Free University of Bozen-Bolzano
Faculty of Computer Science

Interactive Visualization of Differences in Temperature Data Over Time

Tobias Bernard

Supervisor
Prof. Dr. Johann Gamper

Co-Supervisor
Dr. Anton Dignös

July, 2015

Abstract

Visualizing time series data over long periods of time in a concise way is a difficult problem. Time is linear and continuous, but displaying large amounts of continuous data at a reasonable resolution is impossible or impractical in most mediums. Traditional solutions employed in digital systems such as panning and zooming tend to require very granular interactions, causing unnecessary friction and making the process of viewing the visualization unwieldy.

To address this problem, I propose a display of small multiples organized in semantic groups (e.g. years, months, days). This provides a highly efficient overview of high-level patterns in the data. Additionally, one of the small multiples (e.g. one month) is shown in more detail at the bottom of the screen. The element to be shown in detail can be selected from the grid of small multiples interactively.

Another challenge associated with displaying time series data across long time periods is allowing for comparisons between different time intervals. When time series data is not displayed continously, there is no common coordinate system which aids visual comparisons. Since not all of the data can be shown at the same time, comparisions across space are impossible for many of the items.

My solution allows for the user to select one of the small multiples as the basis of comparison. This element's data is then overlayed onto all other elements, enabling efficient visual comparisons of elements across the entire dataset.

To show the viability of my solution, I have implemented it as a client-side web application.

Abstract (Italiano)

Visualizzare serie temporali lungo un periodo di tempo prolungato in maniera concisa è un problema complicato. Il tempo è lineare e continuo, ma mostrare un flusso costante e consistente di informazioni ad una risoluzione ragionevole è impossibile o comunque poco pratico nella maggior parte dei casi. Panning e zooming, soluzioni adottate tradizionalmente in sistemi digitali, causano delle difficoltà nell'utilizzo e richiedono interazioni molto granulari, rendendo il processo di visualizzazione complicato.

Per risolvere questo problema, propongo un'interfaccia costituito da piccoli multipli organizzati in gruppi semantici (per esempio anni, mesi, giorni). In questo modo è possibile ottenere una rappresentazione nei dati ad alto livello e molto efficiente. Inoltre, uno dei piccoli multipli (per esempio un mese) viene mostrato in maniera più dettagliata in basso nello schermo. L'elemento che si vuole mostrare in dettaglio puo' essere selezionato in maniera interattiva dalla griglia dei piccoli multipli.

Un'altra difficoltà associata allo illustrare serie temporali lungo periodi di tempo prolungati riguarda come permettere di confrontare diversi intervalli di tempo. Quando una serie temporale non viene disegnata in maniera continuativa, viene a mancare un sistema di coordinate che aiuta il confronto visivo. Poichè è impossibile mostrare tutti i dati allo stesso tempo, il confronto spaziale è impossible in molti casi.

La mia soluzione permette all'utente di selezionare uno di questi piccoli multipli come elemento base per il confronto. I dettagli dell'elemento sono successivamente sovrapposti a tutti gli altri, permettendo in questo modo un confronto efficace di elementi in tutto il dataset.

Per dimostrare la fattibilità della mia soluzione, è stata implementata una applicazione web client-side.

Abstract (Deutsch)

Zeitreihendaten über lange Zeiträume zu visualisieren ist ein schwieriges Problem. Zeit ist linear, aber eine ununterbrochene Visualisierung linearer Daten bei erfordert ein Medium von variabler Größe. Dieses Problem wird in digitalen Visualisierungen traditionell durch sehr granulare Navigation durch den Benutzer, z.B. Zoomen und Scrollen, gelöst. Dies hat allerdings den Nachteil, dass dabei für das bloße Betrachten der Visualisierung beträchtlicher kognitiver und physischer Aufwand nötig ist.

Um dieses Problem zu lösen, schlage ich ein Layout, welches aus einem Raster von Small Multiples besteht vor, welche die Zeit in semantische Abschnitte (z.B. Tage, Monate, Jahre) aufteilen. Dies gibt dem Betrachter einen schnellen und effizienten Überblick über große Teile des Datensets. Zusätzlich wird eines der Small Multiples (z.B. ein Monat) im unteren Teil der Oberfläche größer angezeigt. Das größer anzuzeigende Element kann interaktiv aus dem Raster ausgewählt werden.

Eine weitere Herausforderung bei der Darstellung von Zeitreihendaten über lange Zeiträume ist es, Vergleiche zwischen verschiedenene Zeitbereichen zu ermöglichen. Dies ist insbesondere schwierig wenn Zeitreihendaten nicht durchgehend dargestellt werden, weil es dabei kein durchgehendes Koordinatensystem gibt, welches visuelle Vergleiche einfach möglich macht.

Meine Lösung erlaubt es dem Betrachter, eines der Small Multiples aus dem Raster als "Vergleichsbasis" zu setzen. Die Daten für diesen Monat werden dann über die Daten aller Monate gelegt. Damit sind schnelle visuelle Vergleiche über das gesamte Datenset möglich.

Um die Praktizität meiner Lösung zu zeigen, habe ich eine Client-side Webapplikation entwickelt, die sie implementiert.

Contents

1 *Introduction* **2**

1.1 Motivation 2

1.2 Problem 3

1.3 Solution 3

1.4 Organization 4

2 *Background* **5**

2.1 Information Design Concepts 5

2.1.1 Time Series 5

2.1.2 Small Multiples 6

2.2 User Interaction 7

3 *Problem Definition* **9**

3.1 Problem Domain 9

3.2 Making Time Tangible 10

3.3 Enabling Visual Comparisons 11

4 *Design* **12**

4.1 Prototypes 12

4.1.1 "Time Blocks" 12

4.1.2 "Diff Areas" 14

4.1.3 "Micro-Macro" 15

4.2 Final Design 17

4.2.1 Example Use Case 18

5 *Technology and Implementation* **19**

5.1 Data 19

5.2 Prototypes 21

5.3 Final Design 22

6 *Feedback and Evaluation* **24**

7 *Conclusion and Future Work* **25**

Bibliography **26**

Chapter 1

Introduction

1.1 Motivation

Information design is similar to writing in many ways. It is about communicating information to people in an understandable way. Just like with writing, there are several factors involved: The accuracy and granularity of the information, the density and efficiency with which it is conveyed, the medium used to convey it, the reader's knowledge of the subject, and the context in which it will be read.

But unlike writing, Information Design is visual. While a written text has to be more or less read linearly, a visualization can communicate complex ideas in a much more condensed form. This is especially true when working with datasets so large that it is impossible to make sense of them by going through them in a linear fashion.

Today, most professions require people to make sense of data in one way or another. Unfortunately, the tools for doing so are not very good in many cases [**?**]. This thesis explores new approaches for designing these tools. In particular, I focused on interfaces for displaying time series data over long periods of time. The data I worked with is temperature data between 2007 and 2014, measured with a frequency of a few minutes at agricultural weather stations across the province of Bolzano. It was provided by *Südtiroler Beratungsring für Obst- und Weinbau*, an agricultural consultancy, who use this data to help local farmers with things like irrigation and pest control.

In this industry, making sense of the large volumes of data provided by the weather stations is crucial, because it is what drives the consulting activities. If the data analysis tools the consultancy has at their disposal makes this easier, the quality of their entire service improves.

1.2 Problem

One of the hardest problems in visualizing these kinds of datasets is enabling viewers to see patterns and make comparisons across long periods of time. Since time is linear and continous, the most intuitive method of displaying it is as a single, continous timeline. However, this is possible only if the dataset is small enough to allow for displaying the entire dataset at a reasonable resolution on the given medium, or if the medium can dynamically expand to match the size of the dataset. In the case of visualizations of large datasets on screens, neither is the case. Thus, it is not possible to show all of the data at once, and some kind of navigation is necessary. The most common approach to this is to allow users to interactively zoom and scroll a continuois timeline. However, this type of interface has the disadvantage that the simple task of *viewing* the data requires a large amount of very granular interactions, and therefore considerable effort, on the part of the users.

My thesis explores new approaches to the design and implementation of interfaces for time series data that facilitate comparisons while minimizing user interaction. Specifically, the two issues I addressed are *making time tangible*, and *enabling visual comparisons across large datasets*.

1.3 Solution

To address the issue of making time tangible, my interface employs a grid of small multiples representing discrete, semantic time intervals (years in my specific case, though the solution works just as well for other interval lengths, like days or weeks). Splitting up the continuous flow of time has the advantage that individual elements are easy to distinguish, thus facilitating comparisons across space. More importantly, however, it gives users a very clear sense of the structure of time.

This also helps with the second goal, enabling comparisons. Since time is already split up into discrete intervals, these intervals can be leveraged to allow for powerful visual comparisons across the entire dataset. This is achieved by overlaying the month that needs to be compared on every other month, and visualizing the difference between them with colored areas. Complementing this high-level overview of the entire dataset, there is a second view which enables users to see specific parts of the dataset in great detail, as Figure 1.1 shows.

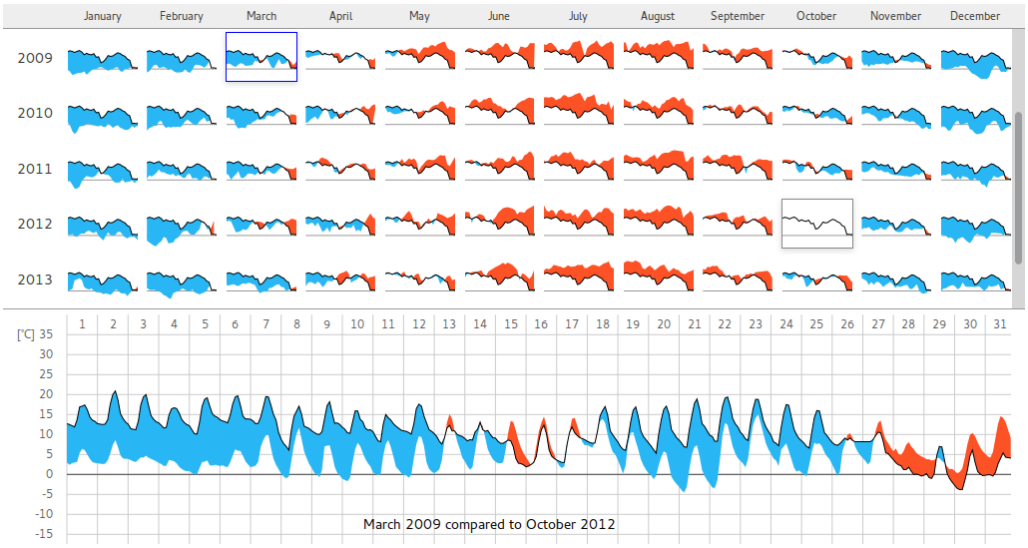

Figure 1.1: Screenshot of the user interface, a grid of small multiples and a detail view which shows a single month in more detail.

1.4 Organization

Chapter 2 outlines some of the theoretical ideas that inpsired the design of my solution, along with some general information on basic information design concepts. *Chapter 3* explains the domain and context of the problems my research aims to address. *Chapter 4* illustrates the design process in detail, showing ideas and prototypes at different stages, as well as how they informed the final design. *Chapter 5* is concerned with the implementation of both the prototypes and final design. It explains different technical choices and considerations. *Chapter 6* sums up the the feedback I got from the experts at the *Beratungsring* meeting. *Chapter 7* concludes the thesis with some general considerations and ideas for possible future work on the project.

Chapter 2

Background

In this chapter I will outline some of the concepts and ideas that have influenced the design and implementation of the project.

2.1 Information Design Concepts

2.1.1 Time Series

A *time series* [**?**] is a set of data points laid out across a period of time, mapping the changes in a variable across said period. A *time series plot* is an information graphic mapping these values to visual variables, usually the horizontal and vertical position of points or lines on a 2-dimensional display. Time series plots are among the most frequently used type of data graphics[**?**], likely because of the simplicity and clarity of time as a single, linear dimension. Edward Tufte describes their appeal in *The Visual Display of Quantitative Information* [**?**]:

> "With one dimension marching along to the regular rhythm of seconds, minutes, hours, days, weeks, months, years, centuries, or millennia, the natural ordering of the time scale gives this design a strength and efficiency of interpretation found in no other graphic arrangement."

Time series plots can take various different forms and are used across variety of different fields and mediums. For example, William Playfair's trade-balance chart (Figure 2.1 [?]), published in *The Commercial Political Atlas* in 1786 shows the difference in England's imports and exports from and to Denmark and Norway over 80 years. As is immediately visible, the exports surpassed the imports around 1755, and went on to increase steadily in the following 25 years. This display, along with Playfair's other work is renowned because it was one of the first examples of statistical graphics being used in such an elegant way to explain complex relationships in a large dataset.

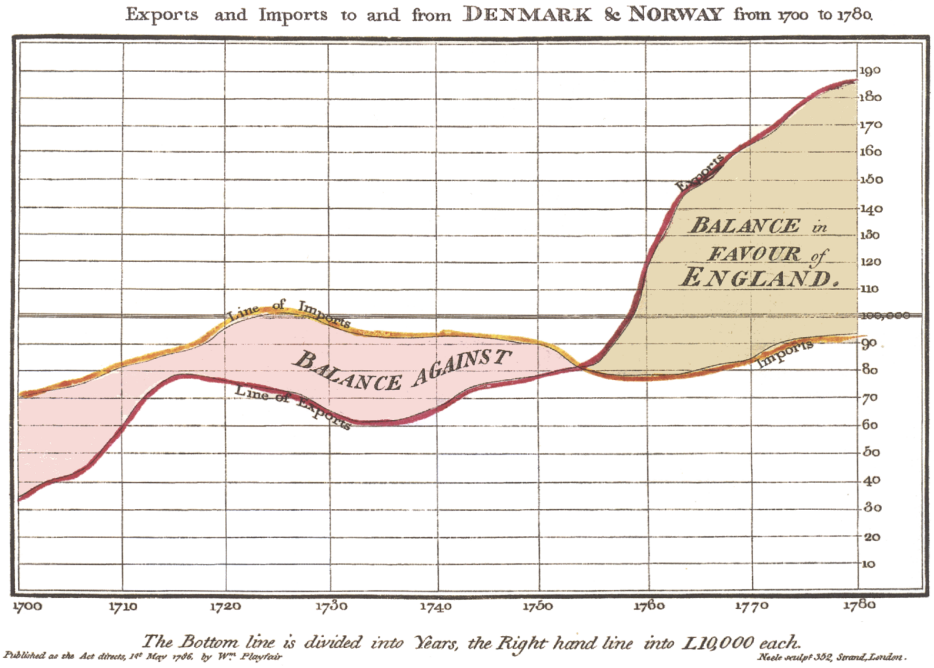

Figure 2.1: Trade-balance time-series chart by William Playfair (1786)

2.1.2 Small Multiples

Small Multiples [?] are series of graphs row or grid, which use the same system of coordinates and show different aspects of a signgle variable. This kind of layout enables powerful comparisons across space, which are almost always preferable to comparisons across time because they are faster and require no user interaction [?].

This data graphic from the New York Times' *How the Recession Reshaped the Economy, in 255 Charts* [**?**], comparing the development of different industries after the financial crisis, is an example of the flexibility and wide applicability of the format.

Figure 2.2: Interactive small multiples visualization on the New York Times website.

The term and concept of small multiples was popularized by Edward Tufte. They are a direct application of his *Shrink Principle* [**?**], which states

> *"Graphics can be shrunk way down."*

Tufte criticizes the fact that most data graphics are not taking full advantange of the resolution at which the human eye can distinguish visible data points. He argues that graphics should be as dense as possible, within the limits of human vision, in order to give viewers as clear a picture of the data as possible.

2.2 User Interaction

User interaction can be an essential part of digital visualizations allowing the a graphic to show more data than the medium can display at a time, and giving the user the option to explore the dataset in more detail. However, interaction can also be harmful, as it can be used to mask bad design decisions in other areas, like the layout of the graphic.

For example, an interactive visualization might hide information about the items of a list in popups and require a user to click each one to make comparisons between the elements. However, if the layout of the list were to already show this information for all elements, comparisons could be made without any interaction.

In *Magic Ink*, Bret Victor divides software into three categories [**?**]: Information software, manipulation software and communication software. *Information software* gives users information and empowers them to "ask and answer questions, make comparisons, and draw conclusions", *manipulation software* allows users to create things, while *communication software* enables them to communicate with other users.

When it comes to interaction, he argues that these categories have to be treated differently, because the use cases for them are very different.

> *"For manipulation software, interaction is perfectly suitable: the user views a visual representation of the model, considers what to manipulate next, and performs a manipulation."*

Information software, on the other hand, *"mimics the experience of reading, not working"*, which is why with information software *"all interaction is essentially navigation around a data space"*. He goes on to explain the considerable cognitive and physical effort [**?**] involved with navigation, explaining that it is really the task of actively recreating the user's mental model in the computer.

Because of this, interaction in information software should be avoided whenever possible. In other words, information software should show as much information as possible without the user having to interact with it, allowing them to focus on *seeing*, not doing.

Chapter 3

Problem Definition

Designing visualizations that are dynamically generated and work with different kinds of datasets, but are still easy to read and allow for efficient reasoning and fast comparisons is a difficult balance to keep.

When it comes to time series data across long periods of time, the difficulty of making a display scalable is further amplified. Time is linear and continuous, but in most cases the medium it is displayed on has a fixed physical size that cannot change with the data displayed. This means that if the graphic is to retain a reasonable resolution, not all of the data can be visible all of the time. This means that we can not avoid introducing some kind of interactive navigation. However, the consequences go further than that: It also leads to a series of problems related to the actual process of viewing. For example, how can elements be compared visually if some of the elements are not currently in view? When balancing all of these different constraints, it is very important to keep in mind what the specific use cases and goals are, and to optimize the display for these.

3.1 Problem Domain

The dataset I worked with is sensor data from agricultural weather stations across the province of Bolzano. It was provided by *Südtiroler Beratungsring für Obst- und Weinbau*, a local consultancy in the agricultural sector, who use it to help their clients, mostly apple and wine farmers from the region. The data consists of 24 metrics, which are measured every few minutes at each

of the 134 stations. These metrics include things like temperature, humidity, precipitation and wind speed.

The data collected by the sensors at the weather stations is at the very core of *Beratungsring*'s business, as all of their consulting activities rely on it heavily. Using this data, they decide when to advise their clients to use pest control, irrigate their plants or a number of other things. This makes it crucial to have an interface to the data which makes it easy to understand the underlying structure, find patterns and make comparisons.

Since *Beratungsring* is currently planning a redesign of their internal information system, they asked me to explore some interesting concepts around the display of time series in the context of their dataset. Because of the very wide applicability and relevance to their specific use cases, we decided that my research would focus on *making time tangible* and *enabling visual comparisons across large datasets*.

3.2 Making Time Tangible

One of the problems with most interactive visualization tools for time series data is that they display the data in a single, continuous chart with time on the x-axis, which can be zoomed and scrolled horizontally. An example is shown in Figure 3.1 [1]. This specific chart allows users to zoom by setting the new time interval via click and drag. To zoom out, one needs to click a button in the top right.

This type of interface requires a large amount of interaction on the user's part to explore, which makes it feel clumsy and complicated. Significant amounts of physical and cognitive effort are necessary to simply view the display [**?**]. It also makes time feel arbitrary and immaterial, as elements change their size and position with very little context.

The first major goal of my thesis was finding a way to avoid these kinds of problems when displaying time series data.

[1] Source: fusioncharts.com

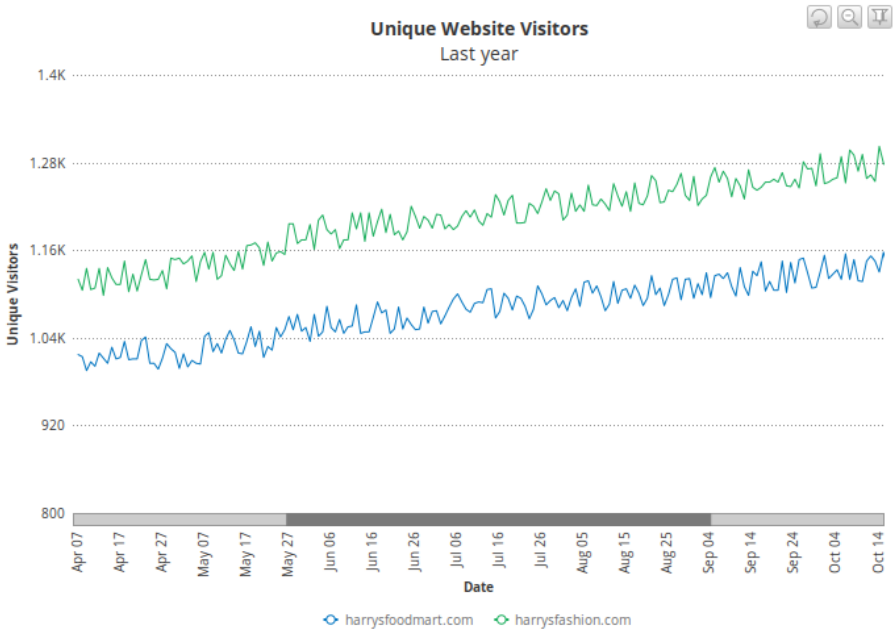

Figure 3.1: Example of an interactive time series plot

3.3 Enabling Visual Comparisons

The other problem I investigated is comparing data between different time intervals, especially in large datasets. In the case of temperatures, specifically, there are very clear patterns in the data, (e.g. winter months are generally colder than summer months, nights are generally colder than days), but comparing these intervals is a very tedious task with conventional visualization methods. Viewers have to go back and forth between the different time periods they want to compare, which is especially tedious in large datasets.

For example, if I wanted to find the differences between the January of 2012 and 2014 in a continous, zooming timeline, I would have to zoom in and out on the graphic several times, remembering the values in certain positions, and comparing them myself. Not only is this slow and inefficient, but the comparison will probably not be very accurate.

Addressing use cases like this was the second area of focus in my research.

Chapter 4

Design

Since the research questions my project addresses are very open-ended, exploration and prototyping were a crucial part of the design process. I ended up building 8 fully functioning prototypes, which explore different approaches to solving the intial problems. By combining the best elements from these experiments, I arrived at a solution which achieves the project's goals in a simple and efficient way.

4.1 Prototypes

4.1.1 "Time Blocks"

In order to address the problem of time feeling arbitrary and immaterial, which is common in continuous zoomable timelines, I experimented with splitting it up into discrete intervals. These 'time blocks', a grid of small multiples, break up the endless continuity of time into small, semantic blocks (e.g. days, weeks, months).

An example with days as the base unit is shown in Figure 4.1. It is a simple week calendar layout of small multiples, where each row corresponds to a week, and each day corresponds to a chart visualizing the temperatures on that day. The charts all use the same scale and coordinate system, making it easy to visually compare the plots.

This approach has several advantages over displaying time continuoisly in a timeline. First of all, it makes navigating time much easier, because rows can be stacked vertically, thus allowing navigation to be as simple as scrolling a website. More importantly, however, it also makes time tangible for the user by splitting it into easily recognizable units of known size, using a common coordinate system.

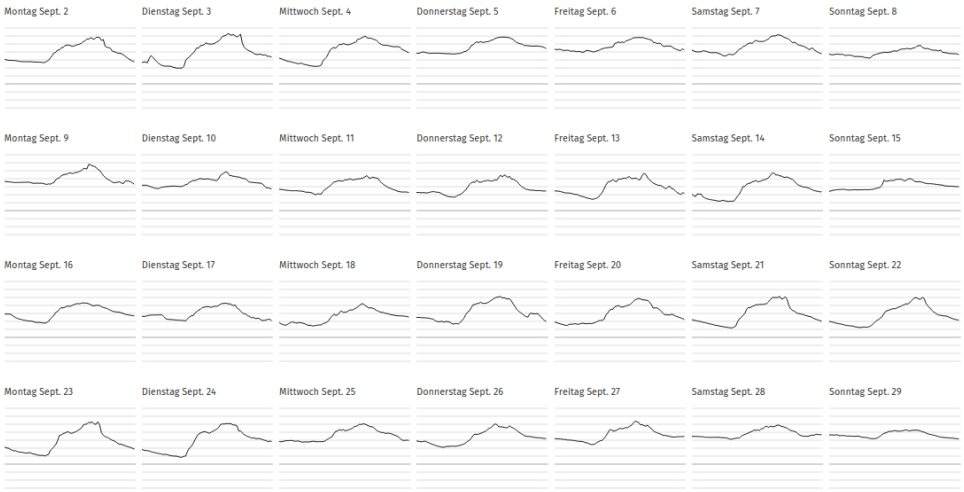

Figure 4.1: One of the first prototypes implementing the 'time blocks' concept

4.1.2 "Diff Areas"

To address the problem of enabling efficient comparisons across long time periods, I explored visualizing the difference in the temperature values between periods of time *on* the actual graphic, thus making comparisons immediate.

This fit in perfectly with the 'time blocks' concepts, because with the data already split into discrete intervals of equal size, it was easy to make them comparable by just showing the diff between two graphs in the time blocks. To visualize these diffs between the different periods the plot for the selected interval is overlaid on every other interval, and colored areas are drawn between the two plots. If the temperature values in an area are lower than on the selected day, it is drawn in blue, otherwise it is drawn in red. Figure 4.2 shows an example of this. Every individual chart can be clicked to set it as the new comparison basis against which the diffs are calculated. Though the lack of affordances makes it hard to see at a glance, the selected chart is the one with no colored areas (September 26).

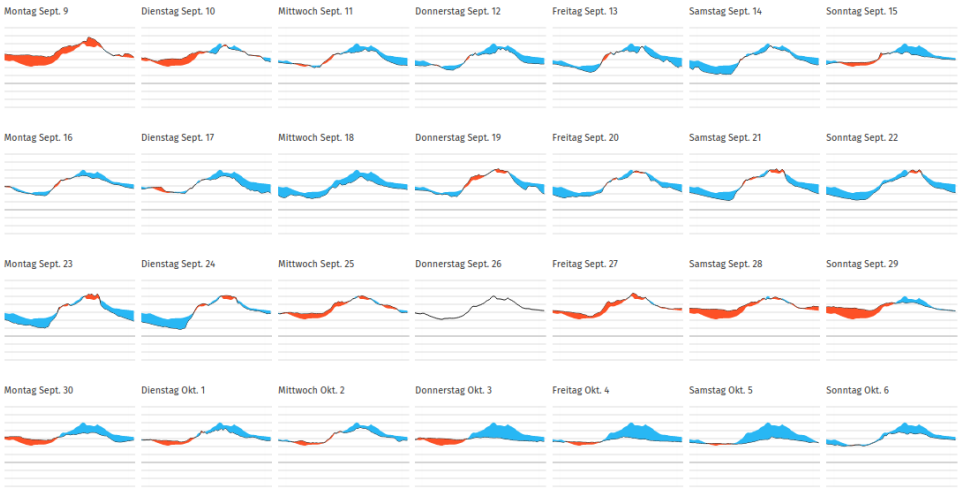

Figure 4.2: Early prototype with 'diff areas'

The bigger the difference in temperature between the compared intervals, the bigger and therefore more visible the colored areas. This makes outliers or uncommon patterns in the data instantly stand out. Inversely, to find the most similar intervals, one simply needs to look for the intervals with the smallest colored areas.

4.1.3 "Micro-Macro"

In trying to get the greatest possible information density from the display, I experimented with different sizes of small multiples. I found that Tufte's *Shrink Principle* really works. The most important trends and patterns in the data are still perfectly recognizable even at miniscule sizes.

For example, the design in Figure 4.3 shows almost the entire dataset at a glance. The base unit of display and comparison are months, allowing the viewer to compare entire years worth of data very efficiently. Since each row is a single year, the vertical dimension allows for semantic comparisons as well. Where other layouts which use weeks and months as rows do not have an inherent periodicity to them (for example, it isn't very meaningful to compare the temperature on Mondays over time), this display enables the viewer to compare the same month across different years vertically. Due to the periodic nature of weather across the year, this can lead to interesting insights on the seasons and large-scale changes in weather and climate over the years.

Figure 4.3: Prototype employing 'diff areas', comparing months across the entire dataset

Small multiples are highly effective in showing trends across large amounts of data, as well as enabling comparisons between intervals. The high-level overview this provides is very powerful for quickly searching the dataset for specific patterns or finding extraordinary events.

However, when it comes to showing the exact numeric values of the data at specific points in time, this type of display is not very useful, precisely because of its high-level nature. Though this aspect was the main focus of my research, the exploratory nature of my application made it necessary to provide a way to see the data at this kind of level as well.

In order to provide this high-detail view of the data as well, I experimented with a second view, which shows one of the small multiples much larger and in very fine detail. Figure 4.4 shows an example of this. On the left side, there is a very small, high-level overview of several months of data, split up into individual days. The day to be shown in detail can be interactively selected from the grid of small multiples.

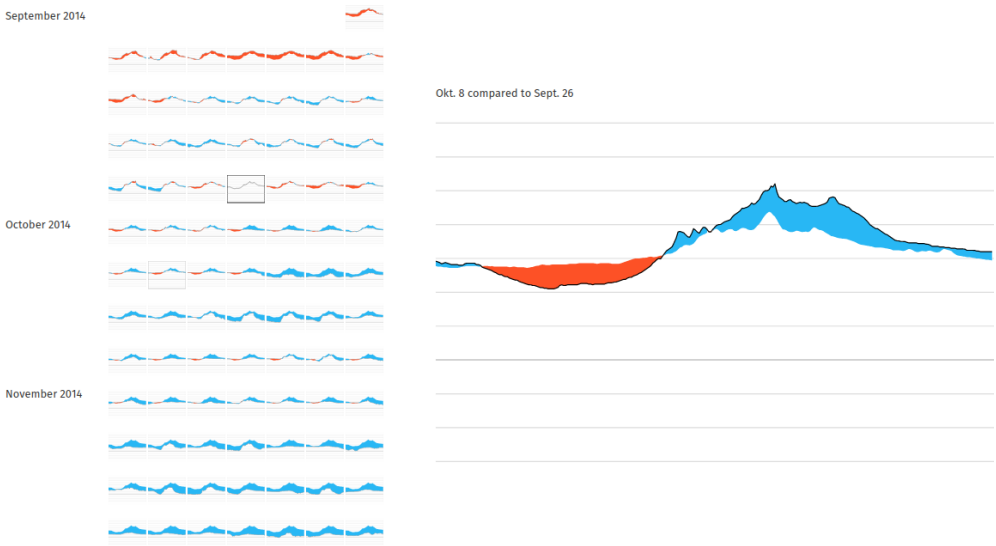

Figure 4.4: Prototype combining the micro-macro layout with 'diff areas'.

4.2 Final Design

Of all my experiments, the combination of the micro-macro layout with the 'diff area' technique is clearly the most compelling. It meets all of the initial research goals, allowing viewers to efficiently compare data at a high level of detail over extended periods of time, all while requiring very little interaction on their part.

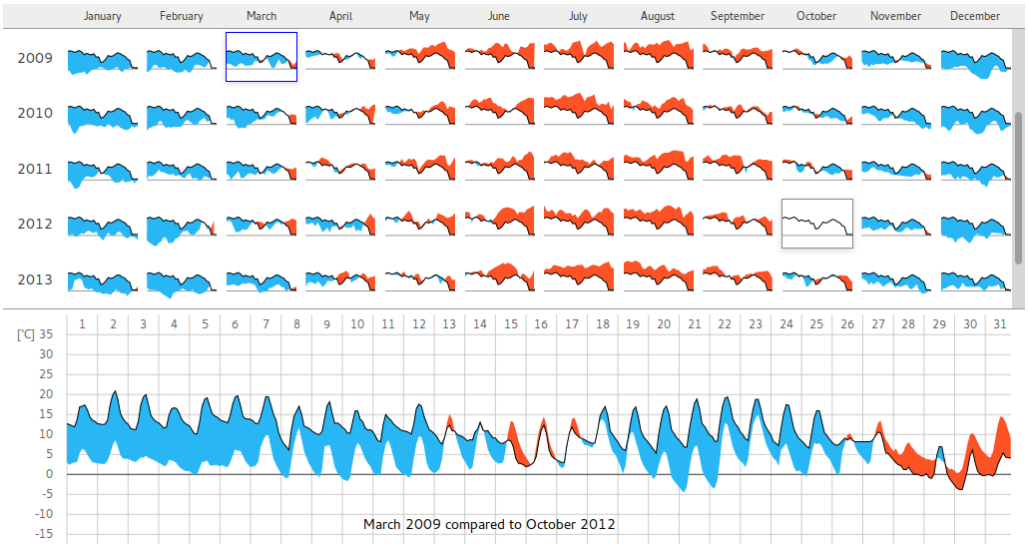

Figure 4.5: Screenshot of the final layout

The final layout (Figure 4.5) uses months as the base unit of time, allowing viewers to compare both months and years aross the entire dataset. The data in the small multiples is aggregated to 24 hour intervals, which means that the day-night fluctuations in the dataset do not distract the viewer from seeing more long-term trends and patterns in the data.

The detail view on the bottom of the display is fixed, while the grid view of small multiples can be scrolled vertically. Left-clicking the small multiples opens them in the detail view, right-clicking them sets them as the new basis for comparison. This works with the year labels as well, allowing comparisons of all months on a year-by-year basis. Rather than comparing every month to the same month of the same year, each month is compared is compared to itself in a different year.

4.2.1 Example Use Case

Let us assume a consultant at *Südtiroler Beratungsring für Obst- und Weinbau* is using the application because he needs to find months with temperatures similar to March of 2012. All they have to do is find March of 2012 in the grid of small multiples, and right-click it. This will re-draw all the "diff areas" across the interface to show how different the temperatures during each month are from the selected month. To find the most similar months our user just needs to find the months where the colored areas are smallest. In this case it is apparent that there are no months that are highly similar, because there are relatively large colored areas in all charts.

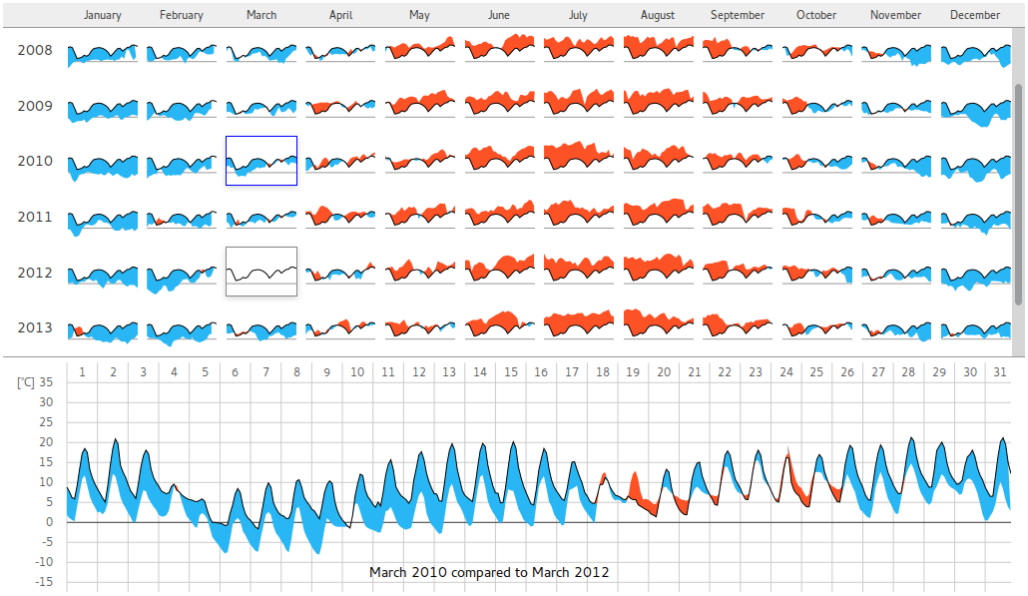

Figure 4.6: Interface with March of 2012 as the comparison basis and March of 2010 shown in the detail view.

What is also visible immediately is that March was much colder in other years compared to 2012. Let us assume our consultant is interested in how that difference is for March of 2010. All they have to do now is left-click the chart for March of 2010 and it will be displayed in the detail view on the bottom of the screen, as Figure 4.6 shows. This view shows the data at a much finer level of detail, allowing the consultant to compare even individual days.

As this example has shown, the application's interface makes it fast and efficient to see differences and patterns in large datasets and provides ample opportunities for exploration.

Chapter 5

Technology and Implementation

For the implementation of both the prototypes and the final design, I used web technologies, namely HTML and CSS for layout and interface elements, SVG for the charts, and Javascript for the application logic. It was an easy decision as no other development environment is so widely adopted and has such a broad range of available technologies and tools so perfectly suitable for data visualization.

5.1 Data

The data I worked with was stored in a CSV file where each row contains the ID number of the station, a timestamp, and the temperature value in degree Celsius, with an interval of 5 minutes between the measurements. Each row corresponds to one measurement.

station id	timestamp	value
3	2007-01-01 00:05:00+01	-4.708
3	2007-01-01 00:10:00+01	-4.858
3	2007-01-01 00:15:00+01	-5.008

In order to use this data in the browser, I had to filter and aggregate the data, and store it in a format the browser can read natively, namely JSON. To do this, I used two scripts, a bash script which filters out only one specific station

from the entire dataset (as only one will ever be displayed at a time), and a node.js script which aggregates the data at different levels of granularity and creates a nested object structure for use in the visualization.

This is the basic structure of the data in its JSON representation:

```
{
  timestamps: [
    '1167649500','1167735900','1167822300', ...
  ],
  values: [
    -5.9,-4.1,-1.4, ...
  ]
}
```

As this example shows, the JSON data used in the browser is much more condensed than its initial representation. In order to minimize the memory footprint, the timestamps are represented in the Unix timestamp format. Temperature values are rounded to one decimal point for the same reason.

For the small multiples, the data is aggregated to a single value per day, both in order to keep the amount of data to be displayed manageable for the browser [**?**], filter out the day-to-day fluctuations of the data and show high-level trends and patterns more clearly. This is done by calculating the average temperature across the entire day (between 00:00 and 24:00), and using the middle of the time interval (noon in this case) as the timestamp for this value in the dataset.

For the detail view, the data is aggregated as well, but to a much smaller degree and purely due to memory and performance considerations. Since the display will only ever take up about 1000px horizontally, it makes no sense to draw more than 1000 data points on standard pixel density monitors. The original dataset contains 288 data points per day, which means the data points per month are between 8064 for a month with 28 days and 8928 for a month with 31 days. This means that on screens with standard pixel density, the aggregation is only visible when the intervals are longer than 3 hours. With this in mind, I set the aggregation interval to 2 hours in order to provide some extra resolution for HiDPI displays.

5.2 Prototypes

For the interface prototypes I built, I needed a process and architecture that could be easily used to build many different kinds of visualizations and that allowed for rapid iteration. This is why, even though view libraries like React provide a much better workflow and produce cleaner code, I decided to use vanilla HTML, CSS and Javascript. This means that I had to do templating, event handling, and view updates manually, but it also meant more flexibility in adding, changing and removing individual parts without affecting the rest of the system.

All the various prototypes have a few things in common, most importantly that the layout structure is completely defined in CSS, and every chart is a self-contained SVG element within the DOM. This approach affords a lot more flexibility in adapting layout to different concepts than, for example, making the entire grid a single SVG with absolutely positioned elements inside of it. This allowed for quick iteration on the layout and made it easy to swap out and change different parts of the interface. Most of the interface is implemented in Flexbox, a cutting-edge CSS layout technology, which allows CSS to define rows or columns with and without wrapping.

Figure 5.1 shows one of the "year-diffs" prototypes, comparing both months and years on a very high level, running in the browser.

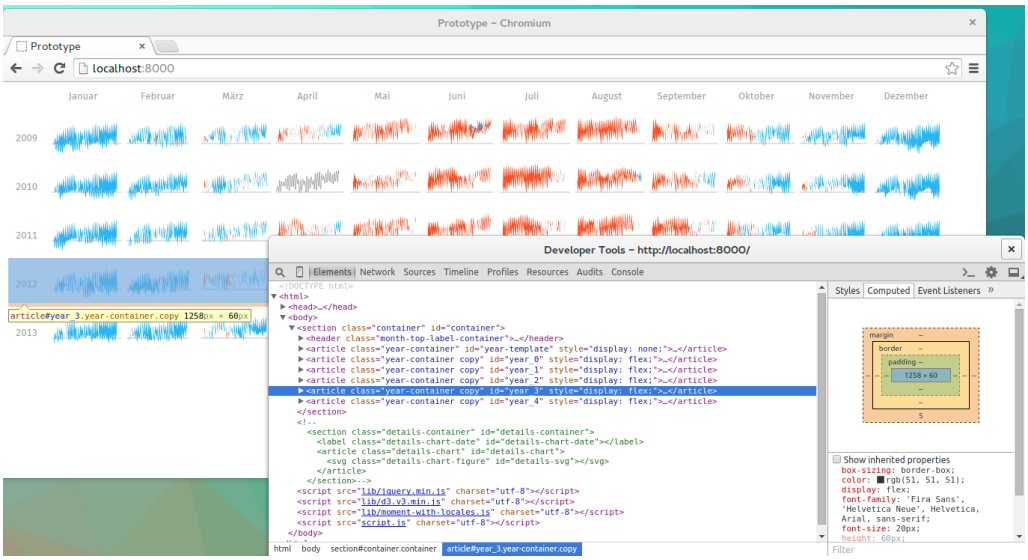

Figure 5.1: Prototype running in Chromium

5.3 Final Design

For the implementation of the final design, I used React for rendering the entire interface using a clean component structure. Since there was no need for the architecture to be easily adaptible to different types of visualizations, it made sense to use React, as its virtual DOM makes it possible to handle rendering with a single view hierarchy using a one-directional data flow. This is a much better paradigm for developing interfaces, because synchronizing state between the DOM and data model, typically one of the hardest problems when doing this by hand, is not an issue with React. Every change in the data automatically permeates to the DOM, but only the parts of the DOM that actually need to be updated are changed, making updates very performant. Figure 5.2 shows part of the DOM rendered by React in my application.

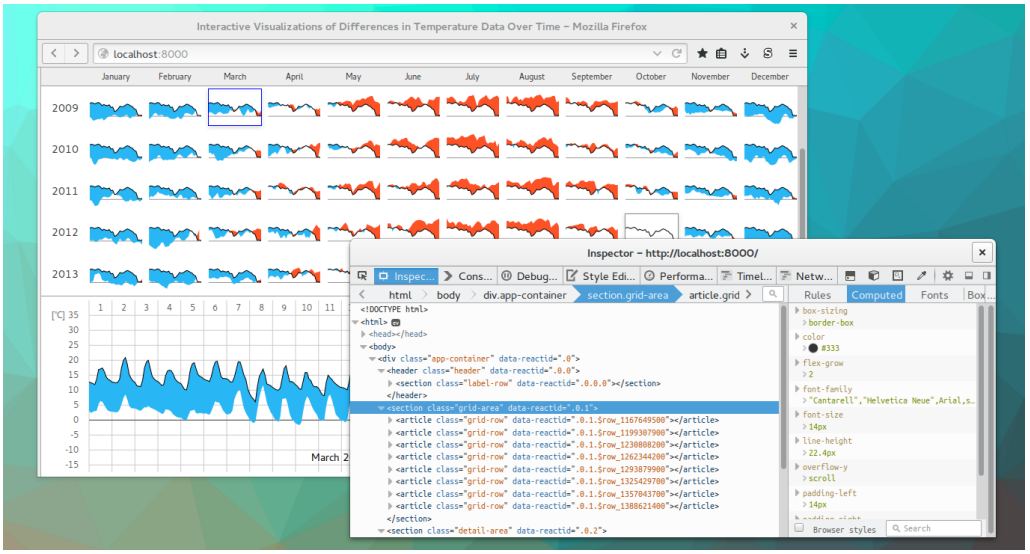

Figure 5.2: The final implementation, running in Firefox.

The view strucutre consists of a single hierarchy with a parent component called *App* at the top. This component handles the initial data retrieval and all consecutive view updates. *App* renders the three main areas of the interface, *LabelRow* (the row of month labels at the top), *GridView* (the scrollable area containing the small multiples) and *DetailArea*, the view at the bottom of the screen showing a single month in detail. The basic React component structure is visible in the React Debugger in Figure 5.3. This browser extension makes it possible to inspect the React component structure, as well as the properties and state of individual components at runtime.

GridView then renders a list of *GridRow* components, one for each year. Every *GridRow*, in turn, renders a list of *GridChart* components, which draw an SVG chart with the data passed to them.

Every *GridChart* has its own event listeners for both left and right clicks, which always trigger a complete re-render of the *App* component at the top of the hierarchy with the data from the component the event came from. The new data then cascades through the component hierarchy, updating every part of the interface accordingly.

Figure 5.3: The React Debugger extension in Chromium.

Chapter 6

Feedback and Evaluation

I presented my ideas and prototypes at a meeting attended by *Südtiroler Beratungsring für Obst- und Weinbau* consultants, as well as agricultural experts from other institutions. Their response was highly positive, as there are currently no tools available for their industry which enable comparisons across time at the level that my proposed solution achieves. They explained that their current process for doing comparisons of this kind involves going through the data and comparing values manually. A more comprehensive tool employing the techniques used in my design (using all variables, not just temperature) could massively improve their workflow in many areas.

They also had some suggestions and ideas for future iterations of the application, some of which I was able to implement in the final version of the project. For example, they proposed aggregating the data more, as the day-to-day variance in the dataset can make it harder to see larger trends in the small multiple overview of the entire dataset. In the implementation of the final design I aggregated values in intervals of 24 hours, cutting down on the noise and making the resulting display more useful in the process.

Another interesting proposal from one of the experts at the meeting was an interface where different years are compared, but where the display is aligned by specific events during the year (e.g. the first day with temperatures below 0 degrees Celsius). This would be useful because for certain kinds of analysis the exact dates are not as important as the time relative to a specific event that year. Though it is too specific for the purposes of my project, this kind of interface taylored to a specific use case would be an interesting starting point for further research.

Chapter 7

Conclusion and Future Work

Because of the inherent limits in human vision and display technology, a visualization is never perfect. However, given a specific set of goals and use cases, some visualizations can be vastly more efficient than others.

Starting from my problem definition, I have designed and developed a display which does exactly this. By employing a grid of small multiples showing high-level patterns in large datasets, combined with a detail view showing parts of this data at a macro perspective, my solution gives both an efficient overview and allows for detailed study of the underlying data.

Making accurate comparisons between individual time intervals in time series, an otherwise notoriously hard task, is reduced to a single click for the user. By combining traditional information design concepts with the interactive capabilities of the web as a medium, this approach provides a a simple and efficient solution to a complex problem.

Though the intial research questions are thoroughly addressed, there are a lot interesting ideas related to the display and comparison of time series data over long time periods that could be explored. In our meeting with the agricultural experts at *Beratungsring* we discussed a number of other areas in which a visualization format specifically optimized for a certain use case could massively improve the workflow currently used by their consultants.

This is a clear signal that domain-specific visualization tools have a lot of potential in improving the everyday jobs and lives of people everywhere. My experiments have shown that even in areas as traditional and conceptually

simple as time series plots, there is always room for improvement over the standard way of doing things.

With the ever-increasing number of sensors and metrics in every area of our lives, making sense of large amounts of data quickly and efficiently will continue to become more important in the future. Custom, domain-spcific visulizations will play a crucial role in making this possible.