FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

TempMunger

Ein Visual Analytics Ansatz zur Transformation Zeitorientierter Daten

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Information & Knowledge Management

eingereicht von

Robert Thurnher, Bakk.
Matrikelnummer 0004297

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dr. Silvia Miksch
Mitwirkung: Dr. Theresia Gschwandtner

Wien, 13. April 2017

Robert Thurnher	Silvia Miksch

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

TempMunger

A Visual Analytics Approach Supporting Transformations of Time-Oriented Data

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Information & Knowledge Management

by

Robert Thurnher, Bakk.
Registration Number 0004297

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dr. Silvia Miksch
Assistance: Dr. Theresia Gschwandtner

Vienna, 13th April, 2017

Robert Thurnher	Silvia Miksch

Erklärung zur Verfassung der Arbeit

Robert Thurnher, Bakk.
Neubaugasse 64-66, A-1070 Wien; robert[at]thurnher.email

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 13. April 2017

Robert Thurnher

Acknowledgements

Dedicated to my **father**[1], my **mother**, and my **wife**.
This wouldn't have been possible if it weren't for you.
May it also motivate my **sister** pursuing her dreams.
♡

I'd like to thank *Theresia Gschwandtner* for her supervision, understanding, and patience.

Further thanks go out to my **brother** and *Michael Jakl* for their support and inspiration.

Generally, I'm grateful to everyone who has test-driven the prototype, proofread drafts, and/or provided feedback – particularly, my **sister-in-law** and *CVAST* research staff.

Thank you!

[1] ***RIP.***

Kurzfassung

Data-Wrangling ist im Allgemeinen die mühevolle Aufbereitung von Daten um diese für nachfolgende Analysen nutzbar zu machen. Normalerweise bedeutet das handgefertigte Skripte anzuwenden, was einen gewissen Grad an technischer Expertise verlangt. Um derartiges Aufbereiten von Daten auch für User, die eher einer "Casual"-Kategorie zuordenbar sind, zugänglich zu machen haben wir zur Erleichterung damit verbundener Aufgaben, im Kontext dieser Arbeit einen *Visual-Analytics*-Prototypen erstellt.

Unser Ansatz ist ein interdisziplinärer, der zeitgemäße Konzepte und Ideen aus diversen Gebieten kombiniert: *Mensch-Computer-Interaktion* und *Usability-Engineering* mit *Information-Retrieval*, *Data-Mining*, *Machine-Learning* sowie *Informationsvisualisierung*.

In diesem Zusammenhang konzentrieren wir uns speziell auf oftmals nötige Transformationsschritte von zeitorientierten Daten. Diese Art von Daten weist einzigartige Charakteristika auf, die bei verschiedenen Lösungsansätzen gesondert berücksichtigt werden müssen. Basierend auf den Ergebnissen einer Analyse des State-of-the-Art in diesem Bereich, haben wir offene Fragestellungen sowie einige Anforderungen für einen solchen Prototypen abgeleitet und folglich einen Software-Prototyp namens ***TempMunger*** in einer agilen Art und Weise iterativ sowohl designt als auch entwickelt. Der Prozess sowie dazugehörige Artefakte werden umfassend dokumentiert und dargestellt.

Unser Prototyp ist eine web-basierte Anwendung, die für den Gebrauch mittels Desktop-Browser zugeschnitten ist. Genauer gesagt, bietet sie interaktive Dashboard-Visualisierungen für möglichst intuitive sowie explorative Transformationsoperationen zeitorientierter Daten. Eine qualitative Evaluierung des Prototypen zeigte, dass diese Funktionalität als nützlich empfunden wird. Außerdem ergaben sich verschiedene Ideen für künftige, weiterführende Arbeiten.

Abschließend kann man sagen, dass Data-Wrangling ein spannendes Forschungsgebiet bleibt, in dem es noch viel zu entdecken gilt.

Abstract

Data wrangling generally denotes the cumbersome task of making data useful for analysis. Usually, this means applying hand-crafted scripts, requiring at least a certain degree of technical expertise. In order to make suchlike data preparation accessible for rather casual users as well, we have built a *visual analytics* prototype in the context of this thesis, easing related tasks.

Our approach is an interdisciplinary one, combining contemporary concepts and ideas from various fields: *human-computer interaction* and *usability engineering* with *information retrieval*, *data mining*, *machine learning*, plus *information visualization*.

In particular we focus on supporting transformations of time-oriented data since this kind of data exhibits unique characteristics which demand for special consideration. After analyzing related state of the art we identified open issues and derived requirements for such a system. We followed an iterative design process to develop a software prototype called ***TempMunger***, in an agile manner. The process as well as corresponding artifacts are documented and presented in this thesis.

Our prototype is a web-based application, tailored for desktop browser usage. More concretely, it offers interactive dashboard visualizations for preferably intuitive and exploratory transformation operations of time-oriented data. A qualitative evaluation of the prototype demonstrates its usefulness and reveals opportunities for future work.

Concluding it is safe to say that data wrangling continues to be an exciting field of research where much is yet to be discovered.

Contents

CHAPTER 1

Introduction

1.1 Motivation

Applied work within data science, and more specifically **Visual Analytics (VA)**, *"the science of analytical reasoning facilitated by interactive visual interfaces"* [TC05] (p. 4), can be seen to roughly consist of three main basic building blocks [Mik10]:

1. **Data wrangling** a.k.a. **munging**
2. Statistics & Machine Learning (ML)
3. Visualization & analysis itself

While the second and last mentioned fields are constantly evolving related tools and techniques, the first one is comparatively still lacking in terms of high-level support. This is when it comes down to actually wrangle usually messy real-world data into a format prepping it useful for analysis.

To this day, it mostly means fiddling around with the data manually, applying hand-crafted transformation scripts. This is a tedious task and discourages analyzing data altogether, especially if the ones intending to work with the data are technically unversed (for instance, journalists or business analysts).

However, combining contemporary knowledge and technology from the domains of *Human-Computer Interaction (HCI)* [Coo04] & *User Experience (UX)* [Nor02] as well as *Information Retrieval (IR)* [MRS08], *data mining / ML* [WFH16], and *Information Visualization (InfoVis)* [Tuf01] [CMS99], could yield substantial improvements here.

Namely, making data wrangling accessible to a wider audience, mainly by providing interactive analytical visualizations allowing for easy data transformations and giving immediate visual feedback.

In this thesis we are investigating the field of data wrangling, plus, designing, and prototypically implementing a VA approach to support the tasks involved. That is, iteratively creating a software prototype which enables users to wrangle data suitable for analysis in an intuitive, interactive, and visual way.

VA and InfoVis enable uncovering the non-obvious by using interactive visualizations. Consequently, this approach is especially useful when applied to complex tasks where clear overview and structuring is necessary in order to achieve effective results efficiently. That is also why employing related techniques can generally be expected to be more successful for the outlined use case than fully automated ones. Hence, an ideal solution would probably be a blend of automated suggestions, augmenting an interactive interface driven by visualizations.

Additionally, the to-be-developed prototype shall, in particular, make it convenient to work on time-oriented datasets. Time itself and time-oriented data, specifically, possess distinct characteristics (for instance, regarding scale, scope, structure, viewpoint, and granularity) that make it worthwhile to dedicatedly treat it as a separate data type, as suggested by Aigner et al. [AMST11]. This is why time-oriented data demands for special data wrangling solutions which have not been adequately tackled yet.

1.2 Problem Statement

The main research question in the context of this work is:

How can we support data wrangling with VA techniques?

More particular, we are to tackle the following subquestions:

- *Which data transformations are best supported by analytical methods and for which transformations is visual support beneficial?*
- *How do concrete data wrangling workflow processes look like and how can these processes be supported by VA methods?*
- *What data wrangling tasks need to be tackled in particular when dealing with time-oriented data and how can we support them with VA methods?*

The emphasis of this thesis is put on evaluating the feasibility of corresponding concepts via iterative design, implementation, and evaluation of a software prototype.

1.3 Aim of the Work

Results to be achieved are:

- Design and implementation of a research prototype

- Evaluation of the prototype to answer research questions
- Detailed documentation of these

At the end of the project, it should be known whether developing such a **prototype** supporting these tasks is feasible and if so, how in detail. As mentioned above, the approach shall combine concepts from HCI & UX with ones from IR, ML, and VA. Special focus will be laid on crafting the **User Interface (UI)** and an underlying **analytical inference engine**, interactively providing the user with respective data quality and potential issue suggestions visually. Furthermore, **direct manipulation** of data should be easy. Plus, transformations shall, generally, be easily applicable to **batches of data** by making use of bulk operations, while again being visual-interactively supported. A central challenge is making this all work in the context of **time-oriented data**.

Answers to the stated research questions are intended to be given by designing, implementing, and evaluating a research prototype which provides VA techniques to improve and support data wrangling tasks.

1.4 Methodological Approach

In order to answer our research questions we follow the nested model by Munzner [Mun09]. In particular we include the following steps:

1. **State of the art review**: giving an overview of the topic, spanning from scientific foundation to various approaches, and serving as first input to our approach.
2. **Requirements analysis**: determining which features to implement, directed to answering why, what, and how.
3. **Design of interactive visualizations**: subsequent iterative design process, making use of state-of-the-art tools.
4. **Iterative prototypical implementation**: following an agile approach until satisfying results are achieved.
5. **Qualitative evaluation of results**: for validating fulfillment of goals and, consequently, leading to answering our research questions.

In the context of this work, we also consider the *data–users–tasks* triangle for VA. This triangle can be seen in Figure 1.1 and is introduced in [MA14]. It is particularly useful for initial requirements analysis of a visualization project. Consequently, we apply it as a starting point. Its basic notion is to think of a product for VA within the three dimensions of data, users, and tasks separately, first, and then in interplay. The latter, thus, defines effectiveness, expressiveness, and appropriateness of the to-be-chosen interactive VA methods, as visualized in the diagram.

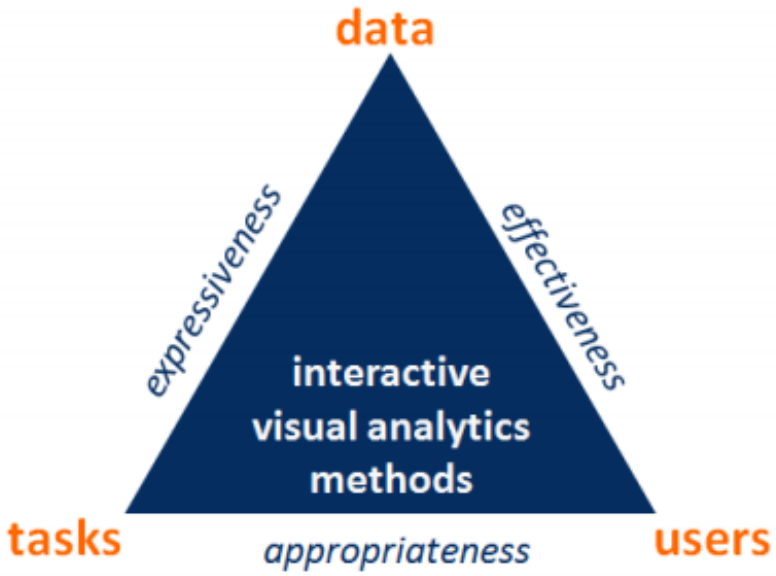


Figure 1.1: The *data–users–tasks* triangle as presented in [MA14].

All steps are accurately documented, emphasizing findings of the evaluation and lessons learned. The thesis covers all aspects and results of the design, development, and evaluation of the prototype, from mockups and architectural diagrams to test results.

1.5 Structure of the Work

The thesis is structured as follows:

1. An **introduction** to the topic and thesis itself: this presents the motivation, problem statement, and methodological approach.

2. A **state of the art** review: this is laying out the scientific foundation regarding our topic and showing various approaches. The latter are, thus, analyzed and compared. It is closed by a crossover to the requirements analysis for our approach, fueled with preceding input.

3. A **design and implementation** chapter: this is describing, explaining, and documenting all connected steps thoroughly. The design section consists of creating personas, wireframing with mockups, and consequently deriving our requirements. The implementation section walks through the developed prototype, looking at it from varied angles, and presents its qualitative evaluation.

4. A concluding section with **critical reflection** of the achieved work: this includes comparing with related work, discussing open issues, and answering our research questions. It is also concerning the requirements we have previously derived.

5. Finally, closing the thesis with a **summary and future work** section: this is just briefly summing up our work and offering an outlook.

6. At the end, there is an **appendix**: this contains a detailed presentation of the software design and architecture of our prototype as well as scientific background, plus further implementation details.

CHAPTER 2

State of the Art

The report or review contained in this chapter focuses on summarizing and comparing approaches, techniques, and related work within the field which can generally be subsumed by the term *"**data wrangling**"* with an emphasis on **visual-interactive support**. This is a relatively young field of interdisciplinary Research and Development (R&D). It is basically about the process of making any data useful for analysis. This reviewing report is mainly geared towards offering an overview of the topic in order to comprehensively show differences of existing approaches as well as commonalities shared between them. Furthermore, special attention is paid to setting it all into the historic evolutionary context of the field. In the end, we can see that there is still some need for exploration and improvements here, particularly regarding visual-interactive components and the support for time-oriented data.

This chapter gives a brief overview of related work, i.e., its scientific underpinnings, followed by a walkthrough of different concrete approaches, techniques, and related tools.

2.1 Literature Study

Generally, [KPHH11] was used as a point of origin to find other relevant references. Then, going through the list of references of these papers, relevance was determined via personal evaluation. Plus, we scanned the reference lists of thus relevant papers.

Moreover, we searched the archives of topic-related journals (i.e., *Information Visualization Journal* and *ACM Journal of Data and Information Quality*) as well as scientific, electronic databases (*ACM Digital Library*, *IEEE Xplore*, and also *Google Scholar*) and proceedings of relevant conferences (*International Conference on Information Visualization*, *IEEE Conference on Visual Analytics*, *ACM SIGCHI Conference on Human Factors in Computing Systems*, etc.).

Our search terms included *"data wrangling"*, *"data cleansing"*, and variations of these phrases. *Mendeley*[1] was used as a tool to conveniently keep track of references and organize our bibliography.

We want to span a comprehensive overview of different approaches to the topic, focusing on the evolution over time here.

Basics of the field are laid out in [DJ03], mainly relating to **Extract, Transform, Load (ETL)** processes as classically known from **data warehousing**. General theoretical foundation of transforming large amounts of data interactively can be found in [DJMS02] and more recently [Hel08]. While the former can be seen as a valuable general resource on the topic with focus on Relational Database Management System (RDBMS), the latter is especially interesting as a contemporary specimen spanning a rather wide range of scientific ground. In particular, [Hel08] explains the statistics theory backing things like outlier detection and puts it into applied context, e.g., via showing corresponding SQL queries implementing this. In the end, there is a section focusing on actual interface design principles regarding the topic of exploratory quantitative data cleaning. To complete the selection here, basic algorithmic theory of interactive schema mapping and its application is covered, e.g., by [CKP08]. The matter is relevant in this regard as it is concerned with interactively (and usually visually) transforming data (schemas in this case) as well.

More recently, another interesting field of research commonly referred to as *"Learning by **Example**"* or more concretely *"**Programming by Example (PBE)**"* has emerged. Traditionally, there has been the approach of *"Learning/Programming by Demonstration (L/PBD)"* which requires users to specify start, end, as well as intermediate states of the respective task at hand to be automated. See [Gul10] for an overview survey of the foundations of this research area, more generally called *"**program synthesis**"*. Based on the work of [Gul11], Microsoft Excel 2013 is incorporating such concepts (i.e., particularly applied PBE) into commercial software for end-users. That is, users can show the application which parts of the data they are interested in by simply marking or highlighting examples and an underlying engine infers corresponding wrangling transformations from that – without requiring demonstration of any intermediary steps.

2.2 Analysis

This section presents more concrete approaches and examples, historically aligned, followed by a comparison and outlook for our prototype.

2.2.1 Potter's Wheel: A Pioneer

A system pioneering this area is called *Potter's Wheel*, [RH01]. It offers a spreadsheet-like Graphical User Interface (GUI) for interactively specifying and executing data transformations (see Figure 2.1 for a screenshot, as presented in the paper).

[1] www.mendeley.com

This can be done either via *L/PBD* (i.e., the user "shows" what he wants to achieve and the system tries to infer actions from this) or somewhat direct manipulation via corresponding menus. Yet, among other things, it does not support "fill" transform operations (that is, automatically filling certain fields with certain data, batch-wise). Plus, naturally, its usability is not up to modern standards.

On the **pro** arguments side: important step into the direction of visual-interactively supporting users in data wrangling tasks.

On the **con** arguments side: as stated above, raw (also due to its pioneering status) implementation with room for improvements and there is no real charting support.

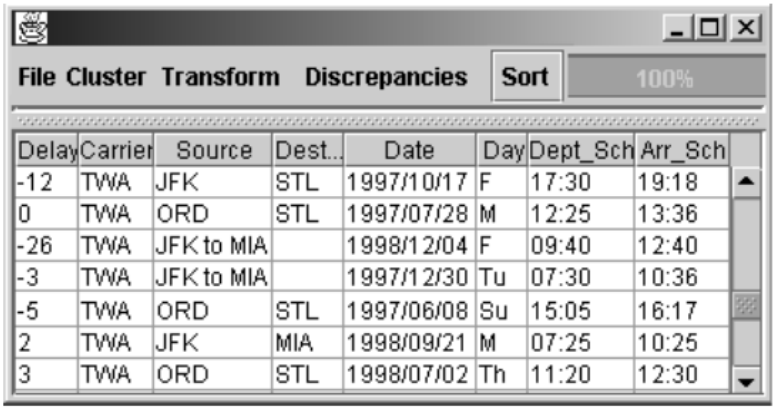


Figure 2.1: Showing Potter's Wheel spreadsheet-like GUI approach. Transform operations are accessible via menu bar [RH01].

2.2.2 PADS: A Domain-Specific Language Approach

The system of [FG05] approaches the topic of transforming large amounts of semi-structured data via a descriptive **Domain-Specific Language (DSL)**. Consequently, it is lacking a visual-interactive component altogether and taken into consideration here for the sake of completeness and demonstrating a different solution view. Users describe the data that is to be transformed in the DSL. Then the *PADS* compiler allows for generating (C-based) parser libraries and built-upon tools for further processing of such data. The project was conducted at *AT&T* and focuses on specific data transformation requirements of the company at that time. Target output data formats are in particular XML and ones for loading the data into RDBMSes (i.e., SQL). Mainly supported input formats are of ASCII (log files), binary (legacy networking protocols), and Cobol kind.

Here is some exemplary web server access logs input data as shown in the paper:

```
207.136.97.49 - - [15/Oct/1997:18:46:51 -0700] "GET /tk/p.txt HTTP/1.0" 200 30
tj62.aol.com - - [16/Oct/1997:14:32:22 -0700] "POST /scpt/dd@grp.org/confirm HTTP/1.0" 200 941
```

A **pro** argument for this approach is that the DSL is relatively concise and easily graspable due to its declarative nature. Furthermore, it frees its users from manually writing parsers and constructive tools as these are generated from the DSL definitions of a concrete data format to work with. Finally, the quality and range of tools can be improved and extended in a convenient way being quite independent from the generated parsers.

If we wanted to find a **con** argument here it could be that while being useful and a general step forward at the time of development, we believe that there are probably better ways to support certain kinds of end-users in these tasks, namely of the visual-interactive kind which we want to evaluate with this thesis after all.

2.2.3 Microsoft BizTalk Schema Mapper Research

In the approach of [RCC05], Microsoft's *BizTalk* schema mapper application is attempted to be improved. Main use case being visual and interactive support of mapping two differing XML schemas. Moreover, the schemas are of considerable big size (i.e., thousands of elements in each schema) to be not processable in a usable way within the traditional UI. That is, too many connections between the schemas are shown in a too complex way to be useful. To improve the UI some InfoVis techniques are applied (Figure 2.2 presents a screenshot showing this).

As described by the authors, these techniques are:

- **Highlight propagation:**
  (de-)emphasizing areas of interest within the mapping UI
- **Auto-scrolling:**
  automatically scrolling to selected items (comparable to IDEs) etc.
- **Coalescing trees:**
  somewhat relating to highlight propagation hiding irrelevant nodes
- **Multi-select:**
  reasonable support for multi-selection of nodes concerted with general visualization
- **Incremental search:** live search results updating the UI while typing
- **Bendable links:** enabling the user to see connections when actually overlayed
- **Focus on linked elements:** improved default behavior of up/down keys usage

The results are evaluated via a user study and have proven to be useful which can be seen as a **pro** argument here as well.

There is little to mention on the **con** side regarding the approach of [RCC05] as it overall demonstrates an advancement in the area of interactive schema mapping.

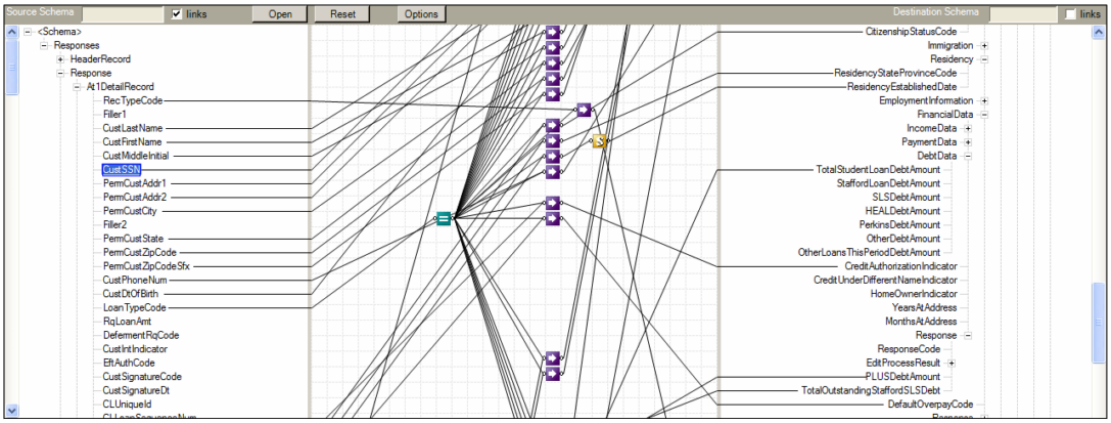


Figure 2.2: BizTalk research GUI demonstrating some applied InfoVis techniques. On each vertical side of the visible pane, elements of a respective schema are represented by a tree-like structure. Mappings between them are visualized as arcs. Center nodes are used for displaying more complex (m:n-like) mappings [RCC05].

2.2.4 Clio: From Research Prototype to Industrial Tool

In [HHH+05] the authors outline the evolution of an IBM research prototype to an industrial-strength tool. *Clio* is an application for descriptive specification of schema mappings. It is mainly targeted towards XML and relational schemas (see Figure 2.3). So, in this respect it is quite similar both by field and application to the aforementioned Microsoft *"BizTalk"* schema mapper product, adding RDBMS/SQL mapping to the mix. Mappings are represented by an abstract query graph allowing for translation of data transformations into specific query languages. This is what makes it interesting and distinguishing, again, when compared with Microsoft's *BizTalk* schema mapping application.

Finally, the paper focuses on revisiting the architecture and algorithms behind the software and then explains issues and solutions on the way to applying industrial usage. Consequently, it is a useful real-world example of transferring science in practice.

2.2.5 Potluck: A Semantic Web Approach

The research prototype in [DFH07] focuses on making heterogeneous ***semantic web*** data accessible to casual users. It provides a web interface for visual and interactive wrangling of data mainly relating to Resource Description Framework (RDF). An interesting feature is allowing users to tag and visualize data with physical locations via a map view (see Figure 2.4). What is also interesting in this approach is its demonstration of increased interest respectively shift towards web technologies at that time. Furthermore, it delivers some kind of preview how far web-based approaches would go in this area. Plus, it illustrates the general trend in the direction of casual end-user applications (enabling non-expert users to operate actually complex tasks).

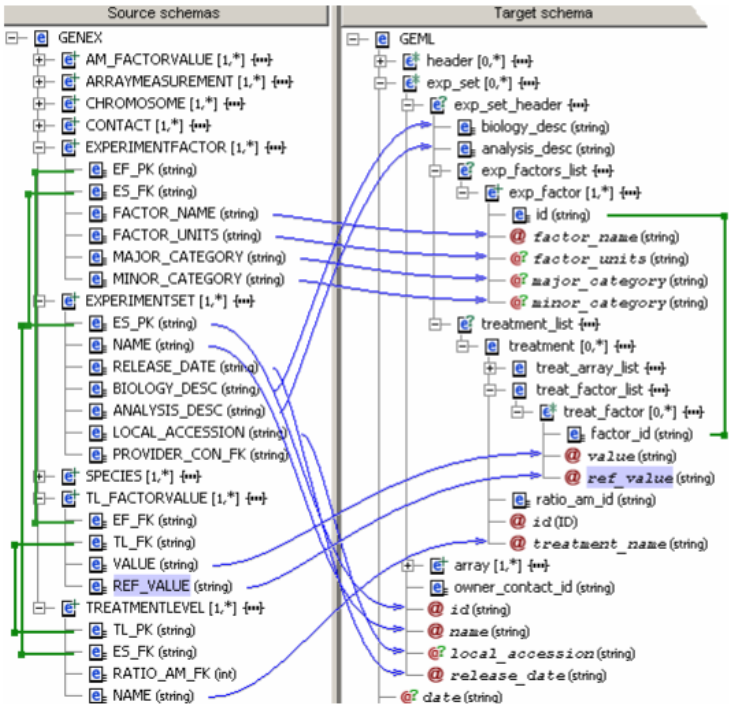


Figure 2.3: Clio GUI, also showing similarities with the MS BizTalk one. Consequentially, again, data schemas (this time from relational DBs, though) on the left and right-hand sides with mappings as arrowed lines [HHH+05].

2.2.6 R&D Veterans: DataWrangler & OpenRefine

The two systems that can be the seen to have set the benchmark in this space are:

1. ***DataWrangler*** (*Stanford Visualization Group* research project[2]) [KPHH11]
2. ***OpenRefine*** (open-sourced[3] product[4] f.k.a. *Google Refine* and *Freebase Gridworks*)

DataWrangler

DataWrangler, cf. Kandel et al. in [KPHH11], is a web-browser-based application oriented towards a visually interactive approach (see Figure 2.5). It contains an inference engine suggesting transforms, plus, data cleaning sessions can be exported and reused as scripts. The creators of the project meanwhile moved on to make a commercial product out of it, called *Trifacta Wrangler*[5].

[2] vis.stanford.edu/wrangler/
[3] github.com/OpenRefine/OpenRefine
[4] openrefine.org
[5] www.trifacta.com/products/wrangler/

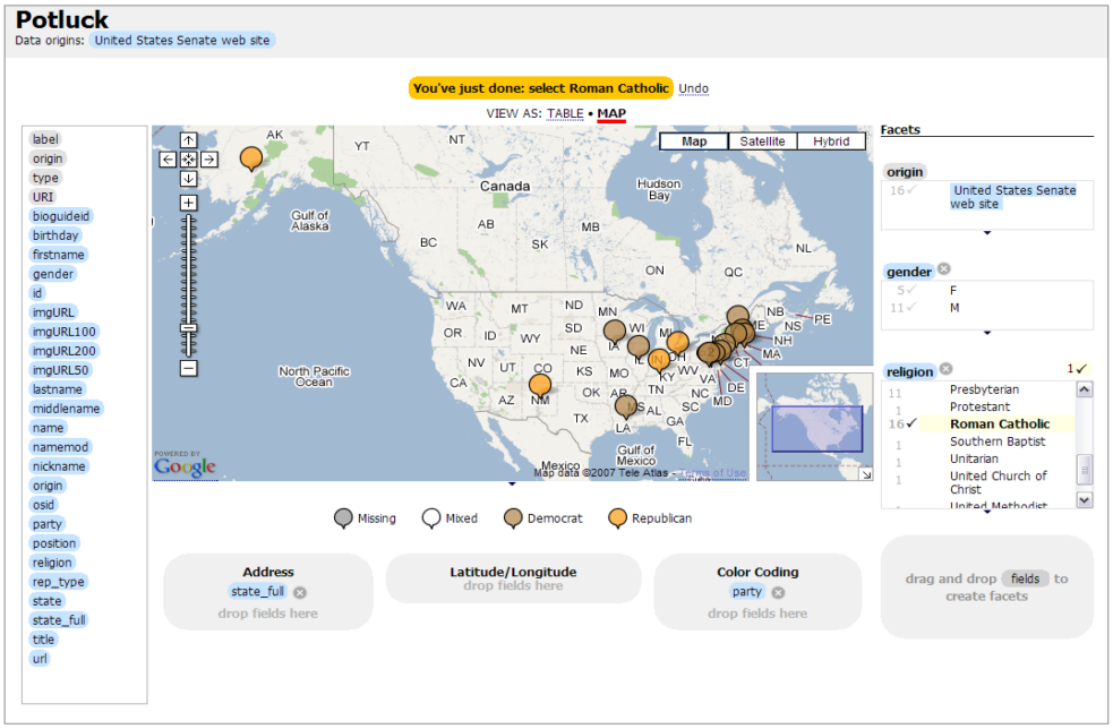


Figure 2.4: Potluck map view which can be seen as a back then innovative visualization in the field. Data is displayed geospatially [DFH07].

One thing which is less supported by *Wrangler* is full-control direct manipulation of data. That is, it is more geared towards L/PBD than to deliberately changing the content of single values via direct input.

In addition to that, the UI is limited which can also be ascribed to the web-based nature of the tool. Things like not truly responsively, and richly interactive UX, especially when the amounts of data that are to be wrangled grow.

DataWrangler was evaluated via a user study and has shown to improve productivity of data wrangling tasks considerably.

Biggest **pro** arguments of the approach:

- Interactively inferred suggested transforms
- Rich collection of transforms to be applied
- Shows the potential of pursuing the visual-interactive path

Main **con** argument is performance-wise constrained usefulness of the tool and its generally improvable UX. Also, no real charting support available, yet.

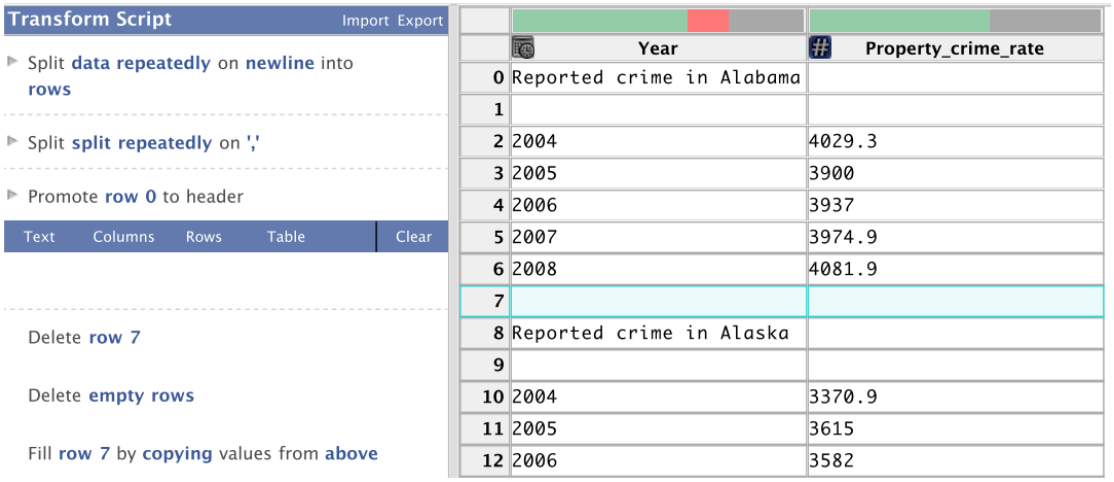


Figure 2.5: DataWrangler UI as a state of the art shaping, innovative approach. Data transform history and related suggestions are located on the left, tabular interaction pane on the right [KPHH11].

OpenRefine

OpenRefine is a browser-based application as well (but running locally on the user's machine, also due to data protection respectively privacy reasons). It allows for direct manipulation of data (see Figure 2.6), yet, it supports less interaction-driven transform operations than DataWrangler (which infers appropriate transformation suggestions from solely pointing the cursor to data in a certain way).

A nice feature is its visual statistical analytics of data distribution via histograms etc. ($\rightarrow$ **pro** argument). But, on the **con** side, it generally lacks some data transformation operations which are supported by DataWrangler (particularly reshaping-related, like un/folding – extracting/merging specific parts of data within a column into/from additional ones).

2.2.7 Temporal Research: On Time Series Data

The research prototype in [BRG+12] focuses specifically on time-series-based data. It is tailored for joint usage of a domain and a data mining expert. Technically, a preprocessing pipeline is visual-interactively created and incrementally adjusted via the tool (see Figure 2.9).

What is interesting here is the application of the domain on time series data. Moreover, its pipeline-oriented workflow UI, offering various statistical visualizations, is a well-designed asset. Finally, the approach is evaluated by applying it to a case study delivering results proving it useful in its context.

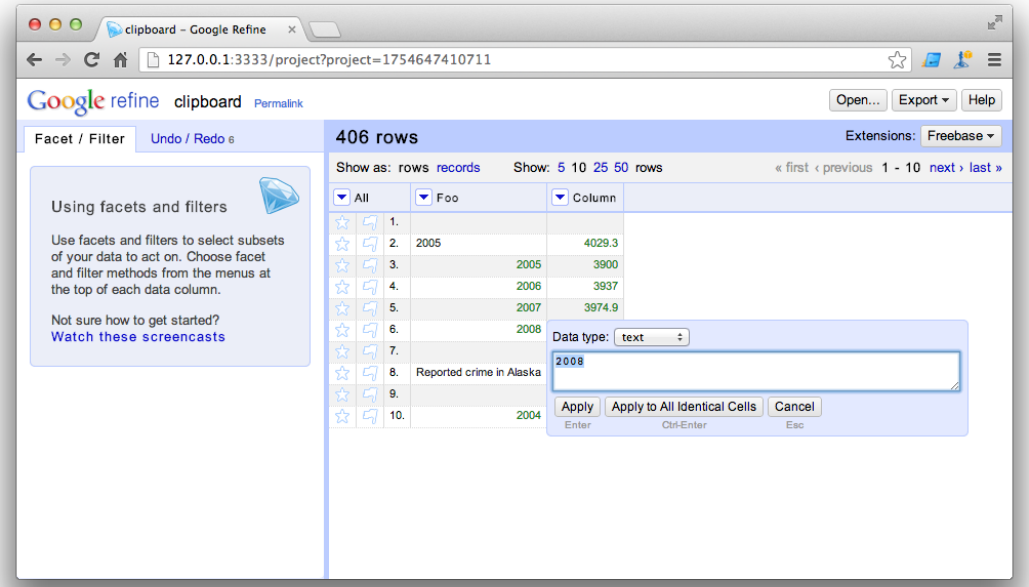


Figure 2.6: Google/OpenRefine UI in action, also offering direct manipulation of data. Again, as in DataWrangler, tabular respectively spreadsheet-like interface.

2.2.8 The Industry: Talend Open Studio

Within the industry a well-known collection of enterprise tools is *Talend Open Studio*[6]. This is an open source software suite with commercial support offerings. Architecturally, it is based on ***Eclipse*** Rich Client Platform (RCP). The product which can be used for data wrangling is *Talend Data Quality*[7] (see Figure 2.10 for its GUI). It allows for interactive specification and execution of data transformations with **visual charting aid**. I.e., data at hand is visualized via meaningful charts while interactively manipulating it.

2.2.9 An Innovative Approach: Kibana Timelion

Kibana is a product by the company *Elastic* which is mainly geared towards visualizing data of their other product *Elasticsearch* in an interactive fashion with charts. As sort of a research project *Timelion* was born (see [Ras15] and Figure 2.7). It allows for interactive exploration and transformation of time series data with a programmatic, mathematical DSL via a Command Line Interface (CLI). Commands trigger chart visualizations accordingly. The way these interaction methods are combined is rather unseen before.

[6] www.talendforge.org
[7] www.talend.com/products/data-quality/

Another Elastic product worth mentioning in our context is called *Logstash*[8]. This is basically a universal data processing engine with transformation capabilities, historically focused on log event data, mainly driven by filter expressions written in *Ruby*-like syntax.

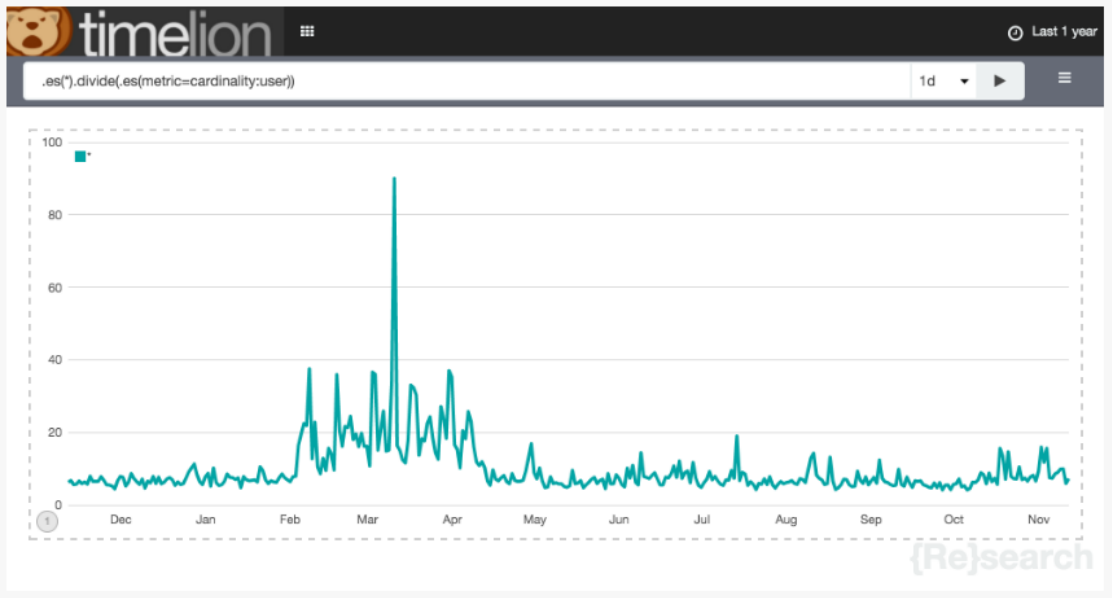


Figure 2.7: Kibana Timelion as an innovative approach to visual-interactively exploring and transforming time series data [Ras15].

2.2.10 Modern Data Science: Jupyter Notebooks

Data science nowadays is commonly pursued with *Jupyter Notebooks*[9].

These interactive notebooks are powered by a web-based application, often running locally and connecting to remote data sources. The technology originally emerged from *IPython*, an interactive CLI in Read–Eval–Print Loop (REPL) style. It since has moved to the web and supports other popular data science languages, like *R*, as well. So one can make use of statistical computations in a scripting manner, interactively creating according chart visualizations, potentially embedded in more standard textual sections.

2.3 Comparison and Summary

As shown by the presentation of related work, a diverse selection of approaches and tools can be found in this field. Yet, there are also quite some shared aspects to be recognized. For example, basic GUIs for visual-interactive schema mapping turned out to develop

[8] www.elastic.co/products/logstash
[9] jupyter.org

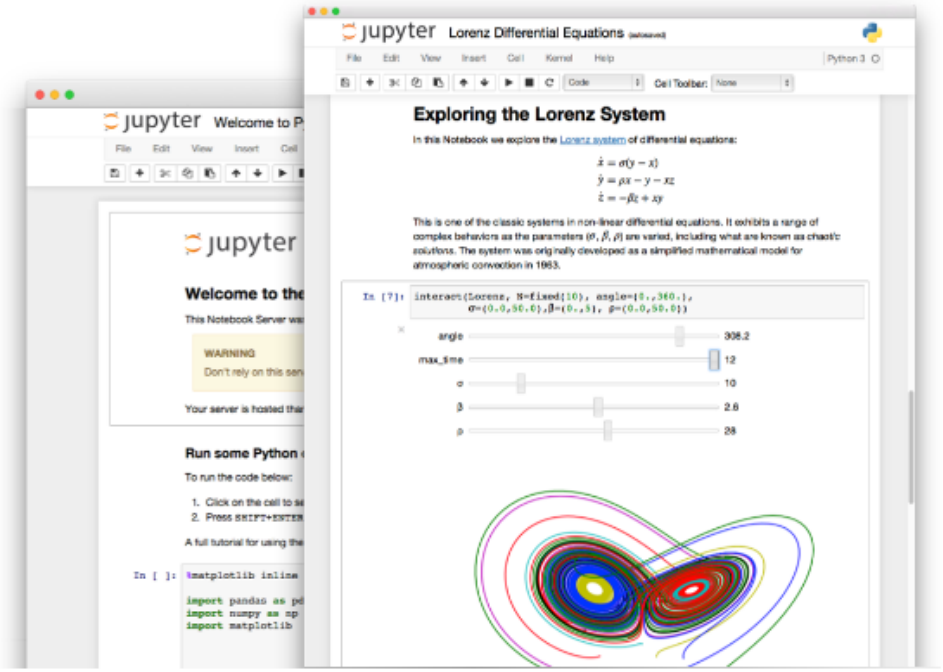


Figure 2.8: Data science with Jupyter Notebooks as advertised on their website.

into pretty similar directions (compare, e.g., the mapping views of *BizTalk* and *Clio* projects as presented in Figure 2.2 and 2.3).

Furthermore, the general UI for displaying datasets in this area is spreadsheet-like (see *Potter's Wheel*, *DataWrangler*, *OpenRefine*, and *Talend*). In addition to that, visual charting aids have started to be incorporated (particularly demonstrated by the more recent time series data related prototype and commercial tools by *Talend*). Moreover, modern and innovative approaches are combining classic *CLI* with scripting, and interactive charting in novel ways (see *Kibana Timelion* and *Jupyter Notebooks*).

Table 2.1 provides a **comparison outline** of these projects as well as a preliminary **requirements list** for our approach. It should be noted that "visual interactivity" is somewhat hard to measure therein, as some approaches lead into quite special directions.

According to these findings there seems to be a need for further research in visual-interactive aid, for example, via meaningful charts. Gaps to be filled here are connected to applying interactive InfoVis techniques in order to further improve support of data wrangling tasks. There are approaches already moving into this direction, yet, in particular interactive charting assistance has the potential to substantially facilitate wrangling tasks while still needing further research. As it turns out, visual-interactively supporting data wrangling tasks is currently still in its infancy. While the general topic of data wrangling is not new and quite some research and practice has been done in this field, combining it

with GUIs offering decent UX is relatively young. Especially, incorporating visualization of the to-be-transformed data and corresponding data transform operations themselves via meaningful charts is something where further R&D is required.

To this end, we are developing a research prototype, called ***TempMunger***, facilitating a visual presentation of the data that enables a better and faster understanding of the data structure and where there is a need for transformation as well as interactive charts for data manipulation and for giving immediate visual feedback of the transformation.

2.3.1 Our Approach: TempMunger

Where we are intending to excel with *TempMunger* here is by bringing concepts from both *DataWrangler* and *OpenRefine* together with enhanced visual charting aid as well as our derived requirements (see Section 3.1.2) and improved UX. The main kind of data to be wrangled in particular will be time-oriented. So, focus will be laid on visual-interactive support of wrangling time-oriented data.

More concrete, possible areas of improvement include:

- Introduction of modern interaction patterns, like drag & drop column merging
- Visualizing data structures via meaningful charts, presenting transform suggestions
- Directly manipulative interaction with these charts to transform underlying data
- Special focus on supporting time-oriented data with meaningful charts and interactive transformations

Further requirements are derived in Section 3.1.2 and we expect that more reasonable functionality, features, and constraints will emerge from the iterative *design – implementation – evaluation* process of the thesis project.

In addition, we are going to tackle the following data wrangling challenges as identified by [KHP+11]:

1. *Diagnosing data problems visually*
2. *Visualizing "raw" data*
3. *Visual assessment and specification of automated methods*
4. *Living with dirty data* (visually; i.e., how to display erroneous data best)

While these InfoVis topics are merely directed towards data profiling, we extend upon this by applying it to transformations connected to data wrangling.

To sum it up with our main research question, stated once again:

How can we support data wrangling with VA techniques?

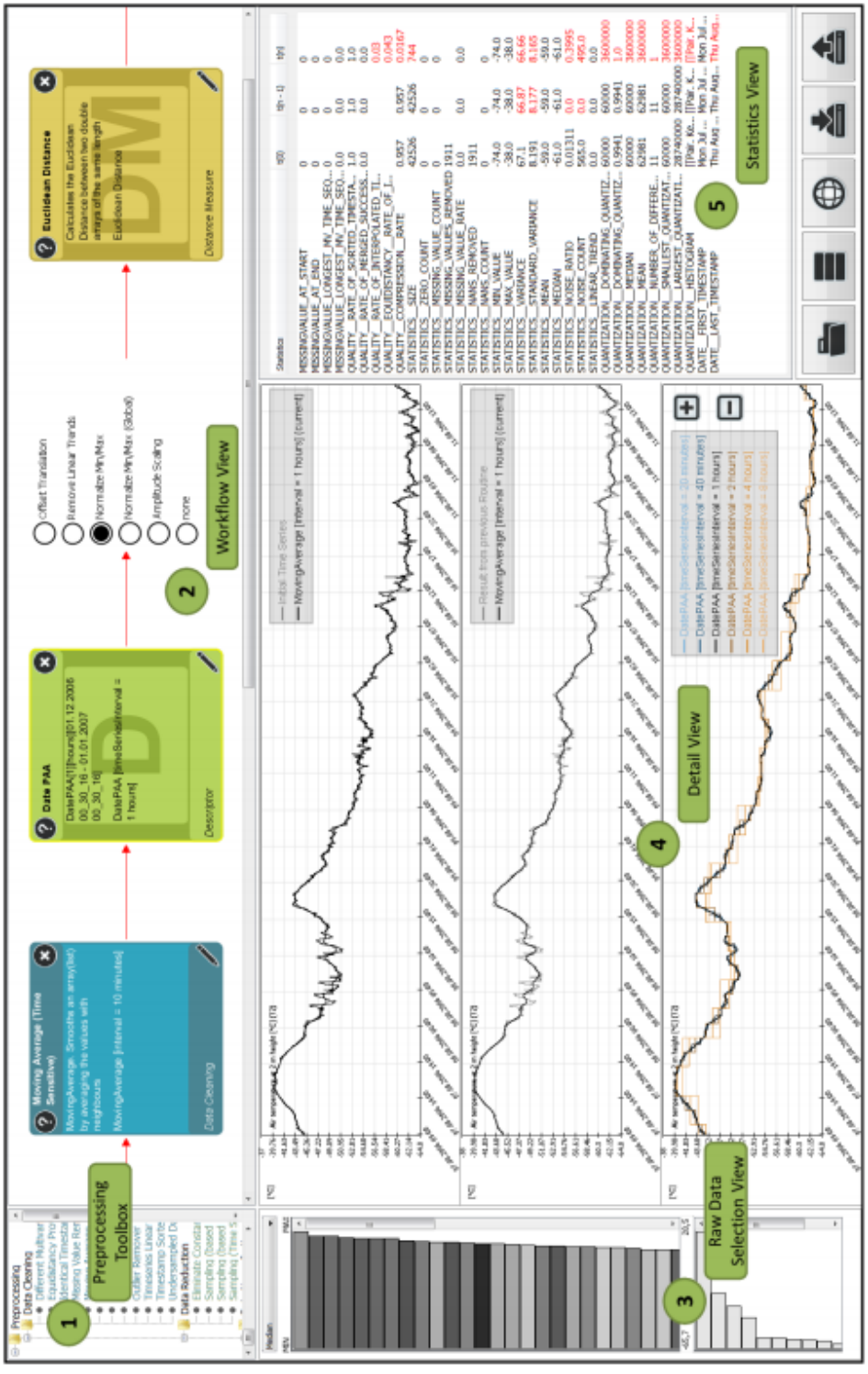


Figure 2.9: Time series research UI with interesting pipeline-based approach (top pane). Various charts (main pane) visualize the data at hand (right pane) [BRG+12].

Project	Platform	Domain	GUI Richness	Visual Interactivity	Charts	Time-Oriented	Dashboard
Potter's Wheel	C++, Swing	Generic	Medium	Medium	No	No	No
PADS	DSL Code, C	Specific	N/A	N/A	No	No	No
BizTalk	Windows .NET	Schema Mapping	Medium	Medium	No	No	No
Clio	Java Desktop	Schema Mapping	Medium	Medium	No	No	No
Potluck	Web	Semantic Web	Low	Medium	No	No	No
DataWrangler	Web	Generic	Medium	Special	No	No	No
OpenRefine	Web	Generic	Medium	High	Yes	Yes	No
Time Series	Unknown	Specific	High	High	Yes	Yes	Partially
Talend	Eclipse RCP	Generic	High	High	Yes	Unknown	Partially
Kibana Timelion	Web	Specific	High	Special	Yes	Yes	No
Jupyter Notebooks	Web	Generic	High	Special	Yes	Yes	No
TempMunger	**Desktop** (Web)	**Generic**	**High**	**High**	**Focus**	**Highly**	**Focus**

Table 2.1: Projects comparison serving as a starting point to derive basic requirements.

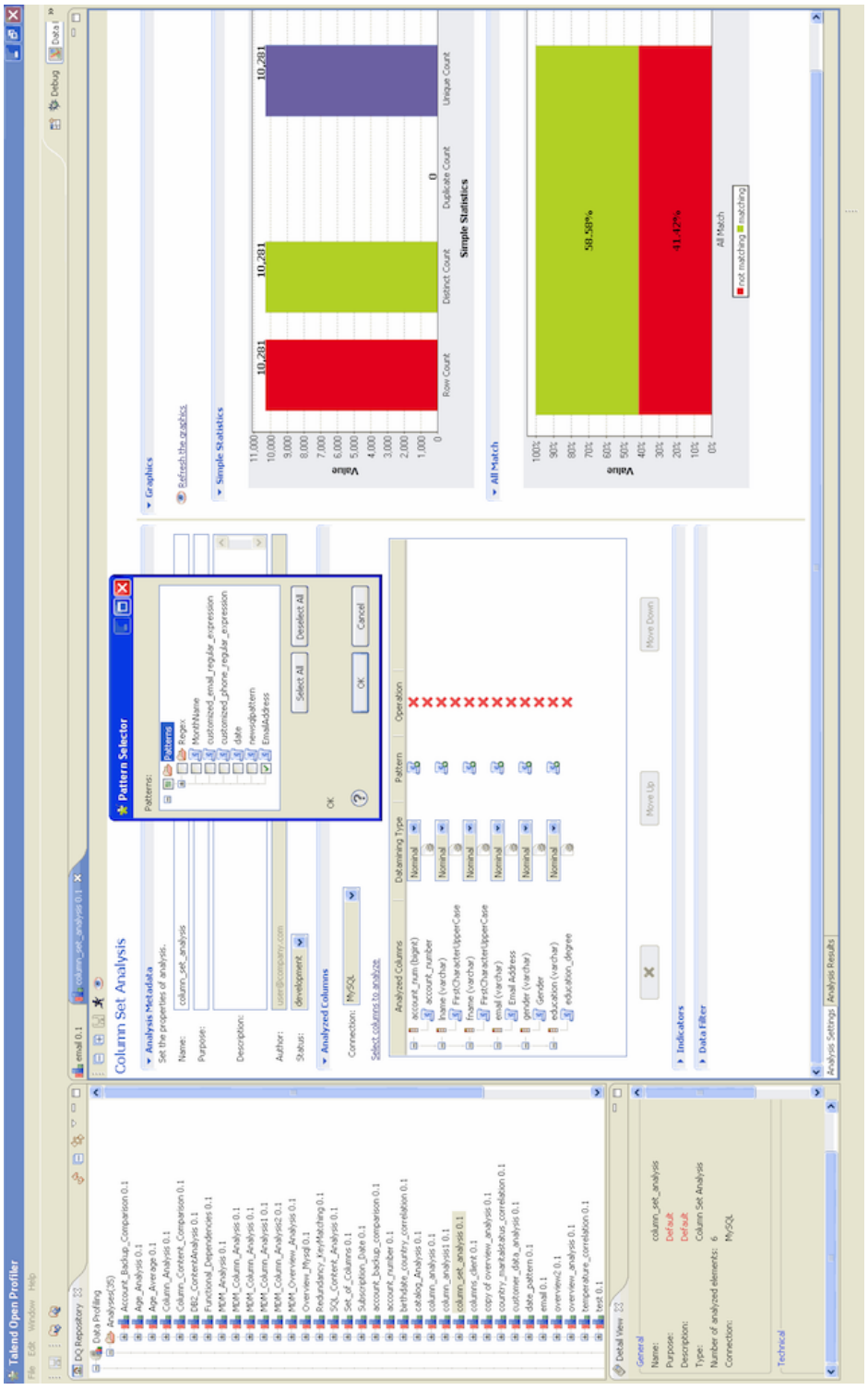


Figure 2.10: Talend Open Studio Data Quality GUI as an industry-standard solution based on Eclipse RCP. Interactively manipulated data (on the left) by transformations (center) is visualized via charting support (right). Screenshot originally taken from product website.

CHAPTER 3

Design and Implementation

3.1 Requirements Analysis

This is a vital part and first step of our methodology leading to a proposed solution.

3.1.1 UX Personas

In order to derive a meaningful list of design requirements we make use of an instrument from designing products called UX personas.

It has been pioneered for usage with software development by Cooper [Coo04]. The basic idea is to come up with some stereotypical "personalities" described by certain characteristics which represent our target user groups. An important aspect is the potential creation of empathy with our future users.

Each of these personas is, typically, illustrated with a profile picture and at least a firstname. The notion is to create some degree of familiarity and identifiability for the UX designer and other parties involved in the design process. Furthermore, usually, a persona is equipped with some demographical coordinates, some sort of tagline which serves as an executive summary, enhanced with background info, and motivations. All of this information is normally pointed and rather skimped. It should support in easily creating a vivid idea and image of the different users of the product in design.

Finally, scenarios or user stories briefly describe ways in which these particular user types would use the imaginary product. In the end, it is important the resulting personas can also be physically tangible – for instance, printed out on cards and pinned onto a whiteboard. Personas are a valuable tool for subsequent design of UI and interactions.

Table 3.1 provides a high-level overview of the personas we came up with for our prototypical software, focusing on skill set distribution. As one can see, persona skills range from overall highly to overall lowly skilled as well as individuals with focus on

Skills	Hugo	Alice	Bob	John	Jane	Walter
Technical	**low**	**high**	low	low	high	low
Scientific	medium	**high**	**low**	medium	high	high
Data-Related	medium	high	**medium**	medium	**high**	medium
Temporal Interest	medium	high	**low**	high	**high**	high

Table 3.1: UX personas skill summary and comparison. Edge entries in **bold**.

certain different skills. This makes for an interesting foundation as, all in all, quite a disperse set of potential users has to be catered for. The following pages contain our personas themselves. The profile pictures were created with an online avatar tool[1].

3.1.2 Requirements List

Through the creation of our personas we were able to properly visualize and dissect corresponding requirements for our prototype. Consequently, we have derived these:

- **R1:** The prototype must be capable of loading and working with diverse datasets
- **R2:** Moreover, it must be intuitive for casual users (i.e., less technically expertized)
- **R3:** Yet, some shortcuts for rather power users should be supported as well
- **R4:** Focus of our approach has to be on visual-interactive charting aid
- **R5:** These charts must be centered on applying time-oriented data transformations
- **R6:** Plus, they should provide extraordinary visual overview of the dataset at hand
- **R7:** Thus, focus has to be put on choosing most effective and efficient visualizations
- **R8:** Furthermore, interactively exploring data must be conveniently possible
- **R9:** A more traditional tabular editor should be available with direct manipulation
- **R10:** Editing time-oriented data should be supported by specific UI controls
- **R11:** Data quality issues need to be easily identifiable and effectually addressable
- **R12:** Conveniently spotting data anomalies respectively outliers should be possible
- **R13:** Concrete time-oriented data transformation operations to be supported:
  - Data cleaning regarding missing and erroneous values
  - Normalization concerning points in time and intervals
  - Merging columns in an intuitive visual-interactive way
  - Formatting cleanup, e.g., inconsistencies or conversion

[1] avachara.com/avatar/

3.1.3 Hugo

Demographics

- Age: 35
- Location: Vienna, Austria
- Job: Business Analyst
- Expertise: Marketing & Statistics

Tagline

"I need to quickly filter out erroneous data from market survey results."

Background

- Studied business administration focusing on marketing and specializing in statistics
- Some years of working experience in the industry
- Responsible for pointing out business opportunities through analyses

Motivations

- Wants to see the "big picture"
- Doesn't want to "lose" any time

Scenarios (User Stories)

- Got huge amounts of messy real-world data from various market surveys
- Wants to "scan" this data quickly for using it in market/business analyses
- Often data is time-oriented, as it denotes market-related developments over time

3.1.4 Alice

Demographics

- Age: 31
- Location: Vienna, Austria
- Job: Academic Researcher
- Expertise: Mathematics & Statistics

Tagline

"I'm interested in spending less time wrangling datasets suitable for analysis."

Background

- Studied mathematics with a focus on statistics resulting in a research position in the field (post-doctoral)
- Special focus of the research group is time-oriented data, being involved in various international projects

Motivations

- Wants to analyze huge datasets, often containing flawed data
- She would rather spend time on analysis than preparation

Scenarios (User Stories)

- Got various sample time-oriented datasets and wants to analyze the data
- Furthermore, wrangling should take less effort to apply Occam's razor

3.1.5 Bob

Demographics

- Age: 34
- Location: Graz, Austria
- Job: Journalist
- Expertise: Journalism & Politics

Tagline

"I would like to be able to handle messy data for analysis to be used in my articles."

Background

- Graduate of communication studies with a specialization in politics
- Worked for several online news agencies

Motivations

- Is held back from doing real "data journalism" due to lack of technical skills
- Would get into this kind of journalism if tools were better suited to his needs

Scenarios (User Stories)

- Got an idea for a current news story based on some quite untidy political/economic data which is often of time-oriented nature
- Is able to conveniently verify justification of story based on respective analysis of wrangled data

3.1.6 John

Demographics

- Age: 40
- Location: Salzburg, Austria
- Job: Political Analyst
- Expertise: Politics & Statistics

Tagline

"I want to conveniently and visually prepare vast amounts of public poll data for analysis."

Background

- Studied political sciences with a focus on statistics (Ph.D.)
- Works for news agencies, especially analyzing electoral situations

Motivations

- Strong need to be able to get lots of data from various polls into unified schema with as little hassle as possible
- Is not particularly technically skilled or interested, just wants to get the data to be able analyzing it

Scenarios (User Stories)

- Electoral poll data, consequently, mainly temporal natured, from various sources needs to get prepared respectively unified for analyzing
- Uses the visual-interactive tool being able to get the job done in a convenient way

3.1.7 Jane

Demographics

- Age: 37
- Location: Munich, Germany
- Job: Industrial Researcher
- Expertise: Biology & Statistics

Tagline

"I need a quick(er) and more reliable way to experiment with biological test data."

Background

- Graduated in bio engineering
- Works at a pharmaceutical company testing new ways of synthesizing cosmetics

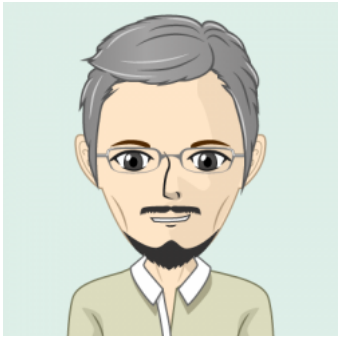
Motivations

- Currently, the whole roundtrip of setting up test labs and analyzing results is cumbersome and takes much time
- Wants to be able to iterate in a quicker mode of operation by improving on wrangling test data applicable for actual analysis

Scenarios (User Stories)

- Is able to reduce testing roundtrips by using the visual-interactive tool for making time series test data useful for analysis
- Based on experience and results from previous iterations she is able to decrease overall throughput time even more

3.1.8 Walter



Demographics

- Age: 46
- Location: Vienna, Austria
- Job: Medical Doctor
- Expertise: Diabetes

Tagline

"I want to be able to conveniently visualize temporal therapy data provided by patients."

Background

- Studied medicine, graduating cum laude
- Works at special center focusing on diabetics treatment

Motivations

- Often, therapy data provided by patients is rather messy, that is, concerning missing respectively erroneous values, formatting, ...
- Wants to be able to visualize data to get to see the "real" picture

Scenarios (User Stories)

- Using the tool he is able to reduce time spent on getting time-series-based therapy data provided by his patients ready for analysis and can focus on actually analyzing
- Might even encourage (at least some of) his patients to use the tool themselves to further reduce overhead

3.2 Design of UI and Interactions

For designing the UI and interactions we have created mockups a.k.a. wireframes [Gar11].

3.2.1 UI Mockups

The design of our prototype should meet our list of requirements. To this end, we created a number of mockups to be able to easily refine our designs.

We have created our UI mockups using *Balsamiq*[2] as productive tool. An important aspect of wireframing is that it should be convenient creating the mockups. One needs to be able to quickly iterate on ideas and throw away things which did not lead into the right direction. Often, simply pencil and paper are being used which is already a good way to get to some first scribbles. A quite common mistake is to skip proper wireframing and jump to design screens immediately. Most of the power within the creative design process and flow is lost this way as design screens take considerably more effort in producing them. Consequently, iterating on these is usually more sluggish and throwing results away rather avoided.

The following pages contain our resulting wireframed mockups including some descriptions and further explanations regarding their functionality and respective underlying reasoning. Mockups were created iteratively and evaluated in qualitative feedback loops until satisfying results have been achieved, that went into prototypical implementation.

Design Process

While iteratively designing with the help of mockups we constantly refined our ideas, adapting our approach, and trashing things that did not work out as expected.
Some material thereby discovered and/or changed:

- Foremost, pie charts are, mostly, not useful in our context of transforming time-oriented data
- On the other hand, bar charts are well suited to communicate quantities (e.g., of different table entries)
- Line charts, as commonly used for time series data, are not being emphasized on in our approach, mainly since we have found calendar heatmap visualizations to be superior for our use case, as our approach is generally not constrained to time series data but should be suitable for any time-oriented data
- Normalization functionality consciously left rather vague, to be fleshed out when actually developing the prototype
- Modal dialogs containing interactive charts make sense for certain actions

[2] balsamiq.com

- Visualizing the context of two different columns next to each other for exploratory comparison is beneficial
- A browser-based application is sufficient and a dedicated desktop one not needed in this case

Upload Dialog

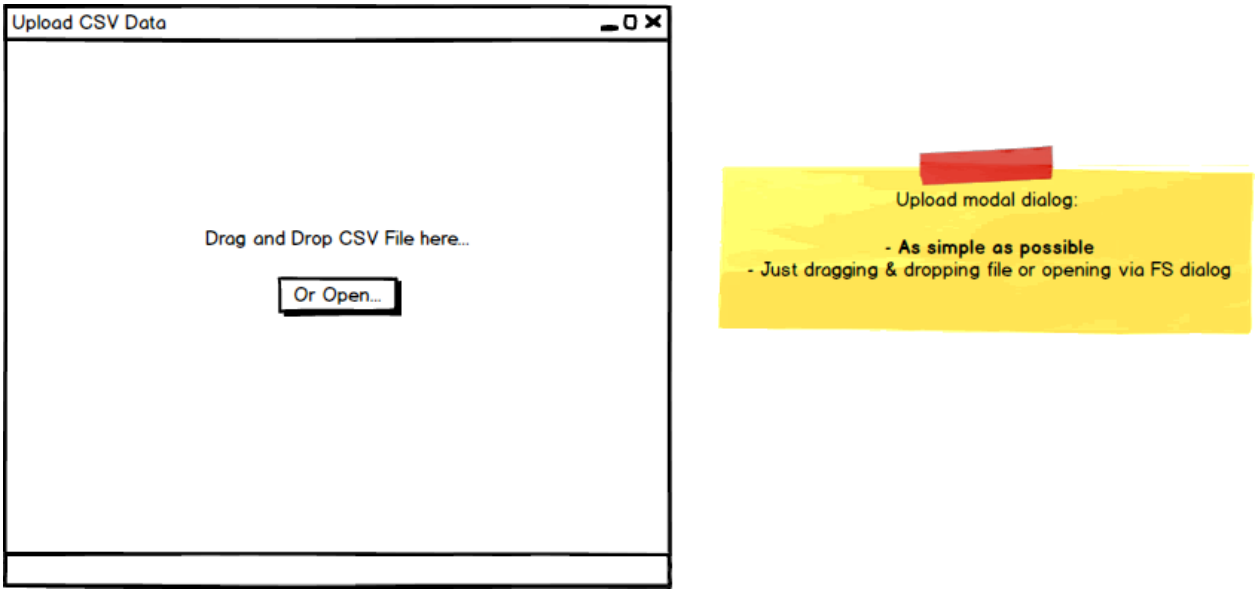


Figure 3.1: UI mockup of the upload dialog.

Naturally, the first UI component we have designed is the one which feeds the application with data to operate on: the upload dialog (see Figure 3.1).

- **Description**
  - The main goal here was simplicity
  - Consequently, truly simple dialog
  - An area for dropping off file
- **Reasoning**
  - As it is central to the application, uploading has to be really simple
  - Thus, with as little effort as possible
  - That is, affordance has to be intuitive

The idea regarding interaction is that as soon as a file is selected, the upload commences automatically, giving visual feedback of its progress to the user via according animation effects, disappearing as soon as it is finished.

Table Editor

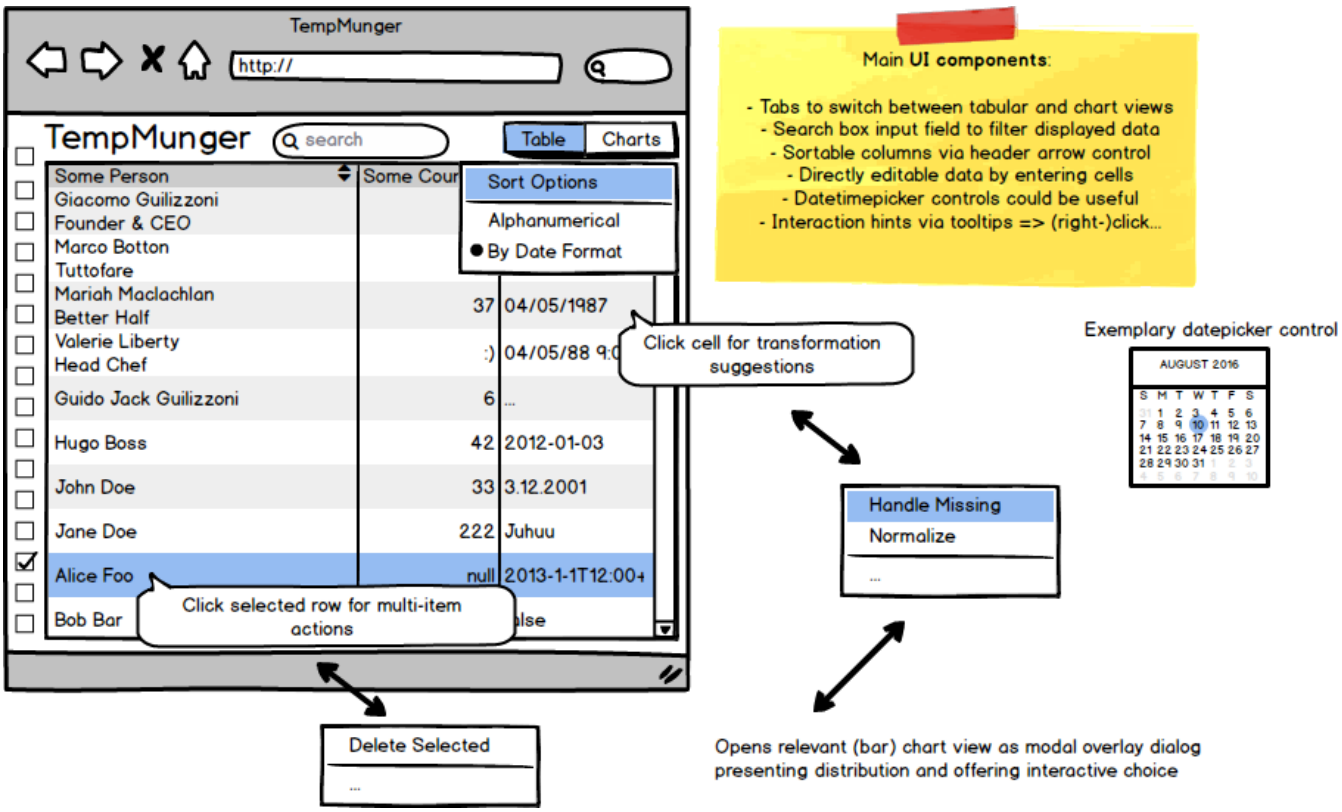


Figure 3.2: UI mockup of the table editor.

The table editor page (see Figure 3.2) has been designed to be one of the two main pages of the application, the charts page being the second one, intuitively accessible via tabbed navigation in the upper right of the screen.

- **Description**
  - Enables direct manipulation editing of data
  - Supports multi-row actions via check box selection
  - Various search, sorting, and filtering options are available
  - Provides menu access to further and more specialized dialogs
- **Reasoning**
  - The table editor is a well-known UI metaphor for this use case
  - Many users are familiar with editing tabular data from MS Excel & co.
  - It supplies the user with a straightforward and efficient mode of interaction

Missing Values Dialog

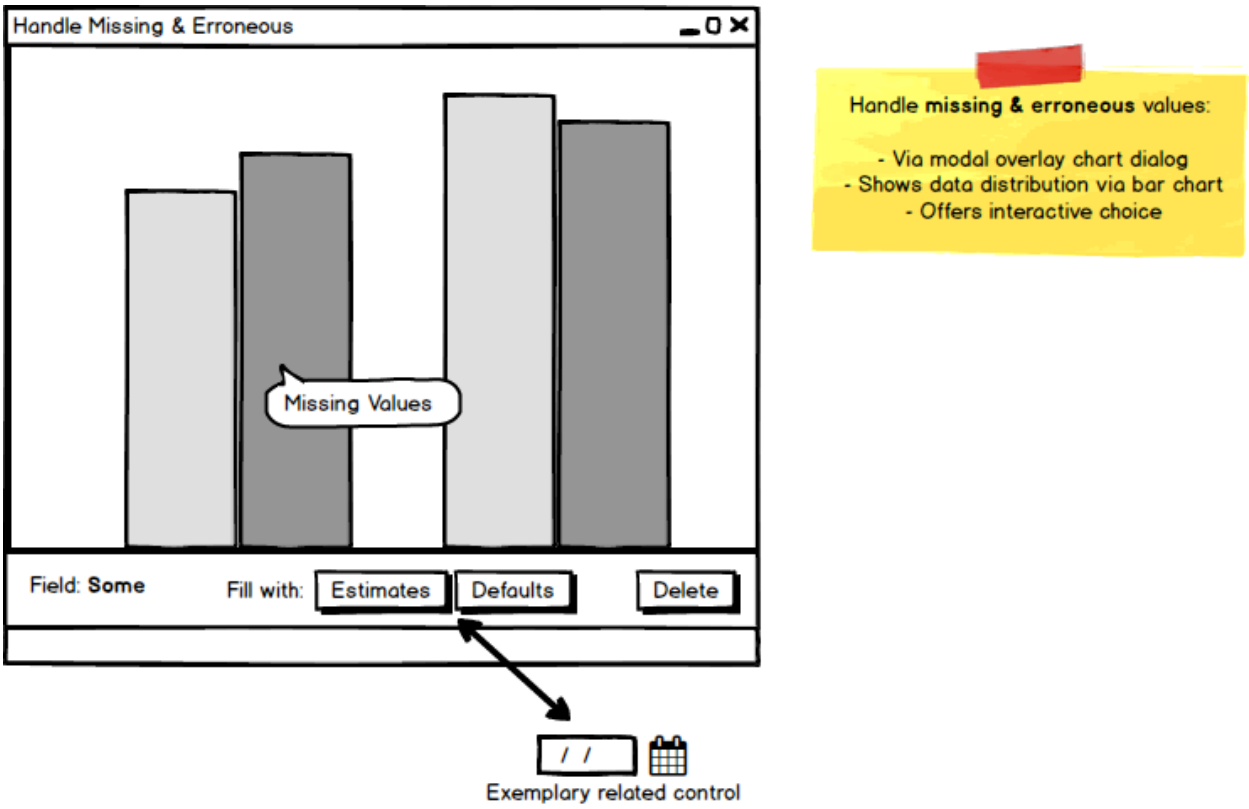


Figure 3.3: UI mockup of the missing values dialog.

The missing values dialog (see Figure 3.3) has been designed to be opened from the table editor page via according menu access.

- **Description**
  - Concrete shape of chart not 100% clear at this point
  - Most probably, bar chart – possibly, a horizontal one
  - User is able to fill missing value entries or delete them
  - Options provided are filling them with estimates or defaults
- **Reasoning**
  - Bar charts are capable of communicating distributions well
  - Another possibility would be pie charts, but they are proven to be misleading
  - To quote Tufte [Tuf01], p. 178: *"Given their low density and failure to order numbers along a visual dimension,* ***pie charts should never be used.***"[3]

[3] Cf. www.edwardtufte.com/bboard/q-and-a-fetch-msg?msg_id=00018S

Normalization Dialog

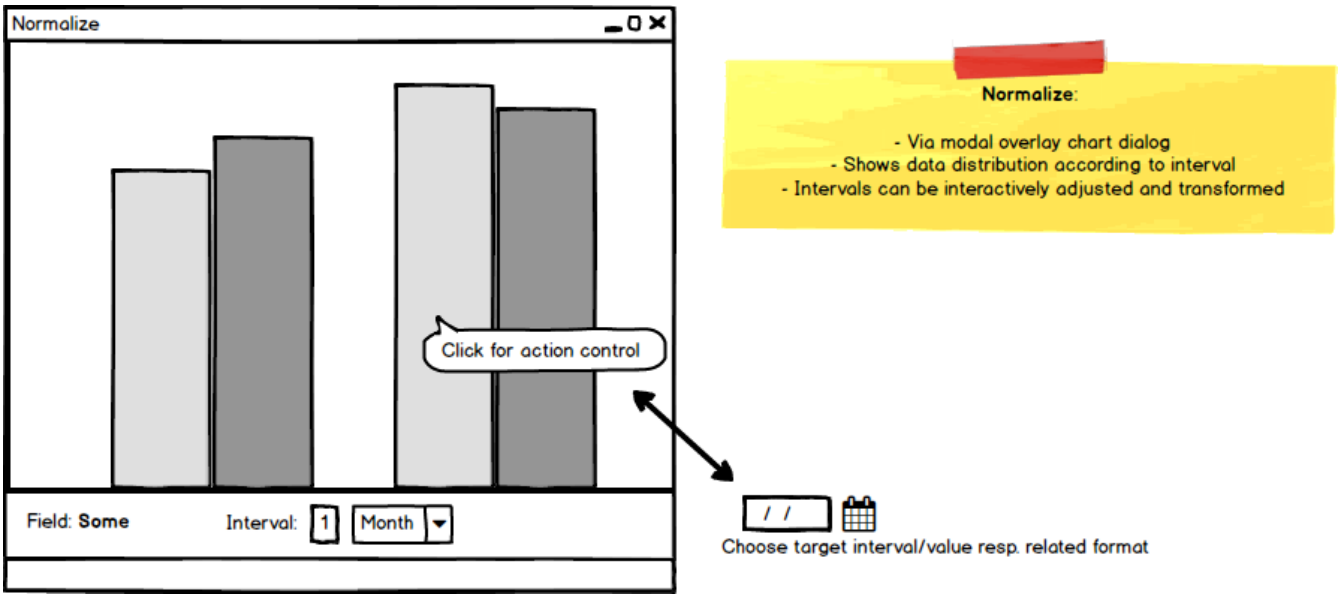


Figure 3.4: UI mockup of the normalization dialog.

The interval-focused normalization dialog (see Figure 3.4) is meant to be accessible via menu from the table editor page, too.

- **Description**
  - Its purpose is supposed to be enabling a user to normalize temporal intervals
  - For batch-wise transforming values of entries within a certain timespan
  - Chart visualization is rather unclear at this stage
  - Probably, bar chart as well – but rather vertical one
  - Interaction via point and click, including with chart
- **Reasoning**
  - The use case being worth covering became evident while gathering requirements
  - Consequently, we will experiment with supporting it via meaningful interactive chart visualization
  - Concrete characteristics will be shaped while iterative development itself
  - Most probably, our color scheme of the chart visualization will be within neutral, plain gray to black range
  - Since explicit coloring should only be used when it can communicate and, hence, convey meaning to the user, being intention-revealing, that is
  - So, in the case of Figure 3.3 it may make sense to use a noticeable color

Outlier Detection Alerting

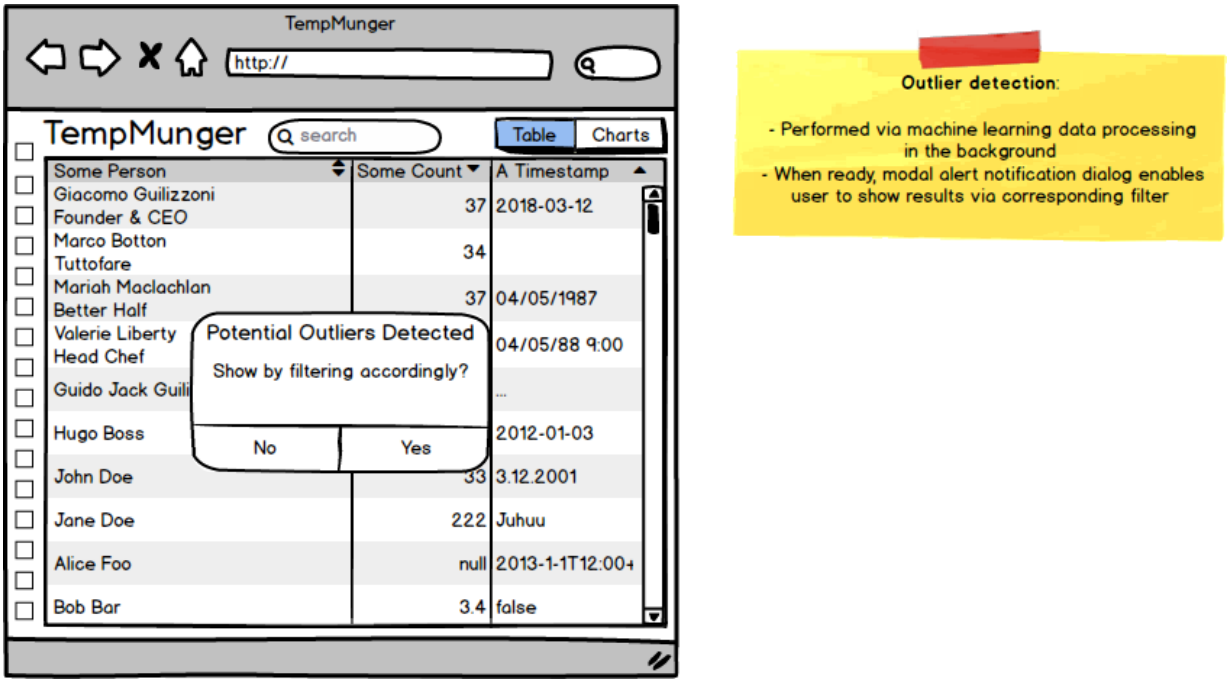


Figure 3.5: UI mockup of the outlier detection info alert.

When outliers are detected we need to show that to the user (see Figure 3.5). As we intend to apply machine learning for this purpose, we need some way to do so without sacrificing good UX.

- **Description**
  - Therefore, a simple modal dialog overlay is chosen
  - It offers the user to display detected potential outliers
  - When the user accepts, views are filtered accordingly
- **Reasoning**
  - Corresponding ML processing has to happen in the background
  - This is due to its potentially longer lasting computation time
  - Consequently, informing the user about results has to be as unobtrusive as possible without interruption
  - So, it should definitely not interfere with the current workflow, goals, and tasks of the user
  - Thus, we intend to make use of an overlay which does not block the UI and stays around for later use, more like an interactive notification-style message

Table Column Merging

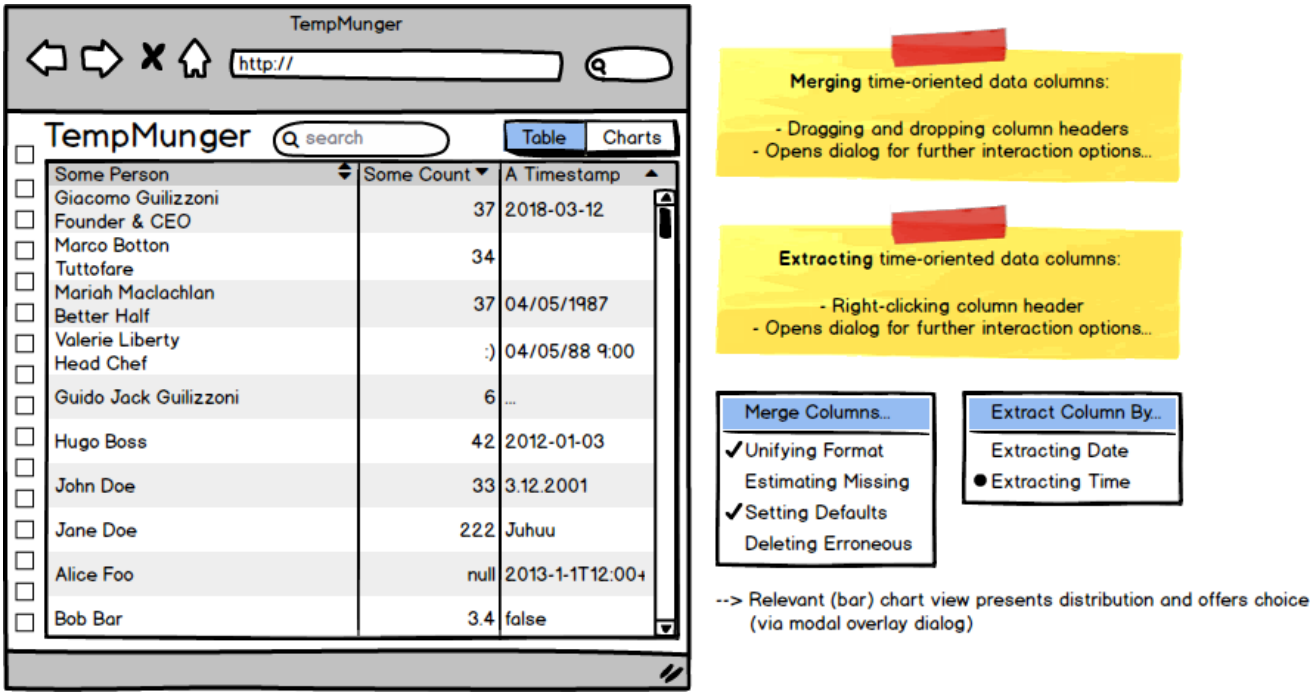


Figure 3.6: UI mockup of merging table columns via drag & drop.

An interesting idea is to enable merging time-oriented data columns via drag & drop interaction metaphor (see Figure 3.6).

- **Description**
  - So when dragging a temporal column unto another, a related merging operation should be initiated
  - The initial idea is to offer optional choice regarding the merge via a menu then
  - Options like what to do with missing values and how to merge values in general
  - Another idea is enabling column extraction via drag & drop as well, still somewhat vague, though
- **Reasoning**
  - Many users are familiar with the basic kind of this interaction from spreadsheet applications like Excel
  - Consequently, when indicating via according cursor on hover it is likely users will give it a spin
  - Corresponding coloring of drop targets while dragging would be helpful to support the user with the interaction

Charts Page

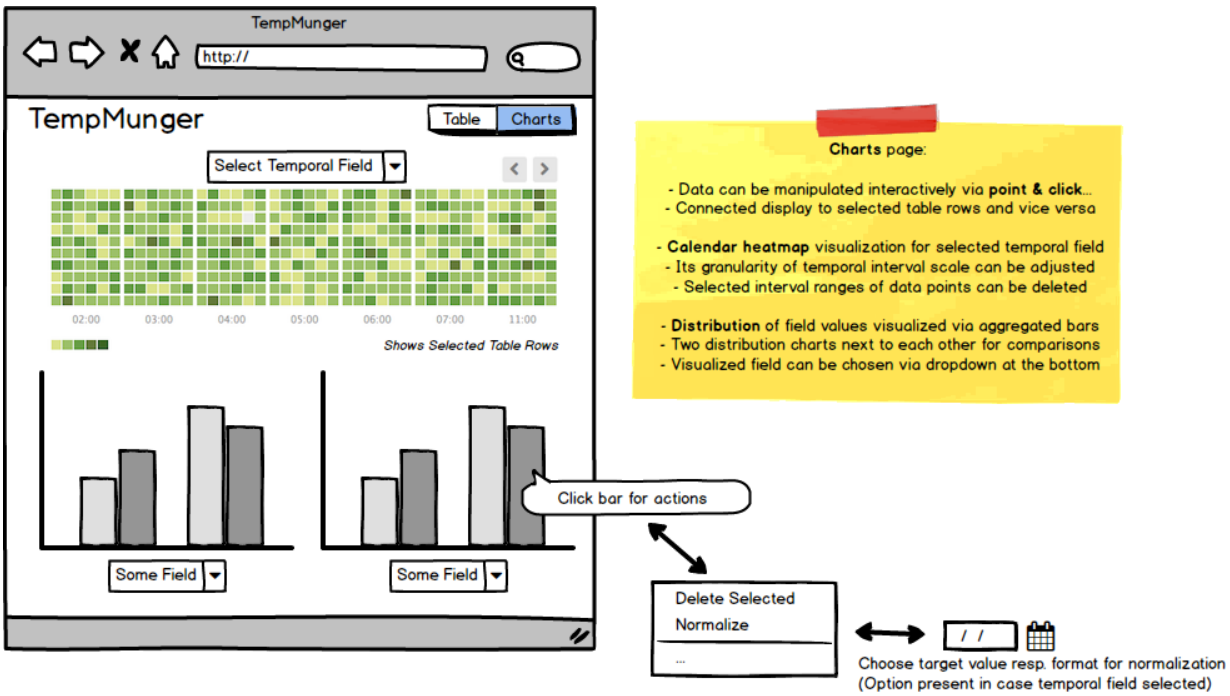


Figure 3.7: UI mockup of the charts page including calendar heatmap visualization.

The charts page is the second of the two main pages of the application, next to the initial table editor one.

- **Description**
  - It is headed by an interactive calendar heatmap visualization
  - Below, two distribution bar charts are next to each other
  - Controls allow interacting with the charts, plus their items should be interactive
  - Table editor filtering is intended to be interconnected with charts page views
- **Reasoning**
  - Calendar heatmap visualizations are particularly useful for displaying time-oriented data distributions
  - Densities of data therein are usually visualized via appropriate color scheming, popularly ranging in the green spectrum
  - Histogram-like bar charts are useful for viewing data distributions in general
  - Having two of the latter next to each other is great for comparisons, interactive exploration, and discovery

3.3 Iterative Prototyping

Following the creation of our UI mockups and agreeing that a satisfiable state had been reached, we started with implementing the corresponding prototypical software.

So, we developed in an agile manner, meaning close contact and collaboration with the "client", in this case the assisting thesis advisor. Plus, developing respective parts of the application iteratively, chunk by chunk, preferably with short iteration cycles. While developing new features there were also regular short phases in between, where focus was laid on bug fixing, cleanup, and polishing of existing things.

Therefore, we have set up a live testing environment, easily accessible for the client, regularly shipping changes, and gathering feedback to be incorporated as promptly as possible. Technical details regarding the setup are described in Appendix A.

As workflows in this project were particularly lean and lightweight, no special issue management software was used. It generally sufficed to make use of simple tools like *Wunderlist*[4], *Simplenote*[5], and email communication for tracking, planning as well as discussing todos, tasks, and issues. Additionally, from time to time when felt necessary and considered potentially fruitful, personal meetings were held. Mainly for hands-on demoing and reviewing purposes, plus, talking about direction-giving decisions.

This process was followed until the prototype eventually reached feature-completeness.

3.4 TempMunger

This section goes into some details regarding the implemented prototype itself. Extensive documentation of related software design and architecture can be found in Appendix A.

3.4.1 Implementation Details

As Integrated Development Environment (IDE), *IntelliJ IDEA*[6] was used.
For conveniently reloading compiled code on the backend without requiring server restarts, *JRebel*[7] was employed. On the frontend, a technique called *Hot Module Replacement (HMR)* is fulfilling similar tasks. *Redux DevTools*[8] is a useful Google Chrome browser extension when developing Redux/React apps, and *PageSpeed*[9] for adhering to website performance best practices. Cross-browser development as well as responsiveness for mobile devices were not part of the thesis prototype. Though, at least basic support may be present due to libraries used. So the application is primarily optimized and tested to run in a Google Chrome **desktop** browser.

[4] www.wunderlist.com
[5] simplenote.com
[6] www.jetbrains.com/idea/
[7] zeroturnaround.com/software/jrebel/
[8] extension.remotedev.io
[9] developers.google.com/speed/pagespeed/

The source code might be made public as open source software at some point in time, most likely on GitHub.

Elasticsearch Aggregations

Foundational background regarding IR and the search engine technology used for our prototype can be found in Appendix B. Its software architecture is covered in Appendix A. Aggregations are a powerful way in which *Elasticsearch* supports real-time analytics. They are used extensively throughout our application. The general idea is to aggregate occurrences of certain values in buckets with corresponding counts. Most of the charts implemented in our solution rely heavily on these. Moreover, Elasticsearch aggregations can be nested which renders lots of analytical variety possible. Thus, we are storing field values non-analyzed for aggregation as well as analyzed for full-text search purposes.

Our Data Model/Storage

The basic data model is a rather schema-less one. So, Elasticsearch is enabled to figure out data types automatically on first indexing of respective data when uploading a dataset to the application. Uploading data issues wiping the index before storing it. Generally, there are two data types made available to our solution, either text or temporal. For recognizing temporal data, various related formats are specified for parsing attempts. Our data model is also quite lenient when it comes to values which fail parsing, simply ignoring the failure and storing the value at hand anyway. This way missing or erroneous values can be treated separately. See Appendix C for a list of supported formats. Temporal values are uniformly stored in our Elasticsearch index in *Coordinated Universal Time (UTC)* timezone respectively *Greenwich Mean Time (GMT)*. When load as local date/time values on the frontend these are converted making use of timezone offset calculations.

On Spark RDDs

As explained in Appendix A, *Apache Spark* is operating on Resilient Distributed Dataset (RDD)s. In our prototype, these are being filled with data by querying Elasticsearch. Extensive caching and pre-loading of data is applied to boost performance. More concretely, for instance, the use case can be to transform all dataset entries within a certain timespan for a specific temporal field to a specified other temporal value.

Via Elasticsearch Bridge

This is being accomplished via an Elasticsearch/Spark bridge, as described in Section A.0.2. Thus, a Spark context can be configured to connect to an Elasticsearch cluster. In the end, one can transparently operate on RDDs with Spark's functional programming model[10].

[10] spark.apache.org/docs/latest/programming-guide.html#transformations

With Seamless Interop

The interop is, all in all, quite seamless. Especially also concerning Kotlin code calling the ES-Hadoop connector Java API as well as Spark's underlying Scala one when necessary. Loading data from and writing it back to Elasticsearch is mostly transparent.

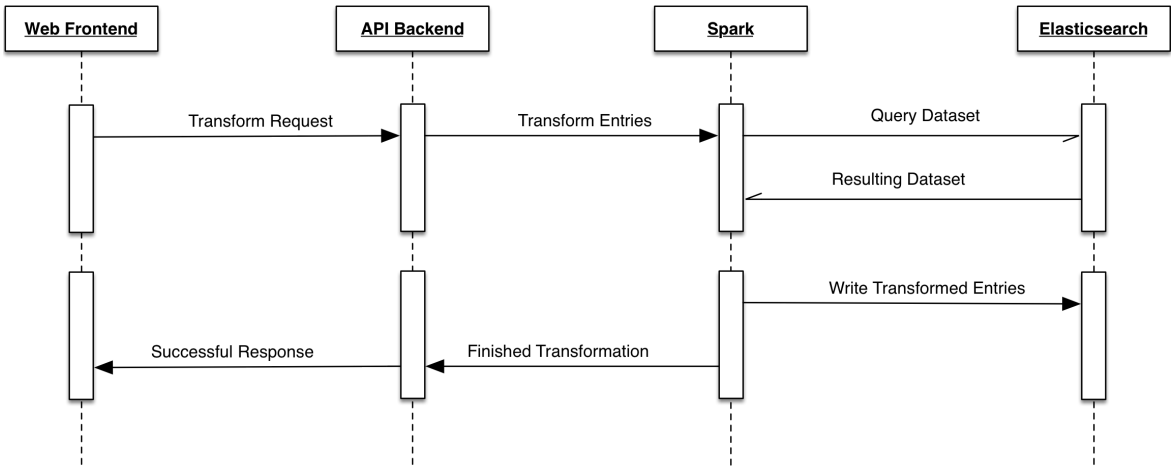


Figure 3.8: Sequence diagram showing the general data transformation flow.

3.4.2 Features of TempMunger

Our prototype possesses the following main, high-level features regarding time-oriented data, primarily focusing on visual-interactive, and particularly charting support:

- **Transformations**
  - **Direct manipulation** via UI controls
  - **Cleaning** of missing and erroneous values
  - **Normalization** concerning:
    - Points in time
    - Intervals
  - **Deletion** of rows
  - **Merging** of columns
  - **Formatting** cleanup
- **Outlier detection**
- **Visual overview**

Furthermore, a more traditional **tabular editor** is available as known from spreadsheet applications like, most prominently, Microsoft Excel. Users are used to the underlying interaction metaphor and, consequently, it makes sense as a foundation to build upon.

3.4.3 Transformations

A central part of the approach is transformation of time-oriented data. Generally, this is being achieved by making use of Apache Spark processing of Elasticsearch data. As mentioned above, transformation operations include **cleaning**, **normalization**, and **merging**. Figure 3.8 is presenting the general underlying flow via a sequence diagram.

3.4.4 Outlier Detection

Our prototype applies some ML techniques for its temporal outlier detection component.

K-Means Clustering

This is a popular algorithm of *unsupervised learning*, i.e., ML which does not rely on manual classification input, but rather classifies recognized patterns autonomously.

Formula 3.1 represents its core principle, partitioning real vectorized observations x into k class cluster sets S by calculating mean distances to respective centers (μ being the mean of points in S_i), generally computationally applying statistics to pattern recognition:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{3.1}$$

The following explains how this can be used for anomaly respectively outlier detection.

Outlier Detection Usage

The algorithm applied for our outlier detection component, basically, works as depicted in Algorithm 3.1.

A peculiar detail of our approach is that there is no dedicated test set of "new" data. This is due to the fact there is only one dataset available at a time with no additional data coming in to extend it. Thus, after training on a randomly split set, the whole dataset is used as test set, in the end, leading to overall satisfactory results. Moreover, we are limiting the number of classes to be clustered to two. Hence, our simple heuristic for determining an outlier class is to take the one of the two with fewer members. When there is only one class, it is assumed no outliers could be detected.

The Temporal Dimension

Our use case revolves around finding outliers in time-oriented data. Figure 3.9 shows the basic, related flow with a sequence diagram. In principle, we are using all time-oriented data values available in the dataset at hand for vectorizing the observations to be input to clustering. Therefore, the corresponding epoch millisecond values are used and if a certain value cannot be parsed it is substituted with a max. large number. Consequently, missing and erroneous values are likely to be subsequently tagged as outliers as well.

Algorithm 3.1: Temporal Outlier Detection

Input: A set of temporal field names φ and a corresponding RDD (dataset) δ
Output: An RDD π consisting of pairs of document ID to cluster class value

```
1  Vectorize dataset δ using field values via φ, see conditional (ll. 2-6);
2  if a field value can be parsed as temporal then
3  |  Use its epoch milli value;
4  else
5  |  Use max. large number;
6  end
7  Get training set τ from dataset δ via random split of 0.9 : 0.1;
8  Create predictive model μ from vectors x⃗ of training set τ;
9  → k-means clustering yielding 2 classes in 20 iterations and 3 parallel runs;
10 Predict points of dataset δ via model μ, resulting in RDD π, see loop (ll. 11-14);
11 for each point ∈ dataset δ do
12 |  Predict cluster class via model μ;
13 |  Add result to RDD π via map op;
14 end
15 return RDD π;
```

To further describe key points of the algorithm:
First, the dataset at hand is vectorized in order to enable applying it for clustering. Then, a training set is generated from it via random split. After that, a predictive model is created from the training set. Afterwards, the dataset at hand is predictively clustered via model. Finally, potential outliers are determined via aforementioned, simple heuristic.

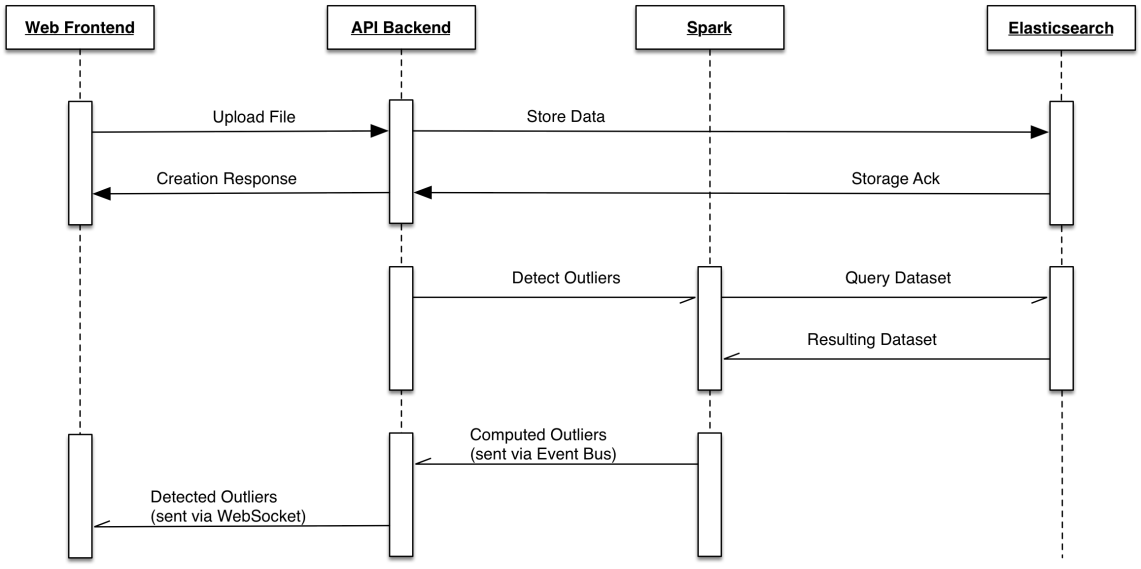


Figure 3.9: Sequence diagram showing the general outlier detection on upload flow.

3.4.5 Workflows and Screens

In this section, workflows plus related screens of the implemented prototype are described and explained. Moreover, special emphasis is laid upon the reasoning behind the chosen path of the solution.

Upload and Outlier Detection

Initially, the user will want to upload some dataset to the application. Therefore, a modal upload dialog is pretty conveniently reachable in the upper right corner of the screen. This button is also visually especially noticeable via its peculiar coloring. The upload dialog itself is designed as simple as possible (see Figure 3.10). One can either simply drag & drop a file to it or select one via File System (FS) dialog. As soon as a file is selected, the upload begins and is indicating its progress via related animations. In general, whenever data is being fetched respectively backend requests are issued, a spinning wheel effect is shown in the upper left corner of the screen. When uploading is finished the dialog disappears and the user is free to interact with the data.

As presented more from its technical side before, when uploading, an outlier detection mechanism is triggered. On finished processing, the user is shown an according notification. This message is sent as desktop browser respectively system notification[11] (see Figure 3.11) as well as within the application itself as some sort of flashing notification from the bottom of the screen (see Figure 3.20). The former stay around while the latter disappear.

When clicking a notification, an action is triggered filtering all dataset entries down to the potentially outlying ones. The user may then proceed to act upon accordingly, having the potential outliers ready in sight and at her/his fingertips. Presence of this filter is indicated via a chip-like control at the top of the screen (see Figure 3.21). It can be removed simply by clicking.

Table Editor

A central component of the UI is the table editor view page (see Figure 3.20).

There the user can directly manipulate the data at hand via editing corresponding cells. Pagination controls at the bottom of the page allow for convenient paging through the data. Page size can be adjusted as well as a particular page selected via related dropdowns. Multi-row deletion is supported via selection checkboxes at the left side of the table. It is possible to select rows one-by-one or (de)select all at once. The connected deletion button is located at the bottom left of the table.

Date/Time Picker

Date and time picker UI controls are used whenever an editable input field contains time-oriented data.

[11] developer.mozilla.org/en-US/docs/Web/API/Notifications_API

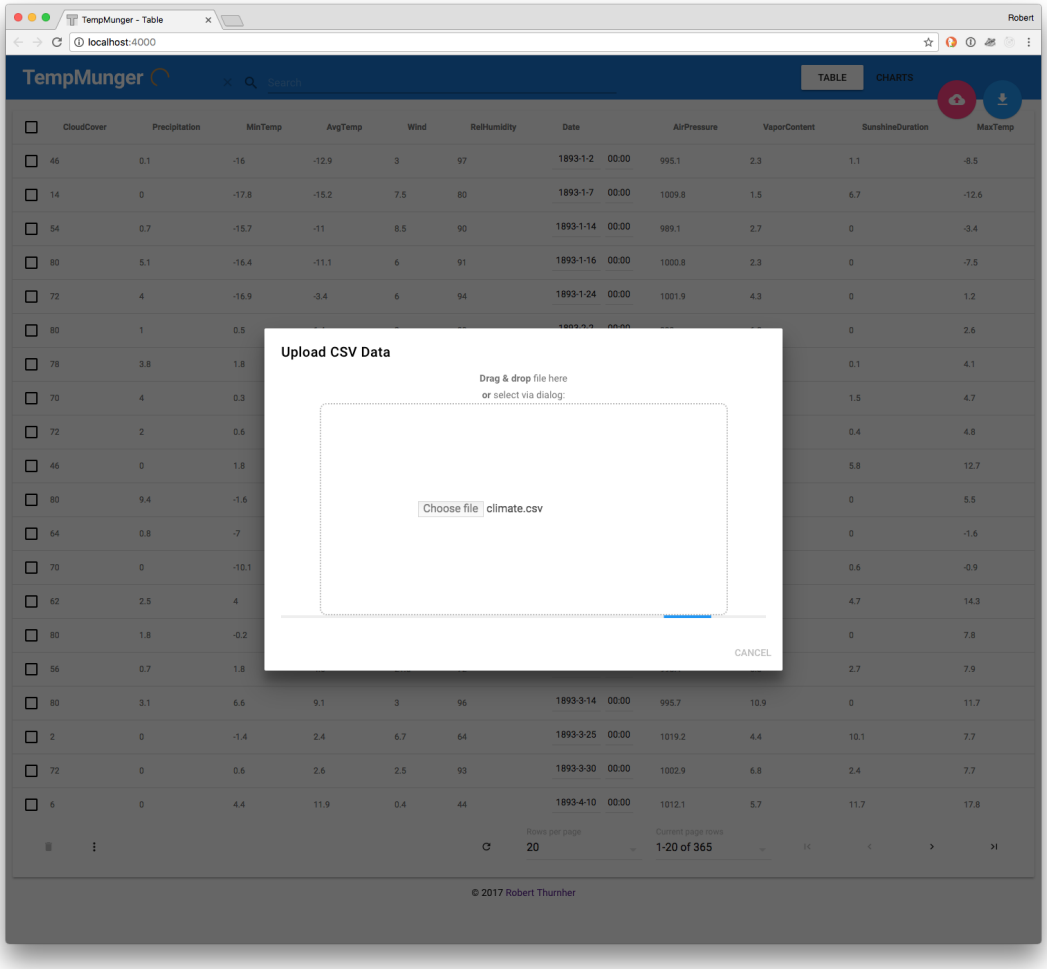


Figure 3.10: Screenshot showing upload with corresponding modal dialog and animated effects regarding progress indication.

The date picker allows the user to choose a date in an interactive way with the metaphor of a more traditional calendar (see Figure 3.12). Whereas the time picker uses the metaphor of an analog clock for choosing a time value (see Figure 3.13).

Both of these controls are commonly used throughout the application and normally in cooperation. For instance, the table editor employs the controls for editing temporal column row values.

Search and Filtering

There are various ways in which filtering, slicing, and dicing the dataset is supported:

Figure 3.11: Screenshot showing desktop browser respectively system notification for interactive suggestive outlier detection indication.

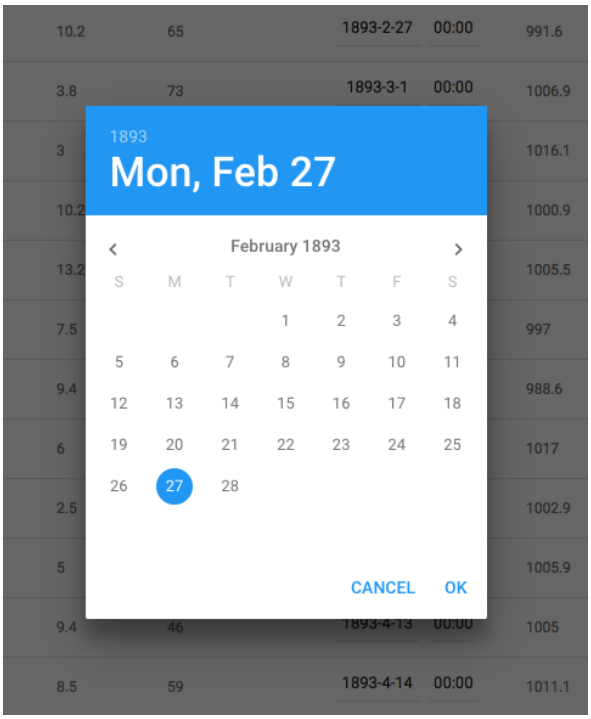


Figure 3.12: Screenshot showing exemplary modal date picker control with its calendar interaction metaphor.

- First of all, a search box is placed quite prominently at the top of the screen. This enables the user to filter data down via full-text search.
- Additionally, as mentioned above, multiple rows can be selected. Filter presence is indicated via aforementioned chip-like control. The filter affects displayed data in the charts view page as well.
- Rows can be sorted by column values in ascending or descending order. For this a simple click on the respective column header suffices.
- Furthermore, on the table view page there is a pagination available, see above.

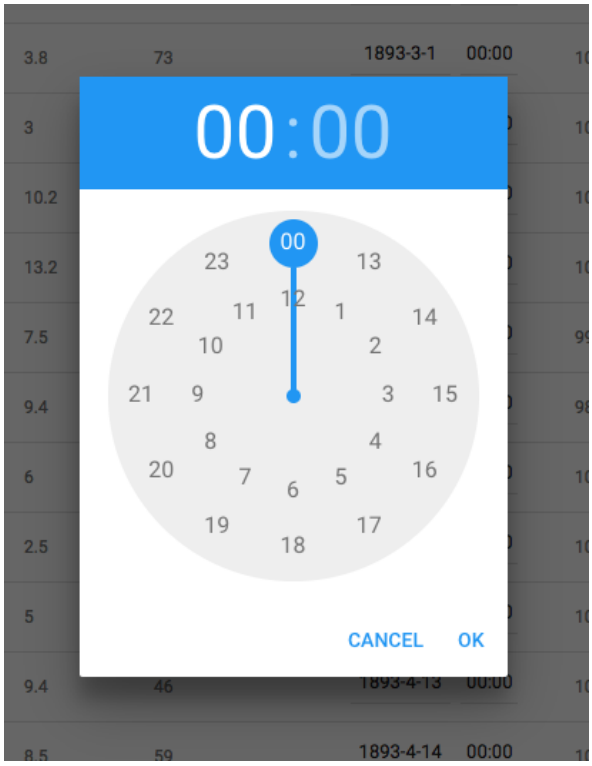


Figure 3.13: Screenshot showing exemplary modal time picker control with its clock interaction metaphor.

All of these filtering options are working together correspondingly (see Figure 3.21).

Missing Values Cleanup

It is possible to clean up missing and/or erroneous values via a modal dialog overlay (see Figure 3.14). Missing respectively erroneous in this context means that the values were not able to be parsed as temporal. The dialog can be accessed from the table editor view page at the bottom left via menu. Its menu option is only available when the currently loaded dataset contains at least one time-oriented data field.

On the modal dialog there is a dropdown to select one of the available temporal fields. Below it there is a horizontal multi-bar chart consisting of two bars. One of them represents all rows, and the other, missing values. The former are colored in a neutral gray, while the latter are colored orange. Consequently, a visual emphasis on the missing values is established. The bars can be displayed grouped or stacked to get a better feel of the quantities at hand. Below the chart there is a date and a time picker control for choosing a target value to fill the missing values with. It is originally set to the average value of all values of the respective field (excluding missing ones). At the bottom of the dialog there are action buttons to either apply the fill operation as described or to delete

all rows with missing values.

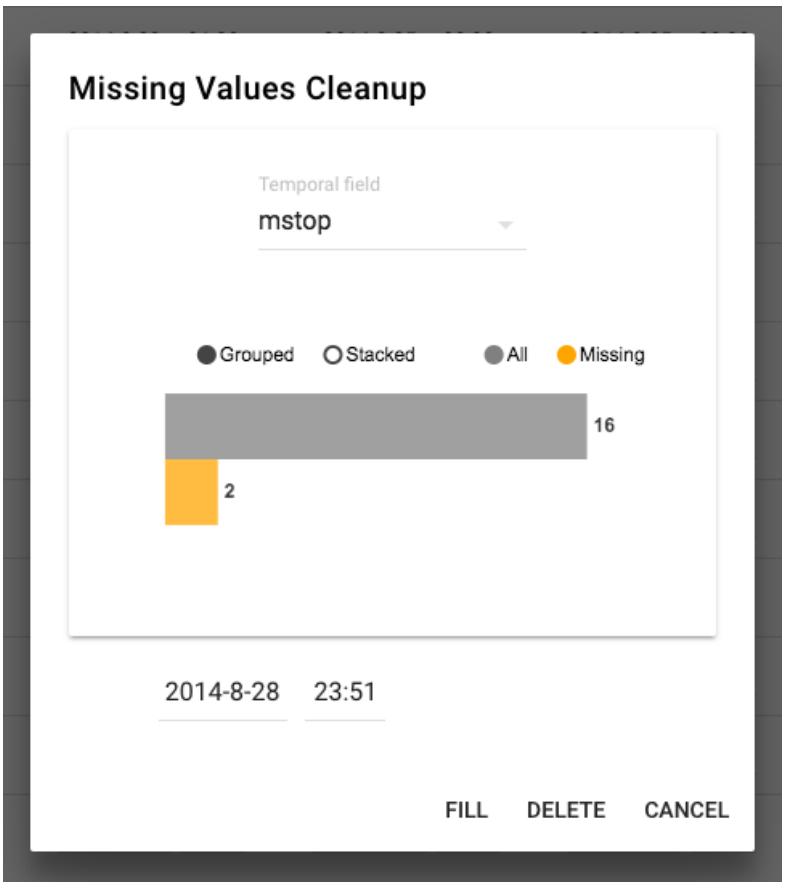


Figure 3.14: Screenshot showing missing values cleanup modal dialog overlay with charts.

Temporal Normalization

There are, basically, two types of temporal normalization operations supported by the prototype – this is not representing normalization in the strictest mathematical sense:

1. Transform all values within a certain timespan or at a certain point in time to a specified date/time
2. Transform all values within a certain interval, effectively "moving" them in time

The former can be accessed either by clicking on a calendar heatmap or a distribution chart bar item on the charts view page (see Figure 3.15). It is generally showing the selected timespan or point in time and enabling the user to set a target value via date and time picker controls for transforming all affected values to. Alternatively, the user can choose to delete all rows within the temporal selection.

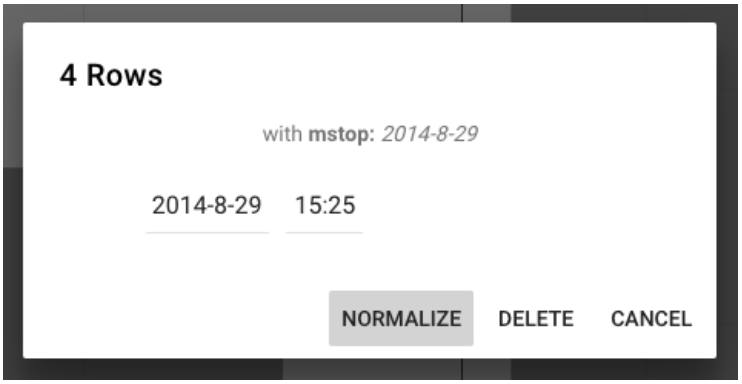


Figure 3.15: Screenshot showing charts page modal dialog on bar or heatmap item click.

The latter is accessible via a dedicated menu item at the bottom of the table editor view page, next to the one for missing values cleanup (see Figure 3.16). There the user can select a temporal field and interval, offering year or month. When month is chosen at max. 10 bars in a chart are displayed each representing a month. When year is chosen the bars represent years. This is an aggregated view visualizing distribution of values with respective amounts sorted in descending order. Again, bars are colored in a neutral gray. When selecting a bar its color changes to black, signalizing the selection. Plus, temporal input field controls show up. In the case of year interval being selected, there is a numeric text input for a target year. In the case of month interval selection, there is additionally a dropdown with the 12 months of a year. In addition to transforming values accordingly, the user can also choose to delete selected rows instead.

Table Column Merging

It is possible to merge time-oriented data columns via drag & drop of table editor headers (see Figure 3.17). Only columns containing temporal data are able to be drag & dropped. Interaction flow, generally, works as follows:

- When starting to drag a header, possible drop targets (i.e., other temporal column headers) are highlighted in light yellow background color
- When dragging over a possible drop target, the hovered column header is highlighted in light green to signalize the drop possibility to the user
- When dragging over a non-temporal column header, it is highlighted in light orange color indicating that it is not a possible drop target
- When the user drops the dragged header on a possible target, a corresponding modal dialog overlay opens

This dialog asks the user to confirm merging columns as specified or cancel otherwise. Merging, basically, works in the following way:

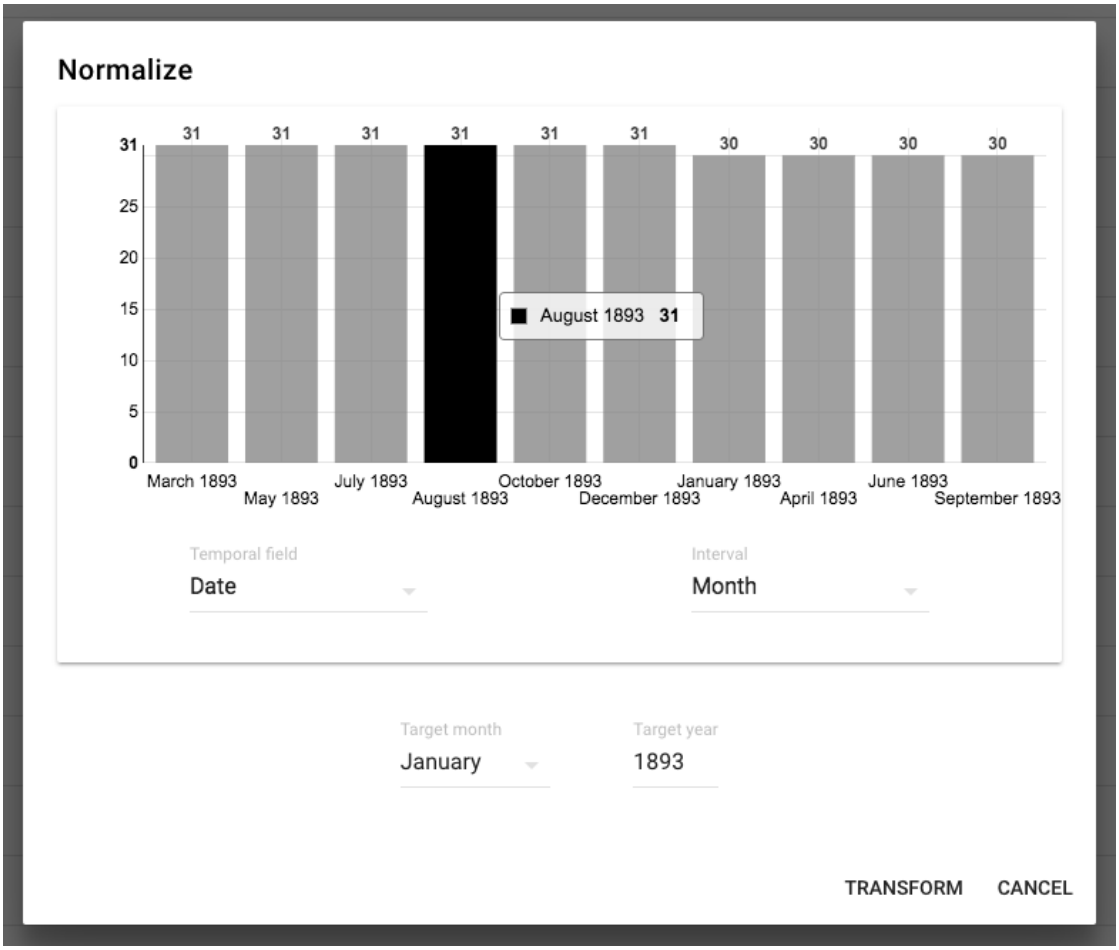


Figure 3.16: Screenshot showing interval normalization modal dialog overlay with interactive bar charts and controls.

- If both column values contain a temporal one, an average of these is used for the merged value
- If only one column value contains a temporal one, the missing one is substituted with the existing
- If both column values contain missing ones, an overall average of all values of the two columns is used

An alternative implementation could have given the user options to choose in a more fine-grained way. Yet, we have found that these sensible defaults should make sense in many cases and the user can still apply further refinements of the column data after merge, if desired. Naturally, on finished operation, the merge respectively drag source column is removed.

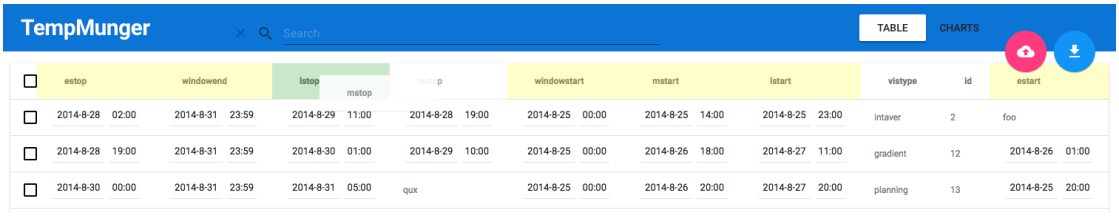


Figure 3.17: Screenshot showing table column merging via drag & drop interaction.

Charts Page

The central UI component regarding charts is kind of a dashboard page (see Figure 3.22).

When there are time-oriented fields present in the respective dataset, it is headed by an interactive calendar heatmap visualization. It gives the user the opportunity to understand the temporal dimension of the data as well as transforming it. Temporal fields can be selected via dropdown.

In any case and below there are two distribution bar charts next to each other. These charts enable the user to get a grasp of the present data, plus transforming it, too. Again, fields can be selected via dropdown.

Calendar Heatmap

The calendar heatmap visualization, generally, allows four temporal scales (see Figure 3.18):

1. Year
2. Month (default)
3. Week
4. Day

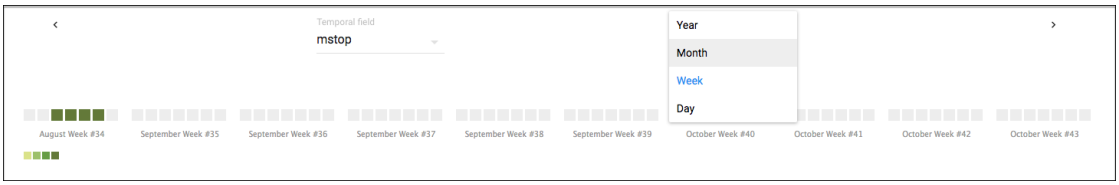


Figure 3.18: Screenshot showing calendar heatmap visualization with interactive controls.

Depending on the chosen scale the calendar view adjusts accordingly. So, for instance, in the case of month scale, it is showing days of each month as boxes. Furthermore, a color scale from gray via light to dark green indicates the amount of dataset entries associated

with a certain temporal value represented by such a calendar item box. On click of an item box, a modal dialog is shown, giving the user transform options.

Distribution Charts

The two distribution bar charts can be seen as some sort of histogram visualizations, showing top aggregations. They are mainly pointed at enabling the user to understand general data distribution qualities of the dataset at hand. Plus, making it possible to easily compare these (see Figure 3.23). Again, clicking a bar opens a modal dialog for further interactive transformation operations, like deletion or time-oriented normalization.

Export

At the end of the day, the user wants to export the wrangled data. Therefore, a button is quite prominently placed in the upper right corner of the application. When there is no time-oriented data included, a button click simply initiates a CSV file export download. Otherwise, a modal dialog overlay is presented first. This dialog lets the user choose a format to apply to all time-oriented data of the to be exported dataset (see Figure 3.19).

Figure 3.19: Screenshot showing modal export dialog with temporal format dropdown.

Three options are implemented and, consequently, available:

1. ISO[12] date/time (e.g., *2017-12-31T12:00:00*)
2. ISO date (e.g., *2017-12-31*)
3. Epoch millis (e.g., *1485037113334*)

Since all time-oriented data is formatted on export in a unified way as well as uniformly stored in UTC/GMT timezone on upload, there is no need to support formatting during previous editing and transformation interactions.

[12] www.iso.org/iso/iso8601

3.5 Qualitative Evaluation

Eventually, the results had to be evaluated. For that matter, we have employed two well-known approaches and tools from the domain of usability engineering. Both of which are introduced and extensively explained in [Nie93]:

1. **Heuristic evaluation**
2. **User/usability tests**

The former basically means that an application and especially its UI is being evaluated by a group of usability experts, step by step examining the to be evaluated system. According to Holzinger [Hol05] three to five usability experts are sufficient for this type of evaluation. Therefore, we have conducted an heuristic evaluation with three experts. Each of the experts should, further, evaluate independently from the others. The evaluation is generally based on heuristics related to usability. The classic ones as described by Nielsen [Nie93] are:

- *Visibility of system status*
- *Match between system and the real world*
- *User control and freedom*
- *Consistency and standards*
- *Error prevention*
- *Recognition rather than recall*
- *Flexibility and efficiency of use*
- *Aesthetic and minimalist design*
- *Help users recognize, diagnose, and recover from errors*
- *Help and documentation*

Forsell and Johansson [FJ10] identified heuristic sets which are especially useful when dealing with applications in the realm of InfoVis. Therefore, we are adding the following:

- *Information coding*
- *Spatial organization*
- *Remove the extraneous*

For the evaluation itself each expert is asked to perform a given set of tasks. Before that, a session going through the application and UI in general is conducted. An observer is present at the evaluation, who is familiar with the application and can be asked related questions. The respective evaluator should note all issues. At the end, found results are gathered and summarized.

These are the tasks we have established for our evaluation:

1. Upload a dataset using a given CSV file
2. Edit time-oriented data via table editor
3. Find potential outliers in the given dataset
4. Identify missing respectively erroneous values
5. Fill these with actual temporal values and/or delete their entries
6. Normalize all entries in a certain month, moving them to another one
7. Move all entries on a specific day to another point in time
8. Delete all entries within a certain timespan or on a certain date
9. Merge time-oriented data columns on the table editor view page
10. Export data as CSV, choosing a format for time-oriented values

User or usability tests, on the other hand, are usually performed in some sort of lab environment. That is, users are given tasks to execute for reaching certain goals with the application and are being observed while doing so. Typically, the test participants should be actually potential users. We have performed such tests with two users, thus, in addition to the previous heuristic evaluation which we have conducted with three different usability and visualization experts. The users of the user/usability tests were given the same tasks to perform as the experts from heuristic evaluation before. Meanwhile testing and particularly afterwards they were interviewed regarding their experience, impressions, and opinions. Finally, we have analyzed and abstracted connected findings.

The main questions all such user test participants as well as heuristic evaluation ones were asked:

1. What is your overall impression?
2. What are the strengths of TempMunger?
3. What are the shortcomings of TempMunger?
4. Do you believe TempMunger can be useful for you?

5. If not, what do you think is needed to make it so?

Combined results of the heuristic evaluation and user tests are as follows, listing found issues and linking them to their respective related, violated heuristics:

1. Insufficient immediate and intuitive visual feedback regarding performed actions, e.g., on missing values cleanup (→ *visibility of system status*)
2. Findability of modal dialogs for missing values cleanup and interval-based normalization is suboptimal (→ *spatial organization*)
3. Separation of the two normalization dialogs as well as connected semantics are not intuitive and could be refined (→ *consistency and standards*)
4. Granularity of temporal scale is partially incomplete (→ *user control and freedom*)
5. Temporal scale coloring in calendar heatmap visualization is sometimes misleading (→ *information coding*)
6. No dedicated highlighting of concrete outlier values, for instance, via corresponding coloring (→ *recognition rather than recall*)
7. Outlier detection action could be made repeatable via button (→ *flexibility and efficiency of use*)
8. Merging operation could be made more useful by offering options and information regarding algorithm (→ *user control and freedom*)
9. Distribution charts for exploratory comparison on charts page could be enhanced, e.g., by adding drilling functionality (→ *information coding*)
10. In some places labels could be added to make controls and respective intention clearer (→ *remove the extraneous*)
11. Missing values cleanup dialog visualization could be refined, for instance, by making stacked charts view the default (→ *information coding*)
12. Calendar heatmap visualization could be enabled to make use of drag & drop interaction (→ *flexibility and efficiency of use*)
13. Time-oriented data could be displayed localized in controls and related input fields (→ *information coding*)
14. Multi-delete action button could be moved to the top of screen or made contextual (→ *user control and freedom*)
15. Large number of columns could lead to displaying glitches on the table editor view (→ *spatial organization*)

The heuristic category which was noted most often is *information coding*, followed by *user control and freedom*. After that, *spatial organization* as well as *flexibility and efficiency of use* both got an equal amount of mentions. Other categories scored only once each. Hence, one can deduct a relative order concerning areas for possible improvements accordingly.

Generally, the approach and prototype was perceived positively and as moving into the right direction. In its current state it was rather seen as a nice proof of concept. To make it a real-world applicable tool it would mainly need to be completed regarding coverage of data types, apart from its focus on time-oriented one now. Additionally, the currently provided set of transformation operations should be completed. Furthermore, the present chart visualizations could be refined and including additional ones was encouraged. Most frequently, possibly dotted, line charts were referred to in the context of time series data visualization as a potentially useful addition. A central issue to address is scalability of the visualizations, also in regard to data granularity.

Aspects of our prototype most praised were the general visual design, the good visually interactive overview offered with charting aid, and smoothness regarding UX. On the other hand, usability was also one of the most controversially discussed topics as it is by its nature a highly subjective and opinionated one. Moreover, an area for future work identified to be desirable would be to provide more transformation suggestions interactively, in a proactive way. Also, additional focus could be laid on further increasing interactive data filtering and drilling capabilities. An interesting idea mentioned was that it could also be useful to some users being able to export visualizations in addition to the raw CSV data which is currently downloadable. Finally, a feedback given by one of the test participants, which we especially appreciate, was *"it does what it's supposed to do"*.

Regarding our interview questions more concretely and in detail: answers to the first question connected to overall impression can be summarized as TempMunger being seen as a nice tool for the use case of working with time-oriented data visual-interactively. Concerning strengths, most commonly pleasantness of general design, UX, and quality of present visualizations were mentioned. The interactive calendar heatmap as well as histogram-like bar chart visualizations were mostly seen as basically fitting for their purpose of conveniently visualizing data distribution focusing on time-oriented aspects and, therefore, useful. Weaknesses were mainly identified to be related to lack of completeness of supported transformation operations and visualizations. Thus, mainly coverage of varied specificity was found to have to be completed in order to make TempMunger a really versatile tool, outgrowing being merely a research prototype.

Further discussion of open issues and future work can be found in the conclusion sections following this one.

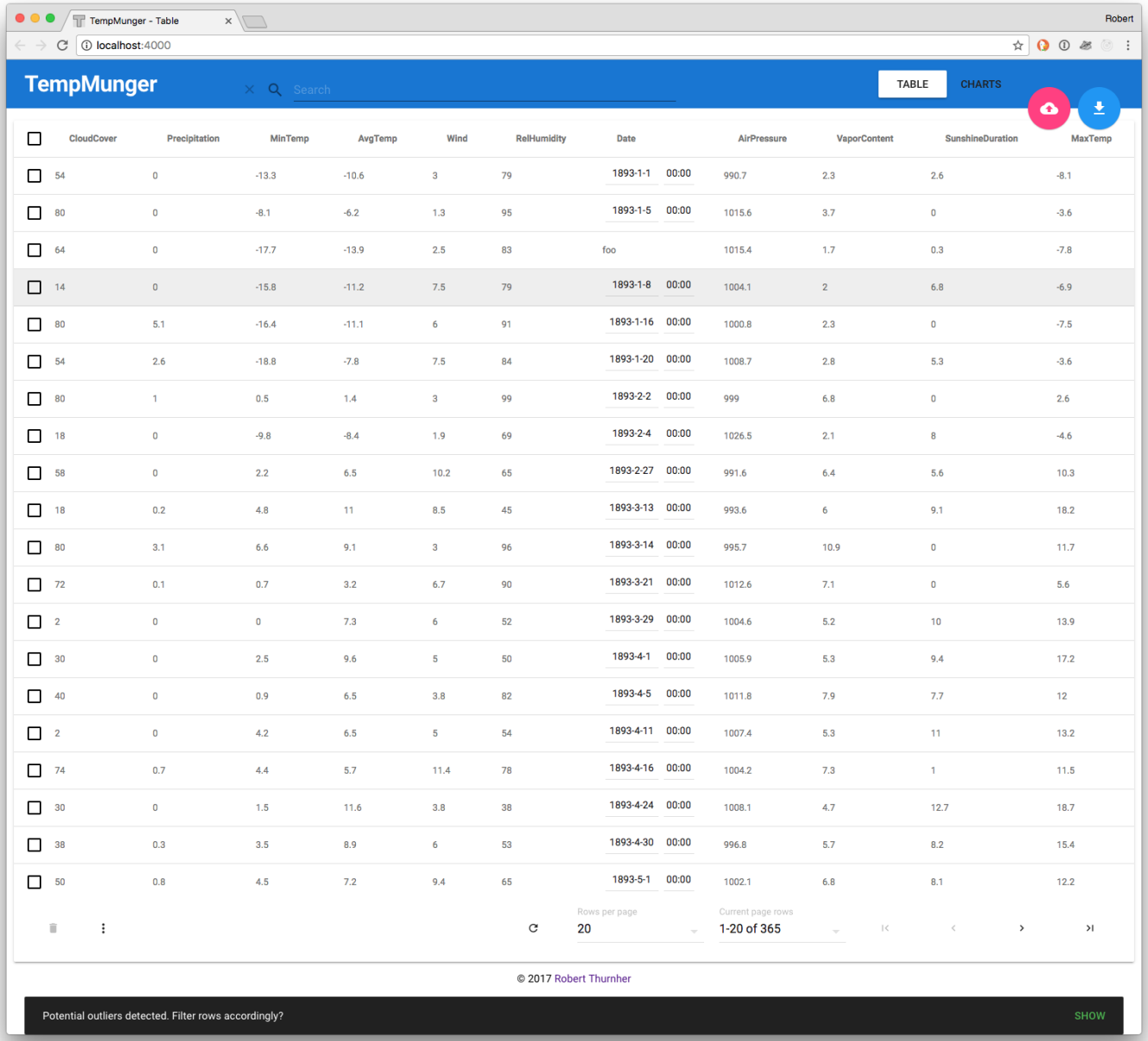


Figure 3.20: Screenshot showing table editor respectively main page including outlier detection info alert.

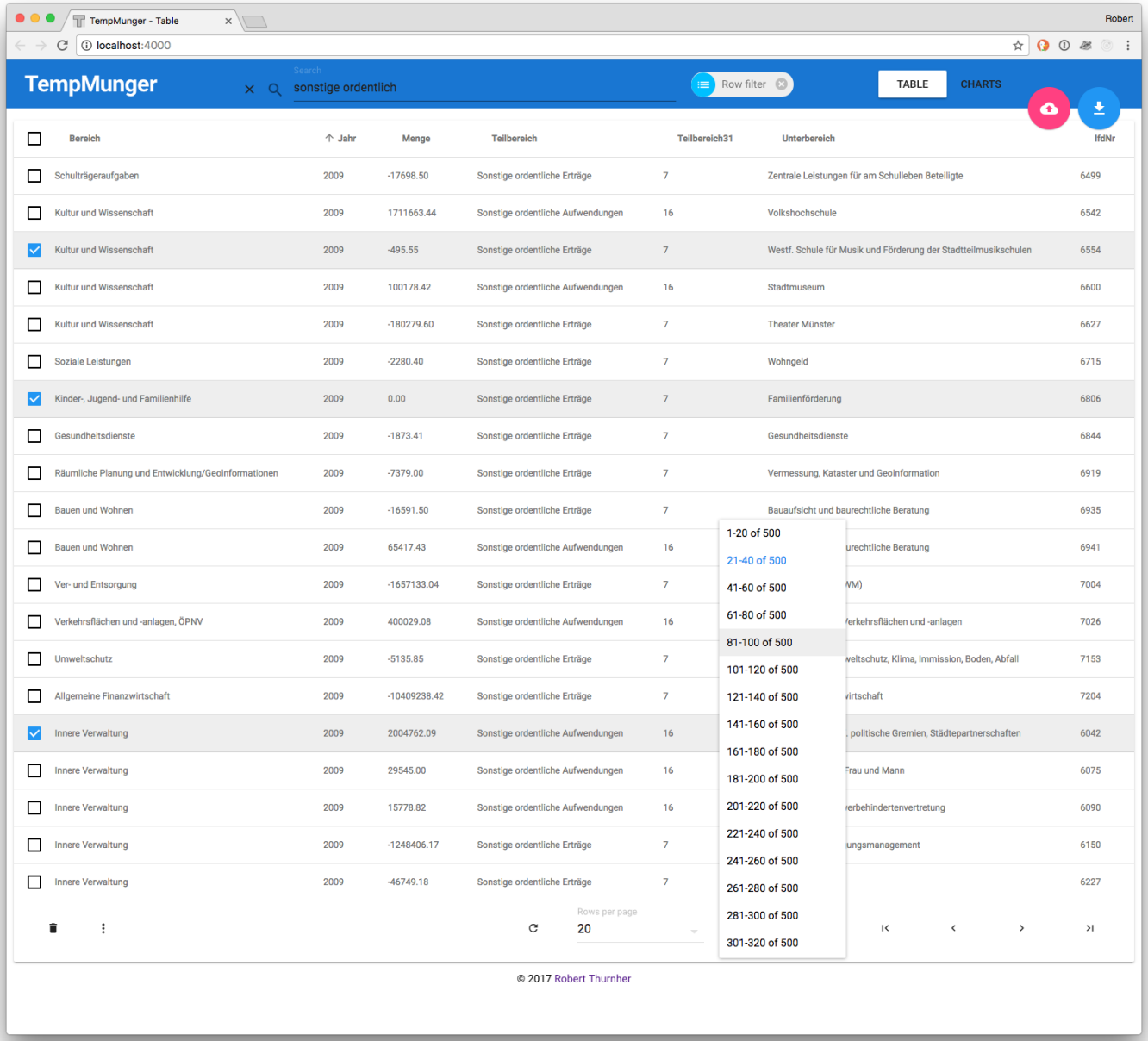


Figure 3.21: Screenshot showing various navigational, search, and filtering options with table editor view.

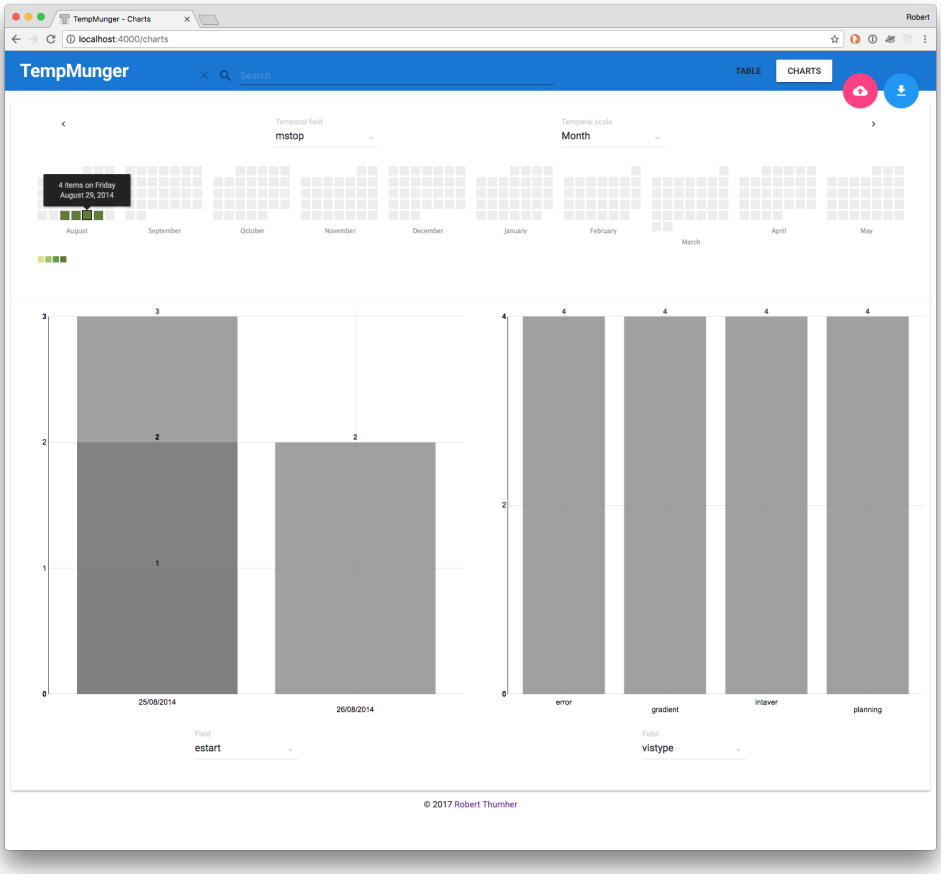


Figure 3.22: Screenshot of charts page with calendar heatmap, offering visual overview.

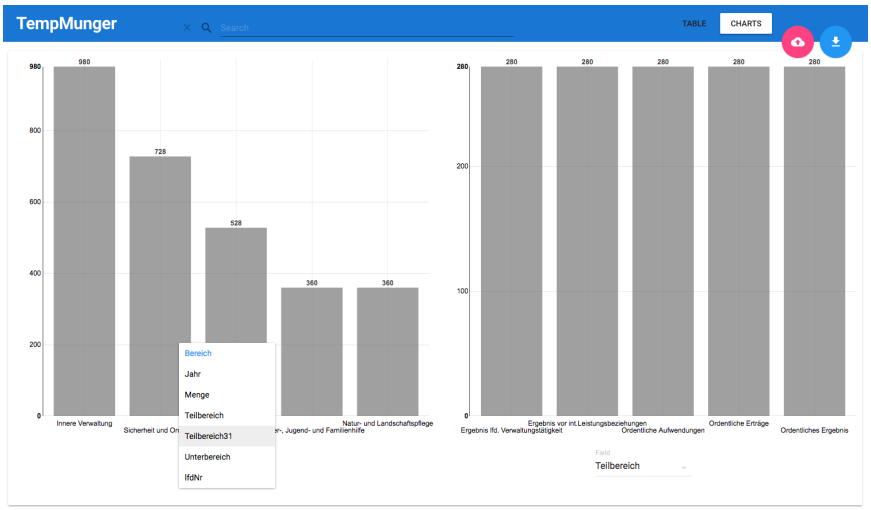


Figure 3.23: Screenshot showing distribution bar charts with their dropdown controls.

CHAPTER 4

Critical Reflection

It is time to reflect on our solution and its results.

4.1 Comparison with Related Work

Therefore, we first conduct a comparison with related work.

Our aim with ***TempMunger*** was to extend mainly the approaches from *DataWrangler* and *OpenRefine*, adding our ideas for improved UX and focusing on time-oriented data support, specifically.

Generally, we believe that we did reach these goals. TempMunger supports wrangling time-oriented datasets, both with the traditional spreadsheet-like UI as well as special, visually interactive charting aids. Evaluation has proven the prototypically implemented approach to be overall useful. Consequently, it can be seen as an advance in the field, at least to some extent. Table 4.1 offers a reflective comparison overview.

Nevertheless, there are some open issues which are being addressed in the next section.

	DataWrangler	*OpenRefine*	*Timelion*	*Jupyter*	***TempMunger***
Time-Oriented	No	Yes	Focus	Supported	Focus
Charting	Little	Some	Extensive	Extensive	Focus
Approach	Spreadsheet	Spreadsheet	CLI + DSL	CLI + REPL	Dashboard

Table 4.1: Reflective comparison of our solution with related work.

4.2 Discussion of Open Issues

So, some issues were found during evaluation which can be subsumed as follows:

- The usage and behavior of the software is not always intuitive
- Immediate visual feedback of system state is partially lacking
- Sometimes possible actions are not completely clear to the user
- Generally, usability can be improved in some parts of the prototype
- Moreover, interactive chart visualizations can be further augmented
- Scalability of visualizations should be addressed more
- The supported transformation operations could be enhanced
- Some aspects are either incomplete or could, at least, be refined

More detailed information concerning evaluation design, process, and tests revealing these issues can be found in the corresponding Section 3.5.

As mentioned above, evaluation has proven the approach to be overall useful, though. Furthermore, feedback regarding and inspiration for future work has been given while conducting qualitative evaluation of the approach and implemented prototype. To sum it up, it appears our achieved results are, all in all, heading towards the right direction.

4.3 Requirements Fulfillment

With our approach and prototype TempMunger we think to, in general, have fulfilled the requirements as derived and listed in Section 3.1.2.

That is, it is capable of loading and working with diverse datasets (**R1**). This is being achieved via easily uploading arbitrary CSV data files. Moreover, it has proven to be, generally, intuitive for casual users (**R2**), while offering some shortcuts for rather power users (**R3**). Focus is indeed on visual-interactive charting aid (**R4**) centering on applying time-oriented data transformations (**R5**). There are numerous interactive chart visualizations provided for this purpose. A visual overview of datasets containing time-oriented data is offered (**R6**). For instance, a special, interactive calendar heatmap visualization is available. We have put emphasis on choosing most effective and efficient visualizations (**R7**). Therefore, we have focused on employing different kinds of bar charts. Interactively exploring datasets is conveniently possible (**R8**). This is backed by various navigational, search, and filtering capabilities. A more traditional tabular editor is supported as well (**R9**). Editing time-oriented data is enhanced with specific date and time picker controls (**R10**). Addressing data quality issues (**R11**) like cleaning missing

and erroneous values, normalizing data, and spotting outliers (**R12**) are all supported via visual-interactive tools and techniques. All the identified data transformation operations we have originally defined in our requirements list are supported, including merging table editor columns via easy drag & drop interaction (**R13**).

4.4 Answering the Research Questions

Consequently, returning to our initial research questions:

1. ***Which data transformations are best supported by analytical methods and for which transformations is visual support beneficial?***

   We have answered this with our approach supporting data quality cleaning, normalization, and merging operations. That is, we have determined these general transformations to be best suited while visual support being benefecial. More in-depth, we are, consequentially, supporting the following transformation operations:

   - Cleanup of missing and erroneous values, allowing fill and deletion
   - Normalization of values regarding points in time and intervals
   - Merging of time-oriented data columns, using average calculations

   Generally, data transformations which require some sort of statistical querying and/or computation are, naturally, best supported by analytical methods. Moreover, whenever data has to be analyzed and/or manipulated in batches or even considering a dataset as a whole, visual support is beneficial for granting necessary overview and insights. The larger and diverse the dataset, the more this comes into effect. Fully automated techniques, on the contrary, make most sense when the transformations to apply are rather straightforward. Our research and design process led to these findings, and our qualitative evaluation confirmed the results.

2. ***How do concrete data wrangling workflow processes look like and how can these processes be supported by VA methods?***

   We have answered this question extensively with the state of the art review and analysis as well as, particularly, through the design of our approach and prototype. More concretely, such processes are oftentimes of exploratory nature. Thus, we are supporting them visual-interactively. Plus, especially repetitive actions, being applied to batches of data via bulk operations, are ones where support by VA methods can shine.

   This way users working on the data can focus on achieving task goals at hand most effectively and efficiently, empowered with superior interactive visualization of the respective dataset. That is, instead of fiddling around with the data manually, mainly being in the dark and applying hand-crafted scripts, clear and direct views of the data, plus its possible as well as plausible transformations, are conveniently presented and at the fingertips of the user.

3. ***What data wrangling tasks need to be tackled in particular when dealing with time-oriented data and how can we support them with VA methods?***

   Again, through the support we have implemented in our prototype for cleaning, normalization, and merging operations we have answered this. I.e., supporting direct manipulation via dedicated UI controls as well as offering interactive charts for the aforementioned operations in a reasonable and intuitive way. The visualizations which we found to be most suitable are mainly bar charts for displaying distributions, think histograms, and calendar heatmap inspired ones.

   Common transformation operations which need to be addressed for data wrangling purposes, in general, are: directly editing single values, deleting rows, also in batches, unifying formats, cleaning up missing and erroneous values, spotting anamolies respectively outliers for subsequent cleanup via according highlighting mechanisms, transforming values of certain fields batch-wise, possibly "normalizing" these to some other specified value, and merging columns according to some algorithm.

   Now, as mentioned above, we have addressed each of these with adequate VA methods, focusing on application to time-oriented data: direct manipulation through dedicated date and time picker UI controls. Bulk deletion of rows via tabular editor controls as well as interactive aggregated charts. Unified formatting is guaranteed via uniform storage and display regarding timezone, plus by export capabilities. Missing values cleanup is, again, supported by histogram-like distribution bar charts. Outlier detection suggestions are enabled via interactive filtering notifications. Batch-wise transformation concerning normalization is backed by interactive bar charts as well as calendar heatmap based interaction.

   Visualizing data distribution via bar charts in a histogram-inspired way and time-oriented data via calendar heatmaps are especially powerful tools in this context. The general usefulness of these visualizations was underpinned by our evaluation.

Therefore, we can conclude that, all in all, we have answered our main research question satisfactorily. The question having been:

How can we support data wrangling with VA techniques?

To sum it all up, we can support data wrangling this way mainly by focusing on tasks which are, on the one hand, related to batched data transformation operations as well as of rather complex nature and, on the other hand, thus requiring special attention and oversight. These are tasks where fully automated approaches fall short, as a human adequately empowered through VA techniques can, still, perform better. Thus, the sweet spot is most probably located somewhere in between, augmenting an interactive interface driven by powerful visualizations with semi-automatic suggestions, reasonably.

CHAPTER 5

Summary and Future Work

This thesis explored applying VA to data wrangling, focusing on time-oriented data.

Hence, after deepened study, presentation, and analysis of related state of the art, an approach has been designed and prototypically implemented. UX personas and UI mockups were valuable tools for our design process. Iteratively developing the prototype in an agile manner was worthwhile as well. As evaluation showed, there are still some issues mainly relating to usability, which can be further improved. Nonetheless, the approach proved to be overall useful.

Our prototype mainly offers interactive dashboard visualizations as a web-based application. Special emphasis was put on crafting the UI as well as applying VA methods reasonably. Therefore, we have built visual-interactive charts supporting time-oriented data transformations. To make this all work well, we have also invested considerable effort in a sound underlying software design and architecture.

Future work should focus on extending the amount of available transformations supported via such visually interactive charting aids. Specifically, giving the user more fine-grained control in some of the already present operations, plus adding completely new ones. Also, additional attention should be paid to addressing scalability issues of the visualizations. Particularly, in terms of data granularity as well as related coverage. Moreover, enabling undo of operations, plus repetition of them via some sort of user-controlled history mechanism and/or storable scripts, somewhat similar to how DataWrangler does it, would be a reasonable addition.

Also, the inference aspect of the approach, interactively providing transform suggestions, could be emphasized more and, thus, enhanced. The outlier detection component we have integrated in our approach was generally well received in evaluation and tests, indicating this is leading into the right direction. Moreover, as it is currently just a quite basic way of applying ML techniques to the problem, there is definitely room for more.

While conducting the qualitative evaluation of our prototype quite some positive statements regarding the overall visual design, smooth UX, and general quality of implemented interactive visualizations were made. Things like *"good overview"*, *"can be easily done"*, and *"it does what it's supposed to do"* come to mind. Consequently, it appears our approach and prototype was generally perceived as well done and valuable.

In our opinion, wrangling time-oriented data being supported by VA methods is an interesting field of research where *TempMunger* just scratched the surface, delivering some input and, hopefully, inspiration to advance it further.

APPENDIX A

Software Design and Architecture

In this appendix, design and architecture of the software prototype is presented. First of all, a general overview of the system is given. Followed by more in-depth dives into various parts and components of the solution. Moreover, technical foundations of the implementation are described and their interplay explained.

A.0.1 System Overview

The prototypically implemented, proposed solution is basically a web application. It can be run locally as well as deployed to a hosted server environment. This hosting is possible to be conveniently performed via *Docker*[1] containerization. The system consists of a web frontend and an API backend. Both of which can be hosted as separate Docker containers. The application is targeting and, hence, optimized for desktop browser clients. So, the frontend mainly emits HTML/CSS/JS and communicates with its respective client via HTTP(S).

The frontend is a *Node.js*[2] application and the backend a *Java Virtual Machine (JVM)* one. On the frontend, the UI is (pre-)rendered server-side in addition to client-side. This technique is called isomorphic or universal rendering. On the backend, data is stored and queried via *Elasticsearch*[3], a high-performance search and real-time analytics engine as well as document store, while transformed and analyzed via *Apache Spark*[4].

The latter is an engine for large-scale, near real-time data processing with convenient access to ML algorithms via its *MLlib* extension library. Communication between backend

[1] www.docker.com/what-docker
[2] nodejs.org/en/about/
[3] www.elastic.co/products/elasticsearch
[4] spark.apache.org

and frontend is conducted with Representational State Transfer (ReST), cf. [Fie00], and WebSockets[5]. Figure A.1 is presenting this system overview from a bird's eye view.

In a Nutshell

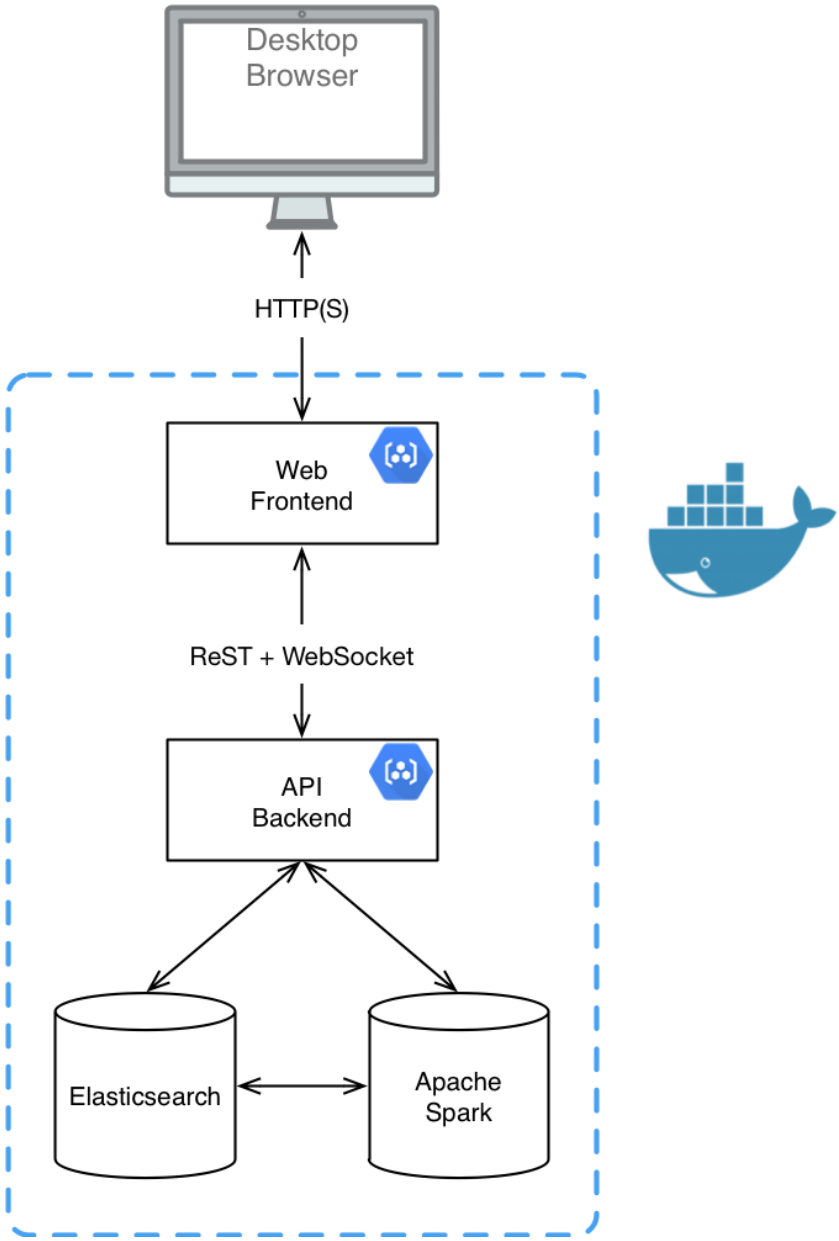


Figure A.1: High-level diagram of our SW architecture.

[5] tools.ietf.org/html/rfc6455

Going Live with Docker

As mentioned above, the system is basically "dockerized" for container-based hosting. I.e., the frontend as well as backend are each equipped with a fitting *Dockerfile*. Container images are based on the lightweight *Alpine Linux*[6] distribution which is particularly well suited for that purpose. In order to support an agile development process, it is easy to build and deploy the application. More concretely, there is a *Makefile* set up which allows for quick deployment of Docker images (incl. building and publishing these via private Docker repositories) to a virtual host. The author chose in this example *DigitalOcean*[7] as hosting service provider due to the convenient *Developer Experience (DX)* it offers.

Multiple host nodes as well as multiple instances of either backend or frontend container were not relevant for the purpose of this prototype. It is generally possible to set this up and basically supported, though. Moreover, Elasticsearch and Spark are simply being run in embedded mode within the backend component. Again, it is generally possible to configure connection to real dedicated clusters of each. Yet, the point of this prototype was not really geared towards proving *"big data"* load capabilities.

On the virtual host an *Nginx*[8] web server is configured to proxy the local containerized application servers to the public Internet. Access to the web application is then restricted via HTTP basic authentication. HTTPS is enabled through *Let's Encrypt*[9].
The interested reader may ask the author for a link with credentials.

Project Structure and Setup

In general, the system is comprised of three software projects. One for the backend application, one for the frontend, and sort of an "umbrella", the master one. The latter pulls the former ones in, via *Git* submodule setup. So, Git is used as Version Control System (VCS), respectively for Source Code Management (SCM) with private *Github* repositories, owned by the developer. FS structure of this master one also loosely resembles *CVAST* research group guidelines[10]. Furthermore, it contains aforementioned Makefile for convenient builds and deployments. The Continuous Integration (CI) tooling of choice is *CircleCI*[11], a Software as a Service (SaaS) provider with a good DX. Build tool of the backend project is *Gradle*[12]. The frontend uses a combination of *Yarn*[13] dependency management and *Gulp*[14] task scripts. Source code documentation is generated with *Dokka*[15] for the backend, and *ESDoc*[16] for the frontend.

[6] www.alpinelinux.org/about/
[7] www.digitalocean.com
[8] nginx.org/en/
[9] letsencrypt.org/about/
[10] www.cvast.tuwien.ac.at/node/27
[11] circleci.com
[12] gradle.org
[13] yarnpkg.com/en/
[14] gulpjs.com
[15] kotlinlang.org/docs/reference/kotlin-doc.html
[16] esdoc.org

A.0.2 Backend

The backend mainly consists of the high-level components as laid out in Figure A.2. It is basically a *Spring*[17] *Boot*[18] application using *Spring MVC* for its ReST controller layer. *Spring Messaging* is used for transparent WebSocket communication, and *Reactor*[19] *Event Bus* for reactive messaging within the outlier detection implementation. The Elasticsearch data layer is based on *Spring Data Elasticsearch*[20]. Connection between Elasticsearch data storage and Apache Spark processing is done via *ES-Hadoop* connector libraries[21].

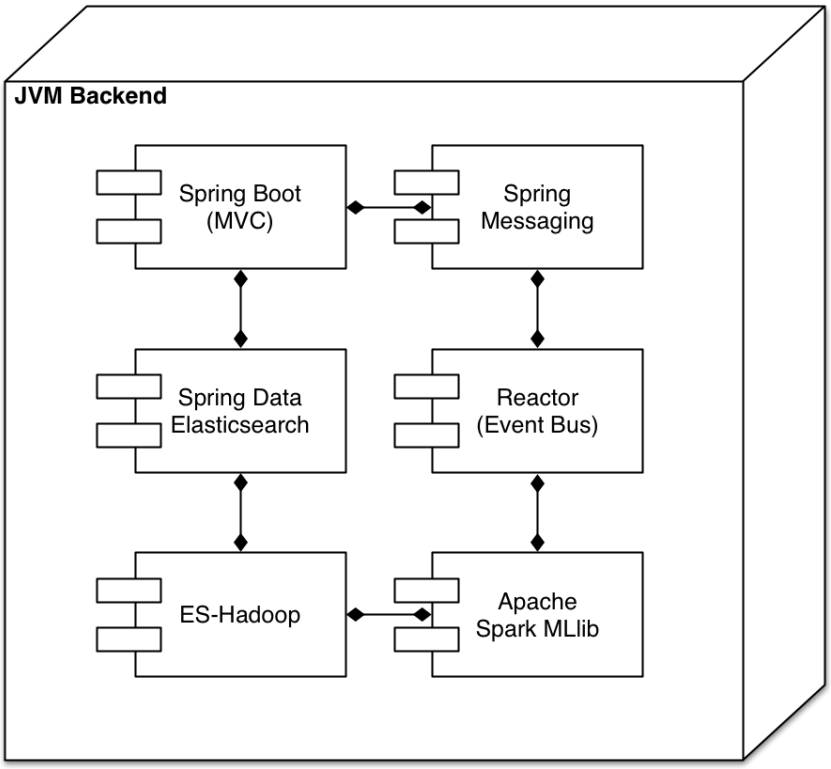


Figure A.2: Backend components diagram.

Documentation of the ReST API is generated via *Spring REST Docs*[22]. This integrates nicely into the automated testing infrastructure of the project. Thus, it can be seen as a superior solution compared to the en-vogue *Swagger*[23] in respect of it is possible to combine automated documentation generation with manually written one via *Asciidoctor*[24].

[17] projects.spring.io/spring-framework/
[18] projects.spring.io/spring-boot/
[19] projectreactor.io
[20] projects.spring.io/spring-data-elasticsearch/
[21] www.elastic.co/guide/en/elasticsearch/hadoop/current/spark.html
[22] projects.spring.io/spring-restdocs/
[23] swagger.io
[24] asciidoctor.org

The backend project itself prefers structuring its top-level packages semantically. E.g., there is a top-level package ***importexport*** containing ***dao*** (as in *Data Access Object*) and ***service*** sub-level packages – not the other way round, as it would be more traditional fashion. Unit and integration tests are present, written with state-of-the-art libraries.

Some deeper dive into the technologies employed for the backend follows.

Kotlin on the JVM

The *Kotlin*[25] programming language, by *JetBrains*, is used on the backend.

It is a relatively young, statically typed, and concise JVM-based language with seamless Java interoperability. Mainly, it is adding quite some syntactic sugar, reasonably, as well as increased type inference support. Plus, it is resolving some of Java's pain points, such as reference nullability – a.k.a. the *"billion-dollar mistake"*[26]. Generally, it is taking a pragmatic stance and focuses on industrial software development needs. As opposed to, e.g., Scala which can be seen as quite similar in various aspects, while being more computer science and particularly research focused, though.

Figure A.3 shows an exemplary code snippet.

```
private fun transform(query: QueryBuilder, field: String, to: YearMonth) {
    withTimedLogging(field) {
        updateRdd(query)
        transform(it, to)
    }
}
```

Convenient lambda syntax
and general reasonable brevity

Figure A.3: Kotlin sample demonstrating some of its useful features.

Microservices with Spring Boot

Spring Boot offers a useful approach to developing and running microservices.
It was, originally, heavily inspired by the *Dropwizard*[27] project.

The topic of building microservices itself is covered well in [New15]. The basic idea here is to develop small, self-contained, and purpose-focused services with corresponding monitoring, provisioning, and orchestration capabilities. This stands in rather stark contrast to the more traditional approach of building monolithic applications. So, in general, Spring Boot provides a lot of features and abstractions with little necessary configuration out-of-the-box. Furthermore, it integrates well with Kotlin, both sharing a pragmatic paradigm of software development philosophy.

[25] kotlinlang.org
[26] lambda-the-ultimate.org/node/3186
[27] www.dropwizard.io

From Elasticsearch to ES-Hadoop

At the heart of data storage and querying, Elasticsearch is in use.
Spring Data Elasticsearch is providing some convenient abstractions.

As mentioned above, data processing, i.e., transformation operations and outlier detection, is performed via Apache Spark and its MLlib. Native bridging of these two popular big data technologies is provided via ES-Hadoop connector libraries. This is, basically, rooted in the fact that Apache Spark was born within the Apache Hadoop[28] ecosystem. Spark can be run completely independent from Hadoop infrastructure, though. The connector libraries are officially supported and developed by *Elastic*[29], the company behind Elasticsearch and related products. In addition to the Scala API, since Spark is written in Scala, a Java API is available which is used in our project.

Apache Spark & MLlib

As aforementioned, Spark is at the heart of data processing.

Its foundational construct is called *RDD*. That is, the main concept is to provide a scalable solution for loading large datasets into memory, utilizing distributed computing. One then can easily perform transformational operations, like the common functional *map* routine, on such an RDD. The underlying complexities are transparently hidden beneath by the abstraction. This can be seen as fitting nicely into the philosophy of *"**simple made easy**"*, as advocated by *Rich Hickey*[30].

Additionally, extension libraries have been built enhancing processing capabilities. One of them is MLlib which provides convenient access to ML algorithms. This is used in our project for its temporal outlier detection implementation.

Reactive Messaging

Reactive programming is rather popular these days.
At its core stands the so-called *Reactive Manifesto*[31].

It is a term for event-based programming. That is, publish/subscribe mechanisms for message-driven communication. Historically, it is build upon the notions from the classic *Observer* pattern enhanced with functional programming techniques. This paradigm fits especially well with the requirement of asynchronous notifications for the outlier detection component of our system.
Consequently, it is employed therein.

Concrete library used is Reactor and its implementation of the *Event Bus* abstraction. The former is, generally, a modern high-level take on the *ReactiveX*[32] approach.

[28] hadoop.apache.org
[29] www.elastic.co
[30] www.infoq.com/presentations/Simple-Made-Easy
[31] www.reactivemanifesto.org
[32] reactivex.io

A.0.3 Frontend

The frontend is a modern Node.js application. It is based on *Este.js*[33], a useful assembly of libraries and best practices for easily bootstrapping a universal *Redux/React* project.

Its further refined stack for our project, basically, comprises of *Isomorphic Fetch*[34] for ReST layer communication, *STOMP over WebSocket*[35] with *SockJS*[36] for communicating with the corresponding Spring Messaging interface on the backend, a Redux[37] layer containing business logic, a React[38] UI as well as related URL path routing, plus UI toolkits including *D3.js*[39] charting or common respectively extensible UI components, and all running on an *Express*[40] web framework based server (see Figure A.4).

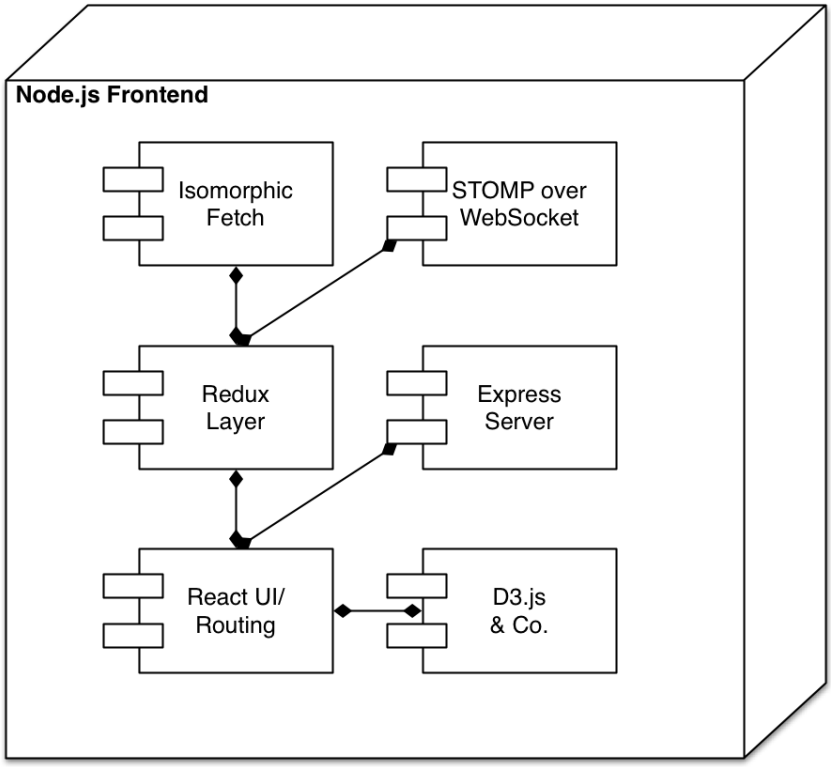


Figure A.4: Frontend components diagram.

Project structure outlines an adapted Este scaffold. Automated testing is based on *Jest*[41].

[33] github.com/este/este
[34] github.github.io/fetch/
[35] jmesnil.net/stomp-websocket/doc/
[36] sockjs.org
[37] redux.js.org
[38] facebook.github.io/react/
[39] d3js.org
[40] expressjs.com
[41] facebook.github.io/jest/

Contemporary JavaScript

The frontend application is written in contemporary JS, that is, *flow-typed*[42] *ES6+*.

Figure A.7 is demonstrating some useful features as well as techniques when used together with Redux/React. Static type-checking is especially valuable for detecting bugs and errors early, at compile time. Compilation to common ES5 is performed via *Babel*[43] plus a *Webpack*[44] toolchain is in place – minifying, optimizing, and bundling web app assets. In addition, *ESLint*[45] helps keeping the code clean and adhering to good practices.

React with Material Design

As mentioned, the UI is based on React. A technology and library created at *Facebook Engineering*. It enables one to build composable UI components in a declarative way, clearly reasoning about their state via cleanly applied *Separation of Concerns (SoC)* pattern. Templating is achieved with so-called *JSX*[46]. That is, XML-like syntax embedded directly in JS code. Moreover, its performance is really good as partial re-rendering of to-be-updated Document Object Model (DOM) tree nodes is supported transparently.

The React programming model usually takes a bit to get accustomed to, but its benefits do pay off, especially in the long run. Particularly, clear separation of application state and DOM is a big leap forward in this space.

General UX of the UI of our application is based on the *Material Design*[47] approach and specification by Google. As implementing components kit, *React Toolbox*[48] is used. It provides a quite comprehensive, carefully crafted set of common UI controls and elements. For CSS needs, transpiled *Sass*[49] is employed, which is pretty popular for this use case.

As mentioned before, the application is rendered client-side, as commonly expected from a Single Page Application (SPA), yet, initially also server-side. This isomorphic pre-rendering on the server improves the UX by minimizing potential, irritating "flicker" effects on initial page load. In particular, when loading a page with an URL which requires routing. Additionally, it helps with Search Engine Optimization (SEO) if applicable. Routing is built on client *HTML5 History API*[50] for clean, universal URL path layout.

From Flux to Redux

The *Flux* architecture was introduced by Facebook Engineering as an approach to better structuring SPAs. It is based on the idea of strictly one-way data flow and binding.

[42] flowtype.org
[43] babeljs.io
[44] webpack.js.org
[45] eslint.org
[46] facebook.github.io/react/docs/introducing-jsx.html
[47] material.io
[48] react-toolbox.com
[49] sass-lang.com
[50] diveintohtml5.info/history.html

This makes it much simpler to reason about an UI system, and therefore less error-prone. The basic concepts are shown in Figure A.5 which stems from official documentation material[51]. So, generally, actions are triggering system state changes and, consequently, UI updates in a unidirectional manner. Redux took theses concepts and further simplified them, iterating upon in its implementation. Solutions for handling asynchronous events, as they are common in SPAs, include injecting a promise middleware into Redux as well as extending it with reactive processing via *redux-observable*[52] plus *RxJS*[53]. Both approaches are used in the prototype to varying degrees with an emphasis on the former one, to get a feel for them. A real product would most probably rather focus on one of them then. ES6+ *async/await* is helpful for coding with promises sequentially, saving one from *"callback hell"*. Immutable data structures and collections provided by *Immutable.js*[54] are also commonly used. Plus, *Ramda*[55] fits in well as a functional programming utilities library. Speaking of utility libraries in our project, ones centered on temporal data processing are *date-fns*[56] and *js-joda*[57] (a *java.time*-compliant JS port).

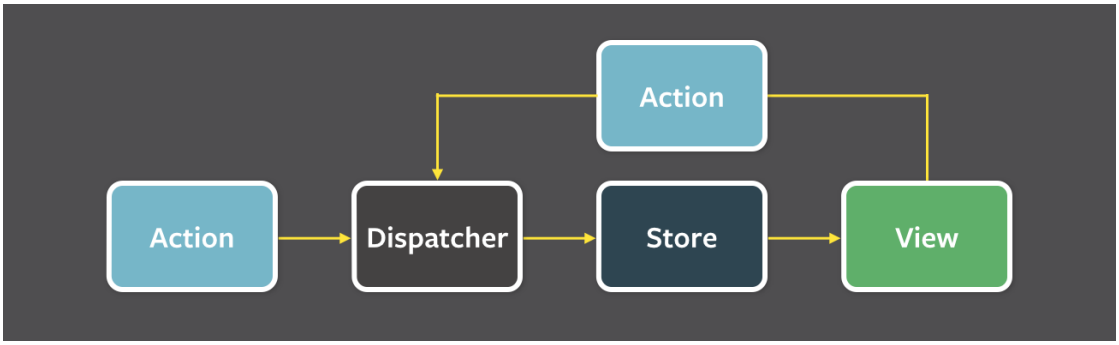


Figure A.5: Simple Flux architecture diagram from Facebook Engineering.

Fetch API & WebSockets

The *Fetch API* is a modern approach to *Ajax* communication. That is, asynchronous client/server communication of a browser application via HTTP. This, traditionally, is based on usage of the so-called *XMLHttpRequest* browser JS API. The rather young Fetch API takes the basic concepts of async browser ReST calls and pours it into a contemporary form, conveniently usable.

WebSockets allow for bidirectional communication. That is, instead of solely pulling from a server, a browser client app is also able to get data pushed by the server. It is especially useful in event-based systems, such as messaging-related ones.

[51] facebook.github.io/flux/docs/in-depth-overview.html#structure-and-data-flow
[52] redux-observable.js.org
[53] reactivex.io/rxjs/
[54] facebook.github.io/immutable-js/
[55] ramdajs.com
[56] www.npmjs.com/package/date-fns
[57] js-joda.github.io/js-joda/

D3.js Charting

D3.js (as in *Data-Driven Documents*) is a very popular JS library, primarily used for interactive charting. Originally, it was developed at *Stanford Vis Group* (see [BOH11]). It is based on Scalable Vector Graphics (SVG) and surrounded by an ecosystem of components as well as extensions.

For most of the charts in our application, *NVD3*[58] is used. It is a useful repository of various ready-to-use charts, all conveniently customizable. Plus, they are nicely crafted with appealing default look & feel, supporting decent animations. Figure A.6 shows an overview gallery from their documenting website. Our calendar heatmap visualization component is based on *cal-heatmap*[59]. It is a widely used implementation with a multitude of customization and tweaking options. Integration of D3 with React works, all in all, relatively straightforward. Yet, some resorting to direct DOM manipulation and event handling is required, unfortunately, due to legacy reasons inflicted by the former.

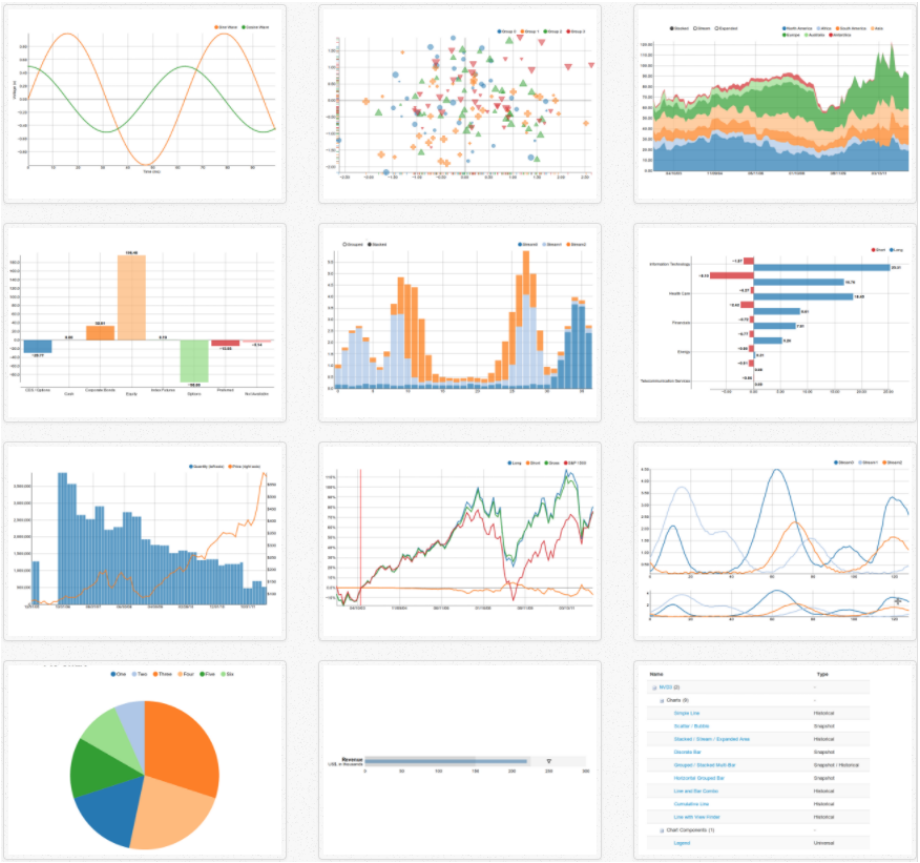


Figure A.6: A screenshot of an NVD3 charts gallery, as presented on nvd3.org.

[58] nvd3.org
[59] cal-heatmap.com/v2/

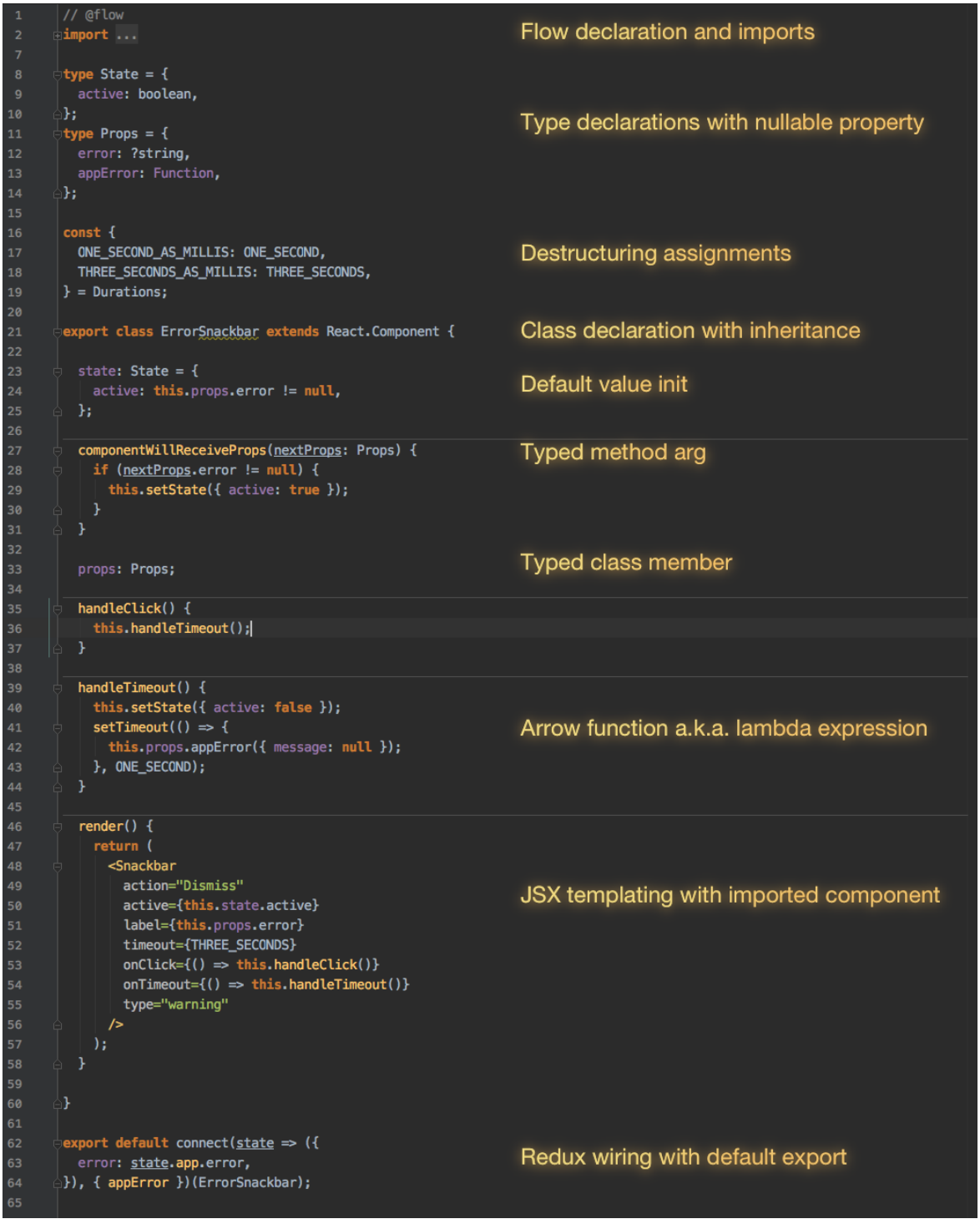


Figure A.7: Flow-typed ES6+ Redux/React UI component showing some useful features.

APPENDIX B

Information Retrieval Background

In this appendix some IR background is presented, as core data storage and querying of our prototype is based on.

TF-IDF & Apache Lucene

The equations B.1 to B.4 lay out some of the most fundamental concepts behind IR.

$$\mathrm{tf}(t, d) = f_{t,d} \tag{B.1}$$

$$f_{t,d} = \begin{cases} 1 & \text{if } t \text{ occurs in } d \\ 0 & \text{otherwise} \end{cases} \tag{B.2}$$

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{B.3}$$

$$\text{tf-idf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D) \tag{B.4}$$

It is *Term Frequency – Inverse Document Frequency (TF-IDF)*[1]. Basically, this is an effective as well as efficient way to score term query search hits based on term occurrences in a corpus of documents. The illustrated formula set uses a simple boolean frequency.

A popular and high-quality search engine implementation in Java is *Apache Lucene*[2]. It is also the foundation on which Elasticsearch is built upon [GT15].

[1] en.wikipedia.org/wiki/Tf-idf
[2] lucene.apache.org

APPENDIX C

Supported Date/Time Formats

Our data model, generally, supports following date/time formats[1] for automatic parsing:

- *ISO_ORDINAL_DATE*
- *BASIC_ISO_DATE*
- *ISO_DATE*
- *ISO_DATE_TIME*
- *ISO_ZONED_DATE_TIME*
- *yyyy/MM/dd*
- *yyyy/MM/dd HH:mm[:ss[.SSS]]*
- *MM/dd/yyyy*
- *MM/dd/yyyy HH:mm[:ss[.SSS]]*
- *dd/MM/yyyy*
- *dd/MM/yyyy HH:mm[:ss[.SSS]]*
- *dd.MM.yyyy*
- *dd.MM.yyyy HH:mm[:ss[.SSS]]*
- *EPOCH_MILLIS*

[1] Cf. docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html

List of Figures

List of Tables

List of Algorithms

Index

Glossary

big data is a buzzword term primarily referring to data exceeding volume and size so that it cannot be properly handled with more traditional approaches and technologies anymore. 69, 72

data mining consists of techniques being applied, mainly , for ML. 1, 14

data munging is a synonym for data wrangling. 1

data science concerns itself with gaining insights from data scientifically. 1, 16, 17, 83

data warehousing is a classic approach to analytical reporting based on data marts mostly periodically fed from multiple sources. 8

data wrangling consists of techniques being applied to transform data. 1–3, 7, 9, 13, 15, 17, 18, 63–65

semantic web is a buzzword term commonly subsuming approaches and technologies supporting the notion of making the mainly text-based, unstructured data on the WWW more accessible by enriching it with machine-readable semantics. 11

Acronyms

CI Continuous Integration. 69

CLI Command Line Interface. 15–17, 61

CVAST Centre for Visual Analytics Science and Technology. 69

DOM Document Object Model. 74, 76

DSL Domain-Specific Language. 9, 10, 15, 61

DX Developer Experience. 69

ETL Extract, Transform, Load. 8

FS File System. 44, 69

GMT Greenwich Mean Time. 40, 52

GUI Graphical User Interface. 8, 9, 15, 16, 18, 21, 83

HCI Human-Computer Interaction. 1, 3

HMR Hot Module Replacement. 39

IDE Integrated Development Environment. 10, 39

InfoVis Information Visualization. 1, 2, 10, 17, 18, 53

IR Information Retrieval. 1, 3, 40, 79

JVM Java Virtual Machine. 67, 71

L/PBD Learning/Programming by Demonstration. 8, 9, 13

ML Machine Learning. 1, 3, 36, 42, 65, 67, 72

PBE Programming by Example. 8

R&D Research and Development. 7, 12, 18

RCP Rich Client Platform. 15, 21, 83

RDBMS Relational Database Management System. 8, 9, 11

RDD Resilient Distributed Dataset. 40, 72

RDF Resource Description Framework. 11

REPL Read–Eval–Print Loop. 16, 61

ReST Representational State Transfer. 68, 70, 73, 75

SaaS Software as a Service. 69

SCM Source Code Management. 69

SEO Search Engine Optimization. 74

SoC Separation of Concerns. 74

SPA Single Page Application. 74, 75

SVG Scalable Vector Graphics. 76

TF-IDF Term Frequency – Inverse Document Frequency. 79

UI User Interface. 3, 10, 13–15, 17, 19, 23, 24, 31–33, 36, 39, 41, 44, 51, 53, 54, 61, 64, 65, 67, 73–75, 77, 83, 84

UTC Coordinated Universal Time. 40, 52

UX User Experience. 1, 3, 13, 18, 23, 36, 56, 61, 65, 66, 74

VA Visual Analytics. 1–3, 18, 63–66

VCS Version Control System. 69

Bibliography

[AMST11] Wolfgang Aigner, Silvia Miksch, Heidrun Schumann, and Christian Tominski. *Visualization of Time-Oriented Data*. Human-Computer Interaction. Springer, 1st edition, 2011.

[BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, December 2011.

[BRG+12] Jürgen Bernard, Tobias Ruppert, Oliver Goroll, Thorsten May, and Jörn Kohlhammer. Visual-interactive Preprocessing of Time Series Data. In Andreas Kerren and Stefan Seipel, editors, *SIGRAD*, volume 81 of *Linköping Electronic Conference Proceedings*, pages 39–48. Linköping University Electronic Press, 2012.

[CKP08] Laura Chiticariu, Phokion G Kolaitis, and Lucian Popa. Interactive Generation of Integrated Schemas. In Jason Tsong-Li Wang, editor, *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 833, Vancouver, BC, Canada, 2008. ACM.

[CMS99] Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann, San Francisco, CA, USA, 1999.

[Coo04] Alan Cooper. *The Inmates Are Running the Asylum*. Sams - Pearson Education, 2nd edition, 2004.

[DFH07] David R Karger David F Huynh, Robert C Miller. Potluck: Semi-ontology Alignment for Casual Users. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference*, ISWC + ASWC '07, pages 903–910, Busan, Korea, 2007. Springer.

[DJ03] Tamraparni Dasu and Theodore Johnson. *Exploratory Data Mining and Data Cleaning*. Wiley Series in Probability and Statistics. Wiley-Interscience, 2003.

[DJMS02] Tamraparni Dasu, Theodore Johnson, S Muthukrishnan, and Vladislav Shkapenyuk. Mining Database Structure; Or, How to Build a Data Quality

Browser. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, SIGMOD '02, pages 240–251, Madison, WI, USA, 2002. ACM.

[FG05] Kathleen Fisher and Robert Gruber. PADS: A Domain-specific Language for Processing Ad Hoc Data. *SIGPLAN Notices*, 40(6):295–304, June 2005.

[Fie00] Roy T Fielding. Architectural Styles and the Design of Network-based Software Architectures. *Building*, 54:162, 2000.

[FJ10] Camilla Forsell and Jimmy Johansson. An Heuristic Set for Evaluation in Information Visualization. *AVI '10 Proceedings of the International Conference on Advanced Visual Interfaces*, 10(3):199–206, 2010.

[Gar11] Jesse James Garrett. *The Elements of User Experience: User-Centered Design for the Web and Beyond*. New Riders, 2nd edition, 2011.

[GT15] Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide*. O'Reilly Media, 1st edition, 2015.

[Gul10] Sumit Gulwani. Dimensions in Program Synthesis. *Principles and Practice of Declarative Programming*, pages 1–1, October 2010.

[Gul11] Sumit Gulwani. Automating String Processing in Spreadsheets Using Input-Output Examples. *SIGPLAN Notices*, 46(1):317–330, 2011.

[Hel08] Joseph M Hellerstein. Quantitative Data Cleaning for Large Databases. United Nations Economic Commission for Europe (UNECE), 2008.

[HHH+05] Laura M Haas, Mauricio A Hernández, Howard Ho, Lucian Popa, and Mary Roth. Clio Grows Up: From Research Prototype to Industrial Tool. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 805–810, New York, NY, USA, 2005. ACM.

[Hol05] Andreas Holzinger. Usability Engineering Methods for Software Developers. *Communications of the ACM*, 48(1):71–74, January 2005.

[KHP+11] Sean Kandel, Jeffrey Heer, Catherine Plaisant, Jessie Kennedy, Frank Van Ham, Nathalie Henry Riche, Chris Weaver, Bongshin Lee, Dominique Brodbeck, and Paolo Buono. Research Directions in Data Wrangling: Visualizations and Transformations for Usable and Credible Data. *Information Visualization*, 10(4):271–288, January 2011.

[KPHH11] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Wrangler: Interactive Visual Specification of Data Transformation Scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3363–3372, New York, NY, USA, 2011. ACM.

[MA14] Silvia Miksch and Wolfgang Aigner. Special Section on Visual Analytics: A Matter of Time: Applying a Data–Users–Tasks Design Triangle to Visual Analytics of Time-Oriented Data. *Computer & Graphics*, 38:286–290, February 2014.

[Mik10] Mike Loukides. What is Data Science? O'Reilly Media, June 2010. https://www.oreilly.com/ideas/what-is-data-science, Accessed: 2017-04-11.

[MRS08] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 1st edition, 2008.

[Mun09] Tamara Munzner. A Nested Model for Visualization Design and Validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, November 2009.

[New15] Sam Newman. *Building Microservices*. O'Reilly Media, 2015.

[Nie93] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann, 1993.

[Nor02] Donald A Norman. *The Design of Everyday Things*. Basic Books, 2002.

[Ras15] Rashid Khan. Timelion: The Time Series Composer for Kibana. Elastic, November 2015. https://www.elastic.co/blog/timelion-timeline, Accessed: 2017-04-11.

[RCC05] George G Robertson, Mary P Czerwinski, and John E Churchill. Visualization of Mappings Between Schemas. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 431–439, New York, NY, USA, 2005. ACM.

[RH01] Vijayshankar Raman and Joseph M Hellerstein. Potter's Wheel: An Interactive Data Cleaning System. *VLDB*, 01:381–390, 2001.

[TC05] James J Thomas and Kristin A Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, 2005.

[Tuf01] Edward R Tufte. *The Visual Display of Quantitative Information*. Graphics Pr, 2nd edition, 2001.

[WFH16] Ian H Witten, Eibe Frank, and Mark A Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 4th edition, 2016.