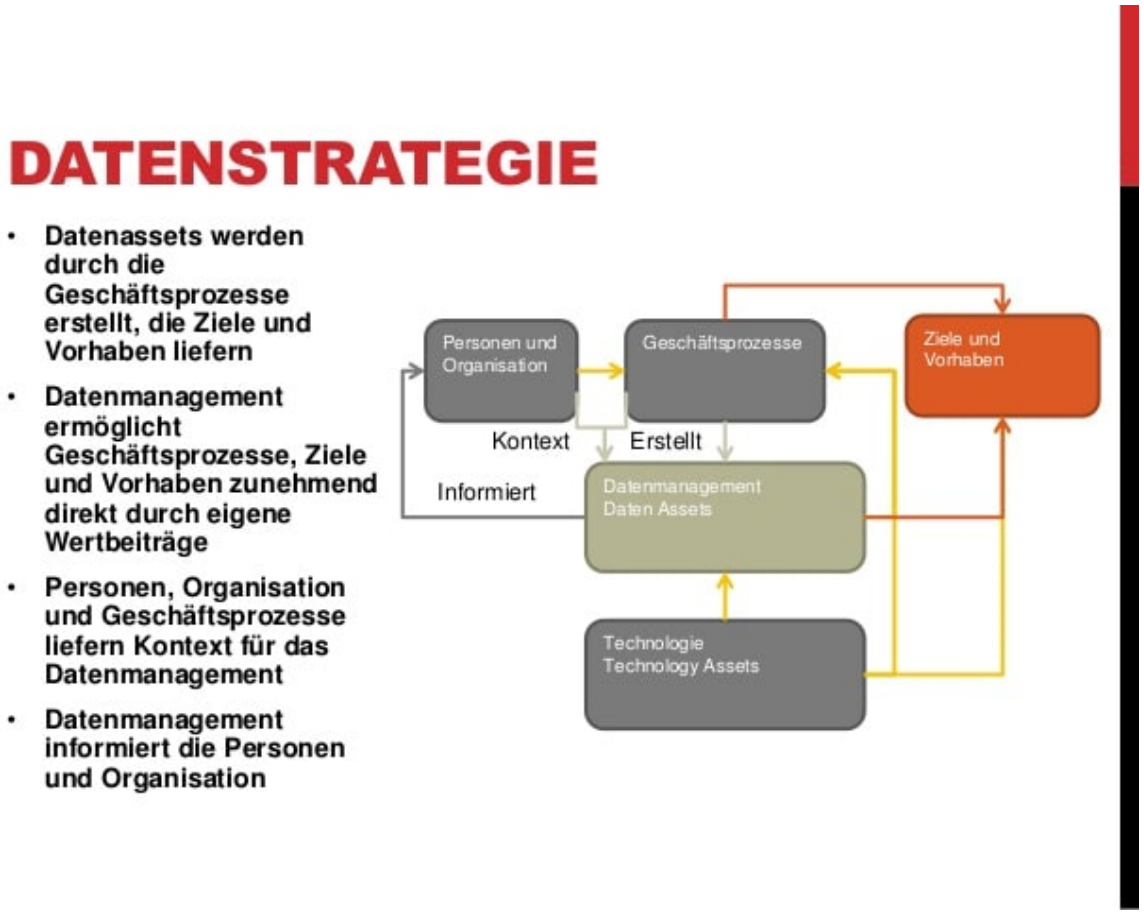
Stärken des Data Scientists mit fortschrittlichen Analysetechniken. Die Häufigkeitsanalyse hilft beispielsweise, Ausreißer und fehlende Werte zu identifizieren, die andere Maße wie Mittelwert, Durchschnitt und Median verzerren können.

Bereinigen der Daten, um Qualität in bestehende Prozesse zu integrieren. Bis zu 40 Prozent aller strategischen Prozesse scheitern an schlechten Daten.

Gestalten der Daten mit flexiblen Manipulationstechniken. Die Vorbereitung von Daten für Analysen erfordert das Zusammenführen, Transformieren, Denormalisieren und manchmal das Aggregieren der Quelldaten aus mehreren Tabellen in eine sehr breite Tabelle (Analytische Basistabelle, ABT).

Metadaten freigeben, über Datenverwaltungs- und Analysedomänen hinweg.

Abbildung 28: Abb. 3.b.1: Datenstrategie

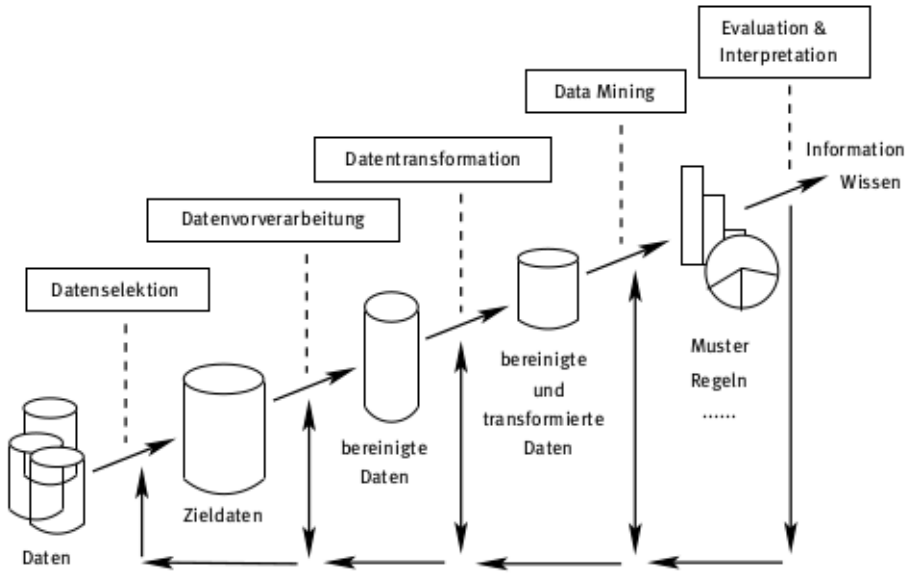


1.4.3 3.c) Prozesse des Data Mining

1.4.3.1 Definition Data Mining

Data Mining *ist ein analytischer Prozess, der eine möglichst autonome und effiziente Identifizierung und Beschreibung von* ***interessanten Datenmustern*** *aus großen Datenbeständen ermöglicht.*

Abbildung 29: Abb. 3.c.1: *Ablauf eines Data-Mining-Prozesses*

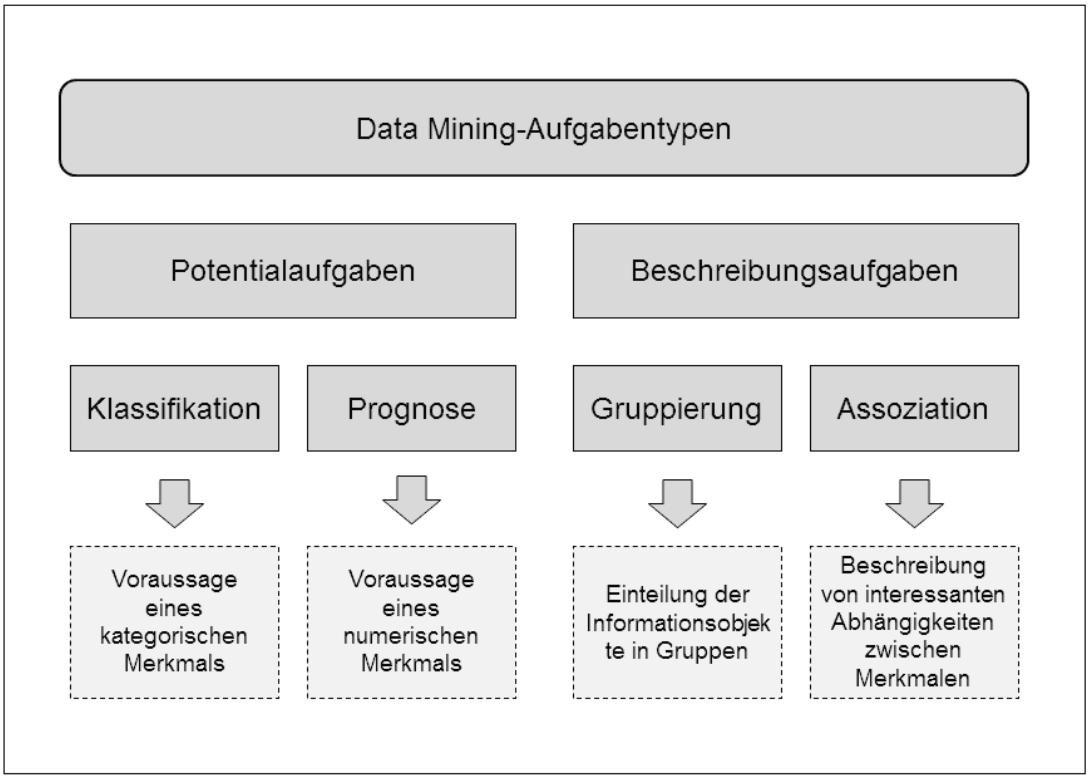


1.4.3.2 Methoden

Die **Methoden** des Data Minings lassen sich grundsätzlich in die Gruppen **Klassifikation**, **Prognose**, **Segmentierung** und **Abhängigkeits- entdeckung** enteilen.

1.4.4 3.d) Techniken des Data Mining

Abbildung 30: Abb. 3.d.1: Data Mining Aufgabetypen



1.4.4.1 Grundsätzliches

Die **Visualisierung** von Datenanalyse Ergebnissen reicht in der Regel alleine nicht aus. Bei der vorherigen **Datentransformation** mittels die **Schlüsselverben** werden wichtige **Variablen** ausgewählt und die wichtigen **Beobachtungen** herausgefiltert. Zudem können neue Variablen erstellt, sowie sinnvolle **Zusammenfassungen** berechnet werden (Wickham & Grolemund, 2017).

1.4.5 3.e) Visuelle und Maschinelle Verfahren des Data Mining

1.4.5.1 Visuelle Verfahren

Unter zur Hilfename von Tools, können **Data Mining Prozesse** und **Algorithmen** einfach konfiguriert und die Ergebnisse zur leichten Verständlichkeit visualisiert werden.

Ein Beispiel ist ***Knime.*** Die Datenmanipulation wird visuell anhand einzelner **Funktionsknoten** dargestellt und kann individuell beschriftet werden. Dadurch werden die einzelnen Prozessschritte greifbar und der komplette Analyseprozess dadurch langfristig **nachvollziehbar** und **reproduzierbar**.

Abbildung 31: Abb. 3.e.1: Data Mining Tools

Maschinelle Verfahren (Datei *mining.py* auf Ilias)

Mittels **Programmierung** (oder Scripting) ist es möglich Datenauswertungen durch die Ausführung von Code zu erstellen. Für die Datenanalyse wird am häufigsten die Programmiersprache ***Python*** verwendet.

```
# Bibliotheken laden
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
import seaborn as sns
from matplotlib import rcParams

# Daten sichten
```

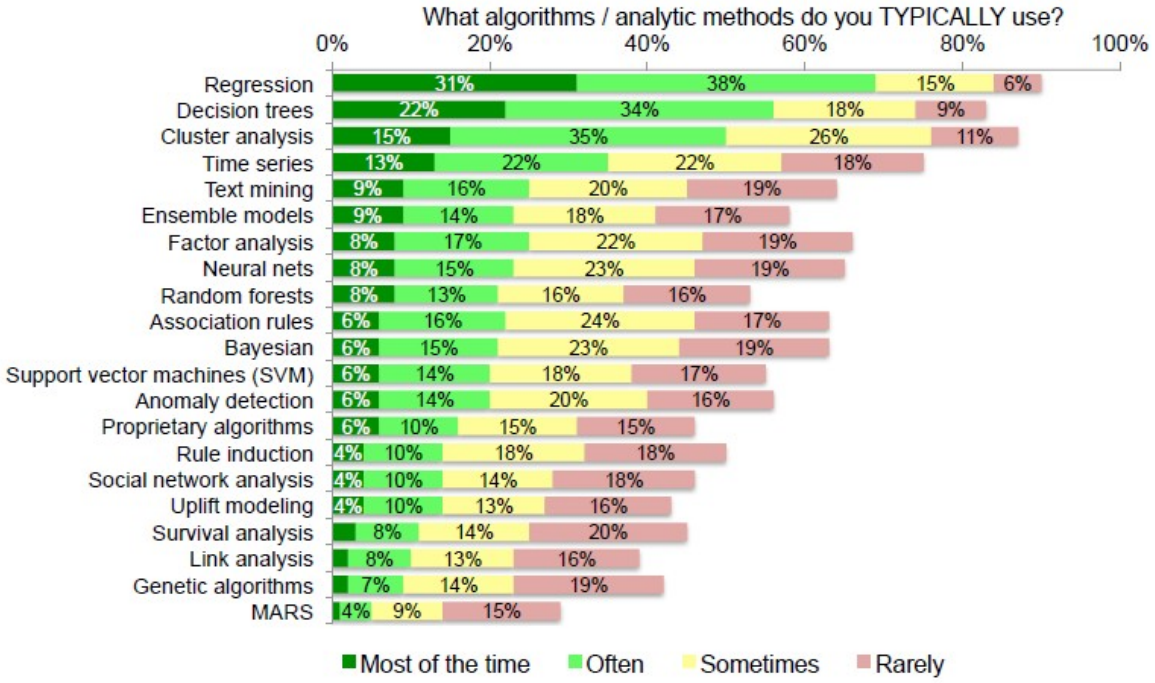
```
# read data from csv file
df = pd.read_csv('./Daten/kc_house_data.csv')
print(df.describe())
```

1.4.5.2 Die Algorithmen

Zur Mustererkennung in Daten kommen **Algorithmen** aus der **Statistik**, **Mathematik** und Informatik zum Einsatz.

1.4.6 3.f) Einführung und Übersicht der Algorithmen des Data Mining

Abbildung 32: Abb. 3.e.1: Data Mining Algorithmen in der Häufigkeit der Anwendung



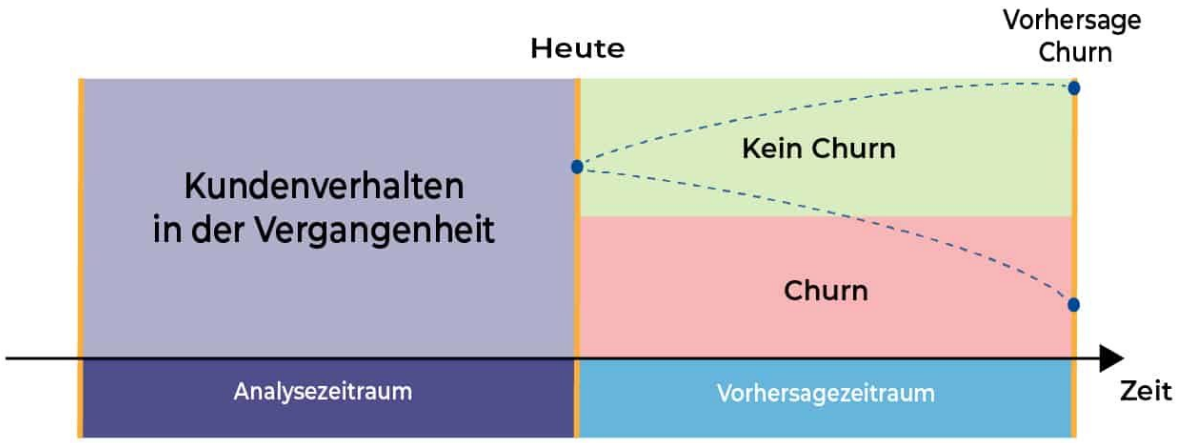
1.4.6.1 Anwendungsbeispiele

- **Customer Relationship Management**
  - CLV-Vorhersagen
  - Cross- und Upselling Optimierung
  - Churn Prediction
- **Produktion und Logistik**
  - Optimierung der Supply Chain
  - Predictive Maintenance
  - Prognose von Nachfrage
- **Medizin / Forschung Allgemein**
  - Erkennung von Krankheiten auf Bildern
  - Therapievorschläge
  - Identifizierung von Anomalien

1.4.7 3.g) Interpretation der

Verschiedenen Algorithmen und Reflektion der Ergebnisse des Data Mining

Abbildung 33: Abb. 3.g.1: Abwanderungsvorhersage



1.4.7.1 Data Mining Algorithmen

- **Lineare Regression**
- **Logistische Regression**
- **Entscheidungsbäume**
  - ID3, C4.5, CART, CHAID, MARS
- **Support Vector Maschine (SVM)**
- **Clustering Algorithmen**
  - K-Means, DB-Scan, Self Organized Maps, Hierarchisches Clustering
- **Random Forest**
- **Gradient Boosted Trees**
- **künstliche Neuronale Netze**
- **Recommernder System**
- **Assoziations Analysen**
  - Apriori Algorithmus, FPgrowth

1.4.8 3.g) Interpretation der Verschiedenen Algorithmen und Reflektion der Ergebnisse des Data Mining

1.4.8.1 Auswahl des passenden Algorithmus

- **Klassifikation**
  - z.B. Kundengewinnung
    - *Entscheidungsbaum / Decision tree* -> Reaktion von Kunden

- **Prognose**
  - z.B. Kundenwert eines Unternehmens (B2B)
    - *Text Mining (z.B. Webseite)* -> steigender Wert
- **Gruppierung**
  - z.B. Kundenverhalten
    - *Cluster Analyse (K-Means) -> Reaktion von Kunden
- **Assoziation**
  - z.B. Regeln unabhängig vom Kunden (ist evtl. nicht bekannt)
    - *Warenkorbanalyse / Basket Analysis* -> steigender Absatz

1.4.8.2 Text Mining

Text Mining befasst sich mit der Analyse von Textdokumenten. Texte sind im Gegensatz zu Datenbanken und Web-Seiten unstrukturiert. Texte genügen natürlich auch Strukturvorgaben wie einer Grammatik, […] letztendlich sind sie unstrukturiert.

Häufig ist man an einer Klassifizierung eines Dokuments, beispielsweise nach Themengebiet oder fachlichem Niveau, interessiert. Ein Ansatz ist, zunächst relevante mengebiet oder fachlichem Niveau, interessiert. Ein Ansatz ist, zunächst relevante Begriffe aus dem Dokument zu extrahieren, um anhand dieser das Dokument einzuordnen. Eine zweite mögliche Anwendung ist die Quantifizierung der Ähnlichkeit von Dokumenten.

[…] Da die Daten nur unstrukturiert vorliegen, ist der erste Schritt das Extrahieren der interessanten Informationen aus den Textdokumenten. Dies können beispielsweise Schlüsselwörter, die Häufigkeitsverteilung von Begriffen oder eine erste Kategorisierung oder hierarchische Gruppierung sein. Irrelevante Wörter (stop words) müssen herausgefiltert und Abkürzungen erkannt werden (Cleve & Lämmel, 2020).

1.4.9 3.h) Grundlagen des Web und Text Mining

1.4.9.1 Web Mining

Anwendungen des **Data Minings**, die das Internet als Datenquelle für die Mustererkennung heranziehen, werden unter dem Themengebiet des Web Minings zusammenge-fasst. In Abhängigkeit von der inhalts- oder nutzungsorientierten Analyse des World Wide Web (WWW) lassen sich die Teilgebiete **Web Content Mining**, **Web Structure Mining** und **Web Usage Mining** voneinander abgrenzen.

Web Content Mining befasst sich mit der Analyse von den im WWW befindlichen Daten. Dazu gehören textuelle und multimediale Informationen jeglichen Formats. […]

Web Structure Mining untersucht Links und Link-Strukturen. Dadurch können wichtige beziehungsweise populäre Seiten gefunden werden. […]

Web Usage Mining dagegen befasst sich mit dem Verhalten von Internet-Nutzern. Bei dieser Form des Web Mining werden Data-Mining-Methoden auf die Protokolldateien des Webservers angewandt, um Aufschlüsse über Verhaltensmuster und Interessen der Online-Nutzer zu erhalten. […]

1.5 Erarbeiten IT-technischer Rahmenbedingungen

Erarbeiten IT-technischer Rahmenbedingungen:

- Anleitung / Installation der Open Source Lösung auf den Rechner der Studenten und Konfiguration von *Microsoft Excel (VBA)*
- Einführung und Anleitung der Applikation *R*
- Datenerfassung sowie das Einlesen der Beispieldaten zur ersten Analyse

1.5.1 4.a) Anleitung / Installation der Open Source Lösung auf den Rechner der Studenten und Konfiguration von *Microsoft Excel (VBA)***

1.5.1.1 Excel Datei mit Makro-Code (VBA) anpassen

- Entwicklertools aktivieren:
- auf der Registerkarte Datei **zu** Optionen -> **Menüband anpassen**
- **Menüband anpassen** und unter **Hauptregisterkarten** das Kontrollkästchen **Entwicklertools** aktivieren
- Erläuterung des Fragebogens
- Excel Datenblätter
- Fragebogen -> Maske zum erheben der Daten
- Daten -> Rohdaten des einzelnen Fragebogens
- Datenspeicherung -> Rohdaten aller Fragenbögen -> Speicherung in CSV-Datei oder XML-Datei möglich
- VBA Code -> auf Menüband auf **Entwicklertools** und links auf

Visual Basic klicken und die entsprechenden Elemente anpassen

Installation der (Text-)Editoren Windows Für das Microsoft Windows Betriebssystem hat sich der kostenlose Editor ***Notepad++*** in Funktionsumfang, Performance und Anwenderfreundlichkeit einen deutlichen Vorsprung gegenüber der Konkurrenz erarbeitet. *Notepadd++* herunterladen unter:https://notepad-plus-plus.org/downloads/

1.5.1.2 MacOS

Für das Apple MacOS Betriebssystem kann man sehr gut auf den kostenlosen Editor ***Brackets*** von Adobe zurückgreifen. *Brackets* herunterladen unter: https://brackets.de.uptodown.com/macUnterhttps://registry.brackets.io lassen sich eine Vielzahl an Erweiterungen downloaden und unter *Datei – Erweiterungsverwaltung* installieren. Für die Vorlesung sind wichtig:

- *Beautify* (Code formatter, z.B. XML-Struktur)
- *Documents-Toolbar* (Toolbar mit Tabs)
- *BracketstoIX* (Top menu IX mit vielen Funktionen)
- *Excel to Table* (Excel Tabellen in Brackets einfügen)
- *Newline* (konvertiert den Zeilenumbruch LF (Mac / Linux) <–> CRLF (Windows))

1.5.1.3 iPadOS

Für das Apple iPad kann der Editor ***Kodex*** über den Apple App Store installiert werden.

4. Erarbeiten IT-technischer Rahmenbedingungen 4.a) Anleitung / Installation der Open Source Lösung auf den Rechner der Studenten und Konfiguration von *Microsoft Excel (VBA)*

1.5.1.4 Installation von *R*

R herunterladen unter: https://cran.r-project.org/bin/macosx/(MacOS) oder https://cran.r-project.org/bin/windows/base/(Windows)

1.5.1.5 Inhalte und Anwendungsgebiet

In dem Installations-Paket sind alle notwendigen Basisfunktionen enthalten (*base-package*) siehe https://stat.ethz.ch/R-manual/R- devel/library/base/html/00Index.html.

R ist eine vollständige, **objekt-orientierte Programmiersprache** und kann vielfältige Operationen mit einer eigenen Syntax ausführen. Häufig wird *R* zur statistischen Datenverarbeitung eingesetzt.

Im ***Comprehensive R Archive Network*** (kurz: *CRAN*), einem Netz aus Webservern, werden über 2000 Funktionserweiterungen, Datensätze und Code in Form von Paketen bereitgestellt.

4. Erarbeiten IT-technischer Rahmenbedingungen 4.a) Anleitung / Installation der Open Source Lösung auf den Rechner der Studenten und Konfiguration von *Microsoft Excel (VBA)*

1.5.1.6 *R* auf dem *iPad* (oder *iPhone*) installieren

Auf dem *iPad* im App Store *TestFlight* installieren. Dann mittels des Links die aktuelle Beta von *iSH-AOK* installieren: https://testflight.apple.com/join/X1flyiqE

Um *R* auf dem *iPad* sinnvoll nutzen zu können, kann die *Dateien* App in den **Splitscreen** an den Bildschirmrand hinzugefügt werden.

In der Dateien App muss vorher das **Laufwerk** *ish* **aktiviert** werden. Die Ausgabe-Dateien liegen unter ish `/home/ish/R/`

Nach der Ausgabe eines Diagrammes zum **Aktualisieren** des Ordnerinhaltes einmal zurück und wieder rein gehen.

Abbildung 34: Abb. 4.a.1: *R* auf *iPad* mit *Dateien* App im Splitscreen

1.5.1.7 *R Studio* und *Xquartz* installieren

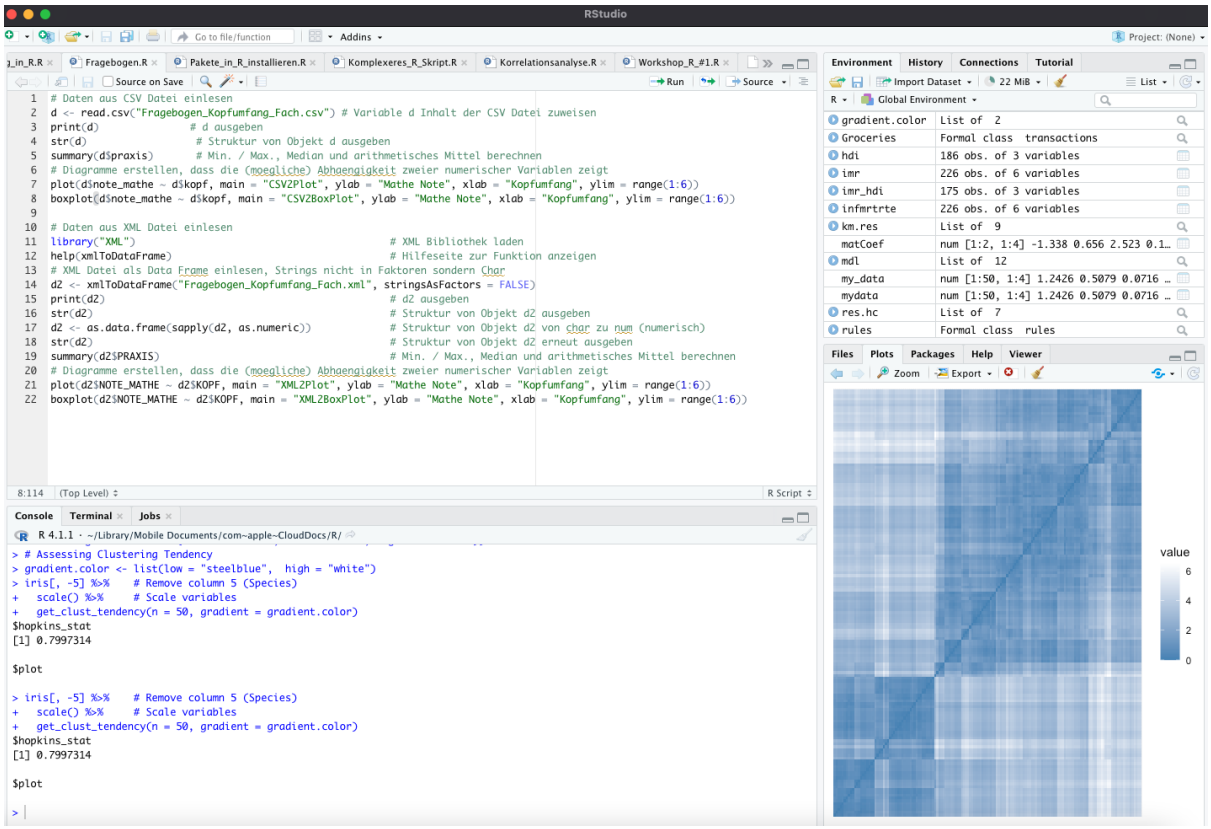
RStudio herunterladen unter (*R* muss vorher installiert sein): https://www.rstudio.com/products/rstudio/download/#download(Windows / MacOS)

XQuartz herunterladen unter: https://www.xquartz.org(MacOS)

Unter *RStudio* lassen sich komplexe Skripte ausführen, das Paketmanagement ist vereinfacht sowie ist die grafische Ausgabe möglich.

XQuartz bietet unter *MacOS* eine X.Org (X Window System) Version. X.Org ist unter Linux / Unix die grafische Benutzeroberfläche.

Abbildung 35: Abb. 4.a.2: *R-Studio*



1.5.2 4.b) Einführung und Anleitung der Applikation *R*

Erste Schritte mit *R* Grundsätzliches

- R unterscheidet zwischen Groß- und Kleinschreibung, d.h. *Oma* und *oma* sind verschiedene Wesen
- R verwendet den . (Punkt) als Dezimaltrennzeichen
- Fehlende Werte werden in R durch *NA* gekennzeichnet
- Kommentare (nicht auszuführender Code) werden mit dem # (Hashtag) eingeleitet
- Variablennamen sollten mit Buchstaben beginnen, sowie können sie Zahlen und _ enthalten
- Um den Inhalt einer Variablen auszulesen, geben wir einfach den Namen an
- Text muss in “ “ oder ’ ‘ (doppelte oder einfache Anführungszeichen) gesetzt werden

1.5.2.1 R Code

```
# Kommentare / nicht ausführbarer Code werden mit der Raute bzw. hashtag eingeleitet
temperatur <- -10

# der Variablen temperatur den Wert -10 zuweisen
print(temperatur)

# Variable temperatur ausgeben
text <- "Dies ist ein Text"

# Text einer Variablen zuweisen
```

```r
print(text)

# Variable text ausgeben
print(text)

# Variable einer anderen zuweisen

text <- temperatur
```

1.5.3 4.b) Einführung und Anleitung der Applikation *R*

1.5.3.1 Einführung in *R*

```r
# Funktionen ausführen
help(c)

# Hilfe für Funktion c aufrufen, Alternativ ?c
temperatur <- c(18, 22, 23, 24)
# Funktion c: fasst Werte in einer Liste zusammen

# ineinander verschachtelte Funktionen
temperatur <- c(18, 22, as.integer(runif(1, -12, 40)), 24)

# Variable mit dem Mittelwert aus temperatur füllen
ein_mittelwert <- mean(temperatur)

# 2 weitere mögliche Schreibweisen
ein_mittelwert <- mean(x = temperatur, trim = 0, na.rm = FALSE)
ein_mittelwert <- mean(temperatur, 0, FALSE)

# Logische Prüfungen
2 < 3 # Kleiner-Prüfung
2 > 3 # Größer-Prüfung
2 <= 3 # Kleiner-Gleich-Prüfung
2 >= 3 # Größer-Gleich-Prüfung
2 == 3 # Prüfung auf Gleichheit
2 != 3 # Prüfung auf Ungleichheit

temperatur < 8 & temperatur > -12 # & verknüpft Bedingungen
temperatur < 8 | temperatur > 0 # | oder Statement
```

1.5.3.2 Einführung in *R* – Diagramme (Datei *Diagramme.R* auf Ilias)

```r
# Library ourdata laden
library(ourdata)

# Balken-Diagramm, veranschaulicht die Verbindung zwischen einer
# numerischen und einer kategorialen Variablen. Das Balkendiagramm
# stellt jede Kategorie als Balken dar und spiegelt den entsprechenden
```

```r
# numerischen Wert mit der Größe des Balkens wider.
barplot(koelsch$Koelsch)

# Streu-Diagramm, zeigt zwei numerische Variablen mit Punkten an,
# wobei jeder Punkt den Wert einer Variablen auf der x-Achse und den
# Wert der anderen Variablen auf der y-Achse darstellt.
plot(kirche$Austritte ~ koelsch$Koelsch)

# Boxplot-Diagramm, zeigt die Verteilung einer numerischen Variablen
# basierend auf fünf zusammenfassenden Statistiken: minimaler
# Nicht-Ausreißer; erstes Quartil; Median; drittes Quartil; und maximaler
# Nicht-Ausreißer. Außerdem zeigen Boxplots die Positionierung von
# Ausreißern und ob die Daten verzerrt sind.
boxplot(koelsch$Koelsch)
```

1.5.3.3 Einführung in *R* – Diagramme (Datei *Diagramme.R* auf Ilias)

```r
# Dichte-Diagramm, zeigt die Verteilung einer numerischen Variablen
# über ein kontinuierliches Intervall. Peaks eines Dichte-Diagramms
# visualisieren, wo sich die Werte numerischer Variablen konzentrieren.
plot(density(koelsch$Koelsch))
# Heatmap-Diagramm, visualisiert einzelne Werte einer Matrix mit Farben.
# Häufigere Werte werden typischerweise durch hellere rötliche Farben
# angezeigt und weniger häufige Werte werden typischerweise durch
# dunklere Farben angezeigt.
heatmap(cbind(fragebogen$alter, fragebogen$praxis))

# Linien-Diagramm, visualisiert Werte entlang einer Sequenz
(z.B. über die Zeit). Liniendiagramme bestehen aus einer x-Achse
# und einer y-Achse. Die x-Achse zeigt normalerweise die Sequenz
# und die y-Achse die Werte an, die jedem Punkt der Sequenz entsprechen.
plot(1:length(fragebogen$alter), fragebogen$alter, type = "l")
```

1.5.3.4 Einführung in *R* – Diagramme (Datei *Diagramme.R* auf Ilias)

```r
# Histogramm, gruppiert kontinuierliche Daten in Bereiche und stellt diese Daten als
# Balken dar. Die Höhe jedes Balkens zeigt die Anzahl der Beobachtungen in jedem Bereich an
hist(fragebogen$alter)
# Paar-Diagramm, ist eine Diagrammmatrix, die aus Streudiagrammen für jede Variablenkombinat
# eines Datenrahmens besteht.
pairs(data.frame(fragebogen$kopf, fragebogen$note_mathe))

# Ein QQplot (oder Quantil-Quantil-Diagramm), bestimmt ob zwei Datenquellen aus einer gemein
# Verteilung stammen. QQplots ziehen die Quantile der beiden numerischen Datenquellen gegene
# Wenn beide Datenquellen aus derselben Verteilung stammen, fallen die Punkte auf einen Wink
qqplot(fragebogen$kopf, fragebogen$note_mathe)

# Venn-Diagramm (oder Primärdiagramm; Mengendiagramm; Logikdiagramm), veranschaulicht alle m
# logischen Beziehungen zwischen bestimmten Datenmerkmalen. Jedes Merkmal wird als Kreis dar
```

```
# wobei überlappende Teile der Kreise Elemente darstellen, die beide Merkmale gleichzeitig a
library("VennDiagram")
plot.new()
draw.single.venn(area = 10)
```

1.5.3.5 Einführung in *R* – Diagramme (Datei *Diagramme.R* auf Ilias)

```
# Kreis- oder Torten-Diagramm, sind zwar weit verbreitet, weisen jedoch einige Nachteile au
# a) Unterschiede zwischen den Anteilswerte sind weniger gut erkennbar,
# da dazu die Fläche der Kreissegmente verglichen werden muss
# b) bei vielen Kategorien wird die Darstellung schnell unübersichtlich
# c) sehr kleine Anteilswerte können oftmals nicht im Kreisdiagramm dargestellt werden
# Aufgrund dieser Nachteile bietet sich die Verwendung von Kreisdiagramme nur in
# selten Fällen an, meist liefern Punkt- oder Balkendiagramme bessere Darstellungen.
pie(table(iris$Species),
    labels=c("Setosa", "Versicolor", "Virginica"),
    col=grey(c(0.1, 0.4, 1)
)
)

# Alle Diagramme sind ganz einfach mittels der Funktion plotter() aus dem
# R-Paket ourdata ausführbar. Ohne Angabe von Parametern, wird ein Menü
# Nutzereingaben erfragen und anschließend das gewünschte Diagramm ausgeben.
library(ourdata)
plotter()
```

1.5.3.6 Einführung in *R* – Diagramme (Datei *Diagramme.R* auf Ilias)

```
# Spiderweb-Diagramm, erlaubt die Darstellung numerischer Werte mehrerer Variablen. Im unter
# sind die mittleren Längen und Breiten der Blütenblätter der Schwertlilie eingezeichnet. Di
# den Wert für die entsprechende Variable an. Radial-Plots können so eine größere Zahl von m
# übersichtlich darstellen. Durch die charakteristischen Formen, die das durch die Werte geb
# kann diese Art von Grafik auch zum Vergleich zwischen verschiedenen Subpopulationen oder V
library(plotrix)
mean.iris <- c(mean(iris$Sepal.Length), mean(iris$Sepal.Width), mean(iris$Petal.Length), mea
radial.plot(lengths=mean.iris,

rp.type="p",
line.col="blue",
labels=names(iris$Sepal.Length),
label.prop=1.1,
radlab=F,
radial.lim=c(0,6),
poly.col="transparent",
grid.col=gray(0.5),
grid.bg=gray(0.95),
grid.left=T,
grid.unit="cm",
point.symbols=20,
```

```
point.col="red",
lty="solid",
lwd=1,
mar=c(4,4,5,5),
main="Durchschnittliche Blattlängen")
```

Fragebogen mit *R* auswerten (Datei *Fragebogen.R* auf Ilias)

```
getwd()

# aktuelles Arbeitsverzeichnis ausgeben, falls nicht richtig dann:

setwd("/<Userordner>/<name>/R") # Arbeitsverzeichnis, Userordner -> Mac: „Users" / iPad: „ho

# Daten aus CSV Datei einlesen\
d <- read.csv("Fragebogen_Kopfumfang_Fach.csv") # Variable d Inhalt der CSV Datei zuweisen p

# d ausgeben

summary(d$praxis) # Min. / Max., Median und arithmetisches Mittel berechnen # Diagramme erst

# Daten aus XML Datei einlesen d2 <- xmlToDataFrame("Fragebogen_Kopfumfang_Fach.xml")
# XML Datei als Data Frame einlesen print(d2)
# d2 ausgeben summary(d2$PRAXIS) # Min. / Max., Median und arithmetisches Mittel berechnen
# Diagramme erstellen, dass die (mögliche) Abhängigkeit zweier numerischer Variablen zeigt
plot(d2$NOTE_MATHE ~ d2$KOPF, ylab = "Mathe Note", xlab = "Kopfumfang", ylim = range(1:6)) h

# Skriptdatei (aus R Studio) in R Konsole ausführen
source("Fragebogen.R")
```

Pakete in *R* installieren (Datei *Pakete_in_R_installieren.R* auf Ilias)

```
# XML Paket installieren
install.packages("XML")

# Paket installieren

library("XML")

# Paket laden, um es ausführen zu können
library("methods")

# abhängiges Paket laden

# Devtools installieren und laden und dann ourdata mittels github installieren und laden
install.packages("devtools", dependencies = TRUE)
library(devtools)

install_github("DrBenjamin/ourdata")
library(ourdata)
```

```
ourdata()
help(ourdata)
print(fragebogen)
```

Im Skript (*Pakete_in_R_installieren.R*) werden alle weiteren benötigten Pakete installiert, Zeile für Zeile ausführen und ggfs. die Installation mit Yes bestätigen.

1.5.4 4.c) Datenerfassung sowie das Einlesen der

Beispiel-Daten zur ersten Analyse

Komplexeres R Skript (Datei *Komplexeres_R_Skript.R* auf Ilias)

```
# Einfacher Vektor, die Funktion c kombiniert Werte in einen Vektor oder eine Liste
# fruitVec wird mit 4 Werten (Strings) gefüllt
fruitVec <- c("Apfel", "Banane", "Orange", "Birne")

# Einfache While-Schleife
# Funktion length: gibt die Länge eines/r Vektors / Liste aus
# Der Vektor fruitVec kann wie ein Array angesprochen werden, um die einzelnen Werte auszule
pos <- 1
while (pos <= length(fruitVec)) {
print(paste0("Die Frucht Nr. ", pos, " ist : ", fruitVec[pos]))
pos <- pos + 1
}

# Einfache For-Schleife
# Funktion paste0 verbindet Strings
# Funktion which ermittelt die Position eines Wertes im Vektor (bzw. Array)
for (fruit in fruitVec) {
    print(paste0("Die Frucht Nr.", which(fruit == fruitVec), " ist: ", fruit))
}
```

1.5.5 4.c) Datenerfassung sowie das Einlesen der

Beispiel-Daten zur ersten Analyse R Skript zum verarbeiten des Dating-Datensatzes (Datei *Dating.R* auf Ilias)

```
# Analyse des Dating-Datensatzes aus dem R-Paket 'pradadata'
# Die Daten wurden zuvor mittels Excel bereinigt (für die Analyse nicht sinnvolle
# Daten wurden entfernt, sowie von nominalen zu metrischen Werten transformiert
# z.B. char-Typen wie 'yes' und 'no' zu numerischen Typen wie '1' und '2'

# bereinigte Datei einlesen
Dates <- read.csv("./Daten/Dates.csv")

# selektieren von Vektoren / Spalten
Dates <- subset(Dates, select = c(car, Age, comm_type))
```

```
# Statistische Auswertungen
plot(Dates\$car \~ Dates\$comm_type, main = "Dating Datensatz", ylab = "Car", xlab = "Kommun
xaxt = "n", yaxt = "n")
axis(1, at = c(1, 2)) # Achsenbeschriftung manuell hinzufügen - da jeweils nur '1' und '2' r
axis(2, at = c(1, 2)) # sind Zwischenwerte auf der Achse irreführend
abline(lm(Dates\$car \~ Dates\$comm_type)) # Regressionsgerade
barplot(table(Dates\$car, Dates\$comm_type)) # Ausgabe Häufigkeit
```

Beispiel-Daten zur ersten Analyse R Skript zum verarbeiten des Dating-Datensatzes (Datei *Dating.R* auf Ilias)

```
# Funktion 'transformer' aus dem 'ourdata' R-Package benutzen, um char-Typen in numierische
# zu konvertieren, z.B. 'female' to '1' und 'male' zu '2' -> Alternative zur Excel Funktion
# Installieren oder Updaten von 'ourdata' R-Package, nur wenn noch nicht geschehen!
library(devtools)
install_github("DrBenjamin/ourdata", force = TRUE)

# Bibliotheken laden (nach Neustart von RStudio immer notwendig!)
library(ourdata)
library(pradadata)

# Hilfe-Seiten lesen!
help(ourdata)
help(transformer)
help(fragebogen)

# strawperson transformieren, char type -> numeric z.B. 'A' zu '1' und 'B' zu '2' etc.
trans_data <- transformer(subset(dating, select = strawperson), verbose = TRUE)
# den transformierten Vector (Werte) der Liste (data frame) zuweisen
dating$strawperson <- trans_data
# Statistische Auswertungen
plot(dating$car ~ dating$comm_type)
heatmap(cbind(dating$car, dating$comm_type))
```

Beispiel-Daten zur ersten Analyse R-Skript zum Verarbeiten des Fragebogen-Datensatzes auf dem iPad (Datei Fragebogen_iPad.*R* auf Ilias)

```
# Diagrammausgabe auf dem iPad (keine grafische Schnittstelle) in PDF oder PNG Dateien

# Datensätze aus den Vorlesungen laden
library(ourdata)
print(kopfumfang) # Ergebnisse des Fragebogens ausgeben
str(kopfumfang) # Struktur von data frame kopfumfang ausgeben
summary(kopfumfang$praxis)# Min. / Max., Median und arithmetisches Mittel berechnen

# Diagramme erstellen und als PDF speichern
pdf("Plots.pdf")

# PDF Export auf iPad
```

```
plot(d$note_mathe ~ d$kopf, ylab = "Mathe Note", xlab = "Kopfumfang", ylim = range(1:6))
boxplot(d$note_mathe ~ d$kopf, ylab = "Mathe Note", xlab = "Kopfumfang", ylim = range(1:6))
dev.off()

# PDF Export beenden (notwendig, sonst leere Datei!)
# Diagramm (nur eines möglich!) erstellen und als PNG speichern
png("Plot.png")

# PNG Export auf iPad
plot(d$note_mathe ~ d$kopf, ylab = "Mathe Note", xlab = "Kopfumfang", ylim = range(1:6))
dev.off()
# PNG Export beenden (notwendig, sonst leere Datei!)
```

Beispiel-Daten zur ersten Analyse.

1.5.5.1 Einlesen der Datensätze aus den Vorlesungen

RStudio ist eine vollständige **IDE** und man kann ohne zusätzliche Tools Software entwickeln und diese für andere zur Verfügung stellen.

Das R-Paket *ourdata* mit den **Datensätzen** und **Funktionen** aus dem Unterricht kann auf *github.com* eingesehen werden (*README.md* für Anleitung lesen):

- Repository *ourdata*

1.5.5.2 Installation und Anzeige von Hilfeseiten

```
library(devtools) # Library devtools laden, um install_github ausführen zu können
install_github("DrBenjamin/ourdata", force = TRUE) # Installation des R-Paketes
library(ourdata) # Library ourdata laden
help(ourdata). # Hilfeseite zum R-Paket anzeigen lassen
help(combine) # Hilfeseite zur Funktion *combine* anzeigen lassen
help(fragebogen) # Hilfeseite zum data frame *fragenbogen* anzeigen lassen
```

R-Analyse-Workshop** Verwende beliebige Daten (aus dem Internet), um eine weitere **Korrelations- analyse** von *zwei Datensätzen mit unterschiedlicher Quelle* durchzuführen.

Mögliche Datensätze (zwei numerische Werte, sowie ein Matching nötig)

- Gesundheitsdaten (z.B. Lebenserwartung, Alkoholkonsum, Covid 19, etc.)
- Sozialdaten (z.B. Scheidungsrate, Singlequote, Lebensgewohnheiten, etc.)
- Technik (z.B. Prozessoren- oder KI-Entwicklung, Datenvolumen Internet, etc.)
- Sport (z.B. Fußballdaten, Wettspiel, Drivelänge beim Golf, etc.)
- Freizeit (z.B. Netflix & Co, Eiskonsum, Computerspiel, etc.)

1.5.5.3 Ablauf

1. 1. Datensatz suchen (z.B. auf Datenportalen wiehttps://stats.oecd.org/)
2. 2. Datensatz suchen (z.B. auf Webseiten wie Statista (https://de.statista.com) aufbereiten (Dezimaltrennung, Variablennamen, Spalten bereinigen)
3. Matching bzw. Bezugspunkt festlegen (z.B. in beiden Listen das Land)
4. Datensätze vereinen (mit Hilfe von Excel, VBA oder R)
5. Statistische Auswertung in R (Korrelationsanalyse)

R-Programmier-Workshop** (Datei *R_Workshop.R* auf Ilias) **Aufgabe:** Programmiere die VBA-Funktion Zusammenführen aus der Excel „Fragebogen“ in R nach, so dass:

- R direkt aus den beiden CSV-Dateien
- IMR: https://www.cia.gov/the-world-factbook/field/infant-mortality- rate/country-comparison
- HDI: https://worldpopulationreview.com/country-rankings/hdi-by-country

die passenden Spalten (Land, IMR und HDI) zusammenfügt

- alle Länder, zu denen keine HDI Angaben vorhanden sind gelöscht werden
- die Diagramm-Ausgaben die identischen Ergebnisse erzielen wie mit der Excel Zusammenführung

1.5.5.4 Hilfsmittel

- evtl. am verfügbaren VBA Code orientieren
- Google Suche (Stichwort „R Funktion xyz“)
- Stackoverflow (https://stackoverflow.com)- R-Hilfe-Seiten help(<Funktionsname>)

1.6 Methodik

- Theoretische Grundlagen der Korrelation Analyse
- Anwendung der Korrelations-Analyse mit Beispieldaten

1.6.1 5.a) Theoretische Grundlagen der Korrelation Analyse

Will man einen **Zusammenhang** zwischen **zwei metrischen Variablen** untersuchen, zum Beispiel zwischen dem Alter und dem Gewicht von Kindern, so berechnet man eine **Korrelation**. Diese besteht aus einem *Korrelationskoeffizienten* (*rho* bei Spearman) und einem *p-Wert*.

Der **Korrelationskoeffizient** gibt die Stärke und die Richtung des Zusammenhangs an. Er liegt zwischen *-1* und *1*. Ein Wert nahe *-1* bezeichnet einen **starken negativen Zusammenhang** (z.B. „*Mehr zurückgelegte Strecke mit dem Auto, weniger Treibstoff im Tank*“). Ein Wert nahe *1* spricht für einen **starken positiven Zusammenhang** (z.B. „*mehr Futter, dickere Kühe*“). Kein Zusammenhang besteht, wenn der Wert nahe *0* liegt.

Der **p-Wert** sagt aus, ob es einen **signifikanten Zusammenhang** gibt. p-Werte kleiner als *0,05* werden als **statistisch signifikant** bezeichnet.

1.6.2 5.b) Anwendung der Korrelations-Analyse mit Beispieldaten

Korrelationsanalyse in R (Datei *Korrelationsanalyse.R* auf Ilias) am Beispiel der IMR (Infant Mortality Rate) und des HDI (Human Development Index)

```
# Bibliothek openintro (enthält Data Sets) laden
library(openintro)
print(infmortrate)
# Deutung des Datensatzes: Es ist neben dem Ländernamen nur der IMR enthalten,
# sonst keine zu vergleichenden Daten. Wir machen uns auf die Suche nach zu verwendenden Dat

# Daten im Internet finden, infant mortality rate (imr) auf der Centrale Intelligence Agency
# Webseite: https://www.cia.gov/the-world-factbook/field/infant-mortality-rate/country-compa
# human development index (hdi) auf World Population Review Webseite:
# https://worldpopulationreview.com/country-rankings/hdi-by-country
# Daten im CSV-Format mit den Namen IMR.csv und HDI.csv speichern
# Ueberprüfen der Daten
imr <- read.csv("IMR.csv")
str(imr)
print(c(imr$value, imr$name))
hdi <- read.csv("HDI.csv")
str(hdi)
print(c(hdi$hdi, hdi$country))
```

Korrelationsanalyse in R (Datei *Korrelationsanalyse.R* auf Ilias) am Beispiel der infant mortality rate (imr) und des human development index (hdi)

```
# Datensätze (Dataframes) kombinieren mit R-Erweiterung „sqldf" um SQL Syntax verwenden zu k
imr_hdi <- sqldf('
SELECT imr.name AS "LAND", imr.value As "IMR", hdi.hdi AS "HDI" FROM imr
INNER JOIN hdi ON imr.name = hdi.country
ORDER BY imr.value DESC
')
str(imr_hdi)
print(imr_hdi)

# Kombinierte und bereinigte Daten auswerten
plot(imr_hdi$imr ~ imr_hdi$hdi, main = "IMR_HDI_Plot", ylab = "Säuglingssterblichkeit (per 1
Lebendgeburten)", xlab = "Human Development Index", ylim = range(1:110))
abline(lm(imr_hdi$imr ~ imr_hdi$hdi), col = "red") # Regressionsgerade
# Quantifizierung der Zusammenhänge mittels Spearman Korrelation Funktion
help(cor.test)
cor.test(imr_hdi$imr, imr_hdi$hdi, method="spearman", exact=FALSE)
# Ergebnis rho: -0.923 -> Negative bzw. Antikorrelation, da Wert nah bei 1
# Deutung rho: Wenn der HDI höher liegt, dann sinkt die IMR
# Ergebnis p-Wert: 2.2e-16 (Gleitkomma) -> 0.00000000000000022
# Deutung p-Wert: Er liegt unter 0.05, es gibt also eine statistische Signifikanz
```

1.7 Linear Regress Analyse

- Theoretische Grundlagen der Linear Regression Analyse

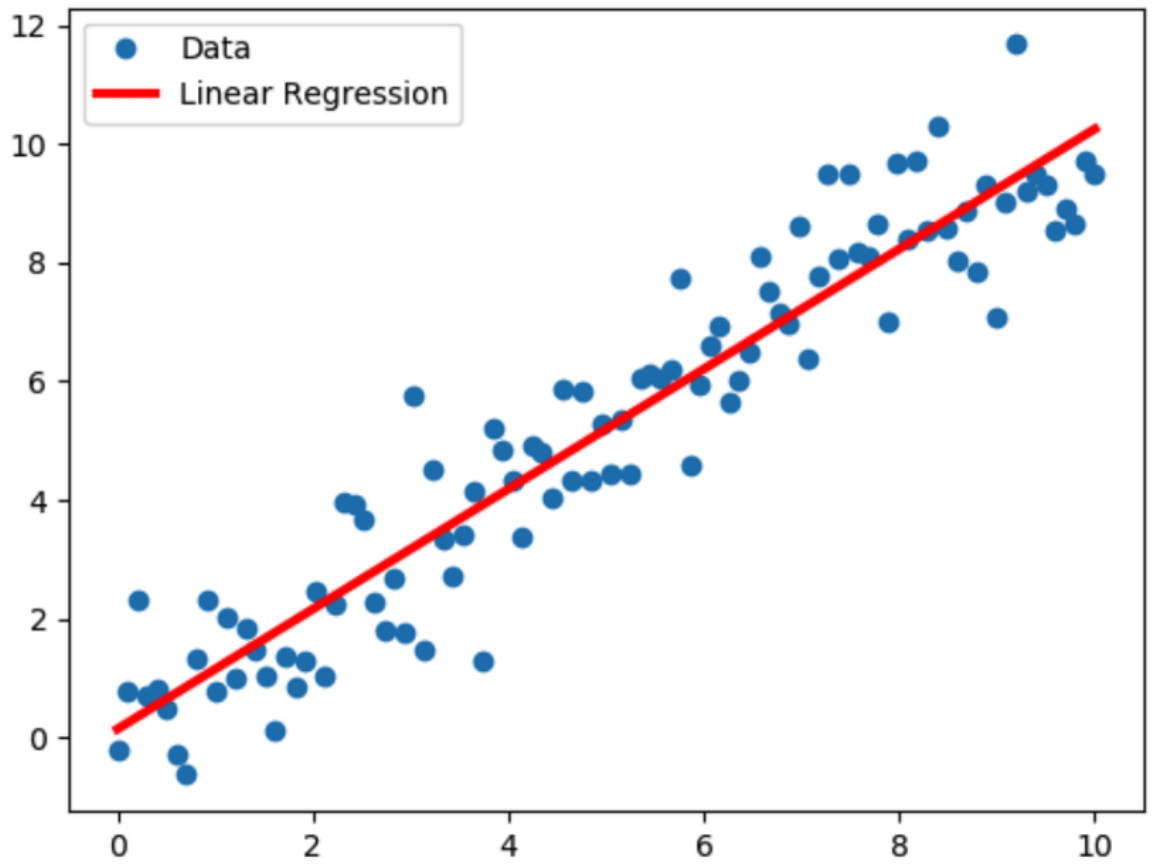
- Anwendung der Linear Regression-Analyse mit Beispieldaten

1.7.1 Linear Regression-Analyse

Lineare Regression ist eines der nützlichsten Werkzeuge in der Statistik. Regressionsanalyse erlaubt es Zusammenhänge zwischen Parametern zu schätzen und somit ein erklärendes Model für das Auftreten gewisser Phänomene zu geben. Wirkliche **Kausalität** wird durch statistische Analysen dieser Art zwar nicht aufgedeckt, die Ergebnisse aus solchen Analysen können aber **Hinweise** in diese Richtung geben (Wozabal, 2007).

1.7.2 6.a) Theoretische Grundlagen der Linear Regression-Analyse

Abbildung 36: Abb. 6.a.1: Lineare Regressionsgrade



Linear Regression-Analyse in R (*Lineare_Regressionsanalyse.R* auf Ilias)

```
# R-Paket 'ourdata' laden und mittels Funktion 'combine' einen vergleichbaren Datensatz erst
IMR und HDI
library(ourdata)
df <- combine(imr$name, hdi$country, imr$value, hdi$hdi, col1 = "Land", col2 = "IMR", col3 =

# Lineares Modell erstellen
```

```r
mdl <- lm(IMR ~ HDI, data=df)

# 'summary' Funktion
summary(mdl)

# Den p-Wert für 'HDI' errechnen
matCoef <- summary(mdl)$coefficients
pval <- matCoef["HDI", 4]
print(paste0("Der Effekt vom HDI auf IMR ist statistisch signifikant p = ", round(pval, 2),

# Diagramm ausgeben
plot(df$HDI, df$IMR, xlab = "Prädiktor", ylab = "Ergebnis", col = "darkblue", pch = 16, mai
Regression")
abline(mdl, col = "darkred")
```

1.7.3 6.b) Anwendung der Linear Regression-Analyse mit Beispieldaten

1.8 Methodik: Basket Analyse (Association Rules)

- Theoretische Grundlagen der Basket Analyse
- Anwendung der Basket-Analyse mit Beispieldaten

1.8.1 Basket Analyse

Die **Basket Analyse** kann zur Entdeckung von **Assoziationen** und Korrelationen zwischen Elementen in riesigen transaktionalen oder relationalen Datensätzen führen.

Das Aufsuchen von **Verbindungen** zwischen verschiedenen Artikeln, die Kunden in ihre „Warenkörbe“ legen ist ein häufiger Einsatz der Analyse. Das Wissen um diese Assoziationen kann für Einzelhändler oder Vermarkter hilfreich sein, um Marketingstrategien zu entwickeln. Das geschieht, indem sie Erkenntnisse darüber gewinnen, welche Artikel von Kunden häufig zusammen gekauft werden.

Wenn Kunden beispielsweise Milch kaufen, *wie* ***wahrscheinlich*** *ist es, dass sie auf derselben Fahrt zum Supermarkt auch Brot (und welche Brotsorte) kaufen?* Diese Informationen können zu einer Umsatzsteigerung führen, indem sie Einzelhändlern helfen, **selektives Marketing** zu betreiben und ihren Verkaufsraum zu planen.

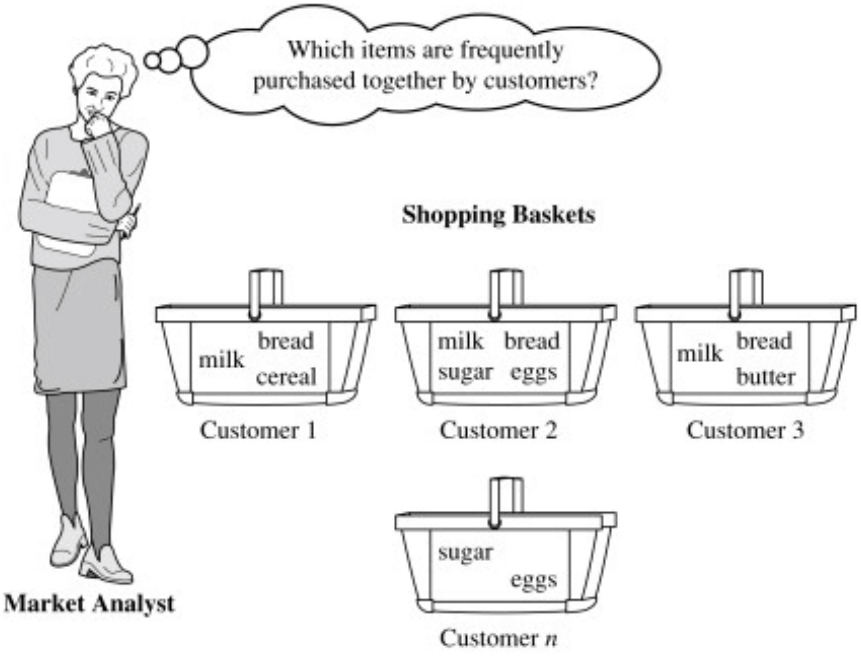
1.9 Methodik: Basket Analyse (Association Rules)

1.9.1 7.a) Theoretische Grundlagen der Basket Analyse

1.9.1.1 Basket Analyse

Stelle Dir das Universum einfach als eine Menge von Artikeln vor, die im Geschäft erhältlich sind. Jeder Artikel eine boolesche Variable, die das Vorhandensein oder Fehlen dieses Artikels darstellt. Jeder Korb kann nun durch einen booleschen Vektor von Werten repräsentiert werden, die diesen Variablen zugewiesen werden. Die booleschen Vektoren können von ***Kaufmustern*** *analysiert werden, die Artikel widerspiegeln, die häufig verbunden oder zusammen gekauft werden. Solche Muster werden in Form von* ***Assoziationsregeln*** *dargestellt.*

Abbildung 37: Abb. 7.a.1: Sinnbild



Basket Analyse in R (Datei *Basket_Analyse.R* auf Ilias)

```
# Bibliotheken laden
library(arules)

# Paket mit Mining Datensätzen und Assoziationsregeln

library(datasets) # OpenIntro Datensätze

# Data frame laden
data(Groceries)

# Frequenz Diagramm für die Top 20 Artikel erstellen
itemFrequencyPlot(Groceries, topN = 20, type = "absolute", horiz = TRUE)
# nach Milch suchen
itemFrequency(Groceries)[grep("milk", itemLabels(Groceries))]

# Die Regeln ableiten
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))
```

1.9.2 7.b) Anwendung der Basket-Analyse mit Beispieldaten

Basket Analyse in R (Datei *Basket_Analyse.R* auf Ilias)

```
# Zeige die Top 5 Regeln (nur 2 Nachkommastellen)
options(digits=2)
inspect(rules[1:5])
rules <- sort(rules, by="confidence", decreasing = TRUE)
```

```r
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8, maxlen = 10))

# Paket mit Visualisations-Assoziationsregeln laden
library(arulesViz)

# Berechnungen der Matrix
subset.matrix \<- is.subset(rules, rules)\
subset.matrix[lower.tri(subset.matrix, diag = T)] <- NA
redundant <- colSums(subset.matrix, na.rm = T) >= 1

# Diagramm ausgeben
plot(rules, method = "graph", engine = "htmlwidget")
```

1.10 Methodik: Cluster Analyse K-Means

- Theoretische Grundlagen der Cluster Analyse K-Means
- Anwendung der Cluster-Analyse K-Means mit Beispieldaten

1.10.1 K-Means Analyse

Der **K-Means-Clusteralgorithmus** ist definiert als eine *unüberwachte* **Lernmethode** mit einem iterativen Prozess, bei dem die Datensätze in eine Reihe von k Gruppen unterteilt werden, wobei k die vom Analysten vorgegebene Anzahl von Gruppen ist.

Bei diesem Algorithmus handelt es sich um einen **iterativen Algorithmus**, der den Datensatz entsprechend den Merkmalen in k vordefinierte, sich nicht überschneidende Cluster oder Untergruppen unterteilt. Er macht die Datenpunkte zwischen den Clustern so ähnlich wie möglich und versucht auch, die Cluster so weit wie möglich beizubehalten.

1.10.2 8.a) Theoretische Grundlagen der Cluster Analyse K-Means

K-Means Analyse in R (Datei *K-Means.R* aud Ilias)

```r
# Bibliotheken laden
library(cluster) library(factoextra) library(pheatmap)
library(factoextra)
library(pheatmap)

# Datensatz von Verbrechern in den USA erstellen
mydata <- scale(USArrests)

# K-Means Clustering
fviz_nbclust(mydata, kmeans, method = "gap_stat")
```

1.10.3 8.b) Anwendung der Cluster-Analyse K-Means mit Beispieldaten

K-Means Analyse in R (Datei *K-Means.R* aud Ilias)

```
# K-Means Clustering
set.seed(123)

# für die Reproduzierbarkeit

km.res <- kmeans(mydata, 3, nstart = 25)

# Visualisieren
fviz_cluster(km.res, data = mydata, palette = "jco", ggtheme = theme_minimal(), barfill = "

# Hierarchisches Clustering
# Cluster Denrogramm
res.hc <- hclust(dist(mydata), method = "ward.D2")
fviz_dend
(res.hc, cex = 0.5, k = 4, palette = "jco")

# Heatmap erstellen
pheatmap(t(mydata), cutree_cols = 4)
```

1.11 Methodik: Neuronale Netze-Analyse

- Theoretische Grundlagen der Neuronale Netze Analyse
- Anwendung der Neuronale Netze-Analyse mit Beispieldaten

1.11.1 Neuronale Netze-Analyse

Ein **neuronales Netz** ist eine informationsverarbeitende Maschine und kann als Analogon zum **menschlichen Nervensystem** betrachtet werden. Genau wie das menschliche Nervensystem, das aus miteinander verbundenen **Neuronen** besteht, setzt sich ein neuronales Netz aus miteinander verbundenen informationsverarbeitenden Einheiten zusammen.

Die informationsverarbeitenden Einheiten arbeiten nicht auf eine *lineare* Weise. Vielmehr bezieht das neuronale Netz seine Stärke aus der *parallelen* Verarbeitung von Informationen, die es ihm ermöglicht, mit **Nichtlinearität** umzugehen. Neuronale Netze sind nützlich, um aus komplexen Datensätzen eine Bedeutung abzuleiten und Muster zu erkennen.

1.11.2 9.a) Theoretische Grundlagen der Neuronale Netze-Analyse

Neuronale Netze-Analyse in R (Datei *Neuronale_Netze.R* auf Ilias)

```
# Bibliothek 'neuralnet' laden library(neuralnet)

# Daten einlesen data <- read.csv("cereals.csv", header=T) print(data)

# Zufälliges Zusammenstellen samplesize = 0.60 * nrow(data) set.seed(80) index <- sample( se
```

```r
# Erstelle Trainings- und Testdaten-Set
datatrain <- data[ index, ] datatest <- data[ -index, ]

# Daten für neurales Netzwerk skalieren
max <- apply(data , 2 , max) min <- apply(data, 2 , min) scaled <- as.data.frame(scale(data
```

1.11.3 9.b) Anwendung der Neuronale Netze-Analyse mit Beispieldaten

Neuronale Netze-Analyse in R (Datei *Neuronale_Netze.R* auf Ilias)

```r
## Neuronales Netz anpassen - Erstelle Trainings- und Testdaten-Set
trainNN <- scaled[index , ]
testNN <- scaled[-index , ]

# Neuronales Netz anpassen
set.seed(2) NN <- neuralnet(rating ~ calories + protein + fat + sodium + fiber, trainNN, hi

# Diagramm erstellen plot(NN)
## Vorhersage mit Hilfe des neuronalen Netzes predict_testNN <- compute(NN, testNN[,c(1:5)],

# Diagramm und Regressionsgrade erstellen
plot(datatest$rating, predict_testNN, col='blue', pch=16, ylab = "predicted rating NN", xlab
```

1.12 Methodiken im Netz

Alle **Methodiken** sind mit Code und Diagrammen **hier** verfügbar.

1.13 Praktische Anwendung mit eigenen Daten / Ethik der Datenanalyse

- Auf Basis eigener Daten eine Analyse entwickeln
  und die Anwendbarkeit prüfen
- Die Ergebnisse als kurze Präsentationen darstellen und eine Hausarbeit erstellen
- Moralische und ethische Aspekte betrachten sowie den Datenschutz der Datenanalyse beurteilen und diskutieren

1.13.1 Hausarbeit und Präsentation

Auf den folgenden Seiten ist die zu erbringende schriftliche Leistung / **Hausarbeit** sowie die mündliche Leistung / **Präsentation** beschrieben. Diese sollen die theoretischen und praktischen Fähigkeiten abfragen, die im Fach *Technical Applications & Data Analytics* aufgebaut wurden.

Die Erstellung der Arbeit muss **allen geltenden Anfordernissen** einer wissenschaftlichen Arbeit genügen und muss als **Struktur** z.B. enthalten:

- Inhaltsverzeichnis
- Literaturverzeichnis / Quellenangaben
- Anhang

1.13.2 10.a) Auf Basis eigener Daten eine Analyse entwickeln und die Anwendbarkeit prüfen

Hausarbeit: „Die Datensammlung“ oder auch:

Was kann ich selber für Daten generieren? Oder welche Daten hat schon jemand gesammelt? Oder welche Daten werden von uns generiert?!

- Gewicht
- Klang (beim Schütteln)
- Inhalt (Figuren)
- Gelenkseinschränkungen
- Beweglichkeit der Wirbelsäule
- Koordination
- Sportart in der Jugend
- Was werden über uns für Daten in den Sozialen Medien / Internet gesammelt?

1.13.2.1 Beispiele

Auswahl:

- *Entweder* **eigene Daten erheben** (evtl. die Excel aus dem Unterricht verwenden) bzw. **auf Datensammlungen zugreifen** (Statista, WWW etc.) und diese kombinieren (mehrere Datensätze vereinen)
- Sport, Wettkampf und Wirtschaft
- Spiele (Brett- oder Computerspiele)
- Sozialdaten (Gesundheit, Bevölkerungsentwicklung, Dating)
- Kurioses und Lustiges (sei kreativ!)
- *oder* **Analyse der Datensammlung** von Tech-Großkonzernen (Google, Apple, MS etc.) im Hinblick auf den Umfang bzw. Kategorien (siehe Grafik *Datensammler* auf Ilias)

1.13.2.2 Ablauf (zu den ersten beiden Verfahren):

- Daten generieren / Auffinden von Daten
- Statistische Auswertung in *R / R Studio* (Beispiel R-Skripte zur *Korrelationsanalyse* auf Ilias zur Hilfe nehmen!)
- wissenschaftliche Arbei mit den Ergebnissen und Interpretationen verfassen

10. Praktische Anwendung mit eigenen Daten / Ethik der Datenanalyse 10.a) Auf Basis eigener Daten eine Analyse entwickeln und die Anwendbarkeit prüfen

1.13.2.3 Prüfungsleistung

- Hausarbeit in Form einer **wissenschaftlich verfassten Arbeit** mit **statistischer Auswertung** (muss in *R / R-Studio* durchgeführt worden sein!) über **12 - 15 Seiten** -> per Mail an mich und auf Ilias hochladen bis zum **07.02.2022 23:55 Uhr**
- Sofern eigene Daten erhoben wurden und die Excel-Datei „*Fragebogen.xlsm*“ verwendet und an die eigenen Erfordernisse angepasst wurde -> per Mail an mich und auf Ilias bis zum **07.02.2022 23:55 Uhr**
- Ergebnisse als **Präsentation** bzw. als **„Pitch“** (ca.15 – 20 Min.) vorführen am **17.12.2021** -> per Mail am Ende der Vorlesung an mich

1.13.3 10.b) Die Ergebnisse als kurze Präsentationen darstellen und eine Hausarbeit erstellen

1.13.3.1 Gliederung der wissenschaftlichen Arbeit (Vorschlag):

- **Inhaltverzeichnis**
- **Hinführung zum Thema / Einleitung**
  - Idee hinter der Analyse
  - Erwartungshaltung in Bezug auf die Ergebnisse
  - Gegenwartsbezug (z.B. COVID-Relevanz)
- **Technische Voraussetzungen**
  - verwendete Software (z.B. Excel, R), benutzte Hardware (z.B. iPad)
  - Dateiformate (z.B. CSV), Datensatzformate (z.B. Tabellen)
- **Methodisches Vorgehen**
  - Art der Arbeit (z.B. empirische Forschung)
  - statistische Verfahren (z.B. Korrelationsanalyse), Diagrammarten (z.B. Boxplot)
- **Fazit und Zusammenfassung**
  - Ergebnisse und Diskussion
  - moralische / ethische Betrachtung, Berücksichtigung des Datenschutzes
- **Literaturverzeichnis**
- **Anhang** (wird max. mit 3 Seiten zur Gesamtseitenanzahl angerechnet)
- eigenen R-Code / Excel- und VBA-Code
- Datenkonvertierungen und -bereinigungen (vorher / nachher)

1.13.3.2 Gliederung der Präsentation (Vorschlag):

- **Einleitung**
  - Grundidee vorstellen
  - Persönliche Erwartungshaltung an das Thema
  - Quellen der Daten offen legen
- **Präsentation und Interpretation der Ergebnisse**
  - Diagramme zeigen (z.B. in R oder Powerpoint-Foliensatz)
  - Datensätze vorstellen (z.B. im Editor)
- **Zusammenfassung**
  - moralische und ethische Aspekte betrachten
  - Datenschutz der Datenanalyse beurteilen

1.13.4 10.c) Moralische und ethische Aspekte betrachten sowie den Datenschutz der Datenanalyse beurteilen und diskutieren

1.13.4.1 Gruppenarbeit

Die Aspekte von Datenerhebungen und der Verwendung von Daten im Hinblick auf **Datenschutz** und **Sicherheit** werden gemeinsam in der Vorlesung erörtert, um eine **Sensibilisierung** für das Thema zu erreichen.

Auch die Rolle von sogenannten **Whistleblowern** soll erwähnt werden.

1.13.4.2 Fragen

- Welche Arten von Daten gibt es?
- Welche davon sind besonders schützenswert?
- Welche Sicherheitsmechanismen benötigen sensible Daten?

1.14 Technical Applications & Data Analytics

1.14.1 R-Package ourdata

Das **R-Paket** ourdata ist im Rahmen der Vorlesungen in den Studiengängen **Audiovisuelle Medien & Online Publishing** und **Game Design & Management** entstanden. Es enthält einen Großteil der zu Übungszwecken verwendeten **Datensätze**, sowie einige selbst-programmierte **Funktionen** mit den sich diese Daten manipulieren bzw. **Diagramme** ausgeben lassen.

1.14.2 Daten data

Das R-Paket ourdata enthält folgende **Daten**:

- fragebogen – Kopfumfang und andere Merkmale aus dem **GD**- und **AVM-Kurs**. – Alter, Geschlecht, Mathenote, vor. Note, Angst, Interesse, Praxis
- hdi – Weltweiter Index der **menschlichen Entwicklung**.
- imr – **Säuglingssterblichkeitsraten** weltweit.
- kirche – **Kirchenaustritte** in Deutschland von *2017* bis *2020*.
- koelsch – Konsum von **Kölsch** von *2017* bis *2020*.

Mit der **Funktion** help() kannst Du Dir **Hilfeseiten** zu allen *Datensätzen* anzeigen lassen. Z.B. help(fragebogen) erklärt den Inhalt des Datenpaketes fragebogen.

1.14.3 Funktionen functions

Das R-Paket ourdata enthält folgende **Funktionen**:

- combine(x, y, …) – Kombiniert zwei Datensätze mit **ID** und **Fremdschlüsselabgleich**.
- ourdata()
  - Druckt eine **Willkommensnachricht** aus.
- plotter(…)
  - Zeichnet interaktiv mit variablen Daten verschiedene **Diagramme**.

- transformer(x, …)
  - Transformiert **Werte** vom **Typ** *char* in *numerische Werte*, – z.B. female zu 1, male zu 2 und `divers` zu 3.
- translate(x, target_lang) – Übersetzt Text in die *Zielsprache* mit der **DeepL API**.

Mit der **Funktion** help() kannst Du Dir **Hilfeseiten** zu allen *Funktionen* anzeigen lassen. Z.B. help(plotter) erklärt die Funktonsweise von der Funktion plotter().

1.15 GM_bac GM_bac & AVM_bac 5.Sem. AVM_bac 5.Sem.

In den Kursen **GM_bac5** und **AVM_bac5** im Fach **Technical Applications & Data Analytics** haben wir viele Themen der
Datenanalyse betrachtet und uns einige **technische Applikationen** genauer angeschaut. Der Schwerpunkt lag auf **R**, einer *Statistik-Programmiersprache*, mit der übrigens auch dieses Dokument erstellt wurde.

Auf **Ilias** sind die Vorlesungsfolien sowie alle *R-Skripte* und sonstiges Material. Hier findest Du in ansprechender Weise aufbereitet **Diagramme** oder **statistische Methoden** und deren *R-Code* (dieser liegt zusätzlich auf Ilias im Ordner "*Beispiel-Datensätze, Python und R-Skripte*"). Der Code wartet darauf von Dir ausprobiert zu werden!

1.16 Literaturverzeichnis

Cleve, J., & Lämmel, U. (2020). *Data Mining* (3. Aufl.). De Gruyter.

Romeijn, J.-W. (2016). Philosophy of Statistics. In E. N. Zalta (Hrsg.), *The Stanford Encyclopedia of Philosophy*. Stanford University.

Sauer, S. (2019). *Moderne Datenanalyse mit R*. Springer Gabler.

Tukey, J. W. (1962). The future of data analysis. *The Annals of Mathematical Statistics*, *33*(1).

Wickham, H., & Grolemund, G. (2017). *R for Data Science*. O'Reilly Media.

Wozabal, D. (2007). *Statistisches Programmieren – Regressionen in R (Session 6)*.

1.17 Anhang

HOCHSCHULE FRESENIUS
UNIVERSITY OF APPLIED SCIENCES

Technical Applications & Data Analytics

R-Package `ourdata`

Das **R-Paket** `ourdata` ist im Rahmen der Vorlesungen in den Studiengängen **Audiovisuelle Medien & Online Publishing** und **Game Design & Management** entstanden. Es enthält einen Großteil der zu Übungszwecken verwendeten **Datensätze**, sowie einige selbstprogrammierte **Funktionen** mit den sich diese Daten manipulieren bzw. **Diagramme** ausgeben lassen.

Daten `data`

Das R-Paket `ourdata` enthält folgende **Daten**:

- `fragebogen`
  - Kopfumfang und andere Merkmale aus dem **GD**- und **AVM-Kurs**.
  - Alter, Geschlecht, Mathenote, vor. Note, Angst, Interesse, Praxis
- `hdi`
  - Weltweiter Index der **menschlichen Entwicklung**.
- `imr`
  - **Säuglingssterblichkeitsraten** weltweit.
- `kirche`
  - **Kirchenaustritte** in Deutschland von *2017* bis *2020*.
- `koelsch`
  - Konsum von **Kölsch** von *2017* bis *2020*.

Mit der **Funktion** `help()` kannst Du Dir **Hilfeseiten** zu allen *Datensätzen* anzeigen lassen. Z.B. `help(fragebogen)` erklärt den Inhalt des Datenpaketes `fragebogen`.

HOCHSCHULE FRESENIUS
UNIVERSITY OF APPLIED SCIENCES

Funktionen `functions`

Das R-Paket `ourdata` enthält folgende **Funktionen**:

- `combine(x, y, ...)`
  - Kombiniert zwei Datensätze mit **ID** und **Fremdschlüsselabgleich**.
- `ourdata()`
  - Druckt eine **Willkommensnachricht** aus.
- `plotter(...)`
  - Zeichnet interaktiv mit variablen Daten verschiedene **Diagramme**.
- `transformer(x, ...)`
  - Transformiert **Werte** vom **Typ** *char* in *numerische Werte*,
  - z.B. `female` zu `1`, `male` zu `2` und ´diverszu3`.
- `translate(x, target_lang)`
  - Übersetzt Text in die *Zielsprache* mit der **DeepL API**.

Mit der **Funktion** `help()` kannst Du Dir **Hilfeseiten** zu allen *Funktionen* anzeigen lassen. Z.B. `help(plotter)` erklärt die Funktonsweise von der Funktion `plotter()`.

GM_bac & AVM_bac `5.Sem.`

In den Kursen **GM_bac5** und **AVM_bac5** im Fach **Technical Applications & Data Analytics** haben wir viele Themen der **Datenanalyse** betrachtet und uns einige **technische Applikationen** genauer angeschaut. Der Schwerpunkt lag auf **R**, einer *Statistik-Programmiersprache*, mit der übrigens auch dieses Dokument erstellt wurde.

Auf **Ilias** sind die Vorlesungsfolien sowie alle *R-Skripte* und sonstiges Material. Hier findest Du in ansprechender Weise aufbereitet **Diagramme** oder **statistische Methoden** und deren *R-Code* (dieser liegt zusätzlich auf Ilias im Ordner "*Beispiel-Datensätze, Python und R-Skripte*"). Der Code wartet darauf von Dir ausprobiert zu werden!

Diagramme plots

In **R** oder **RStudio** kann man verschiedene **Diagramm-Typen** zur Visualisation verwenden. Zur Hilfe dient die **Funktion** plottter() mit der man alle Diagramme (bis auf das Tortendiagramm) mit eigenen Daten ausführen kann.

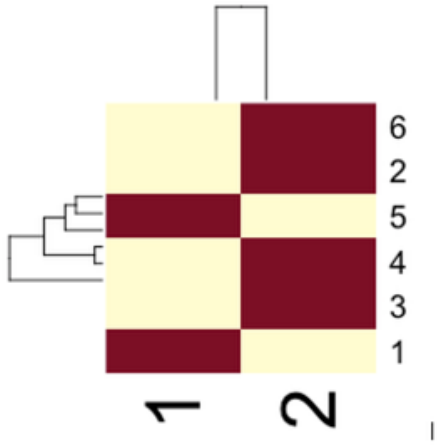


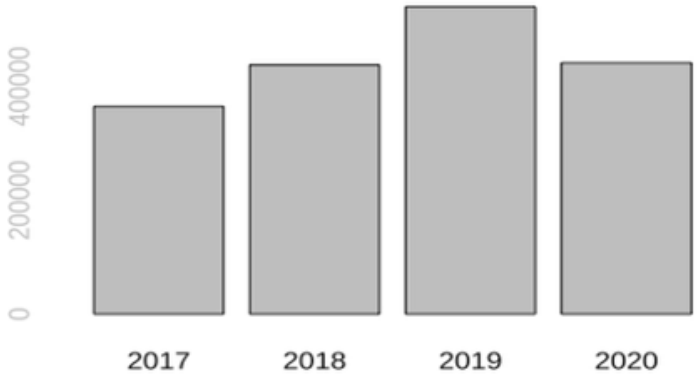
Abb. 2.1a: Heatmap

Hier eine Auswahl der **Diagramm-Typen**, mit dem dazugehörigen **Code**:

Balkendiagramm

Das **Balkendiagramm** veranschaulicht die Verbindung zwischen einer *numerischen* und einer *kategorialen* **Variablen**. Das Balkendiagramm stellt jede **Kategorie** als **Balken** dar und spiegelt den entsprechenden *numerischen* Wert mit der **Größe** des Balkens wider.

```
# Balkendiagramm erstellen
barplot(kirche$Austritte, kirche$Jahr, main = "Kichenaustritte", col.main = "white", col.lab = "white", yaxt = "n", ylab = "Austritte (per 1.000)", xlab = "Jahreszahlen", names = c("2017", "2018", "2019", "2020"))
# Beschriftung für x- und y-Achse verbessern und die Farbgebung für das Dark-Theme anpassen
axis(1, at = 1:4, lwd = 3, lwd.ticks = 3, col = "white", col.ticks = "white", col.lab = "white", col.axis = "white")
ypos <- seq(0, 600000, by = 100000)
axis(2, at = ypos, labels = sprintf("%1.0f", ypos), lwd = 0.5, lwd.ticks = 1, col = "white", col.ticks = "white",  col.axis = "grey")
```

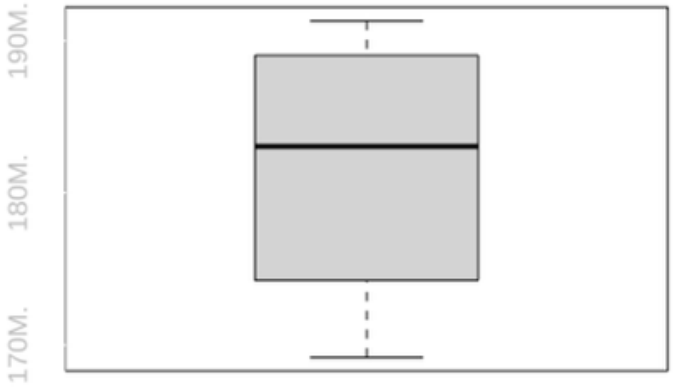


Box-Diagramm

Das **Box-Diagramm** zeigt die **Verteilung** einer *numerischen* **Variablen** basierend auf fünf zusammenfassenden Statistiken: - minimaler Nicht-Ausreißer - erstes Quartil - Median - drittes Quartil - Nicht-Ausreißer

Außerdem zeigen Boxplots die Positionierung von Ausreißern und ob die Daten verzerrt sind.

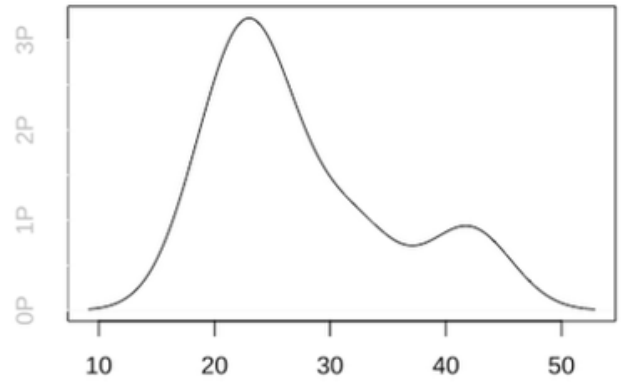
```
# Box-Diagramm erstellen
boxplot(koelsch$Koelsch, main = "Kölschkonsum", col.main = "white", col.lab = "white", yaxt = "n", ylab = "Kölschkonsu
m in Mil. Litern", xlab = "über den Zeitraum 2017 bis 2020", names = "2020")
# Beschriftung für die y-Achse verbessern und die Farbgebung für das Dark-Theme anpassen
ypos <- seq(160000000, 200000000, by = 10000000)
axis(2, at = ypos, labels = sprintf("%1.0fM.", ypos/1000000), lwd = 0.5, lwd.ticks = 1, col = "white", col.ticks = "wh
ite",  col.axis = "grey")
```



Dichtediagramm

Das **Dichtediagramm** zeigt die Verteilung einer *numerischen* **Variablen** über ein **kontinuierliches Intervall**. *Peaks* eines Dichtediagramms visualisieren, wo sich die Werte *numerischer* Variablen **konzentrieren**.

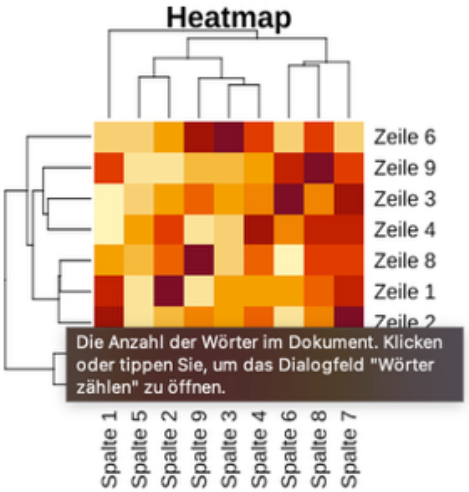
```
# Dichtediagramm erstellen
plot(density(fragebogen$alter), main = "Verteilung des Alters im Kurs", col.main = "white", col.lab = "white", yaxt =
"n", ylab = "Personen (Dichte)", xlab = "Alter (in Jahren)")
# Beschriftung für die y-Achse verbessern und die Farbgebung für das Dark-Theme anpassen
ypos <- c(0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.06)
axis(2, at = ypos, labels = sprintf("%1.0fP", ypos*50), lwd = 0.5, lwd.ticks = 1, col = "white", col.ticks = "white",
col.axis = "grey")
```



Heatmap

Eine **Heatmap*-Diagramm **visualisiert einzelne Werte einer** Matrix **mit** Farben. Häufigere Werte **werden typischerweise durch *hellere rötliche* Farben angezeigt und** weniger häufige** Werte werden typischerweise durch *dunklere* Farben angezeigt.

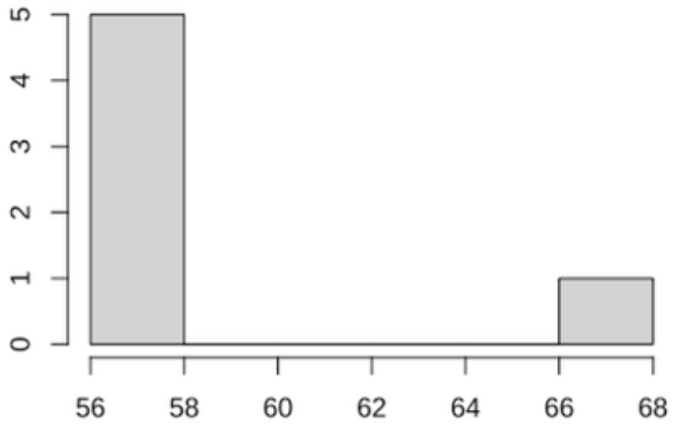
```r
data <- matrix(rnorm(81, 0, 9), nrow = 9, ncol = 9)     # Beispiel Daten erstellen
colnames(data) <- paste0("Spalte ", 1:9)                # Spaltennamen setzen
rownames(data) <- paste0("Zeile ", 1:9)                 # Zeilennamen setzen
# Heatmap erstellen
heatmap(data, main = "Heatmap", col.main = "white", col.lab = "white")
```



Histogramm

Das **Histogramm** gruppiert *kontinuierliche* **Daten** in Bereiche und stellt diese Daten als **Balken** dar. Die **Höhe** jedes Balkens zeigt die **Anzahl** der **Beobachtungen** in jedem Bereich an.

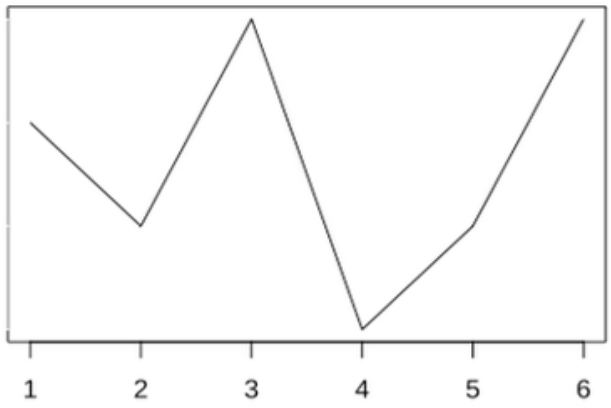
```r
# Histogramm erstellen
hist(fragebogen$kopf, main = "Kopfumfänge", col.main = "white", col.lab = "white", ylab = "Personen (Anzahl)", xlab =
"Kopfumfang (in cm)")
```



Liniendiagramm

Das **Liniendiagramm** visualisiert Werte entlang einer **Sequenz** (z.B. *über die Zeit*). Liniendiagramme bestehen aus einer *x-Achse* und einer *y-Achse*. Die x-Achse zeigt normalerweise die **Sequenz** und die y-Achse die Werte an, die jedem Punkt der Sequenz **entsprechen**.

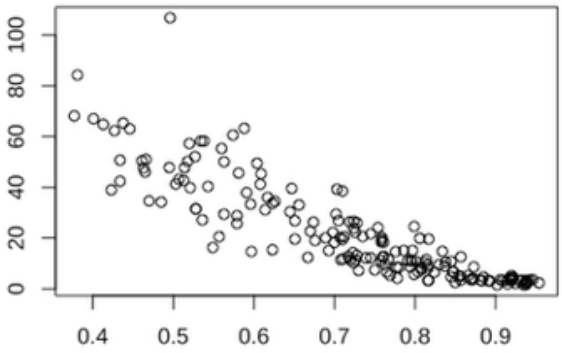
```
# Liniendiagramm erstellen
plot(fragebogen$note_mathe, type = "l", main = "Mathenoten", ylab = "Noten", xlab = "Person x",  yaxt = "n", col.main
= "white", col.lab = "white")
# Beschriftung für die y-Achse verbessern und die Farbgebung für das Dark-Theme anpassen
ypos <- c(2, 3, 4, 5)
axis(2, at = ypos, labels = sprintf("%1.0f", ypos), lwd = 0.5, lwd.ticks = 1, col = "white", col.ticks = "white", col.
lab = "grey", col.axis = "white")
```



Streudiagramm

Das **Streudiagramm** zeigt zwei *numerische* **Variablen** mit **Punkten** an. Jeder Punkt zeigt den Wert einer Variablen auf der *x-Achse* und den Wert der anderen Variablen auf der *y-Achse* an.

```
# Daten zusammenfügen
df <- combine(imr$name, hdi$country, imr$value, hdi$hdi, col1 = "Land", col2 = "IMR", col3 = "HDI")

'data.frame':   175 obs. of  3 variables:
 $ Land: chr  "Afghanistan" "Central African Republic" "Niger" "Chad" ...
 $ IMR : num  106.8 84.2 68.1 67 65.3 ...
 $ HDI : num  0.496 0.381 0.377 0.401 0.438 0.413 0.588 0.446 0.427 0.574 ...

# Streudiagramm erstellen
plot(df$HDI, df$IMR, main = "Einfluss des HDI auf IMR", ylab = "IMR", xlab = "HDI", col.main = "white", col.lab = "whi
te")
```



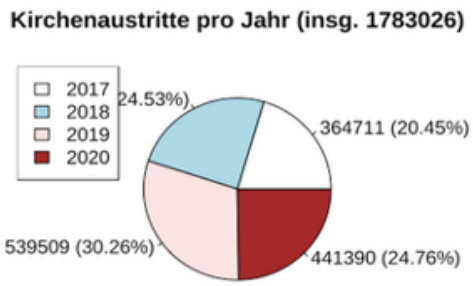
Tortendiagramm

Kreis- oder **Torten-Diagramm**, sind zwar weit verbreitet, weisen jedoch einige Nachteile auf:

- Unterschiede zwischen den Anteilswerte sind weniger gut erkennbar, da dazu die Fläche der Kreissegmente verglichen werden muss
- bei vielen Kategorien wird die Darstellung schnell unübersichtlich
- sehr kleine Anteilswerte können oftmals nicht im Kreisdiagramm dargestellt werden

Aufgrund dieser **Nachteile** bietet sich die Verwendung von Kreisdiagramme nur in *selten Fällen* an, meist liefern **Punkt**- oder **Balkendiagramme** bessere Darstellungen.

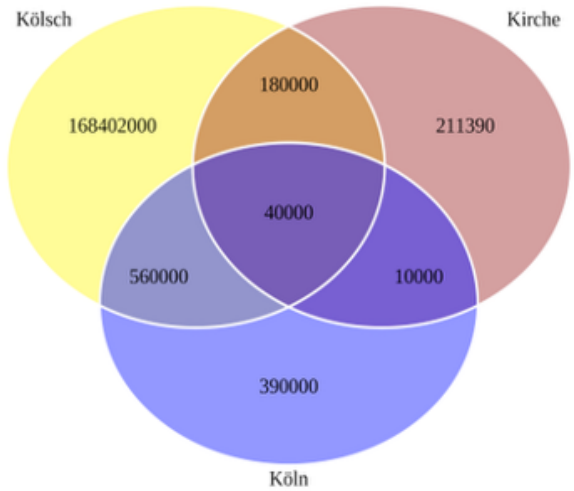
```
# Beschriftung Gesamtzahl Jahr mit Prozentangabe erstellen
pie_labels <- paste0(kirche$Austritte, " (", round(100 * kirche$Austritte/sum(kirche$Austritte), 2), "%)")
# Kuchendiagramm erstellen
pie(kirche$Austritte, main = paste0("Kirchenaustritte pro Jahr (insg. ", sum(kirche$Austritte), ")"), labels = pie_lab
els, col = c("white", "lightblue", "mistyrose", "brown"))
# Legende erstellen
legend("topleft", legend = c("2017", "2018", "2019", "2020"), fill =  c("white", "lightblue", "mistyrose", "brown"))
```



Venn-Diagramm

Ein **Venn-Diagramm** (oder **Primärdiagramm**; **Mengendiagramm**; **Logikdiagramm**), veranschaulicht alle möglichen *logischen* **Beziehungen** zwischen bestimmten **Datenmerkmalen**. Jedes Merkmal wird als **Kreis** dargestellt, wobei *überlappende* Teile der Kreise **Elemente** darstellen, die beide Merkmale **gleichzeitig aufweisen**.

```
# Triple Venn Diagramm erstellen
draw.triple.venn(area1 = koelsch$Koelsch[4], area2 = kirche$Austritte[4], area3 = 1000000, n12 = 220000, n23 = 50000,
n13 = 600000, n123 = 40000, main = "Kölsch -> Kirchenaustritt -> Köln?", fill = c("yellow", "brown", "blue"), category
= c("Kölsch", "Kirche", "Köln"), main.col = "white", sub.col = "white", col = "white")
```



Data Mining data science

Die **Methoden** des **Data Mining** lassen sich grundsätzlich in die Gruppen *Klassifikation, Prognose, Segmentierung* und *Abhängigkeitsentdeckung* einteilen. Dazu kommen **Algorithmen** zum Einsatz.

Ein Algorithmus ist eine formale Handlungsvorschrift zur Lösung von Instanzen einer bestimmten Problemklasse. [1]

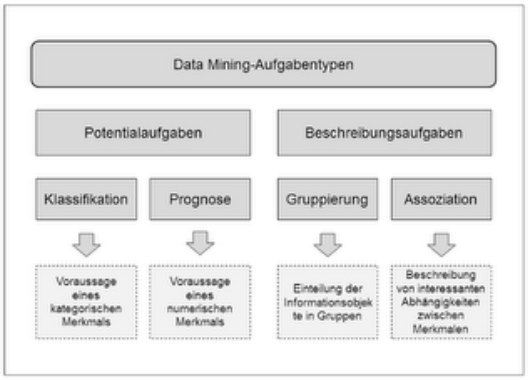


Abb. 2.2a: Gruppen des Data Mining

Hier findest Du die Data **Mining Methoden** aus den Vorlesungen, mit dem dazugehörigen **Code**:

---

[1] Sauer [2019]

HOCHSCHULE FRESENIUS
UNIVERSITY OF APPLIED SCIENCES

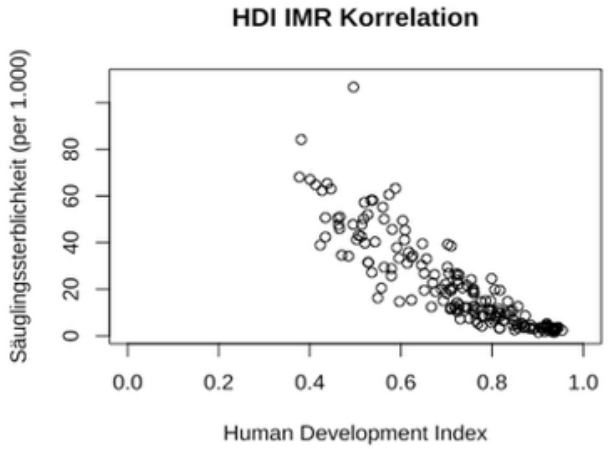
Korrelation Analyse

Will man einen **Zusammenhang** zwischen zwei **metrischen Variablen** untersuchen, zum Beispiel zwischen dem Alter und dem Gewicht von Kindern, so berechnet man eine **Korrelation**. Diese besteht aus einem **Korrelationskoeffizienten** (rho bei Spearman) und einem **p-Wert**.

Der **Korrelationskoeffizient** gibt die Stärke und die Richtung des Zusammenhangs an. Er liegt zwischen `-1` und `1`. Ein Wert nahe `-1` bezeichnet einen starken **negativen Zusammenhang** (z.B. *„Mehr zurückgelegte Strecke mit dem Auto, weniger Treibstoff im Tank"*). Ein Wert nahe `1` spricht für einen starken **positiven Zusammenhang** (z.B. *„mehr Futter, dickere Kühe"*). Kein Zusammenhang besteht, wenn der Wert nahe `0` liegt.

Der **p-Wert** sagt aus, ob es einen **signifikanten Zusammenhang** gibt. **p-Werte** kleiner als `0,05` werden als **statistisch signifikant** bezeichnet.

```
# Beide Listen mit 'SQL JOIN' kombinieren
imr_hdi <- sqldf('SELECT imr.name AS "country", imr.value As "imr", hdi.hdi AS "hdi" FROM imr INNER JOIN hdi ON imr.na
me = hdi.country ORDER BY imr.value DESC')

# Streudiagramm (Scatterplot)
plot(imr_hdi$imr ~ imr_hdi$hdi, main = "HDI IMR Korrelation", ylab = "Säuglingssterblichkeit (per 1.000)", xlab = "Hum
an Development Index", xlim = range(0:1), ylim = range(1:110))
```



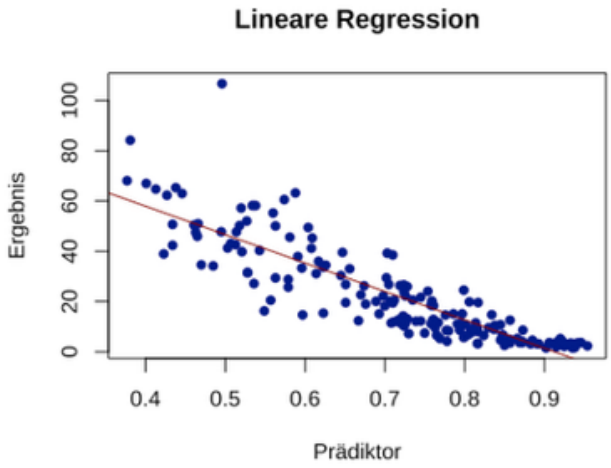

```
# Quantifizierung der Zusammenhänge mittels Spearman Korrelation Funktion
cor.test(imr_hdi$imr, imr_hdi$hdi, method="spearman", exact=FALSE)


    Spearman's rank correlation rho

data:  imr_hdi$imr and imr_hdi$hdi
S = 1719986, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
       rho
-0.9256444
```

Linear Regress Analyse

Lineare Regression ist eines der nützlichsten Werkzeuge in der Statistik. Regressionsanalyse erlaubt es **Zusammenhänge** zwischen Parametern zu schätzen und somit ein erklärendes Model für das Auftreten gewisser Phänomene zu geben. Wirkliche Kausalität wird durch statistische Analysen dieser Art zwar nicht aufgedeckt, die Ergebnisse aus solchen Analysen können aber Hinweise in diese Richtung geben.[2]

```
# Mittels Funktion 'combine' aus dem R-Paket 'ourdata' einen vergleichbaren Datensatz erstellen mit IMR und HDI
df <- combine(imr$name, hdi$country, imr$value, hdi$hdi, col1 = "Land", col2 = "IMR", col3 = "HDI")

# Lineares Modell mit 'lm' erstellen
mdl <- lm(IMR ~ HDI, data=df)

# 'summary' Funktion
summary(mdl)

# Den p-Wert für 'HDI' errechnen
matCoef <- summary(mdl)$coefficients
pval <- matCoef["HDI", 4]
print(paste0("Der Effekt vom HDI auf IMR ist statistisch signifikant p = ", round(pval, 2), " (", pval, ")."))

# Diagramm erstellen
plot(df$HDI, df$IMR, xlab = "Prädiktor", ylab = "Ergebnis", col = "darkblue", pch = 16, main = "Lineare Regression")
# Regressionsgrade erstellen
abline(mdl, col = "darkred")
```

[2] Wozabal [2007]




```
'data.frame':   175 obs. of  3 variables:
 $ Land: chr  "Afghanistan" "Central African Republic" "Niger" "Chad" ...
 $ IMR : num  106.8 84.2 68.1 67 65.3 ...
 $ HDI : num  0.496 0.381 0.377 0.401 0.438 0.413 0.588 0.446 0.427 0.574 ...

Call:
lm(formula = IMR ~ HDI, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
# nach Milch suchen
itemFrequency(Groceries)[grep("milk", itemLabels(Groceries))]

    whole milk    butter milk       UHT-milk condensed milk
    0.25551601     0.02796136     0.03345196     0.01026945

# Die Regeln ableiten
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8))

Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5   0.001      1
 maxlen target  ext
     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 5 6 done [0.02s].
writing ... [410 rule(s)] done [0.01s].
creating S4 object  ... done [0.00s].

# Zeige die Top 5 Regeln (nur 2 Nachkommastellen)
options(digits=2)
inspect(rules[1:5])
```

```
    lhs                          rhs            support confidence coverage lift
[1] {liquor, red/blush wine} => {bottled beer} 0.0019  0.90       0.0021   11.2
[2] {curd, cereals}          => {whole milk}   0.0010  0.91       0.0011    3.6
[3] {yogurt, cereals}        => {whole milk}   0.0017  0.81       0.0021    3.2
[4] {butter, jam}            => {whole milk}   0.0010  0.83       0.0012    3.3
[5] {soups, bottled beer}    => {whole milk}   0.0011  0.92       0.0012    3.6
    count
[1] 19
[2] 10
[3] 17
[4] 10
[5] 11
```

```
rules <- sort(rules, by="confidence", decreasing = TRUE)
rules <- apriori(Groceries, parameter = list(supp = 0.001, conf = 0.8, maxlen = 10))
```

```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen
        0.8    0.1    1 none FALSE            TRUE       5   0.001      1
 maxlen target  ext
     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 9

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [157 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 5 6 done [0.01s].
```

```
writing ... [410 rule(s)] done [0.02s].
creating S4 object  ... done [0.00s].
```

```
# Paket mit Visualisations-Assoziationsregeln laden
library(arulesViz)

# Berechnungen der Matrix
subset.matrix <- is.subset(rules, rules)
subset.matrix[lower.tri(subset.matrix, diag = T)] <- NA
redundant <- colSums(subset.matrix, na.rm = T) >= 1

# Diagramm ausgeben
plot(rules, method = "graph", engine = "htmlwidget")
```

Cluster Analyse K-Means

K-Means ist ein **Clustering-Verfahren**, bei dem die **Datensätze** in eine Reihe von k Gruppen unterteilt werden, wobei k die vom Analysten vorgegebene Anzahl von Gruppen ist.

Die folgenden *R-Codes* zeigen, wie man die optimale Anzahl von Clustern bestimmt und wie man K-Means und PAM-Clustering in R berechnet.

Bestimmung der **optimalen Anzahl von Clustern**:

```
# Bibliotheken laden
library(cluster)
library(factoextra)
library(pheatmap)

# Datensatz von Verbrechern in den USA erstellen
mydata <- scale(USArrests)

# K-Means Clustering
fviz_nbclust(mydata, kmeans, method = "gap_stat")
```

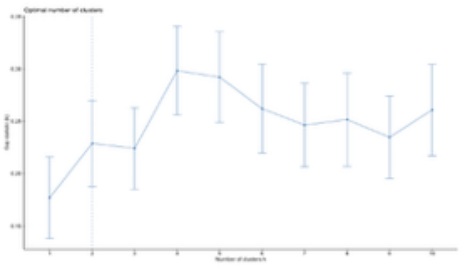


Abb. 2.2.4a: Optimal Number of Clusters

Berechnen und visualisieren des **K-Means Cluster**:

```
set.seed(123) # für die Reproduzierbarkeit
km.res <- kmeans(mydata, 3, nstart = 25)

# Visualisieren
fviz_cluster(km.res, data = mydata, palette = "jco", ggtheme = theme_minimal(), barfill = "red", barcolor   = "red", l
inecolor = "red")
```

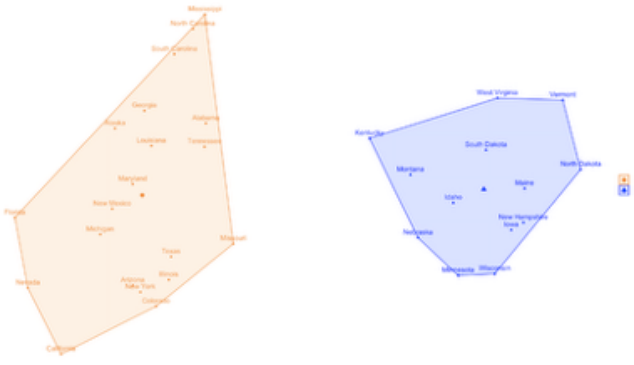


Abb. 2.2.4b: Cluster

Hierarchisches Clustering ist ein alternativer Ansatz zum **partitionierenden Clustering**, um Gruppen in einem Datensatz zu identifizieren. Bei diesem Verfahren muss die Anzahl der zu erzeugenden Cluster nicht im Voraus festgelegt werden.

Das Ergebnis des hierarchischen Clusterns ist eine *baumartige* Darstellung der Objekte, die auch als **Dendrogramm** bezeichnet wird. Die Beobachtungen können in Gruppen unterteilt werden, indem das Dendrogramm auf einem gewünschten **Ähnlichkeitsniveau** geschnitten wird:

```
# Hierarchisches Clustering
# Cluster Dendrogramm
res.hc <- hclust(dist(mydata),  method = "ward.D2")
fviz_dend(res.hc, cex = 0.5, k = 4, palette = "jco")
```

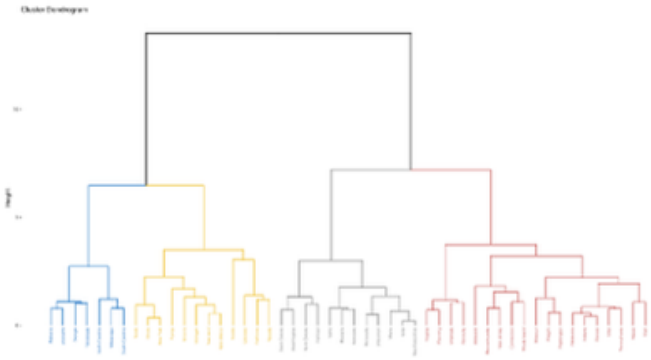


Abb. 2.2.4c: Cluster Dendogramm

Eine **Heatmap** ist eine weitere Möglichkeit, hierarchisches Clustering zu visualisieren. Sie wird auch als **Falschfarbenbild** bezeichnet, bei dem die Datenwerte in eine Farbskala umgewandelt werden. Mit Heatmaps können wir gleichzeitig Gruppen von Proben und Merkmalen visualisieren:

```
# Heatmap erstellen
pheatmap(t(mydata), cutree_cols = 4)
```

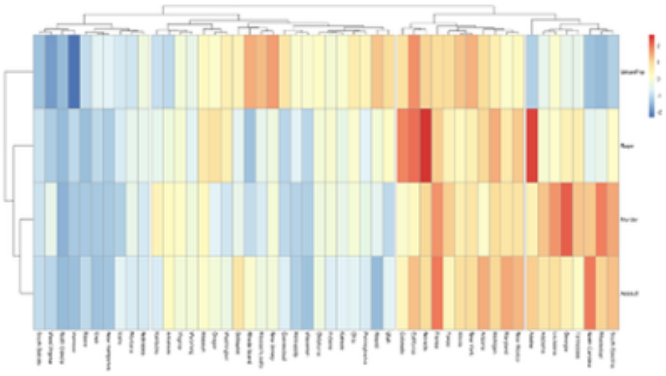


Abb. 2.2.4d: Cluster Heatmap

Neuronale Netze-Analyse

Ein **neuronales Netz** ist eine informationsverarbeitende Maschine und kann als Analogon zum **menschlichen Nervensystem** betrachtet werden. Genau wie das menschliche Nervensystem, das aus miteinander verbundenen Neuronen besteht, setzt sich ein neuronales Netz aus miteinander verbundenen **informationsverarbeitenden Einheiten** zusammen. Die informationsverarbeitenden Einheiten arbeiten nicht auf eine *lineare* Weise. Vielmehr bezieht das neuronale Netz seine Stärke aus der *parallelen* Verarbeitung von Informationen, die es ihm ermöglicht, mit **Nichtlinearität** umzugehen. Neuronale Netze sind nützlich, um aus komplexen Datensätzen eine Bedeutung abzuleiten und Muster zu erkennen.

Hier wird das **neuronale Netz** visualisiert. Unser Modell hat **3 Neuronen** in seiner versteckten Schicht. Die schwarzen Linien zeigen die Verbindungen mit den Gewichten. Die Gewichte werden mit dem zuvor erläuterten Backpropagation-Algorithmus berechnet. Die gelben Linien stellen den Bias-Term dar:

```
# Bibliothek 'neuralnet' laden
library(neuralnet)

# Daten einlesen
data <- read.csv("cereals.csv", header=T)

# Zufälliges Zusammenstellen
samplesize = 0.60 * nrow(data)
set.seed(80)
index <- sample( seq_len ( nrow ( data ) ), size = samplesize )

# Erstelle Trainings- und Testdaten-Set
datatrain <- data[ index, ]
datatest <- data[ -index, ]

# Daten für neurales Netzwerk skalieren
max <- apply(data , 2 , max)
min <- apply(data, 2 , min)
scaled <- as.data.frame(scale(data, center = min, scale = max - min))

## Neuronales Netz anpassen
```

Sonstiges misc

Hier findest Du **Code-Beispiele**, **Lustiges** und **Neuigkeiten** rund um das Thema *technische Applikationen & Datenanalyse*:

Beispiel R-Code

Dieses Beispiel zeigt die Verwendung von **Schleifen** in R. Schleifen kommen in der **Programmierung** häufig zum Einsatz, z.B. zum Überprüfen von **Übereinstimmungen** in zwei verschiedenen Datensätzen.

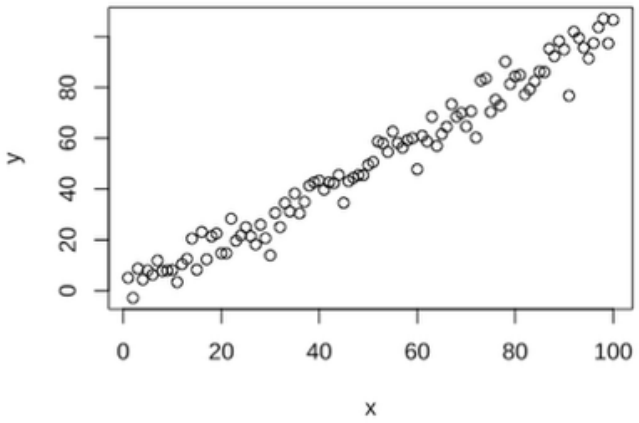
```
# Komplexere R Skript (Programmierung)

# Einfacher Vektor, die Funktion c kombiniert Werte in einen Vektor oder eine Liste
fruitVec <- c("Apfel", "Banane", "Orange", "Birne") # fruitVec wird mit 4 Werten (Strings) gefüllt

# Einfache While-Schleife
# Funktion length: gibt die Länge eines/r Vektors / Liste aus
# Der Vektor fruitVec kann wie ein Array angesprochen werden, um die einzelnen Werte auszulesen
pos <- 1
while (pos <= length(fruitVec)) {
  print(paste0("Die Frucht Nr.", pos, " ist : ", fruitVec[pos]))
  pos <- pos + 1
}

[1] "Die Frucht Nr.1 ist : Apfel"
[1] "Die Frucht Nr.2 ist : Banane"
[1] "Die Frucht Nr.3 ist : Orange"
[1] "Die Frucht Nr.4 ist : Birne"

# Einfache For-Schleife
# Funktion paste0 verbindet Strings
# Funktion which ermittelt die Position eines Wertes im Vektor (bzw. Array)
help(which)
for (fruit in fruitVec) {
  print(paste0("Die Frucht Nr.", which(fruit == fruitVec), " ist: ", fruit))
}
```

```
[1] "Die Frucht Nr.1 ist: Apfel"
[1] "Die Frucht Nr.2 ist: Banane"
[1] "Die Frucht Nr.3 ist: Orange"
[1] "Die Frucht Nr.4 ist: Birne"
```

Ext. R-Code ausführen

Dieser **Code** steht in der Datei `example.R` und wir von **R-Markdown** eingelesen und ausgeführt.

```
# x Werte von 1 und 100 zuweisen
x <- 1:100
# y 100 Werte zuweisen mit x + Zufallszahl (-5 bis +5)
y <- x + rnorm(100, sd = 5)
# Gibt den ersten oder letzten Teil eines Vektors, einer Matrix oder einer Tabelle zurück
head(data.frame(x, y))

  x    y
1 1  5.1
2 2 -2.9
3 3  8.7
4 4  4.4
5 5  8.0
6 6  6.3

plot(x, y)
```



DeepL Übersetzung (P)

Es gibt einen sehr interessanten Artikel auf https://trends.rbc.ru zum Thema **'Big Data**. Nur leider in russischer Sprache :(...

Textauszug:

Что такое Big Data?

Big Data или большие данные — это структурированные или неструктурированные массивы данных большого объема. Их обрабатывают при помощи специальных автоматизированных инструментов, чтобы использовать для статистики, анализа, прогнозов и принятия решений.

Сам термин «большие данные» предложил редактор журнала Nature Клиффорд Линч в спецвыпуске 2008 года [1]. Он говорил о взрывном росте объемов информации в мире. К большим данным Линч отнес любые массивы неоднородных данных более 150 Гб в сутки, однако единого критерия до сих пор не существует.

Hier wird gezeigt wie **Python Code** direkt ausgeführt werden kann:

```
# Python Bibliothek 'DeepL' laden
import deepl

# Übersetzung mit DeepL API (Python)
translator = deepl.Translator("c52a9c7d-3198-063c-2bbf-8f67173820ce:fx")
result = translator.translate_text("Что такое Big Data? Big Data или большие данные\n — это структурированные или нест
руктурированные массивы данных большого объема. \nИх обрабатывают при помощи специальных автоматизированных инструмент
ов, \nчтобы использовать для статистики, анализа, \nпрогнозов и принятия решений. \nСам термин «большие данные» предло
жил редактор журнала Nature \nКлиффорд Линч в спецвыпуске 2008 года. \nОн говорил о взрывном росте объемов информации
в мире. \nК большим данным Линч отнес любые массивы неоднородных данных более 150 Гб в сутки, \nоднако единого критери
я до сих пор не существует.", target_lang="DE")

# Übersetzung ausgeben
print(result)

Was ist Big Data? Große Daten oder große Daten
 - sind strukturierte oder unstrukturierte Datensätze mit großen Datenmengen.
Sie werden mit speziellen automatisierten Werkzeugen bearbeitet,
```

```
die für Statistiken und Analysen verwendet werden sollen,
Prognosen und Entscheidungsfindung.
Der Begriff "Big Data" selbst wurde von der Nature-Redaktion geprägt
Clifford Lynch in einer Sonderausgabe 2008.
Er sprach von der explosionsartigen Zunahme von Daten in der Welt.
Lynch definiert Big Data als jede Menge heterogener Daten, die größer als 150 GB pro Tag ist,
aber es gibt noch kein einziges Kriterium.
```

Deepl Übersetzung (R)

Dieser Artikel kann hier vollständig gelesen werden ;)

Textauszug (Fortsetzung):

ЧДо 2011 года анализом больших данных занимались только в рамках научных и статистических исследований. Но к началу 2012-го объемы данных выросли до огромных масштабов, и возникла потребность в их систематизации и практическом применении.

С 2014 на Big Data обратили внимание ведущие мировые вузы, где обучают прикладным инженерным и ИТ-специальностям. Затем к сбору и анализу подключились ИТ-корпорации — такие, как Microsoft, IBM, Oracle, EMC, а затем и Google, Apple, Facebook и Amazon. Сегодня большие данные используют крупные компании во всех отраслях, а также — госорганы. Подробнее об этом — в материале Кто и зачем собирает большие данные?

Hier wird gezeigt wie eine **Python Funktion** aus **R** ausgeführt werden kann:

```
# R lädt mit 'reticulate::source_python' die Python Datei 'py_deepl.py'
source_python(system.file("extdata", "py_deepl.py", package = "ourdata", mustWork = FALSE))

# Ausführen der Python Funktion 'py_deepl'
py_deepl("ЧДо 2011 года анализом больших данных занимались только в рамках научных и статистических исследований. \nНо к началу 2012-го объемы данных выросли до огромных масштабов, \nи возникла потребность в их систематизации и практическом применении. \nС 2014 на Big Data обратили внимание ведущие мировые вузы, \nгде обучают прикладным инженерным и ИТ-специальностям. \nЗатем к сбору и анализу подключились ИТ-корпорации — такие, как Microsoft, \nIBM, Oracle, EMC, а затем и Google, Apple, Facebook и Amazon. \nСегодня большие данные используют крупные компании во всех отраслях, а также — госорганы. \nПодробнее об этом — в материале Кто и зачем собирает большие данные?", "DE")
```

```
Bis 2011 wurden Big Data nur im Rahmen der wissenschaftlichen und statistischen Forschung analysiert.
Doch Anfang 2012 hatte das Datenvolumen enorme Ausmaße angenommen,
und es entstand die Notwendigkeit, sie zu systematisieren und praktisch anzuwenden.
Im Jahr 2014 haben sich die führenden Universitäten der Welt mit Big Data beschäftigt,
wo angewandte Ingenieur- und IT-Berufe gelehrt werden.
Dann werden IT-Konzerne - wie Microsoft,
IBM, Oracle, EMC, und später Google, Apple, Facebook und Amazon.
```

DeepL R-Code

Dies ist der Übersetzungs-Code der **DeepL API** in **R** geschrieben:

```
## inst/extdata/translate.R
## Function 'translate' for translating text using DeepL (www.deepl.com)

# Funktionsname und Parameter
translate <- function(text = NULL,
                      target_lang = "EN"
                      )
{
    # Variable 'responses' leeren
    responses <- NULL

    # Leerzeichen mit '%20' füllen
    text <- stringr::str_replace(gsub("\\s+", "%20", stringr::str_trim(text)), "B", "b")

    # URL der Deepl API
    url <- "https://api-free.deepl.com/v2/translate?"

    # Auth Key
    auth_key <- "c52a9c7d-3198-063c-2bbf-8f67173820ce:fx"

    # URL mit Parametern zusammensetzen
    urlx <- paste0(url,
                   "auth_key=", auth_key,
                   "&text=", text,
                   "&target_lang=", target_lang
                   )

    # Daten des Deepl Servers erhalten
    response <- httr::GET(urlx)
```