Overview

• **Monday: Rule Mining**
  - Data preparation
  - Mining process

• **Today: Visualization**
  - Backend
    1. Workflow
    2. Initialization
    3. Application of rules
    4. Reverse rule matching
    5. Future prospects
    6. (Demo)
  - Frontend
Workflow

Extracted from WWW:
- Events
- Statistics

Parsed:
- Schema Matching
- Data Cleansing

Rule Mining:
- Events
- Outliers

Rule Detection

Analysis:
- Statistics
- Outliers

Outlier Detection Procedures

Visualization:
- Outliers with events and rules

Events
Statistics
Outliers
Rules
Faceted Filtering

Visualization

Initialization

Application of rules

Reverse rule matching

Frontend

Backend
Initialization

• **Problem**
  • Request takes too much time (>1 minute) if cache does not contain all required elements

• **Optimization**
  • Preloading of statistics, locations, categories, events and their relationships (e.g. hierarchies)
    • Memory needed: ~300 MB
    • Start of application: 3-10 minutes depending on database
    • All data is fetched with the minimum number of requests
    • Advantage: requests answered as fast as possible
    • Disadvantage: restart needed if data changes
Structure of an OutlierSet

OutlierSet

countryOutliers : Map<Location, List<Outlier>>

Statistic

1

tendency : int

Outlier

1..*

MatchItem

score : double

0..*

Rule

1..*

Event

0..*

0..*
### Application of rules

<table>
<thead>
<tr>
<th>Frontend</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>choose statistic</strong></td>
<td></td>
</tr>
</tbody>
</table>

#### Faceted Search:

<table>
<thead>
<tr>
<th>Statistic Category</th>
<th>Statistic Subcategory</th>
<th>Statistic</th>
<th>Location</th>
<th>Analysis Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Economy</td>
<td>Emissions</td>
<td>Income</td>
<td>Germany</td>
<td>DEFAULT</td>
</tr>
<tr>
<td>Population</td>
<td>Investment</td>
<td>Birth rate</td>
<td>France</td>
<td>Regression</td>
</tr>
<tr>
<td>Health</td>
<td>Disasters</td>
<td>diseases</td>
<td></td>
<td>ML</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Appliance of algorithms to the statistic:

1. linear regression
2. non-parametric regression
3. pitch
4. extremal
5. machine learning
6. empiric
...
13. DEFAULT
Abstract matching algorithm

```python
for outlier in outliers:
    for event in events:
        for rule in rules:
            score = rule.getScore(outlier, event)
            if score > 0:
                outlier.addEvent(event)
                outlier.addRule(rule)
```
Application of rules

Frontend

choose statistic

Locations

Statistic

Algorithm-Descriptions

Backend

perform analysis

OutlierSet

match events to outlier

events

rules

iterative

sort events per outlier

Outlier + (+ Events, Rules)

OutlierSet

Outlier

relevance = \sum_{i=1}^{#rules} score_i

- Descending Order: \rightarrow most relevant events first
- Criteria for relevance:
Application of rules

<table>
<thead>
<tr>
<th>Frontend</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>choose statistic</td>
<td>perform analysis</td>
</tr>
<tr>
<td>Statistic</td>
<td>OutlierSet</td>
</tr>
<tr>
<td>Locations</td>
<td>events</td>
</tr>
<tr>
<td>Algorithm-Descriptions</td>
<td>rules</td>
</tr>
</tbody>
</table>

**TransferObject (TO):**
- Design Pattern for the exchange of large objects
- Aim: Send only as many data as the receiver really needs
- The OutlierSet should only contain statistically values for requested locations
  → just add these values to the TO

**Statistic Algorithm - Descriptions**

- Locations
- Algorithm-Descriptions
Application of rules

**frontend**

- Choose statistic
- Locations
- Algorithm Descriptions

**backend**

- Perform analysis
- Match events to outlier
- Sort events per outlier

Iterative:

- OutlierSet
- Events
- Rules

Create TransferObjects

Visualize outliers & events
Application of rules

- **Analysis of the runtime**
  - Join is orders of magnitude slower
  - Join applies score method on all event-outlier pairs that match in time and location
  - No possibility to filter the pairs: generic score() method

- **Possible optimization**
  - Common attributes in rules needed
  - Join only matching event-outlier pairs
Application of rules

- **Analysis is done per request**
  - No performance difference compared to database load
    - Average database load: **247 ms**
    - Average analysis: **59 ms**
  - Advantage: all algorithm combinations can be used
    - Different outliers found
    - Different rules applied

<table>
<thead>
<tr>
<th>Load from db (ms)</th>
<th>35</th>
<th>289</th>
<th>71</th>
<th>36</th>
<th>263</th>
<th>34</th>
<th>33</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>24</th>
<th>847</th>
<th>755</th>
<th>1001</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis (ms)</td>
<td>32</td>
<td>32</td>
<td>171</td>
<td>172</td>
<td>17</td>
<td>29</td>
<td>25</td>
<td>28</td>
<td>24</td>
<td>108</td>
<td>25</td>
<td>38</td>
<td>24</td>
<td>102</td>
</tr>
</tbody>
</table>
Application of rules

- **Rule matching with `getScore()` method**
  - Invoked on any type of rule
  - Only applied if location and time of an event and outlier are equal
    - No location and time shift
  - Returns conviction value for an event-outlier pair
  - Should be fast

- **ManualRules and MinedRules mixed**
  - ManualRule: made by the team & based on categories
  - MinedRules should have higher score values
Reverse rule matching

- **Precalculation of mapping between rules and statistics**
  - In which statistics does a rule match to any outlier & event?
  - For one type of algorithm
    - All rules applied to the join of all events and outliers matching in location and date
    - Rules must consider the used algorithm
  - Only for default algorithm yet (~24h runtime)
  - Calculation for other algorithms needed

<table>
<thead>
<tr>
<th>Rule Type</th>
<th>Rule ID</th>
<th>Statistic ID</th>
<th>Location ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>ManualRule</td>
<td>5</td>
<td>24</td>
<td>132</td>
</tr>
<tr>
<td>MinedRule</td>
<td>789</td>
<td>356</td>
<td>12</td>
</tr>
</tbody>
</table>
Reverse rule matching

<table>
<thead>
<tr>
<th>Frontend</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>choose rule</td>
<td></td>
</tr>
</tbody>
</table>

Choose rule

- Start with a known rule from a previous analysis
Reverse rule matching

<table>
<thead>
<tr>
<th>Frontend</th>
<th>Backend</th>
</tr>
</thead>
<tbody>
<tr>
<td>choose rule</td>
<td>find statistics for rule</td>
</tr>
<tr>
<td>Rule</td>
<td></td>
</tr>
</tbody>
</table>

**Application of rule**

- Use the rule-statistic mapping
- Because the mapping is not preloaded, the statistic information is loaded from the database
Reverse rule matching

Choose rule

Find statistics for rule

Choose statistic

Application of rule

1. Perform analysis
2. Match events to outliers
   - just use the given rule in this step
3. Sort events per outlier
4. Create TransferObjects
Reverse rule matching

**Frontend**
- choose rule
- choose statistic

**Backend**
- find statistics for rule
- application of rule

- Statistic-Location-Mapping
- Statistic
- Locations
- Algorithm-Descriptions
- Rule

- visualize outliers & events
- OutlierSet

**Algorithm Descriptions**

**Locations**

**Application of Rule**
Further completed tasks

- **Calculation of outliers for default algorithm**
  - Saved in database (runtime ~2 h)

- **Cleaning of statistic data**
  - No empty statistics
  - At least 4 values needed per location in a statistic
  - Not yet applied to the data in the database
    - Reload necessary!
Future prospects

- Optimization of the generic getScore() method
  - Common attributes: join only some of the event-outlier pairs

- Evaluate time and location shift in the rules and not in the join algorithm

- Determine threshold for getScore() method

- Remove ManualRules or determine appropriate score values
  - Influences ordering of events

- Evaluation of algorithm parallelism, especially R
  - Allow multiple user requests to run in parallel
<table>
<thead>
<tr>
<th>Year</th>
<th>Month</th>
<th>Day</th>
<th>Value</th>
<th>Tendency</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1961</td>
<td>0</td>
<td>0</td>
<td>0.46031542505449</td>
<td>1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1962</td>
<td>0</td>
<td>0</td>
<td>0.01130974214627</td>
<td>1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1963</td>
<td>0</td>
<td>0</td>
<td>0.020928228487012</td>
<td>1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1964</td>
<td>0</td>
<td>0</td>
<td>-0.00482276344353</td>
<td>-1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1965</td>
<td>0</td>
<td>0</td>
<td>2.0389008235117</td>
<td>1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1966</td>
<td>0</td>
<td>0</td>
<td>0.002212634104945</td>
<td>-1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1967</td>
<td>0</td>
<td>0</td>
<td>0.01086777513565</td>
<td>-1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1968</td>
<td>0</td>
<td>0</td>
<td>13.7645217104721</td>
<td>1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1969</td>
<td>0</td>
<td>0</td>
<td>11.2804632893197</td>
<td>-1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1970</td>
<td>0</td>
<td>0</td>
<td>16.0753026253734</td>
<td>1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1971</td>
<td>0</td>
<td>0</td>
<td>15.94466969113456</td>
<td>-1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1972</td>
<td>0</td>
<td>0</td>
<td>68.7673493685513</td>
<td>-1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1973</td>
<td>0</td>
<td>0</td>
<td>60.195911746892</td>
<td>-1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1974</td>
<td>0</td>
<td>0</td>
<td>101.932960893855</td>
<td>1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1975</td>
<td>0</td>
<td>0</td>
<td>53.9224440340923</td>
<td>-1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1976</td>
<td>0</td>
<td>0</td>
<td>60.8095364437641</td>
<td>-1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1977</td>
<td>0</td>
<td>0</td>
<td>46.1448295294449</td>
<td>-1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1978</td>
<td>0</td>
<td>0</td>
<td>51.5792294089277</td>
<td>1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1979</td>
<td>0</td>
<td>0</td>
<td>115.90958795699</td>
<td>1 (Trend)</td>
<td>0 (Events)</td>
</tr>
<tr>
<td>1980</td>
<td>0</td>
<td>0</td>
<td>53.4839367669556</td>
<td>1 (Trend)</td>
<td>0 (Events)</td>
</tr>
</tbody>
</table>
Overview

- **Monday: Rule Mining**
  - Data preparation
  - Mining process

- **Today: Visualization**
  1. Backend
  2. Frontend
    1. Demo
    2. Tools
    3. Implementation Details
    4. Conclusion
Live Demo!
Google Web Toolkit:

SDK + Eclipse Integration (+ Designer)

Google Chart Tools: Annotated Timeline
Google Web Toolkit

Client Code

Javascript

Browser

Java Code

Servlet/Java Byte Code

Servlet Container (Tomcat)
GWTs RPC Mechanismus

ServiceDefTarget (interface)  
RemoteService (interface)  
RemoteServiceServlet (class)  
StockPriceServiceAsync (interface)  
StockPriceService (interface)  
StockPriceServiceImpl (class)  

stockPricesSvc = GWT.create(StockPriceService.class); (service proxy class)  

Translatable Java code (runs as JavaScript on client)  
Standard Java code (runs as bytecode on server)  

Frontend - Implementation Details

- DatabaseServiceImpl

Server

Client

Control

Data

UI
Frontend - Implementation Details

- DatabaseServiceImpl

Mock + Database implementation
Frontend - Implementation Details

- DBService
- DBServiceAsync
- BlackSwanVis

Diagram:

- Server
- Client
  - Control
  - Data
- UI
Frontend - Implementation Details

- Model
- Data classes

```
Server
---
Client
---
Control
  Data
  UI
```
Frontend – Data classes

Graph
- statisticId
- countryId

Point
- value

Annotation
- category
- subCategory
- fulltext
- sourceName
- sourceLink
- title

RuleVis
- type
- name
- algorithmName
- exampleCount
- statisticCountryMap

Algorithm
- name
- description
- enabled

AlgorithmOption
- name
- description
- defaultValue
- value
Frontend - Implementation Details

- State
- Configuration

Server

Client

Control

Data

UI
Frontend - Implementation Details

- Custom UI elements
## Frontend - Faceted search algorithm

### Reduced by..
- Category:  
- Subcategory: 
- Country: 

### Table

<table>
<thead>
<tr>
<th>Category</th>
<th>Subcategory</th>
<th>Statistic</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Economy</td>
<td>Aid given</td>
<td>Consumer price index</td>
<td>Canada</td>
</tr>
<tr>
<td></td>
<td>Billionaires</td>
<td>Dollar billionaires per million people</td>
<td>Denmark</td>
</tr>
<tr>
<td></td>
<td>Emissions</td>
<td>Education aid (% of total aid)</td>
<td>France</td>
</tr>
<tr>
<td></td>
<td>Geography</td>
<td>Forest Area</td>
<td>Germany</td>
</tr>
<tr>
<td></td>
<td>Inflation</td>
<td>Health aid (% of total aid)</td>
<td>South Africa</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yearly CO2 Emissions</td>
<td></td>
</tr>
</tbody>
</table>
**Frontend - Faceted search algorithm**

Reduced by:
- **Category**: 4, 6
- **Subcategory**: 6
- **Country**: 1, 2, 3, 4, 6

<table>
<thead>
<tr>
<th>Category</th>
<th>Subcategory</th>
<th>Statistic</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Economy</td>
<td>Aid given</td>
<td>Consumer price index</td>
<td>Canada</td>
</tr>
<tr>
<td></td>
<td>Billionaires</td>
<td>Dollar billionaires per million people</td>
<td>Denmark</td>
</tr>
<tr>
<td></td>
<td>Emissions</td>
<td>Education aid (% of total aid)</td>
<td>France</td>
</tr>
<tr>
<td></td>
<td>Geography</td>
<td>Forest Area</td>
<td>Germany</td>
</tr>
<tr>
<td></td>
<td>Inflation</td>
<td>Health aid (% of total aid)</td>
<td>South Africa</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Yearly CO2 Emissions</td>
<td></td>
</tr>
</tbody>
</table>
Frontend - Faceted search algorithm

<table>
<thead>
<tr>
<th>Category</th>
<th>Subcategory</th>
<th>Statistic</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Economy</td>
<td>Aid given</td>
<td>Consumer price index</td>
<td>Canada</td>
</tr>
<tr>
<td></td>
<td>Billionaires</td>
<td>Dollar billionaires per million people</td>
<td>Denmark</td>
</tr>
<tr>
<td></td>
<td>Emissions</td>
<td>Education aid (% of total aid)</td>
<td>France</td>
</tr>
<tr>
<td></td>
<td>Geography</td>
<td>Forest Area</td>
<td>Germany</td>
</tr>
<tr>
<td></td>
<td>Inflation</td>
<td>Health aid (% of total aid)</td>
<td>South Africa</td>
</tr>
</tbody>
</table>

Intersection: 6
## Frontend - Faceted search algorithm

<table>
<thead>
<tr>
<th>Category</th>
<th>Subcategory</th>
<th>Statistic</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>Economy</td>
<td>Aid given</td>
<td>Yearly CO2 Emissions</td>
<td>Canada</td>
</tr>
<tr>
<td></td>
<td>Billionaires</td>
<td></td>
<td>Denmark</td>
</tr>
<tr>
<td></td>
<td>Emissions</td>
<td></td>
<td>France</td>
</tr>
<tr>
<td></td>
<td>Geography</td>
<td></td>
<td>Germany</td>
</tr>
<tr>
<td></td>
<td>Inflation</td>
<td></td>
<td>South Africa</td>
</tr>
</tbody>
</table>
Frontend - Conclusion

- If it's a Java project – use GWT
  - Nice IDE support
  - Good development cycle
- Use of Annotated Timeline visualization:
  - Great for this prototype
  - But: No sourcecode, no changes
  - But: Hacky DOM manipulations
- Not truly Open Source

- Visualize earlier in the development process
- Keep it simple – especially for prototypes
Frontend – Future work

- Security, performance and cross browser testing
- Improve rule exploration
  - Don’t use annotation sidebar of the Annotated Timeline
  - Make rule names changeable
- Improve country names
- Add browser history support
  - Create unique URLs for views on the data