Time Series Visualization

Group 3
Jannik Hildebrandt, Michelle Perkonigg, Patrick Steyer

Graz University of Technology
A-8010 Graz, Austria

29 Jan 2019

Abstract

Time is a variable which has a specific characterization. Therefore it can not be treated the same as other measurable variables, such as temperature. Visualization is a good way to represent the measured data points which form a time-series. To create a good and meaningful visualization three questions must be answered in the following order. What data will be visualized and how is it characterized? Why will this data be visualized and what user task should be fulfilled with this visualization? How to visualize this data properly considering the two questions before. On the basis of these three questions a visualization of a time-series can be created to present the data as good as possible for the specific user task. This user task can be the process of an analysis which ranges from finding patterns over curve fitting to forecasting. The question of how refers to all the different customization possibilities of a chart. Through a carefully chosen customization of a chart the focus can be shifted to a specific part or property of the data. This customization includes important aspects, such as the arrangement, the scope, the type and the scale of the data. Furthermore a variety of different online, as well as offline tools and libraries exist to create charts.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

As one of the four dimensions of our world, the need to visualize time automatically becomes a very important topic in data visualization, but why is there even a difference to visualizing other parameters?

1.1 Characteristics of Time

Time has a special character to consider. It follows some rules which do not necessarily apply to other data sets. [*9 Characteristics of Time* 2019]

- Time Is Involuntary

  Unlike the three space dimensions, where it is possible to just stand in one place for a while, it is not possible to stand at one point of time for a while. There is no possibility to stop time.

- Time Is Irreversible

  Time is a sequence of events, happening one after another. The future is influenced by the past, but not the other way around. What has happened can not be undone.

- Time Is Required

  Nothing can happen without time. Time has to progress in order for anything to happen.

- Time Is Measurable

  There are different units to measure time, such as seconds, minutes, hours, years, ... Important is only that it is possible to measure time, as known from the watches and clocks.

- Time Is Absolute

  Einstein told us that time is relative and strongly depends on the speed of movement. When looking at time on the earth, however, it can be said that time is absolute, which means it is the same whether it is measured in Europe or in America, at the South Pole or at the North Pole.

1.2 Visualize Time

There are different approaches how to transform time into a visual representation. [Aigner et al. 2011]

1.2.1 Map Time to Time

This is the most intuitive way to visualize time. It is a dynamic representation where the time in the animation is correspondent to the time which was measured in real life. Sadly it is not always possible to display dynamic content, so a static representation is needed.

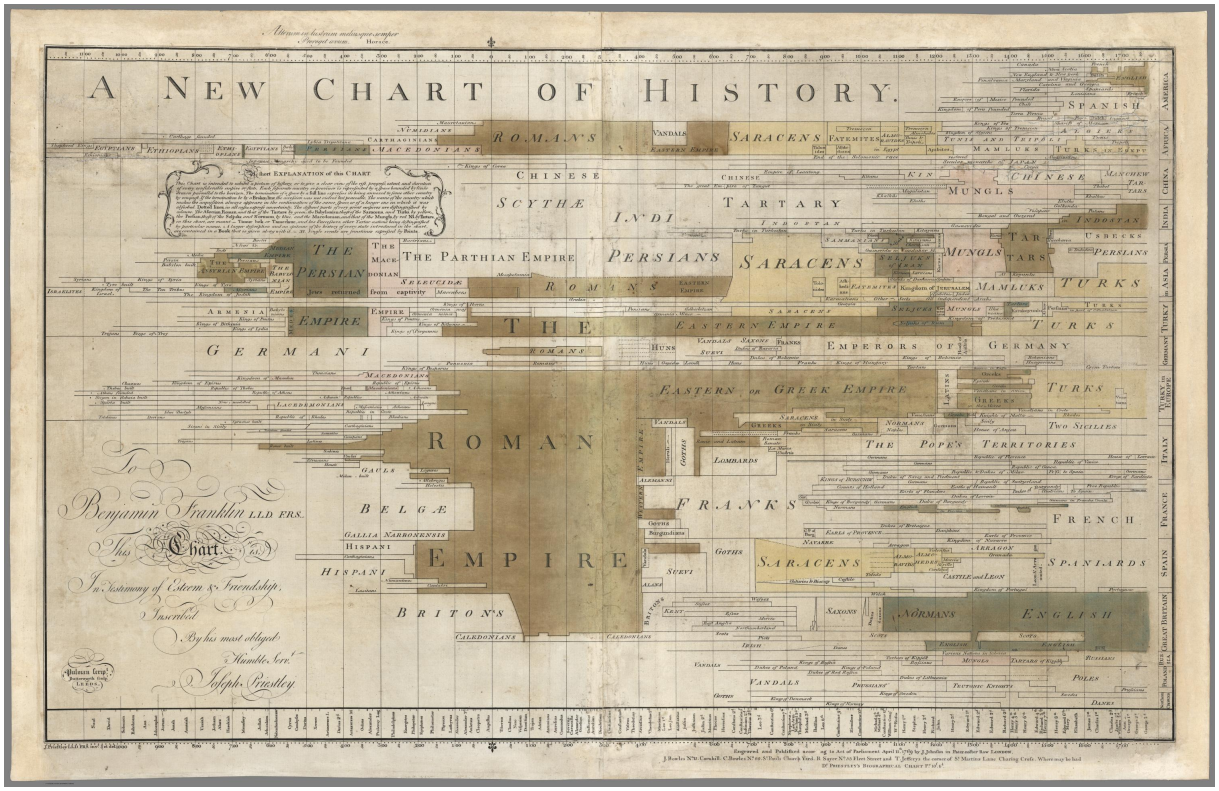

Figure 1.1: This figure shows the 'New Chart of History', as it was created by Joseph Priestley. [Image extracted from `http://www.davidrumsey.com/luna/servlet/detail/RUMSEY~8~1~254828~5519513:A-new-chart-of-history--J--Priestle` Image copyright © 2000 by Cartography Associates. Used under the terms of Creative Commons CC BY-NC-SA 3.0 [*Attribution-ShareAlike 3.0 Unported* 2019]]

1.2.2 Map Time to Space

This is the most common form of representation. Time is converted to a specific distance in space. The best-known example is the time-axis which can be found in many plots. Whether in line charts, bar charts or timelines, this is usually the mapping of choice when it comes to static representations.

1.2.3 Map Time to Visual Variable

Although mapping time to space is so common, it is not the only possible static representation. Another way is to have a visual variable in your graph which changes over time. An example could be to change the colour of the data points over time or to decrease the line width in a line chart over time. There are many possibilities with this mapping and when done well, it can lead to very intuitive visualizations.

1.3 Time-Series vs Timeline

Since this survey only focuses on the analysis and visualization of time-series and not time visualization in general, it is necessary to distinguish between timelines and time-series.

1.3.1 Timeline

A timeline is a chronological display of events, including different events and different time spans. A well-known timeline is the 'New Chart Of History' which can be seen in Figure 1.1. This chart shows the history of empires around the world and is a perfect example for what is not part of this survey.

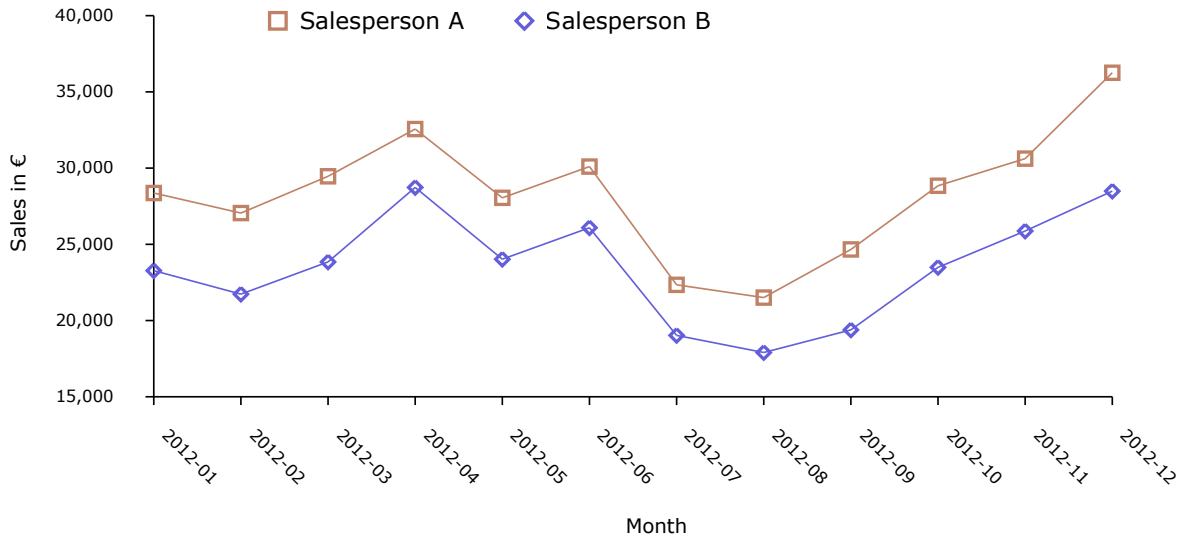

Figure 1.2: This figure shows the time-series of two salespersons. [Graphic used with kind permission of Keith Andrews.]

1.3.2 Time-Series

A time-series on the other hand describes a sequence of measurements over time. The time between those measurements is often constant. Compared to a timeline it contains a much more specific dataset and the changes in this dataset over time have to be determines and visualized. An example of a time-series can be seen in Figure 1.2

Chapter 2

Kinds of Visualizations

To visualize time-series the most common and natural way is to use a simple line chart by simply putting the time dimension on the x-axis and the information giving variable on the y-axis. However, the linear layout does have some limitations. While linear visualizations are good for the comparison of durations, they are not that suitable for stacked or periodic data, since it is not as clear as an area graph or a cyclic graph. Therefore some considerations must be made when choosing the right kind of visualization. The aspect ratio as well as the quantity of the data are important aspects. Furthermore the question arises if the data can be subdivided into subsets to offer a better visualization result [Wills 2011].

In principle three questions must be asked to choose the right kind of visualization. The first question is about the data itself. What data will be visualized and what are its characteristics? The second question is about the reason why the data is visualized. What is the user task for that visualization? By answering those two questions the main factors for question number three are given. The third question talks about how to represent specific data given the specific reason to visualize it. Consequently the question of how is predetermined by the questions of what and why [Aigner et al. 2011].

Axis do have a tremendous impact on the comprehensibility since axis are seen as a guide to understand how the mapping from the data to the visualization was made. Axis serve as a legend for the position of a data point. When creating a visualization it must be kept in mind that the main focus has to stay on the data and the axis should not distract from this. Therefore only include axis if they give necessary information which is needed to get a better understanding of the visualization.

Not only axis can enlarge the information gain of visualizations but also aesthetic formatting, such as coloring by time, sizing by time or shaping by time [Wills 2011].

This chapter mainly focuses on the different types of visualizations, its advantages and disadvantages and the best way to use it in the sense of answering the questions what data and why visualizing it.

Based on the book 'Visualization of Time-Oriented Data' by W. Aigner et al. the representations are categorized based on their scale, scope, arrangement and the data that can be displayed. [Aigner et al. 2011]

2.1 Scale

The scale defines in which way the time between two measurements is regarded. There are three different kinds of scale:

2.1.1 Ordinal

In an ordinal scale, there are only relations between events (data points). It is for example possible to say that event A happened after event B and that event B happened before event C, but it is not possible to say anything about the timing of event A compared to event C. This relationship is displayed in Figure 2.1.

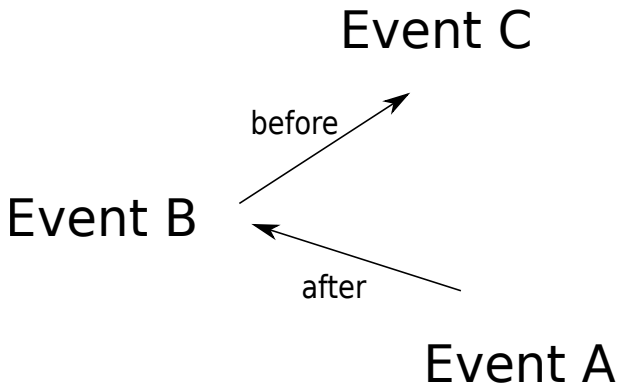

Figure 2.1: This figure shows an ordinal scale, where nothing can be said about the timing between A and C. [Redrawn by the author, original by [Aigner et al. 2011]]

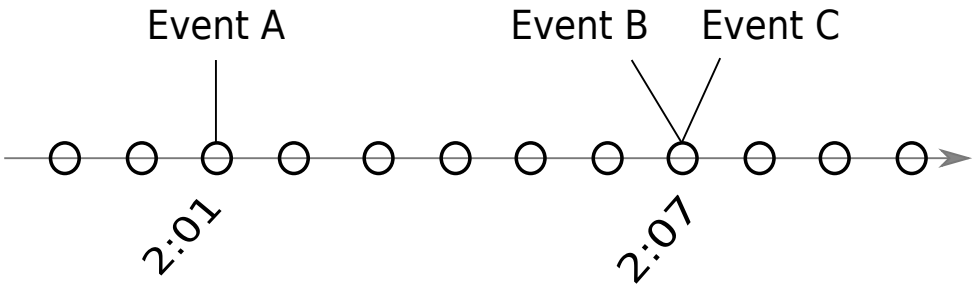

Figure 2.2: This figure shows a discrete scale, where event B and C happen at the same time. [Redrawn by the author, original by [Aigner et al. 2011]]

2.1.2 Discrete

The discrete scale is based on a smallest possible unit. If seconds are defined as the smallest possible unit, it can happen that events happen in the same second. In a discrete scale this means that they happen at the same time (see Figure 2.2).

2.1.3 Continuous

In a continuous scale on the other hand, things never happen at the exact same time. No matter how close two points in time are together, there is always another point of time in between them (see Figure 2.3). The continuous scale is the most realistic scale, but it is not always possible to achieve.

2.2 Scope

The scope defines the structure of the time domain. There are two different types of structures.

2.2.1 Point-Based

In a point-based time domain there is only information about a specific point in time. This point in time has a duration of zero and only assumptions can be made about the time between the certain points where measurements are taken. In time-series it is very common to have a point-based scope, as usually they work with a sequence of measurements taken at specific points in time. An example is to measure the temperature.

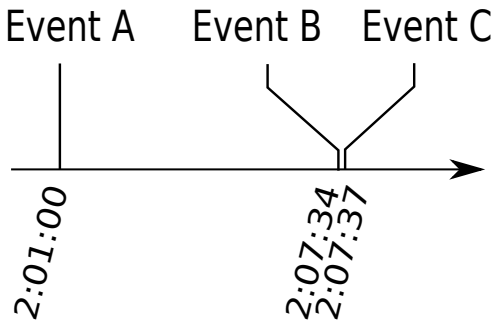

Figure 2.3: This figure shows a continuous scale, where event B and C are still distinguishable. [Redrawn by the author, original by [Aigner et al. 2011]]

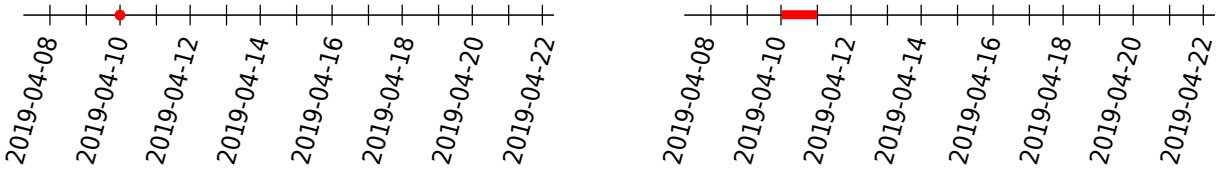

Figure 2.4: On the left is a point-based scope and on the right is an interval-based scope. [Redrawn by the author, original by [Aigner et al. 2011]]

2.2.2 Interval-Based

In an interval-based scope the time is split into subsections and measurements are taken on this section of time. This section of time has a duration greater than 0. An example is to measure the average temperature over a day.

A time value can correspond to both a time point or a time interval. As seen in Figure 2.4, the value 2019-04-10 can either mean the exact time of measurement on the 10th of April, or it can mean the whole day until the start of the next day.

2.3 Arrangement

Naturally time is assumed to be linear and go from past to future and for most representations it is tried to visualize this, but for some cases it is better to use a cyclic arrangement instead.

2.3.1 Linear

As already described above, this is what people usually think of when visualizing time. It goes infinitely far back into the past and from there infinitely far into the future. Out of this line of time a section is taken which is interesting for the given data.

2.3.2 Cyclic

The cyclic approach deals with the periodic nature of data. If changes over a year are displayed, for example, a cyclic representation can be a good idea, as a year is periodic and starts anew once the old year finished. See Figure 2.5 for an example of both a linear arrangement and a cyclic arrangement.

2.4 Data

When trying to visualize data it is also important to look at the data itself. Especially interesting is the number of variables which the data contains.

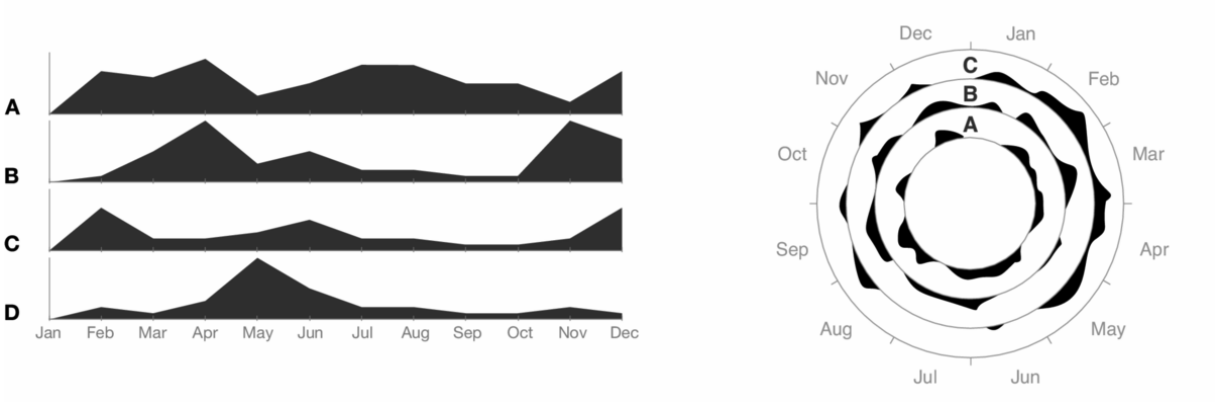

Figure 2.5: On the left is a linear arrangement and on the right is cyclic arrangement. [https://vcg.informatik.uni-rostock.de/~ct/timeviz/pics/full/silhouette-graphs.png - redrawn by C. Tominski and W. Aigner - Original: Harris, R. L.: Information Graphics: A Comprehensive Illustrated Reference. Oxford University Press, 1999.]

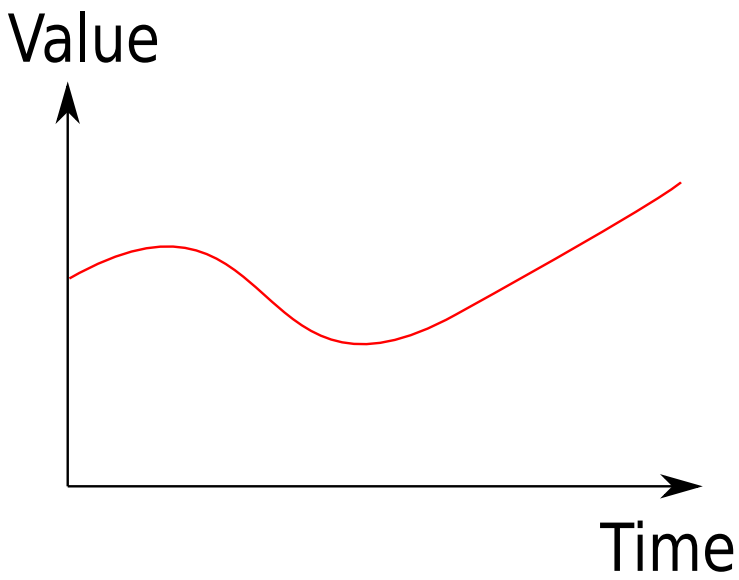

Figure 2.6: There is only one time-dependent variable which has to be displayed. [Drawn by the author.]

2.4.1 Univariate

In an univariate data set there is only one time-dependent variable. The representation only has to show the change of this one variable, as seen in Figure 2.6.

2.4.2 Multivariate

In a multivariate data set there is more than one time-dependent variable, so multiple variables have to be displayed in the same graph (see Figure 2.7). The problem arises how to distinguish between the different variables and how to best show the relationship between the variables.

2.5 Visualizations

Now that the categories to describe the different visualizations have been defined, it is time to have a look at different visualization types and what categories they belong to. An important tool for finding and categorizing the visualizations was 'The TimeViz browser' by C. Tominski and W. Aigner [*The TimeViz Browser* 2019].

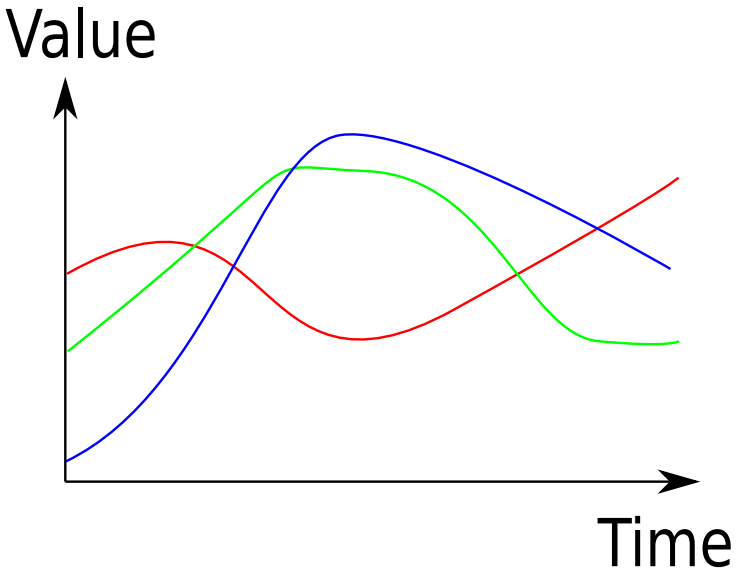

Figure 2.7: There is more than one time-dependent variable which has to be displayed. [Drawn by the author.]

2.5.1 Point Graph

The point graph is also known as point plot or scatter plot. It is the easiest way to represent data. Each data point is represented by one point in the graph, where the position is defined by the data value on the y-axis and the time of measurement on the x-axis (see Figure 2.8).

The point graph belongs to the following categories:

Scale discrete

Scope point-based

Arrangement linear

Data univariate

2.5.2 Line Graph

The line graph (also line plot) is by far the most common visualization for time-series. The points of the scatter plot are connected by lines, which gives the viewer a better impression how the values evolve over time. It is also possible to have more than one time-series per graph, but the more lines there are in the graph, the harder it is to keep it clear. Instead of hard connections between the points, it is also common to use curves to make the line look more appealing (see Figure 2.9).

The line graph belongs to the following categories:

Scale discrete or continuous

Scope point-based

Arrangement linear

Data univariate or multivariate

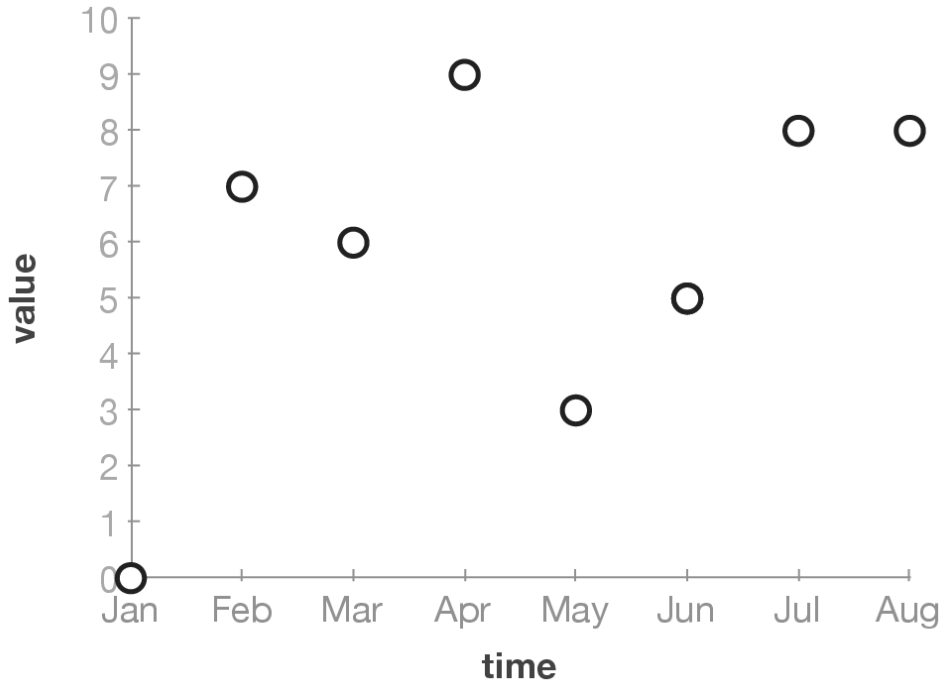

Figure 2.8: A point graph with 8 data points. [https://vcg.informatik.uni-rostock.de/~ct/timeviz/pics/full/scatterplot.png - redrawn by C. Tominski and W. Aigner - Original: Harris, R. L.: Information Graphics: A Comprehensive Illustrated Reference. Oxford University Press, 1999.]

Figure 2.9: On the left is a line graph with linear connections, on the righ is a curved one. [https://vcg.informatik.uni-rostock.de/~ct/timeviz/pics/full/lineplot.png - redrawn by C. Tominski and W. Aigner - Original: Harris, R. L.: Information Graphics: A Comprehensive Illustrated Reference. Oxford University Press, 1999.]

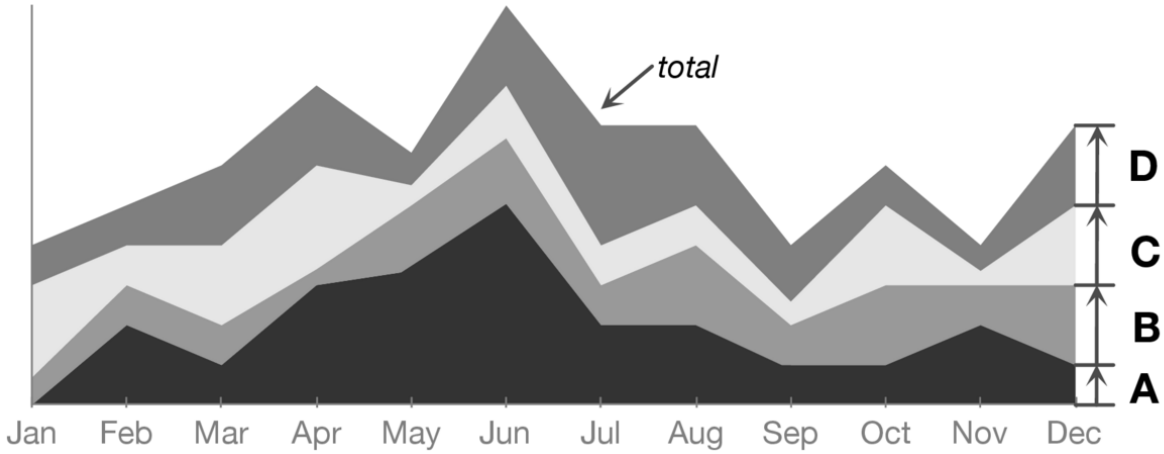

Figure 2.10: A stacked area graph consisting of four different time-series (A,B,C,D). [https://vcg.informatik.uni-rostock.de/~ct/timeviz/pics/full/layer-area-graph.png - redrawn by C. Tominski and W. Aigner - Original: Harris, R. L.: Information Graphics: A Comprehensive Illustrated Reference. Oxford University Press, 1999.]

2.5.3 Stacked Area Graph

The stacked area graph is a layer area graph, where multiple time-series are displayed in the same graph. In order to make any sense, the time-series have to share the same unit and ideally they are measured at the same time. The interesting part about it, is that the values are stacked on top of each other for each data point, meaning that the value of the second time-series is added on top of the value of the first time-series and so on. This makes it possible to see the development of all the time-series together, while still being able to compare the time-series to each other. However, the look of the graph strongly depends on the order of the time-series, so this decision should be made wisely. En example of a stacked area graph can be seen in Figure 2.10.

The stacked area graph belongs to the following categories:

Scale discrete or continuous

Scope point-based

Arrangement linear

Data multivariate

2.5.4 Bar Graph

Bars are used to display the data values, where the length of the bar corresponds to the value. This makes the comparison between two values next to each other much easier. For large time-series a subtype of bar graphs - the spike graph - can be used. In this graph the bar width is reduced to a minimum, so the bars look like spikes, as displayed in Figure 2.11

The bar graph belongs to the following categories:

Scale discrete

Scope point-based

Arrangement linear

Data univariate

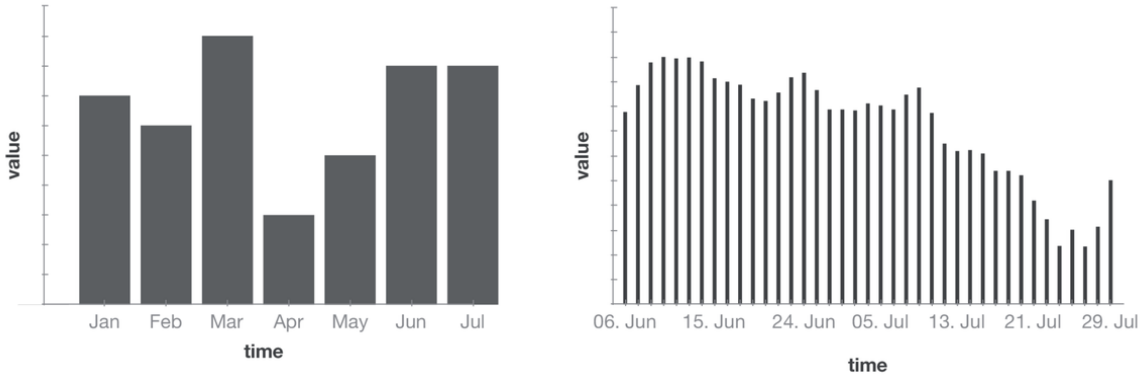

Figure 2.11: On the left side is a usual bar graph, on the right side a spike graph. [https://vcg.informatik.uni-rostock.de/~ct/timeviz/pics/full/bargraphs.png - redrawn by C. Tominski and W. Aigner - Original: Harris, R. L.: Information Graphics: A Comprehensive Illustrated Reference. Oxford University Press, 1999.]

2.5.5 Tile Map

The interesting part about tile maps is that time is mapped to both axis. This means that a tile in the map corresponds to a specific point in time. The value of the data is then displayed by colour or by lightness. This means that it is only possible to display one time-series, but when the axis is chosen well, it can give a very good overview over the data changes, not only linearly but also periodical changes. As seen in Figure 2.12, the tile map is especially intuitive when there are weeks on one axis and weekdays on the other axis, because this layout is well-known from calendars.

The tile map belongs to the following categories:

Scale	discrete
Scope	point-based
Arrangement	linear and cyclic
Data	univariate

2.5.6 Wedge Diagram

The wedge diagram is also known as polar area diagram. It is a purely cyclic diagram, where the values of the time-series are inserted on the radial axis. The arcs to the next point are than shaded. As it is cyclic it is only useful for displaying periodic data. In Figure 2.13 the very first wedge diagram ever created can be seen.

The wedge diagram belongs to the following categories:

Scale	discrete
Scope	point-based
Arrangement	cyclic
Data	multivariate

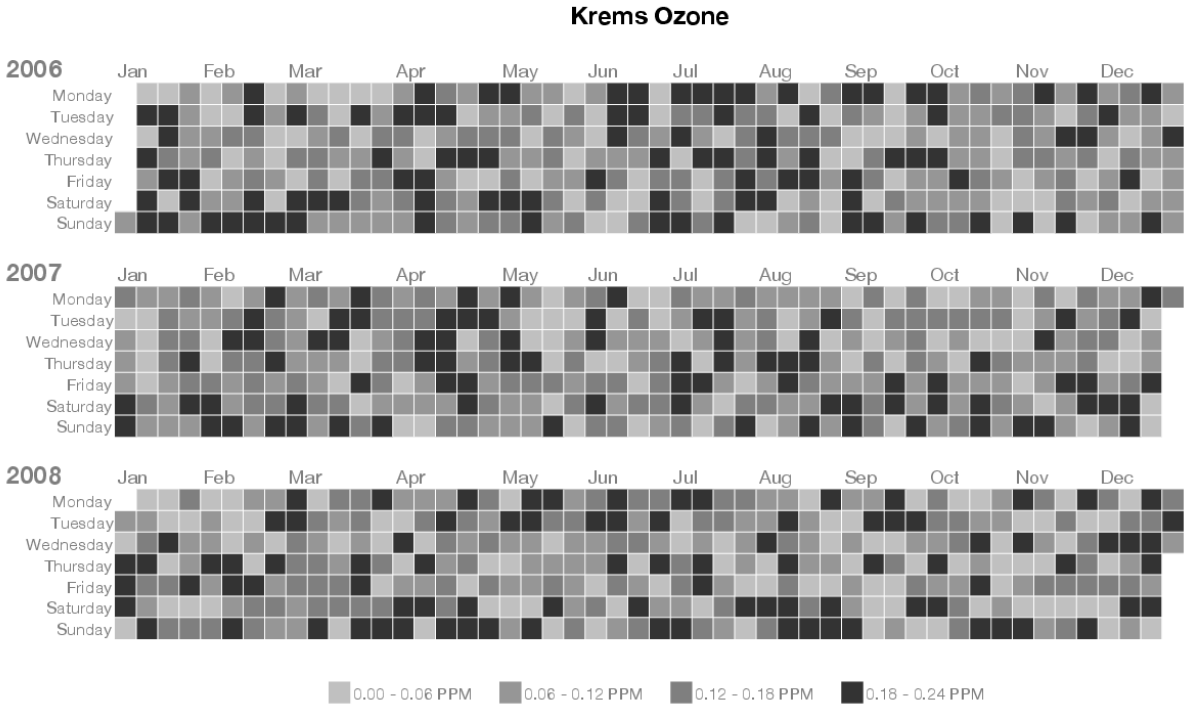

Figure 2.12: The figure shows three different tile maps for the years 2006-2008. The values are encoded by lightness. [https://vcg.informatik.uni-rostock.de/~ct/timeviz/pics/full/tile-map.png - redrawn by C. Tominski and W. Aigner - Original: Mintz, D.; Fitz-Simons, T. & Wayland, M.: Tracking Air Quality Trends with SAS/GRAPH., 1997.]

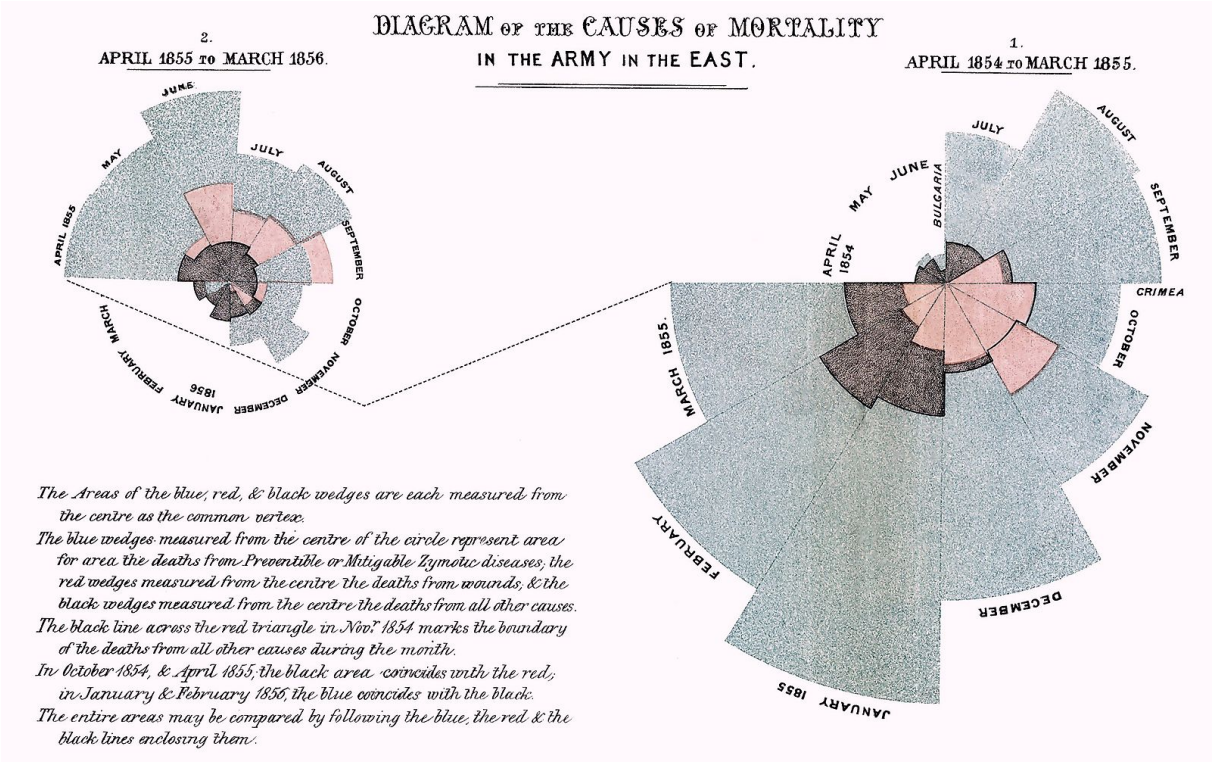

Figure 2.13: First wedge diagram, created by Florence Nightingale in 1858. [https://commons.wikimedia.org/wiki/File:Nightingale-mortality.jpg - used under the terms of the Creative Commons Attribution-Share Alike 3.0 Unported license. [*Attribution-ShareAlike 3.0 Unported* 2019]]

Chapter 3

Time-Series Analysis

In this chapter the focus will be on the different types of time-series analysis. This term describes the procedure of fitting known data values to a proper model. Time-series analysis also contains methods which goal it is to understand the nature of the data and it is also often used for forecasting future data values. [Adhikari and Agrawal 2013]

3.1 Find Patterns

First the main patterns which can be found in visualizations are explained. Those patterns should definitely be known and are therefore also explained with the usage of simple examples.

3.1.1 Trend

A trend is a detectable significant change over time. Such a trend can either be positive or negative and can be detected by statistical procedures. It is also important to understand that such trends can either be local or global and that both scenarios can occur in the same data set. [Longobardi and Villani 2010]

It becomes clearer when having a look at a trend in an example. For instance the overall growth rate of the worlds population between 1950 – 2050 can be seen in Figure 3.1. From 1950 to 1960 a local upward trend can be seen, but on the other hand a global downward trend from 1950 – 2050 can be seen .

3.1.2 Seasonal Cycle:

The next pattern is the seasonal cycle which is a predictable repetitive pattern in the data values. The condition to have such cycles is to have data with a specific interval in it, for instance years or days.

In Figure 3.2 the number of airline passengers which were recorded monthly over the period of 1940 – 1970 can be seen. It can clearly be seen that there is a seasonal trend in the data. In each year during February and March and in the summer months a clear peak can be observed. Since these peaks always occur at the same times of the year a seasonal cycle has been found. It is no problem that there are more passengers from year to year and the level of the peaks increases, it is still predictable and therefore a cycle. [Ihaka 2005]

Another example can be seen in Figure 3.3 where the monthly average temperature of the USA is shown. Here the cyclic pattern of the data can be seen even easier.

3.1.3 Non-Seasonal Cycle

Next on the list are non-seasonal cycles, which are pretty similar to seasonal cycles. They also are repetitive, but the large difference is that they are not predictable. So even though our data has a good interval the cycles just reappear in irregular periods and therefore are unpredictable. This also has the disadvantage that the uncertainty of forecasting is increased tremendously.

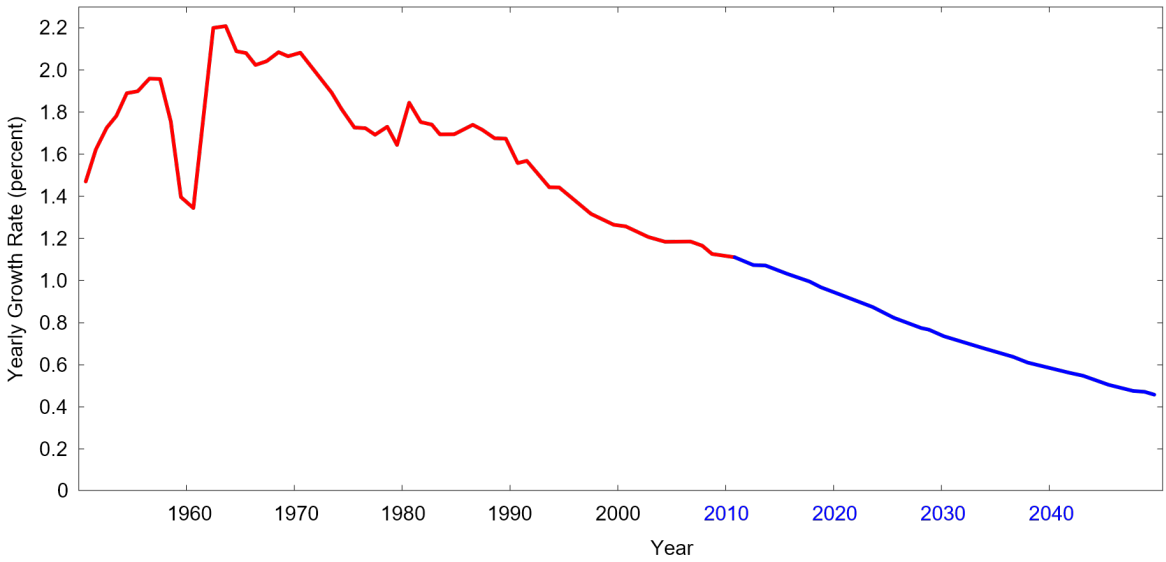

Figure 3.1: This figure shows the world population growth rate from 1950 to 2050. The red parts are real values and the blue parts show forecasted values for the years 2010 to 2050. [Image extracted from `https://en.wikipedia.org/wiki/Population_growth` and used under the terms of the Creative Commons Attribution-Share Alike 3.0 Unported license. [*Attribution-ShareAlike 3.0 Unported* 2019]]

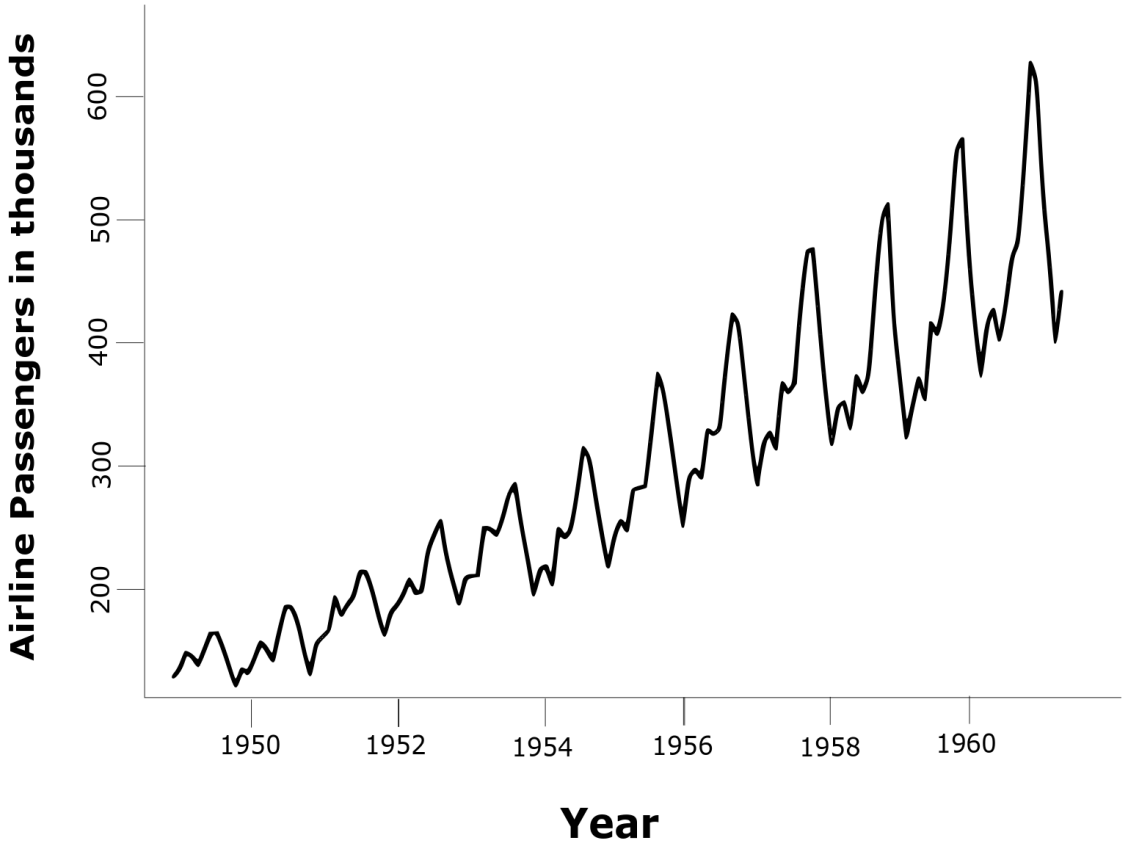

Figure 3.2: International airline passengers in thousands per month. [Image redrawn by the Author. Original from Ross Ihaka [Ihaka 2005]]

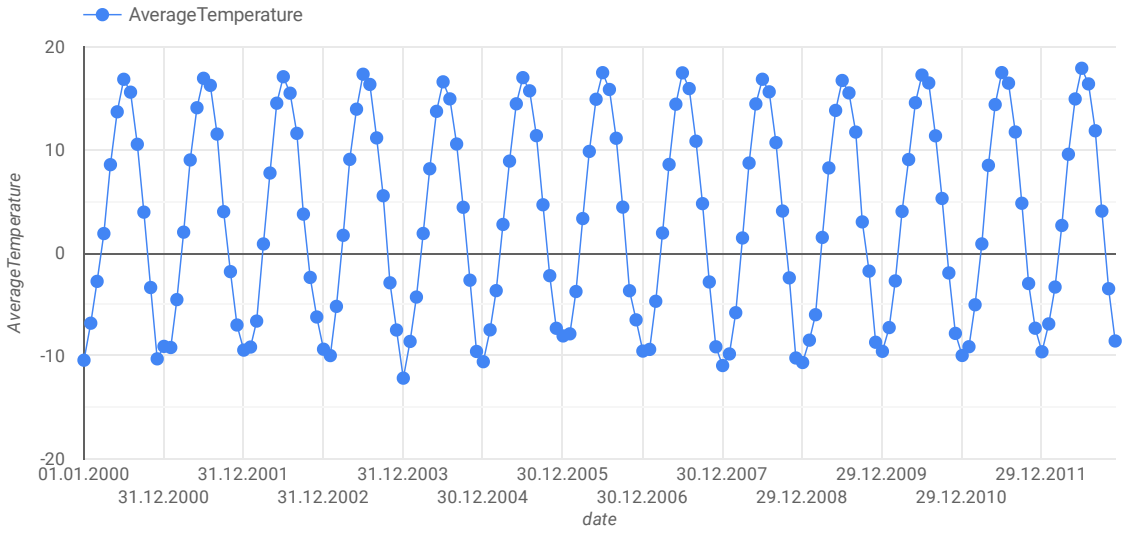

Figure 3.3: Average Temperature in the USA per month. [Image created by the Authors of the paper under the usage of the temperature data provided by Kaggle [*Climate Change: Earth Surface Temperature Data* 2019]]

In Figure 3.4 an example of this pattern can be seen. Already at first glance the cyclic behaviour can be detected. Since it is not possible to find a meaningful interval with which the next cycle could be predicted, a non-seasonal cycle has been found.

3.1.4 Pulses

The next pattern are the pulses which are just a sudden change in the visualization. If it is really a pulse, it should only be a temporary change in the data and return to a "normal" value soon again. In order to be a pulse it is also really important that there is a plausible explanation why it happened. This is so important since in time-series data sets usually gradual changes happen.

3.1.5 Steps

Now that pulses are known the next pattern are the steps. Contrary to pulses a step is a permanent change in the data, but same as for pulses the same rule applies that there should be a reasonable explanation why it happened.

3.1.6 Outliers

The last pattern in this section are the outliers. An outlier is a shift in the data, such as a pulse or step but the change cannot be explained. Since such data points are inconsistent to the rest of the data analysing and also forecasting is affected. There are many types of different outlier types, for example additive or level shift outliers.

3.2 Curve Fitting

In this following section the topic of curve fitting will be explained since it is also often used to analyse time-series data. Generally spoken, curve fitting is an iterative process to construct a curve which fits the data points the best. This means the sum of the errors should be as small as possible[Chatfield 1975].

This technique is usually used when there is non-seasonal data with a clear trend. By using mathematical functions, for instance polynomial functions, the curve is approximated. After each iteration the sum of

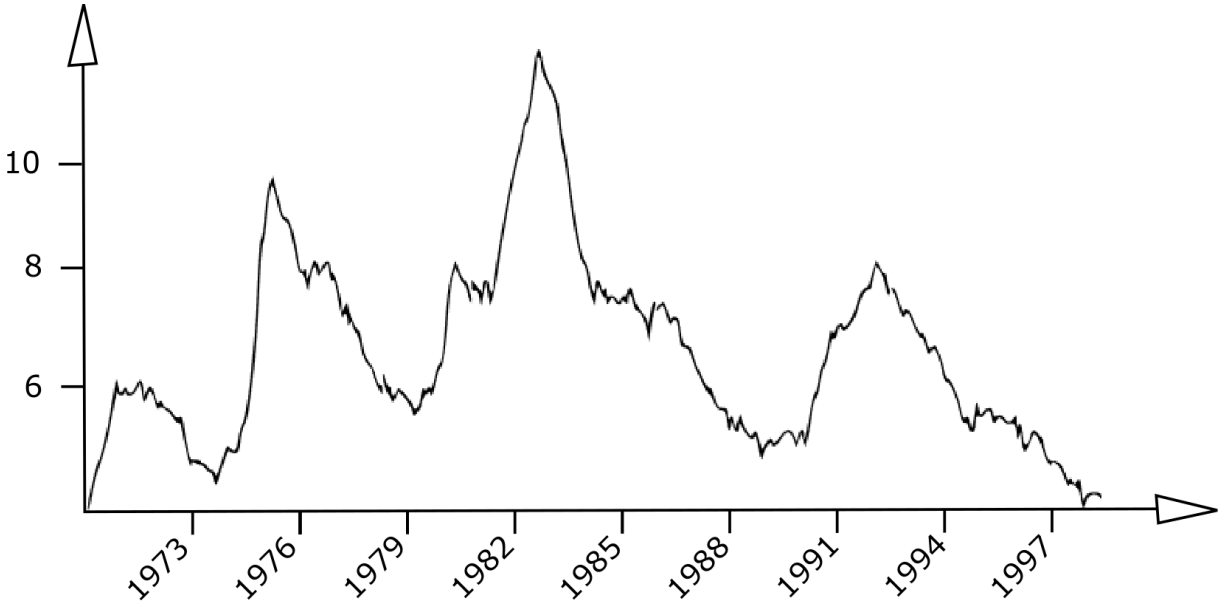

Figure 3.4: Unemployment rate in an unknown country [Image redrawn by the Author. Original from https://www.ibm.com/support/knowledgecenter/SS3RA7_18.2.0/modeler_mainhelp_client_ddita/components/dt/timeseries_nonseasonal.html]

the squared errors is computed. This is done to be able to adjust the model further until a certain error threshold is reached. To be able to get even more accurate curves, interpolation can be used and on the contrary side also smoothing can be applied to get a more visual pleasing curve. This is then of course only an approximation of the data [Chatfield 1975].

Many libraries and packages already have this functionality implemented and can be called by a single command. A good example is the statistical package R which will be covered in Chapter 4.

3.3 Forecasting

The next topic is forecasting and how it is used in time-series analysis.

3.3.1 What Is Forecasting?

Forecasting is when a data point for a future time t + 1 is generated under the usage of the observations until time t. Forecasts are usually needed for a specific period of time and are mostly known as lead time [Box and Jenkins 1990].

3.3.2 Accuracy

Of course the usual goal is to get a forecast as accurate as possible. This means that the current value and the previous predicted value should have a small mean square of the deviations. When a graph with predicted values is seen there are also often confidence intervals around the line which give for instance a probability of 50% that the value will be in this range. The more values are predicted, the larger the covered area of the confidence interval becomes, since with each additional predicted value more uncertainty is added to the data. [Box and Jenkins 1990]

3.3.3 Forecasting Process

In this section the focus lays on the whole process of forecasting data values. The process described below consists of the following six main steps: [Montgomery et al. 2011]

1. **Problem:**

   First step is to know what the expectations for the forecast are. How high or low should the accuracy be, how many values should be predicted and how often are they recalculated (each month, year)?

2. **Data Collection:**

   For the data collection step it is important to gather all the relevant data values which then can be used to forecast new values. The better the data the better the results will be. Missing values and outliers will just tamper with the results.

3. **Data Analysis:**

   In order to choose the best forecasting model, analysing the data to find patterns is important. The most significant patterns were already discussed above in Section 3.1.

4. **Model Selection:**

   As the name already says in this step forecasting models need to be chosen and the models need to be fitted to the data.

5. **Model Validation:**

   Since there are not always many data points to work with in time-series visualization this step is quite challenging. Usually twenty to thirty percent of the data will be used for validation and the rest for fitting the model. This is then called data splitting. If this is not possible since there are too few data points, another suitable way needs to be found.

6. **Monitoring Forecasting Model Performance:**

   It is only natural that over time conditions for models change and the accuracy of the forecast can fall. Therefore monitoring the forecast errors is an important step in this process since it will show if the model needs adjustments.

Chapter 4

Tools for Visualizing

There exist many libraries and tools to visualize time-series. Throughout this chapter different tools and libraries will be presented and compared. To provide a good basis for comparison, all the different tools and libraries have been fed with the same data set to create a line graph. The dataset in use was gathered from www.kaggle.com [*Climate Change: Earth Surface Temperature Data* 2019] and is under a CC BY-NC-SA 4.0 License.

In general the chosen dataset gives information about the climate change in North America and consists of four columns - date, average temperature, average temperature uncertainty and country. The temperature values are represented in degree centigrade. Figure 4.1 shows a small excerpt of the raw dataset. The dataset was preprocessed such that for every year the average temperature values were calculated based on the measured values. Furthermore the period of time is limited to be between 1941 and 2013. This results in 73 data points as input for the charts.

4.1 Libraries

For creating charts in JavaScript there exist many different libraries, such as MetricsGraphics, plotly.js, RGraph and many more. To give some insights on how to create charts using JavaScript the following sections show an example of the usage of some selected libraries. The chosen libraries are D3.js, Dygraphs, ApexCharts, CanvasJS and GoogleCharts. Since only the D3.js library includes the possibility to preprocess the raw data given in a .csv format, the library was used to preprocess the data for all the other libraries as described at the beginning of Chapter 4.

4.1.1 D3.js

D3 is a modern open-source JavaScript library for visualizing data. For visualizing the data web standards are used. For drawing SVG is used. [*D3* 2019]

D3 is a huge library therefore it is a decisive advantage that also only parts of it can be downloaded. This can easily be done since it is known what is needed for the visualization. By using JavaScript there is a wide selection of tools for interaction with the graph, but it needs to be mentioned that good JavaScript skills are needed for implementing these.

4.1.2 Dygraphs

Dygraphs is a open-source JavaScript library used for charting. It can be either downloaded or a hosted version of the library can be used. A link to such a hosted version can be found on the website. [*dygraphs* 2019]

Advantages: [*dygraphs* 2019]

- Fast and easy to use

	A	B	C	D
1	date	AverageTemperature	AverageTemperatureUncertainty	Country
2632	01.11.1987	-2,74	0,18	North America
2633	01.12.1987	-7,38	0,19	North America
2634	01.01.1988	-12,03	0,14	North America
2635	01.02.1988	-9,36	0,11	North America
2636	01.03.1988	-4,23	0,14	North America
2637	01.04.1988	2,22	0,18	North America
2638	01.05.1988	8,87	0,17	North America
2639	01.06.1988	14,39	0,23	North America
2640	01.07.1988	16,98	0,15	North America

Figure 4.1: Excerpt of the dataset used for all libraries and tools. It provides information about the average temperature in North America.

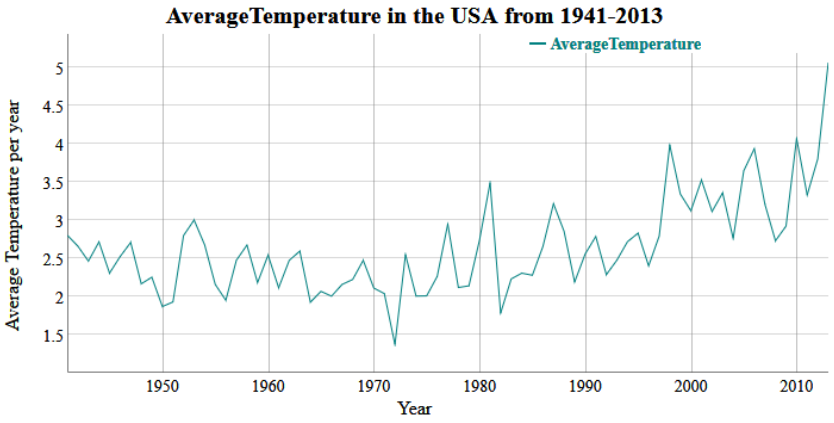

Figure 4.2: A simple visualization created with Dygraphs.

- Customizable: Dygraphs charts are easily customizable by just passing option parameters with the constructor. For a complete list of all option parameters have a look in the documentation.
- Interactive: The default chart created without any options already provides the possibility to zoom in and hover over data points to see their exact values. Zooming also works with all current mobile phones.

To load the data for the graph either the path to an existing .csv file can be given or the raw data itself. Important is that the first column should ideally have the date data in the format YYYYMMDD. A simple visualization of the temperature dataset from America can be seen in Figure 4.2.

In Listing 4.1 a simple example how to create a graph with Dygraphs can be seen. If the the hosted version of the library is used, as used in this example, make sure that "https://" is not missing at the start of the link.

4.1.3 ApexCharts

ApexCharts is an open-source JavaScript charting library, which is modern and interactive. To visualize time-series ApexCharts offers line charts and area charts. To use ApexCharts the library can either be downloaded or a hosted version can be used. Listing 4.2 shows the simple creation of a chart with some options set [*apexcharts.js* 2019].

The creation of a chart is rather easy. To generate and display a chart using ApexCharts, a new object of ApexCharts must be created. The two parameters of this function call are

1. the element in which the chart should be displayed. In Listing 4.2 it is a div-container in a HTML-file

```
new Dygraph(
  document.getElementById("chart"),
  "temperaturePreprocessed.csv",
  {
    title: "Avg. Temp. USA 1941-1950",
    xlabel: "year",
    ylabel: "Avg. Temp.",
    legend: "never"
  }
)
```

Listing 4.1: Dygraphs simple example.

```
var chart = new ApexCharts(
  document.getElementById("chart"),
  {
    chart: {
      height: 400,
      width: 800,
      type: 'line'
    },
    series: [{
      name: 'temperature',
      data: dataset
    }],
    title: {
      text: 'Average Temperature in the US from 1941 to 2013',
      align: 'center'
    },
    xaxis: {
      title: {
        text: 'year'
      }
    },
    yaxis: {
      title: {
        text: 'temperature'
      }
    }
  }
);
chart.render();
```

Listing 4.2: The simple creation of a chart object using ApexCharts and the command to render the chart.

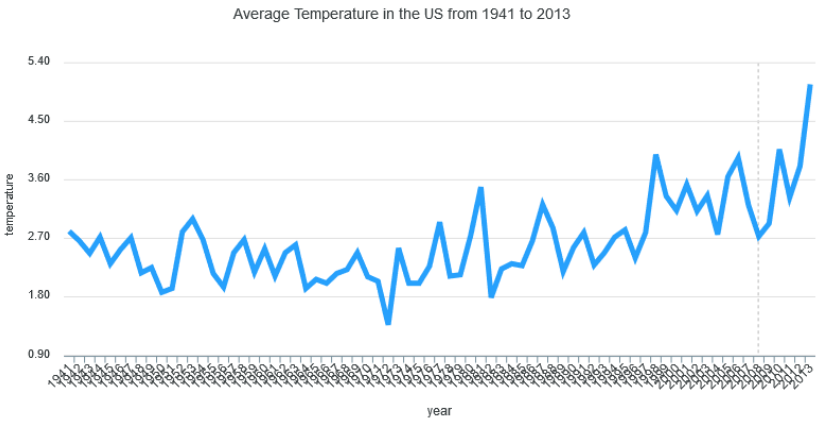

Figure 4.3: Simple line graph created with ApexCharts including a title for the chart as well as for the x and y axis. It shows the average temperature in the US from 1941 to 2013.

which has the id chart.

2. a set of options including the dataset itself and some chart properties. The chart properties were used for customization of the chart. The website of ApexCharts (`https://apexcharts.com/`) includes an overview of all options available.

In Listing 4.2 it can be seen that the data to be visualized is part of the option parameter. The data must follow one of the limited formats that are supported by ApexCharts. D3.js was used to preprocess the data accordingly, since ApexCharts does not offer the possibility to manage this. To generate the line graph, which can be seen in Figure 4.3, the preprocessed dataset follows the format: `{[x: yearValue, y: averageTemperatureValue], [...], ...}`, where `yearValue` is a value between 1941 and 2013 and `averageTemperatureValue` describes the average temperature for that year. Furthermore the line graph was customized in a way that it is assigned a title for the whole chart as well as for the x and y axis. The graph in Figure 4.3 shows the associated chart to Listing 4.2.

An advantage of ApexCharts is that creating a simple chart is by default interactive. Interactive in this case means that hovering over points shows the exact values for that point. Another very profitable feature of ApexCharts is that it provides a download of the graph as .svg or .png by default. Zooming can also be very easily enabled by setting the corresponding option to true. Another positive aspect of ApexCharts is the good documentation of the library which gives the user a good insight in how to create and customize the different chart types available.

There is a limited number of formats for the dataset as input and the selection of the x and y axis value types is a disadvantage of ApexCharts since if not done properly the chart will not be generated correctly or, in worst case, not generated at all.

4.1.4 CanvasJS

CanvasJS is a HTML canvas-based JavaScript charting library which is the proprietary of the company Fenopix [*fenopix* 2019]. To get CanvasJS different licenses are available which can be purchased depending on the type of user task. Fenopix offers a free 30 days trial version and also free licenses for non-commercial use and students. To get the free license a form on the website has to be filled. For this survey the 30 days trial was used.

CanvasJS in general is easy to use and supports many different types of charts. Furthermore the charts can be created to be interactive. To preprocess the data D3.js was used. The creation and rendering of a CanvasJS chart is rather easy. Listing 4.3 shows how the CanvasJS chart for the given dataset was created and rendered [*canvasJS* 2019].

```
var chart = new CanvasJS.Chart(
  "chart",
  {
    data: [{
      type: "line",
      dataPoints: dataset
    }],
    title: {
      text: "Average temperature in the US"
    },
    axisX: {
      title: "Year"
    },
    axisY: {
      title: "Average Temperature"
    },
    zoomEnabled: true,
    exportEnabled: true
  }
);
chart.render();
```

Listing 4.3: The simple creation of a chart object using CanvasJS and the command to render the chart.

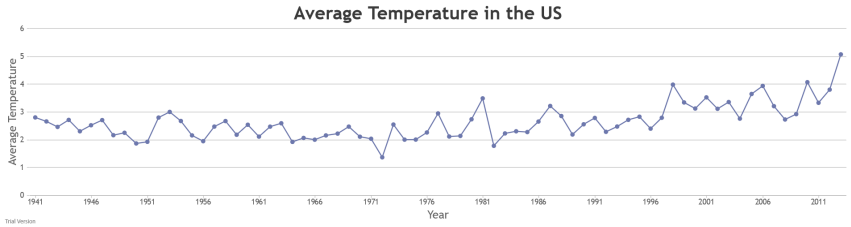

Figure 4.4: A simple line graph created with CanvasJS, including a title for the chart and a title for the x and y axis. It shows the average temperature in the US from 1941 to 2013.

CanvasJS is very easy to read as can be seen in Listing 4.3. The dataset itself is an array of elements and every element has the format: `{label:year, y:averageTemperature}`, where `year` is a value between 1941 and 2013 and `averageTemperature` describes the average temperature value belonging to the corresponding year. To create a new CanvasJS chart the constructor of a CanvasJS chart object must be called with the following two parameters:

1. the id of the container in the HTML-file where the chart should be placed at. In Listing 4.3 the id corresponds to a div-container in a HTML-file with the id 'chart'.

2. a set of options which customize the chart, but also include the data itself.

So in Listing 4.3 the chart is customized by setting a title for the whole chart, for the x axis and also for the y axis. By setting `zoomEnabled` and `exportEnabled` to true, zooming and the export as .png or .jpeg are enabled for that chart. Listing 4.3 leads to the chart which can be seen in Figure 4.4.

The advantages of CanvasJS are the good support and online documentation of the library. Furthermore it is very easy to create and customize charts. The good readability of the code and different options to customize the chart, which were all listed in the documentation of the library, are other important aspects which represent advantages to use this library.

Unfortunately CanvasJS has a bug in its current version which is also a small disadvantage of the library.

Figure 4.5: A simple line graph created with Google Charts, including a title for the whole chart as well as for the x and y axis. It shows the average temperature in the US from 1941 to 2013.

The bug is that when creating and rendering a chart the browser window must be resized, otherwise only the border lines of the chart will be shown but not the graph itself.

4.1.5 Google Charts

Google Charts is an open source charting library written in JavaScript. Before creating a line graph the visualization API must be loaded (see line 1 in Listing 4.4). To start drawing when loading the visualization API has finished, a callback is set on the function `drawChart`. Within this function a line chart object is created. The parameter of the object constructor takes the place, where the chart should be. Afterwards, with `chart.draw()` the chart will be drawn. The parameters for this function are the data and a set of options which customize the chart. The data to be visualized must be provided in a two-dimensional table format, so every row in that table describes one data point except for the first row. The first row represents the titles of the columns in the table. Therefore Google Charts provides the data type `DataTable` [*Google Charts* 2019].

In Listing 4.4 the chart will be customized with a bunch of options. So a title for the whole chart as well as for the x and y axis are set. Furthermore the number of grid lines on the y axis is defined with 10. The line starting with `explorer` enables and defines the zooming behaviour. Zooming into the chart is done via pressing Ctrl-Alt and scrolling. When zoomed-in to the chart there are automatically two buttons arising, one for undoing the zooming and one for changing into pan-mode to pan over the zoomed-in chart. Listing 4.4 results in the chart which can be seen in Figure 4.5. In general when creating a chart with Google Charts hovering over a point shows closer information to that point. The axis get scaled automatically. The library is very easy to use and also the documentation of the library is very good.

4.2 Tools

In contrast to creating a chart with JavaScript there exists some offline and online tools. To demonstrate how to visualize data without the necessity to code in JavaScript the following subsections present some tools and describe their usage for creating a basic line chart. TimeSearcher and Viztree are offline tools to download, while Google Data Studio and VisualizeFree are online webtools.

4.2.1 TimeSearcher

Time Searcher is a visualization tool written in Java and available for academic and non-commercial use. It can be downloaded at the website of the Human-Computer Interaction Lab of the University of Maryland [*TimeSearcher* 2019]. There exist three different versions of TimeSearcher, but for the following report of the tool TimeSearcher 1 was used. This is due to the fact that TimeSearcher2 crashes every time an input file should be loaded and TimeSearcher3 is not freely available for download.

TimeSearcher uses a .tqd file as input. This input file is specified by:

- Title:

  The title of the dataset.

- Static attributes:

  The static attributes describe the x-axis. A name and a data type must be defined here.

```
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

function drawChart() {
  var data = google.visualization.arrayToDataTable(dataset);
  var chart = new google.visualization.LineChart(document.getElementById("chart"));
  chart.draw(data, {
    title: "Average Temperature in the Us",
    legend: 'none',
    vAxis: {
      title: 'Temperature',
      gridlines : {
        count: 10
      }
    },
    hAxis: {
      title: "Year",
      format: '#'
    },
    explorer: {
      axis: 'horizontal',
      keepInBounds: true,
      maxZoomIn: 4.0
    }
  });
}
```

Listing 4.4: The simple creation of a line chart object using Google Charts. Drawing the chart executed with a set of options to customize the chart.

- Dynamic attributes:

  The dynamic attributes describe the y-axis. Again a name a data type must be defined here.

- Number of time points per time-series:

  This part describes the number of data points in the time-series.

- Number of time-series:

  The number of time-series to be shown must be determined.

- Labels for time points:

  The labels describe the names for the static attributes, so they represent the ticks on the x axis.

- Data for each time-series:

  The data for each time-series starts with a name for the time-series and afterwards includes all measured data points separated by a comma.

In case TimeSearcher should display more than one time-series then the additional time-series data has to be within the same .tqd file. Therefore the number of time-series line in the .tqd file has to be adapted depending on the number of time-series which should be displayed. Furthermore the data for each item has to be extend with the additional time-series data. Figure 4.6 shows the preprocessed input file of the dataset.

The GUI of TimeSearches is shown in Figure 4.7. It is separated into five panels. The first panel shows the graph of the time-series. In case the input file includes more than one time-series, all of them will be shown in one graph in the first panel. Therefore the graph can also be customized to show specific features, for example the average, which will then be displayed as a red line. The panel below shows a

```
1   #title
2   Average Temperature in the US 1941-2013
3   #static attributes
4   Date,String
5   #Dynamic attributes
6   average temperature,Float
7   #Number of points
8   73
9   #Number of items
10  1
11  #time points
12  1941,1942,1943,...
13  #data for each item
14  US,2.806166666666666,2.662583333333333,2.4659999999999997,...
```

Figure 4.6: Excerpt of the input file for TimeSearcher.

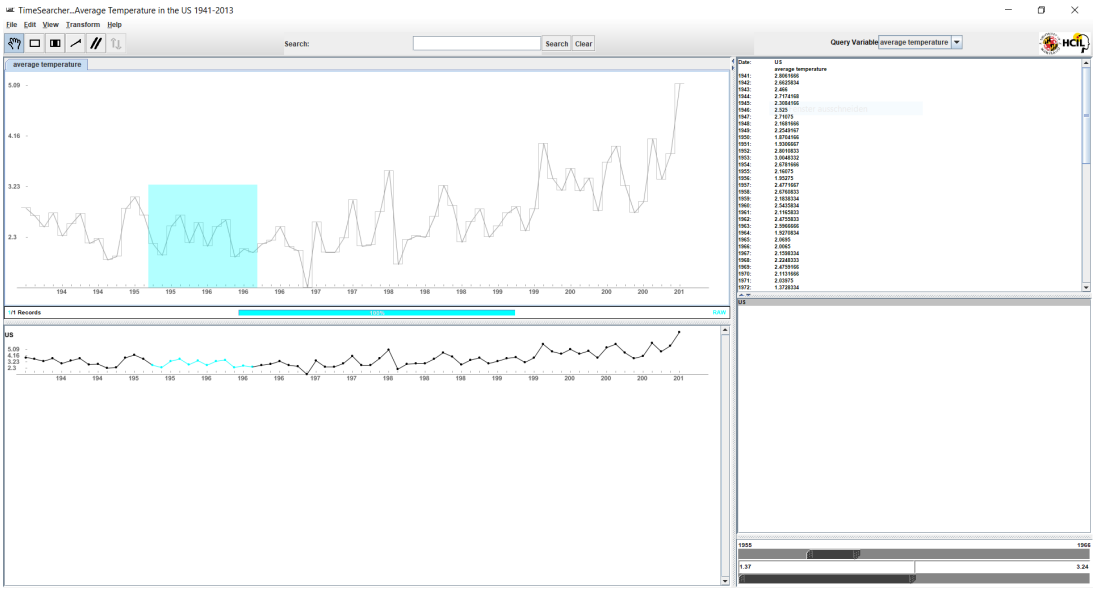

Figure 4.7: The graphical user interface of TimeSearcher 1.

graph for every time-series which is delivered through the input file. The panel on the right shows the data of the selected time-series in the panel below. Finally the last panel consists of two sliders, where query boxes can be customized. In the navigation bar there are some tools which can be selected, such as creating query boxes or creating an average query [*TimeSearcher* 2019].

4.2.2 VizTree

VizTree is a data visualization tool based on suffix trees which is written in C#. It can be downloaded at [*VizTree* 2019]. VizTree uses .dat as input files. In the input file every line represents one time step. For the chosen and preprocessed dataset every line represents a year and holds the average temperature value of this year. Decimal numbers MUST be separated by a comma otherwise VizTree will report an error that the input file cannot be read. Figure 4.8 represents the first ten entries of the .dat file of the preprocessed dataset. The whole preprocessed dataset consists of 73 lines, where every line represents the average temperature values starting from 1941 in line 1 to 2013 in line 73. The .dat file is then inserted into the tool VizTree. As can be seen in Figure 4.9 the user interface of VizTree is build up by four panels to visualize the dataset and a parameter setting area. The panel at the top left contains the whole dataset as a graph, the panel below represents the data as a tree. The panels on the right side of the user interface give more specific insights into the dataset, so the upper panel includes a zoomed-in version of the tree

```
1     2,806166666666666
2     2,662583333333333
3     2,4659999999999997
4     2,7174166666666664
5     2,30841666666666666
6     2,5250000000000004
7     2,7107499999999995
8     2,16816666666666666
9     2,254916666666667
10    1,8704166666666666
```

Figure 4.8: Excerpt of the input file for VizTree, where every line represents one time step.

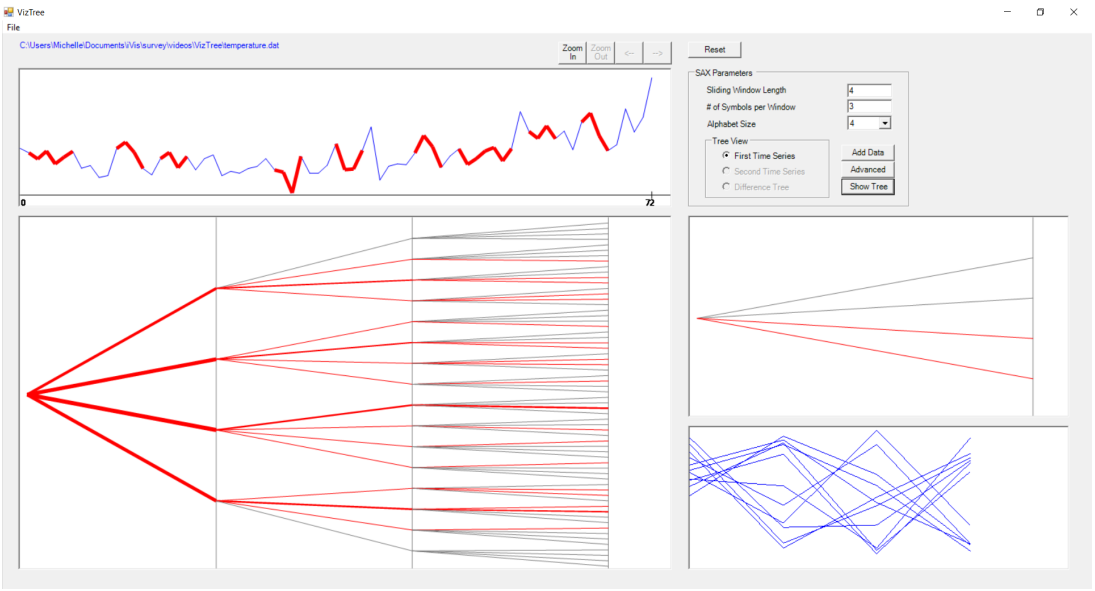

Figure 4.9: Interface of VizTree

and the lower panel contains the subgraphs. In the settings area called SAX parameters the depth of the tree can be set by setting the "# of Symbols per Window". The depth of the tree must have a minimum of 1 (otherwise error occurs) and is in principle limited by a maximum of a two-digit number. The parameter "Alphabet Size" is a value between two and ten. It determines the number of children which a node in the tree has. The "Sliding Window Length" parameter provides information about number of segments per window [Lin et al. 2004].

4.2.3 VisualizeFree

VisualizeFree is a free online tool for visualizing data. To use VisualizeFree an account must be created. After logging in, a dataset can be loaded in form of a CSV file. Since VisualizeFree is able to even preprocess the data it was possible to load our raw dataset. After connecting the dataset a chart can be created by just dragging it into the drawing area. The system is easy to use, since the fields of the dataset just have to dragged onto the placeholders of the graph for axes. In the dataset the data needs to be preprocessed. This can be done by creating a new aggregated field. By right clicking the dataset

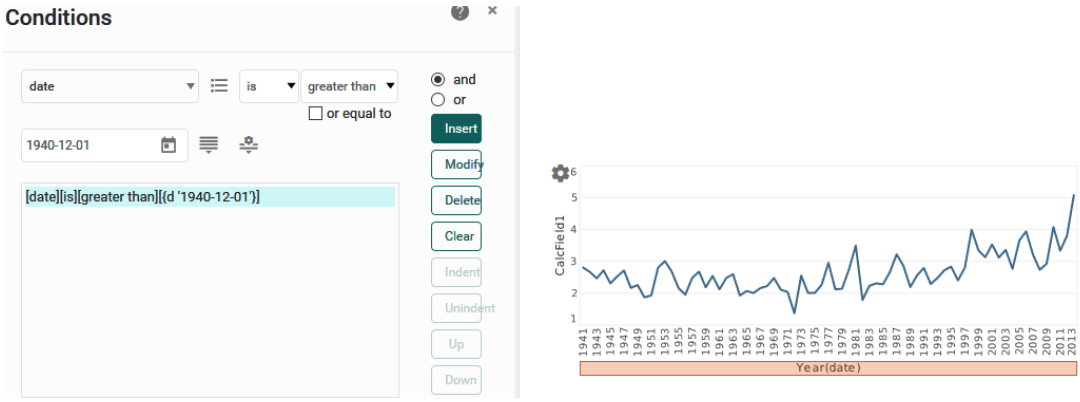

Figure 4.10: This graph was generated with VisualizeFree and the condition window is displayed.

a new calculated field can be generated and by selecting the column and the action to use the field is automatically generated. For this example the average for the averageTemperature field is calculated. It is important to mention that the system detects that the data has a value for each month and therefore calculates the average per year without further instructions. By right clicking the chart conditions for displaying can be set. In Figure 4.10 the generated chart can be seen with the condition window. The condition is again only a small formula and in this case ensures that only the data points newer than December 1940 are displayed.

4.2.4 Google Data Studio

Google Data Studio is a free online tool for visualizing data. To use Google Data Studio a Google account must be used. After signing in with the Google account, a report can be created and a data source connected. To connect a data source Google Data Studio offers different methods. For this survey the upload of a .csv file was used. Unfortunately Google Data Studio does not offer the preprocessing of data therefore the dataset in use was preprocessed earlier and the resulting .csv file was then uploaded. Before finishing the connection with the dataset Google Data Studio provides the possibility to define data types for every field in the .csv file. After connecting the dataset a line chart can be inserted. If the assignment of the data types was done properly, Google Data Studio automatically recognizes what field describes the time dimension and therefore automatically puts it on the x-axis. Since the preprocessed dataset only consists of the fields year and averageTemperature, Google Data Studio also automatically assigns the averageTemperature field on the y-axis and renders the chart.

Figure 4.11 shows the interface of Google Data Studio. The sidebar offers customization of the chart regarding the design but also the data itself. Furthermore filters on the dataset can be applied. Figure 4.11 also shows the line graph generated with the given dataset. In general it can be switched between the following three modes.

- Edit Mode: In edit mode the chart is created and customized. Furthermore filters on the dataset can be applied.
- Preview Mode: In preview mode the chart is shown in an interactive way. So hovering over a point reveals exact information about that point.
- Analyze Mode: This mode is a combination of the preview mode and the edit mode, since the chart can be edited, but is shown like in preview mode. Here the dataset of the chart can also be downloaded as .csv file ore exported to Google Table.

Besides the creation and customization of a chart, Google Data Studios also offers the possibility to share it with others and interactively contribute to it.

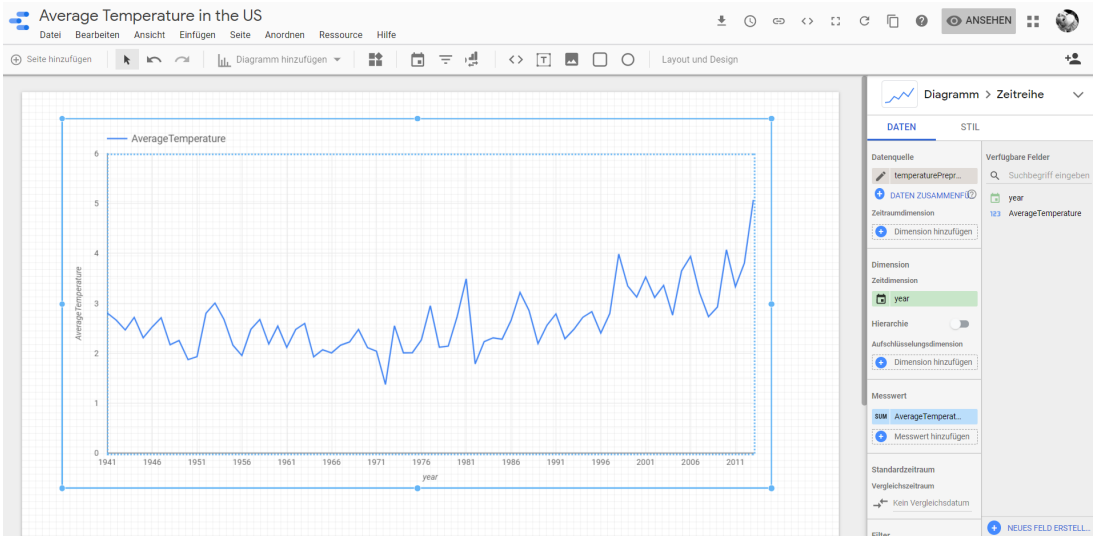

Figure 4.11: A line graph generated with Google Data Studio including its interface. The line graph represents the average temperature in the US from 1941 to 2013.

Name of library	licence	export chart as	website
D3.js	open-source	svg, png	`https://d3js.org/`
Dygraphs	open-source	svg, png	`http://dygraphs.com/`
ApexCharts	open-source	svg, png	`https://apexcharts.com/`
CanvasJS	proprietary of Fenopix	jpeg, png	`https://canvasjs.com/`
GoogleCharts	open-source	svg, png	`https://developers.google.com/chart/`

Table 4.1: The overview of all libraries presented in this chapter.

4.3 Summary

In this chapter all the tools and libraries are shortly summarized in the tables below. Table 4.1 itemizes all libraries which were presented in this chapter and compares them regarding the license type and in what formats the generated charts can be exported. The url of the website is also listed.

Table 4.2 itemizes all tools which were presented in this chapter and compares them regarding the type of tool it is, either an online or offline tool, what type of input is expected by that tool and finally what formats the generated charts can be exported as.

Name of tool	type	input format	export chart as	website
TimeSearcher1	offline	.tqd file	-	`http://www.cs.umd.edu/hcil/timesearcher/`
VizTree	offline	.dat file	-	`https://cs.gmu.edu/~jessica/viztree.htm`
VisualizeFree	online	csv	pdf	`https://visualizefree.com/`
Google Data Studio	open-source	csv, database, ...	pdf, csv	`https://datastudio.google.com/`

Table 4.2: The overview of all tools presented in this chapter.

Chapter 5

Conclusion

Reflecting all the different tools and libraries which were shortly explained in this survey by the example of creating a line graph the following recommendations were made.

When using a library to create a chart the recommendation is to use Dygraphs. This is due to the easy and fast use of the library. Furthermore, the input file (.csv) can be directly added as a parameter to the object creation of Dygraphs. This a decisive advantage since for all the other libraries, except for D3.js, listed in Chapter 4 a specific format of the dataset has to be prepared to use it with the corresponding library. Although D3.js offers the possibility to load an input file, the library is far to complex to use. The following ranking of the libraries is a purely personal opinion and is based on a quick insight into all of the libraries while creating a simple line chart.

1. Dygraphs
2. CanvasJS
3. GoogleCharts
4. ApexCharts
5. D3.js

The reasons for this ranking are the readability of the code, the number of bugs found during the process of creating a simple line graph and the time it took to create the line chart.

Besides the recommendation of Dygraphs as the library to use, the recommended tool is VisualizeFree. This is because the interface of VisualizeFree is very clear and easy to understand. Furthermore a simple way of preprocessing the data is offered by this tool.

Bibliography

9 Characteristics of Time [2019]. 26 Apr 2019. https://simplicable.com/new/time (cited on page 1).

Adhikari, Ratnadip and R. K. Agrawal [2013]. *An Introductory Study on Time Series Modeling and Forecasting*. CoRR (2013) (cited on page 15).

Aigner, Wolfgang, Silvia Miksch, Heidrun Schumann, and Christian Tominski [2011]. *Visualization of Time-Oriented Data*. Springer Science & Business Media, 2011 (cited on pages 1, 5–7).

apexcharts.js [2019]. 01 Apr 2019. https://apexcharts.com/ (cited on page 22).

Attribution-ShareAlike 3.0 Unported [2019]. 22 Apr 2019. https://creativecommons.org/licenses/by-sa/3.0/deed.en (cited on pages 2, 13, 16).

Box, George Edward Pelham and Gwilym Jenkins [1990]. *Time Series Analysis, Forecasting and Control*. San Francisco, CA, USA: Holden-Day, Inc., 1990. ISBN 0816211043 (cited on page 18).

canvasJS [2019]. 01 Apr 2019. https://canvasjs.com/ (cited on page 24).

Chatfield, Christopher [1975]. *The Analysis of Time Series: Theory and Practice*. 6th Edition. Chapman and Hall, 1975. 352 pages. ISBN 978-1-4899-2925-9 (cited on pages 17–18).

Climate Change: Earth Surface Temperature Data [2019]. 01 Apr 2019. https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data (cited on pages 17, 21).

D3 [2019]. 01 Apr 2019. https://d3js.org/ (cited on page 21).

dygraphs [2019]. 01 Apr 2019. http://dygraphs.com/ (cited on page 21).

fenopix [2019]. 01 Apr 2019. https://fenopix.com/ (cited on page 24).

Google Charts [2019]. 01 Apr 2019. https://developers.google.com/chart/ (cited on page 26).

Ihaka, Ross [2005]. *Time Series Analysis Lecture Notes for 475.726*. 2005 (cited on pages 15–16).

Lin, Jessica, Eamonn J. Keogh, Stefano Lonardi, Jeffrey P. Lankford, and Donna M. Nystrom [2004]. *VizTree: a Tool for Visually Mining and Monitoring Massive Time Series Databases.* Dec 2004, pages 1269–1272. doi:10.1016/B978-012088469-8/50124-8 (cited on page 29).

Longobardi, Antonia and Paolo Villani [2010]. *Trend analysis of annual and seasonal rainfall time series in the Mediterranean area*. International Journal of Climatology 30.10 (2010), pages 1538–1546. doi:10.1002/joc.2001. https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/joc.2001 (cited on page 15).

Montgomery, D.C., C.L. Jennings, and M. Kulahci [2011]. *Introduction to Time Series Analysis and Forecasting*. Wiley Series in Probability and Statistics. Wiley, 2011. ISBN 9781118211502. http://www.stat.ipb.ac.id/en/uploads/KS/S2%20-%20ADW/3%20Montgomery%20-%20Introduction%20to%20Time%20Series%20Analysis%20and%20Forecasting.pdf (cited on page 18).

The TimeViz Browser [2019]. 26 Apr 2019. http://browser.timeviz.net/ (cited on page 8).

TimeSearcher [2019]. 01 Apr 2019. http://www.cs.umd.edu/hcil/timesearcher/ (cited on pages 26, 28).

VizTree [2019]. 01 Apr 2019. https://cs.gmu.edu/~jessica/viztree.htm (cited on page 28).

Wills, Graham [2011]. *Visualizing time: Designing graphical representations for statistical data*. Springer Science & Business Media, 2011 (cited on page 5).