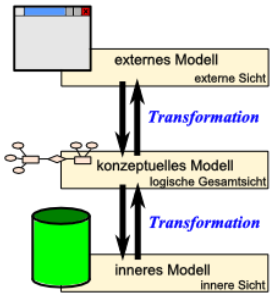
Informatik

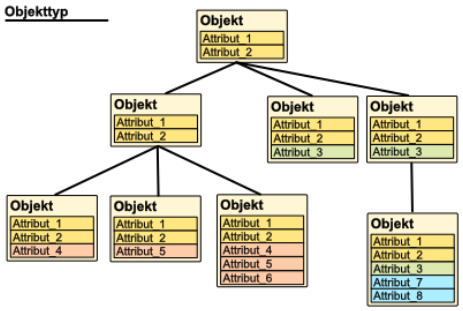
für die Sekundarstufe II

- Datenbanken -

Teil 3: Programmierung, Data Sience, ++

Autor: L. Drews

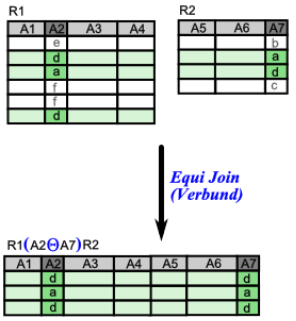




ACCESS

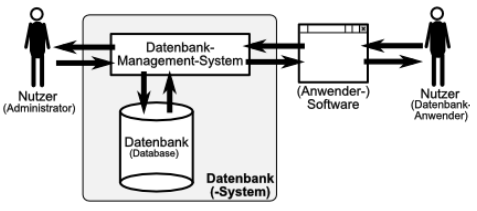
AND | OR | XOR

BASE



Wie jagt ein typischer Programmierer afrikanische Elefanten?

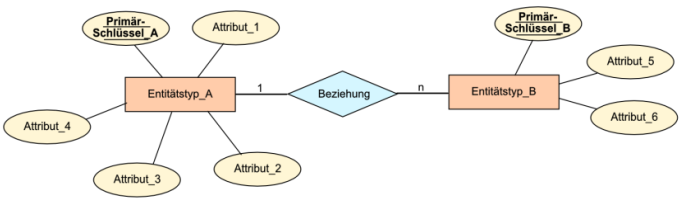
```
{
Gehe nach Afrika
Beginne am Kap der guten Hoffnung
Durchkreuze Afrika von Süden nach Nor-
den bidirektional in Ost-West-Richtung
Für jedes Durchkreuzen tue
   {
   Fange jedes_Tier, das Du siehst
   Vergleiche jedes_Tier mit Elefant
   Halte an bei übereinstimmung
   }
}
```



SQL
NoSQL

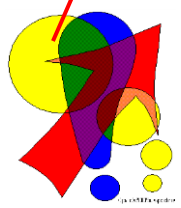
Wie jagt ein SQL-ler afrikanische Elefanten?

```
SELECT elefant
FROM afrika
```



V. 0.12g (2025)

Legende:
mit diesem Symbol werden zusätzliche Hinweise, Tips und weiterführende Ideen gekennzeichnet



Nutzungsbestimmungen / Bemerkungen zur Verwendung durch Dritte:

(1) Dieses Skript (Werk) ist zur freien Nutzung in der angebotenen Form durch den Anbieter (lern-soft-projekt) bereitgestellt. Es kann unter Angabe der Quelle und / oder des Verfassers gedruckt, vervielfältigt oder in elektronischer Form veröffentlicht werden.

(2) Das Weglassen von Abschnitten oder Teilen (z.B. Aufgaben und Lösungen) in Teildrucken ist möglich und sinnvoll (Konzentration auf die eigenen Unterrichtsziele, -inhalte und -methoden). Bei angemessen großen Auszügen gehört das vollständige Inhaltsverzeichnis und die Angabe einer Bezugsquelle für das Originalwerk zum Pflichtteil.

(3) Ein Verkauf in jedweder Form ist ausgeschlossen. Der Aufwand für Kopierleistungen, Datenträger oder den (einfachen) Download usw. ist davon unberührt.

(4) Änderungswünsche werden gerne entgegen genommen. Ergänzungen, Arbeitsblätter, Aufgaben und Lösungen mit eigener Autorenschaft sind möglich und werden bei konzeptioneller Passung eingearbeitet. Die Teile sind entsprechend der Autorenschaft zu kennzeichnen. Jedes Teil behält die Urheberrechte seiner Autorenschaft bei.

(5) Zusammenstellungen, die von diesem Skript - über Zitate hinausgehende - Bestandteile enthalten, müssen verpflichtend wieder gleichwertigen Nutzungsbestimmungen unterliegen.

(6) Diese Nutzungsbestimmungen gehören zu diesem Werk.

(7) Der Autor behält sich das Recht vor, diese Bestimmungen zu ändern.

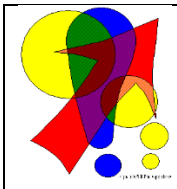
(8) Andere Urheberrechte bleiben von diesen Bestimmungen unberührt.

Rechte Anderer:

Viele der verwendeten Bilder unterliegen verschiedensten freien Lizenzen. Nach meinen Recherchen sollten alle genutzten Bilder zu einer der nachfolgenden freien Lizenzen gehören. Unabhängig von den Vorgaben der einzelnen Lizenzen sind zu jedem extern entstandenen Objekt die Quelle, und wenn bekannt, der Autor / Rechteinhaber angegeben.

public domain (pd)	Zum Gemeingut erklärte Graphiken oder Fotos (u.a.). Viele der verwendeten Bilder entstammen Webseiten / Quellen US-amerikanischer Einrichtungen, die im Regierungsauftrag mit öffentlichen Mitteln finanziert wurden und darüber rechtlich (USA) zum Gemeingut wurden. Andere kreative Leistungen wurden ohne Einschränkungen von den Urhebern freigegeben.
gnu free document licence (GFDL; gnu fdl)	
creative commens (cc) creative commons	BY: od. neu … Namensnennung … nichtkommerziell … in der gleichen Form … unter gleichen Bedingungen

Die meisten verwendeten Lizenzen schließen eine kommerzielle (Weiter-)Nutzung aus!



Bemerkungen zur Rechtschreibung:

Dieses Skript folgt nicht zwangsläufig der neuen **ODER** alten deutschen Rechtschreibung. Vielmehr wird vom Recht auf künstlerische Freiheit, der Freiheit der Sprache und von der Autokorrektur des Textverarbeitungsprogramms microsoft ® WORD ® Gebrauch gemacht.

Für Hinweise auf echte Fehler ist der Autor immer dankbar.

Inhaltsverzeichnis:

Seite

6. Datenbanken - Programmierung

Die Programmierung von und mit Datenbanken gehört heute zu den wichtigsten Aufgaben der Programmierer für Anwender- und Internet-Software.

Grundsätzlich unterscheidet man zwei grundsätzlich verschiedene Programmierungs-Arten. Da ist zum Einen die Erstellung von Software-Lösungen aus dem Datenbank-System selbst heraus. Dabei wird z.B. eine im System enthaltene Makro- und / oder Programmiersprache genutzt. Microsoft ACCESS gehört z.B. zu diesem Programmier-Typ. Mit der Programmier-Umgebung lassen sich alle Komponenten wie Tabellen, Abfragen, Formulare, Berichte und auch die Makros nutzen. Für den Anwender entsteht der Eindruck, er arbeitet mit einem speziellen Programm. Die Datenbank als solche ist für den Nutzer kaum sichtbar.
Die zweite Art ist die Programmierung einer Software mit quasi einer frei gewählten Programmiersprache. Die relativ komplexe Daten-Verwaltung lagert der Programmierer an ein Datenbank-System aus. Über eine spezielle Software-Schnittstelle für die gewählte Programmiersprache kann auf eine oder mehrere Datenbank-Schnittestellen zugegriffen werden. Die meisten verwenden **SQL** (→ 5. SQL – die Datenbank-Sprache). Vereinfacht kann man sich das so vorstellen, dass ein spezieller Output-Befehl – nennen wir in hier **writeSQL** genutzt wird, um der Datenbank etwas mitzuteilen. Mit einem äquivalenten Input-Befehl – hier z.B. **readSQL** – fragt man bei der Datenbank an und erhählt auswertbare Daten für das zu erstellende Programm.
Quasi eine Kombination aus beiden Varianten stellen Web-Datenbanken und zugehörige Applikationen dar. Sie basieren auf Datenbanken unterschiedlichster Art. Das geht bei einfachen CSV- und XML-Dateien los und geht weiter bis zu SQL- und NoSQL-Datenbanken mit professioneller Ausrichtung.
Zur Programmierung der Applikationen werden Script-Sprachen, wie JavaScript, PHP, Perl oder Ruby genutzt. Seit einigen Jahren nimmt auch die Programmierung über HTML5 immer schnellere Fahrt auf.
Wir werden uns zuerst ganz kurz die erste Variante an. Sie ist immer sehr speziell, da die Programmiersprache des Datenbank-System oder z.B. der Office-Suite benutzt wird (→ 6.1. Datenbank-Programmierung mit ACCESS-VBA). Hier gehen wir auch auf die standardisierten Schnittstellen (→ 6.2. universelle Datenbank-Schnittstellen) – z.B. eben ODBC – ein (→ 6.2.1. die ODBC-Schnittstelle).
Die zweite Herangehensweise mit den klassischen Programmiersprachen, wie z.B. Python (ausführlich: → 🕮 **Programmieren mit Python**) und JAVA (ausführlich:→ 🕮 **Strukturierte und Objekt-orientierte Programmierung mit JAVA**) folgen dann anschließend. Für die meisten Kurse wird sicher nur eine der beiden Sprachen Python (→ 6.3. Datenbank-Zugriff mit Python) oder JAVA (→ 6.4. Datenbank-Zugriff mit JAVA) interessant sein.
In einem gesonderten Abschnitt folgen einige Ausführungen zu den Web-Datenbanken und –Applikationen (→ 7. Web-Datenbanken). Hier ist das Feld riesig. Eine weit verbreitete Kombination von mehreren aufeinander abgestimmten Programmen ist XAMPP (→ 7.1. Arbeiten mit XAMPP). Diese lässt sich auch über den IoStick (→ https://tinohempel.de/info/info/IoStick/index.html) von T. HEMPEL nutzen.

6.0. Vorüberlegungen / Grundlagen

In den konzeptionellen Besprechungen (→ 2.1.2. übergreifende / übergeordnete Modelle) haben wir schon mehrere Schicht- oder Ebenen-Modelle vorgestellt. Sie dienten immer einer passenden Modellierung bzw. der Abtrennung von Aufgaben / Funktion.
Will man eine Datenbank programmieren, dann wird man bald von der Komplexität der Aufgaben überwältigt. Hier hat sich, wie fast überall in der Informatik und speziell in der Programmierung eine Aufteilung von Aufgaben in Schichten (sogenannten Tier) bewährt. Man spricht auch von einer Schicht-Architektur. Selbst einzelne Schichten sind meist noch so komplex, dass weitere Aufteilungen erfolgen müssen.
Programmierer und Techniken konzentrieren sich auf einzelne Schichten und füllen diese entsprechend aus. Für Übergänge zu anderen Schichten, werden Schnittstellen geschaffen. Die Kommunikation zwischen den Schichten läuft nur über diese Schnittstellen. Da jede Schicht für sich eigenständig gehändelt werden, lassen sie sich gut testen, ersetzen, aktualisieren oder auch völlig neu konzeptieren.
Die Anwender / Nutzer der Schichten brauchen nur die Schnittstellen kennen, die üblicherweise auch ausführlich dokumentiert sind. Wie die Programmierer oder Techniker eine Schicht intern realisieren, bleibt deren Kompetenz.

Single-Tier

In Single-Tier-Programmen werden alle Funktionen in einem Komplex realisiert. Das macht bei kleinen und sehr speziellen Lösungen Sinn. Für größere Projekte ist eine Aufteilung der Aufgaben in spezielle Schichten unanbdingbar.

Two-Tier

Zu den Zwei-Schicht-Architekturen gehören die klassischen Client-Server-Systeme. Client's und Server stellen dabei eigene Schichten dar. Die Client's stellen Anfragen an die Server. Diese ermitteln eine Antwort und senden diese an die Client's zurück.
In Datenbank-Systemen übernehmen die Server die Aufgaben der Daten-Haltung (Speicherung), Administration und Daten-Zusammenstellung. Die Client's übernehmen die zusammengestellten Rohdaten und bereiten sie für die Nutzer auf. Das ist zumeist die Fein-Darstellung (Formate, …) und Präsentation (Diagramme, Tabellen, …).
Client und Server müssen keine eigenständigen Rechner sein. Sie können als Software auf einem Rechner nebeneinander funktionieren.

Three-Tier

Die klassische Drei-Schicht-Architektur umfasst die Schichten:

- **data-server tier**
  back end
  Datenhaltungs-Schicht
  enthält die eigentliche Datenbank (gespeicherte Daten)
  übernimmt das Indexieren der Datenbestände
  realisiert den Datenschutz
  beinhaltet die Datensicherung
  stellt (große Mengen von) Daten zur Verfügung

- **application-server tier**
  middle tier; business tier
  enterprice tier
  Logik-Schicht
  verknüpft die (großen) Daten-Mengen sinnvoll miteinander
  beinhaltet die eigentliche Daten-Verarbeitung

- **client tier**
  front end
  Präsentations-Schicht
  verantwortlich für die Nutzer-gerechte Darstellung der Daten
  realisert Daten-Ein- und –Aus-gabe

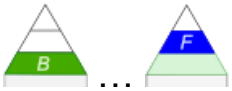
Jedwede Kommunikation und Daten-Transporte erfolgen immer über die – in der Mitte liegende – Logik-Schicht (middle-tier)
Client-Tier ← → Middle Tier ← → Database Tier

Aufgaben:

1. *Welche Schichtungen / Schicht-Modelle der Informatik kennen Sie? Stellen Sie diese ganz kurz vor!*
2. *Informieren Sie sich unter:*
   http://www.dfpug.de/konf/konf_1998/09_tier/d_tier/d_tier.htm
   über die Vor- und Nachteile von Tree-Tier-Programm-Systemen! Notieren Sie sich die Informationen in kompakter Form!
3.

6.1. Datenbank-Programmierung mit ACCESS-VBA

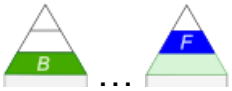
Die Nutzung von Datenbank-Komponenten aus Office-Produkten wird im neuen Rahmenplan "Informatik" (Sek.II; für MV) abgelehnt.
Für Einsteiger-Projekte sind sie aber sehr gut geeignet!



VBA ist die BASIC-artige Programmiersprache der Office-Programme von Microsoft-Office. Dabei handelt es nicht etwa um eine billige BASIC-Variante, sondern um ein sehr Leistungsfähiges Programmier-System.
Das gesamte System zu besprechen, würde dieses Skript bei weitem sprengen. Da gibt es sehr gute Spezial-Literatur. Viele Bücher besprechen nur einzelne Teil-Sprachen für eines der Office-Produkte. Man braucht auch nicht immer unbedingt ein Buch für die aktuellste Version. So ein, zwei Versionen zurück betrachtet, hat sich meist nicht viel getan. Für spezielle Leistungen, der erst in der aktuellsten Version dazugekommen sind, ist natürlich dann auch nur die aktuellste VBA-Version interessant.
Für die älteren Versionen erhält man oft in speziellen online-Buchhandlungen (z.B. terrashop.de oder Jokers.de) stark verbilligte Bücher. Diese sind oft als Mängelexemplare deklariert, aber nicht wirklich dieser Bezeichnung wert.

6.1.1. Arbeiten mit Makro's

Die Nutzung von Datenbank-Komponenten aus Office-Produkten wird im neuen Rahmenplan "Informatik" (Sek.II; für MV) abgelehnt.
Für Einsteiger-Projekte sind sie aber sehr gut geeignet!



Ein guter Einstieg ist die Benutzung von Makro's in einem Office-Programm. Makro's sind Sequenzen von Tätigkeiten in einem Office-Programm, die man sehr häufig braucht.
Der erste Zugang ist vielleicht, ein vorhandenes Makro zu nutzen. Über eine einfache Tasten-Kombination wird dann eine Sequenz von Befehlen in einem Ritt abgearbeitet. Mit der Schrittfolge hat der einfache Nutzer nichts direkt zu tun.
Irgendwann erstellt man ein erstes Makro. Dabei werden z.B. Maus- oder Tasten-Befehle aufgezeichnet. Das Makro erhält einen Namen und man kann ihm eine bestimmte Tasten-Kombination zuordnen. Immer wenn dann später in dem Programm diese Tasten-Kombination gedrückt wird, kommt es zur Abarbeitung der aufgezeichneten Befehls-Folge.
Die Befehl-Folge kann man sich ansehen und sie auch manipulieren / ändern. Das Ansehen bringt schon ein gewisses Verständnis für die Arbeitsweise des VBA-Systems. Kleine Änderungen lassen sich dann auch noch gut ausprobieren.
Das ist quasi die dritte Nutzungs-Ebene von Makro's. Die vierte Ebene schließt dann das Erstellen eines Makro's bzw. eines Programm's über einen Text-Editor ein. Einen solchen findet man im Office. Die höchsten Weihen sind dann Datenbank-Anwendungen, bei denen ein unbedarfter Nutzer gar nicht merkt, dass es sich um eine Office-Anwendung handelt.

6.2. universelle Datenbank-Schnittstellen

Aufgabe von Datenbank-Schnittstellen ist es, den inneren Aufbau und irgendwelche Implementierungs-Details vor fremden Anwendungen und Anwendern zu verstecken. Die Schnittstelle bietet einen standardisierten Zugriff auf das Datenbank-Management-System (DBMS).
Diese Schnittstelle wird mehr oder weniger offen gestaltet und veröffentlicht. Durch die Nutzung der Schnittstelle können die eigene Anwendung und das DBMS völlig unabhängig entwickelt werden, die Funktionalitäten sind stabil und dauerhaft über die Schnittstelle definiert.
Für die eigene Anwendung ist es nicht wichtig, wie und wo die Daten gespeichert sind. Das ist und bleibt die Domäne des Datenbank-Management-Systems.
Eigentlich jedes DBMS bietet eine eigene Schnittstelle für Programmierer (auch die aus der eigenen Firma und eigene Datenbank-Anwendungen) an. Diesen bieten den maximalen Nutzeffekt. Man kann praktisch auf alle Leistungen des DBMS zurückgreifen.
Dieser Vorteil wird dann zum Nachteil, wenn Nutzer andere DBMS benutzen wollen oder müssen. Das können z.B. alte Daten-Bestände sein oder finanzielle Gründe. Heute gibt es schließlich viele DBMS auch in abgespeckter Version für freie Zwecke.
Die meisten “normalen“ Anwendungen benötigen gar nicht den vollen Funktions-Umfang einer Hochleistungs-Datenbank. Vielmehr ist in Zeiten vielen Unbeständigkeiten (Pleiten, Aufkäufe, …) eine Kontinuität, Ersetzbarkeit und Flexibilität gefragt. Dafür wurden schon frühzeitig sehr universelle Schnittstellen erschaffen, von denen wir hier die ODBC-Schnittstelle (→ 6.2.1. die ODBC-Schnittstelle) ausführlicher vorstellen werden.
Als Gegen-Part haben Borland, IBM, Novell und WordPerfect Corp. die IDAPI-Schnittstelle (Integrated Database Application Programming Interface) entwickelt.
Weitere – zum Teil auch breiter angewendete – Schnittstellen sind BDE (Borland Database Engine), die Perl DBI (Perl Database Independet) und für microsoft Visual BASIC die ADO (ActiveX Data Objects).

6.2.0. Grundlagen der Datenbank-Schnittstellen

Im Allgemeinen benötigt man bei Datenbanken zwei unterschiedliche Zugriffe. Das ist zum Einen die Kontakt-Aufnahme zur Datenbank einschließlich der Authentifizierung und zum Zweiten die eigentliche Datenbank-Arbeit.
Man kann sich den Verbindungs-Aufbau wie ein Stecker-System vorstellen. Wenn der Stecker passt, dann können die Daten über die Kontakte ausgetauscht werden.

6.2.1. die ODBC-Schnittstelle

Die ODBC-Schnittstelle ist eine der verbreitesten Schnittstellen-Definition. Sie basiert auf SQL und wurde von Microsoft entwickelt. ODBC steht dabei für Open Datenbase Connectivity.
Viele Datenbank-Management-Systeme bieten eine entsprechende Schnittstelle an. Man kann deshalb ODBC auch als einen allgemeingültigen Standard betrachten.
Wie schon oben schon kurz beschreiben, wird bei ODBC eine Trennung zwischen der Verbindung und dem Daten-Austausch vorgenommen.
Bei der Verbindungs-Definition wird ein Client auf dem Nutz-System eingerichtet. Dieser enthält alle – z.T. auch sehr sensiblen – Daten für die Verbindungs-Aufnahme zur Datenbank. Dazu gehören z.B. der Speicherort oder die Zugangs-Daten.
Über den – meist nur einmalig einzurichtenen – Client kann die Nutzer-Software die Verbindung zur – irgendwo liegenden – Datenbank aufnehmen und dann Daten austauschen.
Der – später folgende – eigentliche Daten-Austausch erfolgt über SQL-Anweisungen.

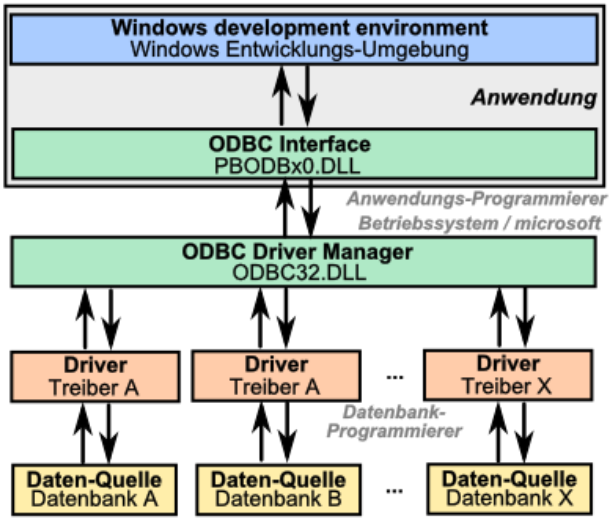
Funktionalitäts-Stufen bei ODBC

- **Core** beinhaltet die Basis-Funktionalität
- **Level 1**
- **Level 2**

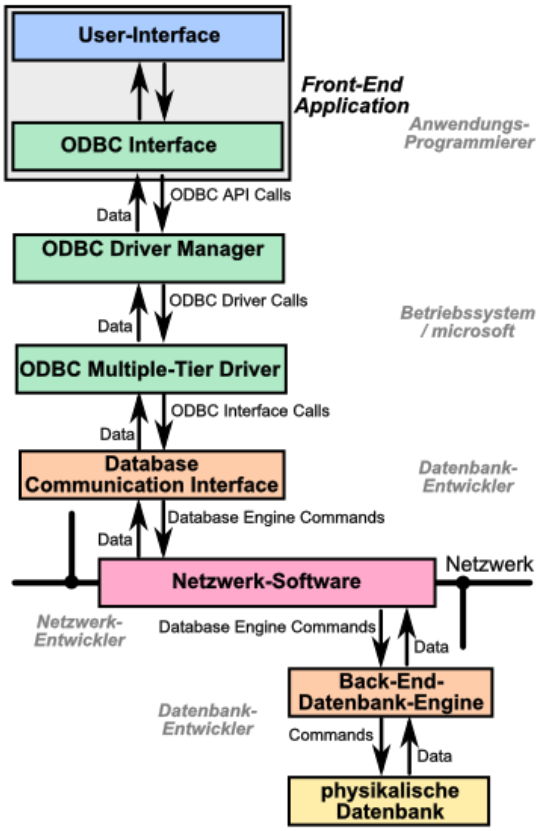
besteht aus "Front-end"- oder Client-Treiber für die Applikations-/Nutzer-Seite und einem Treiber für das DBMS – dem sogenannten "Back-end"- oder Server-Treiber

die Front-end-Seite wird auch ODBC-Client genannt
daneben gehören noch der ODBC-Server und der DBMS-Server zu den drei notwendigen Komponenten für ein funktionierendes ODBC-System
die beiden letzten Komponenten werden typischerweise vom DBMS installiert bzw. bereitgestellt

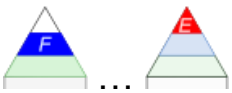
der ODBC-Server gehört aber zu den ODBC-Teilen, die microsoft entwickelt hat, nur der DBMS-Server muss vom Entwickler des DBMS zur Verfügung gestellt werden
für den praktischen Anwender ist eigentlich nur die Einrichtung des ODBC-Clients wichtig, da auf modernen System die ODBC-Treiber meist schon installiert sind



Die Netzwerk-Software beinhaltet dann die ISO/OSI-Schichten. Die Anzahl der genutzten Schichten ist also schon gewaltig. Umso überraschender ist es, dass die Kommunikation doch so reibungslos abläuft. Das spricht für die Qualität der Schnittstellen und der dahinter liegenden Programme.

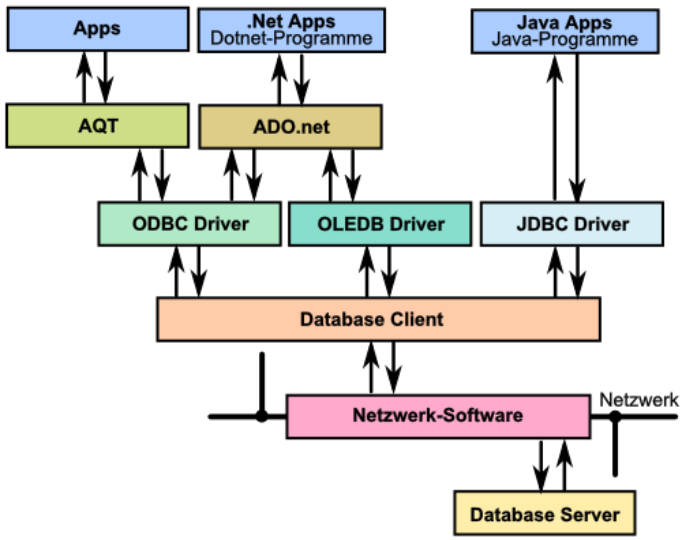


6.2.2. weitere offline-Datenbank-Schnittstellen



den Begriff offline sollte man hier nicht so ernst nehmen
gemeint sind hier Schnittstellen, die für direkte Computer-Verbindungen gemacht worden sind

moderne Fortsetzung / Weiterentwicklung ist die OLEDB, die mit dem dotNet-Framework in Windows-Systeme einzug gehalten hat
praktisch parallel zum ODBC-Treiber



IDAPI (Integrated Database Application Programming Interface) wurde als Alternativ- bzw. Kongurrenz-Produkt von Borland, IBM, Novell und Lotus/Wordperfect entwickelt (1992)

ähnlicher Aufbau und Prinzip
verbesserte Funktionalität, kann z.B. auf mehrere / unterschiedliche Datenbanken gleichzeitig zugreifen
hohe Performanz
in der Praxis ist IDAPI eher eine Erweiterung von ODBC als ein eigener Standard

bietet auch Konnektivität zu Datenbanken, die auf dem Standard xbase basieren

6.2.3. weitere online-Datenbank-Schnittstellen

hier verstehe ich solche Schnittstellen, die auf Internet-Verbindungen basieren
das dürfen natürlich auch interne TCP/IP-Verbindungen sein

6.2.3.x. CKAN

Beispiel:
SQL-Ausdruck

```
SELECT bezeichnung, strasse_name, hausnummer, hausnummer_zusatz
FROM "0c2c4996-afad-4ebe-9a3d-a3a7d7047a4d"
WHERE ST_Within(ST_Transform(geometrie,25833), ST_Buffer(ST_Transform((
    SELECT geometrie
    FROM "e614d725-4fad-4447-820d-4475ca55cff5"
    WHERE bezeichnung = 'Deutsche Med'),25833),300))
```

passender CKAN-Aufruf:
https://www.opendata-hro.de/api/action/datastore_search_sql?sql=SELECT bezeichnung, ⮐
strasse_name, hausnummer, hausnummer_zusatz ⮐
FROM "0c2c4996-afad-4ebe-9a3d-a3a7d7047a4d" WHERE ⮐
ST_Within(ST_Transform(geometrie,25833),ST_Buffer(ST_Transform((SELECT ⮐
geometrie FROM "e614d725-4fad-4447-820d-4475ca55cff5" WHERE bezeichnung =⮐
'Deutsche Med'),25833),300))

Links:
http://docs.ckan.org/en/ckan-2.7.3/api (Beschreibung / Dokumentation der API (engl.))

6.3. Datenbank-Zugriff mit Python

Bei Python steht mehr der schnelle Skript-orientierte Zugriff auf Datenbanken im Vordergrund. Natürlich kann man auch Fenster- und / oder Objekt-orientierte Programme erstellen.

6.3.0. Vorarbeiten / Installation

Eine der Bibliotheken für den Datenbank-Zugriff unter Python wird Connector genannt

für MySQL z.B. unter http://dev.mysql.com/downloads/connector/python/ zu downloaden
vorher Python-Version erkunden

import sys
print(sys.version)

od. bei Start des Python-Systems in der obersten Zeile ablesen
passende Version von der Download-Seite runterladen

mit Admin-Rechten installieren

6.3.1. Vorbereitung

Um eine Datenbank nutzen zu können, müssen einige Vorarbeiten getätigt werden. Dabei handelt es sich im Wesentlichen um den Verbindungs-Aufbau und die Authentifizierung bei der Datenbank.

```
import mysql.connector # neue Bibliothek

Servername = 'localhost' # Rechnername (localhost ist der eigene Rechner)
Benutzer   = 'mustermann'  # Nutzeraccount bei der Datenbank
Passwort   = 'sehrgeheim'  # passendes Passwort
Datenbank  = 'InfoDB'      # Name der Datenbank, die genutzt werden soll

# Verbindung mit der Datenbank
con = mysql.connector.connect(host=Servername)
con.cmd_change_user(username = Benutzer, password = Passwort)
con.database = Datenbank
```

In unserem obigen Code-Schnipsel ist con die Variable, die für eine konkrete Datenbank-Anbindung steht. Über diese Variable läuft dann später der gesamte Daten-Verkehr zwischen der eigenen Applikation (mit den internen Daten-Strukturen) und der Datenbank (mit ihrer internen Daten-Struktur).

6.3.1.1. Zugriff auf entfernte Datenbanken

Sachlich ist die Anbindung einer externen Datenbank – irgendwo im Internet – nicht anderes, als die Verbindung zu einer lokalen. Ändern tut sich eigentlich nur die Host-Adresse.

```
con = mysql.connector.connect(host=Servername)
```

Port 3306 muss ev. bei Firewall's freigegeben werden

```
con.cmd_change_user(username = Benutzer, password = Passwort)

con.database = Datenbank
```

oder alles kombiniert:

```
con = mysql.connector.connect(
            host     = Servername,
            user     = Benutzer,
            password = Passwort,
            database = Datenbank )
```

6.3.2. Daten-Zugriff über SQL-Befehle

Für das Arbeiten mit den Daten einer Datenbank braucht man nur die Verbindung. Ob die Datenbank lokal oder entfernt ist, bleibt für den Nutzer verborgen. Nur wenn sein Netzwerk oder Internet nicht funktioniert, wird er Probleme mit dem Arbeiten bekommen. Dann läuft praktisch nichts mehr. Gute Datenbank-Anwendungen werden die Datenbank oder Teile davon z.B. in der Nacht repliziert haben und als lokale Datenbank irgendwo lokal abgelegt haben. Dann kann der Nutzer mit dieser Datenbank arbeiten und in der nächsten Replikations-Phase werden Änderungen zwischen den entfernten und lokalen Datenbank abgeglichen.
Das verdeutlicht die Vorteile der Trennung von Verbindungs-Aufbau und dem eigentlichen Daten-Austausch.
Es muss aber am Schluß unbedingt die Verbindung zur Datenbank noch geschlossen werden! (→ 6.3.4. Nachbereitung)

6.3.2.1. Daten-Abfrage

```
# SQL-Befehl ausführen
cursor = con.cursor()
SQLBefehl = 'SELECT Name, Einwohner FROM kontinent'
cursor.execute(SQLBefehl)


# Durchlaufen der Ergebnisse
row=cursor.fetchone()
while (row!=None):
  print(row[0], row[1])
  row = cursor.fetchone()
```

fetchone holt einen (passenden) Datensatz, der dann im Folgenden ausgewertet werden kann
beim nächsten Aufruf von fetchone ist dann der nächste (passende) Datensatz gemeint
fetchone lässt sich also gut in Schleifen-Strukturen nutzen
Der Datensatz (hier als Zeile (row) interpretiert) wird als Liste aufgefasst und kann über Listen-Befehle oder Index-Zugriffe manipuliert werden

oder mit Feld-Namen:
und Daten in ein Directory

```
cursor = con.cursor(dictionary=True)
SQLBefehl = "SELECT Name, Einwohner FROM kontinent"
cursor.execute(SQLBefehl)

row=cursor.fetchone()
while (row!=None):
  print(row['Name'], row['Einwohner'])
  row = cursor.fetchone()
```

6.3.2.2. Veränderung von Tabellen

hier kommen die SQL-Befehle UPDATE und INSERT zum Tragen
s.a. → 5. SQL – die Datenbank-Sprache

Struktur-Veränderungen von Tabellen sind immer sehr aufwändig und Risiko-behaftet (s.a. → 6.3.2.2.3. Verändern der Tabellen-Struktur). Besser ist natürlich immer eine vorausschauende Planung einer Datenbank (→ 2.1. Datenbank-Modellierung).

6.3.2.2.1. Erstellen einer neuen Tabelle

```
[...]
SQLBefehle = """
CREATE TABLE IF NOT EXISTS schueler (
  SNR     integer NOT NULL,
  Vorname varchar(30),
  Name    varchar(50) NOT NULL,
  PRIMARY KEY (SNR)
);
CREATE TABLE IF NOT EXISTS fahrstunde (
  SNR    integer,
  Datum  date,
  Preis  integer,
  Zeit   time,
  PRIMARY KEY (SNR, Datum),
  FOREIGN KEY (SNR)
        REFERENCES Schueler(SNR)
);
"""
cursor = con.cursor()

try:
  for einzelCursor in cursor.execute(SQLBefehle, multi=True):
    print(einzelCursor._executed.decode('UTF-8'))


except mysql.connector.Error as err:
  print("Fehler bei der SQL-Ausführung: %s" % (err))
[...]
```

Anweisungs-Struktur	**Erklärung**	**Beispiel / Hinweise**
Attributname **varchar(***Länge***)**	definiert ein Attribut mit dem angegebenen Attributnamen -> Datentyp ist Zeichenkette mit einer angegebenen *Länge*	
Attributname **integer**	… -> Datentyp ist Ganzzahl	
Attributname **real**	… -> Datentyp ist Fließkommazahl	
Attributname **date**	… -> Datentyp ist Datum	
Attributname **time**	… -> Datentyp ist Zeit	
Attributname **blob(***Größe***)**	… -> Datentyp ist unbestimmt; es wird Speicherplatz für maximal *Größe* Byte reserviert	
*Schlüsse*l **integer NOT NULL**	NOT NULL bestimmt, dass dieses Attribut nicht leer sein darf → sonst Fehlermeldung	
PRIMARY KEY (*Schlüssel***)**	legt Attribut Schlüssel als Primär-Schlüssel fest	
FOREIGN KEY (*Schlüssel***) REFERENCES** *Tabellenname(Schlüssel_dort)*	legt Attribut *Schlüssel* als Sekundär-Schlüssel fest (dieser befindet sich in der Tabelle mit dem angegebenen Tabellennamen und heißt in dieser *Schlüssel_dort*)	

```
print("Neuanlage eines Fahrschülers")
print("============================")

neuName    =     input("Name   : ")
neuVorname =     input("Vorname: ")

# höchste Nummer finden
cursor = con.cursor()
SQLBefehl = 'SELECT MAX(SNR) FROM schueler'
cursor.execute(SQLBefehl)
row=cursor.fetchone()

neuSNR = int(row[0])+1


SQLBefehl = "INSERT INTO schueler (SNR, Vorname, Name) VALUES
(%i,'%s','%s')" \
          % (neuSNR, neuName,neuVorname)
cursor.execute(SQLBefehl)
print(cursor._executed.decode('UTF-8'))
cursor.close()
```

bei MySQL ist spezielle Anweisung für die Erzeugung eines Schlüssel verfügbar, der die Schlüssel-Nummerierung automatisiert (AUTO_INCREMENT)

```
CREATE TABLE IF NOT EXISTS schueler (
  SNR          integer NOT NULL AUTO_INCREMENT,
  Vorname      varchar(30),
  Name         varchar(50),
  BevorzugtFNR integer,
  PRIMARY KEY (SNR)
)
```

```
print("Neuanlage eines Fahrschülers")
print("============================")

neuSNR     = int(input("SNR    : "))
neuName    =     input("Name   : ")
neuVorname =     input("Vorname: ")


SQLBefehl = "INSERT INTO schueler (SNR, Vorname, Name) VALUES
(%i,'%s','%s')" \
          % (neuSNR,neuName,neuVorname)

cursor = con.cursor()
cursor.execute(SQLBefehl)
# print(cursor._executed.decode('UTF-8'))
cursor.close()

con.commit()
```

```
; Ändert den Namen des Schülers mit der Nummer 11.
UPDATE schueler
   SET name="Konda"
 WHERE SNR = 11

; Löscht alle schueler, deren SNR kleiner als 20 ist.
DELETE FROM schueler
 WHERE SNR < 20
```

Aufgaben:

1. ***Erstellen Sie ein Python-Programm, dass die Datensätze einer vorgegebenen Tabelle aus einer Datenbank anzeigt!***
2. ***Erweitern Sie das Fahrschul-Programm um die Möglichkeit einzelne Fahrstunden einzugeben und in der Datenbank zu speichern!***
3. ***Konzipieren und realisieren Sie ein Programm, dass einen einfachen Menü-basierten Umgang mit einer Datenbank ermöglicht! Einigen Sie sich im Kurs auf einen minimalen Funktions-Umfang! (Die Kursteilnehmer mit dem erhöhten Anspruch realisieren zusätzliche Funktionen!)***
4.

6.3.2.2.2. Löschen einer Tabelle

6.3.2.2.3. Verändern der Tabellen-Struktur

```
ALTER TABLE schueler
  ADD COLUMN GebDat date DEFAULT NULL
```

6.3.2.3. Abarbeitung großer SQL-Skripte

```
  for einzelCursor in cursor.execute(SQLBefehle, multi=True):
    print(einzelCursor._executed.decode('UTF-8'))
```

SQLBefehle ist im obigen Programm eine vorher zu definierende und zu belegende Liste mit SQL-Statements

6.3.3. SQL-Fehler abfangen

dazu benutzt man die Exception-Strukturen von Python

```
cursor = con.cursor()
SQLBefehl = 'SELECT Name, Einohner FROM kontinent'
try:
  cursor.execute(SQLBefehl)

  # Abrufen der Ergebnisse
  [...]

except mysql.connector.Error as err:
  print("Fehler bei der SQL-Ausführung: %s" % (err))
```

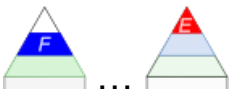
Für einfache Versuche mit Datenbanken sind solche Strukturen oft zu aufwändig. In echten "Arbeits"-Anwendungen sind sie aber elementar wichtig. Ohne Absicherungen kann es zum Absturz der Applikation und damit auch zum Datenverlust kommen.

6.3.4. Nachbereitung

Die Verbindung zu einer Datenbank (→) muss am Ende unbedingt ordnungsgemäß geschlossen werden. Ansonsten wäre eine externe Manipulation unter dem angemeldeten Account möglich.

```
# Ende der Verarbeitung
cursor.close()
# Abmelden
con.disconnect()
```

6.3.5. Gefahren beim programmtechnischen Umgang mit SQL



SQL-Injection

ein Benutzer versucht durch Manipulation Sicherheitslücke oder Anwendungs-Fehler für eigene Zwecke zu benutzen

z.B. wird eine Eingabe (z.B. ein Text durch SQL-Sequenzen ergänzt / ersetzt

Geben Sie einen Namen ein: Meier**' OR '1'='1**

diese werden in die benutzte SQL-Anweisung mit eingebaut und dann ausgeführt
z.B. DELETE "Meier" **OR 1 = 1**

Da "1 = 1" immer zutrifft, wird der SQL-Befehl ausgeführt. In diesem Fall das Löschen für alle Datensätze. Solche Unzulänglichkeiten in Programmen (! und nicht in den Datenbanken!) lassen schnell einen Alptraum wahrwerden. Ganze Datenbanken lassen sich u.U. so löschen.
Die nächste Situation zeigt dieses Horror-Szenario mit einem einfachen SQL-Statement in einer Eingabe.

Geben Sie einen Namen ein: M%**'; DROP TABLE** *Tabellenname1***; DROP TABLE** *Tabellenname2***;**

Maßnahmen zur Reduzierung / Verhinderung von Datenbank-Angriffen

- Beschränkung der Rechte des Anwender (der Applikation) bzw. der Applikation bezüglich / innerhalb der Datenbank
- Kontrolle von Nutzereingaben (soll z.B. nur ein Datensatz mit DELETE gelöscht werden, dann kann durch Mitzählen (cursor.rowcount) erfasst werden, dass mehrere Aktionen ausgeführt worden → dann kann man mit con.rollback() statt con.commit() die gemachten Änderungen sofort wieder rückgängig machen)
- Vermeiden vom Multi-Cursor's

6.3.6. Projekt: Darstellung von Positionen in einer geographischen Karte

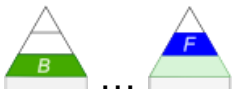
```
# Konstanten: Grenzen der Karte in Grad
cNord = 55.1
cSued = 47.2
cWest = 5.5
cOst  = 15.5

def InPixelNS(bild, grad):
  """ Umrechnung Geokoordinate in Bildkoordinate. Übergabe: Bild (für des-
sen Höhe) und Breitengrad """
  erg = bild.height()-round((grad-cSued)*bild.height()/(cNord-cSued))
  return erg

def InPixelWO(bild, grad):
  """ Umrechnung Geokoordinate in Bildkoordinate. Übergabe: Bild (für des-
sen Breite) und Längengrad """
  erg = round((grad-cWest)* bild.width()/(cOst-cWest))
  return erg
```

Q: https://www.inf-schule.de/information/datenbanksysteme/zugriff/pythonzugriff/projekt_karte

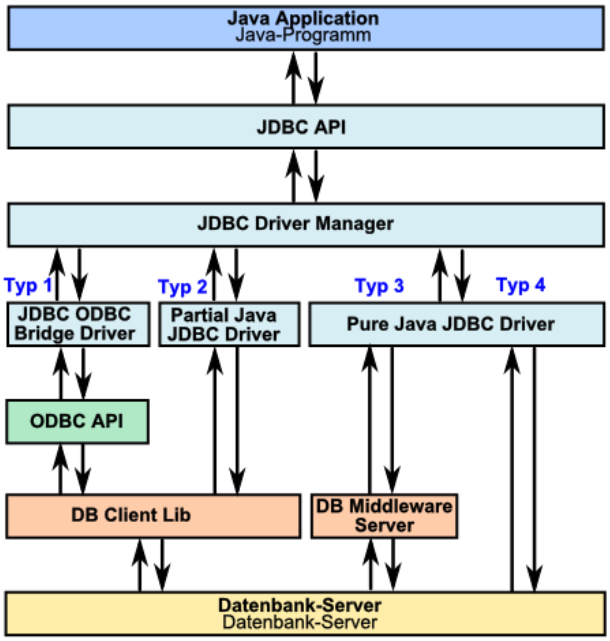
6.4. Datenbank-Zugriff mit JAVA



Bei JAVA steht von Anfang an der Objekt-orientierte Zugang im Vordergrund.

von Sun entwickelt Java Database Connectivity (JDBC)
entspricht im Prinzip der ODBC-Schnittstelle
ist universell verwendbar, aber schon stark auf die Nutzung in JAVA orientiert

Prinzip der Teilung in Verbindungs-Aufbau und SQL-Daten-Verarbeitung



Treiber-Typen bei JDBC

- **Typ-1** kommuniziert über eine sogenannte JDBC-ODBC-Brigde mit der Datenbank, d.h. für die Datenbank muss ein ODBC-Treiber vorhanden und installiert sein
  sollte nur verwendet werden, wenn für die Datenbank eine ODBC-, aber keine JDBC-Schnittstelle gibt

- **Typ-2** hier läuft die Kommunikation mit der Datenbank über eine Plattform-abhängige Programm-Bibliothek auf dem Client

- **Typ-3** setzt die JDBC-Schnittstellen-Befehle (API-Befehle) auf dem Client in generische (originale) Befehle des genutzten DBMS um, diese werden dann über das Netzwerk an einen speziellen Zwischen-Server (Middleware) übertragen, der dann die eigentliche Datenbank-Nutzung realisiert
  eventuelle Ergebnisse einer Datenbank-Abfrage werden, ausgehend vom Middleware-Treiber, über das Netzwerk an den Client übertragen

- **Typ-4** kommuniziert direkt über die JDBC-API-Funktionen mit dem DBMS, der Typ-4-Treiber kommuniziert direkt über das Netzwerk mit dem Datenbank-Server

Anbindung per ODBC (z.B. für MS-Access)

1. Java-Anwendungen sollten auf SQL-Datenbanken nicht per ODBC, sondern möglichst nur per direktem JDBC-Type-4-Treiber zugreifen. Ein Zugriff über die ODBC-JDBC-Bridge ist wesentlich aufwändiger und langsamer.
2. Zur Einrichtung von ODBC und DSN siehe SQL-ODBC.
3. Connection (siehe unten 'Programmierbeispiele'):

```
Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
Connection cn = DriverManager.getConnection( "jdbc:odbc:" + sDsn,
dbUsr, dbPwd );
```

Anbindung an MySQL

1. Zur Installation von MySQL siehe: mysql.htm#InstallationUnterWindows.
2. MySQL-JDBC-Type-4-Treiber (z.B. '`mysql-connector-java-5.1.16-bin.jar`' aus '`mysql-connector-java-5.1.16.zip`') downloaden von: http://www.mysql.com.
3. CLASSPATH muss JDBC-Treiber beinhalten.
4. Connection (siehe unten 'Programmierbeispiele'):

```
Class.forName( "com.mysql.jdbc.Driver" );
cn = DriverManager.getConnection(
"jdbc:mysql://MyDbComputerNameOrIP:3306/myDatabaseName", dbUsr, dbPwd
);
```

Anleitung: Einfaches Programmierbeispiel zum Auslesen einer Tabelle aus einer SQL-Datenbank

Link: www.tortsen-horn.de/techdocs/*

Links:

http://www.lehrer.uni-karlsruhe.de/~za220/htm/kurse/java/Client-Server/Datenbanken.htm
http://www.highscore.de/java/aufbau/datenbankanbindung.html
https://www.java-forum.org/thema/ms-sql-datenbankzugriff-per-jdbc.26835/

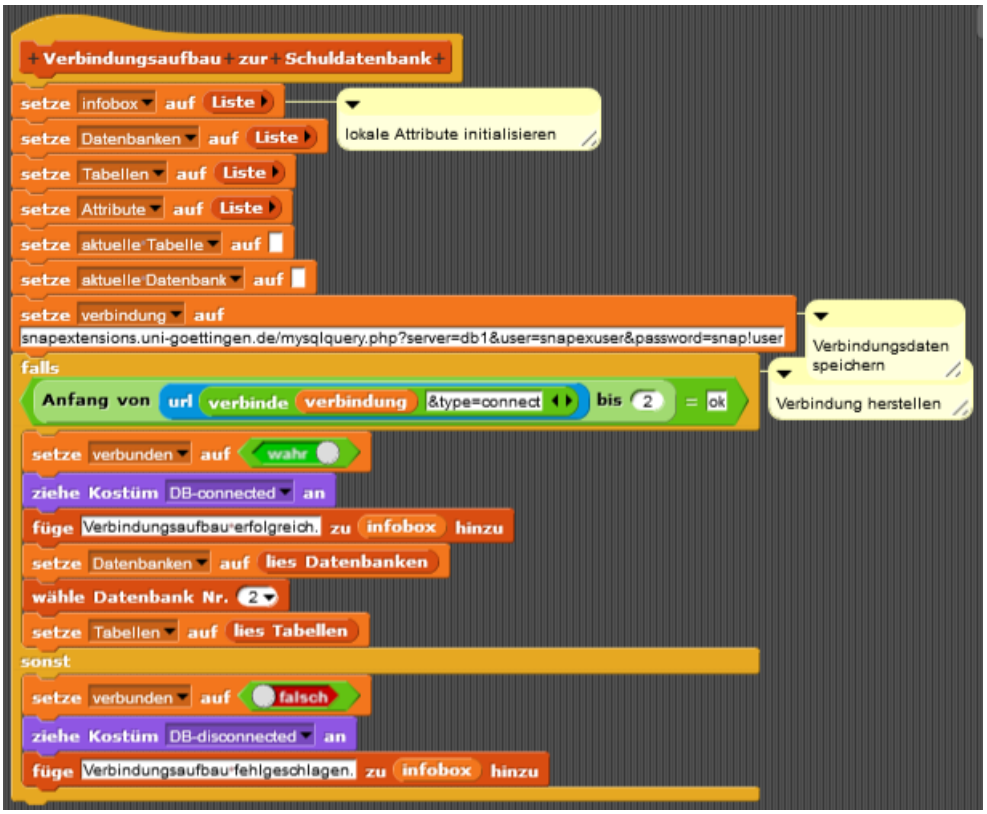
6.5. Datenbank-Programmierung mit Snap!

Auch bei Snap! wird die Programmierung von Datenbank-Zugriffen ganz klassisch realisiert. Die Kommunikation erfolgt über SQL-Statements. Diese werden im Programm zusmmengestellt und dann an den SQL-Server abgeschickt. Dieser antwortet auf das Statement. Die Antworten reichen von einfachen Bestätigungen und enden bei der Rücklieferung großer Datenmengen. Diese können dann wieder vom eigenen Programm in gewünschter Form aufbereitet werden.

notwendig sind zusätliche Blöcke
Importieren der SQL-Blöcke

zuerst Verbindungs-Aufbau zu einer Datenbank
Schon dieser – einfach scheinende – Block hat es in sich.

Genauer aufgelöst sieht er so aus.
Es handelt sich um einen speziellen Verbindungs-Aufbau zu einer online-Datenbank, die von MODROW für Schulungs-Zwecke und Projekte ins Internet gestellt wurde.
Für erste Geh-Versuche reicht uns das auch erst einmal.



Unter dem angegebenen Server (mySQL) sind mehrere Datenbanken verfügbar.
Die Auswahl erfolgt mit dem "wähle Datebbank Nr. ()".
Datenbank 2 ist eine kleine "Schuldatenbank"
Die Datenbank mit der Nummer 5 enthält Verkehrszeichen.

Leider ist der genutzte Verbindungs-Aufbau nicht ganz typisch.

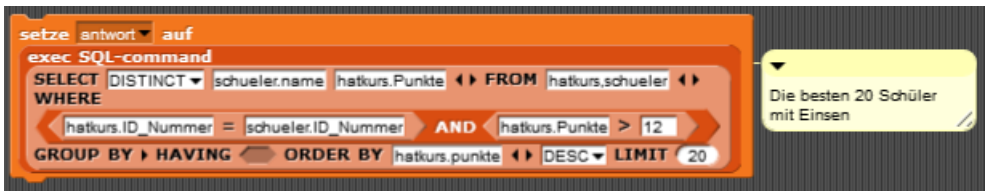
Anzeige der vorhandenen Tabellen (in der gerade verbundenen Datenbank) dann Block "Lies Tabellen"

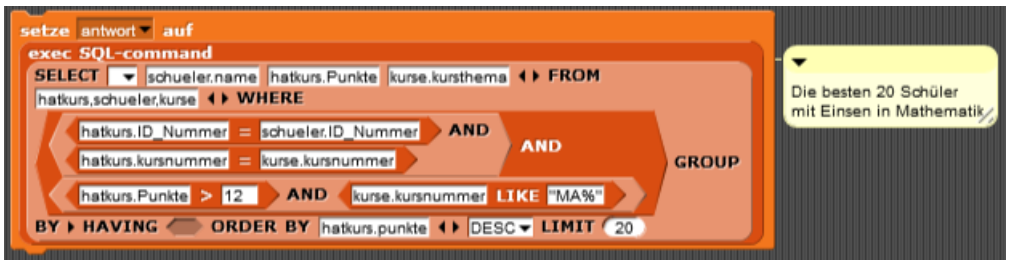
Anzeige von Attributen
Block "Lies Attribute von Tabelle …"

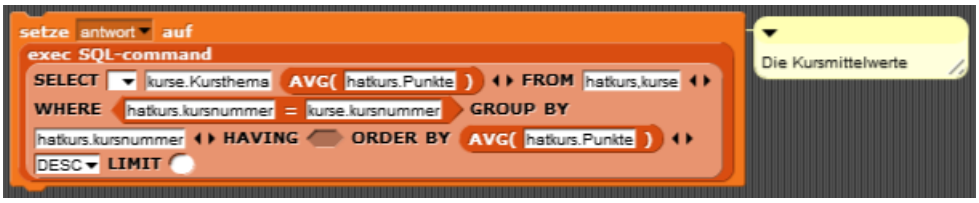
SELECT-Anweisung (einfache Variante) nur mit FROM und WHERE
muss einem SQL-Ausführ-Block übergeben werden.
Ergebnisse von SQL-Anfragen können auf eine Variable gelegt werden

komplexer SELECT-Block

ausgewählte Beispiele (aus open.SAP-Kurs "Einführung in die Informatik" mit Prof. MODROW)







Die Daten aus der Variable lassen sich mit **split** dann in verarbeitbare Datensätze (Tupel) zerlegen. Zugriff ist z.B. über Index möglich. Elemente (der Tupel) sind Komma-getrennt.

hier stößt man dann auch schnell an die Grenzen graphischer / Block-orientierter Programmierung

6.x. Anwendungs-Projekt "Super-Markt 'Kauf-ein' "

Flaschen-Kasse
scant die Flaschen oder Kästen
ermittelt Pfand-Betrag und entscheidet über weiteren Weg der Flaschen (Recycling (Einweg) oder Mehrweg)

Kassen
mit Bar-Code-Lesern zum Erfasen des Artikels und dann Berechnung des zu bezahlenden Betrages
KartenZahlung mit Kredit-Karte (Prüfung der Karten-Nummer)
Einziehen des Betrages von einem (Kunden-)Konto

Obst- und Gemüse-Abteilung
intelligente Waage (erkennt mit Kamera die Obst und Gemüse-Sorten)
zählt sie oder wiegt sie, je nach Preis-Auszeichnung

Lager
Warenwirtschaftssystem mit Produkten, Lieferanten und Kunden
Lager-Positionen für bestimmte Artikel
Nachbestellung, wenn Lager-Bestand unter das Limit gefallen ist

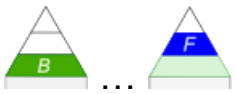
Werbe-Abteilung
die aus den Rennern und Pennern eine Mischung für die Werbe-Anzeige macht

Parkhaus
Kamera's erfassen Autonummer und errechnen aus Ankunft und Abfahrt die Park-Gebühren

Aufgaben:

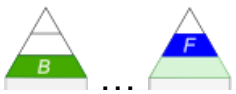
1.
2.
3.

7. Web-Datenbanken



7.0. theoretische Vorbetrachtungen

7.1. Arbeiten mit XAMPP



erste Erläuterungen und Anleitungen im Script: → 🕮 **dynamische Webseiten und Web-Datenbanken**

7.1.0. Vorbereitungen

Download von xampp.org
für einfache Projekte am Besten geeignet ist die portable ZIP-Version (unter weitere Versionen zu finden!)
kann auf einem USB-Stick etc. entpackt werden (also transportabel und ohne Admin-Rechte nutzbar)
aktualisieren und e.v. schnell mal neu installieren geht damit auch am Einfachsten
entpacken in das Wurzel-Verzeichnis eines Laufwerkes (z.B. USB-Stick)

hier auch gut in Kombination mit dem "portable Apps"-System zu nutzen
enthält praktische alle wichtigen Anwendungen, alle frei nutzbar, werden ständig gepflegt, lassen sich bei jedem Start aktualisieren (wenn notwendig und gewollt), auf jedem WINDOWS-Rechner benutzbar (außer bei gesperrtem USB)
Download über portableApps.com
zuerst das Menü-System und dann später die gewünschten Applikationen

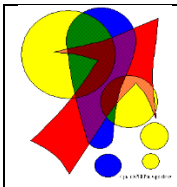
für echte Anwendungen (z.B. Intranet) sollte ev. "normale" Version genutzt werden

7.1.1. Einrichtung und Konfiguration

7.1.x.

gute Quelle; sehr viele Info's und Detail's → www.inf-schule.de

8. Datenbanken aus der Sicht von Wirtschaft und Wissenschaft



Bemerkungen zum folgenden Abschnitt:
Die Inhalte sind hier noch sehr locker und teilweise recht wirr zusammengetragen. Da dieser Bereich nicht direkt ein Kern-Thema eines Informatik-Kurses in der Sek.II betrifft, wird dieser Abschnitt auch als Anhängsel betrachtet.
Interessant könnten aber diverse Detail für Schüler sein, die sowohl Informatik als auch Wirtschaft belegen. Auch für praktische Programmier-Aufgaben gibt es viele Ansatzpunkte.
Nach und nach werden die Inhalte hier weiter ergänzt und schrittweise geordnet und betextet.

viele Informationen, Ideen und Strukturen basieren auf einem online-Kurs "Big Data Analytics" von Prof. Dr. E. MÜLLER bei openHPI (→ open.hpi.de) sowie dem OpenHPI-online-Kurs "Data Science und Data Engineering - Klarheit in den Schlagwort-Dschungel" mit Prof. F. NAUMANN (Jan./Feb. 2020)
diese Kurse können zum Selbststudium genutzt werden und sind sicher sauberer konzeptioniert als die nachfolgenden Seiten
eine echte Kurs-Teilnahme ist nicht mehr möglich, vielleicht wird der Kurs wieder aktiviert, das muss tagaktuell geprüft werden

Beispiel: Data-Warehouse

Schicht / Ebene		
externe Ebene	hohe (Zeit-, Rechen- u. Daten-aufwändige) Anforderungen durch die Nutzer	
konzeptionelle Ebene	redundanzfreie Basis-Tabellen als Dimensions-, Fakten- u. / od. Lookup-Tabellen	
interne Ebene	Rohdaten, häufig in denormalisierter (stark redundanter Form)	

in der Nacht werden die riesigen Aggregations-Tabellen / -Strukturen der externen Ebene erzeugt, diese sind dann hoch verfügbar und können extrem schnell benutzt werden
Aggregationen erzeugen Datenbestände, die bis zu sechsmal so groß sind, wie die Basis- bzw. Roh-Daten, Aggregations-Tabellen befinden sich als gespeicherte Daten (meist Tabellen) dann auch auf der internen Ebene

Definition(en): Data Warehouse
Unter Data Warehouse versteht man ein Informations-System zur Analyse von Unternehmen- und Institutions-Daten.

Data Warehouse's sind aus der immer größeren Datenmange geboren, die von Datenbanken und im täglichen Arbeiten von Informations-Systemen erzeugt und bereitgestellt werden.

bei der Nutzung (z.B. Abfrage einer Tabelle) kann es ein Problem mit der ständigen Aktualisierung von Daten-Bestände geben
Trennung der transaktionalen Datenbank (mit dem online Datenstrom) von der Auswertungs-Datenbank – eben dem Data Warehouse
i.A. ist das Data Wharehouse auch wieder eine – z.B. relationale – Datenbank
auf Daten-Analyse und –Darstellung spezialisiert (aktuelle Verkaufs-Trends, Tages-Berichte, …)
Übertragung der Daten aus der transaktionalen Datenbank in das Data Warehouse z.: Nachts, in einer Geschäfts-schwachen Zeit, zum Feierabend, …
Prozess der Übertragung nennt sich ETL (Extract Transform Load) beinhaltet eben die Vorgänge:

- Extraktion der Daten aus der transaktionalen Datenbank
- Transformation der extrahierten Daten in die Data Warehouse-Struktur
- Laden / Speichern der transformierten Daten im Data Warehouse (ev. mit Filterung und Normalisierung)

vor allem die ersten beiden Vorgänge sind sehr aufwändig
zusätzlich kann es auch noch eine weitere Extraktion der Daten für einen Daten-Markt (Data Market) geben, in dem nur die wichtigsten Daten-(Teil-)Mengen verfügbar sind, hier dann aber effektivere Zugriffe möglich sind

Vorteile des Data Warehouse-Konzepts

- keine Beeinträchtigung des laufenden Betriebes durch Daten-Analyse
- starke Spezialisierung auf Analyse und Präsentation
- es sind Daten-Analysen zu Zusammenhängen möglich, die man sich tagaktuell ausdenken kann, und die nicht von der eigentlichen Struktur der Daten in der transaktionalen Datenbank angelegt waren
- Verknüpfung mit anderen Datenbanken (z.B. Geodatenbank)
-

Nachteile des Data Warehouse-Konzepts

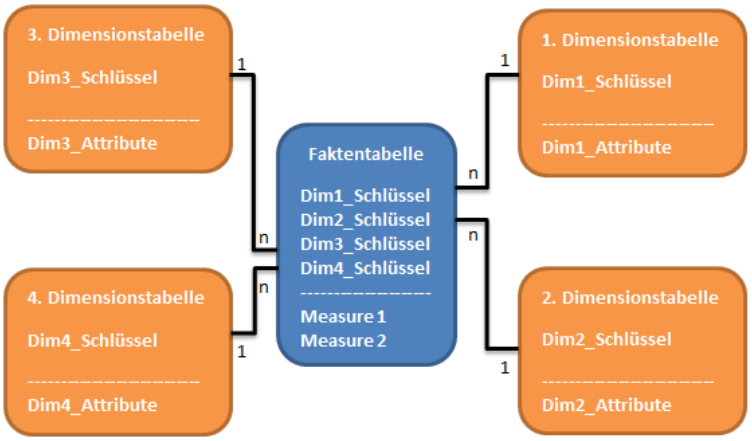
- Daten im Data Warehouse sind immer (etwas) veraltet
- Daten haben im Data Warehouse meist andere – oft eine einfachere – Struktur
- meist noch mal Vergrößerung der Datenmenge durch Umstrukturierung
-

Data Warehouse umfasst Komponenten zur:

- Aufbereitung von Daten
- Daten-Haltung
- Informations-Analyse

sowie den

- Data Warehouse- und Metadaten-Manager



Star- od. Schneeflocken-Schema
Q: www.datenbanken-verstehen.de

Definition(en): Business Intelligence
Unter Bussiness Intelligence werden Methoden und Prozesse zur Erhebung, Speicherung, Verarbeitung und systematischer Auswertung elektronischer Daten zusammengefasst.

Bereiche:
Daten-Analyse (Erfassung und Auswertung von Unternehmens- und Markt-Daten)
Advanced Analytics (frühzeitiges Erkennen von Markt-Entwicklungen, Prognosen)

Technologien der Bussiness Intelligence:

- **Data Warehouse**
- **Online Analytical Processing (OLAP)**
- **Data Mining**

Definition(en): Queries
Queries (sprich:) sind Abfrage-Mechanismen in Datenbanken.
Queries sind Anfragen an Datenbanken.

Definition(en): Schema
Ein Schema (i.S. von Datenbanken) ist eine Sammlung von Metadaten, die die Struktur von Relationen (Tabellen) und ihre Verbindungen (Verknüpfungen, Kardinalitäten) beinhalten.

Definition(en): Skalierbarkeit
Die Skalierbarkeit ist die Fähigkeit Leistungen / Funktions-Umfänge / Algorithmen / … auf kleinere oder größere bzw. größer werdende Datenbestände anzuwenden.

durch horizontale Skalierbarkeit wird erreicht, dass in Datenbanken auf JOINs (→ 4.0. Relations-Algebra / Relations-Kalkül) und Transaktionen verzichtet werden
besonders bei NoSQL-Datenbanksystem bringt das Performance-Vorteile und der organisatorische / administrative Aufwand reduziert sich

aus System- bzw. Betriebs-ökonomischer Sicht werden auch noch die folgenden Faktoren betrachtet

Definition(en): Gesamtbetriebskosten
Sind die absoluten oder relativen Aufwendungen für den Betrieb einer Datenbank in einem bestimmten Zeitraum. Als Kostenfaktoren kommen in Frage: Adminstration, Datensicherungen, Handware, Software, Updates, Energie, …

Definition(en): Leistung
Die Leistung einer Datenbank ist ein Maß für die Menge manipulierter Daten, die produzierten Ergebnisse, Arbeitsdurchläufe, … pro Zeit-Einheit.
Als Leistungs-Ausdruck kann auch der veränderte Aufwand bei gleichbleibenden Resultaten verwendet werden.

je geringer die (Betriebs-)Kosten bei gleichbleibenden Ergebnissen, umso besser / höher ist die Leistung der Datenbank

8.x. BigData, Data Science und Data Engineering

Q: basierend auf den Open-HPI-Kursen:
"Big Data Analytics" von / mit Prof. MÜLLER ()
"Data Engineering und Data Science – Klarheit in den Schlagwort-Dschungel" von / mit Prof. NAUMANN (Jan/Feb 2020)
→ Kurse stehen i.A. zum Selbststudium auf open.hpi.de zur Verfügung!

8.x.0. Historisches

Daten wurden schon immer gesammelt und in irgendeiner Form dargestellt
Geschichten von Jagden, Bau-Aufwendungen bei den Pyramiden, Rezepte für Heilmittel auf Papyrus, Arbeits-Anweisungen für die Mumifizierung, …

aus der Daten-Wissenschaft erwachsen

185? Analyse des Cholera-Ausbruches in London vom John SNOW
Technik der Aggregation (Aufsummierung und graphische Darstellung in einer Karte)
einfache Daten analysieren
eine Hypothesen gebildet

derzeit Flut an Daten; Menge so groß dass wir als Menschen sie gar nicht mehr und selbst Computer kaum alle Informationen aus ihnen herausholen können
heute sind Daten auch besonders vielgestaltig

US Library of Congress: 20 TB Text
Amazon: 42 TB
YouTube: 45 TB
National Energy Research Scientific Computing Center (NSERC): 2,8 PB
World Data Centre for Climate (WDCC): 330 TB + 6 PB auf Magnetbändern
LexisNexus / ChoicePoint: 250 TB (Personen-Daten!)

Data mining
Finden von Regeln (besonders WENN … DANN … Regeln), Zusammenhängen, Klassen (in Klassen einsortieren), Clustern (Gruppieren, Gruppen bilden), Mustern usw.

Klassiker der Regel-Zusammenhänge: die Windel-Bier-Regel
besonder zum Wochenende hin und kurz vor Geschäftsschluß werden Windeln und Bier zusammen gekauft → Produkt-Anordung im Supermarkt darauf hingehend angepasst

Daten-getriebene Anwendungen
(meist recht einfache) Anwendungen zur Analyse von (sehr) großen Daten-Mengen oder Daten-Strömen
typische Nutzungs-Bereiche derzeit:

- Produkt-Management (Begleitung von Produkten über ihres Lebenszeitraum → Erkennung von Problemen)
- Heim-Automatisierung (SmartHome → Lernen der Gewohnheiten)
- Gesundheitswesen (Epidemien, Krankheits-Verläufe, Medikamentationen, …)
- Wasser-Management ()
- Wissenschaft ()

- Markt-Forschung ()
- Informations-Marktplätze (Kauf und Verkauf von Daten)
- Verkehrs-Management (Straßen-Planung, Ampel-Schaltungen (→ grüne Welle), …)
- Energie-Management ()
- Bildungswesen ()
- Politik (→ Beeinflussung von Wahlen, Wähler-Mobilisierung, …)
- …

Maschinelles Lernen (Machine Learning)
Verallgemeinerung / Zusammenfassung / … der heute bekannte Techniken, um aus (vorhandenen) Daten zu lernen, und ev. zu neuen Daten Voraussagen zu machen
Bildung von Modellen

positive Beispiel-Nutzungen für Big Data:

- Vorhersage (Wetter, Natur-Katastrophen, Sensoren- / Maschinen-Ausfälle, Krankheiten / Epidemien, …)
- Optimierung (Verkehrs-Flüsse, Maschinen-Nutzung, Straßen-Verläufe, Positionen von Sendemasten, Logistik, …)
- Personalisierung (Medikamentation, Ausbildung, Produkt-Empfehlungen, …)
- Komfort (Heim-Automatisierung, Anmeldung am Smartphone, (teil-)autonomes Fahren, …)
- Intelligenz (automatische Übersetzung von Texten, Computer-Gegner in Spielen, Robotik, …)
-

(Big) Nudging

(z.T. unterbewußte) Beeinflussung / Anregung des Handelns von Menschen

z.B.:
noch 1 Artikel verfügbar
heute 50 % Rabatt, Mond-Preise
hohe Nachfrage
in der letzten Zeit haben 10 andere Personen auch in diesem Hotel gebucht
Personen haben auch diese Produkte gekauft
Bewertungs-Sterne, Label, …
Platzierung von Waren in Regalen
Platzierung von Essen in einer Mensa (z.B. Salate nach vorn)
Standard-Einstellungen von Programmen / Apps (Meldung des Nutzer-Verhaltens an die Hersteller)
Fake News, Nachrichten-Kanäle (eher solche, die an der eigenen Weltanschauung orientiert sind)
Werbung (ganz allgemein)

negative Beispiele für Big Data:

- Eindringen (in die Privatspäre → Auffenthaltsort über's Handy, Gesichter-Erkennung über die Video-Überwachung, Smart Home, …)

- Klassifizierung (Kaufverhalten, Social Scoring, Gewährung von Freigängen / Bewährungen im Justizwesen, …)
- Fehl-Informationen (Filter Bubble (man sieht nur eine Auswahl von Nachrichten; Fake News))
- Einschreiten (Zensur, Drohnen-Einsatz für die Tötung von Personen)
-

- Beeinflussung von Wahlen (z.B. https://www1.wdr.de/mediathek/av/video-der-fall-cambridge-analytica--102.html)
- Social Scoring (Kontrolle, Bewertung und Sanktionierung der Staatsbürger; z.B. China)
- Überwachung des Internet-Daten-Verkehrs / eMail-Verkehrs / Telefon-/Handy-Gespräche (→ https://de.wikipedia.org/wiki/PRISM ; https://www.sueddeutsche.de/digital/ueberwachung-am-de-cix-betreiber-des-frankfurter-internet-knoten-verliert-klage-gegen-den-bnd-1.3996859 ; https://www.zeit.de/digital/datenschutz/2015-09/gchq-karma-police-internet-ueberwachung ; ...)

8.x.0.1. Historie der Datenbanken

zuerst proprietäre Daten-Sammlungen
Information Management Systems (IMS) / Pointer-basierte Systeme
1960er Jahre
Zeiger-orientierter Zugriff auf Daten-Gruppen
sehr an den Betriebssystemen und vorhandenen Datei- und Speicher-System orientiert
neu verfügbare Hardware (Groß-Rechner od. Mainframe's) wurde für die Daten-Verwaltung genutzt
schwache Trennung von physischer Datenspeicherung und dem Daten-Modell
Programmierer haben die Daten so gespeichert, wie sie angefallen sind oder sie sie für besonders günstig erachtet haben
IMS von IBM wurde für die Teile-Verwaltung des Apollo-Programms entwickelt und wird heute noch (natürlich weiterentwickelt) von der NASA verwendet
kommt auch immer noch bei Banken und Versicherungen zur Anwendung

relationale Datenbanken
Speicherung in definierten und standardisierten Tabellen u.a. mit Fremdverweisen und standardisierten Datentypen
Trennung von Datenspeicherung und Daten-Modell; Nutzer hat keinen Zugriff auf die Daten-Speicherung mehr!

Edgar Frank "Ted" CODD (1923 - 2003):
prägt Begriff des Daten-Modell als mathematisch und systematisch fundiertes Prinzip der Daten-Speicherung
entwickelt genial einfaches Prinzip des relationalen Daten-Modell's

erstes relationales System war System A von IBM
mit SIQUEL/SQL als Anfragesprache
es folgte dann INGRESS von Michael STONEBREAKER (1970er)

heute XML- und JSON-Daten-Modelle (???)
immer mehr Datentypen, die verwaltet werden können

neben einer Tendenz zu immer größeren Datenbanken mit immer größeren zu verwaltenden Daten-Mengen, kommen nun auch immer kleinere Datenbanken dazu (quasi Mini-Datenbanken, Eingebettete Datenbanken) für kleine und kleinste Anwendungen, die aber trotzdem das Datenbank-Konzept als Grundlage nutzen

ab 2010 verteilte Datenbanken und Schlagwörtern wie Big Data, Big Scale, …

"in Memory"-Datenbanken
Speichern und verwalten von Daten nach dem Datenbank-Prinzip direkt im Speicher
z.B. auch zur online-Analyse von Datenströmen

8.x.1. Was sind den nun "Big Data"?

viele Daten oder große Daten-Mengen sind schon lange bekannt
z.B. Volkszählungen
astronomische Daten (Kalendarien, Mond-Phasen, Mond- und Sonnen-Finsternisse, …)

GARTNER-Report (2012) erste breit akzeptierte Definition für BigData

Charakteristika von Big Data (die großen 3 V's)

- **Volume**
  (Umfang / Menge / Daten-Volumen)
  - auch als 1. Dimension betrachtet

  hierauf bezieht sich ursprünglich das "big"
- **Velocity**
  (Geschwindigkeit)
  - auch als 2. Dimension betrachtet
- **Variety**
  (Vielfalt / Vielzahl / Verschiedenheit / Abwechslung / Unterschiedlichkeit)
  - auch als 3. Dimension betrachtet

die drei Dimensionen werden auch als 3-V-Modell betrachtet

begleitet von innovativer Informations-Verarbeitung

ermöglicht tiefere Einblicke in die Inhalte / Strukturen der Daten und verbessert die Möglichkeiten der Prozess-Automatisierung z.B. durch automatisierte Algorithmen

Big Data heute mehr aus dem Bezug auf innewohnende Eigenschaften der Datensammlung bezogen
interessante Eigenschaften, Zusammenhänge, Probleme, … in den Daten werden gesucht

Charakteristika von Big Data durch IBM (die großen 4 V's / the four V's)

- **Volume**
  (Umfang / Menge / Daten-Volumen)
- **Velocity**
  (Geschwindigkeit)
- **Variety**
  (Vielfalt / Vielzahl / Verschiedenheit / Abwechslung / Unterschiedlichkeit)
- **Veracity**
  (Wahrhaftigkeit / Vertraulichkeit / Sicherheit)

vielfach als Big Data werden solche Daten verstanden, die zu groß (zu viele) sind, zu schnell (verarbeitet) werden müssen und / oder zu kompliziert sind, um sie mit gängigen Verfahren zu verarbeiten
als gängige Verarbeitung werden hier die üblichen Datenbank-Systeme verstanden

sehr zukunfts-offene Begriffs-Bestimmung

moderne Definitionen sehen BigData breiter angelegt

Charakteristika von Big Data (die großen 7 V's)

- **Volume (Umfang / Menge / Daten-Volumen)**	- auch als 1. Dimension betrachtet hierauf bezieht sich ursprünglich das "big"
- **Velocity (Geschwindigkeit)**	- auch als 2. Dimension betrachtet
- **Variety (Vielfalt / Vielzahl / Verschiedenheit / Abwechslung)**	- auch als 3. Dimension betrachtet
- **Value (Wert)**	Was ist der Mehrwert der Daten?
- **Variability (Variabilität)**	Wie veränderlich sind die Daten (z.B. weltweit)?
- **Veracity / Validity / Volability (Zuverlässigkeit / Gültigkeit /)**	
- **Visualisation (Visulisierung)**	
-	

von einzelnen Autoren werden noch weitere V's angegeben:

- Viscosity (Viskosität (gemeint Leichtigkeit, mit der die Daten ins System gelangen))
- Venue (Wo sind die Daten?)
- Vocabularity (Vokabular (gemeint sind z.B. unterschiedliche Begrifflichkeiten))
- Virality (Viralität (gemeint ist hier die ungleichmäßig Ausbreitung unterschiedlicher Daten innerhalb des Datenbestandes / eines Netzes / …))
- vagueness (Unklarheit / Unbestimmtheit / Unschärfe / Verschwommenheit)
- ???

vielfach sind diese aber auch gut anderen / den großen 7 V's zuzuordnen
Frage der exakten Begrifflichkeiten

massives Sammeln und strukturiertes Auswerten der Daten mit hoher Geschwindigkeit

- Kauf-Vorschläge (online-Werbung, Bannerwerbung,)
- automatische Such-Begriffs-Ergänzung (Such-Vorschläge)
- Marktforschung
- Web-Statistiken / Tracking (Nutzer-Verfolgung)
- Risiko-Bewertung und Beitrags-Anpassung bei Versicherungen
- digital price discrimination (Angebote an einen potentiellen Kunden mit einem Preis, den dieser wahrscheinlich gerade noch bezahlen würde)
- Bonitäts-Prüfung (Big Data Scoring)
- Entdeckung von Unregelmäßigkeiten bei: (Fraud-Detection)
    - Finanz-Verkehr
    - Aktien-Handel
    - Daten-Verkehr

- Server-Anfragen / System-Angriffe
-
- Bezahl-Systeme für Telekommunikation
- Energie-Verbrauchs-Überwachung und –Steuerung (Smart Metering)
- Geheimdienste (Bewegungs- und Nutzungs-Profile; → gläserner Mensch)
- Personal-Beschaffung / -Einstellung
- Vorhersage von Epidemien
- Panik-Forschung
- Wetter-Vorhersage
- Erdbeben und Vulkanausbruch-Vorhersage

nach und nach treten immer mehr Probleme (für die klassische Datenverarbeitung) auf
z.B. großes **Volumen**:
wenn man einer großen Datei – z.B. mit sortierten Einträgen – irgendwo in der Mitte einen Eintrag einfügen möchte, dann müssen alle nachfolgenden Einträge auf dem Datenträger weiter nach hinten verschoben werden
beim Löschen eines Eintrags in der Mitte müssen alle nachfolgenden Einträge nach vorne verschoben werden
was bei kleinen Dateien praktisch nicht auffällt, wird bei großen Dateien zum Hemmnis
Beispiel: Walmart verarbeitet pro Stunde mehr als 1'000'000 Transaktionen

Daten passen u.U. irgendwann nicht mehr auf einen Datenträger, auf einen Rechner, auf einen Server-Verbund

klassisches Durchsuchen vom ersten bis zum passenden Datensatz dauert bei großen Datenbanken zu lange

Sortieren von großen Datenmengen wird zum fast unlösbaren Problem

z.B. zu schneller Daten-Input (**Daten-Eintritts-Geschindigkeit** / Velority)
klassisches Beispiel ist die Daten-Verarbeitung an der Börse oder in Kontrollzentren von Chemie-Anlagen, Verkehrs-Führung (z.B. Eisenbahn), Karten-Verkauf für beliebte Konzerte, Handels-Plattformen (z.B. am Black Friday)

autonomes Fahren

Bildung von Puffern / Warteschlange in den meisten Fällen nicht möglich
beim welt-umspannenden Handel gibt es praktisch keine Ruhezeiten mehr
als passender Vergleich könnte man die Unannehmlichkeiten nehmen, die z.B. beim Video-Streaming auftreten, wenn die Daten zu langsam

Netzwerk-Überwachung (z.B. zum Erkennen von Hack's) kann nicht mit Verzögerung passieren
Überwachung des Finanzwesens (Kreditkarten-Betrug, Erkennung gespeerter Karten, Geldwäsche, keine Transfers in Embargo-Staaten, …)

Video-Überwachung mit Gesichts-Erkennung nutzt nur dann etwas, wenn gesuchte Personen sofort und sicher erkannt werden

große Daten-Mengen beim Beobachten von Atombomben-Test's od. ä.
Versuche am CERN beim Zusammenschießen von Teilchen
Daten entstehen hier in extrem kurzen Zeiträumen und müssen zumindestens gespeichert oder ausgefiltert, ev. auch ausgewertet werden

einfließende Daten müssen praktisch immer sehr zeitnah verarbeitet werden

Aufgaben:

1. *Beobachten Sie mit einer geeigneten App (z.B.) die Sensor-Daten Ihres Smartphone's (z.B. Lautstärke, …)!*
2. *2.*

zu unterschiedliche Daten (**Heterogenität der Daten** / Variety)
auch (multi-)modale Daten genannt / gemeint
unterschiedliche Sprachen (z.B. engl. und die Muttersprache, Slang's)
unterschiedliche Formate für die Darstellung eines Tages-Datum's, nicht nur in verschiedenen Ländern, sondern auch in einem Land selbst

Aufgaben:

1. *Tragen Sie für sich selbst die verschieden Formate für da heutige Datum zusammen! Wer kennt die meisten?*
2. *2.*

zu ungenaue / unzuverlässige Daten (**Daten-Qualität** / Wahrhaftigkeit / Sicherheit / Fehlerhaftigkeit)

zu komplizierte Eingabe-Masken mit ähnlichen Inhalten / Eigenschaften
hängen nicht selten von Daten-Eingeber ab (Subjektivität → Augenfarbe / Haarfarbe / …)

Unvollständigkeit von Daten ganz allgemein

in unklaren Fällen werden irgendwelche "Ersatzdaten" eingetragen, die von vielen Systemen oder nach Datentransfers nicht mehr als Ersatz-Daten erkannt werden
(z.B. Postleitzahl: 99999 in Deutschland, …)
fehlende Informationen und bestimmende Eingabe-Felder (z.B. bei einer Datenbank das Feld "weiblich": Problem bei fehlerder Information → Ist derjenige nun "männlich" oder fehlt die Information nur?
Eingabe-Masken haben zu wenige Felder, es müssen aber noch andere Daten mit erfasst werden, Daten werden dann in anderen Feldern mit eingetragen, was die Daten in diesem Feld nicht mehr sicher analysierbar macht

auch die maschinelle Übertragung / Umwandlung von Daten ist fehler-anfällig (sehr häufig Programmierfehler beim Umgang mit den Grenzen)

fehlerhafte Sensoren (verschmutzte Oberflächen, abweichende Arbeits-Temperaturen, …)

Aktualität der Daten (wenn z.B. die aktuellen Daten bei einer Klima-Auswertung fehlen, dan kann ein falscher Trend oder eine falsche Voraussage für das Sommerwetter od. ä. abgeleitet werden

nicht zuletzt geschwärzte Texte aus Archiven (Text-Analyse nur begrenzt sinnvoll, weil man nicht weiss, was für Wörter oder die Art von Wörtern unkenntlich gemacht worden)

Herausforderungen

- **Verarbeitung vieler Datensätze**

- **Verknüpfung vieler Datenfelder**
- **schneller Import großer Datenmengen**
- **schnelle (Echzeit-)Verarbeitung der Daten**
- **kurze Latenz- / Antwort-Zeiten**
- **Skalierbarkeit von kleinen bis großen Mengen / Problemen**
- **parallele Abarbeitung vieler gleicher, ähnlicher oder andersartiger Abfragen / Anforderungen**
- **sehr variable / unterschiedliche Daten-Typen auswerten**

→ Herausforderungen führen zu NoSQL-Datenbanken, da SQL solche Anfordeungen nur mit sehr großem technischen Aufwand erfüllen könnte

- Apache Hadoop Framework
- MongoDB
- Aster Data
- Greenplum
-

arbeiten nach MapReduce-Ansatz

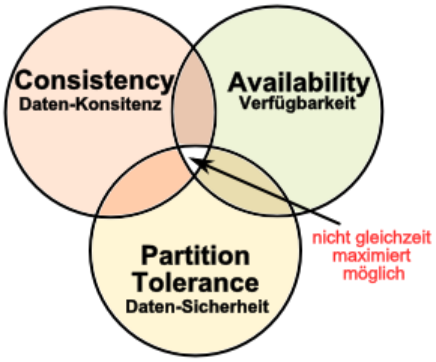
heute geschätzt, für die heute weltweit gesammelten Daten gilt

- 23 % der Daten sind wirklich nutzbar (97 % der daten aber nicht nützlich, da sie nicht indiziert sind)
- 0,5 % der Daten werden tatsächlich genutzt
- 35 % sind schützenswert
- 20 % werden wirklich geschützt (mit Datensicherungen und Datenschutz-Maßnahmen)

Die Zielrichtungen und das Selbstverständnis von Big Data und der Datenschutz sind potentielle und reale Widersprüche.

im Datenschutz gibt es kein "belangloses Datum"
an dem z.B. alte Daten gelöscht werden müssten oder vollkommen anonymisiert
selbst klassisch anonymisierte Daten lassen sich (mit Big Data-Methoden) ent-anonymisieren

CAP-Theorem
Leit- und Lehrsatz, der besagt, dass es nicht gleichzeitig möglich ist, alle drei Eigenschaften (Consistency (Daten-Konsistenz), Availability (Verfügbarkeit) und Partition Tolerance (Datensicherheit)) zu maximieren.



Daten-Quellen

mögliche Daten-Quellen

- **offene Daten**

  strukturierte Daten
  frei zugängliche Daten
  z.B. aus dem Internet
  - **Linked Open Data** (bewußt freigegebene Daten von Organisationen, Institutionen, Regierungen, …
    -
  - konkret: **Goverment Data**
    - statistische Daten der Länder
    - Protokolle von Regierungs-Sitzungen, …
    - geöffnete Archive
    - …
  - …

  z.B. Hidden Web
  - auf Webseiten abgelegte Daten-Bestände (z.B. als Dateien)
  -

  z.B. Wissenschaftsdaten
  - Klima- und Wetter-Daten
  - Sternen-Kartierung
  - Teilchen-Beschleuniger (CERN)
  - Protein-Strukturen
  - DNA-Sequenzen
  - Chemikalien-Informationen / Gefahrgut-Daten
  -

  unstrukturierte Daten
  z.B. Textdaten
  - Webseiten
  - veröffentlichte Dokumente (z.B. PDF-Dateien)
  - Zeitungs-Artikel
  - social media (tweet's, Like's, …)
  - Rechnungen
  - Zeitschriften
  - Patente
  - eMails
  - Produkt-, Hotel- usw. -Bewertungen

- **interne Daten**

  z.B. Geschäftsdaten
  - master data management
  - Bilanzen
  -

  z.B. Transaktions-Daten
  - Einkäufe in Webshop's
  - Überweisungen von Geldbeträgen
  - Währungs-Handel
  - Börsenhandel
  -

  z.B. Protokoll-Daten
  - Login's

- Verbindungs-Aufbau, -Dauer, -…
- Nutzer-Verhalten auf Webseiten
-

z.B. Sensor-Daten
- Strom-Verbrauch über moderne Stromzähler (Smarte Zähler)
- Smarthome
- autonomes Fahren
-

-

Anforderungen an Linked Open Data
- Datenbestand soll verlinkt sein (mit eindeuter ID → URI ID)
- URI ID möglichst im html-Format
- Hintergrund-Informationen
- möglichst weitere Links einarbeiten (→ Erzeugen eines semantischen Daten-Netzwerks)
- möglichst für Menschen lesbar, aber immer für Maschinen!

auch Tabellen auf Webseiten lassen sich selektieren (?erqualen / Quals)

Daten-Formate

- **XML**
- **JSON**
- **RDF**
- **SQL**
-

Beispiele für Daten-Quellen

allgemeine / diverse Daten

- **DBpedia** wikipedia-Daten
-

https://www.kaggle.com/datasets

-

https://opendatainception.io/

-

https://cloud.google.com/public-datasets/

-

https://github.com/awesomedata/awesome-public-datasets

-

https://registry.opendata.aws/

-
-

Wissenschaftsdaten / Science

Data

- Sloan Digital Sky Survey
  https://www.sdss.org/
- Large Synoptic Survey Telescope
  https://www.lsst.org/scientists/understanding-simulations-and-data
- CERN
  http://opendata.cern.ch/
- Menschliches Genom
  https://www.ncbi.nlm.nih.gov/genome/guide/human/
- Genome weiterer Organismen
  https://www.ncbi.nlm.nih.gov/guide/howto/dwn-genome/
- Protein-Strukturen
-
-

Regierungsdaten / Goverment Data

- **Europäische Union**
  https://data.europa.eu/euodp/en/data/
- **Weltbank**
  https://data.worldbank.org/
-
  https://www.who.int/gho/database/en/
- **(USA)**
  https://catalog.data.gov/dataset
- FBI-Welt-Fakten-Buch
- **Weltgesundheitsorganisation**
  https://www.who.int/gho/database/en/
- **NASA**
  https://data.giss.nasa.gov/
- **Statistisches Bundesamt (BRD)**
  https://www.destatis.de/DE/Service/Datenbanken/_inhalt.html
-
-

Definition(en): Big Data

Unter Big Data versteht man die Daten-Mengen / -Bestände (Massendaten), die durch auffallende Größe, hohe Komplezität, Schnellebigkeit und eine schwache Struktur gekennzeichnet sind.

interessante Links:
https://lod-cloud.net/ (Animation zur Entwicklung des "Linked Open Data"-Netzwerkes

8.x.2. Was genau ist "Data Engineering"?

Definition(en): Data Engineering

8.x.3. Was genau ist nun "Data Science"?

Verbindung von Domänen-Wissen (Wissen eines Anwendungs-Gebiet's), Daten-Analyse und Daten-Management

klassischer Wissenschafts-Ansatz "Voraussagen – Modellieren – Testen" (hypothesize, model, test) wird teilweise in Frage gestellt

nach gibt es ein viertes Paradigma der Wissenschaft
Paradigma = Denkweise
heute wird darunter in der Wissenschaft eine allgemein anerkannte, zusammengehörende aufeinander abgestimmte Sammlung von Regeln, Gesetzen, Methoden, … verstanden, die historisch einen längerfristigen Bestand hat
notwendige / überfällige Paradigmen-Wechsel befördern die weitere Entwicklung der Wissenschaft

anders kann man die Paradigmen einer Wissenschaft auch als ihre Entwicklungs-Phasen verstehen

Paradigmen / Entwicklungs-Phasen von Wissenschaften

- **empirisch experimentell** — emperical science
  Sammeln und Notieren / Beschreiben von Phänomenen
  Erfassen, Klassifizieren, Systematisieren von Daten
  Durchführen von (ungerichteten) Experimenten

- **theoretisch modellierend** — theoretical science
  Ableiten von Regeln und Gesetzen
  Bildung von Theorien, Modellen
  experimentelle Prüfung von Theorien / Voraussagen

- **berechnend simulierend** — computational science
  Erstellen und Nutzung von Simulationen

- **Daten-intensiv KI-basiert** — Nutzung von großen Mengen an (auch älteren) empirischen Daten, um neue und mehr Erkenntnisse aus ihnen abzuleiten
  (extrem bis sehr) komplexe Modelle der Realität (, die u.U. auch einzelne Fachbereiche / (Teil-)Wissenschaften überwuchern)
  quasi ein maschinelles Nachvollziehen der Wissenschafts-Entwicklung für ein Problem / eine Teilwissenschaft / …

mir erscheint das 4. Paradigma etwas stark Hype-geprägt bezüglich der derzeitigen Entwicklung von Bid Data usw. zu sein
wenn, dann befinden wir uns auch erst in der Anfangsphase, die auch ersteinmal objektiv reflektiert / begleitet werden muss
Guru's des vierten Paradigma's sehen schon das Ende der klassischen Wissenschaften
sie behaupten teilweise, dass man keine (anderen / klassischen) Modelle / Theorien mehr braucht, weil die Daten selbst die Welt / Realität / das Problem ausreichend beschreiben (Korrelation ist genug)
dabei wird aber vergessen oder unterschlagen, dass eine Daten-Analyse / Auswertung unbedingt Theorien / Modelle / … braucht, um den Daten einen Informations-Gehalt zu geben
Korrelation impliziert noch keine Kausalität

es besteht auch eine große Gefahr mit der Digitalisierung eine Wissenschaft nur aus der Sicht der digitalisierten bzw. digital erfassten Daten zu sehen
die traditionellen / klassischen / analogen Daten werden einfach ignoriert
einfach deshalb, weil sie für die digitalen Native's nicht leicht erfassbar ist
Gefahr des Ableiten falscher Schlüsse (da nur eine kleine / eingeschränkte Daten-Breite genutzt wird)

Beispiele für Fake-Korrelationen:

- Anzahl nicht-kommerzieller Raketen-Start's und der Anzahl von Promotionen in der Sozialogie
- Anzahl der Personen, die in Schwimmingpool's ertranken und der Anzahl von Filmen die Nicolas CAGE produziert hat

Definition(en): Data Science

Künstliche Intelligenz

AI … Artificial Intelligence

offiziell Teil der Robotik
mittlerweise getrieben durch die Problemkreise "maschinelles Lernen" (Machine Learning) und "Deep Learning" mehr losgelöst von der Verkopplung mit mechanischen Aktoren od.ä.
derzeit Sammel-Wissenschaft, die Erkenntnisse der Neurowissenschaften, Mathematik, Informatik (→ Theoretische Infromatik), Logik, Psychologie, Kommunikations-Wissenschaften, Philosophie, Linguistik, … nutzt und teilweise in neue (Wissenschafts-übergreifende) Zusammenhänge bringt
beschäftigt sich mit Wissens-basierten Systemen (Experten-Systeme), Muster-Analyse und –Erkennung, Muster-Vorhersage, Robotik, Künstliches Leben, Modellierung anhand künstlicher Entropie-Kraft, …

problematisch ist, dass es an einem übergreifenden Begriff oder einer Definition von Intelligenz mangelt
oft sehr frei interpretiert

starke KI sind Systeme, die auf dem gleichen Niveau, wie Menschen arbeiten
zur schwachen KI zählt man solche Anwendungen, die konkrete Einzel-Probleme lösen

typische Anwendungen:

- Suchmaschinen
- Gesichtserkennung
- Data-Mining
- Schrifterkennung
- Bilderkennung
- Texterkennung
- autonome Waffen
- Bots
- maschinelle Übersetzung
- humanoide Roboter
- Sprach-Assistenten
- Spracherkennung
- persönliche Assistenten
- autonome Fahrzeuge
- Computer-Vision-Systeme (Großraum-Video-Überwachung)
- Gruppen- und Verhaltenssimulationen
- Computer-Algebra-Systeme
- semantische Suchmaschinen
- Informationsrückgewinnung
- Biometrie
- wissensbasierte System
- Avatare / Gegenspieler

eigenständiges Lernen
Reagieren auf neue Situationen auf der Basis von bekannten Regeln

TURING-Test (1950)
ein Mensch (Probant / zutestendes System) kommuniziert mit einem anderen System (Maschine oder Mensch) und dieser/s kann nicht unterschieden mit welcher Art von Gegenüber er kommuniziert

wenn der TURING-Test bestanden wird, dann wird dem System eine (äquivalente) Intelligenz zugesprochen

Stufen der KI

- **schwache KI** — Lösung von konkreten Anwendungs-Problemen
  z.B.: Sprach-Assistenten (Siri, …), Bilderkennung
- **starke KI**
  allgemeine KI — Lösen mehrerer (komplexerer) Probleme auf Augenhöhe mit dem Menschen
  Anforderungen:
  - Ziehen logischer Schlüsse
  - Allgemeinwissen
  - Planungsfähigkeiten / Zielorientierung
  - Lernfähigkeit
  - Sprach-Verständnis
  - ähnliche Sensor-basierte Umwelterfassung
  - ähnliche Interaktion mit der Umwelt
  -
- **Super-KI** — KI ist intelligenter, als ein Mensch
  ev. selbstreproduzierend
  Problem der Singularität
  KI begreift ev. Mensch als Störfaktor, sind schneller im Fällen von Entscheidungen, nicht Emotions-getrieben

Singularität in der System-Theorie der Punkt, in dem eine kleine Veranderung / Ursache eine große Wirkung hat
hier gemeint der Punkt, an dem die / eine KI die Menschheit übertrumpft und damit die Zukunft der Menschheit danach unbestimmt / ungewiss ist
vielfach auch der Punkt verstanden, an dem eine KI so etwas wie ein Bewußtsein erlangt
(sie könnte aber auch clever genug sein, dieses Ereignis zu verstecken)
erste Konzepte der technologischen Singularität gehen auf Stanislaw ULAM (1965) zurück
nach einem bekannten Computer-Wissenschaftler und Zukunftsforscher Ray KURZWEIL ist mit der Singularität um 2045 zu rechnen

Daten-Kompetenz

Data Literacy

vielfach auch mit dem Codieren und Decodieren von Daten in Zusammenhang gebracht

das Codieren umfasst dabei das Sammeln, Aufbereiten und Analysieren der Daten
das Nutzen dieser Daten wird dem Decodieren zugeordnet

Teilziele:

- Daten-Kultur erreichen (in der Gesellschaft)
- Identifizieren von Daten-(getriebenen)Anwendungen
- Koordinieren von Daten-Anwendungen
- Ermöglöichen einer Daten-Bereitstellung / Daten-Modellierung
- Beachten des Datenschutz
- Reinigung der Daten
- Daten-Integration
- Klären Daten-Herkunft
- Analysieren von Daten
- Verbalisieren / Beschreiben von Daten
- Visualisieren / Präsentieren
- Interpretieren / Erklären von Daten
- Ableiten von Handlungen / Reaktionen
-

Definition(en): Data Literacy / Daten-Kompetenz
Daten-Kompetenz umfasst die Fähigkeiten Daten kritisch zu erfassen, zu verwalten (managen), zu bewerten und anzuwenden auf konkrete und / oder allgemeine Anwendungsfälle.
Unter der Daten-Kompeten werden solche Fertigkeiten und Fähigkeiten, Daten Sinn-entsprechend und Ziel-orientiert zu codieren und zu decodieren.

ethische Aspekte der Daten-Nutzung

in der Informatik sind die Daten sehr dicht an der Realität; weiter entfernt sind z.B. Hardware-Entwickler; erfordert erhöhte Sorgfalt

mit großer Macht-Fülle (Daten-Menge) ist auch immer eine große Verantwortung verbunden
Daten und Daten-Anwendungen sollen mehrfach genutzt werden (dual use)

vermeiden von fragwürdigem oder Sinn-entstellende / manipulierendem Umgang bzw. eine entsprechende Nutzung von Daten

Orientierung darauf, eine negative / fragwürdige Nutzung der Daten zu verhindern

eine gut entwickelte Daten-Kompetenz bewirkt eine Stärkung der positiven Nutzungen von Daten

Einhaltung des Datenschutzes (Gesetzes-konform, vorausschauend, verantwortungsbewußt)

Privatspäre

Wer hat u.U. Zugriff auf meine Daten?

- Sie selbst
- Familien-Mitglieder
- Ihre aktuellen und abgelegten Freunde (→ ev. jetzt Feinde)
- Ihr Internet-Service-Provider
- eigene oder fremde Regierungen / Geheimdienste
- ev. Jeder
    - z.B. bei schlechten Datenschutz-Einstellungen
    - Mitschnitt und Verkauf von Daten
    - durch Hacks und Datenlecks
- Archive speichern ev. über Jahre / Jahrzehnte hinweg
- …

Schutz-Möglichkeiten

- Daten(-Erhebung), … vermeiden
- strenge Gesetzgebung
- Daten-Kompetenz
- statistisches Grundverständnis
-

Themenfeld: Medizin-Daten

(erwartete) Vorteile durch Big Data

- **personalsierte Medizin**
- **Arzneimittel-Planung**
- **Risiko-Sensoren**
- **Forschung**
- **Genom-Analyse**
-

(erwartete) Nachteile durch Big Data

- **Anonymität**
- **"Big Brother"-Effekt**
  Überwachungs-Effekt
- **Korrelation statt / vor Kausalität**
- **Entscheidungs-Kompetenz bei Maschine**
- **elektronischer Code / Algorithmen statt humanistischer Werte**
-

"Nationale Kohorte"

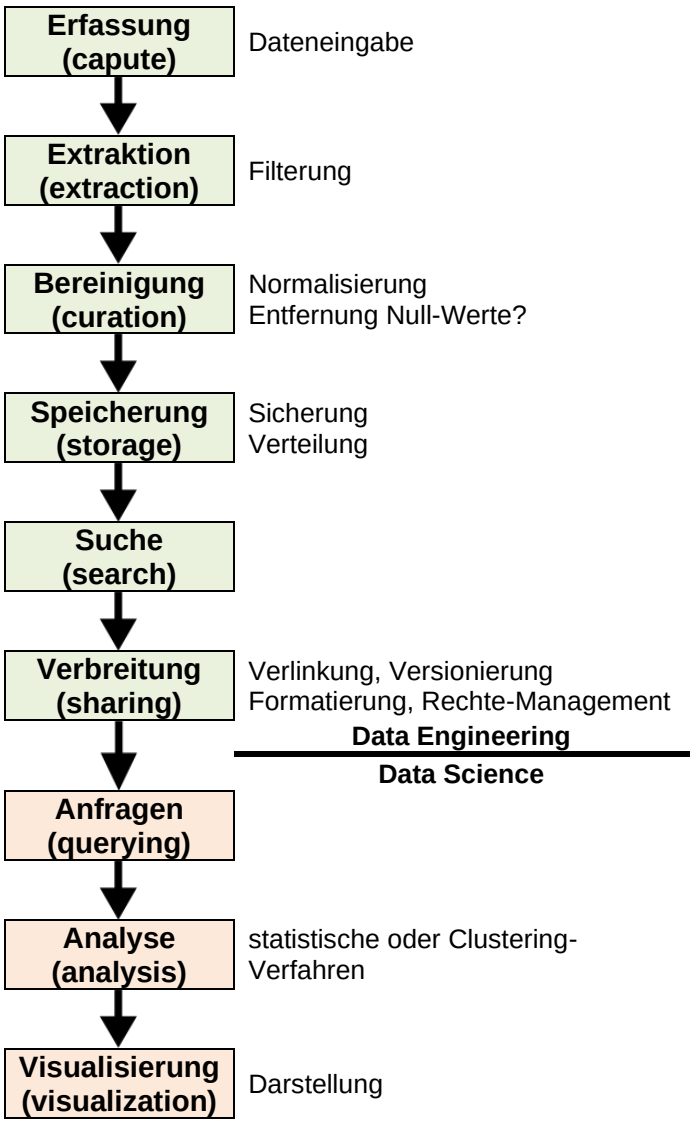
google Flu Trends
Versuch epidermiologische Tendenzen aus Such-Anfragen zu generieren
Projekt bei goggle aufgegeben
von nationalem Interesse (z.B. Planung Impfmittel)
wird derzeit akademisch weiter beforscht (unter Einbeziehung (anderer) sozialer Medien)

8.x.4. Data Science

"Data Science"-Pipline

Data Engineering grünlich hinterlegt

reien Data Science rötlich hinterlegt



Empfehlungs-Systeme / kollaboratives Filtern

Recommander

kollaborative filtering

praktisch umgesetzt in Musik-, Buch-/Lese- oder Kauf-Empfehlungen, Kontakt-Vermittlungen z.B. auch im Ranking von Webseiten (in der Suchmaschine google)

"andere Kunden haben auch dieses Produkt gekauft"

Problem bei neuen Kunden / Produkten (cold start problem)
→Anlage von Profilen / Default-Einstellungen / zufällige Werte

Problem langweilige / doppelte Angebote (nach Kauf eines Fernsehers bekomme ich weitere Empfehlungen für gleichartige Fernseher)
→ Erzeugen "zufälliger Entdeckungen" (Serendipität)

Empfehlungs-Systeme können gut validiert werden (weil z.B. die tatsächlichen (zusätzlichen) Käufe beobachtet werden) und dann wiederum verbessert zu werden

8.x.3. Data Mining

Wissens-Extraktion / Daten-Schürfen
Extraktion interessanter Daten (nicht-trivial, abgeleitet, vorher unbekannt)

Data Mining-Kategorien

- **Prognosen** Finden von Trend's im Datenzeit-Bezug
- **Assoziation** Suchen nach Abhängigkeiten zwischen Daten-Objekten
- **Segmentierung** Schaffen einheitlicher, homogener (zusammenhängender) Objekt-Teilmengen
- **Klassifikation** Aufteilen der Daten-Objekt in vordefinierte Klassen

geht über SQL und normale Daten-Auswertung nach menschlicher Vorgabe hinaus
Computer übernimmt die Analyse und auch die Auswahl der Methoden
Bildung von Brücken zwischen erhobenen Daten und der Entscheidungs-Ebene (Mensch)

klassische Methodik:
- Beobachtung machen (Observation)
- Modelle bilden (Modelling)
- Vorhersagen ableiten (Prediction)
- Entscheidungen fällen (Decision Making)

moderner Grob-Ablauf (BigData Systems)
1. Daten-Integration
2. DataMining
3. Daten-Visualisierung

Wissens-Entdeckung in Daten

z.B. KDD-Prozess (Knowledge Discovery in Databases)
- Daten-Integration und –Bereinigung → DataWarehouse
- durch Transformation, Selektion und Projektion extrahieren der relevanten Daten (Data Mining) → Muster
- Wissen entsteht erst Kopf des Betrachter / Nutzers
- ev. interativer Rückgriff auf die ersten Schritte des KDD-Prozesses
- ev. Nutzung von Visualisierungsmethoden auf die Rohdaten und frühe (abgeleitete) Daten-Strukturen

Daten-Bereinigung und –Integration beansprucht 60 – 90 % des Analyse-Aufwands

Projektion → Auswahl der Spalten
Selektion → Auswahl der Zeilen
Transformation → Ändern des Daten-Typs / Normierung / Diskretisierung / Aggregat-Bildung / Differenzen-Bildung / …

Methoden:
Cluster-Bildung (Clustering, Gruppen-Bildung; Segmente finden;)
Klassifikation (Classification)
Häufig gemeinsam auftretende Beziehungen (Frequent Itemset Mining)

typische OLAP-Operatoren
Daten-Auswahl (z.B.: SELECT * FROM fakten;)

- Roll up (drill-up) → Aggregation (Daten zusammenfassen)
  - z.B.: SELECT … FROM resultate GROUP BY kriterium;
- Drill down (roll down) → Feingranulierung
  - z.B.: … GROUP BY unterkriterium, zusatzkriterium;
- Slice and dice (Scheiben- / Schichten-/ Kategorie-Auswahl)
  - z.B.: SELECT … FROM … WHERE kriterium;
- Pivot (rotate) → Rotation / Tausch von Spalten und Zeilen
- …

Bewertung der Daten:

- SelfExp: Maß für die Abweichung vom Erwartungswert eines Datums
  - Berechnung als Quotient aus der Differenz zwischen Datum und dem Erwartungswert sowie der (Gesamt-)Standardabweichung
- InExp: Maß für die Lage des Datum unterhalb dem Erwartungswert / einer anderen Zelle
  - Berechnung: ist Maximum der SelfExp für die nächste Feingranulierung
- PathExp: Maß für die Abweichung vom Erwartungswert hinsichtlich einer bestimmten Feingranulierung
  - Berechnung: ist Maximum der SelfExp aller Zellen bei einer speziellen Feingranulierung

statistische Methoden / Verfahren für BigData

Was gibt es überhaupt für Daten?

Werte / Daten			
kategorisch (qualitativ)		**numerisch (quantitativ)**	
nominal / ungeordnet	**ordinal / geordnet**	**diskret**	**kontinuierlich / analog**
ohne innere Ordnung	**innere Ordnung / Ränge / Reihenfolge**	**abgezählt**	**unzählbar**
Farben: gelb, rot, blau, …	Kleider-Größen: XS < S < M < L < XL < XXL	Ganze Zahlen in einem Intervall	Zeit / Zeitreihen
Namen: Klaus, Monika, Bert, …	Grade / Noten: 1 > 2 > 3 > 4 > 5 > 6 (…)	Artikelnummer	Blutdruck Außentemperatur

Dimensionen von Daten:

eindimensional	**zweidimensional**
einfache Zahlenreihen	Punkte im Koodinatensystem
dreidimensional	**multidimensional**
Raumdaten	
hochdimensional	**ohne Dimension**
Zeit-Reihen von Meßdaten	Protein-Faltungs-Strukturen

Wie kann man welche Daten statistisch verarbeiten?

diskretive Statistik:
Mittelwert(e), Median, Maximum, Minimum, Spannweite, Abweichung, Quantile, Ausreißer, Mode (häufigster Wert), …

Gesetzmäßigkeiten / Verarbeitungsregeln / Berechnungsregeln:

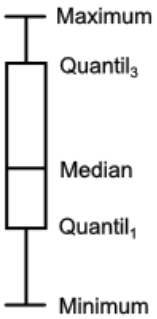
Distributive Maße:
Anzahl(N_1 **U** N_2) = Anzahl(N_1) + Anzahl(N_2)
Summe(N_1 **U** N_2) = Summe(N_1) + Summe(N_2)
Produkt(N_1 **U** N_2) = Produkt(N_1) * Produkt(N_2)
Maximum(N_1 **U** N_2) = Maximum(Minimum(N_1) , Maximum(N_2)
Minimum(N_1 **U** N_2) = Minimum(Minimum(N_1) , Minimum(N_2)

Algebraische Maße (nicht-distributiv!)
Durchschnitt(N) = Summe(N) / Anzahl(N)
Standardabweichung( )
Durchschnitt(N_1 **U** N_2) = Summe(N_1 **U** N_2) / Anzah(N_1 **U** N_2)
Varianz( )

Holistische Maße (nicht distributiv berechenbar)
Median( )
Modalwert() / HäufigsterWert( )
Rang( )

weitere Maße
InterQuantilRang( ) / IQR( ) = $Q_3 – Q_1$
Standardabweichung( )
Ausreißer()

Visualisierung von statistischen Maßen mittels BoxPlot
Darstellung von 5 charakterisierenden Daten / Maßen in einem intuitiv verständlichen System
z.B.: Maximum, 3. Quantil, Median, 1. Quantil, Minimum oder 0%, 25%, 50%, 75% und 100% (entspricht 25 Perzentilen)
weiterhin Visualisierung von Ausreißern, Interquantilsabständen und Streuungen möglich



Visualisierung in Scatterplot Matrizen
bei mehrdimensionalen Daten werden für alle Paare von Dimensionen zweidimensionale Diagramme erstellt → gut für Mustersuche / Beziehungen zwischen den Daten, die über einfache Korrelationen hinausgehen

Parallele Koordinaten-Systeme
ähnlich Netz-Diagrammen
mehrere y-Achsen werden parallel angeordnet und die einzelnen Objekte als verbundene Linien angezeigt → gut für Muster-Erkennung / Gruppierung

Wahrscheinlichkeitsraum ist ein Tripel (Ω, F, P) wobei Ω die Menge der möglichen Erreignisse, F die Potenzmenge der Ereignisse (Ereignisraum) und P die Funktion zur Berechnung des Auftretens einer Wahrscheinlichkeit.

Korrelationen

Spaß-Korrelationen
http://tylervigen.com/spurious-correlations

Daten clustern

K-Means Clustering-Algorithmus

1. Festlegung der (gewünschten / erwarteten) Clusteranzahl n
2. zufällige Festlegung von n Cluster-Zentroiden
3. Wiederholung:
   - Zuordnung der Elemente zu den Zentroiden (geringster Abstand)
   - Berechnung eines neuen Zentroiden aus den gesammten Elementen
4. solange, bis sich Zentroide nicht mehr verändern

Bewertung der Cluster-Bildung mittels Silhouetten-Koeffizienten

- Berechnung der durchschnittlichen Distanz zum Zentroide → a(o)
- Berechnung der durchschnittlichen Distanz zu einem alternativen Cluster / Zentroiden (Unähnlichkeit zu anderem Cluster / Zentroiden) → b(o)
- Silhouette ist 0, wenn
  sonst Quotient aus der Differenz b(o)-a(o) und dem Maximum(a(o),b(o))
- Silhouetten-Koeffizient: Summe(aller Silhouetten) und nachfolge Normierung
- +1 sehr gute Clusterung; -1 sehr schlechte Clustering

statt einem Zentroiden kann auch ein anderer Cluster-Repräsentant verwendet werden (z.B. Medoid)

Dichte-basiertes Clustern

DBSCAN (Density Based Spatial Clustering of Applications with Noise)
braucht keine Vorgaben hinsichtlich der erwarteten Cluster-Zahlen
Ausreißer werden aussortiert und nicht mit in die Cluster einbezogen
Cluster können auch extravagante Formen haben
funktioniert nicht bei hierarchischen / verschachtelte Cluster

DBSCAN – ein Dichte-basierter Cluster-Algorithmus

1. Festlegen des Parameter ε (Radius der ε-Umgebung / max. Abstand)

2. Festlegen des Parameters MinPts (Minimalanzahl Nachbarn in der e-Umgebung um in die wirklich zu clusternden Punkte (Kern-Punkte) zu gelangen
3. Für jeden Punkt:
    - ermittle die Punkte, die innerhalb eines bestimmten Abstandes (ε-Umgebung) liegen → Dichte-Wert
    - Auswählen als Kern-Punkt (core objects), wenn der Punkt mindestens so viele Nachbarn in der ε-Umgebung hat, wie bei MinPts vorgegeben wurde

für jedes Objekt o in der Ausgangsmenge D
    wenn o noch nicht klassifiziert wurde dann
        wenn o ein Kern-Objekt ist dann
            fasse alle Dichte-erreichbaren für o in einem neuen Cluster zusammen
        sonst
            füge o zu den Ausreißer-Objekten hinzu

Um die Parameter ε und MinPts festzulegen gibt es folgende Heuristik:
ε = Knick in der Kurve des Abstand des 4- oder 3-nächsten Nachbarn gegen die (geordneten) Objekte (x-Achse)
MinPts = 2 * d -1 (d … Dimension der Daten)

hierarchische Clusterung

agglomeratives Vorgehen

1. Initialisierung, indem alle Objekt für sich einen Cluster bilden
2. Paarweise Suche der beiden nächstgelegenen Objekte bzw. Cluster und zusammenfassen zu einem Cluster (eine Ebene drüber)
3. entfernen der verschmolzenen Objekte / Cluster aus der Objekt-Menge
4. weiter bei 2 bis alle Objekte / Cluster nur noch ein Cluster bilden (oberste Ebene)

Bestimmung des Abstandes zwischen Cluster / Objekte und Clustern über:
Single-Link (kleinster/minimaler Abstand zwischen den Objekten der einen Gruppe zu den der anderen Gruppe)
Complete-Link (maximaler/größter Abstand zwischen den Objekten der einen Gruppe zu den der anderen Gruppe)
Average-Link (durchschnittlicher Abstand aller Paare aus beiden Gruppen)

diversives Vorgehen

Dichte-basiertes hierarchisches Clustern

adaptives ε

OPTICS (Ordering Points To Identify the Clusteing Structure)
Suche nach dem am nächsten liegenden Punkt bei möglichst kleinem ε
Cluster unterscheiden sich dann durch Wechsel von kleinen ε zu großen und dann wieder zu kleinen ε

Kern-Distanz (core-distance) sucht das e, dass den kleisten Wert darstellt für den vorgegebenen Parameter MinPts

```
für alle Objekte o aus der Datenbank D
        festlegen o.bearbeitet = falsch
für alle Objekte o aus der Datenbank
        wenn o.bearbeitet = falsch ist dann
                füge Objekt o in die Kontroll-Liste ein
        solange Kontroll-Liste nicht leer ist tue
                wähle das erste Element (o, r-dist) aus der Kontroll-Liste
                retrieve Nε(o) und bestimme c_distance = core_distance(o)
                setze o.bearbeitet auf wahr
                schreibe o, r_distance, c_distance in eine Datei
                wenn o ein Kern-Objekt ist mit einer Distanz <= e dann
                        für alle Kern-Objekte {p ∈ Nε(o)}, die noch nicht bearbeitet wurden
                                bestimme r_distanz = reachability_distance(Kern-Objekt,
                                                        Objekt)
                                ist Kern-Objekt noch nicht in der Kontroll-Liste {(p,_) ∉
                                                        ContolList} dann
                                        füge Kern-Objekt {(p, r_distance)} zur Kontroll-Liste
                                                        hinzu
                                sonst wenn {(p,old_r_distance)} Element der Kontroll-List und
                                                        (r_distance < old_r_distance)
                                        aktualisere {(p. r_distance)} in der Kontroll-Liste
```

Klassifizierung

Ziel ist die Beurteilung von neuen Objekten, um sie bestimmten Klassen zuzuodnen. Die Klassen wurden unter der Nutzung der Objekt-Eigenschaften-Kombinationen bestimmt und dynamisch angepasst. Oft ist nicht bekannt, welche Eigenschaften-Kombinationen genau eine bestimmte Ziel-Klassifizierung bedingt, z.B. die Festlegung, ob ein Kunde eine Hochrisiko-Kunde bei einer Versicherung. Klassifizierung in diesen Sinn ist kontrolliertes Lernen (des Systems).
Das entscheidende ist das Erlernen der Unterscheidung der Klassen.

Klassifikation → binär oder diskret
Vorhersage → numerisch, analog

Klassifizierungs-Verfahren:

- Decision trees (Lernen von Unterscheidungs-Bäumen)
- k-nearst neighbor
- Bayes classifier
- Linear discremant function & SVM

Eigenschaften von Klassifizierern

- Korrektheit; Akkuratheit; geringe Fehlerzahl; hohe Güte
- Interpretierbarkeit; Verständlichkeit
- Effizienz (in der Trainings- und in der Nutzungs-Phase
- Skalierbarkeit
- Robustheit

m-fold Cross Validation ()
klassifizierter Datenbestand wird in m Teil-Datenbestände aufgeteilt
aus m-1 Teil-Beständen wird der Klassifizierer bestimmt
dieser wird auf dem übrig gebliebenen Teildaten-Bestand angewendet und mit der originalen Klassifikation geprüft
das wiederholt man für alle m Teil-Datenbestände

leave-one-out ()
wie m-fold Cross, allerdings bei m = n (es wird nur ein Objekt gegengetestet)

Korrektheit ist Quotient aus der Anzahl der richtigen Klassifikationen und der Gesamtmenge
Fehler-Rate ist Quotient aus der Anzahl der unrichtigen Klassifikationen und der Gesamtmenge
Klassifikations-Fehler (in Abhängigkeit von der Eigenschaften-Anzahl (Parameter)) angewendet auf Trainings-Daten (wird kleiner) oder alternativ auf die Test-Daten (wird größer)

Decision trees
Unterscheidung der Daten-Objekt über nacheinander angewendete Beschneidung der Menge anhand von Grenzen
Grenzen müssen lernend angepasst werden, bis sie zur Klassifikation der Test-Daten passen
dazu benutzt man Klassen-reine Teil-Datenbestände

Algorithmus (Decision trees)

```
ID3(Examples, TargetAttr, Attributes):
        create a Root node for the tree;
        if all Examples are positive, return Root with label=+;
        if all Examples are negative, return Root with label=-;
        if Attributes = 0, return Root with label = most common Value of TargetAttr in
                                Examples;
        else
                A = the best dicision attribute for next node;
                assign A as decision attribute for Root;
                for each possible value v_i of A:
                        generate branch coresponding to test A = v_i;
                        Examples_vi = examples that have value v_i for A;
                        if Examples_vi = 0 add leaf node with label = most common value
                                of TargetAttr in Examples;
                else add subtree ID3(Examplesvi, TargetAttr,Attributes\A);
```

Vorteile:
- hierarchisch
- lineare Dimensionen
- schnell auf diskreten und numerischen Daten
- schnelles / effizientes Lernen / Klassifizierung
- gute Genauigkeit der Bäume → gefunden Klassifizierung
- für Menschen gut verständliche Klassifizierung

Nachteile:
- nicht stabil: kleine Veränderungen in den Daten können große Veränderungen im Baum erzeugen

Nearest Neighbor Classifiers
Suche nach dem ähnlichsten (vergleichbaren) Datensatz; Berechnungen von Distanzen
Bewertung / Klassifizierung dann entsprechend diesem Nachbarn
Instanz-basiertes Lernen; Lazy evaluation
gesamte Arbeit erfolgt gleich in der Klassifizierung (kein vorheriges Training notwendig)
geht auch über vorherige Bestimmung von Clustern-Repräsentanten (z.B.: Zentroid)
hohe Klassifizierungs-Güte
gute Anpassung an große Datenmengen

man braucht:

- eine Distanz-Funktion
- Entscheidungs-Menge (Decision set) (empirisch: $1 << k < 10$
- Entscheidungs-Regeln (Decision rule) (z.B. nach Mehrheit aus Entscheidungs-Menge; Normierung mit Kehrwert der Distanz möglich; Häufigkeiten; kleinste Distanz)

nachteilig sind:
relativ naiv (kein hinterlegtes Modell)
zur Bewertung ev. gesamter Datenbestand anzufassen; umgehbar durch Indizierung / Sortierung
bei höherer Dimension in den Daten-Beständen gehen immer mehr irrelevaten Daten mit ein
ev. Festlegung relevanter Attribute notwendig

Bayesian Classifier
Beurteilung über Wahrscheinlichkeiten der Zugehörigkeit zu den Klassen
probalistisch
Klassen müssen vorher analysiert werden (Lern-Phase)
Berechnung der bedingten Wahrscheinlichkeit
schwierig bei hoch-dimensionalen Daten

Linear Classifier
Berechnung einer Trenn-Linie (Gerade) bzw. einer Trenn-(Hyper-)Ebene zwischen den verschiedenen Klassen
sehr einfach
läßt auf nicht-lineare Klassifikation erweitern

empfindlichen gegenüber Ausreißern; also nicht stabil
sehr optimistische Annahme. dass Klassen linear zu trennen sind
bei höheren Dimension recht rechenaufwändig

Support Vector Machines (SVMs)
stabiles Lernen der trennenden Hyper-Ebene
nicht-lineare Trennung möglich
ev. Transformation in einen linear Raum
durch Kernel.Methoden wird Aufwand reduziert

Ensemble Classification
Suche nach dem / einen optimalen Klassifizierer
Kombination mehrerer Klassifizierer
z.B.: Bagging
Durchschnitt mehrerer Klassifizierer über zufällig gebildete Teil-Datensätze aus einem Trainings-Datenbestand
z.B.: Boosting
Kombination mehrerer schwacher Klassifizierer zu einem starken
neu hinzukommende Klassifizierer sollen sich auf die Objekt mit bisher fehlerhaften Zuordnung konzentrieren

Frequent Itemset Mining
Suche nach wiederkehrenden / gehäuften Mustern
in Transaktions-Datenbanken (z.B. Einkäufe)
z.B. Kombinationen von bestimmten Produkten kaufen → Produkt-Vorschläge für weitere Einkäufe (recommendation systems)

erste dokumentierte Beziehung in Amerika, am Freitag Nachmittag gab es gehäufte Einkäufe von Windeln und dazu Bier → Ausnutzung dadurch, dass neben Windel-Aufsteller gleich auch eine Palette mit Bier steht (und umgekehrt)

Apriori-Prinzip
Apriori-Algorithmus
Es wird für jede Anzahl-Element-Kombinationen eine Kandidaten-Menge generiert und diese analysiert (auf Häufigkeit (geforderte Minimal-Häufigkeit)
die häufigen Kandidaten werden in eine Lösungs-Menge übernommen
aus den Lösungs-Kandidaten werden nur Kombinationen mit einem zusätzlichen Element generiert und in die Kandidaten-Menge der Anzahl+1-Element-Kombination übernommen
usw. usf. bis keine Kombination mehr die Minimal-Häufigkeit erfüllt

Frequent Pattern Tree
Elemente werden nach Häufigkeit sortiert und ev. gegen eine Minimal-Häufigkeit abgeschnitten
aus Transaktionen wird unter Beachtung der Häufigkeiten (innerhalb der Transaktionen) ein bedingter Baum mit Häufgkeitszählern in den Knoten konstruiert

Assoziations-Regeln

support(A → B) = Anzahl_der_Transaktionen(A, B) / Gesamt_Anzahl_Transaktionen
(Häufigkeit de Beziehung)
support(A → B) = support(A,B)

confidence(A → B) = Anzahl_der_Transaktionen(A, B) / Anzahl_der_Transaktionen(A)
(Stärke der Beziehung)
confidence(A → B) = support(A,B) / support(A)

unexpected(A → B)
(Unerwartungswert)

actionable(A → B)
((Aktions)Nutzbarkeit)

Lift = Häufigkeit(A,B) / Häufigkeit(A)*Häufigkeit(B) = Häufigkeit(A → B) / Häufigkeit(A)
= Häufigkeit(B → A) / Häufigkeit(B)

Outlier Mining

Ausreißer-Suche und -Klassifizierung
Anwendung:

- Erkennung von Anomalien
  - z.B.: bei Erkrankungen
- Kreditkarten-Mißbrauch
- Wartungs-Zyklen für Bauteile / Ausfallhäufigkeiten
- Auffinden von Rausch-Daten
-

Def. z.B. über kleine Cluster oder seltene Objekte, die scheinbar einem anderen Modell entstammen oder bei klassifizierten Objekten, die keiner Standard-Klasse zugeordnet sind
große Abstände zu anderen Cluster; spärlich besetzte Cluster
Objekte mit geringer Häufigkeit
Objekte an den Rändern von Cluster / Häufigkeits-Verteilungen
völlig neue Daten

Distanz-basierte Suche nach Ausreißern

Auffinden von gehäuften Randwerte bezogen auf eine GAUß-Verteilung z.B. jeweils 2,5 % der Daten, d.h. die restlichen 95 % werden als regulär betrachtet
setzt voraus das es sich bei den Daten um eine GAUß-Verteilung handelt (stimmt selten!)
versagen bei unterschiedlichen Dichten

Dichte-basierte Suche nach Ausreißern

suche / finden von lokalen Ausreißern
Objekte würden nur zur normalen Gruppe gehören, wenn (deutlich) veränderte Parameter benutzt werden müsste
k-nächste-Nachbar-Distanz ist deutlich größer / deutlich größer als die k-nächste-Nachbar-Distanz der bei k Nachbarn (des Prüf-Objektes)
man erhält Liste der k-nächste-Nachbar-Distanzen, die ausgewerte werden kann
z.B. kann der Nutzer die Grenze anhand der sortierten Liste festlegen (Ranging-Grenze)
z.B. über die lokale Erreichbarkeits-Dichte (Bewertung der Dichte um ein Objekt)

lokale Erreichbarkeits-Dichte = 1 / durchschnittliche Erreichbarkeits-Distanz(minPunkte)
LOF Lokaler Outlier Factor = 1 → Inlier; >>1 → Outlier; wenig >1 → lokale Outlier

Ausreißer in multi-dimensionalen Räumen

Subspace Mining

aufgrund verschiedener Sichten auf die Daten ergeben sich unterschiedliche Gruppen
Suche von Clustern und Ausreißern in Teilgruppen der Attribute
relaiv offen hinsichtlich der ausgewählten Attribute
erster Ansatz sind Auswertungen von zufälligen Projektionen

Einsatz von "Monte Carlo"-Algorithmen

Data Mining bei kontinuierlichen Datenströmen

unendliche Daten(-Ströme) / Zeit-Reihen; Gesamtdatenbestand nicht greifbar
z.B. Kursdaten von Aktien
auch Daten aus verschiedenen Kategorien
Finden von Anomalien
z.B. bei Transaktionen (Kreditkarten); Sensordaten (Auto's, Reaktoren, …)

Data Mining bei Graph-Daten / in Graphen

z.B. soziale Netzwerke (Knoten = Personen; Kanten sind Verbindungen)
Problem, dass Anzahl der Kanten deutlich größer ist als die Anzahl der Knoten

semantische Beziehungen in Daten / Wissen (Wissens-Graph) (z.B. auch Bookmarking)

biologische Daten
z.B. Protein-Protein-Interaktionen; Substrat-Protein-Interaktionen; Metabolismen-Netzwerke

attributierte Graphen = Graphen mit Knoten, die mehrdim. Attribute besitzen

Suche nach homophilen Gruppen bzw. korrelierten Attributen

Q: http://www.cs.uiuc.edu/~hanj/bk3
(ältere Version: http://web.engr.illinois.edu/~hanj/bk2/slidesindex.htm)

Links:
http://www.dataminingbook.info (ZAKI, Mohammed J.; MEIRA, Wagner jr.: "Data Mining and Analysis")
https://hpi.de/mueller/tutorials/graph-exploration-sigmod.html ()

Supermarkt-Kette benutzte Kundenkarten
analysierte Kauf-Verhalten von schwangeren Frauen
Interesse lag auch in der Frage, mit welchen Coupon's man Kunden eher glücklich macht

Analyse, was die Kunden neu kauften (z.B. Parfüm-freie Körperlotion, bestimmte Lebensmittel-Ergänzungen (Mg, Ca, Zn)
und was sie nicht mehr kauften (Zigaretten und Alkohol)

es wurden 25 Artikel beobachtet und die junge Frau kam auf 23 Treffer

das System ermittelte eine Wahrscheinlichkeit von 87 %, dass sie schwanger ist und konnte den Geburts-Termin (ungefähre Dekade im Monat) voraussagen

Anekdote:
Ein amerikanischer Vater regt sich bei der Geschäftsleitung der Supermarkt-Kette "Target" darüber auf, dass seine Tochter seit kurzem verstärkt Werbung (eMail's) für Artikel erhält, die etwas mit einer Schwangerschaft zu tun haben. Sie gehe schließlich noch zur Highschool und ob man sie zu einer Schwangerschaft ermutigen wolle?
Der – den Fall bearbeitende – Manager hatte keine Ahnung und kontrollierte die eMails. Sie enthielten tatsächlich Werbung für Mutterschafts-Kleidung und Kinderzimmermöbel sowie lächelnde Baby's. Ein paar Tage später entschuldigte sich dann der Manager für die "nicht-angepassten" eMail's.
Später hat der Vater seine Tochter zur Rede gestellt und diese gab beschämt zu, dass sie schwanger sei und im August gebären werde.
Der Mann entschuldigte sich dann später bei der Supermarktkette für seine unberechtigte Beschwerde. Offensichtlich hätte es in seinem Haus Aktivitäten gegeben, von denen er nichts mitbekommen habe.

Ausnutzung von Beziehungen in Daten (z.B. gemeinsamer Kauf von Windeln mit Bier durch Männer → z.B. Ausnutzung durch gemeinsame Produkt-Angebote / Produkt-Präsentation / Produkt-Plazierung im Markt)

8.x.z. Data Mining an Texten – Text-Analysen, …

Texte sind beliebte / zuverlässige Wissen-Speicher
besonders interessant, weil viele Daten in Text-Form vorliegen

- Bücher
- Zeitungen
- Zeitschriften
- eMail's
- social-media-Beiträge (tweets, likes, …)
- Patente
- Lexika
- wikipedia-Seiten
- Webseiten

Text Mining

Text Mining-Bereiche

- **Information Retrieval** **Text-Analyse**	Statistik
- **Natural Language Processing** **Verarbeitung von (natürlicher) Sprache**	Grammatik Text-Struktur-Erkennung
- **Knowledge Extraction** **Wissens-Extraktion**	Semantik Zusammenhänge von Text-Informationen

TF-IDF (term frequency – inverse document frequency)

term frequency … Term-Häufigkeit (Wort-Häufigkeit)
inverse document frequency … umgedrehte Häufigkeit eines Terms in den Dokumenten
Grund-Maß des Text-Mining's
Maß für die Relevanz von Worten in einem Text-Korpus

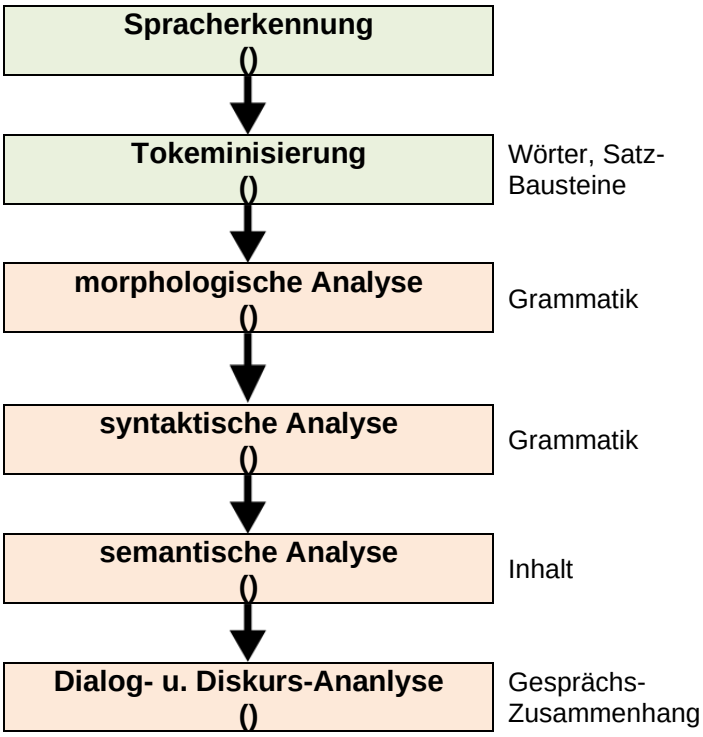
Worte mit einer sehr hohen Dokumenten-Häufigkeit sind wahrscheinlich ohne größere Bedeutung und werden als Stop-Worte verstanden (z.B. Bindeworte, Artikel, …)

ein hoher TF-IDF-Wert besagt, dass ein Wort in einem bestimmten Dokument besonders häufig vorkommen, aber in allen Dokumenten zusammen eher selten ist → Wort mit besonderer Bedeutung

Computer-Linguistik

Natural Language Processing

wenn akustische Signale in geschriebene Texte umgesetzt werden sollen



Tokenisierung zerlegt die Sätze in einzelne Einheiten – meist Wörter
zuerst müssen aber z.B. die Sätze selbst selektiert werden
ev. auch Zerlegung von zusammengesetzten Substantiven od.ä. in einzelne Begriffe, um sie besser einordnen zu können

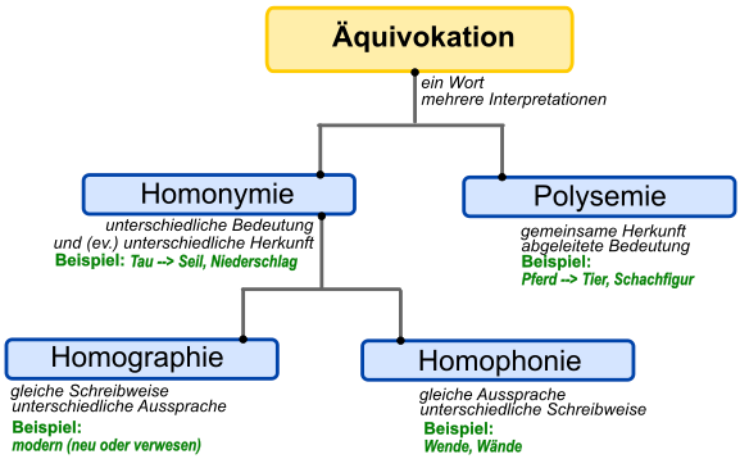
die Grammatik der Sätze steht bei der morphologischen Analyse im Vordergrund
deklinierte Worte werden dann in die Grund-Form – oder sogar nur den Wort-Stamm – zurück übertragen, um sie im Weiteren einfacher auszuwerten

die syntaktische Analyse bestimmt nun die Satz-Bausteine / -Teile mit ihrer grammatikalischen Form
z.B. müssen ja Objekt und Subjekt klar voneinander unterschieden werden können

den Satz-Teilen wird dann in der semantischen Analyse eine Bedeutung zugewiesen
hier sind jetzt auch Bezüge zu anderen (vorlaufenden) Sätzen oder Weltkenntnisse notwendig, um einen Satz inhaltlich verstehen zu können

z.B.: Ein Junge wandert die Straße entlang. Michael singt leise vor sich her.
Das semantische Erkennungs-System muss hier z.B. erkennen, dass Michael ein Junge ist und derjenige ist, der im Satz davor gewandert ist.

besonders schwierig wird das, wenn ein Wort sehr viele unterschiedliche Bedeutungen haben kann und sich die gemeinte Bedeutung erst aus dem anderen Satz-Teilen oder dem gesamten Text ergibt.
weitere Probleme beim Übergang von Namen zu Spitznamen od.ä., indirekten Personen-Bezügen, …
z.B. Hamburger Menü
kann ein spezielles Menü in Gaststätten von Hamburg sein, es kann sich aber auch um ein Menü mit einem Hamburger (einer gebratenes Hackfleisch-Scheibe) handeln
eine weitere Verwendung hat der Begriff bei den modernen Apps für die gestapelten Menü-Einträge, die sich hinter einem Symbol mit meist drei dickeren



Strichen versteckt

Aufgaben:

1. *Wählen Sie sich ein Wort (zusammengesetztes Wort usw.) aus und tragen Sie dazu besonders viele Bedeutungen zusammen! Wer findet das Wort mit den meisten unterschiedlichen Bedeutungen!*
2. *2.*
3. *3.*

Beispiel für eine semantische Suchmaschine ist "WOLFRAM alpha" (https://www.wolframalpha.com/)
hier kann man englischsprachige Fragen und Wortgruppen eingeben und erhält passende Antworten und / oder Suchergebnisse

Named Entity Recognition

Benennung von erkannten Objekten / Begriffen / Identitäten

einfachter Ansatz ist ein Wörterbuch-basierter Ansatz
dort sind die Worte mit ihrer möglicher Bedeutung verzeichnet

heute versucht man mittels maschinellen Lernen die Analyse flexibler und problem-orientierter durchzuführen
als Trainings-Texte können z.B. gut Lexika oder auch Hypertexte (HTML-Seiten) genutzt werden, da hier viele Entitäten als Links auf andere Seiten / Hinweise auf andere Stichworte ausgeführt sind
besonders gut funktioniert dies z.B. mit wikipedia-Texte, die diese sowohl als Lexika und als HTML-ähnlicher Text fungiert
außerdem sind viele Seite kategorisiert

Named Entity Disambiguation

disambiguieren
nach der Erkennung von Worten soll jetzt eine Identifikation (konkrete Bedeutungs-Zuordnung) durchgeführt werden

semantische, maschinen-lesbare Datenbanken:

- **wikidata** https://www.wikidata.org/
- **DBpedia** Datenbank mit semantischen Informationen aus wikipedia https://wiki.dbpedia.org/
- **Yago** https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

Relationship Extraction

Herstellen von gesicherten Beziehungen zwischen den Entitäten
prüfen von bindenden Wörtern zwischen verschiedenen Entitäts-Typen
ermitteln von Richtungen von Beziehungen (Wer liefert was? Was ist die Ursache, was die Wirkung? Wer ist der Erwachsene und wer das Kind? …)

Linguistische Maße

Anzahl der Wörter

durchschnittliche Wortlänge

durchschnittliche Anzahl von Silben in den Wörtern

Anzahl der Substantive, Verben, …

Anzahl der Sätze mit bestimmten Bestandteilen

Anzahl der Nebensätze

Analyse der durchschnittlichen Satzlänge und deren Visulisierung:
https://www.uni-konstanz.de/mmsp/pubsys/publishedFiles/KeOe07.pdf

Verse length (Vers-Länge)

Hapax Legumena
Anzahl der Worte, die einmal im Text vorkommen

je mehr solcher Worte in einem Text vorkommen, um so komplizierter / ausschweifender / phantastischer ist dieser

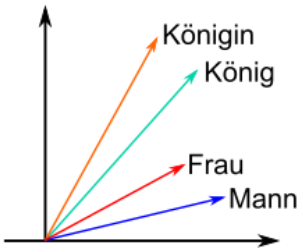
Hapax Dislegumena
Anzahl der Worte, die genau zweimal im Text vorkommen

darüber typische Charakterisierung von Texten und damit auch von Autoren möglich

Word Embedding (Wort-Einbettung)

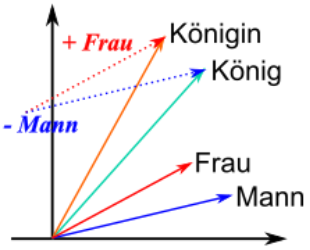
Anordnung von Wörtern in einem mehrdimensionalen Raum in der Form, dass Worte mit ähnlicher Bedeutung näher in diesem Raum stehen als solche, bei denen keine Beziehung besteht

z.B. Erkennen oder Abfragen von Synonymen (diese liegen im Raum dicht beieinander)
läuft praktisch vollständig automatisiert und wird stark durch maschinelles Lernen getrieben



praktisch kann man dann auch mit den Vektoren arbeiten und rechnen
daraus lassen sich dann wieder semantische Beziehungen herstellen

z.B.: gegeben sind die Vektoren von Mann, Frau, König und Königin
über die "Vektor.Berechnung" König – Mann + Frau erhält man dann Königin (oder einem synonymen Begriff (oder zumindestens einen in dessen Nähe))



Sentiment Analysis (Stimmungs-Analyse)

erkunden der Stimmung, Gefühle, Meinung, Haltung, Parteilichkeit, … eines Textes

z.B. Produkt-Bewertungen, Post's in Social Media usw. eintufen

google-mail hat mal getestet, die eMail's vor dem Abschicken zu analysieren und gegebenenfalls empohlen, den Text erst am nächsten Tag abzusenden

z.B. über Stichwort-Listen

auch gute Anwendung für maschinelles Lernen
z.B. aus Kombinationen aus textuellen Bewertungen und vergebenen Sternen

praktisch heute schon in vielen Social Media angewendet → Filter-Blase
Nutzer bekommen nur noch positiv gestimme / anregende / interessante / … Beiträge zu sehen, um ihn weiter im Medium zu halten
praktisch werden vor allem die eigenen Meinungen usw. usf. wieder angezeigt, so dass man den Eindruck hat, dass wäre die Welt-Meinung
echter Pluralismus so nicht (mehr) möglich

8.x.y. BigData zum selber Ausprobieren

8.x.y.1. google BigQuery

BigQuery ist eigentlich Kosten-pflichtiger google-Dienst

mit einem google-Konto aber 60 Tage zum Testen Kosten-frei

cloud.google.com/bigquery

mit einer Meta-Programmiersprache auch digital und automatisiert nutzbar

"BigQuery ML" basiert weitesgehend auf SQL

8.x.y. Smart Data

Verarbeitung großer Datenmengen, die sich aus der Vernetzung von Geräten ergeben
Bereitstellung von "kleinen" effektiven / benutzbaren Daten aus Big Data

???y
welche Daten sind für die aktuelle Station wichtig
welche Daten sind für das Netz relevant, wovor muss gewarnt werden, worauf hingewiesen

8.x.y. Skalierbares Daten-Management

8.x.y.0. Parallelisierung

großer Bedarf an gleichzeitigen Abarbeitungen (Zeit-gleiche Bestellungen, …)

Parallelisierungs-Arten

- **Aufgaben-Parallelismus** **task parallelism**	unabhängige / separate Task's werden erledigt
- **Anweisungs-Parallelismus** **instruction(-level) parallelism**	unter Nutzung von mehreren Prozessoren (CPU's) / Prozessor-Kernen (Core's) / Graphik-Prozessoren (GPU's) werden die gleichen Bearbeitungs-Schritte zur gleichen Zeit erledigt Daten müssen weitgehend unabhängig sein (zumindestens während der Verarbeitung)
- **Daten-Parallelismus** **data parallelism**	verschiedene Daten werden verschiedenen Datenträgern gespeichert sie können aber ohne weiteres gleichzeitig und gleichartig bearbeitet werden typisch für Graphik-Prozessoren (GPU's)

Grenzen

Blade → Rack → Data Center / Rechenzentrum
Ausfälle

- ein typisches Beispiel für ein Rechenzentrum (nach Jeff DEAN (2009)):
  - innerhalb von 2 Jahren 1x eine Überhitzung → Herunterfahren aller Rechner innerhalb von 5 min notwendig und das schrittweise Hochfahren der einzelnen Rechner über 1 bis 2 Tage
  - 1x jährlich Ausfall eines Strom-Verteilers → 500 – 1000 Rechner sind aus; Reparatur- und Wieder-Aktivierungs-Zeit um die 6 Stunden (ev. große Daten-Verluste, weil Daten nur im Hauptspeicher waren)
  - 1x jährlich in einem Bereich fällt die Kommunikation aus, oder funktioniert nicht mehr richtig → rund 6 Stunden um die Geräte zu prüfen und ev. alle neu zu starten
  - alle 2 Monate gehen einzelne Racks in undefinierte / verwirrte Zustände über; meist bei unklaren Ursachen (z.B. im Betriebssystem, …) → bei vielen Maschinen ist mit Daten-Verlusten zu rechnen
  - 1000 Rechner fallen insgesamt aus (praktisch 3 pro Tag)
  - Tausende an Hardware-Komponenten fallen aus (schlechter Hauptspeicher, zulangsame oder überlastete Festplatten, …)
  - Datenleitungen, die von Tieren angefressen werden / bei Baggerarbeiten durchgetrennt werden / …

Energie-Verbrauch steigt expotentiell

AHMDAHLs Gesetz

die Beschleunigung eines Programmes ist durch den nicht-parallelisierbaren Anteil beschränkt
je größer der nicht-parallelisierbare Anteil ist, umso weniger läßt sich das Programm insgesamt beschleunigen

$$S_{max} = \frac{1}{(1-f)+\frac{f}{p}}$$

p … Anzahl der Prozessoren
f … parallelisierbarer Anteil

angenommener Anteil von parallelisierbaren Code-Abschnitten liegt bei 90 % (was allgemein sehr hoch ist (nur bei Graphik-Berechnungen typisch))
der Beschleunigungs-Effekt S liegt z.B. bei 10 statt 1 Rechner nur bei einem Faktor von 5,3 und nicht 10, wie man eigenlich gehofft hat

z.B.:
f = 0,9
bei 10 Prozessoren

$$S_{10} = \frac{1}{(1-0{,}9)+\frac{0{,}9}{10}} \approx 5{,}3$$

bei 20 Prozessoren

$$S_{20} = \frac{1}{(1-0{,}9)+\frac{0{,}9}{20}} \approx 6{,}9$$

8.x.y.z. OLAP

Online Analytical Processing
große Daten-Mengen analytisch auswerten
wenige Anfragen pro Zeiteinheit mit sehr großen Ergebnis- bzw. betroffenen Daten-Mengen
meist sehr aufwändig / rechen-intensiv / viele Arbeitsschritte (z.B. statistische Auswertungen, Visualisierungen, …)

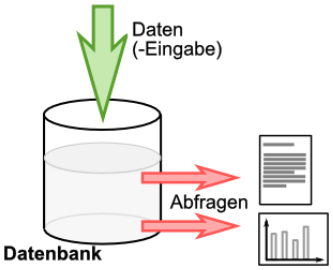
verteilte (Datei-)Systeme

ein Rechner ist der Name-Node (Namens-Knoten) er hat die Funktion eines Server's
bei ihm passiert die Datei-Speichung nur virtuell, er gibt nur "seinen guten Namen" dafür her
praktisch sorgt der Name-Node für die Verteillung der Daten (Dateien oder Datei-Segmente) auf mehreren Data-Note's (Daten-Knoten)
typische Daten-Segmente haben in der Praxis eine Größe von 128 MB
in verteilten Systemen sind die Daten häufig mehrfach (typischerweise dreifach) vorhanden, um Ausfällen vorzubeugen und damit eine hohe Datensicherheit zu garantieren
weiterhin ergeben sich Möglichkeiten zum Lasten-Ausgleich
Nachteile der mehrfachen Speicherung sind erhöhte Hardware-Aufwändungen und eine hohe Daten—Redundanz
die genaue Verteilung und Absicherung der Konsistenz ist ein technisches Problem, was in speziellen verteilten Datei-Systemen realisiert wurde und für den Nutzer nicht sichtbar wird
z.B.: google-Datei-System, S3-System von amazon, HADOOP
heute übernehmen die Daten-Knoten auch noch Bearbeitungs-Aufgaben zu den abgelegten Daten, damit spart man sich den teuren Transport der Daten über das Netz
statt dessen werden nur die Ergebnisse zum Namens-Knoten geschickt
die Aufgaben werden von Client's an den Namens-Knoten gestellt, dieser verteilt sie an die Daten-Knoten (→ verteilte Berechnungen) und die schicken eben die Ergebnisse an den Namens-Knoten zurück
dieser bedient dann die Client's
Beispiel: BOINC-Projekte (verteiltes Berechnungs-System mit Bildschirm-Schoner-Funktion) (→ boinc.berkeley.edu)

Lösung wissenschaftlicher Probleme (Berechnung von Protein-Strukturen, Klima-Modellen, …; Suche nach außerirdischen Funksignalen, Gravitations-Wellen, …; Lösen von mathematischn und kryptologischen Problemen)

Daten-Verarbeitungs-Pinzipien

Batch-Verarbeitung



arbeiten auf einem sehr großen / dem gesamten Daten-Bestand
charakterisiert durch sehr komplexe Programme, die nacheinander (als Stapel (Batch)) abgearbeitet werden (müssen)

meist ohne Nutzer-Interaktionen

Ergebnisse sind große Analysen mit sehr komplexen Visualisierungen usw.
typisch Quartal's- oder Tages-Auswertungen, …
begrenzender Faktor ist hier die Größe des Speicher's; praktisch nur Volume-Problem (→ Big V's)
für die komplexen Analyse-Programme ist aber auch Leistungs-fähige Hardware notwendig

Datenstrom-Verarbeitung

(data)stream processing
Daten-Verarbeitung beschränkt sich auf einen aktuellen Daten-Ausschnitt (laufende Bestellungen oder Bestellungen der letzten Stunde usw.usf.; Sensor-Daten, Log's, …)

neben Volume ist hier auch Velosity interessant

meist nur einfache Programme (z.B. Grenz-Wert-Überwachung)

Ergebnisse sind flüchtige Kennzahlen für die betrachteten Zeiträume
meist nur einfache Visualisierungen; z.B. Kauf-Vorschläge für Artikel, die man sich länger / genauer angesehen hat
Effektivität und die Grenzen werden durch die Rechen-Kapazität bestimmt; Speicher interessiert kaum

an die Datenstrom-Analyse kann dann auch die Speicherung und eine weitere / unabhängige Batch-Verarbeitung anschließen

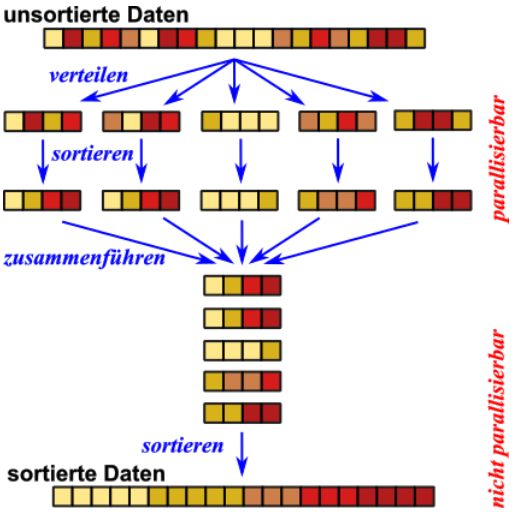
Problem: Skalierung

Wie reagieren meine Systeme, wenn sich die Menge der Daten im größeren Stil verändern?
Was passiert, wenn Geräte ausfallen? Schaffen die anderen dann die Aufgaben?
Was passiert, wenn neue Daten(-banken) mit eingebunden werden sollen?

→ Skalierungs-Muster

???

- Phase 0: Daten verteilen
- Phase 1: Berechnungen auf Teilmengen (der verteilten Daten)
- Phase 2: Zusammenführen der Ergebnisse aus der Teilmengen-Analyse zu einer Gesamt-Analyse (auf die betrachtete, unverteilte Daten-Menge)



die Vorsortierung kleinerer Daten-Pakte auf den verteilten Systemen ist deutlich effektiver, als ein vergleichsweises gemeinsames Sortieren auf nur einem Rechner. Das Vorsortieren lässt sich zudem parallelisieren.
die vorsortierrten Daten-Blöcke werden dann an einer Stelle – meist ist dies der Name-Node – zu einer Gesamt-Sortierung zusammengesetzt.
Dieser Teil ist nicht parallelisierbar, weil jetzt Abhängigkeiten zwischen allen Daten bestehen.
im letzten Schritt kann aber auch wieder Verteilen eingearbeitet werden.
So könnte der Name-Node immer jeweils alle gleichwertigen Element auf einen der Data-Node's schieben und mit den anderen Elementen genau so verfahren. Aus der Sicht des Name-Nod's kann dann über einen Zugriff auf die Data-Note's in der richtigen Reihenfolge die sortierte (Gesamt-)Reihenfolge erzeugt werden.
Neben Sortierungen eignen sich auch Gruppierungen, Klassifikationen und Aggregationen gut für das obige Verfahren. Wichtig ist, dass der parallelisierbare Anteil groß genug ist.
Ein klassisches Beispiel für eine Aggregation ist die Summen-Bildung. Hier kann man auf den verteilten System z.B. Teil-Summen bilden und nur die dann an den Name-Node zurückschicken. Dieser kann aus den teil-Summen dann eine Gesamt-Summe bilden.
Ähnlich funktionier dies für Produkte (Multiplikationen), bestimmte Mittelwerte und Zählungen.

Map/Reduce

ist ein Programmier-Modell für die parallele Abarbeitung von Aufgaben
entwickeln von Jerrey DEAN und Sanjay GHEMAWAT (google Inc.)
gearbeitet wird eine Rahmen aus zwei Funktionen – eben Map und Reduce
innerhalb von Map und reduce können wiederum andere Funktionen aufgerufen werden

Map $(k_1 \times v_1) \rightarrow (k_2 \times v_2)$	nimmt Schlüssel-Wert-Paar (key, value) entgegen, wandelt diese irgendwie geartet um und gibt ein neues Schlüssel-Wert-Paar aus (diese Operationen können vollständig parallel für jedes Schlüssel-Wert-Paar gemacht werden, diese immer unabhängig voneinander sind!)
Reduce $(k_2 \times list(v_2)) \rightarrow (v_3)$	nimmt einen Schlüssel sowie eine Liste mit allen Werten, die genau diesen Schlüssel haben und erzeugt aus dieser Liste einen neuen Wert (z.B. die Summe) hier kann wieder für alle Schlüssel (k_2) parallel gearbeitet werden (weil für diesen wieder alle Werte unabhängig von den anderen sind)

zwischen Map und Reduce liegt aber ein Daten-zusammenfassender Schritt, der sich nicht parallelisieren läßt

die im Map-Teil berechneten Schlüssel-Wert-Paare müssen neu angeordnet / umsortiert / zusammengefasst / gruppiert werde, damit sie dann in der Reduce-Teil wieder parallel weiter verarbeitet werden können

typische Aufgabe: gleiche Wörter-Zählen
(praktisch das Hello-Welt-Beispiel für Map-Reduce)

```
map(dateiname, zeile){
    for each (wort in zeile)
        emit(wort, 1);}
```

aus:	wird:
O Romeo! Warum denn Romeo? Verleugne deinen Vater, deinen Namen, Romeo!	O, 1 Romeo, 1 Warum, 1 denn,1 Romeo, 1 Verleugne, 1 deinen, 1 Vater, 1 deinen, 1 Namen, 1 Romeo, 1

```
reduce(wort, nummern){
    int summe = 0;
    for each (eintrag in nummern){
        summe += eintrag;}
    emit(wort, summe);}
```

aus:	wird:
O, 1 Romeo, 1 Warum, 1 denn,1 Romeo, 1 Verleugne, 1 deinen, 1 Vater, 1 deinen, 1 Namen, 1 Romeo, 1	O, 1 Romeo, 3 Warum, 1 denn, 1 Verleugne, 1 deinen, 2 Vater, 1 Namen, 1

auch der einzelne Text kann parallelisiert verarbeitet werden, vielleicht z.B. so verteilt:

für den Programmier ergibt sich keine veränderung, er schreibt die gleichen Befehle auf

```
map(dateiname, zeile){
    for each (wort in zeile)
        emit(wort, 1);}
```

das System (Compiler, Systemsteuerung) entscheidet, wie in den verteilten Systemen die Teilaufgaben ausgeführt werden

	aus:	wird:
Rechner 1	O Romeo! Warum denn Romeo?	O, 1 Romeo, 1 Warum, 1 denn,1 Romeo, 1
Rechner 2	Verleugne deinen Vater, deinen Namen, Romeo!	Verleugne, 1 deinen, 1 Vater, 1 deinen, 1 Namen, 1 Romeo, 1

auch den Reduce-Teil programmiert man genau so wie bei einem Rechner, ein ev. verteiltes System wird intern mit den paralelelisierbaren Teil-Aufgabe versorgt

```
reduce(wort, nummern){
    int summe = 0;
    for each (eintrag in nummern){
        summe += eintrag;}
    emit(wort, summe);}
```

	aus:	wird:
Rechner 1	O, 1 Romeo, 1 Warum, 1 denn,1 Romeo, 1	O, 1 Romeo, 2 Warum, 1 denn, 1
Rechner 2	Verleugne, 1 deinen, 1 Vater, 1 deinen, 1 Namen, 1 Romeo, 1	Verleugne, 1 deinen, 2 Vater, 1 Namen, 1 Romeo, 1

	aus:	wird:
Rechner 1	O, 1 Romeo, 1 Warum, 1 denn,1 Romeo, 1	O, 1 Romeo, 3 Warum, 1 denn, 1 Verleugne, 1 deinen, 2 Vater, 1 Namen, 1
Rechner 2	Verleugne, 1 deinen, 1 Vater, 1 deinen, 1 Namen, 1 Romeo, 1	

Beispiel: gemeinsame Bekannte finden
Problem z.B.: 1,4 Mrd. Facebook-Nutzer (Stand: 2016) mit durchschnittlich 155 Freunden ergibt 979'999'999'300'000'000 Paare zum Prüfen:

```
map(person, freundesliste){
    for each (freund in freundesliste)
        if(freund < person)
            emit(<freund, person>, freundesliste);
        else
            emit(<person, freund>, freundesliste);}

reduce(<person1, person2>, freundesliste){
    emit(person1, person2>, freundesliste[1] ∩  freundesliste[2]);}
```

Hadoop

Hadoop Distributed File System (HDFS) von der Apache Software Foundation betreut

Q: Apache Software Foundation

erste Version 2006; 2011 die Version 1.0.0 freigegeben

Hadoop Map/Reduce Engine
besteht aus:

- Map/Reduce-Master: Job-Tracker
- Map/Reduce-Slave: Task-Tracker

Arbeits-Verfahren:

1. Input-Phase: Aufteilen der eingegangenen Daten in das HDFS
2. Map-Phase: Ausführen des Map-Teil's
3. Soft & Shuffle Phase: Sortieren und Neuverteilen der vom Map-Teil gelieferten Daten-Paare
4. Reduce-Phase: Kombination zusammengehörender Daten-Paare und Ausführen des Reduce-Teil's
5. Output-Phase: Ausgeben der vom Reduce-Teil berechneten Daten in das HDFS

8.x.y.z. OLTP

Online Transaction Processing
Management von einzelnen Transaktionen (speichern, abzuschließen, prüfen der Vollständigkeit (Ab- und Aufbuchungen), …)
Verarbeiten von sehr vielen Transaktionen innerhalb kurzer Zeiträume – praktisch sofort
Aktualisieren von Lagerbeständen, Aktivieren von Auslieferungen, Umbuchungen, …
viele Anfragen pro Zeiteinheit mit wenigen einzelnen abzuarbeitenden Schritten

verteilte Transaktionen

durchgeführte Operationen, z.B.:

- Senden eines Post's
- Hochladen eines Bildes
- Überweisen eines Geld-Betrages
- Verkauf / Bestellungen eines / des letzten Artikel's
- Mitbieten bei ebay

an einem verteilten System, soll weltweit sichtbar sein
auf viele Daten soll mit rel. kleinen Transaktionen zugegriffen werden

Absicherung der Vielzahl von Operationen, die durch die Transaktion ausgelöst wurden, als Ganzes und vollständig, nur dann ist die Transaktion abgeschlossen, ev. sonst Fehlermeldung oder Abbruch des Vorgangs (z.B. Bestellung)

beim Anmelden eines japanischen Nutzers soll auch eher ein japanischer Server genutzt werden
manche Informationen können verzögert verteilt werden, z.B. japanische Facebook-Einträge müssen nicht sofort auf jedem Server weltweit vorhanden sein

Ziel ist auch Erhöhung der Datensicherheit durch Replikation

ACID und BASE

Säure und Base
Konzepte, die vollständige Transaktionen beschreiben

ACID (Atomicity, Consistency, Isolation, Durability)
deutsch: Atomitizität, Consitenz, Isolation und Dauerhaftigkeit

wichtige Kriterien von Datenbank-Management-Systemen

- **Atomicity** Atomizität
  eine Transaktion kann nur als Ganzes ausgeführt werden
  eine Teilung ist nicht zulässig
  z.B. nur mit Ab- und Aufbuchung ist eine Finanz-Transaktion vollständig (nur ein Teil geht nicht!)

- **Consistency** Consistenz
  Passung aller Daten zueinander
  z.B. Ausschluß von zwei Nutzern mit der gleichen Kennung, eMail-Adressen, …; auch bei nachträglichen Änderungen und Fehlerhaften Korrekturen
  Aktualisierung von Lager-Beständen bei Verkäufen oder Nachlieferungen (Wareneingängen)

- **Isolation** Isolation
  jede Transaktion ist unabhängig von den Anderen
  z.B. bei der Reservierung eines Sitzplatzes im Flugzeug ist der Platz solange für andere Buchungen gesperrt, bis der reservierende Nutzer sich entschieden hat

- **Durability** Dauerhaftigkeit
  abgesicherte und ev. replizierte Speicherung der Daten von Transaktionen

in verteilten Systemen sind alle dieser Kriterien kaum vollständig einzuhalten
z.B. Verkauf des letzten Buches auf einem Server in Japan und einem in Europa zur gleichen Weltzeit

da die ACID-Eigenschaften praktisch nicht 100%ig zu schaffen sind, wurde als alternatives Konzept BASE entwickelt
basiert auf NoSQL-Datenbank-System
Base **A**vailable, **S**oft-state, **E**ventually consistent
garantiert nicht mehr die harten Kriterien von ACID
skaliert deutlich besser; kann sehr viele Transaktionen bearbeiten

wichtige Kriterien von Datenbank-Management-Systemen

- **Base Availabe** Verfügbarkeit besonders wichtig
  keine Transaktion soll verzögert werden, weil andere noch in der Bearbeitung sind

- **Soft-state** keine festen Zustände
  leichte Inkonsistenz wird tolleriert

- **Eventually consistent** an der Herstellung der Konsistens wird laufend gearbeitet

das CAP-Theorem

verteilte System werden von drei Eigenschaften bestimmt:
- Konsistenz (Consistency)
- Verfügbarkeit (Availability)
- Partitions-Tolerenz (Partition tolerance)

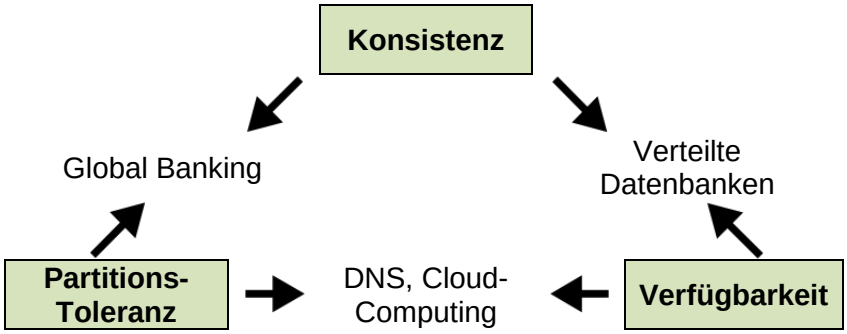
von diesen können aber immer nur maximal 2 auch erfüllt werden

Konsistenz z.B. bei Geld-Geschäften besonders wichtig
daraus folgt, dass nun entschieden werden muss:

- Soll das System immer verfügbar sein?

oder

- Soll die Partitions-Toleranz maximiert werden?



Daten-Partitionierung

Daten-Replikation

Frage, wie oft Daten gespeichert werden sollen?
Verfahren der Replikation

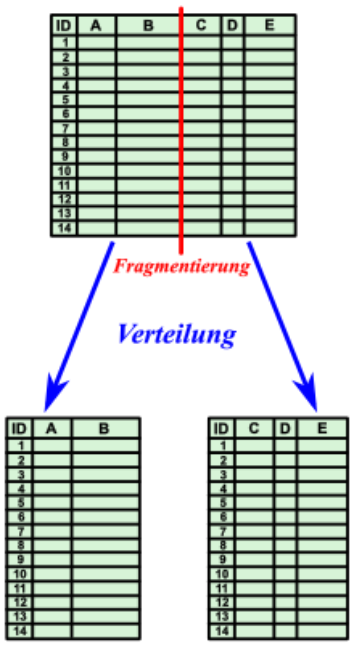
1. Fragmentierung (In welche Stücke sollen die Daten zerlegt werden?)
2. Allozierung (Wohin soll gespeichert werden? Wie oft sollen die Daten gespeichert werden?

Ziele

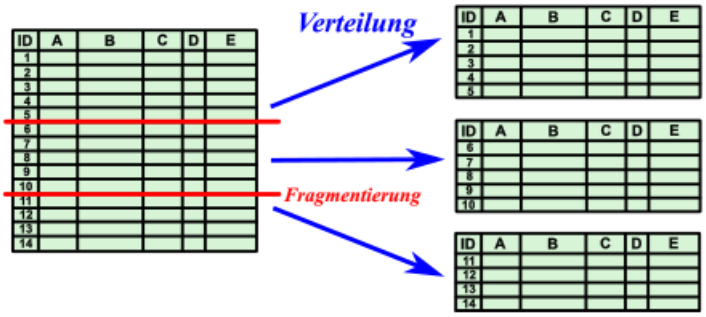
- gleichmäßige Verteilung
- praktische Verarbeitbarkeit (Daten für ein Problem möglichst auf einem Slave)
- Senkung der Daten-Transport-Kosten
- Lasten-Ausgleich (auch für nachgelagerte OLAP-Operationen)
- optimieren der Verfügbarkeit
- anpassen der Granularität der Daten nach Verwendung

Daten Allozierung

vertikale Partitionierung
Teilung der Datensätze / Transaktionen / … nach ihren Attributen
(z.B. Kunden-Daten einer Bestellung kommen auf den einen Slave, die Waren-Daten auf einen anderen
praktisch Tabellen durch vertikale / senkrechte zerlegt



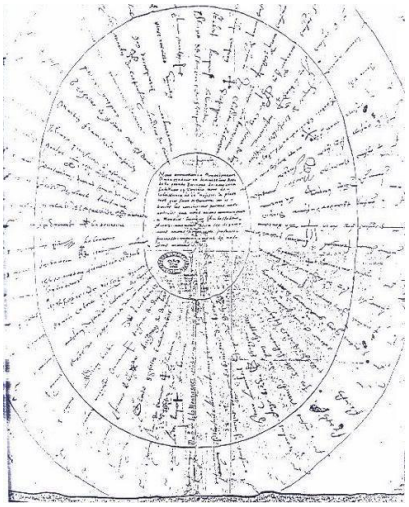
horizontale Partitionierung
Teilung der Daten in Gruppen von Datensätzen / Transaktionen / …
(z.B. die ersten 100 Bestellungen kommen auf den 1. Slave, die nächsten auf den 2. usw. usf.
praktisch Tabellen durch horizontale / waagerechte Linien zerlegt



Round-Robin-Verfahren

auch Rundlauf-Verfahren
Datensätze / Tarnsaktionen / … werden immer reihum auf die Slave's verteilt
z.B. einzeln, als Gruppe oder auch als pationierte Daten, weil Datensatz sehr groß ist

ein runder Robin war ein Beschwerde-Brief, der von seinen Unterstützern rundherum unterschrieben wurde
(alle sind somit gleichrangig beim Unterzeichnen, ein Hauptverantwortlicher (Rädelsführer) ist nicht zu identifizieren)
im 17.Jahrhundert in Frankreich verbreitet



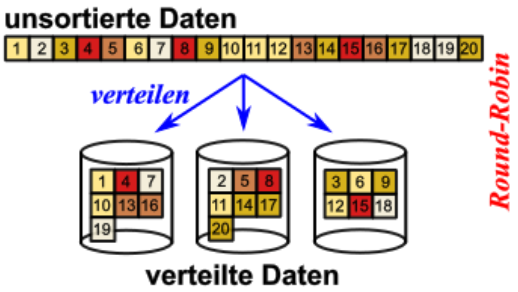
Runder Robin des
Jessé DE FOREST von 1621
Q: de.wikipedia.org (Paul Iee6977)

Vorteile:

- einfache Implementierung
- guter Lasten-Ausgleich
- gleichmäßige Verteilung der Daten

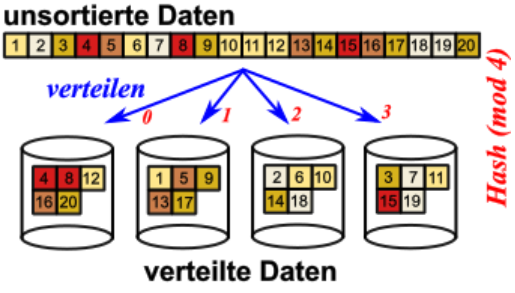
Nachteile:

- schlechte Performane bei Bereichs-Anfragen (Daten müssen dann erst zusammengesucht werden)



Hash-Verfahren

über den Datensatz wird ein Hash berechnet und nach dem berechneten Wert wird dann die Verteilung auf die Slave's / Bucket's (Eimer / Speicher für die gehashten Daten) vorgenommen
relativ einfache Hash-Funktion ist die Modulo-Berechnung
als Operator wird dann die Anzahl der Slave's benutzt



Vorteile:

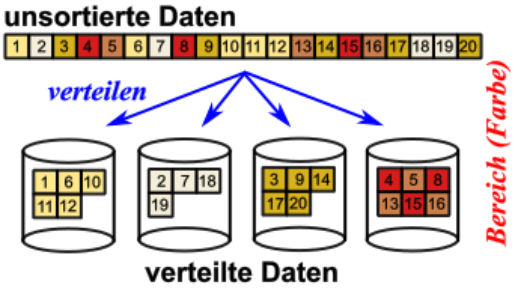
- normalerweise gleichmäßige Verteilung, weil meist zufällige Werte
- Daten mit gleichen Schlüsseln (hier gemeint die Ausgangs-Werte für die Berechnung der Hash-Funktion) landen auf dem gleichen Slave

Nachteile:

- zuzätzlicher Rechen-Aufwand für die Hash-Funktion
- ungleichmäßige Verteilung bei schlechter Hash-Funktion
- bei einem Datenverlust sind wieder viele Berechnungen oder Umsortierungen notwendig
- Bereichs-Anfragen mit erhöhtem Such-Aufwand verbunden

Bereichs-Partitionierung

die Verteilung erfolgt basierend auf bestimmten Attributs-Werte-Bereichen
das können Geschlecht, Körpergröße, Material-Eigenschaften, … sein
in der nebenstehenden Abbildung wurden Farben zu Unterscheidung herangezogen



Vorteile:

- optimal für Bereichs-Analyse bezüglich des Partitionierungs-Kriterium's
- Daten mit ähnlichen Schlüsseln (hier die Farbe) landen auf dem gleichen Slave

Nachteile:

- ungünstig für Bereiche anderer Attribute (, die nicht als Partitionierungs-Kriterium dienten)
- ungleichmäßige Verteilung bei schlechter Wahl des Kriterien
- ev. ungleiche Last-Verteilung
- Daten müssen vorbewertet werden

Consensus-Protokolle

wenn mehrere – relativ unabhängige – Systeme arbeiten, dann können Unstimmigkeiten entstehen
dann muss mittels eines sicheren Verfahrens ein Konsenz zwischen den Systemen hergestellt werden
einfache Beispiele: Zeit-Abgleich, Reihenfolgen

Consensus-Verfahren

- **Two Phase Commit** (z.B.: Hochzeit)
  wenn beide Bteilige sich sicher sind, dann kann eine (gemeinsame) Erklärung verbreitet werden
  wenn Bezahl-Bestätigung (Ab- und Aufbuchung über exter-

nen Bezahl-Dienst) kommt, dann kann die Bestellung weiter ausgeführt werden

- **Tree Phase Commit** zuerst wird Bereitschaft (für das Verfahren) geprüft dann werden die Bedingungen abgeglichenund zuletzt das Ergebnis festgelegt (und synchon umgesetzt)
- **Paxos** mit fünf verschiedenen Rollen (Aufgaben) im Verfahren Client, Acceptor, Proposer, Learner, Leader
- **Blockchain** Einigung über Veränderungen oder Besitztümer
- **Proof of Work**
- **Proof of Stake**
- **Proof of Activity** Mischung aus Proof of Work und Proof of Stake
- **Proof of Elapsed Time**
- **Proof of Burn**
- **Proof of Capacity**
- **Raft**
-

8.x.y.z. Cloud-Computing

Scale up
Hoch-Skalieren
Anschaffung größerer und Leistungs-fähigerer Hardware (Rechner, Speicher, …)
hohe Anschaffungs-Kosten
ev. häufige Wechsel der Hardware notwendig (relativ unflexibel)
umfangreiche Veränderungen der Software oft ebenfalls notwendig (vieles Spezial-Systeme)

Scale out
Hinaus-Skalieren
Anschaffung weiterer (gleichartiger) Hardware (Rechner, Speicher, …)
meist preiswerter, deutlich flexibler
Anschaffung genau nach bedarf notwendig, Hardware muss nicht ständig erneuert werden
Software leicht anpassbar oder Standard-Systeme

Scale in
Hinein-Skalieren
effektiveres Nutzen der vorhandenen Hardware z.B. durch Virtualisierungen (Simulation mehrerer Rechner auf einer Hardware (meist schon so Lasten-Ausgleich möglich)
Hardware (Speicher, Prozessor) wird optimaler ausgenutzt

z.B. auch gemeinsame Nutzung eines (angeschaften) Datenbank-Management-Systems (nur 1x Kosten) für mehrere Kunden (die haben eigene Tabellen, …, ohne Kontakt zu den Daten der anderen Nutzer)

Techniken lassen sich praktisch einzeln und mehrfach kombinieren

Abstraktionen

Legacy-Systeme
geringe Flexibiltät, geringe Virtualisierung (Art der Abstraktion)
Rechenzentren mit spezieller Hard- und Software
Hardware-orientiert

Virtualisierung
mittlere Flexibiltät, mittlere Virtualisierung
Client-/Server-Virtualisierung, für Desktop-Systeme
mehrere / viele Anwendungen auf einem System (Multi-Tasking)
Software-orientiert

Cloud-Virtualisierung
hohe Flexibiltät, hohe Virtualisierung
für Web-Anwendungen
Service- / Dienst-orientiert

Eigenschaften von Cloud-Systemen:

- on-demand access (Zugriff nur auf Bedarf)
- Zugriff unabghängig vom Standort / globaler Zugriff möglich
- Anbieter können Ressourcen sehr flexibel und breit gefächert nutzen
- hohe Elastizität (bei steigenden Anforderungen lässt sich schnell mehr von dem Dienst buchen / Bedarfs-Anpassung)
- wirtschaftlich (meist nur das bezahlen, was man wirklich nutzt)
    - bessere Auslastungen
    - spezialisiertes und professionelleres Personal kann eingestellt werden
    - Strom-Kosten (Verluste bei Strom-Transport teurer, als Verlust-Aufgleich bei Daten)
    - …
- …

off promise cloud
ein separater Cloud-Anbieter mit vielen verschiedenen Nutzern

on promise cloud
Cloud-Struktur innerhalb eines großen Unternehmens
Anschaffung eines Rechenzentrums / Servers für viele / alle Abteilungen

fork-computing
erweitert die Cloud-Systeme um die Leistungsfähigkeiten der Client's zu einer Gesamt-Leistung

Everythink as a Service

Infrastructure as a Service (IaaS)
Cloud-Anbieter stellt nur die Infrastruktur (Hardware, Leitungen, …) zur Verfügung
Nutzer kann die Cloud völlig frei nutzen, muss sie selbst administrieren, pflegen, …
eigene Betriebssysteme, Anwendungen, … möglich

Platform as a Service (PaaS)
Cloud-Anbieter stellt Hardware z.B. mit passendem Betriebssystem bereit
meist wird ein vorinstalliertes, vorkonfiguriertes System angeboten, mit dem der Nutzer wieder machen kann, was er will
eigene Anwendungen möglich

z.B. Server-Hosting bei Strato, …

Software as a Service (SaaS)
völlig vorgefertigtes System mit Anwendungen
Anwendung steht im Vordergrund
Nutzer merkt nichts mehr vom Betriebssystem (meist sehr kosten-günstige ausgewählt (z.B. Linux))
Nutzer verwendet – meist über den Browser – Standard-Software (Textverarbeitung, Tabellenkalkulation, Präsentations-Software, eMailing, Kalender, Aufgaben-Planung, Kontakte, …)

z.B.: google-office, web-Mailing, Prezi, Kahoot, …

8.x.y. Daten-Aufbereitung

8.x.y.z. Informations-Qualität

einfachste Definition für Qualität ist "die Eignung für den Gebrauch"

> Auch wenn wir Qualität nicht definieren können, wissen Sie schon, was gemeint ist.
> Robert PIRSIG
> (Even though quality cannot be defined, you know what it is.)

Komplizierteste Begriffs-Bestimmung listet 179 Dimensionen für den Begriff auf.
Eine recht praktische Merkmals-Liste zeigt die folgenden (15) Dimensionen / Kriterien für Informations-Qualität auf:

- Exaktheit (Accuracy)
- Objektivität (Objectivity)
- (Believability)
- Reputation (Reputation)
- Konsistenz (Consistency)
- Sicherheit (Security)
- Relevanz (Relevance)
- Aktualität (Timeliness)
- Wertschöpfung (Value-Added)
- Vollständigkeit (Completeness)
- (angemessener) Umfang (Amount of Data)
- eindeutige Auslegbarkeit (Interpretability)
- Verständlichkeit (Understandability)
- Zugänglichkeit (Accessibility)
- Übersichtlichkeit (Concise Representation)

Definition(en):

8.x.y.z. Dimensionen der Daten-Qualität

Kategorien der Informations-Qualität und ihre Dimensionen

- **inhärente (Dimensionen)** ➔ **Inhalt**	- (hohes) Ansehen - Fehlerfreiheit - Objektivität - Glaubwürdigkeit
- **system-unterstützende (D.)** ➔ **System**	- Zugänglichkeit - Bearbeitbarkeit
- **darstellungs-bezogene (D.)** ➔ **Darstellung**	- Verständlichkeit - Übersichtlichkeit - einheitliche Darstellung - eindeutige Auslegbarkeit
- **zweck-abhängige (D.)** ➔ **Nutzung**	- Aktualität - Wertschöpfung - Vollständigkeit - angemessener Umgang - Relevanz

nach: Deutsche Gesellschaft für Informationsqualität (dgiq)

Vollständigkeit und Fehlerfreiheit sind die für informatische Zwecke besonders wichtigen Dimensionen der Daten-Qualität

8.x.y.z.1. System-Unterstützung

Zugänglichkiet (Accessibility)

Hier steht die Frage im Raum, ob man auf einfache / zumutbare Art und Weise an die Daten herkommen kann.
Probleme können z.B. durch Schutz-Interessen von Firmen oder Daten-Umstrukturierungen (neues Warenwirtschafts-System) auftreten

Bearbeitbarkeit (Ease of manipulation)

Daten müssen in Formaten zur verfügung stehen, die dem Inhalt entsprechen und den Zweck der Nutzung unterstützen
Negativ-Beispiele: Email-Adressen als JEPG-Datei, Datum's-Formate

8.x.y.z.2. Inhärität

8.x.y.z.3. Darstellungs-Bezug

Kann ein Mensch die Daten anschauen und mit ihnen (praktikabel) weiter arbeiten?

Verständlichkeit (Understandabilty)

Daten in passenden Datentypen (Wohnort als Name statt als GPS-Daten)
keine lesbaren Produkt-Namen, stattdessen nur Nummern oder (kryptische) Kürzel
hilft beim Identitifizieren von Fehler-Quellen

Übersichtlichkeit (Concise representation)

möglichst kurze / knappe Darstellung der Daten
zu viele sowie unnötige Daten verschlechtern die Übersichtlichkeit

einheitliche Darstellung (Consistent representation)

besonders bei der Zusammenführung von Daten ein Problem, da kommen verschiedenste Zahlen- Währungs- und Datum-/Zeit-Formate daher
z.B. werden Geschlechter völlig unterschiedliche erfasst, gespeichert und dargestellt (w, m, d; weiblich: ja oder nein; Anreden; …)
unterschiedliche Einheiten-Systeme (SI oder cgs) sowie unterschiedliche Einheiten (°C und K) oder auch Zahlen-Dimensionen für die Einheiten (mg oder g) sorgen hier für unbedingt zu beachtende und wahrscheinlich zu korrigierende schlechte Daten-Qualität

eindeutige Auslegbarkeit (Interpretability)

der hinter den gespeicherten Daten (Zahlen, …) steckende Inhalt mus eindeutig sein
21 kann alles sein: Temperatur, Anzahl, Uhrzeit oder Laufzeit (z.B. Stunde oder eben Stunden)

z.B. kann in einer Tabelle "Kunden" völlig verschiedene Arten von Kunden erfasst werden, die sind u.U. nicht gleichwertig für eine nachfolgende Bearbeitung / Auswertung
News-Letter-Leser sind anders, als Kauf-Kunden oder Support-Kunden oder Produkt-Interessierte zu verarbeiten und zu verstehen

z.B. auch andere Begrifflichkeiten sehr unterschiedlich auslegbar (vor allem bei internationalen Daten)
z.B. werktags, …
hier hilft nur eindeutige Begriffs-Bestimmung oder die Festlegung von Grenzen

8.x.y.z.4. Zweck-Abhängigkeit

die Qualität der Daten wird erst über die Nutzung / Nutzbarkeit bestimmt

Aktualität (Timeliness)

stehen die Daten rechtzeitig und schnell genug für die weitere Bearbeitung zur Verfügung
z.B. Aktien- oder Wechsel-Kurse

Wertschöpfung (Value-added)

kann man aus den Daten und ihrer Auswertung einen (neuen / zusätzlichen) Wert schöpfen

Vollständigkeit (Completeness)

sind die Daten schon vorbereitet / gefiltert worden
sind Daten von bestimmten Kategorien schon entfernt worden

angemessener Umfang (Appropriate amount of data)

liegen überhaupt genügend Daten vor (z.B. für komplexere Statistiken) oder sind Datenmengen zu groß (für ein bestimmtes Verfahren / einen bestimmten Algorithmus)
handelt es sich um eine ausgewogene Teilmenge z.B. für Verfahren des maschinellen Lernen's
es können aber auch relevante Daten für eine (bestimmte) Analyse fehlen

Relevanz (Relevancy)

werden die Daten für das Verfahren wirklich gebraucht
müssen die Daten gespeichert / verarbeitet werden
sind die Daten für den Zweck notwendig
unpassende Daten-Genauigkeit (z.B. zu viele oder zu wenige Nachkomma-Stellen bei Messwerten)

8.x.y.z. Auswirkungen schlechter Qualität

Die Qualität einer Analyse kann maximal so gut sein, wie die Qualität der eingenenden Daten.

> Wenn man Müll hinein tut,
> dann bekommt man auch nur Müll hinaus.
> (Garbage in, garbage out.)

fehlerhafte Preise im Einzelhandel (80 % der Barcode-Fehler gehen zu Lasten der Verbraucher)
nicht richtig eingearbeitet Angebots-Preise, …

50 – 80 % der Einträge im US-Strafregistern falsch, ungenau oder fehlerhaft
nicht eingehaltene Lösch-Termine

rund 7 % der Massenpost-Sendungen nicht zustellbar, weil die Adressdaten nicht exakt waren

anekdotisches Beispiel: in Irland hat der Autofahrer Prawo Jazdy Unmengen von Ticket's bekommen, die konnten aber niemanden zugeordnet werden
heraus kam, dass hier von der Polizei nicht der Name sonder die polnische Bezeichnung für Führerschein erfasst wurde

Rechnungen (z.B. von Telefon-Anbietern) oder Überweisungen (z.B. Arbeitsamt) in Millionen-Höhe

Folge können rechtlich, ökonomisch, … sein
zusätzlich schnell Image-Schaden wegen schlechter Presse

Beispiele für niedrige Daten-Qalität

8.x.y.z. Daten-Vandalismus

absichtliche Daten-Manipulation, um bestimmte Effekte zu erreichen
häufigste Form ist das Löschen von Daten

politisch motivierte Manipulation

8.x.y.z. Messen der Daten-Qualität

nur gegen die Realität zu prüfen
praktisch sehr aufwändig

mögliche Qualitäts-Kriterien

- beim Subjekt
  - Relevanz
  - Glaubwürdigkeit
  - Verständlichkeit
  - …
- im Prozess
  - Verfügbarkeit
  - Antwort-Zeiten
  - …
- beim (Daten-)Objekt
  - Verfügbarkeit
  - Vollständigkeit (bezogen auf Gesamtheit)
  - fehlende Attribute / Daten-Dichte
  - Aktualität
  - …

8.x.y.z. Daten-Aufbereitung

Data preparation
Daten-Vorbereitung, -Reinigung, -Aufbereitung, -Präparation
altes Problem
Vorrang hat die Umsetzung der Roh-Daten in eine Form, in der sie weiter verarbeitet werden können

typische Probleme:

- Erkennen des Zeilen-Endes
- Erkennen des Endes eines Attributes (Feld-Trenner) od.ä.
- unterschiedliche Zeichen-Sätze (fehlerhaft umcodierte Sonderzeichen, Umlaute, …)
- unterschiedliche Schreibweisen von Zeitangaben (besonders Datum)
- einzelne Fehler in Datensätzen von großen Daten-Beständen
- fehlende Atrribute (die aber für die weitere Verwendung notwendig sind)

viel händischer / Personal-Aufwand
teilweise automatisierbar
heutige Daten-Wissenschaftler (data scientist's) beschäftigen sich zu rund:

- 60 % mit dem Daten-Reinigen und –Strukturieren
- 19 % mit der Daten-Beschaffung
- 9 % Suche, Bearbeitung, Korrektur fehlender Daten(-Teile)
- 4 % Verbessern, Anwenden von Algorithmen
- 5 % anderes

fast mit gleichen Anteilen werden die unbeliebten Tätigkeiten beschrieben:

- 57 % Daten-Reinigen und –Strukturieren
- 21 % Daten-Beschaffung

- 10 % Herstellen von Trainings-Daten für das Maschinelle Lernen
- 4 % Verbessern, Anwenden von Algorithmen
- 3 % Suche, Bearbeitung, Korrektur fehlender Daten(-Teile)
- 5 % anderes

Präparation auf Datei-Ebene

alt ASCII (auf DOS-Ebene weitgehend standardisiert)

gute – alte – und weit verbreitete Standrds für Zeichen sind UTF-8 und UTF-16

modern Unicode
aktuelle Version 12.1 beinhaltet rund 138'000 Zeichen und unterstützt 150 Sprach-Systeme
auf Windows-Rechnern i.A. über Eingabe als 4-Ziffern-Code bei gedrückter ALT-Taste (Zeichen erscheint erst nach dem Freigeben der Alt-Taste)
in alten Windows.System und DOS reicht die Eingabe als 3-Ziffern-Code (erzeugt Zeichen aus der Code-Page 850)

ungünstige (aber sehr beliebte) Austausch-Formate / Daten-Layout's:
z.B. EXCEL: mehrere Tabellen auf einem Tabellen-Blatt

Präparation auf Werte-Ebene

falsch oder schlecht formatierte Daten
- z.B. pseudo-markierte Überschriften / Hinweis-Texte mit Feld-Trennern
- z.B. Überschriften

fehlerhafte Werte (leeres Feld oder NULL oder Null oder WAHR oder FALSCH oder …)

Erkennen von Feld-Trennern und gleich-artigen Normalzeichen
z.B. Komma's als Feld-Trenner und Kommata in Texten (, die eigentlich durch Anführungsstriche umschlossen sind)
Erkennung von Anführungs-Strichen innerhalb von Texten

gleiche Bedeutungen von verschiedenen Zeichen:
z.B. tiefe und hohe Anführungs-Zeichen,

unterschiedliche Bedeutung gleicher Zeichen:
z.B. Anführungs-Zeichen bei Betonungen, wörtlicher Rede oder Ironie

Daten-Reinigung

Daten-Reinigung mit externen Quellen

Nachschlagen von (Wohn-)Adressen in zentralen Datenbanken (z.B. bei der Deutschen Post)

ähnliches für Produkte / Bauelemente / … möglich

Geocoding
Umwandlung einer Adresse in Geo-Daten, meist verbunden mit einer einheitlichen / standardisierten Darstellung

Nutzung von Wissens-Datenbanken (Knowledge Database's)
"Welt-"Wissen ist in strukturierter Form vorhanden, Abgleich mit eigenen Daten möglich
auch Anreicherung der Daten oder Ergänzung (fehlender Einträge)

teilweise auch möglich:
wikipedia, wikidata

Lösung des Matching-Problem's
Fehlerhafte Schreibweisen, überzählige Leerzeichen, Umsetzung von Umlauten, …

Imputation

Ergänzung fiktiver Werte an Fehl-Positionen
ev. Benutzung des Mittelwertes, Median (in der Mitte liegender Wert) oder Modus (häufigster Wert), um Statistiken möglichst wenig zu beeinflussen

problematisch bei vielen fehlenden Daten

Daten-Reinigung mittels Abhängigkeiten

Prüfen der Einhaltung von bestimmten Daten-Eigenschaften
z.B. einmalige Vergabe eines Schlüssel's in einer Daten-Tabelle
ev. korrigierbar durch Vergabe eines neuen Schlüssel's für einen Datensatz (vorher ev. Duplikats-Prüfung notwendig)
Problem bei Verwendung in anderen Tabellen (als Fremd-Schlüssel)

funktionalle Abhängigkeit
interne Bedingungen für Daten
nur einmaliges Vorkommen von bestimmten Attributen (Rangfolge, Teilnehmer bei einem Wettbewerb, …)
Problem die Fehler in den anderen Attributen zu finden
ev. Gegen-Prüfen der Daten auf Plausibilität
Ziel bei unterschiedlichen Fehlern bezüglich bestimmter Daten durch passende Korrektur auf weniger Fehler zu kommen

Beispiel (aus HPI-Kurs):
Bedingung: jeder Sportler soll nu in einer Sport-Art geführt werden

Name	Sport-Art	Distanz	Zeit	Medallie
Michael Phelps	Schwimmen	100 m	49 s	Gold
Usain Bolt	Schwimmen	100 m	11 s	Siber
Usain Bolt	Sprint	200 m	19 s	Gold

für uns offensicht, dass Usain Bolt wohl nicht beim Schwimmen dabei war
für Daten-Verarbeitungs-System aber kein klarer Fakt
Usain Bolt könnte der Name von zwei Sportlern sein
es könnte Sprint oder Schwimmen falsch sein

Gegen-Prüfen mit weiteren Daten und Abhängigkeiten
Wenn Usain Bolt beim Schwimmen über die Distanz von 100 m und Siber gewonnen hat und dabei weniger Zeit gebraucht hat, als der Gold-Gewinner (Michael Phelps), dann ergibt sich ein weiteres Problem.
Wird dagegen das Schwimmen als Fehler erkannt, lösen sich die Probleme auf. Der richtige Eintrag muss ev. nachrecherchiert werden. Die Bedingung der Datenbank ist erfüllt und die Inkonsistenzen der Daten entfallen.

weitere Abhängigkeiten:
- Matching
- Inklusion

Duplikate

typisch:
verschieden geschriebene Adressen für eine Person und eigentlich nur einer Adresse
doppelte Rechnungen (bei verschiedenen Rechnungs-Stellern in einem Dienstleistungs-Unternehmen; doppelte Daten-Übertragungen)

Prüfen, ob zwei Datensätze usw. in einer Sach-Tabelle nur zu einem Realwelt-Objekt gehören und den gleichen Sachverhalt beschreiben
→ Bestimmung der Ähnlichkeit

z.B. Sprachen-Verlinkung bei Wikipedia
Ausgangs-Punkt z.B. "Piotr"

Problem der großen Daten-Mengen / große Komplexität
z.B. Finden von graphischen Duplikaten
praktisch ist Vergleich jedes Objektes mit allen anderen notwendig

Aufgaben:

1. ***Gehen Sie zur Wikipedia-Seite "Poitr" und welchseln Sie dann systematisch die Sprachen (englisch, russisch, spanisch, französisch, italienisch) durch (ev. wieder zum Anfang zurückgehen)! Notieren Sie die Sprache und den angesprungenen Begriff! Stellen Sie den Zusammenhang als Graph dar! Wechseln Sie ausgehend vom angesprungenen Begriff wieder weiter die gleichen Sprachen (jetzt auch deutsch) und erweitern Sie den Graphen!***
2. ***Welche Probleme ergeben sich bei solchen "Übersetzungen"?***
3. ***Erstellen Sie einen Begriffs-Graphen für die Musik-Richtung "Easy Listening" oder einer "Joint Stock Company" in den verschiedensprachigen Wikipedia-Versionen!***
4. ***Wählen Sie einen Begriff, für den Sie Begriff-Verschiedenheiten in unterschiedlichen Sprachen kennen und erstellen Sie einen Begriffs-Graphen, wie oben beschrieben!***
5.

Duplikat-Erkennung

Ähnlichkeit

Suche nach einem Wert, der die Ähnlichkeit beschreibt
Prüfen des Ähnlichkeits-Wert's an einem Schwellen-Wert

Edit-Distanz
minimale Anzahl von Editier-Operationen, um das eine Wort in das andere umzuwandeln

- Austausch eines Zeichens ist eine Operation
- Löschen eines Zeichens ist eine Operation
- Einfügen eines Zeichens ist eine Operation

Problem bei Namens-Änderungen nach Heirat

unterschiedlich bei verschieden häufigen Begriffen / Namen / …
bei seltenen Begriffen / Namen / … ist es eher wahrscheinlich, dass es sich um ein und das selbe Objekt (der Realwelt) handelt, auch wenn sich viele Attribute unterscheiden
bei häufigen Begriffen / Namen / … ist die Ähnlichkeit / Übereinstimmung nur dann gegegeben, wenn viele / alle anderen Attribute übereinstimmen

Ähnlichkeit beim Klang
z.B. für Namen (Meier, Meyer, Maier, Mayer, Mayr, …)

Anzahl gemeinsamer Worte in Sätzen / Texten
Wort-Überlappungen

Beziehungs-basierte Ähnlichkeit

benutzen anderer Datenbestände (Tabellen), um Ähnlichkeiten zu finden
Artikel-Tabelle enthält mehrere zu prüfende Datensätze
für jeden Artikel gibt es in einer anderen Datenbank / Tabelle eine Stück-Liste, dann kann auch über gleiche oder sehr ähnliche Stück-Listen eine Ähnlichkeit der betrachteten Artikel gefunden werden

LEVENSHTEIN-Distanz

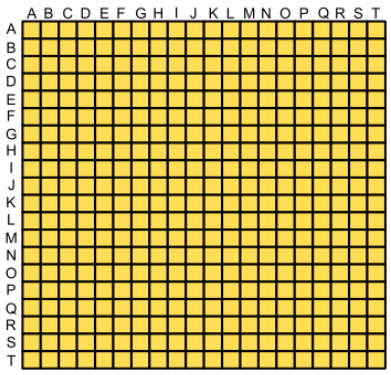
moderne Klassifikation über maschinelles Lernen
Training mit bekannten Duplikaten

Link:
https://phiresky.github.io/levenshtein-demo/ (online-Rechner)

Algorithmen

da die Daten-Menge sehr groß, muss bei den Algorithmen sehr effektiv gearbeitet werden
da jeder zusätzliche Arbeits-Schritt eben tausend- oder millionen-fach ausgeführt werden muss

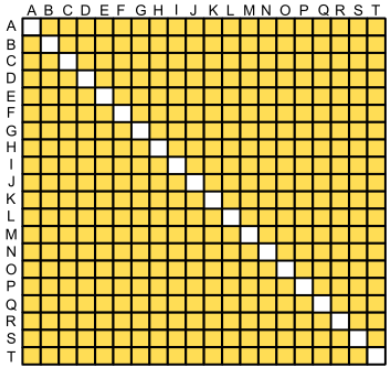
Vorauswahl von Kandidaten, dann genauere Prüfung der Kandidaten



Einsparung bei Bedachtung bestimmter Eigenschaften und Bedingungen

angenommen (nur) 20 Datensätze sollen auf Ähnlichkeit geprüft werden

für schlechten Algorithmus wären 20 x 20 = 400 Vergleiche notwendig (da hier Datensätze betrachtet werden, sind intern immer noch die Attribute einzeln zu vergleichen)



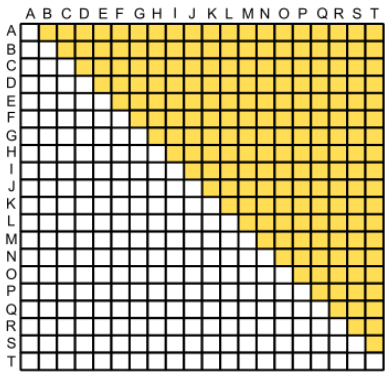
Daten-Satz sowie jedes Daten-Attribut muss nicht mit sich selbst getestet werden

damit spart man auf 20 Datensätzen schon mal 20 Vergleiche
macht 5 % Einsparung aus

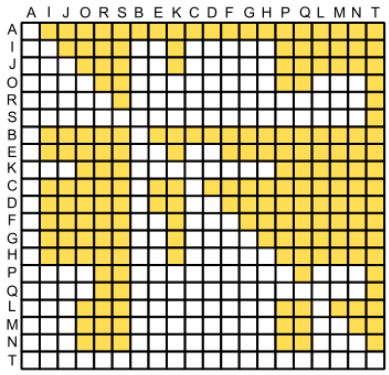
i.A. gilt die Symmetrie der Ähnlichkeit (wenn A ähnlich zu B ist, dann ist (sehr sehr wahrscheinlich) auch B ähnlich zu A

diese Annahme erspart uns die Hälfte der übriggebliebenen Vergleich

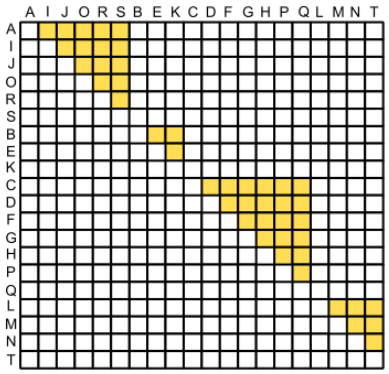
nur noch 380 / 2 = 190



Aufteilung der Datensätze in Gruppen (Partitionierung) → Vergleiche nur innerhalb der Partitionen (Gruppen)
Kriterien der Partitionierung müssen so gewählt werden, dass passend große Gruppen gebildet werden, aber Duplikate nicht übersehen werden können
z.B. Aufteilung eines Landes nach Postleitzahlen (erste oder die ersten beiden Ziffern)



innerhalb der Partitionen (hier durch den Alphabet-Bruch erkennbar) werden die gleichen Verbesserungen der Algorithmen, wie oben vorgenommen, so dass nur noch ein kleiner Teil von Vergleichen übrig bleibt
hier (wegen der Gruppierung) nur noch 15 + 3 + 21 + 6 = 45 Vergleiche



im Beispiel also Reduktion auf fast ein Zentel der ursprünglich notwendigen Vergleiche

→ Findet aber nicht Probleme bei der PLZ
dann nutzt man zwei oder drei weitere Partitionierungen nach anderen Kriterien
tauchen dann mögliche Duplikate in mehreren Ergebnis-Listen auf, dann sind die Duplikate ziemlich sicher und man findet auch weniger wahrscheinliche, ganz spezielle Duplikate

Daten-Fusion

Problem nach dem Finden der Duplikate ist das Löschen
Welcher Datensatz soll erhalten bleiben? Welche können – ohne Daten-Verlust (!) – gelöscht werden?

Möglichkeit ist die Kombination von Duplikaten zu einem reichhaltigeren Datensatz

Kriterien können sein:

- gleicher Attribut-Wert (können gefahrlos übernommen werden)
- größere Länge (meist ist das längere Attribut das mit Information)
- leeres Attribut kann durch ein gefülltes ersetzt / ergänzt werden (Zusammentragen von Ingformationen)
- Nutzung von Minimum, Maximum, Durchschnitt, …
- Anhängen nicht zuordbarer Attribute / Inhalte
- Aktualität (neueres Datum)
- Vertrauenswürdigkeit
- …

ev. nehält man alle Datensätze / Duplikate

macht einen (den informtivsten) zum Haupt- oder Kopf-Datensatz und die (anderen) Duplikate werden als Neben-Datensätze zugeordnet
damit werden z.B. Bestell-Historien erhalten

8.x.y. Informations-Integration

gemeint sind die Probleme, die sich unterschiedliche Strukturen der Daten ergeben

Welche Schwierigkeiten gibt es mit der Informations-Intergration?

Datenbank sollten eigentlich die Lösung des Problem's der vielen Daten in speziellen, dezentralen, individuellen, proprietären Dateien sein
viele Daten-Formate
Ziel war zentrale Datensammlung für alle (Abteilungen / Nutzer / …) mit hoher Aktualität und geringer Redundanz
neue Daten-Typen (Graphiken, Geodaten, …)

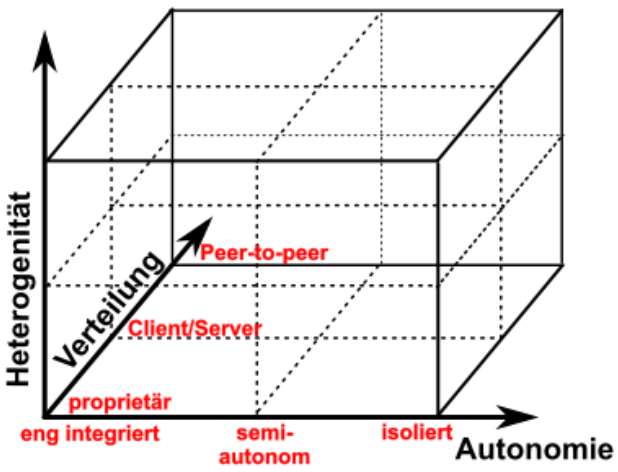
Problem-Felder

- **System-Probleme** — technische Aspekte
  Datei-Formate; Betriebssysteme mit unterschiedlichen Datei-Strukturen
- **soziale Probleme** — inhärenter Widerwille von Mitarbeitern / Abteilungen / …
  ihre Daten für eine breite Verwendung (an andere Mitarbeiter / in anderen Abteilungen / …) bereitzustellen
- **logische Probleme** — unterschiedliche Daten-Modelle
  unterschiedliche Modellierung von Daten
  verschiedene Definitionen von Dingen / Prozessen / …

8.x.y.z. Autonomie und Heterogenität von Daten-Quellen

Autonomie ist hier bezüglich der Eigenständigkeit der verschiedenen Systeme gemeint
sie werden eigenverantwortlich gepflegt, aktualisiert und ev. auch abgeschaltet
Daten-Strukturen können sich ändern, ev. kommen Erweiterungen dazu

daraus folgt Heterogenität der Implementierung, Design der Daten, Daten-Qualität, Zugriffs-Rechten, …



enge Integration
hier wird die Autonomie aufgegeben, um eine breite Integration in verschiedene eigene und fremde Systeme und Anwendungen zu ermöglichen
der Daten-Eigner stellt eine offene Schnittstelle zu seinen Daten bereit, gibt ev. Informationen zum Daten-Modell bekannt, modelliert seine Daten ev. auch um
Ergebnis ist i.A. eine Datenbank mit einem höhreren Mehrwert

semiautonom
der Eigner erlaubt den Zugriff auf seine Daten
behält sich aber Änderungen an seinem Modell, den Daten-Strukturen usw. usf. vor

isoliert
Daten-Quelle hat keine Kenntnis darüber, dass sie Teil einer Integration ist
z.B. irgendwelche Web-Daten, sie stehen einfac so zur Verfügung und werden – ev. auch ohne Kenntnis des Eigners – in andere Projekte eingebunden

Client-Server-Verteilung
Daten stehen auf einem System (Server) zur Verfügung und können von beliebig vielen anderen System (Client's) genutzt / heruntergeladen werden

Peer-to-peer
hier werden die Daten gleichmäßig im Netz auf verschiedenen – gleichberechtigten – System gehalten, die sich in irgendeiner Form gegenseitig aktualisieren

Arten der Heterogenität

- **syntaktisch** betrifft Formate, Codierungen, …
- **strukturell** betroffen siend hier die Schemata, Modelle, …
- **semantisch** Bedeutung / Interpretation von Daten

syntaktische Heterogenität
Hardware-Heterogenität: technische Aspekte (verbaute Technik, Übertragungungs- und Übersetzungs-Komponenten)
Software-Heterogenität: unterschiedliche Betriebssysteme mit z.B. unterschiedlichem Handling von Dateien, Ordnern usw.
verschiedene Protokolle und deren Umsetzung
Software-Versionen
Schnittstellen-Heterogenität: Zugriff auf verschiedene Felder; Implementierungen von Logiken
Logik-Verständnis von Formularen
gebundene Felder (Felder die unbedingt ausgefüllt werden müssen)

strukturelle Heterogenität
unterschiedliche Darstellung eigentlich gleicher Daten:
z.B.: Personen und Geschlechts-Charakterisierung
Person(ID, Nachname, Vorname, männlich, weiblich)
Person(ID, Nachname, Vorname, Geschlecht)
Männer(ID, Nachname, Vorname) ; Frauen(ID, Nachname, Vorname)

Beispiel (einer Analyse des Objektes "Company" in Info-Boxen bei wikipedia)
es wurden von Nutzern 1083 unterschiedliche Attribute zugeordnet, davon kamen 499 nur einmalig vor
in 39 Atrribut-Bezeichnungen kam der Teilstring "name" vor
für 273 Unternehmen war aber nicht einmal etwas bei enem "name"n eingetragen

→ Probleme!
Welches Attribut kennzeichnet nun den "echten" Firmen-Namen?
Was passiert, wenn in vielen Feldern etwas steht? Welches Feld hat Vorrang?
wass passiert, wenn gar kein "name"-Feld ausgefüllt ist?
…
Bei welcher Verwendungs-Zahl beschneidet man die Anzahl von Attributen?
Was passiert mit neuen Attributen, die zuerst ev. nur kleine Vertreterzahlen haben?
…

in Datenbanken gibt es einen ausgeprägten Hang zum Chaos bei Schemata

Schema-Mißbrauch über Homonyme (gleiche Begriffe mit / für unterschiedliche Bedeutungen)

semantische Heterogenität
mit den größten Problemen verbunden, da Computer hier die wenigste Unterstützung gewähren können
Computer können zwar Begriffe erkennen, aber deren Bedeutung kaum
vielfach Bedeutung nur aus dem Sachzusammenhang erschließbar
oft erweiterte Interpretation notwendig

Beispiel:
Andrea und Gina stehen lange mit ihrem Auto am Fluß.
Am nächsten Tag schaut er allein von der anderen Seite zurück.

8.x.y.z. Schema Matching

ist die Verknüpfung unterschiedlicher Daten-Schemata /-Modelle
Probleme durch unterschiedliche Sprachen(first name, Vorname) oder Homonyme usw. usf.
Ergebnis ist i:A. eine Zurdnungs-Tabelle mit Attributs-Paaren (Korrespondenzen) aus beiden Schemata

die Tabellen / Attribute des einen Schema's werden Tabellen / Attributen in einem anderen Schema zugeordnet
ev. händisch
moderne Werkzeuge funktionieren graphisch orientiert, ev. mit Umcodierungen
Hochleistungs-Produkte versuchen das Matching automatisch → erstellen Vorschlag
benutzen gleiche Attributs-Bezeichnungen, übersetzte Attributs-Namen, Vergleiche über die Atrributs-Werte (semantische Vergleiche)

funktioniert besonders gut bei bekannten Duplikaten

eigentliche Daten-Übertragung erledigt das Schema Mapping

8.x.y.z. Schema Mapping

ist das Verfahren der Übertragung und Anpassung der Daten nach dem Matching Schema notwendig bei unterschiedlichen Formaten, Codierungen

8.x.y.z. Materialisierte Integration

das Ergebnis einer Daten-Integration – also eine Datenbank nach dem Schema-Mapping – wird bei der materialisierten Integration lokal gespeichert
die Arbeiten mit dem integrierten Daten passiert ausschließlich auf den zwischen-gespeicherten Daten

die zwischen-gespeicherte Datenbank wird als materialisiertes Integrations-Produkt verstanden (es ist zusätzlicher Speicher notwendig)

Probleme:

- da nicht kontinuierlich intehriert wird, können lokale Daten beraltert sein
- zusätzlicher Speicher-Aufwand
- zusätzlicher Bearbeitungs- / Transformations-Aufwand
- häufig bietet die Quelle nicht den gesamten Datenbestand (zum downloaden) an

8.x.y.z.1. Data Warehouses

entstanden aus der Herausforderung komplexer Analysen über die Daten
außerdem sollen laufende (online-)Daten-Verarbeitungen nicht durch die Analyse-Verfahren gestört werden

praktisch Daten-Lager mit großer Angebots-Fläche
ist ein materialisertes Integrations-Ergebnis
Daten sind i.A. bereinigte Daten in einer für viele Analysen passenden lockeren / flachen Struktur
ev. werden ETL-Prozesse angeschoben
es sind meis riesige Daten-Mengen vorhanden, da man aber im Voraus noch nicht weiss, welche Anfragen kommen werden, werden alle Daten verfügbar gehalten

Problem der Aktualität kann durch geschickte Aktualisierungs-Zeiten reduziert werden

technisch gesehen sind es relationale Datenbanken mit SQL-Zugriff
meist Integration mehrerer unabhängiger Daten-Quellen zu einem großen Daten-Bestand, die als Big data charakterisiert werden kann

nach der Daten-Aufbereitung werden die Daten mehr-dimensional strukturiert, um möglichst kurze Zugriffe auf sie zu haben (und nicht mehrfach geschachtelte Fremdschlüssel, wie in "normalen" relationalen Datenbanken üblich)
Veranschaulicht durch Würfel (Data cube, OLAP-Würfel, Daten-Würfel), praktisch aber viel mehr Dimensionen

Daten werden häufig in ein Starschema gebracht und gespeichert

im Zentrum des Sterns liegt eine Basis-Fakten-Tabelle und darum herum sind diverse Detail-Tabellen angeordnet

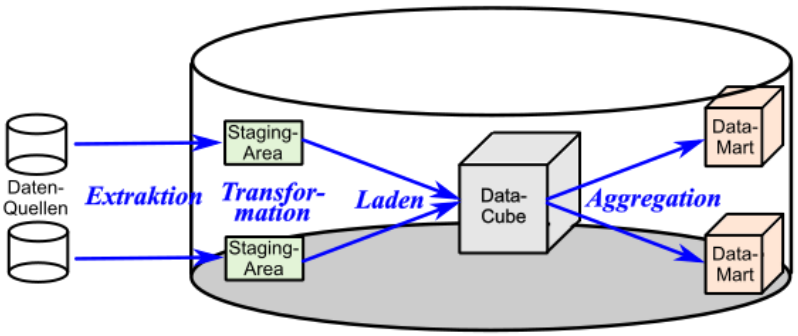
Zugriff auf den großen mehr-dimensionalen Datenbestand durch Slicing und Dising (Herauschneiden von Scheiben / Schichten / kleineren Würfeln aus dem Data Warehouse "Würfel")
Data Warehouse bietet ev. kleine Data cube's an, die einige spezielle Dimensionen (Atrribute) umfassen oder nach Nutzungs-Zweck zusammengestellt wurden
heißen Data mart (Daten-Markt)
in diesen sind kleinere, schnellere, effektivere Analysen möglich

weiter mit 5-10

8.x.y.z. ETL-Prozesse – Extract-Transform-Load

Prozess der Materialisierung der Daten während der Integration

Prozess der Umsetzung der externen Daten in den internen Daten-Bestand (Data-Cube).



Extraktion beschreibt alle Prozesse des Selektierens und Herunterladens der externen Daten von ausgewählten Daten-Quellen
hier muss mit verschiedenen Daten-Strukturen, unterschiedliche Schnittstellen und sehr großen Daten-Mengen gerechnet werden

die Transformation umfasst alle Vorgänge der Daten-Bereinigung und –Strukturierung der heruntergeladenen Daten-Bestände
u.U. auch Filtern, Überprüfen von Daten, …
dazu stellt die Staging Area viele verschiedene Tool's zur Verfügung

diese transformierten Daten werden dann im Lade-Prozess in die interne Datenbank (Ziel-Datenbank) eingespeist Dieser Daten-Bestand wird wegen seinen flachen Struktur aus vielen Dimensionen Daten-Kubus (Data-Cube) genannt

damit sind die eigentlichen ETL-Prozesse beendet und es kann sich noch Aggregations-Prozesse anschließen, die zu kleineren – besser handhabbaren – Daten-Beständen , den sogenannten Daten-Märkten (Data-Mart) führen
hierfür wird vorrangig auf Slicing und Dising ((in Scheiben) schneiden und Würfeln) zurückgegriffen

8.x.y.z. Business Intelligence - BI

Geschäfts-Intelligenz
für
im Unterschied zum Data Mining geht es beim BI mehr um die Nach-Verfolgung und Auswertung der Daten ("nackte" Analyse, vielfach sogar ohne Bewertungen)
Data Mining wird in diesem Zusammenhang eher mit dem Finden neuer Zusammenhänge usw. in verbindung gebracht
sachlich, technisch und informatisch sind BI und DM aber dicht verwandt

statistische Auswertung und Präsentation der Daten; Erstellen von Berichten
umfangreiche Klassifizierungen und Differenzierungen
Trends finden und darstellen
Personal-Analysen (z.B.: Wer verkauft wievel?)
Herstellen neuer Zusammenhänge
zunehmende Aktualität

8.x.y.z. Data Lake's / Data Reservoir's

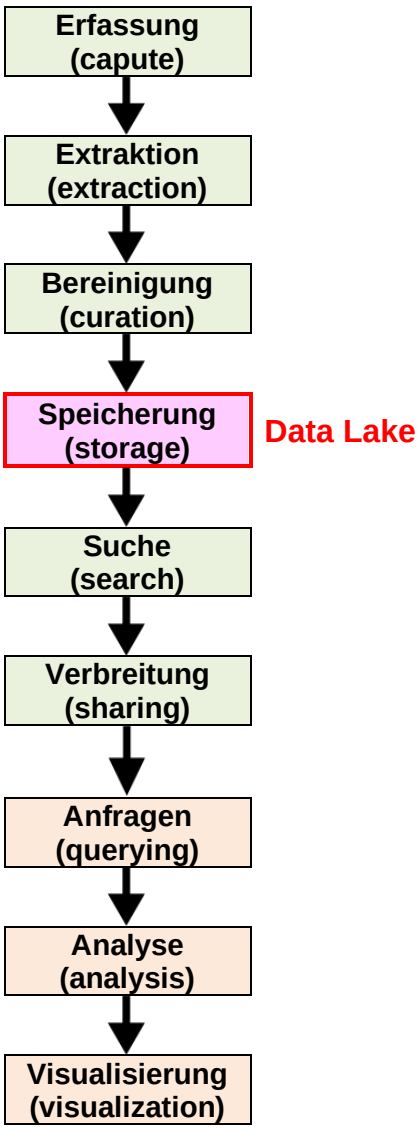
modernes Schlagwort, derzeit leicht gehypt
quasi eine Steigerung des Big Data
Fortsetzung der "Jäger- und Sammler-Mentalität" auf die Ebene der Digitalisierung
da (sehr) viele Daten sehr einfach zugänglich sind und auch billig gespeichert werden können, werden diese auch für weitere / spätere Verwendungen aufgehoben
(Man weiss ja nicht, wofür es später mal gut sein könnte!)

massenhafte Sammlung von unstrukturierten Daten
möglichst in Reinform, um wirklich wenig verfälschte daten zur Verfügung zu haben
die strukturiierten Daten-Bestände in den klassischen Big-Data-Datenbanken werden mit einbezogen

quasi Sammlung von Daten aus hoch-strukturierten Datenbanken, halb-strukturierenten Daten z.B. in JSON- oder XML-Dateien sowie Binär-Daten, wie Bilder, Video's, …

Beispiele microsoft Azure Data Lake, Amazon S3, Apache HADOOP, …

in der Data-Science-Pipeline
findet man Data Lakes



Data Lake mehr im Zusammenhang mit eigenen / internen Daten-Beständen verstanden
Data Hub meint eher Datensammlungen für die Veröffentlichung / den externen Gebrauch

wichtige Aspekte sind die Quellen-Nachweise und die Durchsuchbarkeit für die Daten-Bestände
meist hängen großen Meta-Datenbanken an den Rohdaten
erweiterte Verschlagwortung, nicht nur der Daten in den Tabellen sondern auch für die Tabellen selbst

Log-Daten beschreiben schon vollzogene Zugriffe und ev. auch Transformationen usw. usf.
bei schlechter Arbeit mit den Daten-Beständen und einem fehlendem Konzept können dann auch Data-Swamp's (Daten-Sümpfe) entstehen, mit denen man im schlimsten Fall gar nichts mehr anfangen kann

8.x.y.z.1. Daten-Herkunft

wo kamen die Daten (ursprünglich) her, warum stehen die Daten in dieser Form zur Verfügung
Data origin, Data provenance, Data …

wichtig auch für die Bewertung der Daten-Qualität, für nachlaufende Bewertungen der Daten und Analysen
Analyse der Daten-Pfade und durchlaufenen (vielleicht auch ungünstigen / ungeeigneten) Veränderungen

offensichtliche Passung reicht nicht, da oft bei Quellen-Prüfungen Widersprüche gefunden werden
gerne Schätzungen benutzt oder Quellen verwendet, die gleiche Zahlen (trotz eigentlich unterschiedlistzer Ausgangs-Bedingungen) liefern

WHERE-Provenance
Wo leigen meine (verwendeten) Daten?

WHY-Provenance
Warum entsteht dieses (Analyse-)Ergebnis? (Wie enststand das Ergebnis?)

WHY-NOT-Problematik
Warum sehe isch ein bestimmtes – von mir erwartetes – Ergebnis nicht?
Was fehlt hier? Was ist bei der Analyse schief gelaufen? …

8.x.y.z. virtuelle Integration

quasi Gegen-Stück zur Materialisierung der Daten während der Integration

Nachteile der Materialisierung:
- großer Speicher-Bedarf
- vollständige Aktualität nicht gegeben (Daten sind immer (etwas) veraltet)

Ziel der Virtualisierung:
- Daten sollen bei der Quelle bleiben
- es entstehen gefühlte / temporäre Daten-Bestände

- hohe Aktualität
- Daten-Quellen sollen / wollen Autonomie behalten

da die Daten nicht wirklich existieren, müssen Anfragen usw. wieder zurück an die originalen Quellen gegeben werden, erneut analysiert und dargestellt werden
Daten-Quellen können autonom bleiben

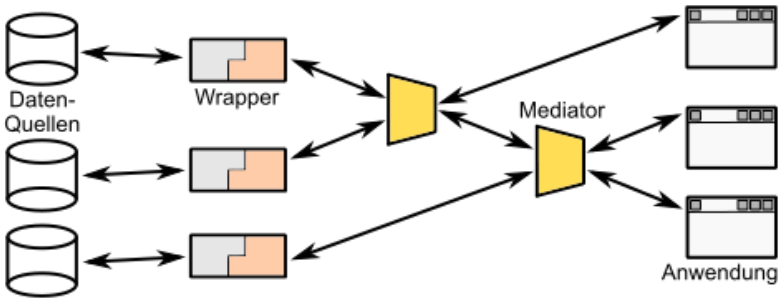
Nachteile der Virtualisierung:

- längere Verarbeitungs-Zeiten
- da Zeit kritische Diemension ist, können Daten nur wenig Transformiert, Bereinigt usw. usf. werden
- man verlässt sich sehr stark auf die Zuverlässigkeit der Daten-Quelle
- hohe Daten-Übertragungs-Mengen
- durch Autonomie der Quellen ist starke Abhängigkeit von deren Beständigkeit gegeben

8.x.y.z.1. Mediatoren / Wrapper

Wrapper sind Verpackungen / äußere Hüllen von Daten-Quellen
haben Schnittstellen-Funktion zwischen Daten-Quellen und Mediatoren
sammelt die Daten zusammen, bereit sie so auf, dass sie einem internen Daten-Modell entsprechen
entscheiden ev. auch über die Auswahl der Quellen

Mediatoren bereiten die Daten auf und stellen sie bedarfsgrecht den Anwendungen zur Verfügung
geben z.B. passende Anfragen an die Quellen weiter (z.B. gefilterte und sortierte Daten) und geben die gelieferten Ergebnisse an die Anwendungen weiter

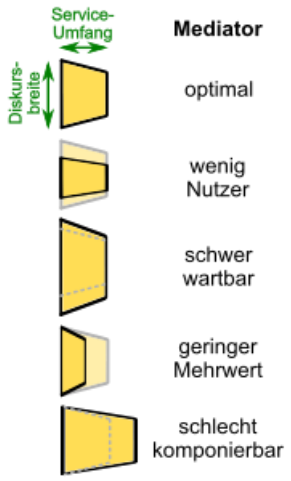


können die Quellen dagegen nicht sortieren oder gefilterte Daten bereitstellen, dann übernimmt diese Aufgabe der Mediator

die Anwendung erhält immer gleichartige Daten(-Strukturen und –Qualitäten)

Diskursbreite (domain scope) = Vielfalt der Daten, die eingelesen werden können

Service-Umfang (service scope) = Umfang der intergrierten Leistungen im Mediator



dünne und dicke Mediatoren
(nach WIEDERHOLD (1999))

8.x.y.z. das Deep Web

beinhaltet solche Daten-Quellen, die nicht ihren gesamten Datenbestand verfügbar machen, sondern nur Teile zugänglich / abrufbar sind
Daten nicht (direkter) Download abrufbar
Daten sind nur durch Such-Anfragen zugänglich
z.B. Katalog von Amazon ist nicht als solcher erreichbar (weil er auch so gar nicht existiert!), wohl aber Details nach einer entsprechenden Anfrage

auch Hidden Web, Invisible Web

meist besonders gut gepflegt
meist professionell erstellt
wenn man die bereitgestellten Daten gut (aus den Anfrage-Formularen) auslesen kann, dann hat i.A. sehr gute Daten

soll – wie bei einem Eisberg – den unter Wasser liegenden Teil – des Internet's ausmachen
oberster Teil wird dementsprechend Surface Web genannt
kurz unter der Wasser-Oberfläche befindet sich das Shallow Web, das im Wesentlichen aus den dynamischen Webseiten besteht, die individuell für den Nutzer zusammengestellt wurden und nicht wirklich irgendwo abgespeichert werden bzw. existieren

unterster Teil des Eisberg's ist dann das Dark Web

8.x. Data Mining, Statistik, Maschinelles Lernen

8.x.1. Statistik

> Traue keiner Statistik,
> die du nicht selbst gefälscht hast.
> Winston CHURCHILL

Obwohl dieses Zitat Winston CHURCHILL und manchmal auch Joseph GOEBBELS zugeordnet wird, ist es wohl nur eine Umformulierung einer Aussage in einem Zeitschriften-Artikel von Hanns-Erich HAACK: "… So viel haben sie schon gelernt,
daß sie nur den Statistiken glauben, die sie selbst gefälscht haben."
Hier im Skript geht aber nicht darum Statistiken zu fäschen, sondern aufzuzeigen, wo die Fallstricke liegen und wie man Manipulationen erkennen kann.

explorative Statistik
erforschende Analyse der Daten z.B. in Visualisierungen (z.B. durch Hereinlegen einer Gerade od.ä.)

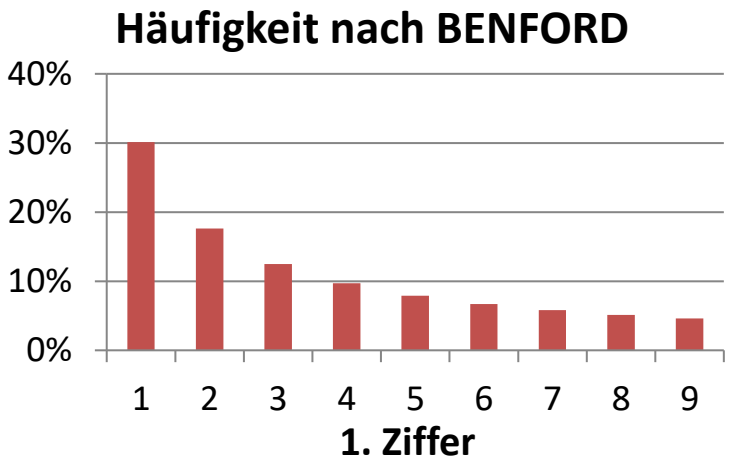
deskriptive Statistik
beschreiben die Daten (als Gesamtheit) z.B. über den Mittelwert
benennen von Kurven-Formen

induktive Statistik
Ableitung von Merkmalen der Gesamtheit aus einer Stichprobe
basiert stark auf Wahrscheinlichkeits-Rechnung

interessante Links:
unstatistik.de / http://www.rwi-essen.de/unstatistik/ (Beispiele für manipulierende Statistiken)

BENFORDsches Gesetz

lt. BENFORD beginnen Zahlen besonders häufig mit einer 1, etwas selterner mit einer 2 usw. usf.
Die Verteilung folgt der Formel:



$$P(d) = \lg(d+1) - \lg(d) \; = \; \lg\left(1 + \frac{1}{d}\right)$$

besser ist eigentlich Regel für diese Beziehung
nicht allgemeingültig, gilt aber für sehr viele Daten
besonders für Zahlen gültig, die in einem sehr großen Zahlen-Bereich (über mehrere Zehner-Potenzen) kommen

lässt sich auch für die Prüfung auf Daten-Manipulationen verwenden
selbst ausgedachte Zahlen, z.B. bei Bilanz-Fälschungen folgen nicht dieser Regel
oder im anderen Extremfall werden Zahlen so manipuliert, dass sie optimal zur Regel passen (wenn der Fälscher die Regel kennt)
wird teilweise sogar vor Gericht verwendet

Schummeln mit Statistik

verschiedene Interpretation für "Durchschnitt"

"Durchschnitt"	Definition	Berechnung	Bemerkungen
arithmetisches Mittel	Quotient aus der Summe aller Werte und der Anzahl der Werte	$\bar{x} = \frac{\sum x_i}{n}$	üblicher Durchschnitt
geometrisches Mittel			
harmonisches Mittel			gleicht Extremwerte aus
Median	ist der Wert der in der geordneten Reihe der Werte genau in der Mitte liegt (bzw. das arithmetrische Mittel der beiden mittleren Werte)		meist besser geeignet, da Ausreißer das Ergebnis weniger verfälschen
Modus	ist der am Häufigsten vorkommende Wert		

Visualisierung

Ziele:

- Darstellung komplexer Zusammenhänge auf einfache, verständliche Art und Weise
- Darstellung zusammenfassender Informationen
- Exploration von Daten
-

Problem:

- sind praktisch immer vereinfachsender Natur
- nur wenige Dimensionen möglich
- unbekannte Zusammenhänge können verloren gehen, weil man sie nicht vermutet und deshalb nicht in die Visualisierung eingeschlossen hat

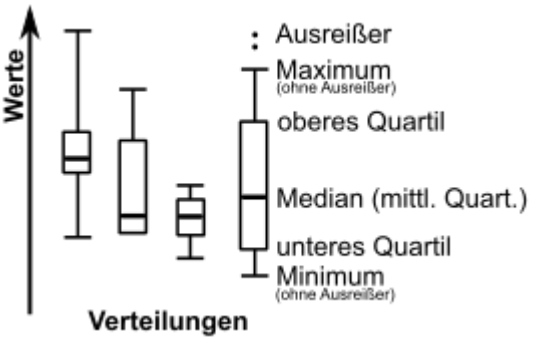
Diagramm-Arten:

- Balken- / Säulen-Diagramm
- Kreis-Diagramm
- Netz-Diagramm
- VENN-Diagramm
- Box-Plot's
- Violin-Plot
- Radar-Diagramm
- Punkt-Diagramme (x-y(-z)-Diagramme)
- Flächen-Diagramm
- …

zusätzlich 3D-Versionen z.B. Torten-Diagramm (statt Kreis-Diagramm) oder Zylinder-Diagramm (statt Säulen-Diagramm) ohne Zusatz-Information in der 3. Dimension (nur grafischer Effekt!)

Boxplot

Kasten-Diagramm
für große Mengen an Zahlen mit Maximum und Minimum (T-Grenzen)
ev. mit Ausreißer die als Einzel-Punkte auf der Ebene der Streuungs-Linie eingezeichnet
wird Streuungs-Linie gestrichelt gezeichnet
Kasten zeigt den Werte-Bereich zwischen oberen und unteren Quartil und damit die mittleren 50 % der Werte (in der geordnenten Liste)



stellt weiterhin z.B. den Median dar (als mittlere (dicke) Linie im Kasten)
gut für den Vergleich mehrerer Verteilungen

Arten der Achsen

lineare Achse
bieten gute Übersicht
Erwartung der Normal-Nutzer

logarithmische Achsen
bietet bessere Übersicht bei Werte-Bereichen über mehrere Dimensionen / Zehner-Potenzen
verbessert die Detail-Ansicht im Bereich kleiner Zahlen

reziproke Achsen

Schummeln mit Visualisierung

z.B. durch andere Achsen-Skalierungen
Ausschnitte auf x- und / oder y-Achse mit ev. variablem Werte-Bereich
fehlerhafte Diagramm-Typen für die Daten

Risiko-Kompetenz

Zahlen- und Kennwerte-Verständnis

Beispiel:
durch die Einführung eines diagnostischen Verfahrens sei die Sterblichkeit um 24 % gesunken

Fehl-Interpretationen:
von 1'000 untersuchten Personen sterben 240 weniger
von 1'000 untersuchten Personen sterben 240

richtig ist:
bei 1'000 untersuchten reduziert sich die Sterblichkeit vom Ursprungswert zum Neuwert um 24 %
→ Normal-Sterblichkeit liegt angenommen z.B. bei 10 % (also 100 von 1'000 Personen), dann liegt sie durch das diagnostische Verfahren jetzt nur noch bei 7,6 % (also rund 8 Personen von 1'000)

SIMPSON-Paradoxon

Gegeben sind z.B. die nebenstehenden Daten von Fußball-Spielern.
Die Frage ist nun, welchen Spieler sollte man von den beiden kaufen? Wer ist der bessere Spieler?

Saison	Spieler	Tore	Torschüsse	Quote
2017/18	Robert	5	20	25 %
	Nils	90	300	30 %
2018/19	Robert	120	330	36 %
	Nils	28	70	40 %

Q: Daten aus HPI-Kurs (fiktiv)

Da Nils in beiden Saison's die höhere Quoto wird man schnell verleitet, Nils auch als den besseren Spieler zu halten.
Betrachtet man aber beide Spielzeiten zusammen, dann ergibt sich anderes Bild. Hier stellt sich Robert als der bessere Spieler heraus, weil er insgesamt eine höhere Quote hat.

Saison	Spieler	Tore	Torschüsse	Quote
2017/18/19	Robert	125	350	36 %
	Nils	118	370	32 %

Q: Daten aus HPI-Kurs (fiktiv)

SIMPSON-Effekt beschreibt Veränderung von statistischen Kennwerten bei variablen Grundgesamtheiten. Die Einzel-Ergebnisse gehen mit unterschiedlichen Gewichten in die Gesamt-Bewertung ein.

beschrieben wurde der Effekt zuerst von Edward Hugh SIMPSON ()

Beispiel (Q: de.wikipedia.org)

	weiblich			männlich		
	bestanden	**gesamt**	**Durchfall-Quote**	**bestanden**	**gesamt**	**Durchfall-Quote**
1. Tag	7	8	12,5 %	1	1	0,0 %
2. Tag	1	2	50,0 %	2	3	33,3 %

An beiden Tagen wurde für die Frauen eine deutlich höhere Durchfall-Quote ermittelt. Betrachtet man aber die Gesamtheit (für beide Tage), dann dreht sich Bild:

	weiblich			männlich		
	bestanden	**gesamt**	**Durchfall-Quote**	**bestanden**	**gesamt**	**Durchfall-Quote**
1. Tag	8	10	20,0 %	3	4	25,0 %

Diskriminierungs-Klage gegen die University of California in Berkeley

Die Zahlen stammen aus dem Herbst 1973 und zeigen eine deutliche Ungleich-Verteilung. Auch ein Signifikanz-Test bestätigte, dass diese Zahlen nicht zufällig sein können.

	Bewerber	zugelassen
weiblich	4321	35 %
männlich	8442	44 %

Bei der genauen Analyse ergab sich aber ein weitaus differenziertes Bild. Von 101 Fakultäten gab es 16 in dem sich nur ein Geschlecht beworben hatte. Bei den restlichen mit Bewerbern beider Gesdchlechter ergab sich, dass in 4 Fakultäten die Männer signifikant höhere Erfolgs-Quoten hatten. Bei 6 Fakultäten ergab sich eine signifikant höhere Chance für Frauen.

8.x.2. Data Mining und Machine Learning

Ziele von Data Mining:

- Aussagen über vorhandene Daten treffen → Statistiken, Visualisierung
- unbekannte Zusammenhänge finden → Cluster-Analyse, Klassifizierungen, Visualisierungen, Assoziations-Regeln, Regressionen, Ausreißer-Erkennung
- Aussagen über die zukünftige Entwicklung der Daten machen → Trend's
- …

Themen zum maschinellem Lernen:

-

deskriptive und prädikative Analyse

deskriptive Analyse ist beschreibend
vorhandene Daten werden analysiert und Kennwerte bestimmt
Was ist passiert?

ev. erweitert durch die
diagnostische Analyse
sucht die Ursachen für die Daten / Werte
Warum ist etwas passiert

prädikative Analyse ist vorausschauend
vorhandene Daten werden dazu benutzt um Voraussagen für die Zukunft zu machen
Was wird vermutlich passieren?

ev. erweitert durch die
präskriptive Analyse
beeinflussen der weiteren Entwicklung der Daten / Werte
Wie kann man es bewirken?

Assoziations-Regeln

aus der Mitte der 90iger Jahre
typisch Warenkorb-Analyse
Was haben die Kunden gekauft?

Analyse von Warenkorben:
K1 = {Cola, Wasser, Whisky}
K2 = {Saft, Cola}
K3 = {Wasser, Bier}
K4 = {Wasser}
K5 = {Bier, Cola, Wasser}
K6 = {Cola, Bier}

K7 = {Bier, Wasser}
K8 = {Wein, Wasser}
K9 = {Wasser, Bier}

R1: WENN K_i = *{Wasser}* DANN K_i = *{Bier}* (Wenn im Warenkorb Wasser ist, dann ist auch Bier enthalten!)

Assoziations-Regel: {Wasser} → {Bier} heißt im OpenHPI-Kurs: Prozentsatz der Warenkörbe mit Wasser, die auch (besser wahrscheinlich: "schon") Bier enthalten
{Gin} → {Tonic}: Warenkörbe indenen sowohl Gin, als auch Tonic enthalten ist

Support:
die Regel X → Y hat einen Support s, falls s% aller Warenkörbe sowohl X als auch Y enthalten

Support s für R1 und die Beispiel-Warenkörbe K1 … K9:
bei 3 Körben von insgesamt 9 ist s = 3/9 = 33,3 %

großer Support bedeutet große Häufigkeit der Kombination, daraus können Maßnahmen abgeleitet werden: z.B.:

- gemeinsam bewerben
- gemeinsam präsentieren
- Kombinations-Pakete anbieten
- …

Konfidenz:
die Regel X → Y hat eine Konfidenz c, falls c% aller Warenkörbe, die X enthalten auch Y enthalten

Konfidenz c für R1 und die Beispiel-Warenkörbe K1 … K9:
es enthalten 7 Körbe Wasser, aber nur bei 3 auch Bier: c = 3/7 = 42,9 %

gesucht sind dann Regeln, die einen Mindest-Support sowie / und / oder eine Mindest-Konfidenz haben

Mining von Assoziations-Regeln

A-Priori-Algorithmus (Apriori-Algorithmus)

1. Finde alle Produkt-Teilmengen mit dem Mindest-Support
2. Leite daraus Assoziations-Regeln mit der Mindest-Konfidenz ab

Problem sind die riesigen Produkt- Teilmengen
praktisch 2^n Möglichkeiten

Support kann bei steigender Produkt-Menge nur sinken

→ Wenn eine Produkt-menge den Mindest-Support nicht erreicht, dann kann es auch keine Obermenge.
→ Bottum-up Kandidaten-Generierung

für die Beispiel-Warenkörbe bedeutet das, dass es bei so seltenen Einkäufen mit Whisky, dessen Support hier sehr wahrscheinlich unter einem sinnvollen Wert liegt, so dass keine Kombinationen Whisky mit Cola usw. usf. untersucht werden müssen

Ableitung der Assoziations-Regeln nur noch aus dem Rest der Warenkörbe

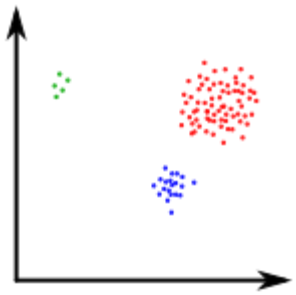
A-Priori-Algorithmus
→ http://www.vldb.org/conf/1994/P487.PDF

Clustering

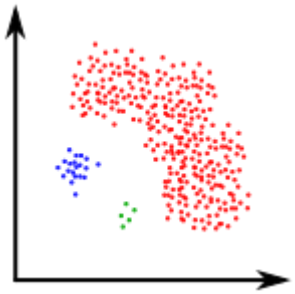
Gruppen-Bildung
Partitionieren, Clustern
sind meist entdeckerische, nicht-überwachte (Lern-)Verfahren

meist werden die Anzahl der gesuchten Cluster und das Ähnlicjkeits-Maß vorgegeben

Ähnlichkeit der Objekte innerhalb einer Gruppe soll sehr groß sein und die Abweichung / Nicht-Ähnlichkeit zu allen anderen Gruppen möglichst groß
als Maß wird die Distanz (Punkt-Abstand) benutzt
→ Objekte mit kleiner Distanz zueinander gehören zu einem Cluster, wenn auch die Distanz zu den anderen Cluster möglichst groß wird / ist
es gehen aber auch andere Ähnlichkeits-Merkmale → s.a. Duplikat-Erkennung

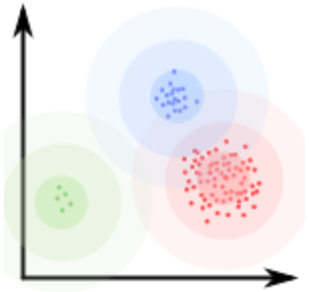


Problem bei Gruppen, die nicht Kreis-förmig sind
hier kann Abstand zwischen zwei zugehörigen Objekten größer sein, als der zu einem anderen Cluster



Dichte-basiertes Clustern
Bildung von Clustern, in den zu mindestens einem anderen Objekt eine große Ähnlichkeit besteht (und zu dem Nächsten Anderen der anderen Cluster möglichst größere Abstände)

Wahrscheinlichkeit-basiertes Clustern / weiches Clustern
für alle Objekt werden die Wahrscheinlichkeiten berechnet, wie sie zu einem Cluster gehören (über die Ähnlichkeit bezüglich des Cluster-Zentrums)



k-Means-Clustering

k steht für die Anzahl der Cluster, die herausgesucht werden sollen
Mean … Mittelwert

Verfahren:

1. zufälliges Auswählen von k Objekten als angenommene Cluster-Zentren
2. Ermittle für jedes andere Objekt den Abstand zu den Cluster-Zentren und ordne es dem dichtesten zu
3. Bildung eines neuen Cluster-Zentrums (Suche des Objektes im Zentrum) aus den in einem gesammelten Cluster Objekte
4. Wiederhole solange bei 2. bis keine Neuzuordnungen von Objekten mehr stattfinden

im originalen k-Means-Verfahren werden nicht Objekte als Cluster-Zentrum benutzt, sondern zufällige Positionen
Verfahren findet WORONOI-Zellen (auch VORONOI-Zellen)

Probleme:

- Anzahl der gesuchten Cluster muss vorgegeben werden
- bei ungünstigen Start-Zentren kann das Verfahren recht lange dauern, da sich die Abstände (Objekt - Zentrum) nur wenig unterscheiden
- Ausreißer werden immer zugeordnet

Verbesserungen sind:

- k-Median
- k-Means++

überwachtes und unüberwachtes Lernen

Ziel ist die Klassifizierung von (neuen) Objekten
Zuordnung eines (neuen) Objekt's zu vorhandenen Gruppen / Klassen

un-überwachtes Lernen (Unsupervised Learning / Training)

Verfahren funktionieren von sich heraus
es werden alle Daten für das Verfahren genutzt
es gibt keine vorgegebenen Trainings-Daten
keine

optimal sind besonders viele Attribute (oder abgeleitete Werte) einzubeziehen
ist aber etsprechend rechenaufwendig, also tendenz zu möglichst wenigen – aber klar funktionierenden Attributen (oder abgeleiteten Werten)

überwachtes Lernen (Supervised Learning / Training)

hier müssen Nutzer-Annotationen (Klassifizierungs-Informationen durch Nutzer) dazu kommen
verlangt also Trainings-Daten (oder ein Teil der Roh-Daten muss vorbearbeitet werden)

Problem ist die aufwendige Trainings-Phase, ev. werden echte Experten gebraucht
ev. können Wissens-Datenbanken benutzt werden

aktives Lernen (Active Learning)
System schlägt besonders wichtige / interessierende Atribute vor, die für den Lern-Prozess wichtig sind
damit Experten / Trainer nicht unnötig viele Daten beurteilen müssen, werden vom System nur die problematischen Objekte vorgeschlagen, die dann vom Experten annotiert werden sollen / müssen

Trainings-Daten / Test-Daten

Ziel ist die Verallgemeinerbarkeit des zugrundelegenden Modell's
fehlerfreies Funktionieren des Modell's bei neuen Daten
sichere Klassifizierung neuer Objekte

im Normalfall werden mehrere Modelle verglichen
sollte eine zeitliche Trennun möglich sein, dann sollte diese erfolgen
(z.B. bei Hass-Kommentaren nur mut Daten trainieren, die vor einem Stichtag erschienen sind, getestet wird dann mit Daten nach dem Stichtag)

Schritte beim Training eines Neuronalen Netzwerkes

- zufällige Initialiserung der Gewichte
- Vorhersage auf ein Trainings-Beispiel
- Berechnung der Abweichungen der Vorhersage zur Annotation

Trainings-Daten

sind (der größere) Teil der annotierten Daten, die für das Anlernen des System benutzt werden
sie werden benutzt, um die Parameter des Modell zu optimieren
ev. in mehreren Durchläufen / Interationen

Test-Daten

sind (ein kleinerer) Teil der annotierten Daten, die nur für den abschließenden Test des System benutzt werden
typischerweise rund 20 % der zur Verfügung stehenden annotierten Daten
diese Daten sind in der Entwicklungs- und Optimierungs-Phase des Modell's nicht einzusetzen

bei der Cross-Validierung (Kreuz-Validierung, f-fold cross validation)
Aufteilung der annotierten Daten auf k annähernd gleich große Partitionen
benutzt man z.B. einen Teil (meist rund 10 %) der Daten für den abschließenden Test, die anderen 90 % werden als Trainings-Daten benutzt
also Training mit k-1 Partitionen
man nimmt dann aus dem gleichen Gesamt-Datenbestand wieder einen Umfang von 90 % zum Trainieren und den restlichen Teil für das Testen
Qualitäts-Test mit den restlichen Partitionen
usw. usf.

weitere Verbesserung möglich, wenn neben den üblichen 20 % Test-Daten noch mal rund 20 % Validierungs-Daten abgetrennt werden
mit den restlichen rund 60 % wird trainiert
mit den Validierungs-Daten wird das System selbst feingetunt / optimiert
hier kann man auch noch mit der Cross-Validierung arbeiten

Overfitting (Überanpassung)

passiert, wenn das Modell zu stark an die Trainigs-Daten angepasst ist
tritt häufig dann auf, wenn das Modell sehr komplex ist und sehr viele Attribute für die Klassifizierung eine Rolle spielen

System erkennt etwas anderes, weil Trainings-Daten nicht optimal ausgewählt wurden, oder auch wenn Trainings-Daten auch zum Testen benutzt werden

im Extremfall ist es so, dass das System für jedes Trainings-Objekt genau seine Zuordnung gelernt hat

Beispiel: Erkennung von Wolfs- und Haski-Bildern
System sollte beide Hunde-Artigen unterscheiden lernen, was mittels Trainings- und Test-Daten auch perfekt geklappt hat.
Bei einem Praxis-Test erkannte das System die Wölfe aber nicht richtig, bzw. einige Haski's als Wölfe.
Bei der Suche nach der / den fehler-Quellen stellte sich heraus, dass das System nicht Hunde und Wölfe unterschied, sondern Bilder mit Schnee und ohne.
Die Wölfe waren immer auf Bildern mit Schnee abgebildet, und genau dass hatte das System gelernt.

Ähnliches Beispiel aus dem Militär-Bereich:

System sollte feindliche und eigene Panzer unterscheiden lernen. Klappte auch wieder perfekt. Im Praxis-Test und der nachfolgenden Fehler-analyse stellete sich heraus, dass das System nur Schönwetter-Foto's der eigenen Panzer, von den Schlechtwetter-Foto's der feindlichen Panzer unterscheiden konnte.

Klassifizierung

Aufgabe ist es, ein neues Objekt in eine der vordefinierten / erlernten Klassen einzuordnen

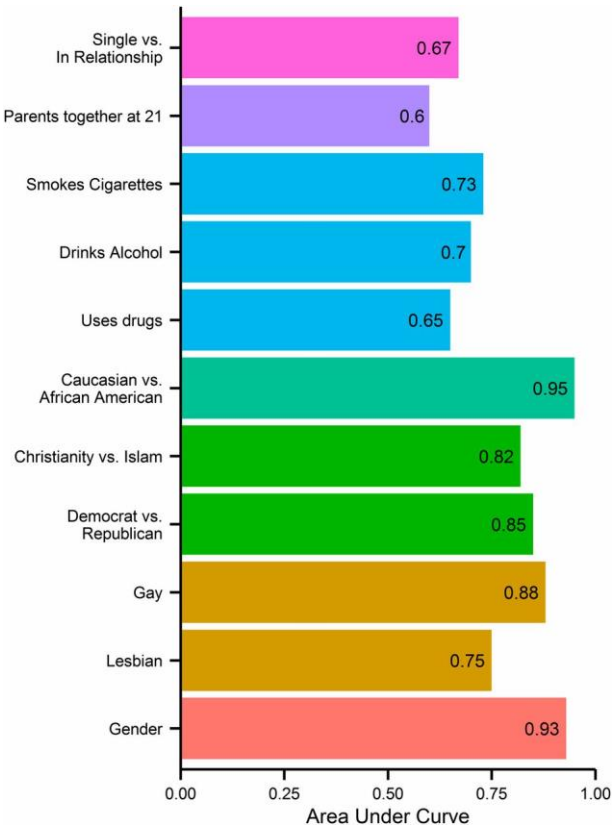
typische Beispiele:

- Bild-Erkennung
- Betrugs-Erkennung
- Personen
- Tumore
- Auto-Kennzeichen
- autonomes Fahren
- Duplikat-Erkennung
- Stimmungs-Erkennung
- Hass-Kommentare
- Stimm- und Sprach-Erkennung
- Finger-Abdrücke
- Bonitäts-Prüfung
- Erkennung von Risiko-Fahrern
- Empfehlungs-Systeme
- …

Beispiel: Klassifizierung von Facebook-Nutzern anhand der Likes zu Webseiten oder Produkten (nicht zu anderen Personen!)
Aufgabe war die Ableitung anderer Eigenschaften / bzw. eine neuartige Klassifizierung in Gruppen, zu denen kein Attribut-Bezug bestand

es wurden nur binäre Attribute betrachtet

die Wahrscheinlichkeit für die Exaktheit der vorausgesagten Eigenschaften / Klassifizierungen sind in der Abb. dargestellt
im

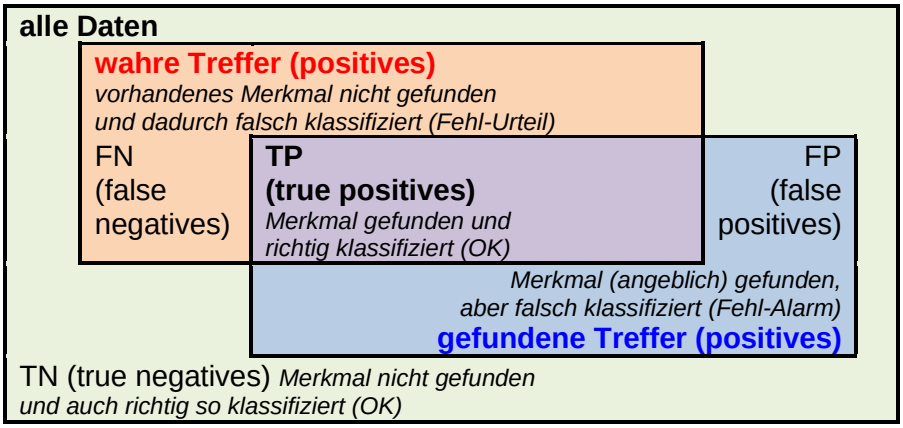


Q: https://www.pnas.org/content/110/15/5802

Erfolgs-Maße

für die binäre Klassifizierung, also Zuordnung zu der einen oder der anderen Klasse

gemeint immer ein Merkmal (ist weiblich, hat Tumor, ist Hund, …) und das Gegenstück dazu (ist nicht weiblich, hat kein Tumor, ist kein Hund, …)



Mensch	System		Realität
Bewertung der Beurteilung / Klassifizierung	**Beurteilung Klassifizierung Klasse**	**Gruppe**	**Attribut Merkmal Eigenschaft**
OK	Merkmal erkannt / angezeigt / zugeord.	**TP** r_p	**Merkmals-Träger**
Fehl-Alarm / -Urteil		**TN** r_n	**kein Merkmal**
Fehl-Urteil	Merkmal nicht erkannt / angezeigt / zugeord.	**FP** f_p	**Merkmals-Träger**
OK		**FN** f_n	**kein Merkmal**

Akkuratheit / Korrektheit / Treffergenauigkeit / Vertrauens-Wahrscheinlichkeit / Korrekt-Klassifikations-Rate / :

Accuracy = ACC = (TP + TN) / (TP + TN + FP + FN) = (TP + TN) / n = (TP + TN) / (P + N)

n = P + N = TP + TN + FP + FN

Präzision / Genauigkeit / Spezifität / Richtig-Negativ-Rate / Relevanz / Wirksamkeit : positiver Vorhersage-Wert, positiver prädikativer Wert
Specificity, correct rejection rate, true negative rate, positive predicative value (PPV)

Precision = PPV = TP / (TP + FP) = 1 – FDR

TNR = TN / (TN + FP) = TN / N = 1 – FPR **=** Specificity = $r_n / (r_n + f_p)$

Vollständigkeit / Erkennungs-Rate / Sensitivität / Richtig-Positiv-Rate / Treffer-Quote:
Sensitivity, true positiv rate (TPR), hit rate

Recall = TPR =TP / (FN + TP) = 1 – FNR **=** Sensitivity = $r_p / (r_p + f_n)$
= TP / P

Forderung: hohe Korrektheit bzw. hohe Genauigkeit bei hoher Vollständigkeit
→ Ziel-Konflikt
zur Auflösung wird als Maß das harmonische Mittel (F-Maß, F_1-Score) aus Precision und Recall berechnet

F-Measure = (2 * Precision * Recall) / (Precision + Recall))
F1 = (2 * PPV * TPR) / (PPV + TPR) = (2 * TP) / (TP + TN + FP + FN)

Ziel ist möglichst hoher Wert für F-Measure
harmonisches Mittel bestraft schon geringfügige fehlerhafte Präzisionen und Vollständigkeiten (deutlicher als das arithmetrische Mittel)

		Gold-Standard		
		Gold positiv	Gold negativ	
Predicted Entity	Pred positiv	**wahr positiv** true positive	**falsch positiv** false positive	**Präzision** precision
	Pred negativ	**falsch negativ** false negative	**wahr negativ** true negative	
		Rückruf recall		

weitere Maße:

Falsch-Negativ-Rate / :
false negative rate (FNR) / miss rate

= FNR = FN / (FN + TP) = 1 – TPR = FN / P **=** = f_n / (r_p + f_n)

Falsch-Positiv-Rate / Ausfall-Rate:
false positive rate (FPR) / fallout

Fallout = FPR = FP / (FP + TN) = FP / N = 1 – TNR **?** = r_n / (r_n + f_p)

False discovery rate (FDR):

FDR = FP / (FP + TP) = 1 – PPV

False omission rate (FOR):

FOR = FN / (FN + TN) = 1 – NPV

Trenn-Fähigkeit / Segreganz / negativer Vorhersage-Wert:

negative predicative value (NPV)

= NPV = TN / (TN + FN) = 1 – FOR = = r_n / (r_n + f_n)

Falsch-Klassifikations-Rate / Klassifikations-Fehler:

= (f_p + f_n) / (r_p + f_p + r_n + f_n) = (f_p + f_n) / n = 1 - Accuracy

likelihood ratio (LR) / Likelihood-Quotienten-Test:

LRpositiv = Sensitivität / (1- Spezifität)

LPnegativ = (1- Sensitivität) / Spezifität

MATTHEWS-Korrelations-Koeffizent (Matthews correlation coefficent):

MCC = (TP * TN – FP * FN) / Wurzel((TP + FP) * (TP + FN) * (TN + FP) *(TN + FN))

Threat score (TS) / Critical Sucess Index (CSI):

TS = CSI = TP / (TP + FN + FP)

Bookmaker Informedness (BM) / Informedness:

BM = TPR + TNR – 1

Markedness (MK):

MK = PPV + NPV – 1

Effektivitäts-Maß (E):

E = 1 / (α(1 / P) + (1 – α)/R
bei α = 1 → Genauigkeit bei α = 0 → Treffer-Quote
0 … beste Effektivität 1 … schlechteste Effektivität

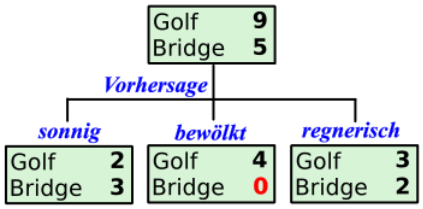
interessante Links:
https://www.spiegel.de/wissenschaft/mensch/medizinische-tests-und-statistik-denken-sie-immer-falsch-positiv-a-1087042.html (Tücken der Statistik (fiktives Scharlach-Beispiel))
https://www.spiegel.de/gesundheit/diagnose/viele-aerzte-verstehen-statistiken-zu-diagnosen-nicht-a-844210.html (falsches Statistik-Verständnis unter Ärzten)

Entscheidungs-Bäume (decision trees)

Klassifizierungs-Verfahren

Beispiel (von OpenHPI)
Was machen wir heute? Gehen wir raus golfen oder spielen wir drinnen Brigde?

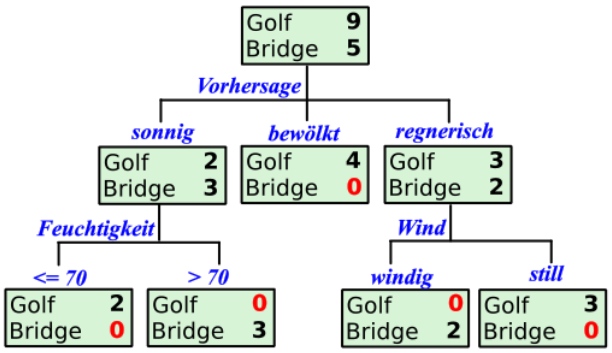
In einem ersten Verfahren könnte man z.B. versuchen eine Entscheidung nur über die Wetter-Vorhersage zu machen. Dabei ergäbe sich der folgende Baum:



Trainings-Daten

Vorhersage	Temp. (°F)	Feucht. (%)	Wind	Aktivität
sonnig	85	85	still	Bridge
sonnig	80	90	windig	Bridge
bewölkt	83	78	still	Golf
regnerisch	70	96	still	Golf
regnerisch	68	8	still	Golf
regnerisch	65	70	windig	Bridge
bewölkt	64	65	windig	Golf
sonnig	72	95	still	Bridge
sonnig	69	70	still	Golf
regnerisch	75	80	still	Golf
sonnig	75	70	windig	Golf
bewölkt	72	90	windig	Golf
bewölkt	81	75	still	Golf
regnerisch	71	80	windig	Bridge

Für "bewölkt" ergibt sich hier schon eine eindeutige Klassifizierung zu "Golf", da "Bridge" hier mit 0 Datensätzen in den Trainings-Daten vertreten ist.
Nun müssen die Gruppen "sonnig" und "regnerisch" noch weiter unterschieden werden. Bei der Wahl der Feuchtigkeit und einem Trigger-Wert von 70 lässt sich eine saubere Zweitteilung erzielen, wobei jede Teilgruppe eine 0-Gruppe enthält.
Die Gruppen "regnerisch" wir mittels des Attribut's "Wind" in die zwei Teilgruppen "windig" und "still" geteilt, die ebenfalls eine 0-Gruppe enthalten.
Damit ist ein Entscheidungs-Baum aufgebaut worden.



Vorteile:

- für Menschen gut bachvollziehbar
- anschaulich
- einfach zu implementierbarer Algorithmus

Ist dieser Baum aber der günstigste / schnellste / effektivste?
Bei mehr-dimensionalen Ansätzen kommt man zu Entscheidungs-Wäldern, in deren Ergebnis meist Wahrscheinlichkeiten für Klassifikationen stehen.
Support-Vektor-Maschine (Support vector machine) betrachten die Objekte in einen vieldimensionalen Raum und ziehen eine Trenn-Linie / - -Ebene / -???, um die Objekte jeweils der einen oder anderen Seite zuzuordnen.
Beim "Naive Bayes"-Verfahren (Verfahren der naiven Basis) werden die Wahrscheinlichkeiten für die Klassifikation berechnet und dann angewendet. Das Verfahren geht "naiv" davon aus, dass alle Attribute unabhängig voneinander sind.

neuronale Netze

versuchen mittels digitaler Technik das assoziative Denken den Mensch nachzubilden / zu simulieren

bestehen aus kleisten Elementen, den "Neuronen"
sind vernetzte Rechen-Elemente, die über Rück-Kopplungen die Bedeutung der verschiedenen Eingangs-Signale erlernt und dann an Ausgabe-"Neuronen" dann die Klassifikationen ausgeben

neuronale Netze sind i.A. vielschichtig, jedes "Neuron" ist mit jedem "Neuron" der nächsten Schicht verbunden
die Signal-Stärke, mit der es die einzelnen Eingaben / Eingabe-Neuronen beachtet wird über die Gewichte gelernt, dazu dient die Rück-Kopplung (Back-Propagation))

Neuron bestimmt zusätzlich seine Ausgangs-Signal-Stärke über die Aktivierungs-Funktion

Die Topologie eines Neuronalen Netzwerkes wird durch die Anzahl der Schichten / Ebenen und durch die Anzahl der Neuronen pro Schicht charakterisiert.
Die Kapazität eines Neuronalen Netzwerkes wird durch dessen Größe / Komplexität bestimmt.
Bei einer sehr großen Kapazität eines Neuronalen Netzwerkes besteht wieder die Gefahr, dass das Netzwerk die Trainings-Daten "auswendig" lernt und nicht wirklich für neue Klassifikationen taugt.

Deep Learning

derzeit stark gehypt
bietet sehr viele (neue) Möglichkeiten
besonders für große Daten-Mengen und komplexe Daten (z.B. Bilder-Inhalte) geeignet

typische Anwendungs-Gebiete:

- Bilder-(Inhalte-)Erkennung
- Erkennung von Tier-Arten
- Kennzeichen-Erfassung
- Personen-Verfolgung
- Gesichter-Erkennung
- Authentifizierungs-Verfahren
-

Schichten, die zwischen Eingangs- und Ausgangs-Schicht liegen sind i.A. für den Benutzer verdeckt (hidden layer's).
verfolgt man, was genau in den einzelnen Schichten passiert, dann stellt man fest, dass i.A. die ersten Schichten sehr abstrakte Klassifizierungen vornehmen (z.B. linke obere Seite des Bildes ist heller usw. usw.)
mit jeder weiteren Schicht werden die Klassifikationen –bezogen auf die Ziel-Klassifikation immer konkreter
hier werden jetzt konkrete Formen / Umrisse / Gebilde / Farbverläufe usw. usf erkannt, die dann in der letzten Schicht zur Ziel-Klasse kombiniert werden

Training erfolgt mit vielen Bildern und Foto's vom Objekt und zufälligen Start-Gewichten und Aktivierungs-Funktionen
Gegen-Training mit alternativen Bildern / Foto's

Fairness / systematische Abweichung

interne Autonomie des Lernens / Training's der Neuronalen Netzwerke sowie deren eigenständiges Agieren in der Nutz-Phase bergen diverse Gefahren:

- Anpassung von Schwell-Werten / Triggern in von Menschen nicht gewollte Richtungen (rassistisches / sexistisches Agieren; Grenzwerte bei Geld-Überweisungen (vielleicht ist Überweisung von 30'000 Euro schon ein zeichen für Geldwäsche / Terrorismus-Unterstützung oder nur ein Auto- oder Haus-Kauf)
-

typisches Beispiel:
Compass-Skandal
System sollte die Entscheidung fällen, ob eine inhaftierte Person auf Bewährung frei kommen soll
es sollte die Wahrscheinlichkeit der Rückfälligkeit ermittelt werden
System wurde so erstellt, dass es weiße Personen bevorteilte und farbige benachteiligte, obwohl sie andere vergleichbare Attribut-Werte hatten

Ursache war das Training des Sstem mit früheren Fällen und Entscheidungen von Richtern (, die i.A. zu Ungunsten von Farbigen entscheiden (statistisch gesichert))
mehr Farbige in den Trainings-Daten

das System hat nur den vorhandenen Rassismus gelernt und weiter angewendet

chinesische Kriminellen-Erkennung (an Gesichtern):
Trainings-Daten waren Foto's von Krimminellen und Foto's von anderen Personen aus dem Internet

systematische Fehler:
Kriminellen-Foto's eher düster, deprimiert, wütend, abweisend, aggressiv, …
Internet-Protrait-Foto's sind eher gut gelaunt, liebenswürdig, lustig, kontaktfreudig, aufgeschlossen, …
dazu kommt die Subjektivität der Richter (Vorverurteilung von dreckigen, düsteren, widerspenstigen, … Personen) einschließlich politisch motivierter Urteile, die aber gar nicht wirklich kriminelle Hintergründe / Wahrheiten beinhalten

Terroristen-Erkennung durch Geheimdienste:
Trainings-Daten bestanden aus 7 Telefon-Daten von Terroristen gegen 100'000 andere Telefon-Verhaltensweisen

Problem Korrelation gegen Kausalität (→)

erklärbare KI

explainable AI (XAI)
ist der Versuch und die Wissenschaft um die Erkundung und Erklärung der Vorgänge in der KI und besonders in Neuronalen Netzen (NN)

Nachvollzug von Entscheidungen der KI / Neuronalen Netzen

Schaffung von Rechtssicherheit (z.B. bei Personen-Klassifizierung (Krininelle, Terroristen, …))

funktioniert gut bei Entscheidungs-Bäumen (→) aber deutlich schlechter bei NN oder Support Vektor Maschinen

derzeit gibt es gute Erkenntnisse auf dem Gebiet der Bild-Erkennung
ebenfalls bei Stimmungs-Anlaysen (für texte), weil man die bedeutsamen Worte anzeigen lassen kann

Problem-Beispiel aus der Bild-Erkennung
System lernt Pferde-Bilder und andere "Nicht-Perde"
System versagte in der Praxis
Trainings-Daten hatten auf jedem Foto eine copyright-Kennung, die Nicht-Pferd-Bilder nicht
letztendlich hatte das System nur das copyright-Zeichen gelernt

9. Daten-Analyse mit R

nach einem OpenHPI-Kurs "Programmieren mit R für Einsteiger" (März/April 2022) von Berry BOESSENKOOL, Pia Francesca RISSOM und Bert ARNRICH

9.0. Einführung / Allgemeines / Historie

Software(-Umgebung) für Computer-gestützte Daten-Analyse, Statistik und Visualisierung (graphische Darstellung)

offizielles Logo
Q: r-project.org

reproduzierbare Analyse von Daten
programmierte Analyse-Abläufe
effektive und produktive Arbeit möglich

kostenlos, OpenSource

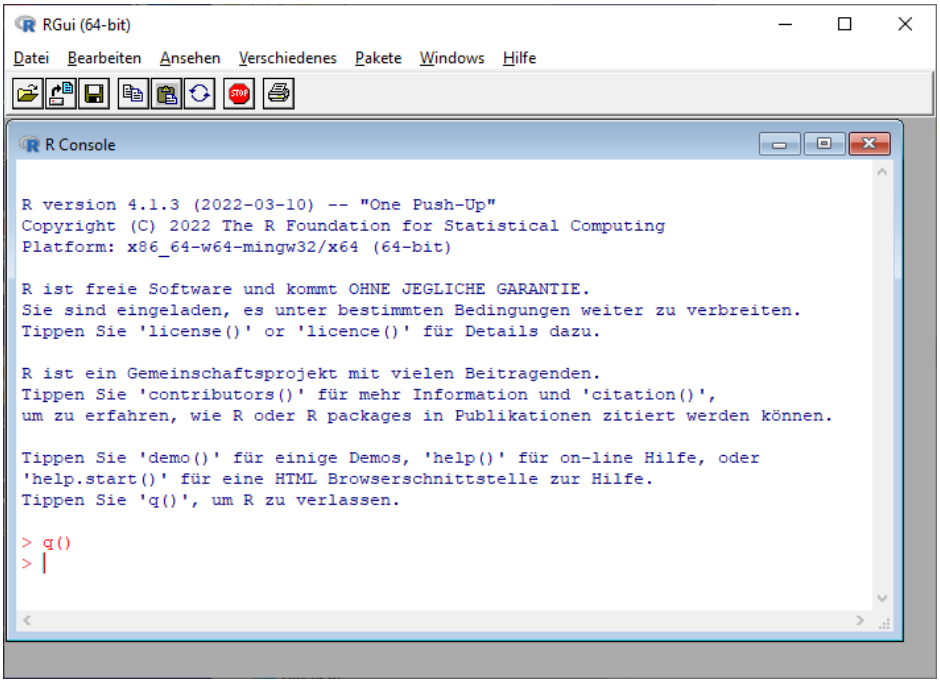
riesige Community → viel Hilfe, viele Methoden, …

weit verbreitet in Forschung und Wirtschaft

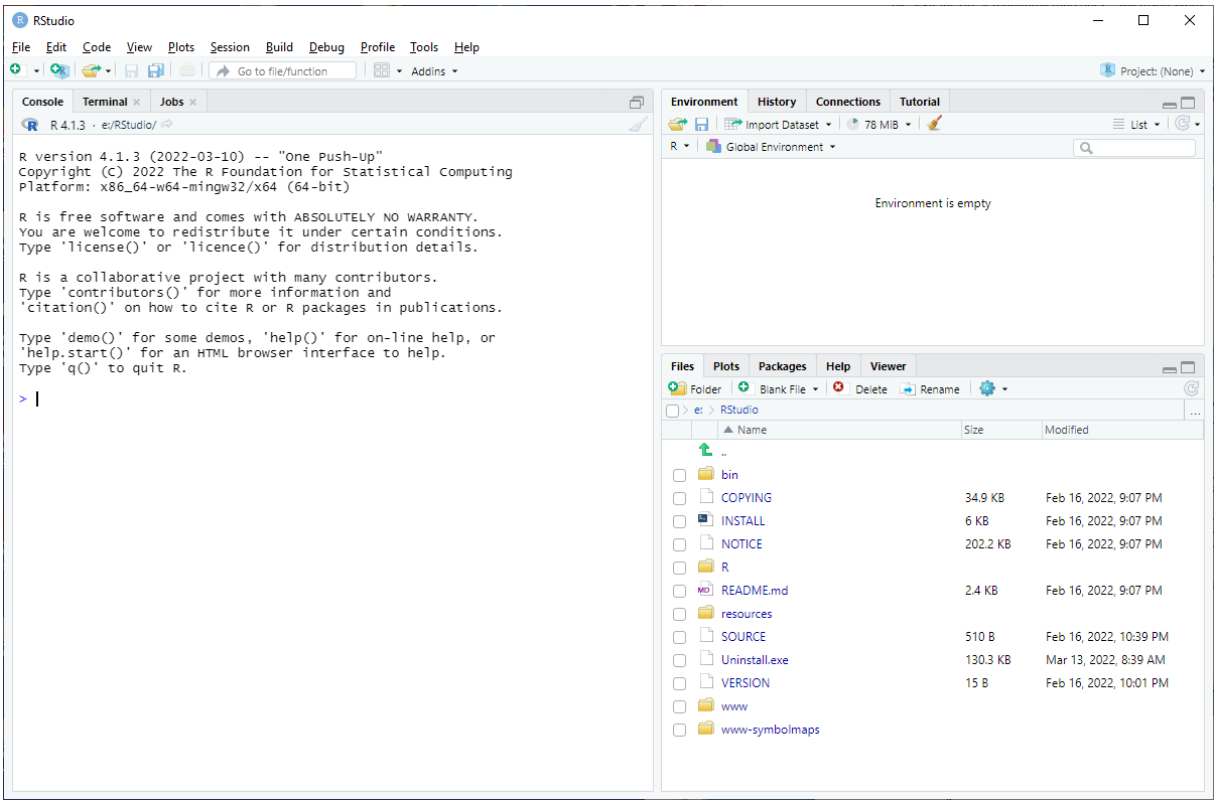
Gründe für R:

- riesiger Funktions-Umfang
- wird ständig weiterentwickelt
- neueste Methoden usw. werden schnell implementiert
- breite und kompetente Community
- gute Voraussetzung für Studium und Beruf
- kostenlos
- …

Download R
→ https://cran.r-project.org/bin/windows/base/
enthält einfache GUI
wird auch für die deutlich Leistungs-fähigere GUI RStudio gebraucht



RStudio
ist eine IDE für R
→ https://www.rstudio.com/products/rstudio/download/



9.1. Einrichtung von RStudio

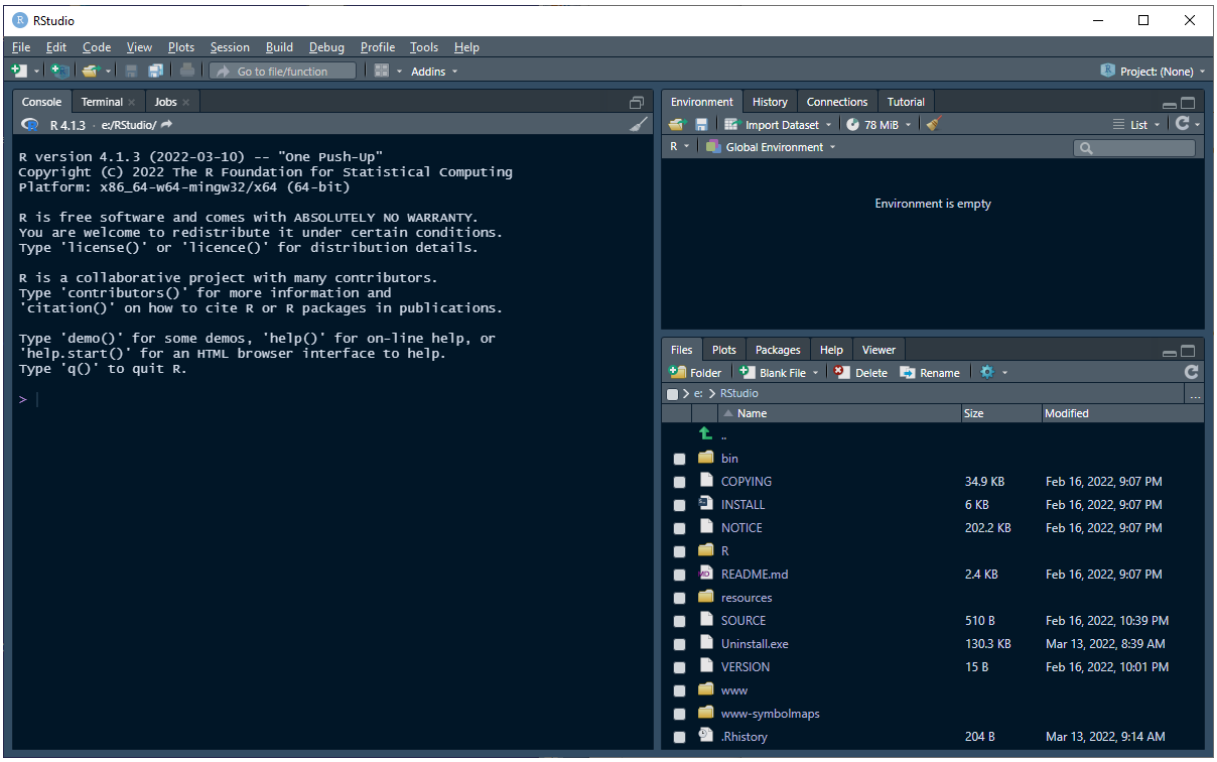
Download: https://rstudio.org

interaktives Arbeiten
praktisch alles, was man braucht ist im RStudio erreichbar
z.B.:
Tastatur-Befehle
Download's für Cheat Sheet's
…

Vorteile des RStudio's:

- Auto-Vervollständigung (um z.B. Tipp-Fehler zu minimieren)
- ein Arbeiten offline ist dann möglich
- Testen und Ausführen einzelner Zeilen
- Tastatur-Kürzel für viele Aufgaben / Funktionen
- Debugging-Tools
- Hilfe
- Pket-Verwaltung
- …

diverse Themes
u.a. auch mehrere Dark Modes
bei vielen Programmierern beliebt



weitere Themes Download → https://r-coder.com/rstudio-themes/

gebraucht wird nur die *.rstheme-Datei
diese im Programm-Ordner in den Unterordner resources/themes kopieren
beim nächsten Aufruf der Einstellungen "Tools" "Global Options …" steht das Thema unter "Appearance" "Editor theme" bereit

im Skript bleiben wir bei einem hellen Thema, da dadurch der verbrauch an Farbe bei Ausdrucken usw. reduziert wird

es lassen sich über einen speziellen Editor die Theme's auch anpassen
online-Editor für diverse Theme's https://tmtheme-editor.herokuapp.com/#!/editor/theme/

Übersicht über das RStudio

Skript-Bereich (links, oben)

Skripte im Text-Datei-Format
Code (Befehle) in Zeilen zusammengetragen

Skript wird in R ausgeführt (organisiert die IDE)
[Strg] + [Enter] .. Ausführen der aktuellen Zeile oder markierter Abschnitt der Zeile
[Strg] + [⇧] + [S] .. Ausführen des gesamten Skript's

Objekt-Verwaltung (rechts, oben) / Environment-Fenster

Konsole (links, unten)

Objekt- und Grafik-Anzeige (rechts, unten)

Arbeits-Verzeichnis einstellen:
für einfache Analysen
unter "Session" "Set Working Directory" "To Source Path Location"

echte Projekte über das Projekt-Management unter "File"
"File" "New Project …" erstellt ein neues Projekt in einem Ordner
setzt dann das Arbeits-Verzeichnis auf den Projekt-Ordner

wichtige Einstellungen unter: "RStudio" "Tools" "Global Options" "General"
Empfehlung die folgenden Optionen zu setzen:
Save workspace to .RData on exit: Never
Restore .RData into workspace at startup

sorgen für eine saubere Arbeits-Umgebung ohne alte Daten / Variablen / …

nach dem Öffnen eines Projektes muss aber das zugehörige Skript einmal ausgeführt werden, damit die notwendigen Daten usw. dann wieder im Speicher sind

Beim Schließen des RStudio's wird man gefragt, ob der aktuelle Workspace bzw. das Global Environment gespeichert werden soll.
Diese Daten werden beim nächsten Start von RStudio wieder geladen. Da sind dann auch alle Fehler, … enthalten. Allgemein wird empfohlen die Speicher-Anfrage mit "Nein" zu beantworten.

Installation von (zusätzlichen) Paketen / Library's

"Tools" "Install Dependencies"

praktische Pakte	
•	
•	
• **psych**	zusätzliche Funktionen aus der Welt der Psychologie (und Sozial-Wissenschaften)
•	
•	
•	
•	

Installation eines Paket's über die Konsole, hier mit Installation weiterer – ev. notwendiger / abhängiger - Pakete

```
install.packages("Paketname", dependencies = TRUE)
```

installierte Pakte werden nicht automatisch beim Start von RStudio geladen
laden eines Paket's mit

```
library(Paketname)
```

Nutzung von CodeOcean und OpenHPI-Übungs-Aufgaben im RStudio

in RStudio:

```
install.packages("remotes")
remotes::install_github("openHPI/codeoceanR")
```

unter Linux:

```
install.packages("curl")
install.packages("openssl")
```

und ev. noch weitere Pakete mit z.B.:

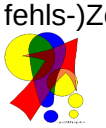
```
sudo apt install libcurl4-openssl-dev
```

Übungsaufgaben händeln

- Aufgabe in CodeOcean (im OpenHPI-Kurs) öffnen
- Aufgabe in einen Ordner downloaden (ev. entpacken (ist optional)) → ist ZIP-Paket
- dannach unbedingt die Aufgabe (die Register-Karte im Browser) schließen (es läuft ein automatischer Aktualisierungs-Mechanismus, der das Händling mit RStudio stört)
- im RStudio: `codeoceanR::rt_create()` ausführen
- Datei oder das ZIP-Paket auswählen und öffnen
- → Bearbeiten der Aufgaben
- jederzeit eine Prüfung / Bewertung (im Hintergrund auf CodeOcean) möglich mit: `rt::score()`
- ganzes Skript mit [Strg] + [⇧] + [S] ausführen → Fehlermeldungen erscheinen dann im Konsolen-Bereich
- Überprüfungen mit Score ist beleibig oft und jederzeit möglich
- am Schluß für die Anrechnung in OpenHPI einmalig übertragen ("Submit")

9.1.1. Arbeiten in der Konsole

Eingabe / Prompt ist >
beginnt eine Zeile mit einem Plus (+), dann gehört diese zur vorhergehenden
das Plus ist hier keine mathematische Operation, sondern das Verknüpfen der beiden (Befehls-)Zeilen



mit einem Plus beginnende Zeile wird beim Ausführen mit [Strg] + [Enter] im Normalfall eine Fehler-hafte Ausgabe erzeugen

[Strg] + [L] .. Löschen der aktuellen Konsole
[Strg] + [ ] ..
[Strg] + [ ] ..

Hilfe(n)

```
help("Begriff")
help(Begriff)
?Begriff
```

in Rstudio: Cursor auf den Begriff und [F1]

Suchen in den Hilfen-Texten
`help.search("`Begriff`")`
`??`Begriff

```
help.start()
```
zeigt offline-Handbücher und –Matrial an

online-Forum:
StackOverflow

weitere Links / Quellen:
https://bookdown.org/brry/course/ressources
https://rdrr.io/snippets online-Arbeits-Umgebung zu R
https://cocalc.com online-Arbeits-Umgebung zu R
https://colab.to/r online-Arbeits-Umgebung zu R
https://r4ds.had.co.nz/ online-Buch: R for Data Science (WICKHAM, H.; GROLEMUND, G.) O'REILLY-Verl.
SHORT, T.; STEIN, J.: Reference Card
→ https://github.com/jonasstein/R-Reference-Card/raw/master/R-refcard.pdf)

Tricks und Tips für die Konsole

Anzahl von Nachkommastellen
```
alteOptionen <- options(digits=4)
alteOptionen # enthält bisherigen Wert
```

zum Zurücksetzen:
```
options(alteOptionen); rm(alteOptionen)
```

Ausgabe in eine Text-Datei (statt in der Konsole (/ Bildschirm)
```
sink("TextdateiName.TXT")
```

Zurücksetzen auf Bildschirm-Ausgabe
```
sink()
```

9.2. Elemente in R

9.2.x. Kommentare

mit **#** beginnen

Schreibung mit Leerzeichen zwischen Operatoren usw. und Operanden kein Problem, verbessert die Lesbarkeir

9.2.x. Zahlen-Schreibung

reele Zahlen mit Punkt als Dezimal-Trenner
wissenschaftliche Zahlen in Exponenten-Schreibweise:
`3.25e+3`

sollen größere Zahlen ausgeschrieben werden, dann lässt sich über:
`options(scipen=Stellenanzahl)`
die auszuschreibende Zahlenlänge festlegen

9.2.x. typische Operatoren:

+ ..
- ..
***** ..
/ ..
^ .. Potenz-Funktion (z.B.: a^b = a^b)
%% .. Modulo / Rest der ganzzahligen Division
%*% .. Matrizen-Multiplikation
%/% .. ganzzahlige Division / ganzzahliger Teiler (ohne den Rest)
< .. Vergleich "Kleiner als"
<= .. Vergleich "Kleiner oder Gleich als"
> .. Vergleich "Größer als"
>= .. Vergleich "Größer oder Gleich als"
== .. Vergleich exakt "Gleich mit"
!= .. Vergleich "Ungleich mit"
| .. ODER / OR /
& .. UND / AND /
! .. NICHT / NOT / Negation
|| .. testet nur das erste Element eines Vektor's mit der ODER-Operation
&& .. testet nur das erste Element eines Vektor's mit der UND-Operation
sqrt() .. (Quadrat-)Wurzel
abs() .. Absolut-Betrag
log() .. natürlicher Logarithmus (!!!)
log10() .. dekadischer Logarithmus

exp() .. Exponential-Funktion (e^x)
factorial() .. Fakultät (x!)
median() .. Angabe des Median's (mittlerer Wert (NICHT Mittelwert!)

9.2.x. Objekte

Objekte fassen Daten zusammen
Benennung mit einem Namen notwendig
Groß- und Kleinschreibung wird unterschieden
entspricht einer Variablen
Variablen-Zuweisung mit **<-** Zuweisungs-Operator (in RStudio Tasten-Kürzel [Alt] + [-])

```
wert <- 30.5
```

Objekt- bzw. Variablen-Namen möglichst mit Kleinbuchstaben beginnen (Empfehlung)
kurze verständliche Benennung in
lowerCamelStandard
oder mit_unterstrich
sachlich auch zulässig: punkt.schreibweise (wird nicht mehr empfohlen, da Verwechslungs-Gefahr mit Bibliotheks-Funktionen besteht)
Zuweisung ist sachlich auch mit = möglich, dieses sollte aber nicht gemacht werden, da Verwechslungs-Gefahr z.B. bei Argument-Namen (bei Funktionen) besteht

Abrufen des Variablen-Wertes mit
```
wert
```

Neu-Belegen z.B
```
wert <- 23.1
wert <- wert + 2.6
```

`ls()` liefert eine Liste der aktuell verfügbaren Objekte (Variablen)
`rm(Variable)` löscht die benannte Variable (das gesamte Objekt)

9.2.x. Konstanten

pi

Achtung!: Konstanten können vom Code überschrieben werden

9.2.x. Funktionen (nutzen)

mit Funktionen lassen sich Abläufe / Arbeiten vereinfachen / effektivieren
s.a. bei den Operatoren
Funktionen besitzen immer ein angeschlossenes Klammer-Paar: ( )
Argumente in der Klammer je nach Funktion, können u.U. auch fehlen
die Argumente können über die Reihenfolge zugeordnet werden
oder über eine Benennung (lässt u.U. das Auslassen von Argumenten zu; es werden dann die Standard-Werte genutzt)
Argument-Namen lassen sich abkürzen, solange diese eindeutig sind

```
log(x=1000, base=10)
log(1000, b=10)
log(1000, 10)
```

Erstellen einer Sequenz
```
sequenz <- seq(1,10)
```

bei Reihen kann auch die Schrittweite mit angegeben werden
```
sequenz <- seq(1,100,10)
```

Anzeige der Sequenz aber erst nach Angabe des Variablen-Namens

erzeugt eine Liste von Zahlen, hier von 1 bis (einschließlich!) 10

9.2.x. Programm-Ablauf-Strukturen

9.2.x.y. Verzweigungen

```
if(Bedingung)
   {
   # then-Teil
   }
```

9.2.x. Arbeiten mit Daten-Dateien

Einlesen einer TXT-Datei mit tabellarischen Daten

```
wetter <- read.table("wetter.txt")
```

wenn 1. Zeile Tabellen-Kopf enthält, dann mit zusätzlichem Argument header

```
wetter <- read.table("wetter.txt", header=TRUE)
```

Zugriff auf Daten aus den Spalten der Tabelle mit $-Zeichen

```
wetter$Regen
```

Einlesen von Daten auch möglich aus XLS, XLSX (Excel-Dateien); CSV (Komma-/Semikolon-separierte Text-Dateien); …

9.2.x. (erste, einfache) Grafiken / Plot's

Empfehlungen für die Diagramm-Auswahl

- Verteilung einer Gruppe → Balken-Diagramm, Histogramm, Dichte-Diagramm
- Vergleichen von Medianen → BoxPlot
- Verteilungen verschiedener Gruppen → BoxPlot
- Zusammenhang zweier Varianzen → Streu-Diagramm
- Zusammenhang nominaler Variablen (Häufigkeiten) → Fliesen-Diagramm

Grafik erstellen mit plot

```
plot(wetter$Sonne,wetter$Temperatur)
```

zusätzliche Argumente für Farben usw., z.B.:

```
plot(wetter$Sonne,wetter$Temperatur, col="orange")
```

pch .. point charakter: 16 .. gefüllter Punkt
type .. beschreibt Diagramm-Typ: "l" .. line (Linie)
lwd .. line width (Linienstärke): Zahl für Punkte
xaxt .. (X-Achsen-Typ: "n" .. nicht / none / ohne Achse

Manipulation der Daten, z.B. anpassen der Datum-Angaben (in Text-Form) in ein reguläres Datum (echtes Datum-Format)

```
wetter$Datum <- as.Date(wetter$Datum)
```

Installation von Paketen
z.B. BerryFunctions

```
install.packages("berryFunctions")
```

Laden des Packages

```
libary(berryFunctions)
```

9.2.x. Vektoren

elementarer Bestandteil größerer R-Skripte

geordnete Menge (ordered set of values)
alle Elemente müssen den gleichen Datentyp haben
vergleichbar mit Array / Feld / geordneter Liste

Definition(en): Vektoren
Vektoren sind eine grundlegende Daten-Struktur in R und beinhaltet eine bestimmte Menge Komponenten / Elemente des gleichen Daten-Typ's.

Erstellen mit der Funktion `c()`

```
vektor <- c(1, 2.4, 5, 9.3)
```

Erweitern / Ergänzen von Vektoren um … / Anhängen von Elemente(n)

```
vektor <- c(vektor, 6.3)
vektor <- append(vektor, 7.2)
vektor[length(vektor)+1] <- 5.7
```

Elemente können benannt werden (named vector) → vergleichbar mit Wörterbüchern / Dictonary's (Namen mit Leerzeichen müssen in Anführungsstriche geschrieben werden → besser vermeiden!):

```
noten( Klaus=12, Monika=13, "Lena Marie"=12)
```

explizites Anzeigen mit Optionen:

```
print()
```

```
print(vektor, digits=2)
```

zeigt die Werte im Vektor mit 2 Nachkommastellen an (sonst Standard 6 Nachkommastellen)

Folge von Zahlen
einfache Folge ganzer Zahlen mit Doppeltpunkt: von`:`bis

```
4:13
```

lässt skalare Multiplikation (also Multiplikation jedes einzelnen Elementes) zu

```
4:13 * 2
## [?]  8, 10, …
```

diese ungewöhnliche Notation des Faktor's (hier die 2) wird in R Recycling genannt
auch mit Vektoren möglich

```
4:13 * c(2,4)
## [?]  8, 16, 10, 20, 12, 24, …
```

bleibt ein Rest bei den Kombinationen, dann erhält man Warnung → Berechnungen / Daten sollten geprüft werden

Wiederholung von Elementen: **rep()** (repeat)

```
rep( Folge, Wiederholungen )

rep(3:7, times=2)
## [?]  3 , 4, 5, 6, 7, 3 , 4, 5, 6, 7
```

mit each Bestimmung, wie oft Elemente der Vektoren für sich wiederholt werden sollen

```
rep(3:5, each=3, times=2)
## [?]  3, 3, 3, 4, 4, 4, 5, 5, 5, 3, 3, 3, 4, 4, 4, 5, 5, 5
```

Sequenz-Funktionen **seq()**

```
seq( Start, Ende, Sprungweite )

seq(from=2, to=6, by=2)
## [?]  2 , 4, 6
```

mit length.out legt R selbst die Schrittweite fest, angegeben wird die Anzahl erwarteter Werte in der Sequenz

```
seq(10, 90, length.out=3)
seq(10, 90, len=3)
```

Zugriff auf Elemente eines Vektor's

```
vektor[ Elementnummer ]
vektor[ Start:Ende ]
vektor[ Indexvektor ]
vektor[ -Elementnummer]   schließt bestimmte Elemente aus
vektor[ -(Start:Ende) ]   schließt bestimmte Element-Gruppe aus
vektor[ Elementnummer ] <- NeuerWert


vekor[3]
vektor[2:4]
vektor[ c(3, 1, 2, 2, 3) ]
vektor[-3]
vektor[3] <- 23
```

Abruf benannter Elemente über den Namen möglich

```
noten(Klaus) <- noten("Lena Marie")
```

9.2.x. wichtige Funktionen für Vektoren

`print()`
verbesserte Anzeige mit diversen Optionen

```
head(Vektor)
```
Anzeige nur einer bestimmen Anzahl (Standard: 6) von Elementen vom Anfang / Kopf des Vektors

```
head(Vektor, AnzahlElemente)
```

```
tail(Vektor)
```
Anzeige nur einer bestimmen Anzahl (Standard: 6) von Elementen vom Ende / Schwanz des Vektors

```
tail(Vektor, AnzahlElemente)
```

```
str(Vektor)
```
zeigt Struktur des Vektor's an
→ Datentyp, [Dimension], erste Elemente

```
class(Vektor)
```
liefert den Datentyp der Vektorelemente zurück
möglich sind
numeric, logical, factor, character

```
names(Vektor)
```
liefert bei einem benannten Vektor die Namen der Elemente (ohne ihren Wert) zurück

lässt auch nachträgliche Benennung z.B. mit Buchstaben zu:
```
names(Vektor) <- LETTERS[1:3]
```
benennt die Elemente mit den ersten 3 Großbuchstaben

```
names(Vektor)[Elementnummer] <- "NeuerName"
```

```
length(Vektor)
```
gibt die Länge eines Vektor's / die Anzahl der Elemente zurück

9.2.x. Funktionen II (Aufbereitung und Analyse von Vektoren)

geg. Vektor:
```
groesse <- c(175, 165, 163, 192, 167, 178, 180, 177, 165)
```

Aufbereitung
Sortierung
```
sort(Vektor)
```

```
sort(groesse)
```

umgedrehte (absteigende) Sortierung

```
sort(Vektor, decreasing=TRUE)

sort(groesse, decreasing=TRUE)
```

Rang-Folge

liefert die Rang-Folge-Positionen zurück an der die (sortierten) Werte in der Original-Mess-Reihe stehen

```
order(Vektor)

order(groesse)
```

z.B. beim Zuordnen von Daten aus einem weiteren Vektor zu den sortierten Daten des 1. Vektor's
geg.:

```
gewicht <- c(76, 69, 84, 92, 73, 73, 83, 63, 84)

gewicht[order(groesse)]
```

```
rang()
```

Entfernen von Duplikaten

mit Beigehaltung der ursprünglichen Reihenfolge

```
unique(Vektor)

unique(gewicht)
```

```
duplicated(Vektor)
```

prüft, ob ein Duplikat vorhanden ist → TRUE oder FALSE
Option: `fromLast=TRUE/FALSE` ..

zufällige Auswahl von Werten (aus einem Vektor)

zufälliges Ziehen ohne Zurücklegen

```
sample(Bereich, size=AnzahlWerte)

sample(0:100, 5)
```

zieht 5 Werte aus dem Bereich von 0 bis (einschließlich) 100

Ziehen mit Zurücklegen

```
sample(Bereich, size=AnzahlWerte, replace=TRUE)

sample(0:100, 5, replace=TRUE)
```

Ziehen aus speziellen Verteilungen

`rnorm(n=10, mean=100, sd=4.0)`	… z.B. aus Normal-Verteilung
`rexp(n=10, rate=1/15)`	… z.B. aus Exponential-Verteilung
`runif(n=10, min=20, max=80)`	… z.B. aus Gleich-Verteilung (uniform)
`rbeta(n=10, shape1=4, shape2=8)`	… z.B. aus Beta-Verteilung
`rpois(n=10, lambda=10)`	… z.B. aus POISSON-Verteilung
`rbinom(n=10, size=100, prob=1/5)`	… z.B. aus Binominal-Verteilung

Runden

normales Runden

```
round(Vektor)

round(groesse)
```

mit angabe der Rundungs-Stelle:

```
round(groesse, digits=-1)
```

rundet auf Zehner

```
round(Vektor, digits=2)
```

rundet auf Hunderstel

Trick: auf 5er-Runden

```
round(Vektor/5)*5
```

weitere Rundungs-Möglichkeiten:

`floor(Vektor)`	.. Abrunden
`ceiling(Vektor)`	.. Aufrunden
`signif(Vektor)`	..
`trunc(Vektor)`	..
`formatC(Vektor)`	..

Berechnungen / Berechnungs-Funktionen

`sin()` .. Sinus-Funktion
`cos()` .. Cosinus-Funktion
`tan()` .. Tangens-Funktion
`sinh()` .. Sinus-Hyberbolicus-Funktion
`cosh()` .. Cosinus- Hyberbolicus-Funktion
`tanh()` .. Tangens- Hyberbolicus-Funktion
`asin()` .. Arcus-Sinus-Funktion
`acos()` .. Arcus-Cosinus-Funktion
`atan()` .. Arcus-Tangens-Funktion
`asinh()` .. Arcus-Sinus-Hyberbolicus-Funktion
`acosh()` .. Arcus-Cosinus- Hyberbolicus-Funktion
`atanh()` .. Arcus-Tangens- Hyberbolicus-Funktion
`gamma()` ..
`lgamma()` ..

String- / Text-Bearbeitung
siehe auch: → 9.2.x. Arbeiten mit Zeichenketten
`paste()` .. Einfügen / Ergänzen von Zeichenketten / String-Vektoren / …
`substr()` .. Finden eines Teil-String's
`nchar()` .. Bestimmen der Anzahl Zeichen in einem String
`strsplit()` .. Aufteilen einer Zeichenkette in Teil-String's auf der Basis von Trennzeichen
`toupper()` .. in Groß-Buchstaben wandeln
`tolower()` .. in Klein-Buchstaben wandeln
`sub()` ..

allgemeine Kennzahlen
Minimum, Maximum
```
min(Vektor)
max(Vektor)

min(groesse)
max(groesse)
```

Wertebereich
```
range(Vektor)

range(groesse)
```

Mittelwerte
arithmetrisches Mittel
```
mean(Vektor)

mean(groesse)
```

Achtung!: besondere Daten-Verarbeitung bei logischen Operationen! (→ Logik und logische Funktionen)

Median
mittlerer Wert, wobei die Hälfte der anderen Werte größer (, und die andere eben kleiner) sind; Ausreißer-unabhängig
mittlerer Wert in der geordneten Mess-Reihe
```
median(Vektor)

median(groesse)
```

Qualitäts-Kennzahlen
Varianz
```
var(Vektor)
```

```
var(groesse)
```

Standardabweichung (standard deviation)
Wurzel aus der Varianz; beschreibt die Streuung der Einzelwerte um den Mittelwert
```
sd(Vektor)

sd(groesse)
```

absolute Abweichung vom Median (median absolute deviation)

```
mad(Vektor)

mad(groesse)
```

Quantile
gibt die Werte der sortierten Mess-Reihe, wobei 0, 25, 50, 75 und 100% der Einzel-Werte kleiner sind (das 0%-Quantil ist das Minimum, 100%-Quantil etspricht eben dem Maximum, das 50%-Quantil ist gleich dem Median)
```
quantile(Vektor)

quantile(groesse)
```

für eigene Quantil-Parameter:
z.B. für 95%
```
quantile(Vektor, probs=Grenze)

quantile(groesse, probs=0.95)
```

Schätzungen
Wahrscheinlichkeits-Dichte
```
density()
```

lineare Regression
```
lsfit()
```

zusammengefasste Analyse (wichtigste Kennzahlen)
enthält Minimum, Maximum, 1. und 3. Quantil, Median und Mittelwert
gibt auch Anzahl fehlender Werte (NA-Werte) an
```
summary(Vektor)

summary(groesse)
```

weitere Funktionen zu Vektoren
`sum(Vektor)` .. bildet Summe über alle Elemente des Vektor's

`prod(Vektor)` .. bildet Produkt über alle Elemente des Vektor's
`cumsum()` .. bildet kummulierte Summe
`cumprod()` .. bildet kummuliertes Produkt
`cut()` .. klassifiziert die Werte eines nummerischen Vektor's in Kategorien

```
Daten$neueSpalte <- cut(x=Daten$Spalte, breaks=c(Grenzen),
                        labels=c(Bezeichnungen))
```

breaks muss die untere und obere Grenze, sowie dazwischen die Trenn-Grenzen enthalten; labels (muss um 1 kleine als breaks sein, da die Bereiche beschriftet werden)

`tabulate()` ..

Achtung!: besondere Daten-Verarbeitung bei logischen Operationen! (→ Logik und logische Funktionen)

Bsp:

```
wetter$warm <- cut(x=wetter$Temperatur, breaks=c(-100,10,20,100), labels=c("kalt", "mittel", "warm"))
```

`any(Vektor)` .. bildet logische Summe über alle Elemente des Vektor's
→ logische Operationen (→ Logik und logische Funktionen)

Funktionen für Matrizen
`t(Matrix)` .. tranponiert die Matrix
`var(Matrix)` .. gibt Kovarianz der Matrix zurück

9.2.x. eigene Funktionen erstellen

```
FunktionsName <- fubction([Argument], {Argument ,} )
    {
    …
    return(RückgabeWert)
    }
```

Aufruf, wie oben beschrieben
Argument können Namen bekommen
Identifizierung der Argumente über die Reihenfolge in der Definition oder über die Namen (abgekürzte Namen sind beim Aufruf möglich, solange sie eindeutig sind)
mit = lassen Default-Werte festlegen

```
loeseQuadGl_PQ <- function(p=0, q=0) # Vorbeleg. für y = x^2 = x^2 + px +q
    {
    wurzel <- sqrt(p^2/4 - q)
    pHalbe <- p/2
    c(-pHalbe+wurzel, -pHalbe-wurzel)
    }
```

`return` kann weggelassen werden, dann wird das Ergebnis der letzten Quell-Code-Anweisung zurückgegeben

alle anderen Variablen / Objekte sind nur temporär in der Funktion existent / gültig / nutzbar
die geschweiften Klammern (**{ }**) für den Funktions-Körper-Block können weggelassen werden, wenn nur eine Anweisung in der Funktion ausgeführt wird
üblicherweise die Anweisung gleich mit in die Definitions-Zeile schreiben

```
normalisiere <- function(x)  (x - min(x)) / (max(x) - min(x))


meinMAD <- function(v) {dev <- v-median(v); median(abs(dev))
```

zufällige Auswahl eines Element's aus einer vorgebenen Liste

```
auswahl <- function() {
    liste <- c("Kopf", "Zahl")   # Liste der verfügbaren Elemente
    return(liste[sample(1:length(liste),1)]
```

N-tes Maximum finden:

```
nthMax <- function(x,n=1)
  {
  sortX <- sort(x)
  return(sortX[length(sortX)-n+1])
  }
nthMax(1:9)     # 9
nthMax(1:9, 2) # 8
nthMax(1:9, 3) # 7
```

9.2.x. Logik und logische Funktionen

Wahrheitswerte: TRUE und FALSE auch möglich T und F (diese sind aber überschreibar!)
intern und für Rechnungen wird TRUE durch eine 1 und FALSE durch eine 0 repräsentiert
mit der Funktion `as.numeric()` lässt sich die Anzeige in Zahlen erzwingen

logische Operationen lassen sich auch auf Vektoren anwenden, dann wird die logische Operation auf jedes Element angewendet und ein Vektor mit Wahrheitswerten zurückgeliefert

Arbeits- / Beispiel-Vektor

```
werte <- c(2, 4, 8, 12, 19)
```

`which(Bedingung)`
liefert die Stellen / Positionen / Indizes der Werte zurück, die dem logischen Ausdruck entsprechen

```
which(werte < 10)
[?]  1  2  3

which(LETTERS > "M")


which.max(Vektor)
```

liefert die **1.** Position(en) / Stelle / den **1.** Index für den Maximal-Wert aus einem Vektor

`any(Bedingung)`
testet ob die Bedingung mindestens 1x Wahr (TRUE) ist

```
any(werte == 10)
```

`all(Bedingung)`
testet ob die Bedingung für **alle** Elemente Wahr (TRUE) ist

```
all(werte > 0)
```

`sum(Bedingung)`
summiert nur die (nummerischen) Wahrheits-Werte des Vergleich's
entspricht also einer Zählen-Funktion

`mean(Bedingung)`
bildet aus den (nummerischen) Wahrheits-Werte des Vergleich's den Mittelwert (**nicht** aus den eigentlichen Elementen!)

Anwendung logischer Operationen zum Filtern (von Werten aus Vektoren)

```
gefilterteWerte <- werte[werte <10]

posMittel <- function(zahlen)
  {
  posZahlen <- zahlen[!(zahlen < 0)]
  mean(posZahlen)
  }
posMittel(-3:5) # soll 2.5 sein
```

mehrere zu betrachtende / verarbeitende Vektoren müssen gleichlang sein!

```
namen <- c("x1", "x2", "x3", "x4", "x5")
gruppe <- c("P", "T", "T", "P", "T")

namen[werte > 5]
[?]  "x3"  "x4"  "x5"

werteGruppe <- werte[gruppe == "T"]
```

Finden des 2. größten Wertes

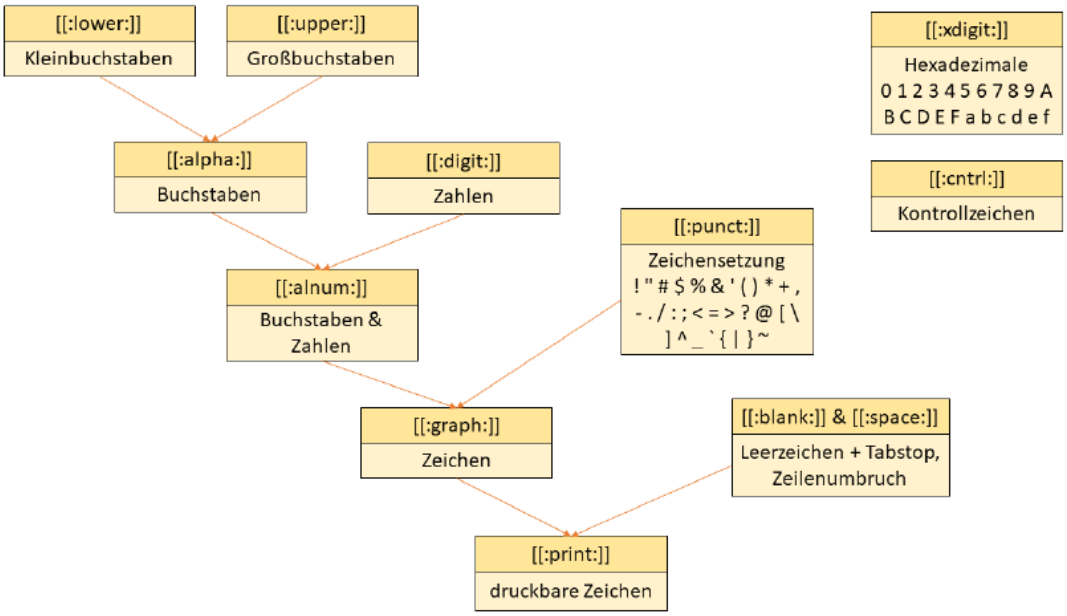
```
zweitesMaximum <- function(vektor) {
    hilfsVek <- vektor
    lokMin <- min(hilfsVek)
    hilfsVek[which.max(hilfsVek)] = lokMin
```

```
  erg <- max(hilfsVek)
return(erg)
}
```

9.2.x. Arbeiten mit Zeichenketten

Zeichenketten werden durch einfache oder doppelte Anführungs-Zeichen (' ' oder " ") begrenzt
Darstellung besonderer Zeichen:
`\n` … Zeilenumbruch (new line)
`\\` … Backslash selbst
`\\.` … Punkt selbst
`\"` … Anführungs-Zeichen
`\'` … einfaches Anführungs-Zeichen, Hochkomma
`\U{0B00}` … Unicode-Zeichen, hier: °
`\t9` … Tabstop

in R definierte Sammlungen von Zeichen



Q: open.hpi.de (Kurs: R-Programmierung)

`class(Variable)`
liefert den Datentyp der Variable zurück
ist bei Zeichenketten `character`

`nchar(Zeichenkette)`
liefert die Anzahl der Zeichen in der Zeichenkette zurück

`tolower(Zeichenkette)`
wandelt die Groß-Buchstaben einer Zeichenkette in Klein-Buchstaben um

`toupper(Zeichenkette)`

wandelt die Klein-Buchstaben einer Zeichenkette in Groß-Buchstaben um

`substr(Zeichenkette, start=StartPosition, stop=EndPosition)`
liefer einen Teil-String / eine Teil-Zeichenkette von der Start-Position bis zur En-Position zurück

`cat(Zeichenkette)`
ausgeben einer Zeichenkette

`paste(Zeichenkette, Start:Ende)`
hängt die Werte des Vektor's (von Start bis Ende) an die Zeichenkette (mit Leerzeichen (als Standard-Separator)) an

`paste(Zeichenkette, Start:Ende, sep="..")`
hängt die Werte des Vektor's (von Start bis Ende) an die Zeichenkette und dem direkt folgenden Separator `sep` an

`paste("", 1:10)`
erzeugt z.B. Vektor aus Zeichenketten mit den Zahlen 1 bis 10, ev. mit führenden Leerzeichen formatiert
`[?]  " 1"  " 2" …`

```
buchstaben <- paste("Buchstabe", LETTERS, sep="_")
```
erzeugt Vektor mit Groß-Buchstaben hinter einem Unterstrich
`[?]  "Buchstabe_A"  "Buchstabe_B"  …`

`paste0(Zeichenkette, Start:Ende)`
hängt die Werte des Vektor's (von Start bis Ende) an die Zeichenkette (ohne Leerzeichen) an

`paste0(Zeichenkette, Start:Ende, collapse="- -")`
hängt die Werte des Vektor's (von Start bis Ende) an die Zeichenkette (ohne Leerzeichen) an alle Teil-Wörter werden über den collapse-String miteinander verbunden

`paste0("", 1:10)`
erzeugt z.B. Vektor aus Zeichenketten mit den Zahlen 1 bis 10 ohne führende Leerzeichen
`[?]  "1"  "2" …`

Nutzung für die Kombination von Speicher-Pfad und Datei-Namen

```
pfad <- "~/meinOrdner/meineRDaten"
datei <- "Daten.txt"

paste0(pfad, "/", datei)
paste(pfad, datei, sep="/")
```

`toString(TeilStringVektor)`
erzeugt einen Komma-getrennten

`strsplit(Zeichenkette, split=" ")[[1]]`
zerlegt eine Zeichenkette an den Trennzeichen (split) und liefert eine Liste der "Worte" / aufgespliteten Teil-Strings zurück
doppelte eckige Klammern (`[[ ]]`) zur Auswahl eines Element's aus der Liste

`match(Element, Zeichenkette)`
sucht nach eigenständigen Vorkommen (in Trennzeichen eingeschlossens) des Element's in der Zeichenkette

`%in%`
logische Funktion, prüft, ob eine Teilstring in einem anderen String vorkommt

`grep(Element, ZeichenkettenVektor, value=TRUE)`
(**g**lobal search for a **r**egular **e**xpression and **p**rint out matched lines; kurz: **g**lobal/**r**egular **e**xpression/**p**rint)
findet die Vorkommen des Element's in der Zeichenkette
lässt reguläre Ausdrücke (regular expression, regex) für die Suche zu
mit Option `value=TRUE` wird das ganze Wort ausgegeben, das den Teilstring / Element enthält
mit der Option `ignore.case=TRUE` bestimmt man, dass Groß- und Kleinschreibung nicht beachtet wird
mit der Option fixed=TRUE kann man die Suche über reguläre Ausdrücke abschalten, es wird dann nach dem angegebenen String gesucht (z.B. wenn man nach Texten mit den Meta-Symbolen von regulären Ausdrücken suchen möchte)
zusätzliche Möglichkeiten zur speziellen Suche über `startsWith` und `endsWith`

Suche auch nach Anführungs-Zeichen
`"??? \" ???"` oder kürzer `'??? " ???'`

Nutzung von regulären Ausdrücken
`^F` … besagt, dass der Ergebnis-String mit F beginnen muss
`ion$` … besagt, dass der Ergebnis-String mit ion enden muss
`\\.` … steht für echten Punkt
`.` … steht für ein beliebiges Zeichen
`.{3}` … steht für drei beliebige Zeichen
`.*` … steht für beliebig viele beliebige Zeichen
`x|y` … steht für entweder x oder y
`[dkr]` … steht für eine mögliche Auswahl von Zeichen (hier: d, k und r)
`[^dkr]` … steht für ein Nicht-Auftreten von Zeichen (hier: d, k und r)
`[a-g]` … steht für eine mögliche Auswahl von Zeichen (hier von a bis g)
`x?` … Zeichen kann, muss aber nicht vorhanden sein (Auftreten also 0 oder 1x)
`x*` … Zeichen kann mehrfach, muss aber nicht vorhanden sein (Auftreten also 0 oder x-mal)
`x{3}` … Zeichen muss 3x vorkommen
`x{3,}` … Zeichen muss 3 oder mehrfach / öfter vorkommen

`x{2,4}` … Zeichen muss 2 oder 4-fach vorkommen

`regexpr(Ausdruck, Vektor)`
liefert die (erste) Position des Ausdruck's innerhalb jedes Elementes im Vektor zurück, mit -1 wird gekennzeichnet, dass Ausdruck nicht im Element vorkommt
z.B.: `erg <- c(regexpr("???", wortvektor))`

`regexpr(Ausdruck, Vektor)`
liefert die (erste) Position des Ausdruck's innerhalb jedes Elementes im Vektor zurück, mit -1 wird gekennzeichnet, dass Ausdruck nicht im Element vorkommt
z.B.: `erg <- gregexpr("???", wortvektor)`

`grepl(Teilstring, ZeichenkettenVektor)`
prüft für jedes Element im Text-Vektor (Zeichenketten-Vektor), ob derTeilstring enhalten ist
liefert die logischen Werte zurück

Existenz eines Wortes in einem Vektor

```
existiertWort <- function(wort, liste) {
    return(any(grepl(wort, liste)))
    }
```

Existenz eines Wortes in einem Vektor unanbhängig von Groß- oder Klein-Schreibung

```
wort_existiert_case <- function(wort, liste) {
  return(any(grepl(toupper(wort), toupper(liste))))
  }
```

`sub(pattern="Muster", replacement="Ersatz", x=Vektor)`
ersetzt (substituiert) in einer Zeichenkette alle Übereinstimmungen / Muster durch einen neuen TeilString (aber nur das 1. Auftreten des Muster's im Element)

`gsub(pattern="x/y", replacment="X/Y", x=ZeichenkettenVektor)`
ersetzt (substituiert) in einer Zeichenkette alle Übereinstimmungen / Muster durch einen neuen TeilString (nun jedes Auftreten des Muster's im Element)

ersetzen alle Kleinbuchstaben "o" durch Großbuchstaben "O"

```
gsub(pattern="o", replacement="O",
     x="oh, das ist ein grosses Hallo World!")
```

```
textVerarbeiten <- function(z)
  {
  hStr <- z[!(startsWith(z, "fake_"))] #Entfernen von passenden Elementen
  return(gsub(pattern="dummy",  replacement="--",  hStr))  #  Ersetzen  von
Teilstring
  }
textVerarbeiten(c("Sachen","mit","fake_zeug","und dummy Code"))
# Soll sein: "Sachen",  "mit",  "und -- Code"
```

9.2.x. Kategorien

besonderer Datentyp, auch kategoriale Variablen
auch Faktoren genannt
geeignet für Klassifizierungs-Aufgaben, Häufigkeits-Analysen
intern als Zahlen repräsentiert, deshalb auch Ausgabe / Anzeige mit `as.numeric(Kategorie)` möglich
zu beachten sind die unterschiedlichen Zahlen-Werte, wenn Zahlen die Level's von Kategorien bilden
z.B. die Zahlen (Level's) 100, 200, 400, 1000
werden intern durch die Zahlen 1, 2, 3 und 4
repräsentiert
`as.numeric(Kategorie)` liefert die Umsetzung der Level's in Zahlen
erst `as.numeric(as.factor(100, 200, 400, 1000))` liefert die interne Index-Nummer

Definition über Schlüsselwort factor
```
factor(Vektor)

factor(c("Tier", "Tier", "Bakterium", "Pflanze", "Tier"))
[?]  Tier  Tier  Bakterium  Pflanze  Tier
Levels: Bakterium  Pflanze  Tier
```

die Levels sind normalerweise alphabetisch geordnet, wenn das nicht geünscht ist, muss die Definition mit der Level-Option erfolgen
```
factor(c("Tier", "Tier", "Bakterium", "Pflanze", "Tier"),
      levels=c("Tier", "Pflanze","Bakterium"))
```

Zuweisung zu einer Variable
```
kategorieLebewesen <- factor(c("Tier", "Tier", "Bakterium", "Pflanze",
"Tier"))
```

Erkunden der Klasse:
```
class(kategorieLebewesen)
[?]  "factor"
```

Hinzufügen eines Level's
```
levels(neueKategorie) <- c(levels(neueKategorie), Level)

neueKategorie <- factor(c(as.character(neueKategorie), Level))
```

eingebaute Daten-Bestände in R:
`state.region` ... amerikanische Staaten (mit Regionen als Kategorien)

Anzeigen der Level's einer Kategorie
```
levels(Kategorie)
```

Anzeige der Level's und der auftretenden Häufigkeit:
table(Kategorie)
zeigt Kategorie, die Levels und die Häufigkeiten (in 3 Zeilen untereinander)

```
table(kategorieLebewesen)
```

Achtung!: table ist also keine klassische Funktion für irgendwelche Tabellen-Operationen od.ä., sondern die Erzeugung einer speziellen Häufigkeits-Tabelle für Kategorien

```
table(Kategorie, dnn=NULL)
```
Option `dnn=NULL` sorgt dafür, das der Name eines Faktor's (Dimension) nicht angezeigt wird

Auslesen der Kategorien zu einem Faktor
```
names(table(Kategorie))
```
werden dann als Vektor von Zeichenketten bereitgestellt

Erstellen einer Kreuz-Tabelle aus zwei Kategorien
```
table(Kategorie1, Kategorie2)
```

Gruppen-weises Anwenden einer Funktion
`tapply()`
tagged (grouped) apply
```
tapply(X=Kategorie1, INDEX=Kategorie2, FUN=Funktion)
tapply(X=Werte, INDEX=Kategorie, FUN=Funktion)
```
`INDEX` … Gruppierungs-Merkmal
`FUN` … anzuwendende Funktion

```
meanCharLen <- function(x) mean(nchar(x))

meanCharLen(state.name)

tapply(X=state.name, INDEX=state.region, FUN=meanCharLen)
```

Funktion kann auch direkt in die `tapply`-Funktion (als anonyme Funktion) hineingeschrieben werden
```
tapply(X=state.name, INDEX=state.region, FUN= function(x) mean(nchar(x))
```

Code-Beispiel:
```
personenMitFarbe <- function(namen, farben)
x)    {
    id <- wich.max(table(farben))
    col <- names(table(farben))[id]
    namen[col == farben]
    }


personenMitFarbe <- function(namen, farben)
    {
    id <- wich.max(table(farben))
```

```
    col <- names(id)
    namen[col == farben]
    }



geraete<- c("PC", "Laptop", "PC", "PC", "Tablet", "PC", "Tablet")
tapply(1:7, geraete, sum)



gruppenMedian <- function(values, groups)
  {
  return(tapply(X=values, INDEX=groups, FUN=function(x) median(x)))
  }
gruppenMedian(1:14,
c("A","B","A","C","U","S","B","A","R","A","C","C","U","S"))
```

9.2.x. Pakete

gesammelter Quell-Code für spezielle Aufgaben
Funktions-Sammlungen
enthalten üblicherweise den Quell-Code, Anleitungen und Beispiele
meist auf CRAN (Comprehensive R Archive Network, → cran.org)
Pakete müssen immer geprüft werden!
Hinweise auf sichere / funktionierende Pakete:

- höheres Alter
- redelmäßige Updates / letztes Update erst kürzlich
- aktive Community
- viele Download's
- …

mitgelieferte / vorinstallierte Pakete

- **base**
- **datasets**
- **utils**
- **grDevices**
- **graphics**
- **stats**
- **methods**
-
-
-
-

sind sofort verfügbar und deren Funktionen brauchen nicht mittels Paketnamen aufgerufen werden

weitere Pakete sind vorinstalliert, aber noch nicht geladen
müssen quasi manuell zur Arbeits-Umgebung / Quellcode hinzugefügt werden (→ Nutzen eines Paket's / der Funktionen etc. aus einem Paket)

direkt nachladbare Pakete

- **compiler**
- **grid**
- **parallel**

- **splines**
- **tcltk**
- **tools**
-
-
-

Installation eines Paket's

```
install.packages("PaketName")
```

i.A. nur einmalig notwendig
kann jeder Nutzer mit seinen Rechten
ev. Abhängigkeiten müssen zuerst installiert werden
Paket ist dann in der lokalen Bibliothek (Library) verfügbar

Vorabinformation z.B. über die Seite → www.rdocumentation.org möglich, dort sind die Dokumentationen gesammelt
Pakete werden dabei nicht installiert

Updaten der Pakete

```
update.packages()
```

Entfernen eines Paket's

```
remove.packages("PaketName")
```

Nutzen eines Paket's / der Funktionen etc. aus einem Paket

Laden eines Paket's aus der lokalen Bibliothek (Library)

```
library("PaketName")
library(PaketName)
```

muss für jede R-Sitzung gemacht werden!
am Besten gleich mit in das R-Skript mit aufnehmen
bei gleichnamigen Funktions-Namen in mehreren Paketen dominiertt die zuletztgeladene Funktion die anderen Versionen
deshalb ist ein Funktions-Aufruf mit Paketname üblich und zu empfehlen

```
PaketName::Funktion()
```

Prüfen im Code, ob ein Paket installiert ist und gegebenenfalls das Paket installieren

```
if(!requireNamespace("PaketName", quietly=TRUE))
    install.packages("PaketName")
```

z.B. "berryFunctions"

```
if(!requireNamespace("berryFunctions", quietly=TRUE))
    install.packages("berryFunctions")

x <- 1:10
y <- c( 10 Werte )
berryFunctions::linReg(x, y, pos1="topleft")
```

Option `pos1` bestimmt die Position für die Beschriftung

```
berryFunctions::funSource()
```

lädt einen Quellcode von github herunter und zeigt ihn mit Synthax-Highlighting an

für die klassischen Funktion aus dem R-Base-Paketen ist
die Quelle z.B.: github.com/wch/r-source/src/library/base (stats, parallel, …)
für die CRAN-Pakete: github.com/cran

sollte für einen Quellcode UseMethods angezeigt werden, dann kann man sich mit methods(Funktion) die passenden Methoden für unterschiedlichen Datentypen usw. anzeigen lassen

steht bei der Quellcode-Anzeige ein `.Primitives` , dann handelt es sich um eine integrierte Funktion, die in der Programmiersprache C erstellt wurde

9.2.x. Daten in R

Tabellen / data.frames

Tabelle → data.frame
Zeile → observation (Datensätze, Beobachtungen)
Spalte → Variables (Zellen)

Spalten haben immer einen Daten-Typ (z.B. numeric, character, factor, logical, …)

Definition(en):

Anlegen eines data.frames
übliche Abkürzung / Variable / Bezeichnung df

```
df <- data.frame(SpaltenName=Werte { , SaltenName=Werte } )
```

Bsp:

```
bdf <- data.frame(Zahlen=11:14, Buchstaben=letters[1:4], Booleans=(1:4)>2)
```

Struktur eines data.frames anzeigen lassen
structur

```
str(DataFrame)
```

Bsp:

```
str(bdf)
```

nrow()

ncol()

Anzeige der Spaltennamen
über `colnames` (column names)

```
colnames(DataFrame)
```

auch geeignet, um die Namen der Spalten (nachträglich) zu ändern

```
colnames(DataFrame)[ SpaltenIndex ] <- "neuerSpaltenName"
```

Bsp:

```
colnames(bdf)[2]="Zeichen"
```

Hinzufügen von Namen für die Zeilen (Datensätze)

```
rownames(DatenFrame) <- c( Bezeichnungen )
```

Zugriff auf Zeile über Zeilennamen

```
DatenFrame["ZeilenName",  ]
```

ergibt praktisch den genannten Datensatz

erste Daten-Analyse

mit summery
liefert Kennwerte zu den enthaltenen Daten
Ausgabe ist spezifisch / unterschiedlich für den Daten-Typ

```
summery(DataFrame)
```

Bsp:

```
summery(bdf)
```

Auswahl einzelner Daten
über den Index

```
DataFrame[ ZeilenIndex, SpaltenIndex ]
```

Zählung beginnt jeweils bei 1
es können Indizies weggelassen werden, wenn die ganze Zeile / Spalte genutzt werden soll

```
DataFrame[  , 3]
DataFrame[ 3, ]
DataFrame[2:4, ]
DataFrame[c(4,2), ]
```

bei Gruppen, die über die Doppelpunkt-Definition erstellt werden, können durch umgedrehte Reihen (z.B: 3:1) auch die Reihenfolgen der Spalten geändert werden
Ausschließen von Zeilen / Spalten über ein vorgesetztes Minus-Zeichen vor dem Index

Bsp:

```
bdf[1,2]
bdf[-2,2]
```

Auswahl über den Spalten-Namen (Spalten-Bezeichnung / Spalten-Kopf)

```
DataFrame[ , "SpaltenName" ]
DataFrame$SpaltenName
```

Bsp:

```
bdf[ , "Zahlen"]
dbf$Zahlen
```

Filtern von Daten

```
DataFrame[DataFrame$SpaltenName==Bedingung,  ]
```

Bsp.:

```
bdf[bdf$Booleans==TRUE,  ]
```

Ändern von Werten in DataFrames

```
DataFrame$SpaltenName <- 3:6
```

wird ein neuer SpaltenName angegeben, dann wird die DatenTabelle um die Spalte ergänzt (entspricht → Hinzufügen einer Spalte)

```
DataFrame$neuerSpaltenName <- c( Werte )
```

Anzahl der Werte müssen zur alten Datenstruktur passen!

Bsp:
```
bdf$werte <- c(10, 20, 30, 20)
```

Löschen einer Spalte
indem diese auf `NULL` gesetzt wird
```
DataFrame$SpaltenName <- NULL
```

Lösen der gesamten DatenTabelle
```
DataFrame <- NULL
```

Hinweise:
`nrow(DataFrame)` liefert z.B. `NULL` zurück, wenn DataFrame ein Vektor ist
`NROW(DataFrame)` zeigt die Länge des DataFrame, wenn DataFrame ein Vektor ist
sind keine Namen für die Spalten vergeben, dann können die Bezeichner X oder V (für Variable) lauten (ev. mit Nummerierung)

Konvertieren einer Matrix in einen DataFrame
```
DataFrame <- as.data.frame(Matrix)
```

`tabelle[ , Spalte, drop=FALSE]` liefert ein data.frame (/eine Tabelle) (statt eines Vektor's)

`DatenFrame[ , "SpaltenName"]` liefert einen Vekor mit den Daten aus der Spalte zurück
`DatenFrame[ , "SpaltenName", drop=FALSE)` liefert eine Tabelle (data.frame) mit der Spalte zurück^
???
`DatenFrame["SpaltenName"]` liefert eine Tabelle (data.frame) mit der Spalte zurück

Matrizen

praktisch ist eine Matrix ein Vektor mit der Information, wie dieser in zwei Dimensionen angeordnet werden soll

Matrix erstellen
```
matrix(data=Werte, nrow=ZeilenAnzahl, ncol=SpaltenAnzahl)
```

füllt die Werte Spalten-weise hinzu, wenn Zeilen-weise gewünscht ist, dann kann man die Option `byrow=TRUE` benutzen

Bsp.:

```
mtx <- matrix(data=1:6, nrow=2, ncol=3, byrow=TRUE)
```

Erstellen mit Funktionen

```
matrix(rep(0:1, each=3), ncol=3)    # hier: Wiederholungs-Funktion rep()
```

Erstellen einer Matrix aus einem data.frame / Konvertieren eines DatenFrame's in eine Matrix

```
Matrix <- as.matrix(DatenFrame)
```

Informationen über Attribute einer Matrix

`dim(Matrix)` … liefert ZeilenAnzahl und SpaltenAnzahl zurück
`nrow(Matrix)` … liefert ZeilenAnzahl zurück
`ncol(Matrix)` … liefert SpaltenAnzahl zurück
`length(Matrix)` … liefert die Anzahl der Werte in der Matrix zurück (, also praktisch ZeilenAnzahl * SpaltenAnzahl)

`class(Matrix)` … liefert die Klassen der Matrix zurück, das sind "matrix" und "array" (Array ist eine Überklasse von Matrix)

setzt man mit `dim(Matrix) <- NULL` die Diemension einer Matrix als `NULL` , dann wird aus der Matrix ein Vektor mit den Werten der Matrix

Filtern von Werten aus einer Matrix (in einen Vektor)

```
Vektor <- Matrix(which(Bedingung))
```

Bsp.:

```
ergVektor <- DatenMatrix(which(DatenMatrix < 10))
```

Operationen über Matrizen

alle Elemente einer Matrix müssen den gleichen Daten-Typ haben (s.a. Vektoren → 9.2.x. Vektoren)
praktisch also auch keine `NA`-Werte zugelassen

Element-weise Multiplikation

```
Matrix * Faktor
```

andere Rechen-Operation genauso

Matrixen-Multiplikation

```
Matrix1 * Matrix2
```

andere Rechen-Operationen genauso

Matrizen-Multiplikation
```
Matrix1 %*% Matrix2
```
multipliziert die Spaltensumme der 1. Matrix mit den Werten aus der 2. Matrix

Transponieren / Spiegeln eine Matrix
tauscht Zeilen und Spalten
```
t(Matrix)
```

Benennen der Spalten:
```
colnames(Matrix) <- c( Bezeichnungen )
```

Benennen der Zeilen:
```
rownames(Matrix) <- c( Bezeichnungen )
```

Zugriff auf einzelne Elemente über Indizies
```
Element <- Matrix[Zeile,Spalte]
Matrix[Zeile,Spalte] <- c(Wert)
```

Auswahl einer Zeile
```
Matrix["ZeilenName",  ]
```

Auswahl einer Spalte
```
Matrix[  , "SpaltenName" ]
```

Überprüfen, ob eine Zeile wirklich von einem Datentyp ist – also ein Vektor ist
is.vector(Matrix["ZeilenName",])
ergibt `TRUE` oder `FALSE`

Hinweis:
wird durch eine Zuweisung ein Datum mit einem anderen Datentyp in die Matrix eingeführt, dann wird der Datentyp der gesamten Matrix geändert / angepasst

Funktionen für Matrizen
`rowSums(Matrix)` … berechnet die ZeilenSumme über die Werte (einer Zeile)
`rowsum(Matrix)` …
`rowMeans(Matrix)` … berechnet den arithm. Mittelwert über die Werte (einer Zeile)

`colSums(Matrix)` … berechnet die SpaltenSumme über die Werte (einer Spalte)
`colsum(Matrix)` … `colMeans(Matrix)` … berechnet den arithm. Mittelwert über die Werte (einer Spalte)
`apply(Matrix, MARGIN=`Orientierung`, `FunktionsName`)` … wendet die Funktion auf alle Werte der Spalte an; die Anwendungs-Richtung wird über die Option `MARGIN` bestimmt, wobei 1 die Zeilen-weise und 2 die Spalten-weise Anwendung bestimmt
`apply(Matrix, MARGIN=`Orientierung`, FUN=`FunktionsName`)`
`apply(Matrix,` Orientierung`,` FunktionsDefinition`)` wobei dann eine "anonyme" Funktion definiert und angewendet wird

Bsp:
```
apply(matrix, 1, function(x) sum(x==2))
apply(matrix, 1, FUN=function(x) cat(toString(x), " - ")
```

na.strings="

9.2.x. Daten einlesen

allgemeingültige Struktur:
```
Daten <- read.table(file="Dateiname.txt")
Daten <- read.table("Dateiname.txt")
```

Hinweise:
das Daten-Einlesen unbedingt überprüfen → häufigste Fehler-Quelle für fehlerhafte Auswertungen
z.B.:
```
str(Daten)
summary(Daten)
```

statt einem Dateinamen, kann auch eine URL angegeben werden (Daten müssen nicht vorher heruntergeladen werden)
wichtige Optionen / Argumente für die read.table-Funktion:
`header` … haben die Daten / Spalten Überschriften? [TRUE | FALSE]; praktisch: soll die erste Zeile als Spalten-Überschriften genutzt werden?
`dec` … welches Zeichen wird als Dezimal-Trenner (decimal) benutzt → in Deutschland üblich "**,**", aber in den meisten Daten-Beständen eher "**.**" (engl./amerik. Zahlen-Darstellung)
`sep` … welches Zeichen ist der Trenner (Separator) zwischen den Elementen in einer Zeile? → üblich "**\t**" für Tabulator, aber auch "**;**" oder "**,**" für z.B. CSV-Dateien; Standard ist ein Leerzeichen, dann kann `sep` auch weggelassen werden
`fill` … füllt bei unvollständigen Zeilen / Datensätzen, die fehlenden Daten-Elemente mit NA; Standard ist FALSE
`skip` … bestimmt die Anzahl Zeilen, die am Anfang ignoriert werden sollen, z.B. weil sie meta-Daten, Kommentare od.ä. enthalten
`comment.char` … welches Zeichen ist der Identifizierer / Starter für Kommentare; Standard ist "**%**"
`na.strings` … welches Einträge sind für Fehlwerte einzusetzen; z.B.: `na.strings=c(-999, "NN")` …

`stringsAsFactors` … bestimmt, ob Zeichenketten als Kategorien (Faktoren) umgesetzt werden sollen (ab Version 4 bei R: Standard: FALSE)
`text` … legt einen kleinen Beispiel-Datensatz fest

wird ein **Fehler** (error) angezeigt / geworfen, dann bricht die Funktion das Einlesen ab → Daten stehen dann **nicht** vollständig und exakt importiert zur Verfügung
wird eine **Warnung** (warning) angezeigt, dann ist das Einlesen vollständig durchgelaufen, aber es hat Probleme mit einzelnen Daten-Elementen oder –Sätzen gegeben → diese sollten geprüft werden, sonst könnte die weitere Daten-Verarbeitung Fehler-behaftet sein
speziellere Datei-Einlese-Funktionen
auf bestimmte Datei-Typen spezialisiert
rufen praktisch immer `read.table` auf

`read.csv()` … Einlesen von Komma-separierten Text-Dateien
`read.fwf()` … Einlesen von Dateien mit fixen Spaltenbreiten (fixed width formated data)
`readLines()` … Einlesen ganzer Zeilen als Vektoren
`scan()` … Grundfunktion zum Einlesen von Daten

`readxl::read.excel()` … Einlesen von EXCEL-Dateien (XLS, XLSX); benötigt aber extra Paket: readxl
`readBin()` … Einlesen von binären Dateien (z.B. aus älteren / proprietären Programmen)
`raster::raster()` … Einlesen von Geo-Daten (grd, asc, tif)
`rgdal::readGDAL()` … Einlesen von Geo-Daten (grd, asc, tif)

Konstrukt zum Einlesen von Zahlen mit Tausender-Kennzeichen (z.B.: 2.659.300,5)
```
df4spalte <- as.numeric(gsub(".","", df$spaplte)
```

Setzen des Arbeits- / Daten-Verzeichnis-Pfad's
```
setwd("Pfad")
```
set working directory
es gelten die Schreibweisen mit einem Slash "**/**"; Windows-Pfade mit Backslash "\" müssen also angepasst werden!
Angabe des Pfad's als rleativer Pfad:
z.B.:
`UnterOrdner/Daten.TXT` … Daten-Datei in einem Unterordner des Arbeits-Verzeichnises
in R_studio wird die Arbeit mit Projekten empfohlen → alle Projekt-bezogenen Daten sind dann in einem Ordner

`dir()`
Anzeigen der verfügbaren / existierenden Dateien
`dir(recursive=TRUE)` … auch Dateien in weiteren Unterordnern anzeigen

Auslesen des aktuellen Arbeits- / Daten-Verzeichnis-Pfad's
`getwd()`
liefert den Pfad zurück
in R-Studio wird der Pfad auch im oberen Bereich angezeigt

`file.create()` … Erstellen einer Datei

`file.rename()` … Umbenennen einer Datei
`file.remove()` … Löschen einer Datei
`file.copy()` … Kopieren einer Datei
`file.exists()` … Prüfen, ob Datei existiert

`dir.create()` … Erstellen eines Ordner's
`dir.exists()` … Prüfen, ob Ordner existiert

Hinweise:
Daten niemals in den Original-Dateien ändern, sondern Daten erst einmal einlesen und dann anpassen / fixen
so sind Veränderungen nachvollziehbar und ev. auch nachträglich korrigierbar

wenn's unbedingt notwendig ist, dann kann eine Datei in R editiert werden und dann einem neuen Daten-Bestand zugeordnet werden (quasi: "Speichern unter …")
```
neueTabelle <- edit(Tabelle)
```

z.B. mögliche Korrekturen etc. im Skript:
```
neueTabelle[Zeile, Spalte] <- neuerWert   # Kommentar dazu
neueTabelle[Zeile, Spalte] <- NA   # Kommentar dazu
```

bei:
`fix(Tabelle)` wird in den Original-Daten geändert / manipuliert (quasi: "Speichern"); alte (originale) Daten sind dann nicht mehr verfügbar

Beispiel für ein flexibles Einlesen von Datum-Angaben mit verschiedensten Trennern (u.a. auch "Enter" → also Tage und Monat in neuen Zeilen)
gesamte Breite dr Datums-Angaben sei / ist aber immer 10 Zeichen!
```
tage <- read.fwf(file="R33d7_tage.txt", width=10, header=FALSE, )
tage[ ,2] <- substr(tage[ ,1], start=6, stop=7)
tage[ ,3] <- substr(tage[ ,1], start=9 , stop=10)
tage[ ,1] <- substr(tage[ ,1], start=1, stop=4)
```

Links:
Stations-Liste für Wetter-Daten des DWD:
http://opendata.dwd.de/climate_environment/CDC/observations_germany/climate/daily/kl/recent/KL_Tageswerte_Beschreibung_Stationen.txt
Tageswerte "Rostock-Warnemünde"
http://opendata.dwd.de/climate_environment/CDC/observations_germany/climate/daily/kl/recent/tageswerte_KL_04271_akt.zip

9.2.x. Daten (in Dateien) speichern

allgemeingültige Struktur:
```
write.table(x=Daten, file="Dateiname.txt")
```

wichtige Optionen:
`quote` … verhindert (FALSE) die Angabe von Anführungsstrichen (") (bei Text-Daten) in den Zeilen
`row.names` … besagt, ob die erste Spalte Zeilen-Namen sein sollen
`fileEncoding` … zu verwendender Zeichensatz ("UTF-8" ; universellstes Format über die Betriebssysteme hinweg
`quote` …

9.2.x. Tabellen zusammenführen / verbinden

Wichtig: Dimensionen von Tabellen müssen passen!
Bsp.-Daten:

```
daten1 <- data.frame(Zahl=11:13, Gruppe=leters[1:3])
daten2 <- data.frame(Poisson=rpois(3,80)

daten3 <- data.frame(Zahl=21:25, Gruppe=leters[6:10])
```

Verbinden Spalten-orientiert / Tabellen nebeneinander legen
`cbind()`
col bind

```
Daten <- cbind(Daten2, Daten)
Daten <- cbind(Daten2, Daten1)
Daten <- cbind(SpaltenName=Werte, Daten)
Daten <- cbind(SpaltenName=Werte, Daten1)
```

Verbinden Zeilen-orientiert / Tabellen (untereinander) anhängen
`rbind()`
row bind
Spaltennamen müssen übereinstimmen

```
Daten <- rbind(Daten3, Daten)
Daten <- rbind(Daten3, Daten1)


Daten <- rbind(Daten,Zeile)
Daten <- rbind(Daten1, Daten2)
Daten1 <- rbind(Daten1, Daten2)
```

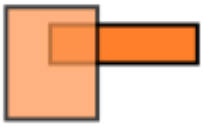
für Matrizen gleiches Verfahren und gleiche Befehle

(Ein-Mischen) von Daten

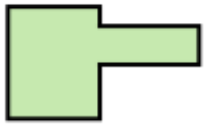
`merge()`

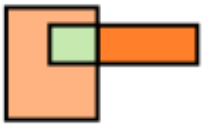
`merge(Daten1, Daten2, by.x="SpaltenName1", by,y="SpaltentenName2")`
`by.x` und `by.y` sind quasi die Primär-Schlüssel (Verknüpfungs-Schlüssel), über die hier verknüpft wird

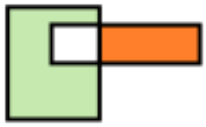
Ergebnis-Tabelle enthält nur Daten, die in beiden Teil-Datenbeständen repräsentiert waren, es also sowohl in der einen als auch in der anderen den gleichen Verknüpfungs- bzw. Primär-Schlüssel gibt
ev. kann man vor dem Mergen die Spaltennamen zueinander passende Spalten umbenennen

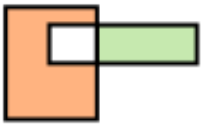

originale Mengen X und Y

Option `all=TRUE` bewirkt, dass auch nicht in beiden Tabellen passende Elemente mit importiert werden, fehlende Werte werden dann auf `NA` gesetzt
weitere Möglichkeiten über `all.x` und `all.y`

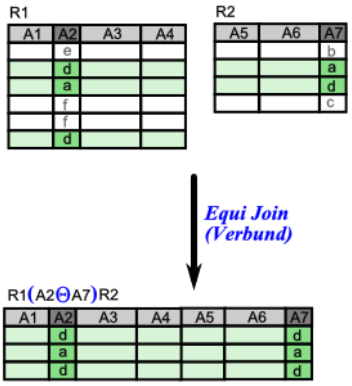

Verknüpfung X **ODER** Y
all=TRUE
full outer join


Verknüpfung X **UND** Y
all=FALSE
natural join


Verknüpfung X **-** Y
all.x=TRUE
left outer join


Verknüpfung Y **-** X
all.y=TRUE
right outer join

für Tabellen (Relationen) sieht die Verknüpfung über all=TRUE dann so aus:
Es werden nur die Zeilen (Datensätze) in die Ergebnis-Tabelle übernommen, für die jeweils ein passender Primärschlüssel in den Ausgangs-Tabellen existiert



werden zwei DatenFrame's kombiniert, die keine gleichen Spaltennamen besitzen, dann wird ein DatenFrame gebildet, der alle Kombinationen aus den originalen Daten-Frame's enthält → praktisch das kartesischen Produkt aus den beiden Original-DatenFrame's

Reduzierendes Kombinieren von Daten

```
reduce()

reduce(function( … ) merge(   , all=TRUE, Daten))
```

Bsp.:

```
DatensatzListe <- list( data.frame(date=1:4, AA=11:14),
                        data.frame(date=2:6, BB=22:26),
                        data.frame(date=3:7, AA=33:37))

reduce(merge, DatensatzListe)
```

weitere Möglichkeit:

```
Daten <- lapply(Daten, function(x) {colnames(x) <- c("date", "XX"); x})
```

weitere Möglichkeit:

```
do.call(rbind, DatensatzListe)
```

Kombination einer Matrix mit einem DatenFrame ergibt ein DatenFrame

9.2.x. Umgang mit fehlenden Daten

missing value's
Kennzeichnung in R mittels `NA`

mit der Funktion `is.na()` wird ein ein Vektor / eine Matrize mit logischen Werten erstellt, bei denen die Fehlwerte mit `TRUE` gekennzeichnet sind
ein direkter Vergleich a'la: x == NA funktioniert nicht!

Entfernen problematischer Zeilen / Datensätze mittels

```
na.omit(Daten)
```

Option na.rm=TRUE in vielen Funktionen möglich
Berechnung der Funktion passiert dann ohne die NA-Werte
z.B.:

```
mean(Daten$Spaltenname, na.rm=TRUE)
```

NA-Imputation
z.B. Auffüllen mit Maßzahlen (z.B. Mittelwert, median, Minimum, Maximum, …)

```
Daten$SpaltenName[is.na(Daten$SpaltenName)]    <-    mean(Daten$SpaltenName,
na.rm=TRUE)

Daten$SpaltenName[is.na(Daten$SpaltenName)]   <-   median(Daten$SpaltenName,
na.rm=TRUE)
```

z.B. Auffüllen mit letzter Beobachtung / Vorgänger-Wert
locf: last obeservation carried forwards
Paket: zoo (set order observation)

```
zoo::na.locf(Daten$SpaltenName)
```

z.B. Auffüllen mit linear interpoliertem Wert

```
approx(Daten$SpaltenName, n=length(Daten$SpaltenName))$SpaltenName
```

od.

```
zoo::na.approx(Daten$SpaltenName)
```

Zählen der NA's Spalten-weise

```
colSums(is.na(Daten))
```

9.2.x. weitere Daten-Strukturen in R

Definition(en): Listen
Listen sind grundlegende Daten-Strukturen in R, die auch verschiedenartige / verschieden typisierte Komponententen / Elemente enthalten können. Typische Elemente sind Zahlen, Zeichenketten, Vektoren, Matrizen und Funktionen.

Definition(en): Matrizen
Matrizen sind in R (eindimensionale) Vektoren mit hinzugefügten weiteren Dimensionen. Matrizen enthalten nur Daten eines Daten-Typ's. Üblicherweise sind dies Zahlen.

Definition(en): Array
Ein Array ist in R eine Daten-Struktur, die Daten in mehreren Dimensionen aufnehmen kann.

ein Array mit den Diemensionen (2,3,4) erstellt 4 Matrizen mit 2 Zeilen und 3 Spalten

9.3. Umgang mit eventuell Fehler-werfenden Code's

```
result1 <- try(log("2"), silent=TRUE)  ;  result1[1] # Die Fehlermeldung
result2 <- try(log(222), silent=TRUE)  ;  result2    # Das Ergebnis
```

9.4. Daten visualisieren

9.4.x. Streu- / Punkt-Diagramme (Scatterplots)

`plot(x, y)`
entspricht XY-Diagrammen (klassischen 2D-Funktions-Diagrammen)
der Standard-Type type="p" braucht nicht benutzt werden
stellen die Beziehungen / den funktionellen Zusammenhang zwischen zwei Variablen dar

```
plot(x=Daten$DatenSpalte1, y=Daten$DatenSpalte2)
```

wird nur ein Daten-Vektor übergeben, dann wird dessen Daten als Y-Werte interpretiert und auf der X-Achse der Index der Werte (Zählung von 1 bis n) genutzt
`plot(Daten$DatenSpalte1)`

zusammenliegende Spalten im DatenFrame oder in der Matrix können – quasi als xy-Paar übergeben werden
`plot(Daten[ , Spalte1:Spalte2])`

Optionen:
`main` … Zeichenkette zur Beschriftung des Diagramm's (Diagramm-Überschrift) (mehrzeilige Überschrift mittels `\n`)
`font.main` … beschreibt Schriftart für Überschrift
`xlab` … Zeichenkette zur Beschriftung der X-Achse (x label)
`ylab` … Zeichenkette zur Beschriftung der Y-Achse (y label)
`las` … Skalen-Beschriftung der Y-Achse wird horizontal (aufrecht) notiert (label axis style)
`xlim` … Daten-Bereich der X-Achse (x limit) in Form von Vektor (c(Minimum, Maximum)) (! wird Max und Min getauscht, dann wird Achse umgedreht)
`ylim` … Daten-Bereich der Y-Achse (y limit) in Form von Vektor (c(Minimum, Maximum)) (! wird Max und Min getauscht, dann wird Achse umgedreht)
`xaxt` … X-Achse ohne Werte `xaxt="n"`
`yaxt` … Y-Achse ohne Werte `yaxt="n"`
`xaxs` … X-Achse ohne linken und rechten Abstand / Rand zur Box Werte `xaxs="i"` (internal) (Standard: `xaxs="r"` (regular (4% Rand)))
`asp` … beschreibt das Verhältnis (aspectratio) der Skalen-Maßstäbe von x zu y (bei `asp=1`) haben beide Achsen den gleichen Maßstab)
`color` … Farbe der Daten-Punkte (color); es folgt Auswahl bzw. Prinzip-Beispiele

Farbcode	Farbe (deutsch)
1	
2	
3	
4	
5	
6	
7	
8	
#000000	RGB-Codierung
#00000000	RGB + Transparenz
grey	
grey1	
grey5	
grey20	

Farbcode	Farbe (deutsch)
white	
beige	
black	
brown	
green	
maroon	
orange	
orchid	
purple	
violet	
yellow	
magenta	
skyblue	
darkblue	

grey40	
grey60	
grey80	
red	
tan	
tan1	
tan3	
tan4	
blue	
cyan	
gold	
navy	
pink	

seagreen	
cadetblue	
darkgreen	
lightblue	
limegreen	
olivedrab	
orangered	
royalblue	
steelblau	
blueviolet	
mediumblue	
forestgreen	
deepskyblue	

`bg` … Füll-Farbe der Daten-Punkte (background)

`pch` … Art / Form der Daten-Punkte (point character)

Code	Zeichen
0	□
1	○
2	△
3	+
4	✕
5	◊
6	▽
7	Quadrat mit Diagonalen
8	✳

Code	Zeichen
9	Raute mit Diagonalen
10	⊕
11	✡
12	Quadrat mit Kreuz
13	Kreis mit Kreuz
14	Quadrat mit Dreieck
15	■
16	●
17	▲

Code	Zeichen	Bemerkung
18	♦	
19	●	
20	•	
21	○	befüllbar
22	□	befüllbar
23	◊	befüllbar
24	△	befüllbar
25	▽	befüllbar

`cex` … Ausdehnung der Daten-Punkte (character expresion)

`berryFunctions::addAlpha("Farbe")` … Farbe der Daten-Punkte wird transparent dargestellt (Transparenz auch durch Hexadezimalzahl (00 .. FF) hinter dem RGB-Code

bei der Farbe kann man auch die Katagorisierung (Faktor (in R)) von Daten nutzen, z.B. die interne Umsetzung von Kategorien in nummerische Werte

```
plot(x=Daten$DatenSpalte1, y=Daten$DatenSpalte2, col= Daten$DatenSpalte3)
```

spezielle Farben / Farb-Kombinationen können als Vektor übergeben werden

```
FarbVektor <- c("Farbe1", "Farbe2", "Farbe3")
plot(x=Daten$DatenSpalte1, y=Daten$DatenSpalte2,
     col= FarbVektor[Daten$DatenSpalte3])
```

soll der gesamte DatenFrame graphisch analysiert (Übersicht), dann bietet sich eine Diagramm-Matrix (pairs-Plot) an

```
plot(Daten, col= Daten$DatenSpalteX, lower.panel=panel.smooth)
```

9.4.x. Linien- / Kurven- / Zeitreihen-Diagramme (Lineplots)

```
plot(type="l", …)
```

Kombination von Linien mit Punkten `type="b"` (both) (mit kleinen Lüchen zwischen den Punkten und den angrenzenden Linien
ohne Lücken mit `type="o"` (overploted), also Punkte direkt auf der Linie
nur die Linien ohne die Daten-Punkte (, aber Lücken an den Punktstellen) mit `type="c"` (central)
es geht auch `type="n"` , dann wird keine Linie gezeichnet

Linien-code	Beschreibung
"l"	Linie
"b"	Linie mit Daten-Punkten (Lücke zu den Linien)
"o"	Linie mit Daten-Punkten (keine Lücke zu den Linien)
"c"	Linie mit Lücke (ohne Daten-Punkte)

Linien-code	Beschreibung
"n"	keine Linie
"s"	stufige Linien (erst waagerecht, dann senkrecht)
"S"	stufige Linien (erst senkrecht, dann waagerecht)
"h"	Säulen (Strich-artig), Histo-gramm-artig

Optionen (ergänzend zu den Optionen von Punkt-Diagrammen):
`lwd` … Strichstärke / Linien-Dicke (line width) in Punkten (Standard ist 1.0)
`lty` … Linien-Art (line type)

Linien-code	Beschreibung
0	"blank"
1	Standard, durchgezogen; "solid"
2	gestrichelt (dashed)
3	gepunktet (dotted)

Linien-code	Beschreibung
4	Strich-Punkt-Linie (dotdash)
5	langgestrichelt (longdash)
6	Strich-Zweipunkt-Linie (twodash)

9.4.x. Balken- / Säulen- / Zeitreihen-Diagramme (barplots)

```
barplot(Daten$DatenSpalte1, names.arg=Daten$DatenSpalte2)
```

der erste Vektor beschreibt die Daten, die als Höhe (im Säulen-Diagramm) genutzt werden sollen
`names.arg` beschreibt die Beschriftung der Säulen (also praktisch die X-Achse)

verkürzte Schreibweise (Formula Interface) mit der Tilde (~) für "abhängig von" (gemeint ist hier die klassische Diagramm-Beschreibung "Abhängigkeit der Größe A von B")

```
barplot(DatenSpalte1 ~ DatenSpalte2, data=Daten)
barplot(Daten$DatenSpalte1 ~ Daten$DatenSpalte2)
```

mehrere abhängige Daten darstellen

```
barplot(cbind(DatenSpalte1, DatenSpalte2) ~ DatenSpalte3, data=Daten)
```

ergibt ineinander gestaffelte Säulen

Balken-Diagramm
`horiz=TRUE`

Farben für die Säulen können über col als Vektor übergeben werden
`col=rainbow(10)` erzeugt einen Farb-Vektor mit 10 Farben von rot bis violett
zusätzlich noch border die Farbe für den Rahlen der Säulen / Balken

weist man die Graphik einer Variable zu, dann enthält diese einen Vektor mit Koordinaten zu den Säulen / Balken
`Variable <- barplot( … )`

Darstellung von Daten aus Matrizen:
`barplot(Matrize)` … ergibt **gestapelt**es Säulen-Diagramm

Optionen (ergänzend zu den Optionen von Punkt- bzw. anderen Diagrammen):
`beside=TRUE` … ergibt **nebeneinander angeordnete Säulen**
`legend=TRUE` … erzeugt eine Legende
`height` … bestimmt Höhe der Säulen

mit `ylim=größererWert` ev. Platz für die Legende schaffen

spezielle Fälle / Beispiele:
Häufigkeits-Diagramm für einn Vektor
`barplot(table(Daten$DatenSpalte) [, …])`

Balkendiagramme können auch abhängig von mehreren Gruppierungen dargestellt werden.
Der eingebaute Datensatz 'sleep' hat für 10 Studenten (Spalte ID) die zusätzliche
Schlafdauer (extra) gemessen, abhängig davon ob sie ein Schlafmittel
genommen hatten oder nicht (group).
```
summary(sleep)
sleep # 20 Zeilen, gerne auskommentieren wenn Ausgabe zu lang.
# Mit einer legendären Legende sieht das so aus:
barplot(extra~group+ID, data=sleep,
        legend=TRUE, args.legend=list(title="Schlafmittel", x="topleft"))
```

9.4.x. Kreis- / Torten-Diagramme (pie charts)

```
pie(DatenVektoren)
```

9.4.x. Objekte zu Diagrammen hinzufügen

"high level"-Plot-Befehle:

- `plot()`
- `barplot()`
- `histoplot()`
- `pie()`
-

erzeugen komplette, **neue** Graphiken
Graphiken werden in R nicht gespeichert (quasi einmaliger Export von visualisierten Daten)

"low level"-Plot-Befehle:

- `points()`
- `lines()`
- `abline()`
- `axis()`
- `legend()`
- `box()`
- `title()`
- `text()`
- `arrows()`
- `segments()`
- `rect()`
- `polygon()`
- `mtext()`
-

akzeptieren viele der üblichen Graphik-Optionen

`points()`
erzeugt einen (zusätzlichen) Punkt in der letzten Graphik

```
points(x=Position, y=Position, cex=Ausdehnung, col=Farbe, lwd=Linienstärke)
```

Verwendung z.B. zum Highlighten eines Punktes, Hinzufügen eines Referenz-Punktes, …

`lines()`
erzeugt eine (zusätzliche) Linie (Polygon) in der letzten Graphik

```
lines(x=XWerte, y=YWerte, col=Farbe, lwd=Linienstärke)
```

XWerte und YWerte können Bereiche (z.B. 4:8) oder Vektoren (z.B. c(3,6,5,4,3)) sein
Verwendung z.B. zum Hinzufügen einer Orientierungs-Linie (bestimmtes Level, Grenzen, …), einer Regressions-Geraden…

`abline()`
erzeugt eine (zusätzliche) Gerade (mit den Parametern a und b entsprechend y = a + bx) in der letzten Graphik

```
abline(a=AWert, b=BWert, col=Farbe)
```

AWert und BWert sind die klassischen Geraden-Parameter (oft auch als **n** und **m** bezeichnet) in der üblichen Geraden-Gleichung y = n + mx (! Achtung Parameter-Reihenfolge beachten!)
Verwendung z.B. zum Hinzufügen einer Regressions-Geraden…

```
abline(h=YLevel, v=XLevel)
```
fügt Linien-Kreuz auf der Höhe YLevel und der X-Verschiebung XLevel ein

Verwendung z.B. zum Hinzufügen einer Orientierungs-Linie (bestimmtes Level, Grenzen, …), Koordinaten-System (h=0, v=0) einer Regressions-Geraden…

```
axis()
```
hinzufügen von Koordinaten-Achsen
ev. kombiniert mit der Option `xaxt="n"` bzw. `yaxt="n"` in einem vorgelagerten plot-Befehl

```
axis(side=Seite, at=Bereich, col=LinienFarbe, c.axis=BeschriftungsFarbe)
```

Seite kann 1 … unten, 2 … links, 3 … oben und 4 … rechts sein

```
legend()
```
fügt eine (zusätzliche) Legende hinzu, z.B. über weitere Graphik-Merkmale zu beschreiben

```
legend("PositionsBeschreibung", legend=BeschriftungsVektor,
       col=MerkmalsFarben, pch=PunktArt)
```

Positions-Beschreibungen: "topright" (oben rechts), "topleft" (oben links), "" () usw. usf.
BeschriftungsVektor kann z.B. eine Datenspalte Daten$DatenSpalte sein
PunktArt ist der klassische DatenPunkt-Code (→ 9.4.x. Streu- / Punkt-Diagramme (Scatterplots))

```
box()
```
fügt eine Umrandung hinzu bzw. formatiert sie neu

```
box(colFarbe, lwd=LinienStärke)
```

```
box("outer", col="red")
```
erzeugt (roten) Rahmen um die gesamte Graphik

```
title()
```
erzeugt Diagramm-Überschrift

```
title(main="DiagrammTitel", adj=1)
```

adj=1 … rechtsbündig, 0 … linksbündig (an Y-Achse orientiert), -0.2 … weiter nach links (in Richtung Diagramm-Rahmen)

weitere Möglichkeiten:
```
title(xlab="Beschriftung", line=Position)
```

`line` … Nummer, wie oben bei axis und mit – kann man auch eine Beschriftung **innen** erreichen

```
text(XPos, YPos, "Text", col=Farbe)
```
hinzufügen eines Textes

XPos und YPos beziehen sich auf die Achsen!
Zeilen-Umbruch mit `\n`

```
arrows()
```
fügt einen Pfeil ins Diagramm ein

```
arrows(x0=XStart, y0=YStart, x1=XEnde, y1=YEnde,
       col=Farbe, lwd=LinienStärke)
```

```
segments()
```
erzeugt Segment- / Bereichs-Abgrenzungen

z.B.:
```
segments(x0=1:3, y0=4:2, x1=5:7, y1= 5:3, col=Farbe)
```

```
rect()
```
erzeugt ein Rechteck im Diagramm

```
rect(xleft=XStart, ybottom=YStart, xright=XEnde, ytop=YEnde,
     col=Farbe, density=FüllCode)
```

```
polygon()
```
erzeugt ein geschlossenes Polygon im Diagramm

```
polygon(x=XPositionsVektor, y=YPositionsVektor, border=Farbe)
```

```
mtext()
```
erzeugt Beschreibungs-Text im Diagramm (außerhalb der Daten-Fläche (Graphik-Fläche)

```
mtext(side=PositionsCode, text="BeschriftungsText",
      line=Stärke, adj=Bündigkeit)
```

Bsp.:
```
mtext(c("margin/side 1", 2:4), side=1:4, las=1)
```

beschriftet die Seiten mit den Ansprech-Nummern / Positionen in vielen Optionen

9.4.x. Zusammenstellen von Graphiken -- Komposition

`par()`
für parameter
setzt Option global für mehrere / folgende Graphiken
gültig immer für ein Ausgabe-Kanal / Device / … (z.B. Bildschirm, PDF, …

```
par(las=1,                  … setzt ab sofort die aufrechte Beschriftung
    bg=Farbe,               … die Hintergrundfarbe
    ann=F                   … Annotationen ausschalten (FALSE)
    mar=c(2,3,1,0.5)        … verbessert die Rand-Einstellungen (→ größe Graphik)
    mgp=(2.1, 0.5, 0)       … (margin placment) Abstände (zwischen Achse und deren
                              Beschriftung, zwischen Achse und deren Zahlen-Werten,
                              zwischen Achse und Diagramm-Flächen-Umrandung)
    mfrow=c(3,2)            … (multiple figures rowwise) mehrere Figuren (Einzel-
                              Diagramme) Zeilen-weise eingezeichnet (Vektor gibt Zeilen
                              und Spalten vor)
    mfcol=c(3,2)            … (multiple figures rowwise) mehrere Figuren (Einzel-
                              Diagramme) Spalten-weise eingezeichnet (Vektor gibt Zei-
                              len und Spalten vor)
    )
```

Graphik-Optionen speichern

```
alteOpt <- par(mar=c(2,2,1,0), las=1, bg="white")
```

gespeicherte Graphik-Optionen wieder anwenden

```
par(alteOpt)
```

(aktuelles) Device schließen / beenden / neu starten

```
dev.off()
```
dannach sind für den nächsten Plot alle Standards wieder eingestellt

alle Devices schließen / beenden / neu starten

```
graphic.off()
```

weitere Möglichkeiten / Optionen:

```
oma=c(0,4,3,0)              … (outer margins)
```

(ev. kombiniert mit Option outer=TRUE in der Plot-Funktion)

Setzen von Rahmen um die einzelnen Figuren / Digramm-Flächen eines Multi-Panel-Diagramm's

```
par( … mfrow(c(2,2), oma=c(0,1,3,0) …)
```

```
box()=box('plot')
box('figure')   nach par(mfrow) oder layout()
box('inner')
box('outer')   wenn äußere Ränder über par(oma) eingestellt wurden
```

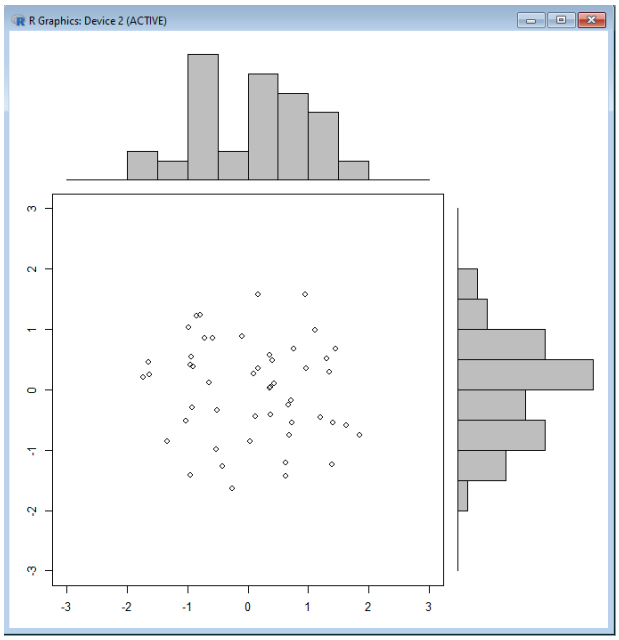
flexible Diagramm-Layout's mittels

layout

1	2
5	
3	4

```
meinLayout <- layout(matrix(1,1,2,2,
                            5,5,2,2,
                            3,3,4,4,
                            3,3,4,4), ncol=4, byrow=TRUE),
                     widths=c(6,6,4,4))
layout.show(meinLayout)
```

für ein Verteilungs-Diagramm mit zwei Histogrammen (für x und y) (marginal histogram) gibt es im Example-Abschnitt von `?layout` den passenden Code (scatterplot with marginal histograms)




```
x <- pmin(3, pmax(-3, stats::rnorm(50)))
y <- pmin(3, pmax(-3, stats::rnorm(50)))
xhist <- hist(x, breaks = seq(-3,3,0.5), plot = FALSE)
yhist <- hist(y, breaks = seq(-3,3,0.5), plot = FALSE)
top <- max(c(xhist$counts, yhist$counts))
```

```
xrange <- c(-3, 3)
yrange <- c(-3, 3)
nf <- layout(matrix(c(2,0,1,3),2,2,byrow = TRUE), c(3,1), c(1,3), TRUE)
layout.show(nf)

par(mar = c(3,3,1,1))
plot(x, y, xlim = xrange, ylim = yrange, xlab = "", ylab = "")
par(mar = c(0,3,1,1))
barplot(xhist$counts, axes = FALSE, ylim = c(0, top), space = 0)
par(mar = c(3,0,1,1))
barplot(yhist$counts, axes = FALSE, xlim = c(0, top), space = 0, horiz =
TRUE)

par(def.par)  #- reset to default
```

9.4.x. Diagramme für Verteilungen

9.4.x.y. Histogramme

zeigen die Anzahl von Elementen je (Werte-)Bereich

`hist()`
Histogramm

```
hist(Daten$DatenSpalte, las=1, main="Überschrift")
histDef <- hist(Daten$DatenSpalte, las=1, main="Überschrift")
```

in der Rückgabe-Variable sind Daten zum Diagramm enthalten:

`$breaks`	: int	[1:n]	Vektor mit den Bereichsgrenzen
`$counts`	: int	[1:n-1]	Vektor mit den gezählten / zugeordneten Elementen
`$density`	: num	[1:n-1]	Dichte
`$mids`	: num	[1:n-1]	Mittlerer Wert der Daten-Elemente im Bereich
`$xname`	: chr		Name des Histogramm's
`$equidist`	: logi		

Optionen:

`breaks` … Anzahl Bereiche / Klassen (wird in R ev. etwas angepasst, um besser geeignete Bereiche zu bekommen)

9.4.x.y. Boxplot's

Vergleichen von Gruppen
Verteilungen anhand ihrer statistischen Kennzahlen darstellen
dargestellt werden:

- Median
- Minimum

- Maximum
- Quantile
- Bereich von 50% der Werte

man sieht u.U. auch:
- Ausreißer (outlayer; nach internen Kriterien berechnet)

```
boxplot()
```
erstellt ein Boxplot

```
boxplot(Daten$DatenSpalte, …)
```

mehrere Spalten und Boxplot's mittel formula-Interface

```
boxplot(DatenSpalteY ~ DatenSpalteX, data=Daten, range=0, las=1)
boxplot(DatenSpalteY ~ DatenSpalteX1+DatenSpalteX2, data=Daten, …)
```

Optionen:
`range=0` … Anzahl Unterdrückung der Berechnung und Darstellung von Ausreißern
`horizontal=TRUE` … horizontale Darstellung des Boxplot's

Hinzufügen der Mittelwerte als Punkte
```
averages <- tapply(DatenSpalteY, DatenSpalteX, mean
points(1:n, averages, pch=PunktForm, …)
```

9.4.x.y. Violinen-Diagramme – Kombination aus Boxplot und Histogramm

```
vioplot(DatenSpalteX, DatenSpalteY, DatenSpalteZ, h=0.5, horizontal=TRUE)
```

9.4.x. Arbeiten mit Device's / Exportieren von Graphiken

Vorteile von PDF:
- Vektor-Graphik
- mehrere Seiten möglich
- gewisse Manipulations-Sicherheit

Exportieren als PDF in eine Datei:

```
pdf("Dateiname.pdf")
plot(…)

pdf("Dateiname")
par(…)  # lokale Parameter-Einstellungen
plot(…)
```

werden mehrere Plot's gemacht, dann entstehen immer neue Seiten in der PDF

Optionen:
`width` … Breite der Grafik (Standard = 7 in = 17,8 cm)
`height` … Höhe der Grafik (Standard = 7 in = 17,8 cm)
!!!Werte nur in inches eingebbar

Schließen des Device, damit wieder z.B. eine Bildschirm-Ausgabe erfolgt

```
dev.off()
```

weitere Device's.

```
jpeg()
png()
postscript()
bmp()
tiff()
xfig()
svg()
```

da JPG mit Anti-Aliasing arbeitet und damit zu Verschmierungen / Vermischungen mit der Umgebung kommt, ist JPG für Graphiken eher nicht zu empfehlen
PNG für klassische Abbildungen ausreichend (→ Raster-Graphik), meist guter Kompromiss zwischen Datei-Größe und Qualität der Abbildung
bei den Graphik-Datei-Formaten wird bei einem erneuten oder mehrfachen Export von Plot's die Vorgänger-Version überschrieben, es wird nur die letzte Grafik gespeichert
mit einem Trick können aber fortlaufende Nummerierungen bei den Dateinamen erzeugt werden:
"Dateiname_%03d.png" … (%3d) erzeugt fortlaufende Nummerierungen: _001, _002, …
%3d … erzeugt Nummerierungen mit Leerzeichen: _ 1, _ 2, …
%d … erzeugt Nummerierunegen (ohne Leerzeichen): _1, _2, … (nicht zu empfehlen, da die Datei-Anzeige u.U. ungünstig sortiert

Optionen für PNG-Exporte (teiweise auch für andere Grafik-Formate gültig):
`width` … Breite der Grafik (Standard = 7 in = 17,8 cm)
`height` … Höhe der Grafik (Standard = 7 in = 17,8 cm)
`units="in"` … Einheit für die Grafik-Höhe / -Breite (sollte man beibehalten, um leicht auf PDF umsteigen zu können (, da PDF nur in kann)); "cm" möglich
`res=300` … Auflösung
`pointsize=14` … Größe eines Grafik-Punktes
`bg="transparent"` … Hintergrund-Farbe

Anzeige einer Datei (PDF) im Standard-Programm:

```
file.show("Dateiname")
```

9.4.x. Tricks und Tips

in RStudio können mehrere Befehle mit geschweiften Klammern (**{ }**) zu einem Block zusammengefasst werden
diese werden dann gemeinsam ausgeführt

mit dem Sumatra-PDF-Viewer kann eine PDF life angesehen und im Hintergrund geändert werden → praktisch für die interaktive Entwicklung von Grafiken
den Sumatra-Viewer gibt es aber nur für Windows-Systeme

Animationen in Graphiken / annimierte Graphiken

```
set.seed(12)
volc <- apply(volcano, 1:2, function(x) x+cumsum(rnorm(100)))
library(animation)
library(pbapply)
saveVideo(pbapply(1:100, function(i) filled.contour(vol[i,, , ],
    zlim=range(volc))), video.name="volc.mp4", interval=0.07,
    ffmpeg="C:/ff_folder/bin/ffmfmpeg.exe
```

Graphiken mit ggplot2

grammar of graphics for plotting 2nd edition

interaktive Karten / Graphiken mit leaflet

```
library(rdwd)
data(geoIndex)
library(leaflet)
leaflet(geoIndex) %>% addTiles()
    %>% addCircles(-lon, -lat, radius=900, stroke=F, color0-col)
    %>% addCircleMarkers(-lon, -lat, popup=-display, stroke=F, color-col)
```

Cheat Sheet's

Basics in R Cheat Sheet (DRAFT) by enigmo

This cheatsheet has been created for a Journocode workshop for data analysis, given in Dusseldorf, Germany in March 2017.

This is a **draft** cheat sheet. It is a work in progress and is not finished yet.

R als Taschenrechner

1 + 2	Addiert die Zahlen
6 * 9	Multipliziert die Zahlen
85 / 17	Dividiert die Zahlen
2 ^ 10	Berechnet die zehnte Potenz von 2
sqrt(225)	Berechnet die Wurzel aus 225

Variablen in R

x <- 3	Weist der Variable **x** die Zahl 3 zu
x = 3	Alternative Zuweisung von Variablen
x + y	Addiert die Variablen **x** und **y**
y / x	Dividiert die Variable **y** durch **x**
y * x	Multipliziert die Variablen **x** und **y**

Mit Vektoren rechnen in R

c(2, 7, 5, 9)	Erstellt einen Vektor und wirft den Inhalt in der Konsole aus
v <- c(2, 7, 5, 9)	Weist der Variable **v** den Vektor zu
v	Wirft den Inhalt der Variable **v** in der Konsole aus
v / 2	Teilt alle Werte in Variable **v** durch 2
v - w	Subtrahiert die Werte in **w*** von den Werten in **v**
v * w	Multipliziert die Werte in **v** mit den Werten in **w**

Dezimalzahlen und Strings

4.7	Dezimalzeichen: "**.**"
"qwertz"	Erstellen von Strings mit Anführungszeichen
q <- "Goodbye"	Zuweisen von Strings
paste(q, w)	Verketten von Vektoren
paste0(q, w)	Verketten von Vektoren ohne Lücken

Datenanalyse mit dplyr

install.packages()	Pakete installieren - Paketname muss in Anführungszeichen
library()	Lädt ein installiertes Paket
::	Nimmt Funktion aus spezifiziertem Paket, z.B. **dplyr::summarise**
>%>	magrittr Pipe-Operator, verbindet Funktionen
group_by()	dplyr-Funktion, gruppiert Datensatz nach einer Spalte
filter()	dplyr-Funktion, filtert Datensatz nach einer Spalte
summarise()	dplyr-Funktion, fasst Ergebnisse zB durch Summierung zusammen
mutate()	dplyr-Funktion, fügt eine neue Spalte hinzu

arrange()	dplyr-Funktion, sortiert Ergebnisse zB der Größe nach
split()	dplyr-Funktion, schneidet Ergebnis an einer bestimmten Stelle ab

Datenstrukturen in R

seq(11, 91, 10)	Erzeugt eine Zahlenfolge nach bestimmten Kriterien
x <- c(seq(11 ,91, 10))	Weist **x** einen Vektor mit der Sequenz zu
matrix(c(x, z), 9, 2)	Erzeugt aus **x** und **z** eine Matrix mit 9 Zeilen und 2 Spalten
data.frame(x, y, z)	Erzeugt aus **x**, **y** und **z** einen Dataframe

Logische Operatoren und Indizes

%in%	Bsp.: 3 %in% c(1,2,3) ist TRUE
$	Zugriff über Spaltennamen, Bsp.: data$namen
data[i,j]	Indizierung - **i**-te Zeile, **j**-te Spalte
|	"oder"
&	"und"
== und **&&**	"ist gleich" - logischer Operator, ob links gleich rechts ist

Datenimport und Datenexport

getwd()	Zeigt das aktuelle working directory
setwd()	Setzt das working directory
read.csv()	Importiert eine CSV-Datei
?read.csv	Öffnet die Hilfeseite von **read.csv** in RStudio
View()	Zeigt den Dataframe als Tabelle an
head()	Zeigt die ersten x Einträge eines Dataframes (unsortiert)
tail()	Zeigt die letzten x Einträge eines Dataframes (unsortiert)
nrow()	Zeigt die Anzahl der Zeilen eines Dataframes an
ncol()	Zeigt die Anzahl der Spalten eines Dataframes an
length()	Zeigt die Anzahl Einträge an, z.B. eines Vektors
class()	Zeigt die Klasse eines Objekts an, z.B. "data.frame"
str()	Zeigt Struktur eines Objekts an, z.B. "data.frame" + die Struktur der Spalten, z.B. "numeric"
is.na()	Logical Operator für fehlende Werte in einem Datenobjekt
any()	Erfüllt irgendein Wert im Objekt eine bestimmte Bedingung? Bsp.: **any(is.na())**
unique()	Filtert alle Duplikate heraus
duplicated()	Logical Operator für "Bist du ein Duplikat?"
na.omit()	Zeigt ein Objekt ohne fehlende Werte an
sum()	Berechnet die Summe von Werten
max()	Zeigt den größten Werte in einem Objekt an
min()	Zeigt den kleinsten Wert in einem Objekt an
mean()	Zeigt den Mittelwert eines Objekts an
table()	Zeigt die Häufigkeitstabelle eines Objekts an
sort()	Sortiert die Einträge eines Objekts
as.character()	Setzt die Klasse eines Objekts auf "character"
as.factor()	Setzt die Klasse eines Objekts auf "factor"
as.numeric()	Setzt die Klasse eines Objekts auf "numeric"

Q: https://cheatography.com/enigmo/cheat-sheets/basics-in-r/

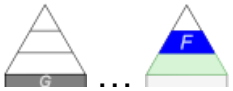
es gíbt diverse weitere Cheat Sheets (meist im praktischem PDF-Format zum Ausdrucken) diverse "offizielle" des RStudio's unter → https://www.rstudio.com/resources/cheatsheets/

10. Datenbanken und Datenschutz

11. Suchmaschinen

https://open.hpi.de/courses/searchengine2017 (abgelaufener OpenHPI-Kurs zum Selbststudium)

12. komplexe und Übungs-Aufgaben



Aufgaben:

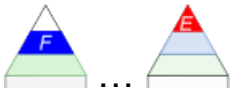
1.

2.

3. Bestimmen Sie die Kardinalitäten der nachfolgenden Beziehungen! Erläutern Sie Ihre Entscheidung!

a)	Schüler	-----	hat	-----	Lehrer
b)	Auto	-----	hat	-----	Motor
c)	Auto	-----	hat	-----	Reifen
d)	Schüler	-----	erhält	-----	Zeugnis
e)	Student	-----	darf_arbeiten_an	-----	PC
f)	Student	-----	besucht	-----	Kurs
g)	Schüler	-----	ist_befreundet_mit	-----	Schüler
h)	Leser	-----	leiht_aus	-----	Buch
i)	Schüler	-----	hat	-----	Zeugnis

4.



Aufgaben für die gehobene Anspruchsebene:

1.

2. Testen Sie Ihr Datenbank-Wissen! Was können Sie davon schon?

http://gisbsc.gis-ma.org/GISBScL4/de/html/GISBSc_VL4_VL_WC.html

3.

Literatur und Quellen:

/1/
ISBN

/2/
ISBN

/3/
ISBN

/4/
ISBN

/5/ GIERHARDT, Horst:
Datenbanken: Entity-Relationship-Model.-Bad Laasphe, Städtisches Gymnasium, 2014.-horst@gierhardt.de
(http://www.oberstufeninformatik.de/Datenbanken/ERMTheorie.pdf)

/10/ SELLE, Stefan:
Künstliche Neuronale Netzwerke und Deep Learning.-Saarbrücken (2018)

Die originalen sowie detailliertere bibliographische Angaben zu den meisten Literaturquellen sind im Internet unter http://dnb.ddb.de zu finden.

Kurz-Referenz auf Quelle

/###/ Syntax-Diagramme für SQLite
https://www.sqlite.org/syntaxdiagrams.html

Internet-Seiten, etc.

/A/ Wikipedia
http://de.wikipedia.org

Abbildungen und Skizzen entstammen den folgende ClipArt-Sammlungen:

/A/ 29.000 Mega ClipArts; NBG EDV Handels- und Verlags AG; 1997

/B/

andere Quellen sind direkt angegeben.

Alle anderen Abbildungen sind geistiges Eigentum:

/I/ lern-soft-projekt: drews (c,p) 1997 - 2025 lsp: dre

verwendete freie Software:

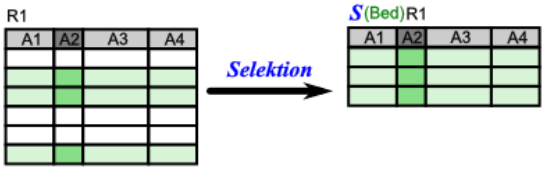
- **Inkscape** von: inkscape.org (www.inkscape.org)
- **CmapTools** von: Institute for Human and Maschine Cognition (www.ihmc.us)

⊞- (c,p) 2015 - 2025 lern-soft-projekt: drews -⊟
⊞- drews@lern-soft-projekt.de -⊟
⊞- http://www.lern-soft-projekt.de -⊟
⊞- 18069 Rostock; Luise-Otto-Peters-Ring 25 -⊟
⊞- Tel/AB (0381) 760 12 18 FAX 760 12 11 -⊟

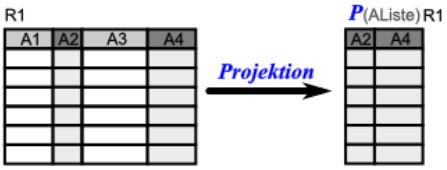
derzeit Aussortiertes

Methoden für eine Tabelle (oder Abfrage)

- **Selektion** — Auswahl von Zeilen (Tupeln) aus einer Relation aufgrund einer oder mehrerer Bedingungen

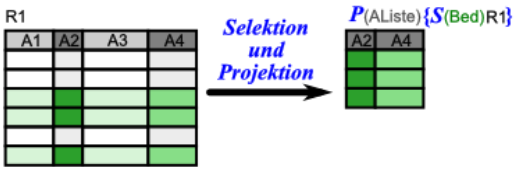


- **Projektion** — Auswahl (/ Einschränkung) von Spalten (Attribute) aus einer Relation aufgrund einer Auswahlliste



möglich auch Kombinationen, praktisch sinnvoll um die anzuzeigende / bereitgestellte Daten-Menge noch weiter zu reduzieren:

- **Selektion und Projektion** — Auswahl von Zeilen (Tupeln) und auch (Einschränkung) von Spalten (Attributen) aus einer Relation



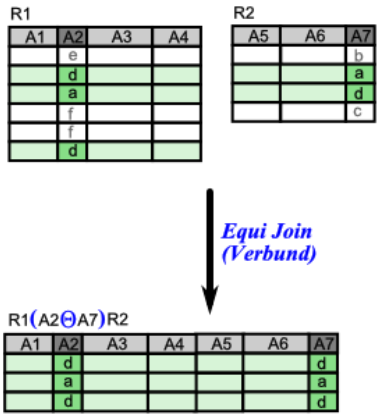
-

-

Methoden für mehrere Tabellen und / oder Abfragen

- **Verbund**
  Join
  (Equi Join)

  Verknüpfung von Relationen (Tabellen) aufgrund von Attributs-Beziehungen



- **Mengen-Operation**

  Bildung von Vereinigung, Differenz und / oder Durchschnitt auf Relationen mit gleicher Struktur