Data Models & Processing and Statistics

Lecture BigData Analytics

Julian M. Kunkel

julian.kunkel@googlemail.com

University of Hamburg / German Climate Computing Center (DKRZ)

16-10-2015

Outline

Basic Considerations About Storing Big Data

- New data is constantly coming (Velocity of Big Data)
  - How can we update our derived data (and conclusions)?
    - Incremental updates vs. (partly) re-computation algorithms
  - How can we ingest the data?
- Storage and data management techniques are needed
  - How can we diagnose causes for problems with data (e.g. inaccuracies)?
- Efficient processing of data is key for analysis

Management of data

- Idea: Store facts (truth) and never change/delete them
  - Data value may degrade over time, garbage clean old data
- Raw data is usually considered to be **immutable**
  - Implies that an update of (raw) data is not necessary
- Create a model for representing the data

Terminology

Data [1, 10]

- **Raw data**: collected information that is not derived from other data
- **Derived data**: data produced with some computation/functions
- **View**: presents derived data to answer specific questions
  - Convenient for users (only see what you need) + faster than re-computation
  - Convenient for administration (e.g. manage permissions)
  - Data access can be optimized

Dealing with unstructured data

- We need to extract information from raw unstructured data
  - e.g. perform text-processing using techniques from computer linguistics
- **Semantic normalization** is the process of reshaping free-form information into a structured form of data [11]
- Store raw data when your processing algorithm improves over time

Terminology for Managing Data [1, 10]

- **Data life cycle**: creation, distribution, use, maintenance & disposition
- **Information lifecycle management** (ILM): business term; practices, tools and policies to manage the data life cycle in a cost-effective way
- **Data governance**: "control that ensures that the data entry ... meets precise standards such as business rule, a data definition and data integrity constraints in the data model" [10]
- **Data provenance**: the documentation of input, transformations of data and involved systems to support analysis, tracing and reproducibility
- **Data-lineage** (Datenherkunft): forensics; allows to identify the source data used to generate data products (part of data provenance)
- **Service level agreements** (SLAs): contract defining quality, e.g. performance/reliability & responsibilities between service user/provider

Data-Cleaning and Ingestion

- Importing of raw data into a big data system is an important process
  - Wrong data results in wrong conclusions: Garbage in – Garbage out
- **Data wrangling**: process and procedures to clean and convert data from one format to another [1]
  - **Data extraction**: identify relevant data sets and extract raw data
  - **Data munging**: cleaning raw data, converting it to a format for consumption
- Extract, Transform, Load (**ETL process**): data warehouse term for importing data (from databases) into a data warehouse

Necessary steps

- Define and document data governance policies to ensure data quality
  - Identifying and dealing with duplicates, time(stamp) synchronization
  - Handling of missing values (NULL or replace them with default values)
- Document the conducted transformations (for data provenance)
  - Data sources
  - Conversions of data types, complex transformations
  - Extraction of information from unstructured data (semantic normalization)
- Implementation of the procedures for bulk loading and cleaning of data

Datawarehousing ETL Process

- Extract: read data from source databases
- Transform
  - Perform quality control
  - Improve quality: treat errors and uncertainty
  - Change the layout to fit the data warehouse
- Load: integrate the data into the data warehouse
  - Restructure data to fit needs of business users
  - Rely on batch integration of large quantities of data

Data Models[1] and their Instances [12]

- A data model describes how information is organized in a system
  - It is a tool to specify, access and process information
  - A model provide operations for accessing and manipulating data that follow certain semantics
  - Typical information is some kind of entity (virtual object) e.g. car, article
- Logical model: abstraction expressing objects and operations
- Physical model: maps logical structures onto hardware resources (e.g. Files, bytes)

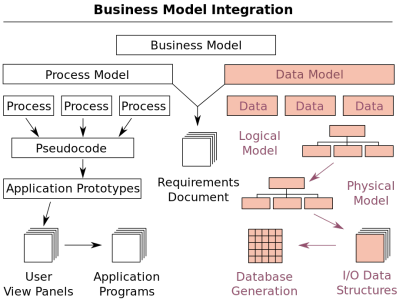

Figure: Source: [12]

- DM theory: Formal methods for describing data models with tool support
- Applying theory creates a **data model instance** for a specific application

[1] The term is often used ambivalently for a data (meta) model concept/theory or an instance

Process Models [13]

- Models describing processes
  - Process: “A series of events to produce a result, especially as contrasted to product.” [15]
- Qualities of descriptions
  - Descriptive: Describe the events that occur during the process
  - Prescriptive
    - Define the intended process and how it is executed
    - Rules and guideliness steering the process
  - Explanatory
    - Provide rationales for the process
    - Describe requirements
    - Establish links between processes

Programming Paradigms [14]

Programming paradigms are process models for computation

- Fundamental style and abstraction level for computer programming
  - **Imperative** (e.g. Procedural)
  - **Declarative** (e.g. Functional, **Dataflow**, Logic)
  - **Data-driven** programming (describe patterns and transformations)
  - **Multi-paradigm** supporting several (e.g. **SQL**, Scala)
- There are many paradigms with tools support available
- Parallelism is an important aspect for processing of large data
  - In HPC, there are language extensions, libraries to specify parallelism
    - PGAS, Message Passing, OpenMP, data flow e.g. OmpSs, ...
  - In BigData Analytics, libraries and domain-specific languages
    - MapReduce, SQL, data-flow, streaming and data-driven

Domain-specific Language (DSL)

- Specialized programming language to an application domain
  - Mathematics e.g. statistics, modelling
  - Description of graphs e.g. graphviz (dot)
  - Processing of big data
- A contrast to general-purpose languages (GPL)
- Standalone vs. embedded
  - Embedding into a GPL (e.g. regex, SQL) with library support
  - Standalone requires to provide its own toolchain (e.g. compiler)
  - Source-to-source compilation (DSL to GPL) an alternative
- High-level of abstraction or low-level
  - Low-level: includes technical details (e.g. about hardware)

Selection of Theory (concepts) for Data Models

- I/O Middelware: NetCDF, HDF5, ADIOS
- Relational model (tuples and tables)
  - e.g. can be physically stored in a CSV file or database
- Relational model + raster data
  - Operations for N-dimensional data (e.g. pictures, scientific data)
- NoSQL data models: Not only SQL[2], lacks features of databases
  - Column
  - Document
  - Key-value
  - Named graph
- Fact-based: built on top of atomic facts, well-suited for BI [11]

Data modeling [10]

The process in software-engineering of creating a data model instance for an information system

[2] Sometimes people also call it No SQL

Semantics

- Describes operations and their behavior
  - Application programming interface (API)
  - Concurrency: Behavior of simultaneously executed operations
    - Atomicity: Are partial modifications visible to other clients
    - Visibility: When are changes visible to other clients
    - Isolation: Are operations influencing other ongoing operations
  - Availability: Readiness to serve operations
    - Robustness of the system for typical (hardware and software) errors
    - (Scalability: availability and performance behavior with number of requests)
    - Partition tolerance: Continue to operate even if network breaks partially
  - Durability: Modifications should be stored on persistent storage
    - Consistency: Any operation leaves a consistent system

CAP-Theorem

It is not possible to fulfill all three attributes in a distributed system:

- Consistency (here: immediate visibility of changes among all clients)
- Availability (we'll receive a response for every request)
- Partition tolerance (system operates despite network failures)

Example Semantics

POSIX I/O

- Atomicity and isolation for individual operations, locking possible

ACID

- Atomicity, consistency, isolation and durability for transactions
- Strict semantics for database systems to prevent data loss

BASE

- BASE is a typical semantics for Big Data due to the CAP theorem
- Basically Available replicated Soft state with Eventual consistency [26]
  - Availability: Always serve but may return a failure, retry may be needed
  - Soft state: State of the system may change over time without requests due to eventual consistency
  - Consistency: If no updates are made any more, the last state usually becomes visible to all clients
- Big data solutions often exploit the immutability of data

Columnar Model

- Data is stored in rows and “columns” (evtl. tables)
- A column is a tuple (name, value and timestamp)
- Each row can contain other columns
  - Columns can store complex objects e.g. collections
- Examples: HBase, Cassandra, Accumulo

Row/Column:	student name	matrikel	lectures	lecture name
1	"Max Mustermann"	4711	[3]	-
2	"Nina Musterfrau"	4712	[3,4]	-
3	-	-	-	"Big Data Analytics"
4	-	-	-	"Hochleistungsrechnen"

Table: Example columnar model for the students, each value has its own timestamp (not shown). Note that lectures and students should be modeled with two tables

Key-Value Store

- Data is stored as value and addressed by a key
- The value can be complex objects e.g. JSON or collections
- Keys can be forged to simplify lookup evtl. tables with names
- Examples: CouchDB, BerkeleyDB, Memcached, BigTable

Key	Value
stud/4711	<name>Max Mustermann</name><attended><id>1</id></attended>
stud/4712	<name>Nina Musterfrau</name><attended><id>1</id><id>2</id></attended>
lec/1	<name>Big Data Analytics</name>
lec/2	<name>Hochleistungsrechnen</name>

Table: Example key-value model for the students with embedded XML

Document Model

- Documents contain semi-structured data (JSON, XML)
- Each document can contain data with other structures
- Addressing to lookup documents are implementation specific
  - e.g. bucket/document key, (sub) collections, hierarchical namespace
- References between documents are possible
- Examples: MongoDB, Couchbase, DocumentDB

```
<students>
  <student><name>Max Mustermann</name><matrikel>4711</matrikel>
    <lecturesAttended><id>1</id></lecturesAttended>
  </student>
  <student><name>Nina Musterfrau</name><matrikel>4712</matrikel>
    <lecturesAttended><id>1</id><id>2</id></lecturesAttended>
  </student>
</students>
```

Table: Example XML document storing students. Using a bucket/key namespace, the document could be addressed with key: “uni/stud” in the bucket “app1”

Graph

- Entities are stored as nodes and relations as edges in the graph
- Properties/Attributes provide additional information as key/value
- Examples: Neo4J, InfiniteGraph

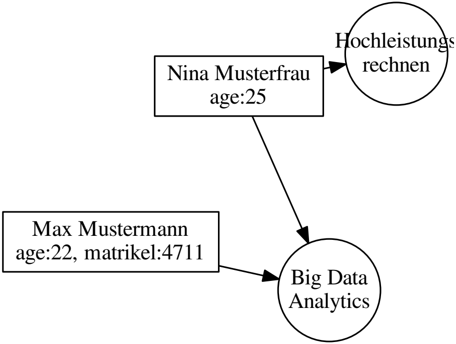

Figure: Graph representing the students (attributes are not shown)

Relational Model [10]

- Database model based on first-order predicate logic
  - Theoretic foundations: relational algebra and relational calculus
- Data is represented as tuples
- Relation/Table: groups tuples with similar semantics
  - Table consists of rows and named columns (attributes)
  - No duplicates of complete rows allowed
- In its raw style no support for collections in tuples
- Schema: specify structure of tables
  - Datatypes (domain of attributes)
  - Consistency via constraints
  - Organization and optimizations

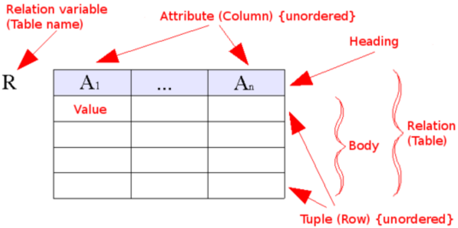

Figure: Source: Relational model concepts [11]

Example Relational Model for Students Data

Matrikel	Name	Birthday
242	Hans	22.04.1955
245	Fritz	24.05.1995

Table: Student table

ID	Name
1	Big Data Analytics
2	Hochleistungsrechnen

Table: Lecture table

Matrikel	LectureID
242	1
242	2
245	2

Table: Attends table representing a relation

Fact-Based Model [11][4]

- Store raw data as timestamped atomic facts
- Never delete true facts: Immutable data
- Make individual facts unique to prevent duplicates

Example: social web page

- Record all changes to user profiles as facts
- Benefits
  - Allows reconstruction of the profile state at any time
  - Can be queried at any time[3]

Example: purchases

- Record each item purchase as facts together with location, time, ...

[3] If the profile is changed recently, the query may return an old state.

[4] Note that the definitions in the data warehousing (OLAP) and big data [11] domains are slightly different

Wishlist for Big Data Technology [11]

- High-availability, fault-tolerance
- (Linear) Scalability
  - i.e. 2n servers handle 2n the data volume + same processing time
- Real-time data processing capabilities (interactive)
  - Up-to-date data
- Extensible, i.e. easy to introduce new features and data
- Simple programming models
- Debuggability
- (Cheap & ready for the cloud)

  ⇒ Technology works with TCP/IP

Components for Big Data Analytics

Required components for a big data system

- Servers, storage, processing capabilities
- User interfaces

Storage

- NoSQL databases are non-relational, distributed and scale-out
  - Hadoop Distributed File System (HDFS)
  - Cassandra, CouchDB, BigTable, MongoDB [5]
- Data Warehouses are useful for well known and repeated analysis

Processing capabilities

- Interactive processing is difficult
- Available technology offers
  - Batch processing
  - “Real-time” processing (seconds to minutes turnaround)

[5] See http://nosql-database.org/ for a big list

Alternative Processing Technology

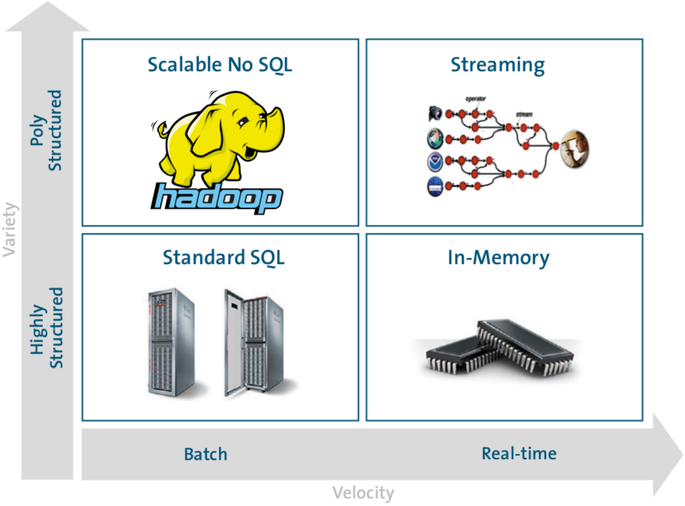

Figure: Source: Forrester Webinar. Big Data: Gold Rush Or Illusion? [4]

The Hadoop Ecosystem (of the Hortonworks Distribution)

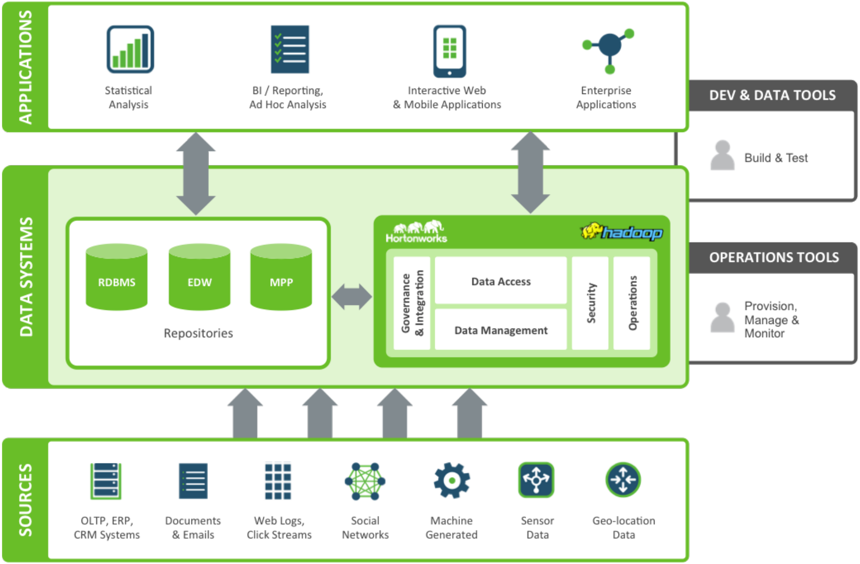

Figure: Source: [20]

The Lambda Architecture [11]

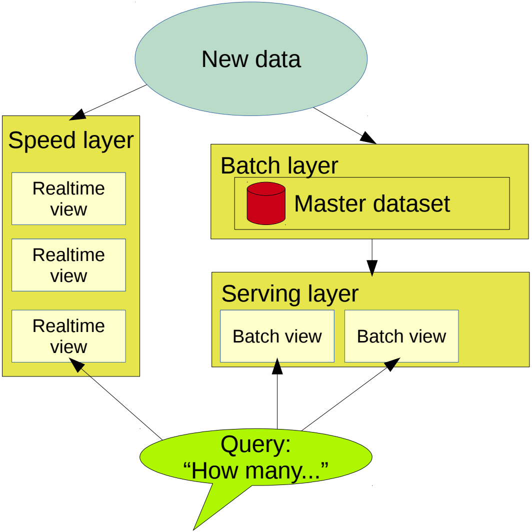

Figure: Redrawn figure. Source: [11], Fig. 2.1

- Goal: Interactive Processing
- Batch layer pre-processes data
  - Master dataset is immutable/never changed
  - Operations are periodically performed
- Serving layer offers performance optimized views
- Speed layer serves deltas between batch and recent activities
- Robust: Errors/inaccuracies of realtime views are corrected in batch view

Statistics: Overview

Statistics is the study of the collection, analysis, interpretation, presentation, and organization of data [21]

Either **describe** properties of a sample or **infer** properties of a population

Important terms [10]

- **Unit of observation**: the entity described by the data
- **Unit of analysis**: the major entity that is being analyzed
  - Example: observe income of each person, analyse differences of countries
- **Statistical population**: complete set of items that share at least one property that is subject of analysis
  - Subpopulation share additional properties
- **Sample**: (sub)set of data collected and/or selected from a population
  - If chosen properly, they can represent the population
  - There are many sampling methods, we can never capture ALL items
- **Independence**: one observation does not effect another
  - Example: Select two people living in Germany randomly
  - Dependent: select one household and pick married couple

Statistics: Variables

- **Dependent variable**: represents the output/effect
  - Example: Word count of a Wikipedia article; income of people
- **Independent variable**: assumed input/cause/explanation
  - Example: Number of sentences; age, educational level
- **Univariate analysis** looks at a single variable
- **Bivariate analysis** describes/analyze relationships between two variables
- **Multivariate statistics**: analyze/observe multiple dependent variables
  - Example: chemicals in the blood stream of people, chance for cancers
    Independent variables are personal information / habits

Descriptive Statistics [10]

- The discipline of quantitatively describing main features of sampled data
  - Summarize observations/selected samples
- **Exploratory data analysis** (EDA): approach for inspecting data
  - Using different chart types, e.g. Box plots, histograms, scatter plot
- Methods for Univariate analysis
  - Distribution of values, e.g. mean, variance, quantiles
  - Probability distribution and density
  - t-test (e.g. check if data is t-distributed)
- Methods for Bivariate analysis
  - Correlation coefficient[6] describes linear relationship
  - Rank correlation[7]: extent by which one variable increases with another var
- Methods for Multivariate analysis
  - Principal component analysis (PCA) converts correlated variables into linearly uncorrelated variables called principal components

[6] Pearson's product-moment coefficient
[7] By Spearman or Kendall

Example Dataset: Iris Flower Data Set

- Contains information about iris flower
- Three species: Iris Setosa, Iris Virginica, Iris Versicolor
- Data: Sepal.length, Sepal.width, Petal.length, Petal.width

R example

```
> data(iris) # load iris data
> summary(iris)
Sepal.Length    Sepal.Width     Petal.Length
Min.   :4.300   Min.   :2.000   Min.   :1.000
1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600
Median :5.800   Median :3.000   Median :4.350
Mean   :5.843   Mean   :3.057   Mean   :3.758
3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100
Max.   :7.900   Max.   :4.400   Max.   :6.900

Petal.Width          Species
Min.   :0.100   setosa    :50
1st Qu.:0.300   versicolor:50
Median :1.300   virginica :50
Mean   :1.199
3rd Qu.:1.800
Max.   :2.500

# Draw a matrix of all variables
> plot(iris[,1:4], col=iris$Species)
```

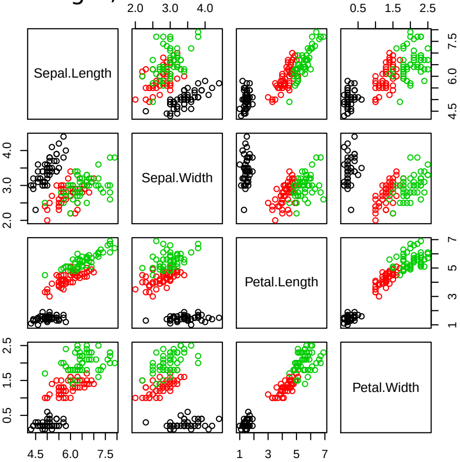

Figure: R plot of the iris data

Distribution of Values: Histograms [10]

- Distribution: frequency of outcomes (values) in a sample
  - Example: Species in the Iris data set
    - setosa: 50
    - versicolor: 50
    - virginica: 50
- Histogram: graphical representation of the distribution
  - Partition observed values into bins
  - Count number of occurrences in each bin
  - It is an estimate for the probability distribution

R example

```
# nclass specifies the number of bins
# by default, hist uses equidistant bins
hist(iris$Petal.Length, nclass=10, main="")

hist(iris$Petal.Length, nclass=25, main="")
```

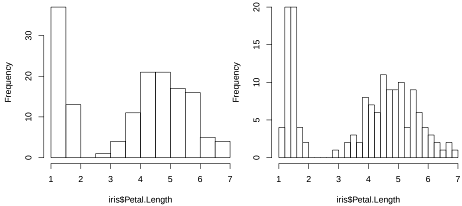

Figure: Histograms with 10 and 25 bins

Distribution of Values: Density [10]

- Probability density function (density):
  - Likelihood for a **continuous** variable to take on a given value
  - Kernel density estimation (KDE) approximates the density

R example

```
# The kernel density estimator moves a function (kernel) in a window across samples
# With bw="SJ" or nrd it automatically determines the bandwidth i.e. window size
d = density(iris$Petal.Length, bw="SJ", kernel="gaussian")
plot(d, main="")
```

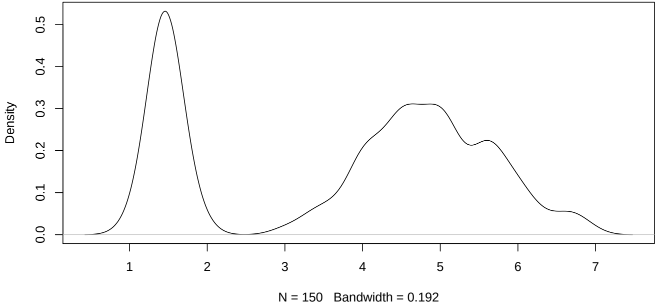

Figure: Density estimation of Petal.Length

Distribution of Values: Quantiles [10]

- Percentile: value below which a given percentage of observations fall
- q-Quantiles: values that partition a ranked set into q equal sized subsets
- Quartiles: three data points that split a ranked set into four equal points
  - Q1=P(25), Q2=median=P(50), Q3=P(75), interquartile range iqr=Q3-Q1
- Boxplot: shows quartiles (Q1,Q2,Q3) and whiskers
  - Whiskers extend to values up to 1.5 iqr from Q1 and Q3
  - Outliers are outside of whiskers

R example

```
> boxplot(iris, range=1.5) # 1.5 interquartile range
> d = iris$Sepal.Width
> quantile(d)
  0%  25%  50%  75% 100%
 2.0  2.8  3.0  3.3  4.4
> q3 = quantile(d,0.75) # pick value below which are 75%
> q1 = quantile(d,0.25)
> irq = (q3 - q1)
# identify all outliers based on the interquartile range
> mask = d < (q1 - 1.5*irq) | d > (q3 + 1.5*irq)
# pick outlier selection from full data set
> o = iris[mask,]
# draw the species name into the boxplot
> text(rep(1.5,nrow(o)), o$Sepal.Width, o$Species,
      ↪ col=as.numeric(o$Species))
```

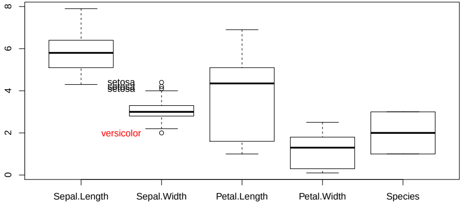

Figure: Boxplot

Density Plot Including Summary

```r
d = density(iris$Petal.Length, bw="SJ", kernel="gaussian")

# add space for two axes
par(mar=c(5, 4, 4, 6) + 0.1)
plot(d, main="")
# draw lines for Q1, Q2, Q3
q = quantile(iris$Petal.Length)
q = c(q, mean(iris$Petal.Length))
abline(v=q[1], lty=2, col="green", lwd=2)
abline(v=q[2], lty=3, col="blue", lwd=2)
abline(v=q[3], lty=3, col="red", lwd=3)
abline(v=q[4], lty=3, col="blue", lwd=2)
abline(v=q[5], lty=2, col="green", lwd=2)
abline(v=q[6], lty=4, col="black", lwd=2)
# Add titles
text(q, rep(-0.01, 5), c("min", "Q1", "median", "Q3", "max", "mean"))

# identify x limits
xlim = par("usr")[1:2]
par(new=TRUE)

# Empirical cummulative distribution function
e = ecdf(iris$Petal.Length)
plot(e, col="blue", axes=FALSE, xlim=xlim, ylab="", xlab="", main="")

axis(4, ylim=c(0,1.0), col="blue")
mtext("Cummulative distribution function", side=4, line=2.5)
```

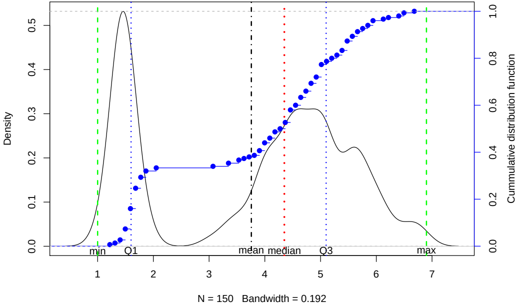

Figure: Density estimation with 5-number summary and cumulative density function

Correlation Coefficients

- Measures (linear) correlation between two variables
  - Value between -1 and +1
  - >0.7: strong positive correlation
  - >0.2: weak positive correlation
  - 0: no correlation, < 0: negative correlation

R example

```
library(corrplot)
d = iris
d$Species = as.numeric(d$Species)
corrplot(cor(d), method = "circle") # linear correlation

mplot = function(x,y, name){
  pdf(name,width=5,height=5) # plot into a PDF
  p = cor(x,y, method="pearson") # compute correlation
  k = cor(x,y, method="spearman")
  plot(x,y, xlab=sprintf("x\n cor. coeff: %.2f rank coef.: %.2f", p, k))
  dev.off()
}

mplot(iris$Petal.Length, iris$Petal.Width, "iris-corr.pdf")
# cor. coeff: 0.96 rank coef.: 0.94

x = 1:10; y = c(1,3,2,5,4,7,6,9,8,10)
mplot(x,y, "linear.pdf") # cor. coeff: 0.95 rank coef.: 0.95

mplot(x, x*x*x , "x3.pdf") # cor. coeff: 0.93 rank coef.: 1
```

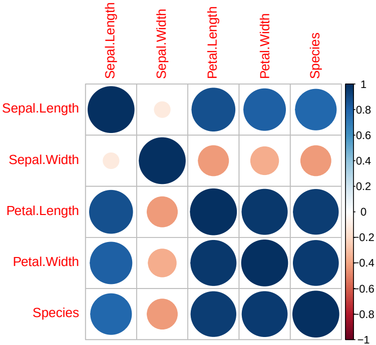

Figure: Corrplot

Example Correlations for X, Y Plots

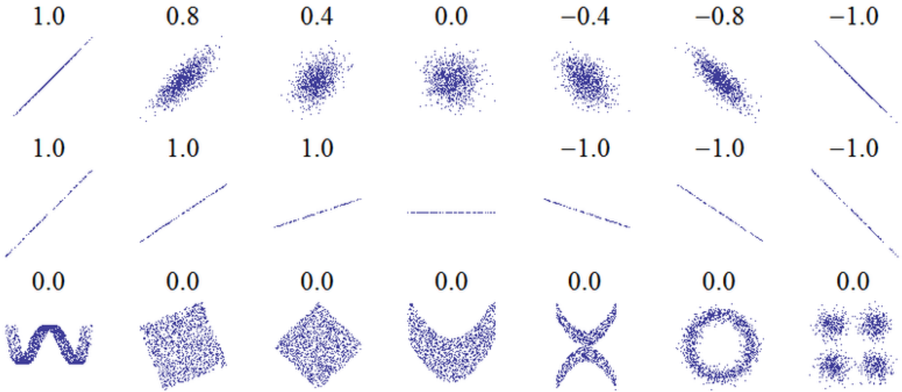

Figure: Correlations for x,y plots; Source: [22]

Inductive Statistics: Some Terminology [10]

- Statistical inference is the process of **deducting properties** of a population by analyzing samples
  - Build a statistical model and test the hypothesis if it applies
  - Allows to deduct propositions (statements about data properties)
- **Statistical hypothesis**: hypothesis that is testable on a process modeled via a set of random variables
- **Statistical model**: embodies a set of assumptions concerning the generation of the observed data, and similar data from a larger population. A model represents, often in considerably idealized form, the data-generating process
- **Validation**: Process to verify that a model/hypothesis is likely to represent the observation/population
- **Significance**: A significant finding is one that is determined (statistically) to be very unlikely to happen by chance
- **Residual**: difference of observation and estimated/predicted value

Statistics: Inductive Statistics [10]

Testing process

1. Formulate default (null[8]) and alternative hypothesis
2. Formulate statistical assumptions e.g. independence of variables
3. Decide which statistical tests can be applied to disprove null hypothesis
4. Choose significance level α for wrongly rejecting null hypothesis
5. Compute test results, especially the **p-value**[9]
6. If p-value $< \alpha$, then reject null hypothesis and go for alternative
   - Be careful: (p-value $\geq \alpha$) $\not\Rightarrow$ null hypothesis is true, though it may be

Example hypotheses

- Petal.Width of each iris flowers species follow a normal distribution
- Waiting time of a supermarket checkout queue is gamma distributed

[8] We try to reject/**null**ify this hypothesis.

[9] Probability of obtaining a result equal or more extreme than observed.

Checking if Petal.Width is Normal Distributed

R example

```
# The Shapiro-Wilk-Test allows for testing if a population represented by a sample is normal distributed
# The Null-hypothesis claims that data is normal distributed

# Let us check for the full population
> shapiro.test(iris$Petal.Width)
# W = 0.9018, p-value = 1.68e-08
# Value is almost 0, thus reject null hypothesis =>
# In the full population, Petal.Width is not normal distributed

# Maybe the Petal.Width is normal distributed for individual species?
for (spec in levels(iris$Species)){
  print(spec)
  y = iris[iris$Species==spec,]

  # Shapiro-Wilk-test checks if data is normal distributed
  print(shapiro.test(y$Petal.Width))
}

[1] "virginica"
W = 0.9598, p-value = 0.08695
# Small p-value means a low chance this happens, here about 8.7%
# With the typical significance level of 0.05 Petal.Width is normal distributed
# For simplicity, we may now assume Petal.Width is normal distributed for this species

[1] "setosa"
W = 0.7998, p-value = 8.659e-07 # it is not normal distributed

[1] "versicolor"
W = 0.9476, p-value = 0.02728 # still too unlikely to be normal distributed
```

Linear Models (for Regression) [10]

- **Linear regression**: Modeling the relationship between dependent var Y and explanatory variables X_i
- Assume n samples are observed with their values in the tuples $(Y_i, X_{i1}, ..., X_{ip})$
  - Y_i is the dependent variable (label)
  - X_{ij} are independent variables
  - Assumption for linear models: normal distributed variables
- A linear regression model fits $Y_i = c_0 + c_1 \cdot f_1(X_{i1}) + ... + c_p \cdot f_p(X_{ip}) + \epsilon_i$
  - Determine coefficients c_0 to c_p to minimize the error term ϵ
  - The functions f_i can be non-linear

R example

```
# R allows to define equations, here Petal.Width is our dependent var
m = lm("Petal.Width ~ Petal.Length + Sepal.Width", data=iris)

print(m) # print coefficients
# (Intercept)  Petal.Length   Sepal.Width
#     -0.7065        0.4263        0.0994
# So Petal.Width = -0.7065 + 0.4263 * Petal.Length + 0.0994 * Sepal.Width
```

Compare Prediction with Observation

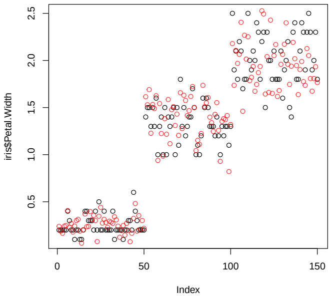

Figure: Iris linear model

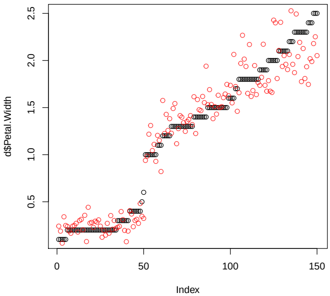

Figure: With sorted data

```
# Predict petal.width for a given petal.length and sepal.width
d = predict(m, iris)

# Add prediction to our data frame
iris$prediction = d

# Plot the differences
plot(iris$Petal.Width, col="black")
points(iris$prediction, col=rgb(1,0,0,alpha=0.8))

# Sort observations
d = iris[sort(iris$Petal.Width, index.return=TRUE)$ix,]
plot(d$Petal.Width, col="black")
points(d$prediction, col=rgb(1,0,0,alpha=0.8))
```

Analysing Model Accuracy [23]

- Std. error of the estimate: variability of c_i, should be lower than c_i
- t-value: Measures how useful a variable is for the model
- $Pr(> |t|)$ two-sided p-value: probability that the variable is not significant
- Degrees of freedom: number of independent samples (avoid overfitting!)
- R-squared: Fraction of variance explained by the model, 1 is optimal
- F-statistic: the f-test analyses the model goodness – high value is good

```
summary(m) # Provide detailed information about the model
# Residuals:
#      Min       1Q   Median       3Q      Max
# -0.53907 -0.11443 -0.01447  0.12168  0.65419
#
# Coefficients:
#              Estimate  Std.Error t value  Pr(>|t|)
# (Intercept)  -0.70648   0.15133  -4.668  6.78e-06 ***
# Petal.Length  0.42627   0.01045  40.804   < 2e-16 ***
# Sepal.Width   0.09940   0.04231   2.349    0.0201 *
# ---
# Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 " " 1
#
# Residual standard error: 0.2034 on 147 degrees of freedom
# Multiple R-squared:  0.9297,  Adjusted R-squared:  0.9288
# F-statistic: 972.7 on 2 and 147 DF,  p-value: < 2.2e-16
```

- Akaike's Information Criterion (AIC)
- Idea: prefer accurate models with smaller number of parameters
- Test various models to reduce AIC
- Improve good candidates
- AIC allows to check which models can be excluded

Time Series

- A time series is a sequence of observations
  - e.g. temperature, or stock price over time
  - Prediction of the future behavior is of high interest
- An observation may depend on any previous observation
  - Trend: tendency in the data
  - Seasonality: periodic variation

Prediction models

- Autoregressive models: AR(p)
  - Depend linearly on last *p* values (+ white noise)
- Moving average models: MA(q)
  - Random shocks: Depend linearly on last *q* white noise terms (+ white noise)
- Autoregressive moving average (ARMA) models
  - Combine AR and MA models
- Autoregressive integrated moving average: ARIMA(p, d, q)
  - Combines AR, MA and differencing (seasonal) models

Example Time Series

- Temperature in Hamburg every day at 12:00
- Three years of data (1980, 1996, 2014)

```r
d = read.csv("temp-hamburg.csv", header=TRUE)
d$Lat = NULL; d$Lon = NULL
colnames(d) = c("h", "t")
d$t = d$t - 273.15 # convert degree Kelvin to Celcius
plot(d$t, xlab="day", ylab="Temperature in C")

pdf("hamburg-temp-models.pdf", width=5,height=5)
plot(d$t, xlab="day", ylab="Temperature in C")

# Enumerate values
d$index=1:nrow(d)

# General trend
m = lm("t ~ index", data=d)
points(predict(m, d), col="green")

# Summer/Winter model per day of the year
d$day=c(rep(c(1:183, 182:1),3),0)
m = lm("t ~ day + index", data=d)
points(predict(m, d), col="blue"); dev.off()

library(forecast)
# Convert data to a time series
ts = ts(d$t, frequency = 365, start= c(1980, 1))
# Apply a model for non-seasonal data on seasonal adjusted data
tsmod = stlm(ts, modelfunction=ar)
plot(forecast(tsmod, h=365))
```

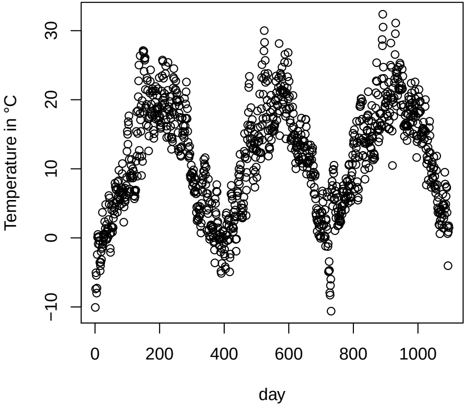

Figure: Temperature time series

Example Time Series Models

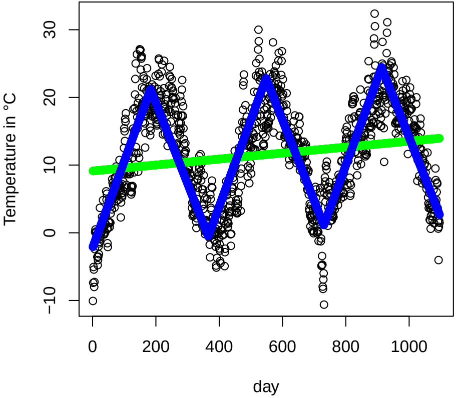

Figure: Linear models for trend and winter/summer cycle

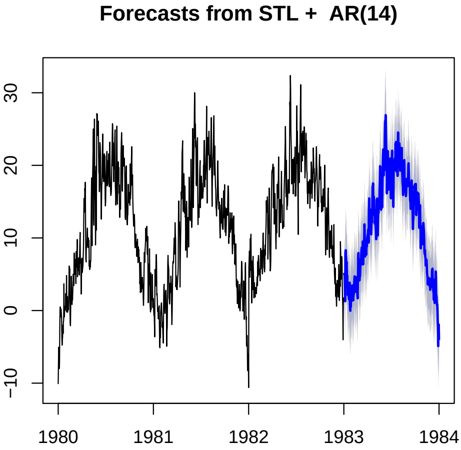

Figure: Using the forecast package/stlm()

Summary

- Data-cleaning and ingestion is a key to successful modeling
- Big data can be considered to be immutable
- Data models describe how information is organized
  - I/O middleware, relational
  - NoSQL: Column, document, key-value, graphs
- Semantics describe operations and behavior, e.g. POSIX, ACID, BASE
- Process models and programming paradigms describe how to transform and analyze data
- Hadoop ecosystem offers means for batch and real-time processing
- Lambda architecture is a concept for optimizing real-time processing
- Descriptive statistics helps analyzing samples
- Inductive statistics provide concepts for inferring knowledge

Diagrams with Python

```python
import numpy as np
import matplotlib
# Do not use X11 backend
matplotlib.use('Agg')
import matplotlib.pyplot as plt
plt.style.use('ggplot') # draw like R's ggplot
# Create 4 subplots
(fig, axes) = plt.subplots(ncols=1, nrows=4)
fig.set_size_inches(15,15) #  x, y
matplotlib.rcParams.update({'font.size': 22})
(ax1, ax2, ax3, ax4) = axes.ravel()
# Create a 2D scatter plot with random data
(x, y) = np.random.normal(size=(2, 10))
ax1.plot(x, y, 'o')

# create a bar graph
width = 0.25
x = np.arange(4) # create a vector with 1 to 4
(y1, y2, y3) = np.random.randint(1, 10, size=(3, 4))
ax2.bar(x, y1, width)
# color schemes provide fancier output
ax2.bar(x+width, y2, width,
        color=plt.rcParams['axes.color_cycle'][2])
ax2.bar(x+2*width, y3, width,
        color=plt.rcParams['axes.color_cycle'][3])
ax2.set_xticklabels(['1', '2', '3', '4'])
ax2.set_xticks(x + width)
# Draw a line plot
ax3.plot([0,1,2,3, 4, 5], [1,0,2,1, 0,1] )
# Draw a second line
ax3.plot(range(0,5), [0,2,1, 0,1] , color="blue")

# Draw a boxplot including Q1, Q2, Q3, IQR=0.5
ax4.boxplot((x, y), labels=["A", "B"], whis=0.5)
# Store to file or visualize with plt.show()
fig.savefig('matplotlib-demo.pdf', dpi=100)
```

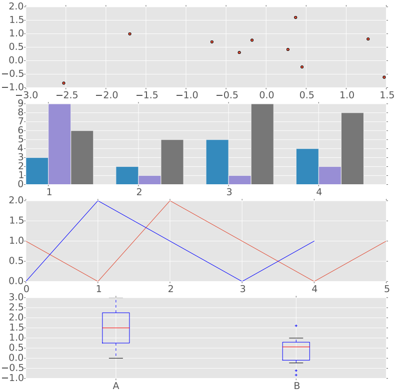

Figure: Output of the sample code

See http://matplotlib.org/gallery.html

Bibliography

4 Forrester Big Data Webinar. Holger Kisker, Martha Bennet. Big Data: Gold Rush Or Illusion?

10 Wikipedia

11 Book: N. Marz, J. Warren. Big Data – Principles and best practices of scalable real-time data systems.

12 https://en.wikipedia.org/wiki/Data_model

13 https://en.wikipedia.org/wiki/Process_modeling

14 https://en.wikipedia.org/wiki/Programming_paradigm

15 https://en.wiktionary.org/wiki/process

20 http://hortonworks.com/blog/enterprise-hadoop-journey-data-lake/

21 Book: Y. Dodge. The Oxford Dictionary of Statistical Terms. ISBN 0-19-920613-9

22 https://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient

23 http://blog.yhathq.com/posts/r-lm-summary.html

24 Forecasting: principles and practise https://www.otexts.org/fpp

25 Hypothesis testing http://www.stats.gla.ac.uk/steps/glossary/hypothesis_testing.html

26 Overcoming CAP with Consistent Soft-State Replication https://www.cs.cornell.edu/Projects/mrc/IEEE-CAP.16.pdf