



Master's thesis
Master's Programme in Data Science

Visualizing Changes over Time in Hierarchical Customer Data Using the Plotly Python Graphing Library

Fanni Kovapohja

October 19, 2022

Supervisors: Professor Giulio Jacucci
Doctor Chen He

Examiners: Professor Giulio Jacucci
Doctor Chen He

UNIVERSITY OF HELSINKI

FACULTY OF SCIENCE

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

Tiedekunta — Fakultet — Faculty		Koulutusohjelma — Utbildningsprogram — Degree programme	
Faculty of Science		Master's Programme in Data Science	
Tekijä — Författare — Author			
Fanni Kovapohja			
Työn nimi — Arbetets titel — Title			
Visualizing Changes over Time in Hierarchical Customer Data Using the Plotly Python Graphing Library			
Työn laji — Arbetets art — Level		Aika — Datum — Month and year	
Master's thesis		October 19, 2022	
		Sivumäärä — Sidantal — Number of pages	
		98	
Tiivistelmä — Referat — Abstract			
<p>Time-dependent hierarchical data is a complex type of data that is difficult to visualize in a clear manner. It can be found in many real-life situations, for example in customer analysis, but the best practices for visualizing this type of data are not commonly known in business world.</p> <p>This thesis focuses on visualizing changes over time in hierarchical customer data using the Plotly Python Graphing Library and is written as an assignment for a Finnish company. The thesis consists of a literature survey and experimental part. The literature survey introduces the most common hierarchical visualization methods, and the different possible encoding techniques for adding time dimension on top of these hierarchical visualization methods. Moreover, the pros and cons of different visualization techniques and encodings are discussed about.</p> <p>In the experimental part of the thesis, visualization prototypes are designed using the Plotly Python Graphing Library. A company customer data set of the commissioning company is partitioned into hierarchical customer segments by a hierarchical industrial classification TOL 2008, and changes over time in a continuous variable are visualized by these segments. Two hierarchical visualization techniques: the sunburst chart and treemap, are used to create two prototype versions, and the combination of color, typography, and interaction is used to encode time dimension in these prototypes. The same prototypes are also exploited to visualize customer segments by an artificial hierarchy created by combining multiple categorical features into a hierarchical structure.</p> <p>The prototypes are validated in the commissioning company by arranging an end user study and expert review. Concerning the prototypes by the industrial classification: According to the end users and experts, both prototype versions are very useful and well-implemented. Among the end users, there was no significant difference in which one of these prototype versions is faster to use, but the clear majority of the respondents regarded the sunburst chart version as their favorite prototype. The two experts who participated in the expert review had different opinions on which one of the prototype versions they would select to be utilized in practice. Concerning the prototypes by the artificial hierarchy: These prototypes also received positive feedback, but the possibility to change the order of features in the hierarchy was considered as an extremely important development idea.</p> <p>ACM Computing Classification System (CCS): Human-Centered Computing → Visualization → Visualization Techniques Human-Centered Computing → Visualization → Empirical Studies in Visualization</p>			
Avainsanat — Nyckelord — Keywords			
Hierarchical visualization, information visualization, time-dependent hierarchical data, customer data			
Säilytyspaikka — Förvaringsställe — Where deposited			
Muita tietoja — Övriga uppgifter — Additional information			

Contents

1	Introduction	2
2	Thesis Commission	4
2.1	Motivation	4
2.2	Data Set of Company Customers	5
2.3	Research Questions	6
2.4	Methodologies	8
3	Hierarchical Data	10
3.1	Definition of Hierarchical Data	10
3.2	Hierarchical Customer Data	11
4	Hierarchical Visualization Techniques	13
4.1	Explicit Methods	15
4.2	Implicit Methods	18
4.2.1	Adjacency Diagrams	18
4.2.2	Enclosure Diagrams	21
5	Visualizing Changes over Time in Hierarchical Data	26
5.1	Different Types of Changes in Hierarchical Data	26
5.2	Different Techniques for Visualizing Changes in Hierarchical Data	28
5.2.1	Color	29
5.2.2	Typography	31
5.2.3	Interaction	33
5.2.4	Animation	35
5.2.5	Small Multiples	37
6	Designing the Experimental Visualizations	39
6.1	Domain Problem and Data	39
6.2	Data Types and Operations	40
6.2.1	Data Wrangling for (E_1)	42

6.2.2	Data Wrangling for (E_2)	43
6.3	Visual Encoding and Interaction	46
6.3.1	Requirements	46
6.3.2	Selecting the Suitable Hierarchical Visualization Techniques	47
6.3.3	Selecting the Suitable Time Encoding Techniques	48
6.4	Final Visualization Products	50
6.4.1	Prototypes for (E_1)	51
6.4.2	Prototypes for (E_2)	54
7	Validating the Experimental Visualizations	59
7.1	End User Study	59
7.1.1	Structure of the End User Study	60
7.1.2	Results of the End User Study	61
7.2	Expert Review	64
7.2.1	Expert Review of $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$	64
7.2.2	Expert Review of $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$	69
8	Conclusion and Future Work	72
8.1	Conclusion of the Literature Survey	72
8.2	Conclusion of the Experimental Part	73
8.3	Future Work	76
	Bibliography	78
	Appendix A Figures: $Sunburst_{(E_1)}$	82
	Appendix B Figures: $Treemap_{(E_1)}$	86
	Appendix C Figures: $Sunburst_{(E_2)}$	90
	Appendix D Figures: $Treemap_{(E_2)}$	95

1. Introduction

Customer data is information about customers which can be either personal or company customers depending on the type of the customer relationship. Regardless of whether the customers are persons or companies, it is useful to analyze the customer data in order to better understand the structure of the customer base and enhance customer acquisition and retention strategies. However, customer data is often very complex and can contain variables of any data type; also hierarchical variables. For instance, geographical location consisting of multiple hierarchical levels, such as country, region, and city, is a common hierarchical variable that is usually known for both personal and company customers. Moreover, one very essential hierarchical variable related to company customers is industry that classifies businesses into hierarchical groupings based on similar products or services. By utilizing a hierarchical variable, customers can be partitioned into hierarchical customer segments that become more precise as the hierarchical level increases. Hierarchical customer segments can also be created artificially by dividing customers into segments by utilizing multiple categorical variables successively, for instance, by partitioning company customers first by country and second by turnover.

When customer base is divided into hierarchical segments by exploiting a hierarchical variable or multiple categorical variables, other features related to customers, such as number of orders, are compared between the different customer segments. To add even one more dimension into this scenario, it would also be interesting to know how the number of orders has developed over time in different customer segments rather than focusing on only one distinct time point. From the strategic point of view, it is very useful to explore in which customer segments the trend is increasing or decreasing. In other words, the task is to analyze changes over time in a numerical target variable by a hierarchical explanatory variable. In this thesis, solutions to this problem are examined by means of data visualization. The research area was given as an assignment from a Finnish company who is interested in researching how to visualize changes over time in an important continuous variable by hierarchical customer segments.

The research area of the thesis is relevant when considering customer data analytics in general since it would be very useful for many businesses if there existed proven methods to visualize target variable changes in hierarchical customer segments. Moreover, the

thesis can certainly provide a perspective on visualizing time-dependent hierarchical data sets that are not necessarily related to customers. There is a lot of academic literature on visualizing only hierarchical data or only time series data, but there is a relatively little discussion on how to visualize time-dependent hierarchical data, and also for this reason the topic is significant. One additional point that needs to be taken into account is that the data visualizations designed to be exploited in business use have to be very clear and uncomplicated since they are utilized for decision-making by non-technical persons. The threat is that if the visualizations are difficult to understand, they will never be exploited in practice. However, one of the purposes of the thesis is to encourage the utilization of also other kind of business visualizations than only bar and pie charts in order to be able to observe more complex relations and phenomena from data.

In the thesis work, the commissioning company requested to first conduct a comprehensive general-level research on the most commonly used hierarchical visualization techniques and different possible encoding techniques for adding time dimension on top of these hierarchical visualization techniques. The purpose of this general-level research is to give a thorough overview of the alternative options to visualize hierarchical data and time-dependent hierarchical data. Second step is to utilize the gathered information in practice by designing experimental visualization prototypes visualizing changes in the commissioning company's hierarchical customer data set using the *Plotly Python Graphing Library* [30]. Moreover, these visualization prototypes are validated in the commissioning company by arranging an end user study and expert review.

The thesis is structured as follows: Chapter 2 describes the thesis commission and motivation behind it, introduces the commissioning company's customer data set that is visualized later in the thesis, and presents research questions and methodologies of the thesis. Chapter 3 covers the definition of hierarchical data and discusses about hierarchical customer data. Chapter 4 presents the most prominent and commonly used hierarchical visualization techniques and considers the advantages and disadvantages of each of them. Chapter 5 covers the different types of changes in hierarchical data and different techniques for visualizing changes in hierarchical data. Chapter 6 walks through the whole design process of the experimental visualization prototypes created for the commissioning company. In Chapter 7, the experimental visualization prototypes are validated by presenting and analyzing the results of the end user study and expert review. Finally, in Chapter 8, the thesis is concluded and future work is discussed.

2. Thesis Commission

This master’s thesis is written as an assignment for a Finnish company. The general problem area is to systematically examine and compare different hierarchical visualization techniques and time encoding techniques in order to find the most suitable techniques for visualizing changes over time in hierarchical customer data using the *Plotly Python Graphing Library* [30]. This chapter describes the thesis commission and motivation behind it. Moreover, the data set used in the experimental visualizations of the thesis is introduced. Lastly, specific research questions and methodologies are presented.

2.1 Motivation

The commissioning company maintains a large and versatile data set of their company customers and aims to utilize this data effectively for improving their customer insight and making better decisions. The data exists, but the challenge is to develop proper techniques for finding useful relations and phenomena from the vast amounts of data. Fortunately, this is the situation in which data visualization turns out to be very beneficial. By visualizing, large amounts of data can be condensed into a single view, and many different feature dimensions can be displayed simultaneously by using different visual encodings such as length, area, color, and position.

However, since there are numerous visualization layouts and techniques from which to select, it can be difficult to know, which one to exploit in each case. Finding the most suitable visualization techniques often requires systematic research and comparison between all the possible options. Therefore, also the commissioning company is interested in examining different data visualization techniques in order to find the most suitable ways to visualize their customer data set. This research work about systematically comparing different visualization techniques that could be exploited to visualize the customer data set of the commissioning company is the main agenda of the thesis.

The customer data set includes a continuous variable V that is especially interesting and followed closely in the commissioning company. All the other variables related to the customers are categorical, and one of these categorical features is also hierarchical: *Finnish Standard Industrial Classification TOL 2008* consisting of five hierarchical levels [31]. By

exploiting the categorical features, the large data set of all customers can be partitioned into smaller customer segments which are easier to deal with.

Changes over time in the variable V are vital to monitor regularly in the commissioning company. Moreover, it is important to examine how the changes in V differ among different customer segments. Actually, the visualization prototypes presented later in the thesis aim to answer especially to this question. Customers can be divided into segments in multiple different ways according to which common characteristics are considered at each time, but this thesis focuses primarily on visualizing the customer segments formed by the hierarchical variable *Finnish Standard Industrial Classification TOL 2008*. In addition, this thesis considers how to visualize customer segments formed by multiple categorical features.

2.2 Data Set of Company Customers

Table 2.1: Variables in the data set of company customers.

Variable	Description	Data Type
<i>Time Point</i>	Equally spaced time points	Discrete
<i>Customer ID</i>	A unique identifier for each customer	Categorical, Identifier
V	The main variable of interest	Continuous
<i>Industry</i>	<i>Finnish Standard Industrial Classification TOL 2008</i> consisting of five hierarchical levels [31]	Categorical, Hierarchical
<i>Business Entity</i>	The five most common business entities of Finland	Categorical
<i>Region</i>	The 19 regions of Finland	Categorical
<i>Size</i>	<i>Small, medium-sized or large</i> enterprise by number of employees	Categorical
<i>Turnover</i>	Seven turnover categories, for instance, €1M - €3M	Categorical

The data set of the company customers of the commissioning company consists of the variables introduced in Table 2.1. Each customer has a unique identifier, *Customer ID*. The main variable of interest, continuous V , is updated regularly, and therefore there exists a time series of V for each customer. The data set includes the following categorical variables: *Business Entity*, *Region*, *Size*, and *Turnover* of a company. Moreover, the categorical and hierarchical variable of *Finnish Standard Industrial Classification TOL*

2008 is included in the data set and referred shortly as *Industry*. It consists of five hierarchical levels presented in Table 2.2 with illustrative example classes.

Table 2.2: The hierarchical structure of *Industry* (=Finnish Standard Industrial Classification TOL 2008 [31]).

Level	Coding	Example
Level I	Character code	R - Arts, entertainment and recreation
Level II	2-digit code	93 - Sports activities and amusement and recreation activities
Level III	3-digit code	932 - Amusement and recreation activities
Level IV	4-digit code	9329 - Other amusement and recreation activities
Level V	5-digit code	93291 - Skiing centre activities

2.3 Research Questions

The research questions of the thesis are divided into two categories: general-level and experimental research questions. The general-level research questions cover hierarchical visualization in general, while the experimental research questions relate to specific hierarchical visualization problems of the commissioning company. Answering first to the general-level research questions forms a great base to start working with the experimental research problems which are solved by applying the general ideas in practice. Both the general-level and experimental research questions are listed on the next page.

The first general-level research question, (G_1), strives for carrying a systematic research on different hierarchical visualization techniques in order to receive a comprehensive understanding of all the commonly used alternatives to visualize hierarchical data. Also the pros and cons of these different visualization methods are addressed by the question (G_2). The third general-level research question, (G_3), considers how time dimension could be added on top of these hierarchical visualization techniques.

The research work concerning the general-level research questions is of significant value for the commissioning company since they are interested in hierarchical customer data visualization also outside of the specific experimental research problems of the thesis. They would like to receive a comprehensive overview of the whole area of hierarchical visualization, especially how to visualize changes over time in hierarchical data, at a general level before focusing on the specific visualization prototypes. This way, they can utilize the information of the thesis also in the future for solving different kinds of hierarchical visualization problems they may encounter.

General-level Research Questions

- (G_1) Which are the most prominent and commonly used hierarchical visualization techniques?
- (G_2) What are the main advantages and disadvantages of these hierarchical visualization techniques?
- (G_3) How time dimension can be added on top of these hierarchical visualization techniques?¹

Experimental Research Questions

- (E_1) How changes over time in a continuous variable V could be visualized by a hierarchical explanatory variable *Industry* using the Plotly Python Graphing Library? In other words, how to depict simultaneously the hierarchical structure of the explanatory variable and the development of the continuous variable over time?¹
- (E_2) How hierarchical visualization techniques could be utilized to visualize changes over time in a continuous variable V by multiple different categorical features using the Plotly Python Graphing Library?¹

¹ In (G_3), (E_1), and (E_2), hierarchical structures are considered to remain unchanged over time.

The experimental research questions aim at visualizing the commissioning company's data set of company customers. More specifically, the experimental research questions (E_1) and (E_2), focus on visualizing changes over time in a continuous variable V by hierarchical structures. *Industry* is the hierarchical structure of the first research question, and the second research question considers the hierarchical structure formed by multiple different categorical features. A hierarchical structure divides the customers into hierarchical customer segments that become more precise as the hierarchical level increases. Since the topic area is very broad and challenging, the commissioning company requested that the thesis should primarily focus on (E_1) and regard (E_2) as more of an additional research question.

The purpose of the experimental visualizations is to display in a single figure how the variable V has changed over time in the whole customer population as well as at all the precision levels of the hierarchical customer segments. In order to create visualization prototypes answering to the experimental research problems, both a proper hierarchical

visualization technique and a proper time encoding technique have to be selected. The research work conducted for the general-level research questions is exploited to make these design decisions. There is a lot of information to be visualized, and therefore there is a need to design proper layouts and interactions to alleviate information overload. The visualization tool used in the thesis is the Plotly Python Graphing Library, which also allows a user to create interactive visualizations.

2.4 Methodologies

The general-level research questions are answered to by conducting a literature survey. The previous work about hierarchical visualization is examined thoroughly in order to receive comprehensive answers to the questions (G_1) , (G_2) and (G_3) . The extensive use of different scientific sources offers various different perspectives of the topic area and builds a versatile understanding around the general-level research questions. Based on the results of this literature survey, a suitable visualization techniques are selected in order to create visualization prototypes answering to the experimental research questions. Two alternative visualization prototypes are designed for both experimental research questions, (E_1) and (E_2) , using the Plotly Python Graphing Library.

The nested model for visualization design and validation (hereinafter referred as the *nested model*) introduced by Munzner [22] is utilized to design and validate the experimental visualization prototypes of the thesis. The nested model consists of the following four nested steps:

- Step 1. Identifying the domain problem and data.
- Step 2. Abstracting the problem into data types and operations.
- Step 3. Designing visual encoding and interaction techniques.
- Step 4. Creating algorithms.

Each of these steps are executed separately but the decisions made during previous steps have an inevitable effect on the following steps [22]. A mistake made while identifying a domain problem will reflect the final product even if all the rest of the steps are implemented perfectly. Munzner presents distinct threats and validation methods considering each individual step, and these are taken into account while designing the visualizations of this thesis. Validation process can be divided into upstream and downstream validation according to whether the validation of a step is carried out before the following steps are accomplished (upstream) or after all the steps are accomplished (downstream). It is also noteworthy, that the steps can be taken iteratively by returning to refine the design

decisions of the previous steps. In this thesis, the Step 4 is omitted since the ready-made algorithms of the Plotly Python Graphing Library are utilized.

After the visualization prototypes are finished, an end user study and expert review about the pros and cons of the different prototypes are arranged within the commissioning company. This is part of the validation process of the nested model. The end user study is meant for the end users of the visualizations, and its purpose is to measure the clarity and real-world applicability of the prototypes. With an expert review, also more analytical perspectives about the visualizations can be received. It is very useful to listen to both the end users of the visualizations and the experts from the field. Finally, by exploiting the results of the end user study and expert review, the final conclusions about the different visualization prototypes are drawn, and possible directions for future work are discussed about.

3. Hierarchical Data

Every visualization process begins from the profound understanding of data. Therefore, before going deeper into hierarchical visualization, the definition and characteristics of hierarchical data are briefly covered. After that, it is discussed about how hierarchies relate to customer data.

3.1 Definition of Hierarchical Data

The word *tree* can be regarded as a synonym for *hierarchy*. A natural tree, consisting of the root, branches and leaves, is used to describe a data structure in which nodes are arranged at different levels and connected to each other by edges. At the first and highest level of a tree, there is only one node, which is called the *root*. The nodes connected directly to the root node form the second level of a tree. From each of the nodes at the second level, can branch new nodes, which form together the third level of a tree. Analogously, unlimited number of additional levels can be created. If node *A* is connected to the node *B*, and *A* is at one level higher than *B*, *A* is called the *parent* of *B*, and *B* is called the *child* of *A*. All the nodes in a tree have exactly one parent, except the root node that has no parent. If a node has no children, it is called a *leaf* node, and if a node has at least one child, it is called an *interior* node. The mathematical definition of a tree is "a simple, undirected, connected, acyclic graph" [29].

In summary, it could be said that a hierarchy is a tree-like arrangement of data elements. *Hierarchical* instead is an adjective describing that something is arranged into a hierarchy. Thus, *hierarchical data* refers to data that is organized into a tree structure. As usual, in this thesis, the hierarchical level of the root node is referred as the highest level in a hierarchy, and as moving further away from the root, the hierarchical level becomes lower. Expressed using ordinal numbers, the root level is considered as the first hierarchical level, and as moving further away from the root, the ordinal number of the hierarchical level increases.

It is noteworthy that hierarchical data contains both the *hierarchical structure* about the arrangement of nodes and edges, and *content information* about the feature (or features) linked to the nodes [27]. Content information refers to the target variable values,

while the hierarchical structure is an explanatory structure by which the target variable is being interpreted.

Hierarchical data can be divided into two categories according to whether the hierarchy of data is *natural* or *artificial*. These may not be universal terms, but in this thesis they are defined as follows: Natural hierarchy refers to a hierarchical structure that originally exists in data, while artificial hierarchy is data that is artificially organized into a hierarchical form. Examples of both types of hierarchical data are given in the next section in the context of customer data.

3.2 Hierarchical Customer Data

Customer data often includes naturally hierarchical variables. For instance, regardless of whether a customer is a personal customer or company customer, the geographical location of a customer is usually known. Location is a good example of a natural hierarchy since it consists of naturally hierarchical levels: the earth, continent, country, region, city, district.

By using a naturally hierarchical variable, customers can easily be divided into hierarchical customer segments that allow examining customers at different precision levels. The higher levels, such as the continent and country levels in the case of geographical location, provide a good large-scale overview of customer base, while the lower levels, such as regions and cities inside a single country, specify the scope into a lot more precise and smaller set of customers. By exploiting different hierarchical visualization techniques, this kind of hierarchy can be displayed in various interesting and useful ways. Another example of a naturally hierarchical feature is the industry of a company customer, which is the variable of interest in the experimental research question (*E1*).

Even data not including any natural hierarchies may be possible to be arranged into a hierarchical form. Artificial hierarchies can be created, for example, by using hierarchical clustering methods, or as its simplest, by merely organizing data into a tree format by multiple interesting hand-picked variables. For example, in the paper [19], the personal customer data set of a bank is arranged into a hierarchy by three hand-picked variables: bank branch, region, and banking business. The personal customers are first divided by bank branch, after which each bank branch is partitioned by region. Each region is further divided by banking business, and finally, the individual customers in each banking business are listed. Content information about the amount of money held in each account and the income of each customer are attached to this artificial hierarchy. After being created, the artificial hierarchy is visualized using a hierarchical visualization technique, the *treemap*, in which the nested arrangement of customers encodes the structure of the artificial hierarchy, color encodes the amount of money held in each account, and area encodes the income of each customer. Furthermore, it is annotated in the article that the

structure and information content of the visualization can be modified a lot by altering the order in which the variables are used to partition the customers, and by attaching different features (content information) to the customers.

In the context of this thesis, for example the following three categorical features in the commissioning company's data set of company customers: *Business Entity*, *Region*, and *Size*, could be combined into a tree format by (1) partitioning customers by *Region*, (2) partitioning customers in each *Region* additionally by *Business Entity*, and (3) partitioning each customer segment formed by *Region* and *Business Entity* additionally by *Size*. This way hierarchical customer segments that become more precise as the hierarchical level increases from (1) to (3) are obtained. There are many customers that belong to the same *Region* but certainly fewer customers that belong to the same *Region*, same *Business entity* and same *Size* category. Expectedly, the main variable of interest V could be attached to the nodes. Analogously to the paper [19], a suitable hierarchical visualization technique could be exploited to visualize this artificial hierarchy.

Actually, to answer to the experimental research question ($E2$), similar steps to the previous example need to be taken: pick sensible categorical variables from the data set of company customers, use the selected variables in a sensible order to create an artificial hierarchy, and after that, examine what would be the most suitable visualization techniques for visualizing the created artificial hierarchy taking into account that time dimension needs somehow to be added on top of the visualization.

In customer data analytics, it can be very fruitful to combine multiple variables into an artificial hierarchy and this way generate hierarchical customer segments. Customers can be analyzed at different precision levels, which are first coarser but become more precise as more variables are considered simultaneously. This kind of segmentation allows to observe the distinct effects of each variable as well as the combined effect of all of them. A large data set becomes condensed into a form that is easier to deal with and analyze. Moreover, artificial hierarchy can be displayed in various different ways by utilizing different hierarchical visualization techniques.

4. Hierarchical Visualization Techniques

In this chapter, the most prominent and commonly used techniques for visualizing hierarchical data are covered in order to receive a comprehensive overview of different alternatives for the experimental visualizations. There can also exist completely other hierarchical visualization techniques or different variations of the techniques presented in this chapter, but the scope of the thesis is limited only to the most prominent ones. In other words, this chapter is a literature survey answering to the general-level research questions ($G1$) and ($G2$). In Chapter 5, the general-level research question ($G3$) is answered to by considering how time dimension could be added on top of the hierarchical visualization techniques presented in this chapter.

Different hierarchical visualization techniques are applicable in different use cases, so it is reasonable to discuss what are the special characteristics, advantages, and disadvantages of different alternatives. The choice of a suitable visualization technique obviously depends on what kind of and what size of data set is to be visualized, but it also depends a lot on what is desired to be acquired from the visualization. There are trade-offs, for example, (1) between the amount of presented information and the clarity of a visualization, (2) between the clear presentation of content information and the clear presentation of the hierarchical structure, and (3) between space-efficiency and the clear presentation of the hierarchical structure. Furthermore, it has to be decided how much information is displayed in a single view, and which part of the information is only possible to be seen through interactions.

The hierarchical visualization techniques can be divided into two categories: the *explicit* and *implicit* methods, according to how the parent-child relationships are encoded [25]. The explicit methods clearly display the edges connecting the nodes, while the implicit methods do not [25]. Instead, the implicit methods use the arrangement of nodes to encode the parent-child relationships [25]. The implicit methods can further be partitioned into the *adjacency diagrams* and *enclosure diagrams* according to whether adjacency or containment represents the hierarchical arrangement of nodes [14]. The most common visualization techniques from all of these categories are introduced in the

following sections with illustrative example figures. As a side note, there also exists map-based hierarchical visualization techniques, such as drill down maps, regarding specifically geographical hierarchies. However, map-based techniques are not covered in this thesis since the data set of company customers does not contain any geographical hierarchies.

The example figures of this chapter are created using an artificially constructed data set introduced in Table 4.1. It is easy to compare the characteristics of different visualization techniques as the same data is used on the base of all the example visualizations. Moreover, the data set is an example of a hierarchical customer data set and therefore suits well to the topic area of the thesis. The hierarchical variable in the example data set is *Industry*. In this case, only the character code *F - Construction* in addition to the 2-digit and 3-digit code levels branching from it are considered. Thus, the hierarchy consists of three levels. All the 13 nodes belonging to the hierarchy are listed in Table 4.1 with the code of *Industry* in the first column and name of *Industry* in the second. In the third column, the hierarchical structure is revealed by presenting the parent of each node. However, the root node *F* has no parent. The last column describes the values of an imaginary numerical target variable related to the customers. The values of the target variable sum to 9 000 000 at each hierarchical level, and these values aggregate from child nodes up to their parent node.

Table 4.1: An artificially constructed hierarchical customer data set for the example visualizations.

Code	Name of <i>Industry</i>	Parent	Value
F	Construction		9 000 000
41	Construction of buildings	F	4 000 000
42	Civil engineering	F	3 000 000
43	Specialised construction activities	F	2 000 000
411	Development of building projects	41	3 000 000
412	Construction of residential and non-residential buildings	41	1 000 000
421	Construction of roads and railways	42	1 500 000
422	Construction of utility projects	42	1 000 000
429	Construction of other civil engineering projects	42	500 000
431	Demolition and site preparation	43	1 000 000
432	Electrical, plumbing and other construction installation activities	43	500 000
433	Building completion and finishing	43	250 000
439	Other specialised construction activities	43	250 000

4.1 Explicit Methods

The explicit visualization methods visualize hierarchical data by encoding edges as visible links between nodes [25]. These methods resemble a lot of a natural structure of a tree. Generally, the strength of the explicit methods is in displaying the hierarchical structure in a clear and intuitive manner, whereas they are usually not so appropriate for expressing content information. The explicit methods easily become obscure if the sizes or colors of nodes are used to encode content information [21]. Moreover, the explicit methods take a lot of space so they are not very useful for visualizing large hierarchies [21]. Interactions, such as drill down and up, can alleviate this problem, but not even them are sufficient if the hierarchy is extremely large [21].

Traditional Node-link Diagram

Maybe the most intuitive way to visualize a hierarchy is to use explicit nodes and links, and place the root node at the top and other nodes underneath it. In addition to this, in the *traditional node-link diagram* (or only *node-link diagram*), the nodes at the same level of a tree are at the same horizontal level in the visualization. The prefix *traditional* is used to avoid confusion since sometimes all explicit visualization methods are referred as node-link diagrams. There are various different algorithms to arrange the nodes into this kind of *tree layout*, for example, *Reingold-Tilford* algorithm optimizes the layout in such a way that minimum amount of space is wasted [14].

The paper [27] summarizes well the pros and cons of the traditional node-link diagram: It is an excellent choice to visualize small hierarchies since it is very clear and easily understandable. However, the weakness of the traditional node-link diagram is its inefficient use of space of which a large proportion is filled by only background. Partly due to this, the layout is so intuitive, but on the other hand, therefore it is not suitable for visualizing large hierarchies that would spread out over an unpractically large area. Moreover, it is difficult to attach content information to the nodes that are quite small in this layout, while the hierarchical structure is very clearly displayed.

An example of the traditional node-link diagram is shown in Figure 4.1. The figure is created based on the data in Table 4.1. However, since content information is difficult to attach to the traditional node-link diagram, only the hierarchical structure without node values is displayed. The hierarchical structure is a lot easier and faster to perceive from Figure 4.1 compared to just trying to interpret it from Table 4.1.

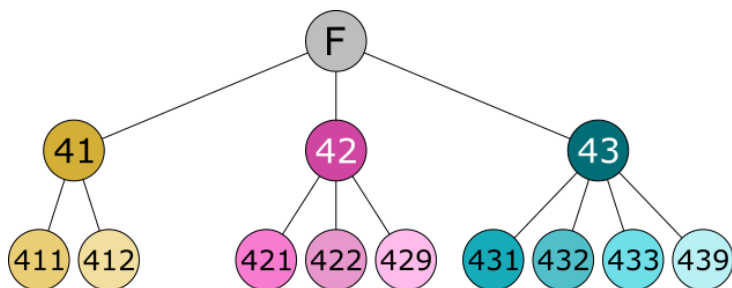


Figure 4.1: An example figure of the traditional node-link diagram. The figure is created based on the data in Table 4.1. However, only the hierarchical structure without node values is visualized.

Dendrogram

In another layout option, the *dendrogram*, explicit nodes and links are also used, and the root node is at the top. However, instead of congruence between hierarchical levels and horizontal levels, all the leaf nodes are put at the same horizontal level [14]. The dendrogram is also called the *cluster layout* [14] and is often used in the context of hierarchical clustering. It is quite similar to the traditional node-link diagram, so the strengths and weaknesses are roughly mutual. Nevertheless, unlike the traditional node-link diagram, the dendrogram always allows observing all the leaf nodes alongside each other even if they are at different levels of a tree, which can be very useful in some use cases.

The difference between the traditional node-link diagram and dendrogram is visualized in Figure 4.2. It is the only figure in this chapter that is not created based on the data in Table 4.1. The reason for this is that in the example data set in Table 4.1, all the leaf nodes are at the same hierarchical level. In that exceptional case, the traditional node-link diagram and dendrogram would look exactly the same, and the difference between them could not be visualized.

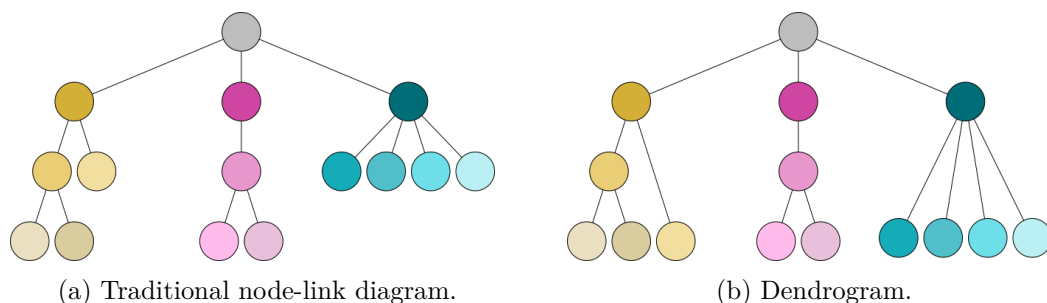


Figure 4.2: Two different explicit layout versions of the same tree. (This is the only figure in this chapter that is not created based on the data in Table 4.1.)

Indented Tree

The *indented tree* (or *indented list*) exploits tabular layout to visualize a hierarchy. All the nodes are presented one below another, and the nodes at the same hierarchical level of a tree are at the same indented level in the visualization. All the levels are usually not displayed simultaneously, but a user can navigate between different levels through interaction capabilities. The indented tree is utilized, for example, to visualize the structure of a Microsoft Windows File Explorer and other similar file manager systems consisting of nested folders and files inside them [27].

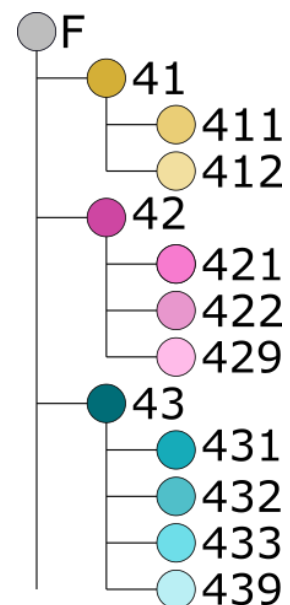
The indented tree is very common and easily interpretable but it faces also criticism: The layout requires a massive amount of vertical space [14]. Since all the levels are often not shown at the same time, a user needs to transfer a lot between different levels [27]. Furthermore, if the hierarchy is large, the hierarchical structure is difficult to perceive [27]. For example, it can be hard to say how many levels there are, which nodes are at the same level, and how many nodes there are at each level [27]. On the other hand, the article [14] points out that particular nodes can efficiently be searched by interactive properties, node labels are smoothly and quickly readable, and content information, for instance, file size, can easily be attached to the nodes in a written form. Each node is located at its own row so there is a lot of space to put text beside the nodes. In addition, color of a node (or color of the whole row of a node) can be used to encode the value of a target variable.

An example of the indented tree is displayed in Figure 4.3. The figure is created based on the data in Table 4.1. However, content information is not attached to this figure either; only the hierarchical structure without node values is shown. Similarly to the traditional node-link diagram, also the indented tree shows the hierarchical structure very clearly and intuitively.

Alternative Layouts

The traditional node-link diagram and dendrogram can also be oriented differently, for example, in such a way that the root is on the left and the hierarchy expands onto the right (instead of the top-to-bottom layout). Analogously, the indented tree can be oriented in such a way that the nodes are arranged from left to right (instead of from top to bottom), and the indented levels are drawn by vertical distances (instead of horizontal distances).

Figure 4.3: An example figure of the indented tree. The figure is created based on the data in Table 4.1. However, only the hierarchical structure without node values is visualized.



Furthermore, the traditional node-link diagram, dendrogram, and indented tree can all be visualized by using polar coordinates in order to obtain the *radial layout* [7, 14]. Compared to the *Cartesian layout*, the radial layout appears more interesting and saves space more efficiently [14]. Which one is the clearer and more practical layout, may depend on the situation.

4.2 Implicit Methods

Instead of drawing visible links between nodes, the hierarchical structure can also be visualized by arranging nodes either by adjacency or containment [14]. These kind of techniques are called the implicit [25] (or *space-filling* [14]) methods. The paper [21] describes the advantages of the implicit visualization methods compared to the explicit methods: The implicit methods utilize space more efficiently than the explicit methods so they are better suited for visualizing large hierarchies. However, also the implicit methods can benefit from interactions if the hierarchy is very large. Furthermore, the implicit methods are better at displaying content information by both node size and color. On the other hand, the hierarchical structure is not as intuitive and fast to read as in the case of the explicit methods. When comparing the explicit and implicit methods, the trade-off between the clear presentation of content information and the clear presentation of the hierarchical structure can be observed.

4.2.1 Adjacency Diagrams

In the adjacency diagrams, adjacency reveals the parent-child relationships. Nodes are encoded as solid bars or arcs, and their place in the hierarchy is represented by their position in relation to consecutive adjacency nodes [14]. The adjacency diagrams are a compromise between the explicit methods and enclosure diagrams exploiting the containment encoding [18]. Since edges are not displayed explicitly, the adjacency diagrams are more space-efficient than the explicit methods [18]. On the other hand, using the enclosure encoding instead of adjacency would save even more space [18]. However, the hierarchical structure is easier to perceive when using the adjacency diagrams [18], which reveals the trade-off between space-efficiency and the clear presentation of the hierarchical structure. In this work, the adjacency diagrams created by using the *partition layout* [14] are only discussed about. These kind of adjacency diagrams are space-filling variants of the traditional node-link diagram. However, the paper [14] points out that the adjacency diagrams can also be created by using the cluster layout similarly to the dendrogram.

Icicle Chart

The *icicle chart* exploits solid bars adjacent to each other to visualize a hierarchy [14]. The icicle chart can be regarded as a space-filling version of the traditional node-link diagram (in the Cartesian coordinates) since the root is at the top and other nodes are underneath it in such a way that the nodes at the same level of a tree are at the same horizontal level in the visualization [14]. It is also common to rotate the layout in such a way that the root is on the left and child nodes expand onto the right. This horizontal layout is, for example, the default icicle chart layout of the Plotly Python Graphing Library [30].

Similarly to other implicit methods, the icicle chart is good at displaying content information. The bar length can easily be utilized to encode the quantity of a target variable. The value of a target variable in a parent node is usually the sum of the target variable values in its child nodes. Thus, the length of the root node bar represents the total sum of a target variable, and the different levels of the icicle chart reveal how the quantities of a target variable distribute among the hierarchical structure. Another target variable can be attached to the figure by coloring the bars.

The advantages of the icicle chart include the clear presentation of the hierarchical structure and the easy integration of a target variable into every node from the root to

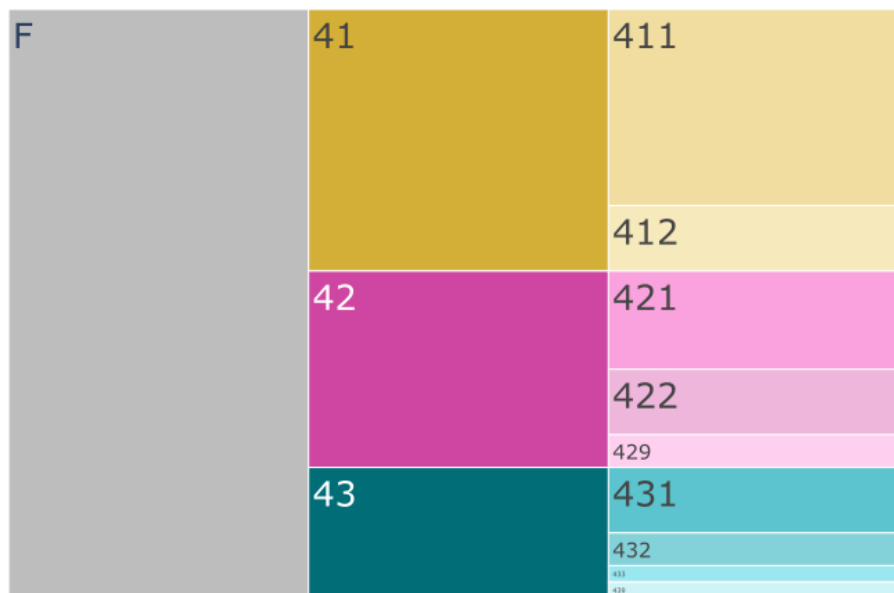


Figure 4.4: An example figure of the icicle chart. The figure is created based on the data in Table 4.1 using the Plotly Python Graphing Library. Both the hierarchical structure and node values are visualized in this figure as well as in all the rest of the figures in this section dealing with the implicit methods.

leaves [9]. However, the icicle chart is not very space-efficient [9], and therefore it does not suit for visualizing very large hierarchies. However, drill down and up interactions can alleviate this issue to some extent.

An example of the icicle chart is displayed in Figure 4.4. The figure is created based on the data in Table 4.1 using the Plotly Python Graphing Library. Since content information is easier to include in the implicit methods, both the hierarchical structure and node values are shown in this figure as well as in all the rest of the figures in this section dealing with the implicit methods. Both the hierarchical structure and content information are easy and fast to perceive from Figure 4.4, but the smallest leaves are already very difficult to distinguish even though the data set is not very large.

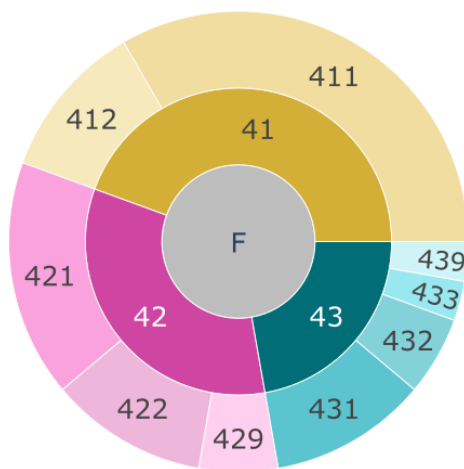
Sunburst Chart

The *sunburst chart* is a radial version of the icicle chart [14]. Solid arcs adjacent to each other are used to visualize a hierarchy [14]. The root node is at the centre, and as the hierarchical level increases, the nodes are further away from the centre. The nodes at the same level of a hierarchy are always as far away from the centre and altogether they form a circle. The widths and colors of arcs can be used to encode content information similarly to the icicle chart.

The paper [21] explains well how the sunburst chart should be interpreted and why it is more space-efficient than the icicle chart: The quantity of a target variable is displayed by the relative width

of an arc compared to the sum total width of all the arcs at the same level. Therefore, in the case of two nodes at different levels representing the same quantity of a target variable, the node at the lower level is larger than the node at the higher level. Due to this, the sunburst chart saves more space than the icicle chart. The higher levels with fewer nodes are given less space, and as the level, and for that reason number of nodes, increases, also the amount of space increases.

Figure 4.5: An example figure of the sunburst chart. The figure is created based on the data in Table 4.1 using the Plotly Python Graphing Library.



Thus, details in the lower level nodes, for example in leaf nodes, are easier to distinguish when using the sunburst chart compared to the icicle chart. On the other hand, comparing the quantities of two nodes being at different hierarchical levels is easier with the icicle chart, since the same length of a bar represents always the same quantity regardless of a level. However, it is noteworthy that in the sunburst diagram, the width of an arc can also be interpreted as an angle of a circular segment [38]. If angle is considered instead of width, it is not a relative measure anymore; the same angle of a circular segment represents the same quantity at all hierarchical levels.

The paper [39] states that the advantage of the sunburst chart is its space-efficiency combined to its ability to present the hierarchical structure very clearly. As a drawback, the paper mentions that small slices can be difficult to distinguish. However, drill down and up interactions can mitigate this problem, which was already mentioned in the case of the icicle chart. The paper [9] points out that also the sunburst chart can contain a lot of empty space despite it is regarded as more space-efficient than the icicle chart. The paper also states that the clarity of the hierarchical structure suffers compared to the icicle chart as the reading direction changes from top to bottom to inside out.

An example of the sunburst chart is shown in Figure 4.5. The figure is created based on the data in Table 4.1 using the Plotly Python Graphing Library. Both the hierarchical structure and content information are clearly displayed. Furthermore, the smallest leaves are easier to distinguish compared to Figure 4.4 of the icicle chart even though Figure 4.5 is smaller than Figure 4.4.

4.2.2 Enclosure Diagrams

In the enclosure diagrams, containment reveals the parent-child relationships. Nodes are encoded as rectangles or circles, and their place in the hierarchy is represented by their position in relation to nested rectangles or circles [14]. The enclosure diagrams are the most space-efficient method to visualize hierarchical data so they suit well for visualizing large hierarchies [18]. However, the downside of these methods is that they do not display the hierarchical structure as clearly as explicit methods or adjacency diagrams [18].

Treemap

The *treemap*, introduced for the first time in the paper [16] in 1991, encodes nodes as nested rectangles. It is created by successively dividing the visualization space into smaller rectangles as the level of a hierarchy increases [1]. The paper [21] explains well the interpretation of the treemap: Each parent node is a rectangle enclosing all of its children that are smaller rectangles. Thus, the root node is the largest and outermost rectangle enclosing all the other nodes, whereas leaf nodes are the innermost rectangles. The area

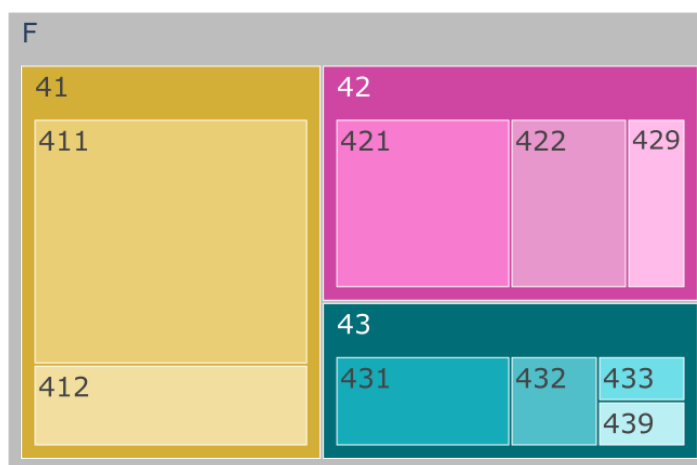
of a rectangle usually represents the value of a target variable, and in that case, the value of a target variable in a parent (the area of a parent) is the sum of the target variable values in its children (the total area of its children) [21]. Actually, to be precise, the slight paddings around nodes make the area of a parent a little larger than the total area of its children.

The area encoding makes it easy to observe the magnitudes of target variable values. Especially, it is effortless to notice which nodes are large and which nodes are small. However, comparing the values of nodes that are almost of the same size is not so intuitive, and small nodes can become very difficult to distinguish. Since nodes fill the whole visualization space and there is no empty space left, the node sizes are bigger than in the implicit methods or adjacency diagrams. Thus, there is more space to attach labels and content information to the nodes by exploiting encodings such as text, color, and texture. The paper [21] points out that the treemap is very space-efficient but it does not display the hierarchical structure very clearly. In order to alleviate this issue, for example, padding around rectangles or saturation of rectangles can be exploited [14].

It is also possible to transform the area and color attributes of the treemap by functions in order to highlight different phenomena in data [33]. *Inversion* is one option: instead of encoding the largest values of a target variable by the largest rectangles, the smallest values of a target variable could be encoded by the largest rectangles [33]. *Logarithmic* functions can also be useful [33]. These kind of transformations can obviously be used with other visualization methods as well.

The layout of the treemap can vary a lot depending on which algorithm is used to partition the visualization space. The quality of a treemap algorithm is usually measured by two criteria: the aspect ratios of rectangles and the stability of the treemap [26]. The aspect ratios should be low since very thin rectangles are difficult to distinguish. The stability of the treemap refers to how sensitive the layout is to change as the level of detail

Figure 4.6: An example figure of the treemap. The figure is created based on the data in Table 4.1 using the Plotly Python Graphing Library.



or underlying data changes [26, 38]. As data changes in an unstable layout, the position of a node can at worst change from the upper right corner to the lower left corner, and the neighboring nodes can change, whereas in a stable layout the nodes are more locked to the same position and neighbors.

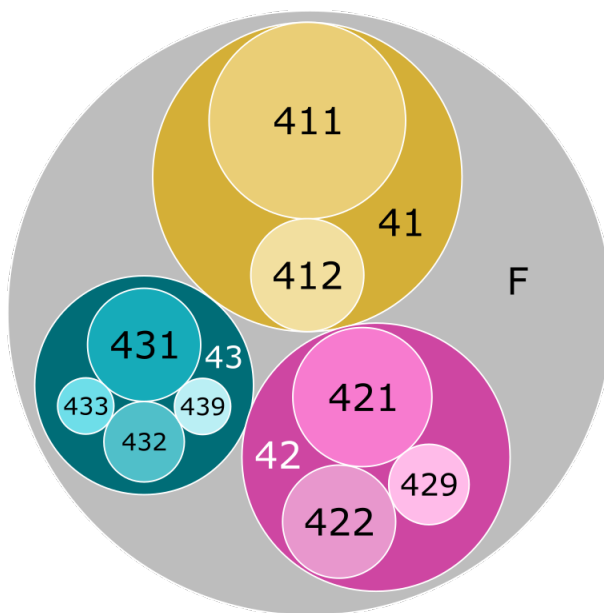
The *slice-and-dice* algorithm is the original treemap algorithm, which results in relatively stable layout but very high aspect ratios [32]. *Squarified treemaps*, using almost square rectangles, produce low aspect ratios, and therefore better readability, but only medium stability [14, 32]. For instance, the Plotly Python Graphing Library exploits squarified treemaps [30]. Other treemap algorithms are also proposed but they are not very commonly used [14].

An example of the treemap is displayed in Figure 4.6. The figure is created based on the data in Table 4.1 using the Plotly Python Graphing Library. The layout is very space-efficient, and as the data set is quite small in this case, the hierarchical structure is easy to read. The node sizes display the values of a target variable in a very clear manner.

Circular Treemap

The *circular treemap* is a version of the treemap that exploits nested circles instead of nested rectangles to visualize a hierarchy [1]. The paper [21] explains well how the circular treemap should be interpreted: Each parent node is a circle enclosing all of its children that are smaller circles. Thus, the root node is the largest and outermost circle, whereas leaf nodes are the innermost circles. The areas of circles are usually exploited to reveal the values of a target variable as in the case of the traditional treemap. However, the total area of a parent is always bigger than the summed area of its children, while in the traditional treemap they are equal. Therefore, the same area do not represent the same quantity of a target variable at different levels of a hierarchy, and areas are not comparable between different levels.

Figure 4.7: An example figure of the circular treemap. The figure is created based on the data in Table 4.1.



The areas of the nodes at the same hierarchical level can, however, be directly compared since the area of a node is relative to the sum total area of all the nodes at the

same level. In the regard that inside-level size comparisons can be made but inter-level comparisons are difficult, the sunburst chart and circular treemap are similar. The areas of the leaf nodes can directly be compared only if they are all at the same hierarchical level. Nevertheless, if the leaf nodes are at different hierarchical levels, the circular treemap could also be created in such a way that the areas of the leaf nodes are relative to the sum total area of all the leaf nodes rather than using level-wise relativeness. In this case, the leaf node areas would directly be comparable to each other, but the areas of the nodes at the same hierarchical level would not be. Which total area node areas are relative to (and which node areas are directly comparable) can be a bit confusing if it is not explained.

The advantage of the circular treemap compared to the traditional treemap is the clearer presentation of the hierarchical structure and groupings [1, 21]. This is due to the extra space between nodes that makes the containment encoding easier to distinguish [21]. On the other hand, this makes the circular treemap less space-efficient than the traditional treemap.

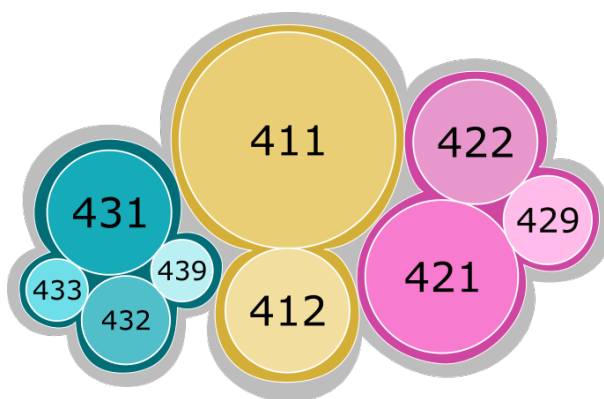
An example of the circular treemap is shown in Figure 4.7. The figure is created based on the data in Table 4.1. The hierarchical structure can be faster perceived than in Figure 4.6 but at the expense of space-efficiency of the visualization. Also, the content information is clearly displayed by node areas. All the leaf nodes are at the same hierarchical level so no problems are encountered with level-wise and leaf-wise area comparisons.

Bubble Treemap

The *bubble treemap* is another treemap version exploiting circles [21]. In the bubble treemap, only leaf nodes are circles and the rest of the hierarchical structure is shown by nested contours that conform the leaf node circles [21]. Thus, the bubble treemap saves space compared to the circular treemap but shows the hierarchical structure clearer than the traditional treemap [21].

The areas of nodes can obviously be used to encode the values of a target variable. The bubble treemap is more illustrative than the circular treemap in the sense that the area of a parent node is the sum of the areas of its children with the slight paddings. Nevertheless, the areas of the other nodes than leaf nodes can be of very dif-

Figure 4.8: An example figure of the bubble treemap. The figure is created based on the data in Table 4.1.



ferent shapes, which make their sizes difficult to compare. Furthermore, the paper [9] criticizes that the absence of a clear reading direction makes it difficult to compare nodes at the same level.

An example of the bubble treemap is displayed in Figure 4.8. The figure is created based on the data in Table 4.1. Regarding space-efficiency and the clear presentation of the hierarchical structure Figure 4.8 is a compromise between Figure 4.6 and Figure 4.7. By using a little larger paddings, there would also be space to add the character code and 2-digit codes of *Industry* into the figure, but in this case they have been omitted.

5. Visualizing Changes over Time in Hierarchical Data

There is quite a lot of discussion and literature about different visualization techniques for hierarchical data. However, only relatively little research has been conducted on how to visualize changes over time in hierarchical data, which is one of the main research problems of the thesis. In many circumstances, when a target variable is being analyzed by a hierarchical structure, the values of a target variable are not only interesting at a specific point in time, but in addition, it would be very useful to know how the target variable values have developed over time. For example, it could be examined how the world population consisting of hierarchical continent, country and city levels has changed over time or how a stock portfolio value has grown or decreased by the hierarchical industrial classification.

Adding time dimension on top of the pre-existing two dimensions of hierarchical data: the hierarchical structure and content information, makes hierarchical data richer, but on the other hand, more complicated. This type of data is difficult to visualize clearly and without causing information overload. In this chapter, a literature survey is carried out considering how time dimension can be added on top of the hierarchical visualization techniques presented in Chapter 4. In other words, the general-level research question (G_3) is answered to. However, first, it is covered in which different ways hierarchical data can change over time, and what type of changes are in the scope of this thesis.

5.1 Different Types of Changes in Hierarchical Data

Hierarchical data can change over time in multiple different ways. As already mentioned, the values of a target variable can change, but also the hierarchical structure can change if nodes are added or removed or if the arrangement of nodes alters. Moreover, if the values of a target variable change, there are two different cases according to whether the leaf node values aggregate up to the interior nodes or not [13]. For example, population in a geographical hierarchy aggregates from children up to their parent while salaries in an organizational chart do not.

The paper [13] points out that most of the scientific literature concerning tree comparison has concentrated only on changes in hierarchical structures rather than changes in target variable values, and the limited work relating to target variable changes has often utilized treemaps. The paper [13] is written in 2013 and the topic has gained more attention from those days. However, visualizing changes in target variable values of a hierarchical structure is relatively little studied topic despite its many possible practical use cases.

The paper [13] identifies five different tree comparison types, and if these types are only considered in the case of a changing tree (the same tree at different points in time) rather than considering separate trees that are compared, the five tree comparison types could be defined as follows:

Type 0. The hierarchical structure changes in such a tree whose nodes do not contain any target variable values.

Type 1. The hierarchical structure does not change but the leaf node values change in such a tree whose leaf node values aggregate up to the interior nodes.

Type 2. The hierarchical structure does not change but the leaf node values and interior node values change in such a tree whose leaf node values do not aggregate up to the interior nodes.

Type 3. Both the hierarchical structure and the leaf node values change in such a tree whose leaf node values aggregate up to the interior nodes.

Type 4. The hierarchical structure, leaf node values and interior node values change in such a tree whose leaf node values do not aggregate up to the interior nodes.

The scope of this thesis is only restricted to Type 1 changes by the request of the commissioning company whose company customer data set encounters primarily target variable changes and no changes in the hierarchical structures regardless of whether the hierarchical structure of *Industry* (E_1) or the artificial hierarchical structure created by combining multiple categorical variables (E_2) is considered. Moreover, the target variable V aggregates from the leaf nodes up to the interior nodes.

In theory, for example, the whole industrial classification system might change, which do not happen very often, or more likely, the commissioning company might receive a customer from a whole new industrial class different from any of the industrial classes of their pre-existing customers, or the commissioning company might lose their last customer

from a specific industrial class. In the aforementioned cases, the certain industrial class is only represented at another but not at the other point in time. From the point of view of the hierarchical structure of *Industry*, it means that a node is added or removed from a tree. Thus, in order to process the hierarchical data set of company customers to the fullest extent, the more complex changes of Type 3 should be dealt with. However, in this thesis, it is assumed for the simplicity, and since it mostly holds, that the hierarchical structures remain unchanged and only target variable values change over time.

According to the paper [13], target variable changes in a tree can be examined from the following dimensions:

Direction of change. Has the target variable value increased, decreased or remained the same?

Absolute change. What is the actual increase or decrease of the target variable value using the target variable units?

Percentage change. What is the absolute change of the target variable value in relation to the previous target variable value?

Comparing changes. Comparing the three aforementioned change dimensions between different nodes in a tree, for example: In which nodes the target variable value has increased or decreased? In which nodes the target variable value has increased or decreased more than in the others (either the absolute change or percentage change)?

If the hierarchical structure could change, also the fifth dimension of change should be taken into account: creating, removing or moving the nodes [13], but as already mentioned, in this thesis, the hierarchical structures are regarded as immutable.

5.2 Different Techniques for Visualizing Changes in Hierarchical Data

A lot of different hierarchical visualization techniques are presented in Chapter 4. What kind of visual encodings could be exploited to add time dimension on top of these hierarchical visualization techniques, is the topic of this section. More specifically, only Type 1 changes introduced in Section 5.1 are considered, that is, the scope is in aggregated trees in which only target variable values can change and hierarchical structures remain unchanged. The paper [12] identifies four different visual encodings that can be used to display tree differences: color, typography, interaction, and animation. One in-

tuitive option is also to use small multiples, that is, a series of similar charts. Obviously, multiple different visual encodings, such as color and typography, can also be utilized simultaneously.

There can also exist other techniques for visualizing target variable changes in hierarchical data, but this thesis only concentrates on the aforementioned visual encodings. For example, the papers [8] and [17] visualize target variable changes in hierarchical data by exploiting traditional time series type of static visualization context in which time is on the x-axis and a target variable is on the y-axis. The paper [8] implements this by using a streamgraph based visualization, while the paper [17] by using what they call "temporal treemaps".

The next subsections cover one by one how each visual encoding: color, typography, interaction, animation, and small multiples, can be used to visualize changes over time in hierarchical data, and what are the advantages and disadvantages of each option. Moreover, a lot of examples are given on how the different encodings are exploited in the academic literature.

5.2.1 Color

Color can easily be utilized to display changes in hierarchical data when using the implicit hierarchical visualization methods: the icicle chart, sunburst chart and different treemap variations, since in the implicit methods, node sizes are larger and there is more space to attach color to them compared to the explicit methods with smaller node sizes. In the case of the adjacency diagrams, the bar length (the icicle chart) or arc width (the sunburst chart) can be used to encode the latest value of a target variable, while the color of a bar or arc can be used to encode the change of a target variable from the previous point in time to the latest point in time.

Accordingly, in the case of the enclosure diagrams, the area of a rectangle or circle can be used to encode the latest value of a target variable, while the color of a rectangle or circle can be used to encode the change from the previous value. It is also possible, but maybe not as practical (since node sizes are smaller), to use color encoding with the explicit hierarchical visualization methods: the traditional node-link diagram, dendrogram and indented tree. When using the explicit methods, for instance, typography (text labels) could be used to encode the latest values of a target variable, and color of the nodes could be used to encode the changes.

Color can represent either the direction of change, absolute change or percentage change. It is a very effective encoding in giving a fast perception of in which nodes the values have increased or decreased, and roughly what are the magnitudes of changes. However, color cannot clearly display very subtle differences or exact numerical values.

Moreover, the limitation of color encoding is that a target variable value can only be compared to a single previous value, and longer time series consisting of more than two time points cannot be visualized.

Especially, when using intuitive color scales that are commonly associated with decrease and increase, such as red and green or temperature color scale varying from red to blue, color can be a very quickly perceivable encoding for visualizing changes over time in hierarchical data. However, if necessary, color scales should be designed to be colorblind-friendly, when, for example, red and green would not be an acceptable color combination.

In four different scientific papers: [13], [15], [32], and [37] color encoding is used to visualize changes in a treemap or treemap-like visualization. For example, in the paper [13], U.S. Federal Budget is arranged into a tree format by Agencies and Bureaus in such a way that each node of a tree contains the amount of money spent on a specific Bureau during a fiscal year. This tree is visualized using the treemap visualization in such a way that the area of each rectangle encodes the total dollar value indicated for a specific task, and a continuous color scale (in combination with the exact numerical values written inside the rectangles) encodes the percentage change from the previous year. In the color scale, brown and yellow gradients represent decrease and green represents increase. However, the paper criticizes the treemap for the reasons that it can only display either the absolute change or percentage change at a time but not both, and it cannot display negative target variable values by using area encoding.

The paper [37] introduces the visualization design on the website of SmartMoney Map of the Market reporting stock market data about hundreds of publicly traded companies. This visualization utilizes the treemap to display a hierarchical stock market data set. Companies are classified hierarchically by sectors and industries, and their market capitalization is revealed by the area attribute of the treemap. A continuous color scale varying from red to green indicates the percentage change from the previous market close. Black represents no change. Nevertheless, the paper mentions that they have an alternative color scale for colorblind people. The paper describes that the visualization is simple but effective, and it gives a fast but comprehensive overview of a complicated data set. Also, the paper [15] utilizes color-coded treemap in the context of stock market analysis. They use a little different version of the treemap visualization, the *treemap bar chart*, but they use analogously to the paper [37] red to green color scale to encode stock price changes.

According to the paper [13], the approach like in SmartMoney Map of the Market [37] is popular, but it lacks the possibility to display changes in the hierarchical structure. Fortunately, this approach is very suitable for visualizing Type 1 changes which are in the scope of this thesis. However, the paper [32] introduces their own version of the

treemap that can also display changes in the hierarchical structure (Type 3 changes): *Contrast Treemap* splits each rectangle of the treemap diagonally into two triangles of complementary colors. The color shade and hue as well as the areas of triangles represent changes in the hierarchical structure and changes in the target variable values. The paper [13] comments the limitations of this approach: (1) possibility to information overload since color is used to encode two kinds of changes, (2) it is only suitable for aggregated trees, and (3) it shows only moved nodes but not created and removed nodes.

It is also important to notice that color has three different dimensions: hue, brightness (or lightness), and saturation, which can be utilized to encode different things. Quantitative features can effectively be encoded by exploiting different brightness levels of the same hue and saturation [33]. If hue is constant, the color scale can be transformed into a grey scale, which removes the problem of color deficiencies [33]. With qualitative attributes, utilizing different hues while keeping brightness and saturation constant can be useful instead [33]. It is one thing, what kind of color scales are effective in emphasizing differences in data, and another thing, what kind of color scales are aesthetically pleasing. In many use cases, aesthetics also have its value and should be taken into account. For example, companies often use specific brand colors in all of their visualizations to give a impression of consistency. In any case, it is always important to attach an explanation of the used color scale into a visualization.

5.2.2 Typography

Changes in hierarchical data can simply be encoded by using typography, that is, text labels. The main advantage of typography is that it can easily be attached to any hierarchical visualization method by just writing text labels inside or beside nodes. However, because of their space-efficiency, the implicit hierarchical visualization methods can usually contain more text compared to the explicit methods. Through typography, the exact numerical values can be seen. On the other hand, typography generally cannot offer a fast but comprehensive overall understanding of data by just a glance as, for example, color encoding can do. Nevertheless, utilizing an encoding such as color in combination with typography, brings the good properties of both encodings together.

A good example of the simple use of typography in the context of visualizing changes in hierarchical data is presented in the paper [13]. The paper introduces a table visualization of hierarchical data that contains four columns: node name, node level, and absolute and percentage changes of a target variable. The cells in the column of percentage changes are colored using a red to green color scale to give a faster perception of which nodes have increased or decreased. The paper notifies that the table is insufficient in displaying the structure of the hierarchy. However, as a new development idea to the visualization, this

problem could be solved by exploiting the design of the indented tree: The column of the node names could be organized into an indented tree in which the nodes at a lower level of a tree would be at a larger indented level in the visualization, and the children of each node would be right under their parent. Otherwise the table would remain unchanged. This kind of table with nodes (or row names) arranged into an indented tree can both display the structure of the hierarchy and contain limitless amount of features attached to the nodes by just adding new columns into the table. For instance, in addition to the change columns, the latest and previous values of a target variable could be added into the table as their own columns.

The advantages of this kind of indented tree table visualization are its simplicity, which is very important with non-technical end users, and its capability to show exact numerical values of many features. One of the disadvantages of the visualization is its inefficient use of space, which could, however, be improved by hiding the lowest levels of the hierarchy and using drill down and up interactions. Another disadvantage of the table is that it does not utilize the most effective visual encoding techniques for quantitative features. The paper [20] ranks the most perceptually accurate visual encodings for the quantitative data, and for example, length and area are higher in that ranking compared to color saturation and hue. However, the table do not exploit length or area encodings as the implicit hierarchical visualization methods do. All in all, the indented tree table is a good visualization for a detailed analysis of specific nodes, but the implicit visualization methods can give a better overall understanding of data. Sorting the table by a quantitative column, could be one way to enhance the effectiveness of the visualization.

The paper [12] presents a node-link diagram that utilizes typography to visualize absolute and percentage changes of a target variable. These two values are simply written one below other beside each node. However, typography is not used alone to visualize changes; in addition, absolute changes are visualized by replacing nodes with the *Bullets*. The height of the Bullet reveals the amount of absolute change similarly to bar chart. The orientation of the Bullet in combination with color hue shows whether the change is positive or negative, and color brightness represents cardinality. The visualization can even display which nodes are created or removed by white or black borders around the Bullets. This visualization can show a lot of information, but it is only suitable for relatively little hierarchies.

As in these examples, in many situations, typography is not used alone to visualize changes in hierarchical data but in combination with another visual encoding, such as color or height, to display the exact numerical values. In a color-coded approach, such as the treemap visualization in SmartMoney Map of the Market [37], the exact numerical values could be written inside the nodes if there is enough space (as made in the treemap of the paper [13]) or in a hover tooltip, which is an interaction.

5.2.3 Interaction

User interaction can be exploited to display changes in hierarchical data. Interactions are very flexible since they can be added into any hierarchical visualization method, and they can be implemented in many different ways. Interaction can be, for instance, a button to another visualization, filter menu, adjustable timeline, highlight option, hover tooltip including text and charts, or drill down and up possibility. At least some minor interactions are included almost any visualization in combination with other encodings.

The paper [33] lists three different ways to filter nodes in the treemap in order to focus on specific features. These filtering techniques can be useful when visualizing changes in the treemap or other hierarchical visualization methods. Nodes can be filtered in such a way that only the nodes satisfying defined conditions are shown for a user [33]. When visualizing changes in a hierarchy, it might be useful to be able to see, for example, only such nodes in which the value of a target variable has decreased from the previous point in time. In addition, not showing all hierarchical levels concurrently and utilizing drill down and up capabilities [33] can release space for attaching the changes of a target variable somehow into the visualization. If a tree visualization can be zoomed, additional information can become visible only after zooming. To give an example of a scalable tree visualization, *SpaceTree* is a node-link diagram based tree browser enabling dynamic rescaling of tree branches while optimizing the fit to the screen space [23]. Furthermore, one possible filter option is to expand a specific node in order to know details of that node [33]. Each node can be a button to another visualization or reveal a tooltip when hovering over it. Details can include information about how the value of a target variable has changed over time in that specific node. It can be textual information or, for example, a time series visualization, such as the line graph about historical development.

The paper [6] presents a little different but useful interaction for the treemap. The treemap can show at most two features at the same time via area and color encodings, but the amount of features can be increased by utilizing interaction. The paper suggests that a user could change the feature encoded by color using a filter, but the feature encoded by area would remain unchanged so that the treemap layout would not change. For example, a user could select whether color encodes absolute or percentage change. This interaction could obviously be used with other hierarchical visualization methods as well. In addition, one very simple interaction is to have a slider to change the time point in a hierarchical visualization representing a single point in time. With this interaction, a user can compare the values at different time points, but as a downside, the comparison only relies on a user's memory.

Treversity2, introduced in the paper [13], is a good example of an interactive visualization tool meant for analyzing changes over time in hierarchical data. *Treversity2*

presents a novel implicit hierarchical visualization method, *StemView*, that allows comparing tree changes between two time points at multiple hierarchical levels simultaneously. The design of StemView is inspired by the icicle chart since the hierarchical levels are presented one below another as rows of equal sizes. At each level, the horizontal space is filled by boxes representing the nodes, and the width of each node reveals the latest value of a target variable. The hierarchical structure is interpreted similarly to the icicle chart. However, StemView expands the icicle chart by dividing each level vertically into two parts of equal size in order to also visualize absolute and percentage changes. The horizontal line in the middle represents zero line, and the height of each node box is defined by its percentage change from the previous value. The height is drawn upward from the zero line if the change is positive and downward from the zero line if the change is negative. The actual changes are encoded by color. Greens are used for decrease, and yellows and reds for increase, but the color scales can be altered. The created and removed nodes are encoded by white or black borders around the boxes.

StemView is a very informative visualization since it can display simultaneously the latest target variable values, absolute and percentage changes from the previous values as well as created and removed nodes while revealing the structure of the hierarchy [13]. Moreover, Treeversity2 contains line graphs about the whole historical development of a target variable in each node and a reporting tool listing outliers by typography. The line graphs are on the left side of the display space, StemView is the largest visualization in the middle, and interaction menu including various interactive options is on the right.

As mentioned, Treeversity2 offers a lot of interactive possibilities [13]. For instance, a user can change line graphs into another visualization format, *TimeBlocks*, in order to compare differential values. Also, StemView can be changed into a node-link diagram based Bullet visualization presented already in the paper [12]. A user can change the two time points being compared, and highlight particular nodes by hovering mouse over them. In addition, hovering reveals tooltips including a lot of information about each node. Changing the variable mappings is also possible. There are five possible value modifiers: the actual change, the percentage change, the latest value, the previous value, and the maximum of the latest and previous values. A user can decide how these modifiers are encoded by color, width, height and sorting order. In summary, Treeversity2 is a very useful tool that enables versatile data analysis possibilities by exploiting multiple views and interactions. On the other hand, it requires a relatively lot of familiarization, it cannot be used with very large trees, it does not have enough space for labels, and it does not cope with real-life data sets having significant outliers [13].

As discussed in this subsection, interactions have many advantages. However, there are also a couple of trade-offs between interactive visualizations and static visualizations that should be taken into account. One trade-off is between the time to carry out inter-

actions and the space that multiple static visualizations require [40]. Interactions take time, but without them the display space increases. On the other hand, if there are many interactive options, going over all possible combinations can take hours [35]. Another trade-off is between the subjectiveness which can happen as a result of interactions and the useful outcomes of customization [35]. A user can modify the visualization intentionally or unintentionally in such a way that the numbers look like desired, but this might not be the whole truth, and important things can remain unnoticed. On the other hand, customization of a visualization can be very useful if a user knows what specific things to look for or if the data set is very large. Because of the disadvantages relating to interactions, the article [35] recommends to generally avoid interactions and utilize them only when it is necessary and clearly improves the visualization.

5.2.4 Animation

Animation can be used to visualize changes over time in hierarchical data. In an animated visualization, multiple static visualizations at subsequent time points are presented one after another. Also, a transition animation shown as time point changes makes the animation smoother and easier to follow. The advantage of animation is that since only one time point is shown at a time, time dimension does not steal space from other dimensions of the visualization. For this reason, animation can be added on top of almost any visualization technique. However, creating smooth transition animations can be a challenge. The main disadvantage of animation is that detecting the changes in data relies only on a user's memory. Moreover, a user cannot watch simultaneously multiple different things in a continuously changing view. Animation can be a very captivating technique for giving a good overall understanding of data, but it is not very useful for exact analysis. However, for instance, pausing and zooming interactions can alleviate this issue.

The paper [10] covers animating the treemap and discusses about the following things that have to be taken into account in the design process: When visualizing two different treemap representations of the same hierarchy, node areas change, and nodes can be added or removed from the hierarchy. These changes can be analyzed by utilizing animation. However, some areas shrink and others expand in such a way that the transition between two views can be difficult to follow. Moreover, when not-perfectly-stable squarified treemaps are used, the algorithm often shifts some of the rectangles far from their original place, which makes the following even more complicated. Briefly explained, the squarified treemap algorithm arranges the rectangles in rows and columns, and with each rectangle, the decision is made whether the rectangle will be added at the current row or in next column (or vice versa) based on which one of the alternatives improves the aspect ratios of the treemap [5]. Thus, rectangles can leap distractingly in an animation.

However, the paper [10] tries to take the aforementioned problems into account in their animation implemented using *OpenGL*. They describe their transition animation between two tree views as "smooth". Furthermore, they try to alleviate the problem of leaping rectangles by alternating rows and columns of the treemap within a spiral counterclockwise. Due to this, rectangles rotate always to the same direction and do not leap far away from their previous location. Thus, the continuity of the animation does not break. The animated treemap visualization in the paper [10] appears to be very useful and convenient. However, the structure of the hierarchy is a bit difficult to distinguish in the example figures of the paper so it could be improved, for instance, by using larger paddings. Another option is to utilize color encoding to make the structure of the hierarchy faster to perceive: different tree branches could be colored by different hues, and nodes belonging to the same branch could be colored by different brightness or saturation levels of the same hue in order to highlight nesting of rectangles.

The animated treemap of U.S. population implemented with *D3.js* library is also an interesting example of an animated hierarchical visualization [3]. The visualization animates the absolute change of U.S. population from 1790 to 1990 by regions and states. The treemap animation is smooth and layout is stable. The whole size of the treemap expands as the time goes by and U.S. population increases. Color encoding is used to differentiate regions from one another. Moreover, region and state labels in combination with exact numerical values are attached to the figure, which is very useful. It is also possible to pause the animation. One disadvantage is that at the start of the time period, the visualization is very small and therefore difficult to read.

Treemap algorithms that can be exploited to visualize time-dependent hierarchical data must be stable while maintaining visual quality [36]. The paper [36] has carried out an extensive quantitative analysis of different rectangular treemap algorithms for time-dependent hierarchical data. They notice that the performance of the algorithm is dependent on the used data set, and they identify four features that can be used to classify the underlying data. They perform experimental tests on different algorithms using different kinds of data sets in order to help users to select the optimal treemap algorithm in each use case.

Visualizing time-dependent hierarchical data is also discussed in the paper [28]. In fact, the paper covers multivariate hierarchical data containing one to three separate target variables related to the hierarchy. The target variables are visualized by utilizing 3D versions of the sunburst chart and icicle chart. In these visualizations, width of an arc or length of a bar is used to encode the first target variable, and the height of an arc or bar the second. It is also possible to encode the third target variable using color. Changes over time in target variables are visualized by exploiting animation. Both width (or length) and height are interpolated over time so that the transitions between time points will

be smooth. Also, color transitions over time are continuous and smooth. Moreover, the visualization includes many interactive features, for instance, a user can select between 2D and 3D view, and between icicle and sunburst chart, turn off coloring, select only a specific hierarchical level, select nodes up to a specific hierarchical level, and select nodes by a target variable threshold value. A user can also examine the line chart visualizations of a selected node. The visualizations of the paper [28] are very versatile and useful since they can convey so much information: a multivariate hierarchy and its development over time. Interactive features make the system very flexible and customizable. However, there is a threat of information overload, and visualizations might appear difficult to understand for non-technical users.

5.2.5 Small Multiples

Changes over time in hierarchical data can be visualized utilizing small multiples, that is, a series of similar charts. Each individual chart represents only one specific point in time, and therefore a small multiple visualization is easy to implement by using any hierarchical visualization method. The main advantage of small multiples is that a user can analyze the distribution of a target variable clearly at distinct time points in such a way that time dimension does not clutter the hierarchical visualization which is often complex per se. A user can compare the charts representing different time points, and see in which nodes the value of a target variable has increased or decreased over time. In order to compare the values reliably, the different charts must use same units, for instance, same area should represent the same value (or the same percentage) of a target variable in all the charts. On the other hand, it can also be regarded as a downside of small multiples that none individual visualization visualizes changes per se, but the changes can only be perceived by comparing different figures.

Another advantage of small multiples is that changes in the hierarchical structure can easily be added to the visualization and highlighted, for instance, by using color. The main disadvantage of the small multiples is that the display view is divided between multiple charts, and therefore the sizes of individual charts diminish. The more charts are shown in a single view, the smaller they have to be. For this reason, a small multiple view is unpractical with very large hierarchies that become difficult to distinguish in a smaller size. Nevertheless, different interactions can improve small multiples, for instance, zooming and a possibility to select how many and which time points are shown in a view.

The paper [33] implements an experimental visualization utilizing a small multiple view of the treemap to visualize six months of financial data of the product hierarchy. The top-down treemap algorithm is exploited to create the figures. In the treemaps, the area of a rectangle encodes the revenue of the products, and the color of a rectangle encodes

the profit. The users were able to compare positions, sizes and colors of the rectangles between the six months. In addition, the users were given an "animated" version of the same visualization including a slider to change the month as an alternative visualization scheme. The users preferred the "animated" version based on the layout of the view, information content and capabilities. Thus, maybe it can be concluded, that the data set was too large to be visualized using small multiples or there were too many months displayed in a single view. A small multiples view can easily become cluttered.

Analogously, the paper [11] exploits a small multiple view of the treemap to visualize as much as 14 trees in a single view according to their example figure. The visualization is meant for comparing the structures of the hierarchies rather than target variable values, and the trees do not represent different time points but different classification versions of the same taxonomy. Despite these differences, the visualization is a good example of utilizing small multiples in the context of hierarchies.

BarcodeTree (BCT), introduced in the paper [18], is a visualization technique for comparing both the structures and target variable values of multiple hierarchies by utilizing small multiples. It could be exploited to visualize time-dependent hierarchical data. Each tree is visualized using a technique reminding of a barcode. Therefore, each tree takes only a single row and multiple trees can be stacked vertically. Moreover, matching nodes are in the same horizontal position which makes the comparison between different trees easier. In BCT, each node is encoded using a rectangle, and the rectangles are arranged horizontally side by side. There are two different versions of BCT: In *BCT_w*, the hierarchical level of a node is encoded by the width of a rectangle, and height and color of a rectangle can be used to encode target variable values. In *BCT_h* instead, the hierarchical level of a node is encoded by the height of a rectangle, and color of a rectangle can be used to encode target variable values.

Since the BCT visualization technique is relatively simple, it is suitable for trees with only small amount of nodes and few hierarchical levels [18]. On the other hand, the advantages of BCT are that a large number of trees can be presented one below other with only small vertical distances from each other. Actually, the results of an experiment in the paper suggest that the two versions of BCT are better for visual comparison of trees than the icicle chart because of the smaller vertical distances between trees. Moreover, the small multiples view includes multiple interactions that help analyzing the hierarchical structures and comparing trees.

6. Designing the Experimental Visualizations

Now that the literature survey answering to the general-level research questions (G_1), (G_2), and (G_3) has been conducted, a vast amount of information has been acquired on different hierarchical visualization methods as well as on different techniques for adding time dimension on top of these methods. Moreover, various example visualizations from scientific papers visualizing target variable changes in hierarchical data have been covered. By utilizing all this gathered information, useful and appropriate visualization prototypes can be designed to answer to the experimental research questions (E_1) and (E_2). In addition, the commissioning company can exploit all the acquired information for solving their future hierarchical visualization problems.

In this chapter, the design process of the visualization prototypes answering to the experimental research questions (E_1) and (E_2) is covered. As stated in Section 2.4, the nested model [22] is utilized in the design and validation of these prototypes. Each step of the nested model is addressed in its own section.

6.1 Domain Problem and Data

The first step of the nested model is to identify the domain problem and data [22], in other words, answer to the following two questions: What are the exact problems of the target audience, that is the end users of the visualizations, in the vocabulary used in the commissioning company? What kind of data do the commissioning company have for solving these problems? To begin with the first question, the definite problem area was framed in cooperation with the commissioning company by discussing about their ongoing projects, and what are the most important but less-researched aspects in their customer data investigating of which would help them to understand their customers better and improve their operations. In this discussion, the end users of the visualizations were carefully listened to, and they validated the final problem characterization. This upstream validation is a crucial part of the process since the main threat during the domain problem step of the nested model is that the problems of the target audience

become mischaracterized, which in the worst case, can eventually lead to completely useless final product [22].

The discussion ended up to the domain problem of investigating changes in a continuous variable V by the hierarchical structures: *Industry* in (E_1) and an artificial hierarchy formed by combining multiple categorical features in (E_2) . In this characterization, also the data becomes identified, but to further specify the data, the data set of all the company customers of the commissioning company contains a time series of a continuous V , a hierarchical variable *Industry*, and multiple categorical features for each company customer. The data set is more specifically described in Section 2.2.

6.2 Data Types and Operations

The second step of the nested model is to map the domain problem and data from the domain vocabulary into the vocabulary of information visualization [22]. In other words, the domain problem is abstracted into data types and operations. The main threat during this step is bad data type and operation abstraction, which inevitably leads to bad visualizations [22], and therefore also this step was carried out carefully in cooperation with the commissioning company. By operations, the nested model refers to abstract tasks such as filter, search, and sort [22]. In the case of the prototypes, drill down and up are very important operations, which emerged in the discussions with the end users of the visualizations. The end users highlighted that they need to examine the hierarchical data at all of its levels. The highest non-root level of the hierarchy gives a good overall understanding of all the customers, while the lower levels enable deeper analysis on specific customer segments. It is very useful that a visualization view can be drilled down and up in such a way that either a larger set or smaller and more specific set of customers is shown in a single view depending on what information needs to be acquired.

Moreover, the raw data must be abstracted into such data types that the data will be in an appropriate form to be given as an input for the next step of the nested model: visual encoding and interaction [22]. This includes first considering which is the sensible form for the data, and after that, implementing the data wrangling, which, in this thesis, was done by utilizing the Python library *Pandas* [24]. To explain briefly the structure of the raw data, the data set of company customers included originally 8 different columns: *Time Point*, *Customer ID*, the main variable of interest V , *5-digit Code of Industry*, *Business Entity*, *Region*, *Size*, and *Turnover*. The descriptions and data types of these variables are introduced in Table 2.1.

The commissioning company instructed that for the visualization prototypes of the thesis, it would be sensible to include only two separate time points instead of longer time series. As the first data wrangling step, this filtering was done for the data. Two years,

2020 and 2021, were decided to be compared with one another. Two separate data sets were created in such a way that the other data set represented the data set of company customers for the year 2020 and another for the year 2021. Thus, the *Time Point* column was omitted. The two data sets included all the columns of the original data set (except *Time Point*), and the name of the column *V* was changed to *V 2020* or *V 2021* according to the year of the data set.

Two columns in the data set of 2020: *Customer ID* and *V 2020*, were merged with the whole data set of 2021 by the common column *Customer ID* by including only the *Customer IDs* of 2020 that also exist in 2021. In the combined data set, for the *Customer IDs* that exist only in 2021 but not in 2020, the value in the column *V 2020* was set to zero. Finally, the *Customer IDs* that exist only in 2020 but not in 2021 were filtered from the data set of 2020, and by including all the columns, these rows were added to the combined data set by setting the value in the column *V 2021* to zero. By this way, the combined data set contained all *Customer IDs* from 2020 and 2021 regardless of whether a *Customer ID* occurs only in 2020, only in 2021 or in both years. In the categorical columns, the information from 2021 was used if it existed; otherwise the information from 2020 was used. Moreover, the two columns *V 2021* and *V 2020* were subtracted from one another in order to get a new column: *Difference of V*. Now, a new data set, *V 2020 and 2021*, had been created.

Starting from that point, the data wrangling for (E_1) and (E_2) separated, and these processes are presented in their own subsections. However, one more decision regarding both experimental research questions had to be made: which one of the change dimensions introduced in Section 5.1 would be used in the visualizations. We discussed about this with the commissioning company and ended up choosing absolute change as the change dimension of the prototypes. One reason behind this decision was that using absolute changes in a hierarchical visualization contains one interesting and useful property: the absolute changes can be aggregated by summing from the lower hierarchical levels up to the higher hierarchical levels, and in these aggregations, the changes of the same sign strengthen one another, while the negative and positive changes neutralize one another. Through this property, it is possible to observe how the changes at lower level nodes actually affect to the higher level nodes and the whole customer base. For instance, multiple small positive changes in a lower hierarchical level can aggregate up to a large and significant positive change in the next level higher, or a certain anomalous negative change in a lower hierarchical level can have only a small effect on the next level higher level if the changes in the other nodes at the same level neutralize and diminish its effect.

6.2.1 Data Wrangling for (E_1)

For (E_1), the data set V 2020 and 2021 had to be transformed into a form containing hierarchical customer segments by the hierarchical variable *Industry*. Together with the commissioning company, we decided to omit all the other categorical variables in the data set, and therefore, multiple columns were dropped. Only the columns *Customer ID*, *5-digit Code of Industry*, V 2020, V 2021, and *Difference of V* were selected. After this, the data set was grouped by the column *5-digit Code of Industry* in such a way that the values in the columns V 2020, V 2021, and *Difference of V* were summed and the number of different *Customer IDs* in 2021 was counted. This data set was named V *Industry*.

In order to create the hierarchical structure for the data, an additional data set relating to *Industry*, named *TOL 2008*, had to be utilized. The commissioning company provided this data. The data set included 10 different columns. The first column in *TOL 2008* was *5-digit Code* containing all the existing 5-digit codes in the industrial classification. The next four columns represented into which 4-digit code, 3-digit code, 2-digit code or character code class each 5-digit code belongs to. The rest five columns contained labels for each of the code levels, for example, if the *3-digit Code* column had an entry of 932, the *3-digit Label* column would have an entry of 932 - *Amusement and recreation activities*.

The code columns were extracted from *TOL 2008*, and this data set was transformed into a *code-parent-form* in which all the possible *Industry* codes from all the hierarchical levels were listed one below another in the *Code* column, and a parent code of each code was presented in the *Parent* column. For instance, the parent of the code 01110 is 0111, and the parent of the code 0111 is 011. For the character codes, the value in the *Parent* column was set to *Total*. A new data set, *Industry Code-Parent*, was created.

Own data sets considering each of the hierarchical level of *Industry* had to be created. The data set $D(5\text{-digit})$ was created by left-merging V *Industry* with *Industry Code-Parent* by the column *5-digit Code of Industry* in V *Industry* and the column *Code* in *Industry Code-Parent*. The data set $D(5\text{-digit})$ contained six columns: *Code*, *Parent*, V 2020, V 2021, *Difference of V*, and *Customer Count*. The data set $D(4\text{-digit})$ was created by dropping the *Code* column from the data set $D(5\text{-digit})$, and by renaming the *Parent* column to *Code*. After this, the data set was grouped by the *Code* column in such a way that the values in the columns V 2020, V 2021, *Difference of V*, and *Customer Count* were summed. Moreover, this data set was left-merged with *Industry Code-Parent* by the common column *Code*, and analogously to $D(5\text{-digit})$, the new data set $D(4\text{-digit})$ contained six columns: *Code*, *Parent*, V 2020, V 2021, *Difference of V*, and *Customer Count*. Correspondingly, $D(3\text{-digit})$ was aggregated from $D(4\text{-digit})$, $D(2\text{-digit})$ from $D(3\text{-digit})$, and $D(\text{Character})$ from $D(2\text{-digit})$.

Finally, all these five data sets having equal column names were concatenated into a new data set named $D(E_1)$. The label columns in the original *TOL 2008* data set were organized one below another in the same order as the *Code* column in $D(E_1)$, and added as a new column, *Label*, to $D(E_1)$ to explain the codes in the *Code* column. Moreover, the total row was added into $D(E_1)$ by summing the values of all the numerical columns in the data set $D(5-digit)$. In that point, $D(E_1)$ was a complete data set to be visualized. The final form of $D(E_1)$ is presented in Table 6.1.

The data set $D(E_1)$ in Table 6.1 is not the real data set of $D(E_1)$ but a completely artificially generated example data set having similar structure to the real data set. The values of V and *Customer Count* do not represent the real values of the commissioning company nor are derived from them. Therefore, the distribution of customers among different *Industries* can appear unrealistic, for example, a lot of customers and a relatively large value of V from a very rare *Industry* class. Rather than analyzing the real domain problem, the example data set is meant for presenting the visualization prototypes created originally by using the real data of the commissioning company.

Table 6.1: The structure of the data set $D(E_1)$. The values of V and *Customer Count* are artificially generated and do not represent the real values of the commissioning company nor are derived from them.

<i>Code</i>	<i>Label</i>	<i>Parent</i>	<i>V 2020</i>	<i>V 2021</i>	<i>Difference of V</i>	<i>Customer Count</i>
01110	01110 - Growing of cereals (except rice), leguminous crops and oil seeds	0111	0.20	0.06	-0.14	14
...
U	U - Activities of extraterritorial organisations and bodies	Total	0.09	0.05	-0.04	1
Total	Total		473.81	476.20	+2.39	5,395

6.2.2 Data Wrangling for (E_2)

For (E_2), the data set *V 2020 and 2021* had to be transformed into a form containing hierarchical customer segments by multiple categorical features. We discussed with the commissioning company, which these categorical variables would be, and in which order

they would be used to create the artificial hierarchy. We ended up selecting five categorical variables that were used in the following order to create the hierarchy: *Region*, *Size*, *Character Code of Industry*, *Turnover*, and *Business Entity*. In other words, the customers were first divided into customer segments by *Region*, these customer segments were additionally partitioned by *Size*, then by *Character Code of Industry*, *Turnover*, and finally by *Business Entity*.

The data set *V 2020 and 2021* included the following nine columns: *Customer ID*, *5-digit Code of Industry*, *Business Entity*, *Region*, *Size*, *Turnover*, *V 2020*, *V 2021*, and *Difference of V*. Since the 5-digit code of *Industry* had to be replaced by the character code, the aforementioned data set was left-merged with the two columns, *5-digit Code* and *Character Code*, in *TOL 2008* by the common column *5-digit Code of Industry / 5-digit Code*. After this, the column *5-digit Code of Industry* was dropped from the merged data set, and the *Character Code* column was simply renamed to *Industry*.

The aforementioned data set was first grouped by all the possible combinations of the five features: *Region*, *Size*, *Industry*, *Turnover*, and *Business Entity* in such a way that the values in the columns *V 2020*, *V 2021*, and *Difference of V* were summed and the number of different *Customer IDs* in 2021 was counted. A new column *Business Entity ID* was added to the data set by combining the values of all the five features in each row, for example, one entry in this column could be: *Ahvenanmaa | Small | A | < €0.5M | Ay*. Analogously, new columns *Turnover ID*, *Industry ID*, and *Size ID* were added to the data set in such a way that the respective entries would be *Ahvenanmaa | Small | A | < €0.5M* (in the column *Turnover ID*), *Ahvenanmaa | Small | A* (in the column *Industry ID*), and *Ahvenanmaa | Small* (in the column *Size ID*). *Region* did not require ID column since it was the second highest level in the hierarchy after the root node. This data set was named *V Multiple Features*.

Own data sets considering each hierarchical level in *V Multiple Features* had to be created. The data set *D(Business Entity)* was formed by selecting only the following seven columns from *V Multiple Features*: *Business Entity ID*, *Business Entity*, *Turnover ID*, *V 2020*, *V 2021*, *Difference of V*, and *Customer Count*. The column *Business Entity ID* was renamed to *ID*, *Business Entity* to *Label*, and *Turnover ID* to *Parent*. After this, *D(Turnover)* was formed by selecting only the following seven columns from *V Multiple Features*: *Turnover ID*, *Turnover*, *Industry ID*, *V 2020*, *V 2021*, *Difference of V*, and *Customer Count*. This data set was grouped by *Turnover ID* in such a way that the values in the columns *V 2020*, *V 2021*, *Difference of V*, and *Customer Count* were summed, and from the values in the columns *Turnover* and *Industry ID*, the first values were selected (regarding these columns, each group contained a constant value defined by a specific *Turnover ID*). The column *Turnover ID* was renamed to *ID*, *Turnover* to *Label*, and *Industry ID* to *Parent*. Analogously to *D(Turnover)*, the data sets *D(Industry)*, *D(Size)*,

and $D(\text{Region})$ were formed from the data set V *Multiple Features*. However, the data set $D(\text{Region})$ did not contain the *Parent* column, but this was afterwards added to the data set, and its value was set to *Total* in each row.

Finally, all these five data sets having equal column names were concatenated into a new data set named $D(E_2)$. Moreover, the total row was added into $D(E_2)$ by summing the values of all the numerical columns in the data set $D(\text{Business Entity})$. In that point, $D(E_2)$ was a complete data set to be visualized. The final form of $D(E_2)$ is presented in Table 6.2.

Analogously to the data set $D(E_1)$, the data set $D(E_2)$ in Table 6.2 is not the real data set of $D(E_2)$ but a completely artificially generated example data set having similar structure to the real data set. The values of V and *Customer Count* do not represent the real values of the commissioning company nor are derived from them. Therefore, the distribution of customers among certain variable combinations can appear unrealistic, for example, a lot of customers and a relatively large value of V from a very rare variable combination. Rather than analyzing the real domain problem, the example data set is meant for presenting the visualization prototypes created originally by using the real data of the commissioning company. Moreover, the values or value magnitudes of V and *Customer Count* are not, in any sense, compatible with the data set of $D(E_1)$.

Table 6.2: The structure of the data set $D(E_2)$. The values of V and *Customer Count* are artificially generated and do not represent the real values of the commissioning company nor are derived from them.

<i>ID</i>	<i>Label</i>	<i>Parent</i>	<i>V 2020</i>	<i>V 2021</i>	<i>Difference of V</i>	<i>Customer Count</i>
Ahvenanmaa Small A < €0.5M Ay	Ay	Ahvenanmaa Small A < €0.5M	1.12	0.89	-0.23	1
...
Varsinais-Suomi	Varsinais-Suomi	Total	87,394.82	87,322.99	-71.83	4,224
Total	Total		1,662,029.04	1,662,515.89	+486.85	84,480

6.3 Visual Encoding and Interaction

Now that the domain problem has been characterized, the data types and operations have been defined, and the data has been transformed into a correct form, the third, and in this case the last, step of the nested model: designing the visual encoding and interaction techniques [22], can be implemented. The main threat during this step is that the used encoding and interaction techniques turn out to be ineffective at conveying the desired information [22]. A simple upstream validation method to avoid this threat is to justify the chosen encoding and interaction techniques [22], and therefore this section concentrates on justifying why the different design decisions were made.

6.3.1 Requirements

I decided to create two alternative visualization prototypes answering to (E_1) and utilize these same prototypes for answering to (E_2) by changing the underlying data. Therefore, the visualization prototypes were designed in such a way that they are suitable for both data sets: $D(E_1)$ and $D(E_2)$. Nevertheless, these data sets are very similar in many respects so this was not an actual issue. The commissioning company requested that the 2021's values of V in combination with the value changes from 2020 would be the two main dimensions in the prototypes, while the 2020's values of V and *Customer Count* would only be additional details. An operational wish was drill down and up possibility as already mentioned since showing all the five hierarchical levels in a static view would make the visualization too cluttered. However, the commissioning company added that at least a couple of hierarchical levels should be displayed simultaneously in a single view without a need to drill.

In order to select the most suitable visual encodings, the characteristics of the data sets $D(E_1)$ and $D(E_2)$ were analyzed: First, both of the hierarchies are very large since they contain five hierarchical levels and a great number of nodes. The artificial example data set of $D(E_1)$ contains 1,806 nodes, and the corresponding data set of $D(E_2)$ contains 100,421 nodes. The real data sets of the commissioning company contain slightly fewer nodes since some of the possible customer segments in $D(E_1)$ and $D(E_2)$ are not represented among their customers. Especially, many of the feature combinations in the artificial hierarchy of $D(E_2)$ are not very sensible, for instance a very large number of employees (*Size*) but a minimal *Turnover*, so many of the 100,421 nodes were actually omitted. Anyway, the selected visualization technique should be able to display large hierarchies. Second, the values of V differ a lot between different nodes so the selected visualization technique should be able to clearly display nodes of different sizes so that also the smaller nodes can be distinguished. Third, the absolute differences can be posi-

tive or negative, and the sign of a change as well as its magnitude should be apparent and effortlessly observable in the visualization. Fourth, the values of V as well as the absolute differences aggregate by summing as going from a lower hierarchical level up to a higher, and the selected visualization technique should highlight this.

6.3.2 Selecting the Suitable Hierarchical Visualization Techniques

The two most appropriate hierarchical visualization techniques were selected from the alternatives presented in Chapter 4. Since the implicit visualization methods are better in displaying content information compared to the explicit methods, only them were considered as possible options. This was a reasonable choice since both of the hierarchies, $D(E_1)$ and $D(E_2)$, are very large and include two target variable dimensions that have to be encoded: 2021's value of V and its absolute difference from the previous year. In order to select two clearly different techniques, one adjacency diagram and one enclosure diagram were picked. The adjacency diagrams show the structure of the hierarchy more clearly, while the enclosure diagrams are more space-efficient. Thus, they have different advantages that can be compared in the context of this use case.

From the two alternative adjacency diagrams: the icicle chart and sunburst chart, the icicle chart is not as space-efficient as the sunburst chart for which reason the sunburst chart suits better for large hierarchies. Moreover, by comparing Figures 4.4 and 4.5, it can be noticed that the sunburst chart is better in displaying small node sizes distinguishably. According to these criterion, the sunburst chart was selected as the first visualization technique for the prototypes. Its angle (i.e. width) encoding suits also well for highlighting the aggregation of data since the same angle represents the same quantity at all levels.

From the three alternative enclosure diagrams: the traditional, circular and bubble treemap, the circular treemap does not visualize the aggregation of data very effectively by its area encoding since the circle areas are not comparable between different hierarchical levels. Therefore, it is left out of consideration. The traditional treemap and bubble treemap are both very space-efficient so they suit well for large hierarchies and are able to display also small node sizes distinguishably. Moreover, in both charts, the same area represents the same quantity at all levels (taking into account slight paddings), and therefore they are effective in highlighting the aggregation of data. Thus, both methods appear to be good options. However, in the bubble treemap the areas of other nodes except leaf nodes have vague shapes which make the areas difficult to compare with each other. On the other hand, the bubble treemap emphasizes the structure of the hierarchy better than the traditional treemap. In this use case, the ability to compare the target variable values encoded by area is considered as a little more important feature than maximizing

the clear hierarchical structure, and therefore the traditional treemap is selected as the second visualization technique for the prototypes. The Plotly Python Graphing Library utilizes squarified treemap algorithm resulting in low aspect ratios and good readability. Nevertheless, it could also be interesting to test the bubble treemap in the future.

Based on the literature survey conducted in Section 5.2 (which is not completely all-encompassing but contains a lot of practical examples), the treemap appears to be a very frequently used hierarchical visualization technique when visualizing time-dependent hierarchical data, while the sunburst chart does not appear to be as common. Therefore, there is also an interesting aspect of comparing common and less common techniques with each other. Examining the usability of the sunburst chart with time-dependent hierarchical data is a relatively novel and less researched point of view. It is noteworthy that the more used technique might not necessarily be the best one.

The arc width in the sunburst chart and the area of a rectangle in the treemap can be used to encode the 2021's values of V . However, if there exists a node in which the value of V 2021 is zero, then the arc width or rectangle area also becomes zero. This means that the node in question is not drawn in the figure at all even if it had a non-zero value of V in 2020. Because of this limitation, the customer segments that are only represented in 2020 but not in 2021 are not visible in the prototypes even though these negative changes might be interesting. Anyway, after the hierarchical visualization techniques were selected, it had to be decided which encoding techniques would be used to encode time dimension, *Difference of V* , in the prototypes. This selection process is presented in the next subsection.

6.3.3 Selecting the Suitable Time Encoding Techniques

I decided to utilize the same time encoding techniques for both prototype versions, the sunburst chart and treemap. The most appropriate time encoding techniques were selected from the alternatives presented in Section 5.2. The selection process began by excluding the unsuitable techniques: Since the prototypes of the thesis are meant for specific and exploratory data analysis rather than conveying a fast overview of the data, a more static visualization is preferred over animation. Moreover, animation would suit better for a little longer time series consisting of more than only two distinct time points. Thus, it was not considered as an option in this use case. Small multiples could be exploited to present the two separate years in their own figures side by side so that they could be compared with each other. However, a small multiple visualization is not very effective for large hierarchies since the separate figures can become difficult to distinguish in a single view. In addition, the absolute differences would not be directly visible in this kind of visualization, but they should be derived and interpreted by comparing a certain node

between two figures. Therefore, the signs and magnitudes of the changes could not be observed effortlessly. For these reasons, the small multiples view was also excluded.

After rejecting animation and small multiples, the remaining alternatives were color, typography and interaction. I decided to combine the advantages of each them by using them all together in the prototypes. Color was selected as the main time encoding technique since color-coded arcs and rectangles enable observing the signs and magnitudes of the absolute changes very fast and effortlessly without cluttering the view. A red-white-green color scale was selected because of its intuitivity and connotations with decrease and increase. This color scale is obviously not colorblind-friendly, and a different color scale should be selected if color deficiencies had to be taken into account. Color encoding is also a good option for the reason that it highlights the aggregation of the absolute changes. For instance, multiple light green nodes (small positive changes) at a lower level of the hierarchy aggregate up to a darker green parent node (larger positive change), or an equally weighted green (positive) node and red (negative) node neutralize each other and aggregate up to a white (no change) parent node.

The exact color codes of the color scale were selected by adjusting the *HSL* color values (hue, saturation, lightness). First, the base hues of red and green were selected: the bright red $HSL(0^\circ, 100\%, 50\%)$ and bright green $HSL(120^\circ, 100\%, 50\%)$. The first value with a degree sign defines the hue. The second value, 100%, means that the saturation is at its maximum level. The third value, lightness level of 50%, corresponds to the completely bright color without mixing any white or black into it. The lightness levels of 0% and 100% correspond to completely black and white respectively. Rather than utilizing only these two colors in combination with white, the gradient color scale with five steps was designed so that the color scale could encode larger set of different values distinguishably. If only three color steps are used, two negative or two positive nodes might seem to be almost of the same color even if representing different magnitudes of changes. Additional two color steps to the color scale make the magnitude differences to distinguish more clearly. Therefore, one additional red and green were selected.

The additional red and green were added to the ends of the color scale by mixing black to the selected bright colors. The dark red color $HSL(0^\circ, 100\%, 27\%)$ and dark green color $HSL(120^\circ, 100\%, 27\%)$ were created from the bright colors by keeping hue and saturation unchanged and decreasing lightness. In other words, the first version of the gradient color scale was formed from the following colors in the following order: the dark red $HSL(0^\circ, 100\%, 27\%)$, bright red $HSL(0^\circ, 100\%, 50\%)$, white $HSL(0^\circ, 0\%, 100\%)$, bright green $HSL(120^\circ, 100\%, 50\%)$, and dark green $HSL(120^\circ, 100\%, 27\%)$. This color scale was a lightness symmetric color scale since the lightness values were symmetric on the red and green side of the color scale. However, the green side of the color scale turned out to be a little misleading since the bright green intuitively appears to be more positive than

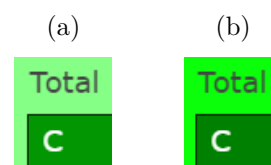


Figure 6.1: The selected color scale (a) is compared to the lightness symmetric color scale (b). In the lightness symmetric color scale, the bright green might intuitively appear more positive than the dark green. The selected color scale is designed to be more intuitive in this respect. The red scale is equal between the two figures. See also Figure 6.2.

the dark green even though it is not. For this reason, the green side of the color scale was altered in such a way that the lightness level of the bright green was increased from 50% to 75%, but the dark green remained unchanged. The commissioning company regarded this new color scale as more illustrative compared to the lightness symmetric color scale so it was selected as the color scale to be used in the prototypes. The comparison of these alternative color scales is illustrated in Figures 6.1 and 6.2. The downside of the selected color scale is that the greens being close to the center of the color scale are very light and can be a little difficult to distinguish from one another.

As already mentioned, also typography and interaction were utilized as additional time encoding techniques in the prototypes. In order to encode more detailed information, a hover tooltip, a combination of interaction and typography, was attached to each node. The tooltip contains the label of a node as well as the specific numerical values of V_{2020} , V_{2021} , $Difference\ of\ V$, and $Customer\ Count$. The advantage of tooltips is that they can contain a lot of useful information for a detailed analysis but do not require any extra space. Moreover, drill down and up interactions were added into the prototypes so that only a couple of hierarchical levels are displayed simultaneously, and the other levels can be examined by drilling. Thanks to this interaction, the prototypes are a lot more easier to read and understand. Showing all the five hierarchical levels simultaneously would lead to information overload.

Figure 6.2: The color of $Total$ represents smaller positive change than the color of C . The selected color scale (a) encodes this more intuitively compared to the lightness symmetric color scale (b).



6.4 Final Visualization Products

Based on the aforementioned justified design decisions, the visualization prototypes were created using the Plotly Python Graphing Library. The algorithm design step of the nested model was omitted since the ready-made algorithms of the Plotly Python Graphing

Library were utilized. The final visualization products are introduced in this section. Simultaneously, a downstream validation method of the visual encoding and interaction step is carried out in the form of a qualitative image analysis of the final visualization products [22].

The real data sets of the commissioning company are not used in the example figures presented in this section nor in the appendices. Completely artificial data sets were generated for the example figures of the thesis.

6.4.1 Prototypes for (E_1)

The final two prototypes answering to the experimental research question (E_1) are introduced in this subsection. Both prototypes, $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$, visualize the variable V in 2021 and the difference of V from 2020 to 2021 by the hierarchical variable *Industry*. The initial view of the first prototype, $Sunburst_{(E_1)}$, is presented in Figure 6.3. Moreover, a full-page version of this figure can be found in Appendix A in Figure A.2. In this prototype, V 2021 is encoded by arc width and *Difference of V* by color. Moreover, the arcs are illustratively sorted by width. The initial view of the prototype displays the character code level and 2-digit code level of *Industry* as well as the total sum in the middle. In other words, the middle node and next two hierarchical levels branching from it are shown in a single view.

The rest of the hierarchical levels can be examined by drilling down the hierarchy. When a specific node is pressed, the view transforms to show the selected node in the

V in 2021 and the Difference of V from 2020 to 2021 by Standard Industrial Classification TOL 2008

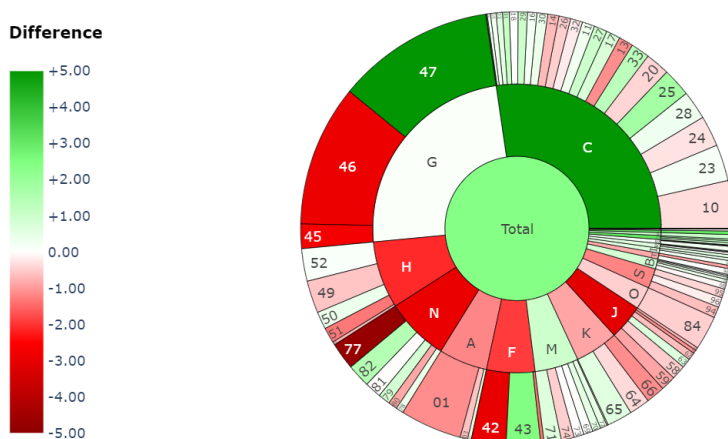


Figure 6.3: The initial view of $Sunburst_{(E_1)}$ displays the character code level and 2-digit code level of *Industry* as well as the total sum in the middle.

V in 2021 and the Difference of V from 2020 to 2021 by Standard Industrial Classification TOL 2008

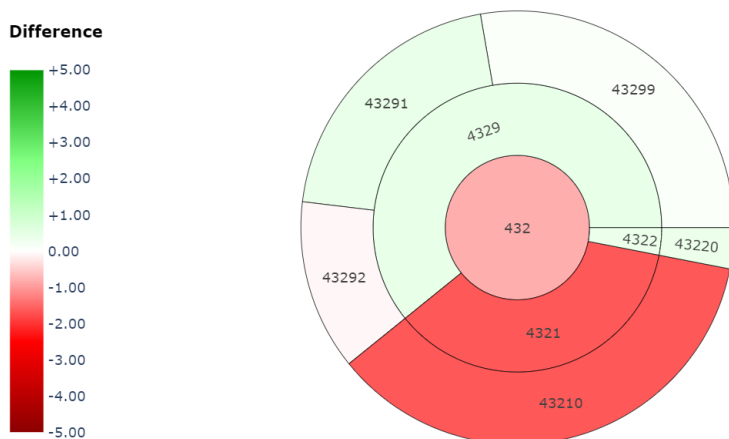


Figure 6.4: The drilled view of $Sunburst_{(E_1)}$ displays the selected 3-digit node of *Industry*, 432 , as well as all the 4-digit and 5-digit nodes branching from it.

middle surrounded by the next two hierarchical levels branching from it. In addition, the Plotly Python Graphing Library runs a stylish and illustrative transition animation during drilling. Figure 6.4 presents the drilled view of $Sunburst_{(E_1)}$. The 3-digit node 432 is selected by drilling a few steps down from the initial view. All the 4-digit and 5-digit nodes branching from 432 are displayed in the figure. Drilling up can be done by pressing the middle node. Then the parent of the previous middle node becomes the new middle node and the two surrounding hierarchical levels update. Also, a more comprehensive drilling example of $Sunburst_{(E_1)}$ demonstrating all the possible phases from the initial view down to the most specific hierarchical level is presented in Appendix A in Figure A.1.

The initial view of the second prototype, $Treemap_{(E_1)}$, is presented in Figure 6.5. Moreover, a full-page version of this figure can be found in Appendix B in Figure B.2. In this prototype, V_{2021} is encoded by the area of a rectangle and *Difference of V* by color. The initial view of the prototype displays the character code level and 2-digit code level of *Industry* as well as the total sum in the largest rectangle enclosing all the other rectangles. In other words, the outermost rectangle and next two hierarchical levels branching from it are shown in a single view.

Analogously to $Sunburst_{(E_1)}$, the rest of the hierarchical levels can be examined by drilling down the hierarchy. Also for the treemap, the Plotly Python Graphing Library has designed a beautiful transition animation for drilling. Figure 6.7 presents the drilled view of $Treemap_{(E_1)}$. The same node as in Figure 6.4, 432 , is selected by drilling a few steps down from the initial view. Again, all the 4-digit and 5-digit nodes branching from 432 are displayed in the figure. Drilling up can be done by either pressing the outermost

V in 2021 and the Difference of V from 2020 to 2021 by Standard Industrial Classification TOL 2008

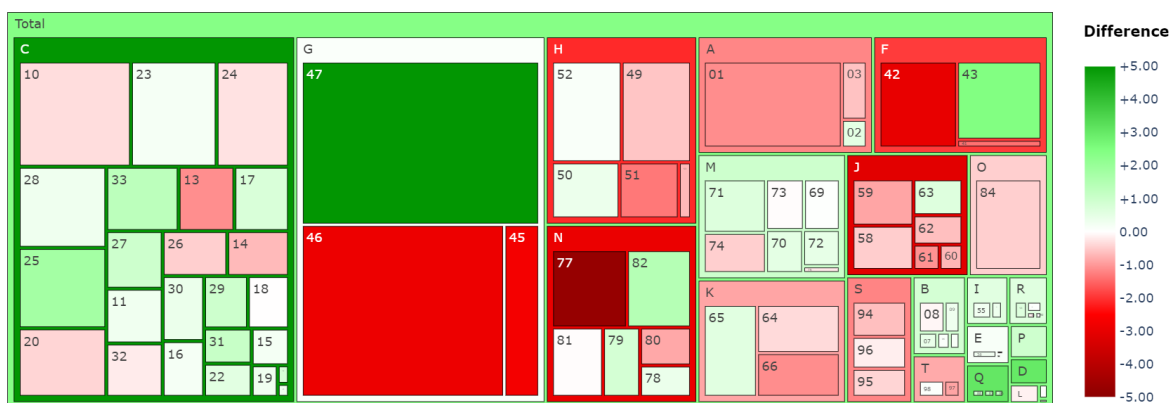
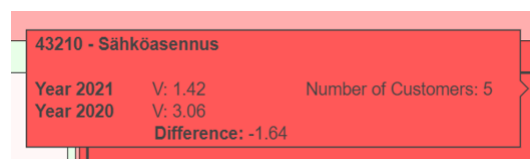


Figure 6.5: The initial view of $Treemap_{(E_1)}$ displays the character code level and 2-digit code level of *Industry* as well as the total sum in the largest rectangle enclosing all the other rectangles.

rectangle or some of the nodes in the top bar. The top bar is a very useful navigation tool in the treemap implementation of the Plotly Python Graphing Library, but it has unfortunately not been implemented for the sunburst chart. Also, a more comprehensive drilling example of $Treemap_{(E_1)}$ demonstrating all the possible phases from the initial view down to the most specific hierarchical level is presented in Appendix B in Figure B.1.

The two prototypes, $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$, have similar hover tooltips. An example tooltip in Figure 6.6 contains detailed information about the node 43210. The background color of the tooltip is illustratively the color of the absolute difference, and the title of the tooltip is the node label consisting of the *Industry* code and name (in this example, the name is in Finnish). Moreover, the tooltip contains the exact numerical values of V 2020, V 2021, *Difference of V*, and *Customer Count*.

Figure 6.6: $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$ have similar hover tooltips. This example tooltip contains detailed information about the node 43210.



As a qualitative image analysis of the figures presented in this subsection, it can be said that both prototypes, $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$, answer well to the experimental research question (E_1). Both prototypes succeed in depicting simultaneously the hierarchical structure of *Industry* and the development of V over time. The prototypes are implemented in a very clear and effective manner, and they are easy to understand. Even though the hierarchical structure is large, the selected space-efficient hierarchical

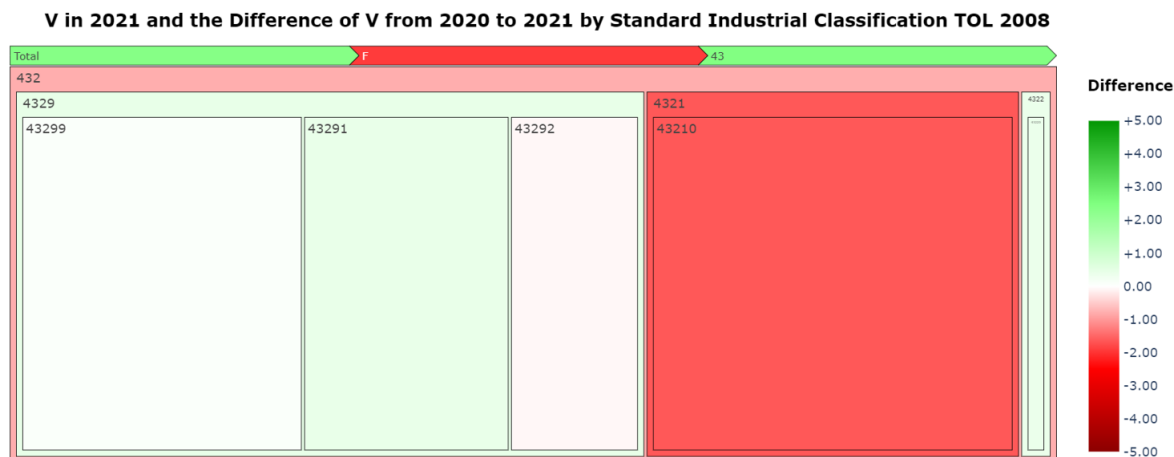


Figure 6.7: The drilled view of $Treemap_{(E_1)}$ displays the selected 3-digit node of *Industry*, 432, as well as all the 4-digit and 5-digit nodes branching from it. The selected node is the same as in Figure 6.4.

visualization techniques in combination with drill down and up interactions prevent information overload. The arc width and rectangle area encodings reveal very distinctly how V_{2021} is distributed among the hierarchical industrial classification. The color encoding is very effective in displaying the signs and magnitudes of *Difference of V*. Furthermore, the aggregation of the values of V_{2021} and *Difference of V* across different hierarchical levels is very apparent in both prototypes. Moreover, the tooltips offer very useful detailed information. All these things considered, both prototypes appear to be very potential to be utilized in practice. To compare the two prototypes with each other, $Sunburst_{(E_1)}$ displays the structure of the hierarchy a bit more clearly, but on the other hand, the smallest nodes are easier to distinguish in $Treemap_{(E_1)}$. In addition, navigation might be a bit easier and faster with $Treemap_{(E_1)}$ that contains the useful top bar navigation tool contrary to $Sunburst_{(E_1)}$.

6.4.2 Prototypes for (E_2)

The final two prototypes answering to the experimental research question (E_2) are introduced in this subsection. Both prototypes, $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$, visualize the variable V in 2021 and the difference of V from 2020 to 2021 by the artificial hierarchy formed by combining the following five features in the following order: *Region*, *Size*, *Character Code of Industry*, *Turnover*, and *Business Entity*. These prototypes are otherwise equal to $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$, but the underlying data has been changed. Therefore, the encoding and interaction techniques of $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$ are equal to the ones of $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$ respectively (see Subsection 6.4.1).

V in 2021 and the Difference of V from 2020 to 2021 by Region, Size, Industry, Turnover, and Business Entity

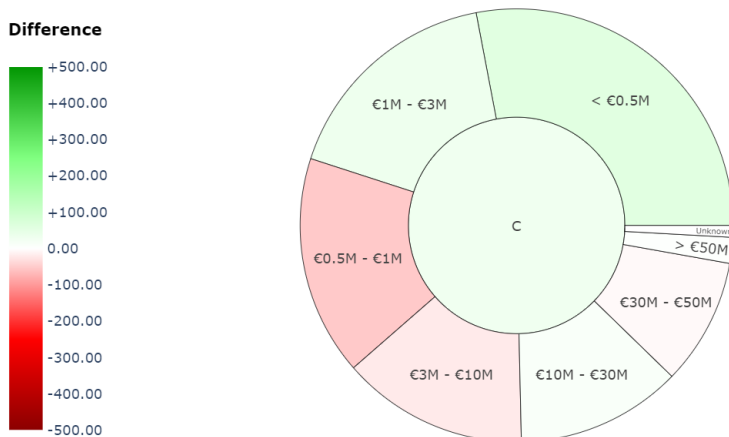


Figure 6.9: The drilled view of $Sunburst_{(E_2)}$ displays the customers that simultaneously belong to the following three categories: *Keski-Suomi* (*Region*), *Small* (*Size*), and *C* (*Character Code of Industry*) as well as the branching *Turnover* classes.

of V by color. The initial view of the prototype displays the *Region* and *Size* levels of the artificial hierarchy as well as the total sum in the largest rectangle enclosing all the other rectangles. Displaying two hierarchical levels in addition to the outermost rectangle was not a problem with the treemap that is a more space-efficient visualization technique compared to the sunburst chart.

Drilling down and up in $Treemap_{(E_2)}$ work equally to $Treemap_{(E_1)}$. Figure 6.12 presents the drilled view of $Treemap_{(E_2)}$. The figure shows the customers that simultaneously belong to the following three categories: *Keski-Suomi* (*Region*), *Small* (*Size*), and *C* (*Character Code of Industry*) as well as the branching *Turnover* classes similarly to the Figure 6.9. However, since two hierarchical levels are showing in addition to the outermost node, also the branching *Business Entity* classes are visible contrary to Figure 6.9. In addition, a more comprehensive drilling example of $Treemap_{(E_2)}$ demonstrating all the possible phases from the initial view down to the most specific hierarchical level is presented in Appendix D in Figure D.1.

The two prototypes, $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$, have similar hover tooltips. The tooltips are very similar to the ones of

Figure 6.10: $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$ have similar hover tooltips. This example tooltip contains detailed information about the node *Keski-Suomi* | *Small* | *C* | $\text{€}0.5\text{M} - \text{€}1\text{M}$ | *Oy*.



$Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$: the background color of the tooltip is the color of the absolute difference, and the tooltip contains the exact numerical values of V 2020, V 2021, $Difference$ of V , and $Customer$ Count. An example tooltip in Figure 6.10 contains detailed information about the node *Keski-Suomi | Small | C | €0.5M - €1M | Oy*. The title of the tooltip reveals the values of *Region*, *Size*, *Character Code of Industry*, *Turnover*, and *Business Entity* that are combined to create the customer segment in question.

As a qualitative image analysis of the figures presented in this subsection, it can be said that both prototypes, $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$, answer well to the experimental research question (E_2). Both prototypes succeed in utilizing hierarchical visualization techniques for visualizing changes over time in V by multiple different categorical features. Five categorical features are combined into an artificial hierarchy after which equal visualizations to $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$ are exploited. Thus, the qualitative evaluation of $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$ in the end of Subsection 6.4.1 also apply to $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$ and is not unnecessarily repeated.

However, observations concerning exclusively $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$ can also be made. Regarding the experimental research question (E_2), the treemap succeeds in conveying more information in a single view compared to the sunburst chart. $Sunburst_{(E_2)}$ displays only one hierarchical level in addition to the middle node, while $Treemap_{(E_2)}$ shows two hierarchical levels and the outermost node. With a very large hierarchy and long node labels, the sunburst chart becomes more easily cluttered than the treemap, which in this case was fixed by dropping one hierarchical level out of the sunburst chart. Moreover, when drilling deeper down the hierarchy, it is easier to keep track of all the

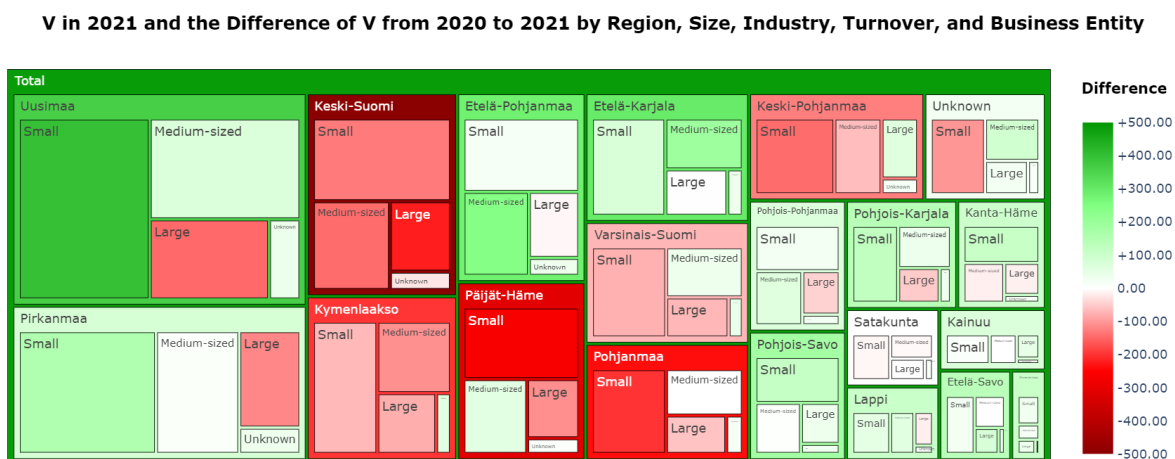


Figure 6.11: The initial view of $Treemap_{(E_2)}$ displays the *Region* and *Size* levels of the artificial hierarchy as well as the total sum in the largest rectangle enclosing all the other rectangles.

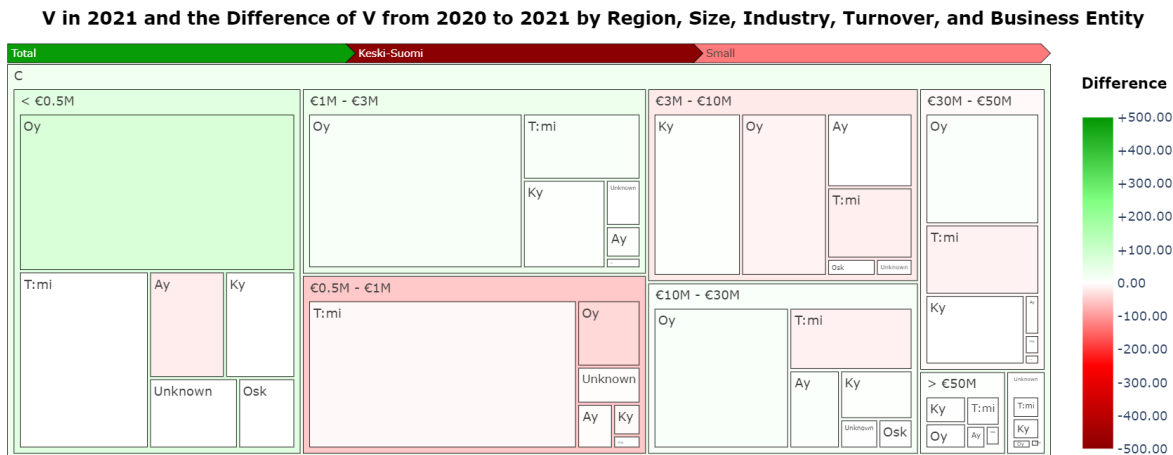


Figure 6.12: The drilled view of $Treemap_{(E_2)}$ displays the customers that simultaneously belong to the following three categories: *Keski-Suomi* (*Region*), *Small* (*Size*), and *C* (*Character Code of Industry*) as well as the branching *Turnover* and *Business Entity* classes. The selected node *Keski-Suomi | Small | C* is the same as in Figure 6.9.

selected features with $Treemap_{(E_2)}$ than $Sunburst_{(E_2)}$ thanks to the navigation top bar listing all the selected features. With $Sunburst_{(E_2)}$, a user cannot tell the already selected values of higher hierarchical levels without checking the tooltips.

When evaluating $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$, it should also be considered whether creating the artificial hierarchy generally was a useful technique for combining multiple categorical features in a single visualization or not. It appears that interpreting the artificial hierarchies of $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$ is quite straightforward and easy, and these prototypes visualize the customer segments formed by multiple features clearly and effectively. Especially, if the chosen order of the features is reasonable in the domain context, these kind of prototypes can be very useful. But if the features should be possible to be combined more flexibly in different orders, these kind of prototypes can turn out to be too focused on only one perspective.

7. Validating the Experimental Visualizations

In this chapter, the prototypes for (E_1) and (E_2) presented in Section 6.4 are validated and discussed. The validation emphasis is on the prototypes for (E_1) by the request of the commissioning company, and (E_2) is treated as more of an additional research question. In the nested model, a user study is another downstream validation method of the visual encoding and interaction step in addition to a qualitative image analysis that was carried out in Section 6.4 [22]. Thus, an end user study was arranged of the two prototypes for (E_1) . Since the time was limited, it was not possible to arrange own user study of the prototypes for (E_2) . However, the participants of the end user study also quickly tested the prototypes for (E_2) and commented them briefly.

Moreover, Munzner’s paper mentions an expert review as one possible validation method of the visual encoding and interaction step [22]. An expert review could be exploited as an upstream validation method during designing the visualizations, but in this case, it was exploited as a downstream validation method. With the expert review, more comments also regarding the prototypes for (E_2) were gathered, but the questions focused on the prototypes for (E_1) .

With the end user study, it was examined how non-technical business users can exploit the visualizations, and what are their opinions about them, while with the expert review, also more analytical opinions about the visualizations were collected. It is very useful to listen to both the end users of the visualizations and data science experts in order to receive reliable results of the usefulness of the visualizations.

7.1 End User Study

The end user study was meant for the end users of the visualizations, and its purpose was to measure the clarity and real-world applicability of $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$. Seven non-technical business persons from the commissioning company, who might benefit from the prototypes as a part of their work, participated in the study. Since the number of participants is very small, any statistically significant results cannot be received. However,

the commissioning company is interested in learning the signals of the potential end users, and the results of this study can be regarded as indicative of a broader user study. The study consisted of an introduction to the topic area and prototypes, experimental user test measuring time spent on certain tasks, and a couple of qualitative questions. After these steps, the participants were also asked to answer to the System Usability Scale (SUS) questionnaire [34].

7.1.1 Structure of the End User Study

The experimental user test consisted of two separate tests, one test regarding each prototype. Both tests included six tasks that needed to be answered to by exploiting the prototype assigned for the test. The tasks were relatively complex search problems in which a respondent needed to drill down and up in the hierarchy and compare certain nodes either by *V 2021* or *Difference of V*. The test questions between the two tests did not concern the same exact nodes to avoid the learning effect, but the two tests were designed to be as similar as possible. Before the participants started taking the actual tests, the prototypes were introduced to them; what information is showing, how the different encodings should be interpreted, and how do the interactions work. Moreover, the participants took a practice test containing similar tasks to the actual tests so that the task instructions would be clear when taking the actual tests. During the practice test, the participants were also allowed to ask questions.

After the introduction and practice phases, the participants took the two user tests one after another, and their time spent on each test was measured. One threat that might corrupt the test results is that after completing the first test, the participants are a little more experienced in taking the second test. This learning effect was tried to be mitigated by familiarizing the participants with the tasks beforehand using the aforementioned practice test and alternating which one of the actual tests is taken first. Half of the respondents took first the test on *Sunburst_(E₁)*, while the other half took first the test on *Treemap_(E₁)*. Since there were an odd number of respondents, actually three people started with *Sunburst_(E₁)* and four people with *Treemap_(E₁)*. The participants were asked to complete the tests as fast as possible but avoid wrong answers.

After completing both tests, the users were asked to answer to the following qualitative questions: (1) Which one of the prototypes was your favorite and why? (2) If you had to mention one property that was better in your least favorite prototype, what would it be? I had three hypotheses regarding the experimental user test: (H_1): Completing one test takes at most five minutes. (H_2): The participants complete the test on *Sunburst_(E₁)* faster the test on *Treemap_(E₁)*. (H_3): *Sunburst_(E₁)* is the favorite prototype of the participants.

The purpose of (H_1) was to measure the complexity of the prototypes. The limit of five minutes was estimated to be such that a user has a reasonable amount of time to read and understand the tasks and a reasonable amount of time to search the answers from the visualization. If many participants required more than five minutes to complete one test, that would mean that the time spent on searching is very long, and the visualization could be considered as overly complex and difficult to use. Instead, if the test results supported the hypothesis and most of the participants could complete one test faster than in five minutes, the prototype could be considered as simple enough to be utilized in practice. The purpose of (H_2) and (H_3) was to compare the two prototype versions by user performance and preference respectively. It was supposed that *Sunburst*_(E_1) performs better and is also the favorite prototype of the participants since the sunburst chart displays the hierarchical structure more clearly than the treemap, and this should be an advantage in the test tasks that were related to searching certain nodes in the hierarchy as fast as possible.

After the experimental user study, the participants completed the SUS questionnaire including 10 qualitative questions rated using a 5-step scale from Strongly agree to Strongly disagree [34]. The purpose of SUS is to measure the usability of the designed system, for instance software or application, and it has become a very commonly used tool referenced in over 1300 publications [34]. The reasons why SUS was selected to be utilized in this thesis include that it offers reliable outcomes even with small sample sizes and it is very effective in revealing whether the system is usable or not [34]. In the SUS questionnaire, the respondents were instructed to consider the prototype version which was their favorite. However, in many regards, the two prototype versions are identical, and the results of SUS can be regarded as considering the shared properties of both prototype versions and answering to the question: Is it useful to utilize these type of hierarchical visualizations in this particular domain context? Scoring the results of SUS was implemented by following the instructions of the article [4]. The answers of each respondent were converted into a SUS score having a range of 0 to 100. However, the scores should not be interpreted as percentages [34].

7.1.2 Results of the End User Study

The results of the experimental user test are presented in Table 7.1. The time spent and the number of correct answers are listed for each participant regarding both tests: the test on *Sunburst*_(E_1) and the test on *Treemap*_(E_1). Moreover, Table 7.1 contains the information on which test was taken first, which test was completed faster, how many seconds was the difference, and which one of the prototypes was the user's favorite. In the last row, average time and score are calculated for both tests.

Table 7.1: The results of the experimental user test.

ID	<i>Sunburst</i> _(E₁) Time & Score	<i>Treemap</i> _(E₁) Time & Score	First Test	Faster Test	Favorite Prototype
ID1	3:01 & 6/6	2:41 & 5/6	<i>Sunburst</i> _(E₁)	<i>Treemap</i> _(E₁) (0:20 faster)	<i>Treemap</i> _(E₁)
ID2	2:29 & 6/6	3:23 & 6/6	<i>Treemap</i> _(E₁)	<i>Sunburst</i> _(E₁) (0:54 faster)	<i>Sunburst</i> _(E₁)
ID3	2:49 & 6/6	2:23 & 5/6	<i>Sunburst</i> _(E₁)	<i>Treemap</i> _(E₁) (0:26 faster)	<i>Sunburst</i> _(E₁)
ID4	2:41 & 6/6	3:30 & 6/6	<i>Treemap</i> _(E₁)	<i>Sunburst</i> _(E₁) (0:49 faster)	<i>Sunburst</i> _(E₁)
ID5	3:53 & 6/6	4:02 & 6/6	<i>Sunburst</i> _(E₁)	<i>Sunburst</i> _(E₁) (0:09 faster)	<i>Sunburst</i> _(E₁)
ID6	2:30 & 6/6	3:11 & 6/6	<i>Treemap</i> _(E₁)	<i>Sunburst</i> _(E₁) (0:41 faster)	<i>Sunburst</i> _(E₁)
ID7	4:20 & 5/6	4:24 & 6/6	<i>Treemap</i> _(E₁)	<i>Sunburst</i> _(E₁) (0:04 faster)	<i>Sunburst</i> _(E₁)
Average	3:06 & 5.86/6	3:22 & 5.71/6			

This data sample clearly supports the hypothesis (H_1) since all the respondents completed both tests faster than in five minutes. Also the average time was a lot shorter than five minutes for both tests: the average time spent on the *Sunburst*_(E₁) test was 3:06 and the average time spent on the *Treemap*_(E₁) test was 3:22. Thus, there is a relatively strong evidence that the prototypes are simple enough to be utilized in practice.

Regarding the hypothesis (H_2), it cannot be drawn any conclusions about the performance differences between the two prototypes. The time difference between the faster test and slower test is not very significant with any participant. Neither there is a significant difference between the average times spent on the tests. Respondents made very few mistakes on both tests. Five people completed faster the test on *Sunburst*_(E₁), while two people completed faster the test on *Treemap*_(E₁). However, all the other participants except one completed the second test faster than the first which implicates that the learning effect has affected the results. Thus, there is no significant support for the claim that either one of the tests was generally completed faster than the other.

Instead, there is relatively strong support for the hypothesis (H_3) since six out of seven respondents regarded *Sunburst*_(E₁) as their favorite prototype. When these six respondents were asked why *Sunburst*_(E₁) was their favorite prototype, they told the following reasons: "the hierarchical structure of TOL 2008 was more clearly displayed", "more

logical structure and faster to read”, ”more logical and the different industrial classes were easier to find”, ”easier to learn, the sizes were easier to perceive, and the visualization was more pleasant”, ”easier to read”, and ”especially finding the right character level was easier”. When the same people were asked what would be the one property that was better in *Treemap*_(E₁), they told the following features: ”easier return navigation”, ”the value of V was easier to perceive from the rectangles”, ”it was easier to compare the sizes of different classes”, ”the nested rectangles were clear” [this comment does not really compare the two prototype versions], and ”takes better advantage of the screen space”. The only respondent who considered *Treemap*_(E₁) as the favorite prototype, reasoned the decision as follows: ”maybe a little easier to use, the traveled path was more clearly displayed, and it was more apparent whether a certain class was red or green”. This respondent commented that one property that was better in *Sunburst*_(E₁) was the following: ”it was easier to tell which class is larger than the other, especially at the more precise levels”.

Based on the end user feedback, the following conclusions can be made: It appears that *Sunburst*_(E₁) was more preferred especially for the reason that it displays the hierarchical structure more clearly, and therefore the nodes being searched are easier to find. *Sunburst*_(E₁) was also considered as a more logical prototype and either easier or faster to read by many respondents. The end users apparently regarded the top bar navigation tool as one main advantage of *Treemap*_(E₁) since it offers easier return navigation and keeps better track on the traveled path compared to *Sunburst*_(E₁). Moreover, one respondent commented that *Treemap*_(E₁) utilizes the screen space more efficiently. It is difficult to conclude with which prototype it is easier to perceive and compare the node sizes, since there were comments on behalf of both prototypes. There was only one comment about the colors, and according to this respondent, the colors were easier to perceive when using *Treemap*_(E₁). Accordingly, there was only one respondent who commented which one of the prototypes was visually more pleasant, and it was *Sunburst*_(E₁).

The participants had also an opportunity to leave informal comments and development ideas about *Sunburst*_(E₁) and *Treemap*_(E₁). One respondent suggested that the labels of the nodes at the character code level could contain the names of the industrial classes in addition to the mere character codes since no one remembers the character code meanings by heart. Other participant wrote that he had not seen these kind of hierarchical visualizations including drill down and up interactions before, and in his opinion, the visualizations were implemented very well. He also added that it was easy to learn how to use the drill down and up interactions. A couple of participants pointed out that the tooltip implementation was a bit disturbing as it often emerged unnecessarily to fill the screen space. Moreover, one respondent commented that it should be considered whether all the hierarchical levels are relevant or whether the visualizations could be simplified by removing a couple of hierarchical levels from the view.

The results of SUS are promising. The average SUS score is 83.9 which can be regarded as a very good result. If a SUS score is more than 68, it is above average [34]. The article [2] presents different rating scales for average SUS scores. By the adjective rating (*Worst Imaginable, Poor, Ok, Good, Excellent, Best Imaginable*) the average SUS score of 83.9 receives the second best rating *Excellent*. By the traditional school grade scale (*F, D, C, B, A*), this result receives the second best grade *B*, and by the acceptability scoring (*Not Acceptable, Low Marginal, High Marginal, Acceptable*), the best score *Acceptable*.

As already mentioned, the end user study did not address the prototypes $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$ due to lack of time. However, the participants quickly tested these prototypes and commented them briefly. They considered the prototypes as very useful and inventive. One of the participants commented that the prototypes could definitely be utilized in practice. He thought that these visualizations succeeded very well in combining multiple feature dimensions into a single view, which is a very useful property in customer analysis. Another respondent regarded the prototypes as even more useful than $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$.

7.2 Expert Review

Two data science experts from the commissioning company familiarized themselves with the prototypes for (E_1) and (E_2) and answered to an analytical questionnaire concerning the prototypes. The validation emphasis was on the prototypes for (E_1) by the request of the commissioning company, but also a couple of questions concerning the prototypes for (E_2) were asked. The expert review is very valuable for the commissioning company since the experts have a very good understanding both on data science and the specific needs of the business, and thus they are the most competent persons to review the implementation and usefulness of the visualization prototypes. The experts gave very diverse and interesting comments on the prototypes, which could certainly be utilized when considering the choice between two alternative prototype versions or when further developing the prototypes. Therefore, the whole expert review is documented in the following two subsections so that the commissioning company can return to these comments also in the future. The expert review is divided into six distinct parts by different topic areas.

7.2.1 Expert Review of $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$

In this subsection, the two prototypes for (E_1) are reviewed. In summary, the experts considered the prototypes as useful and well-implemented, and thought that the selected encoding and interaction techniques work well. In addition, they believed that non-technical end users would learn to use the prototypes relatively fast. They had different

opinions on which one of the prototype versions should be selected to be utilized in practice. Thus, maybe it could be concluded that also subjective preferences affected the results of this review, and there does not exist a clear answer to the question of whether $Sunburst_{(E_1)}$ or $Treemap_{(E_1)}$ should be selected as a final visualization version to be utilized in the commissioning company. However, the experts agreed that the value of V is easier to perceive with $Sunburst_{(E_1)}$, but $Treemap_{(E_1)}$ is clearer and easier to use.

Part 1: Concerning the Shared Properties of $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$

1. How useful for the commissioning company would it be to follow V and changes of V over time by the hierarchy of TOL 2008?

Exp.1: In my opinion, absolutely important in terms of expanding the business. Following these kind of changes helps us to understand our markets, and this way we are able to react early, for example, regarding the strategy.

Exp.2: Useful. It will definitely tell something about the attractiveness of the commissioning company if in the industry X we are expanding, while in the industry Y we are shrinking.

2. How useful exactly these kind of prototypes are for following V and changes of V over time by the hierarchy of TOL 2008 (for example, compared to a table containing the same information)?

Exp.1: An interactive visualization is almost always better than a table, in my opinion. The largest changes cannot be highlighted as clearly in a table. Moreover, a table often cannot respond to specific user needs.

Exp.2: Great, since the value magnitudes are more intuitively displayed compared to a table. On the other hand, it always takes some time to get used to a new type of visualization. But if the visualization becomes used on a regular basis, that will not be a problem.

3. In the prototypes, the arrangement of nodes encodes the hierarchical structure of TOL 2008, arc width or rectangle area the value of V in 2021, and color the absolute change of V from the previous year.

- (a) How well do these three encoding techniques work in the implemented prototypes if considering each encoding separately?
- (b) How well does the combination of these three encoding techniques into a single visualization work, or is there information overload?
- (c) What is your opinion about the used color scale?

Exp.1: (a) In my opinion, color is the best encoding method for depicting the changes of V , and area or width for depicting the volume of V . I think that using these encoding techniques for these purposes is the clearest way to visualize this data considering users of all skill levels.

(b) I believe that a non-technical end user can become a little confused when a single view includes this much information.

(c) The color scale is illustrative but could it be reasonable to reduce the number of colors, for instance, to only three green tones and three red tones in addition to white? White could represent "No significant change", and the color tones could represent only significant changes. I would also be interested to see the similar visualization with percentage changes.

Exp.2: (a) They work well.

(b) These prototypes offer very deep-level information, which is useful. However, in addition to these visualizations, simpler and more general-level visualizations are needed.

(c) The color scale works well.

4. (a) **What is your opinion about the interactions of the prototypes (drill down/up and tooltips)?**

(b) **Could the interactions somehow be improved?**

Exp.1: (a) In my opinion, they work very well. This kind of implementation can respond to the needs of many different users.

(b) The usability of the visualization might withstand adding a couple of filters into the view. I do not believe that the visualization would become too cluttered.

Exp.2: (a) They work very well.

(b) There could be an info box about the existence of drilling. It could help some of the users.

5. (a) **What do you think: How clear these prototypes would be for a non-technical end user?**

(b) **How quickly do you think a non-technical end user would learn to use these prototypes?**

(c) **What do you think: How much support a non-technical end user would need from a technical person when using these prototypes?**

Exp.1: (a) I believe that with a few improvements the prototypes would be clear enough for a non-technical end user.

(b) I do not think there would be any problems with learning. A user might have more problems with finding the real business needs to follow these things and changing his/her actions based on the observations.

(c) Hardly any support, I believe.

Exp.2: (a) When a non-technical person sees these prototypes for the first time, it can require a bit of work to understand them.

(b) I believe that everyone would learn to use the prototypes after a couple of minutes of exploring.

(c) No support from a technical person would be needed. However, short instructions embedded into the view could be added.

6. Can you imagine a whole different way to visualize the same information?

Exp.1: No, I cannot.

Exp.2: I cannot imagine any better way at least. Certainly, the information could be presented in a table or hierarchical table, but the table would contain mainly numbers the perception of which is not as intuitive as the perception of colors or sizes.

Part 2: Comparing $Sunburst_{(E_1)}$ and $Treemap_{(E_1)}$

7. (a) Is the hierarchical structure of TOL 2008 easier to perceive with $Sunburst_{(E_1)}$ or $Treemap_{(E_1)}$?

(b) Why?

Exp.1: (a) $Sunburst_{(E_1)}$

(b) It is easier to perceive what are the subindustries of a certain industry. Moreover, it can easily be seen that the number of industrial classes increases as moving to an outer circle. In the treemap version, the eye does not perceive the structure immediately.

Exp.2: (a) $Treemap_{(E_1)}$

(b) It groups the industrial classes more clearly into separate boxes.

8. (a) Is the value of V in 2021 (arc width or rectangle area) easier to perceive with $Sunburst_{(E_1)}$ or $Treemap_{(E_1)}$?

(b) Why?

Exp.1: (a) $Sunburst_{(E_1)}$

(b) Different industrial classes can more clearly be juxtaposed when they are

presented on the same "axis". In the treemap version, the eye has to search the boxes from different locations.

Exp.2: (a) *Sunburst*_(E₁)

(b) There is no big difference between the two prototypes. However, comparing the different industrial classes being at the same hierarchical level is a bit easier with the sunburst version because the classes are closer to each other.

9. (a) Is the navigation easier with *Sunburst*_(E₁) or *Treemap*_(E₁)?

(b) Why?

Exp.1: (a) *Treemap*_(E₁)

(b) The animation is a bit clearer, and therefore it is easier to perceive inside which box the view shifted during drilling.

Exp.2: (a) There is no difference.

(b) –

10. (a) Which one of the prototype versions, *Sunburst*_(E₁) or *Treemap*_(E₁), is clearer and easier to use?

(b) Why?

Exp.1: (a) *Treemap*_(E₁)

(b) Equal reasons to the question 9(b).

Exp.2: (a) *Treemap*_(E₁)

(b) The lower hierarchical levels are displayed more clearly. In the sunburst version, the lower hierarchical levels become very narrow slices.

11. (a) Which one of the prototype versions, *Sunburst*_(E₁) or *Treemap*_(E₁), is visually more pleasant if the clarity and ease of use are not taken into account?

(b) Why?

Exp.1: (a) *Sunburst*_(E₁)

(b) Round shapes look more modern.

Exp.2: (a) I cannot tell. Both are equally pleasant.

(b) –

12. (a) Which one of the prototype versions, *Sunburst*_(E₁) or *Treemap*_(E₁), would you select to be utilized in practice?

(b) Why?

Exp.1: (a) *Sunburst*_(E₁)

(b) It is easier to compare the values of V between different industrial classes and the appearance is more modern.

Exp.2: (a) *Treemap*_(E₁)

(b) A slight preference towards the treemap version, but I also liked some of the properties of the sunburst chart.

Part 3: Informal Feedback on *Sunburst*_(E₁) and *Treemap*_(E₁)
13. Would you like to give informal feedback on *Sunburst*_(E₁) or *Treemap*_(E₁)?

Exp.1: The character level industrial classes could be labeled by illustrative icons instead of plain characters. It is easy to invent simple small icons for these classes with the help of which a user could immediately know which character level industry is under inspection. Without the icons a user has to check the tooltips continuously in order to remember the selected class. No one remembers the TOL 2008 classification by heart. I am not sure how the lower level classes could be improved.

Exp.2: I would like to see how the sunburst chart would look like as a little larger version so that there would be more space for the outer circles.

7.2.2 Expert Review of *Sunburst*_(E₂) and *Treemap*_(E₂)

In this subsection, the two prototypes for (E_2) are reviewed. In summary, both experts were a bit sceptic about the usefulness of the artificial hierarchy, and regarded the possibility to change the order of the features in the hierarchy as an extremely important development idea for the prototypes. Moreover, the experts believed that the artificial hierarchy would be more difficult to adopt by a non-technical end user compared to the hierarchy of *Industry*. Again, they had different opinions on which one of the prototype versions should be utilized in practice.

Part 4: Concerning the Shared Properties of *Sunburst*_(E₂) and *Treemap*_(E₂) and Comparing to the Prototypes for (E_1)

14. (a) How useful for the commissioning company would it be to follow V and changes of V over time by the artificial hierarchy formed by combining the following five features in the following order: *Region*, *Size*, *Character Code of Industry*, *Turnover*, and *Business Entity*?

- (b) **By which one of the two hierarchies, TOL 2008 or the artificial hierarchy, would it be more useful to follow V and changes of V over time?**

Exp.1: (a) The commissioning company should follow the changes happening in different customer segments very closely. However, only following the changes is not enough, but the reasons behind these changes should also be visualized. After this, a user should change his/her actions based on the observations.

(b) In my opinion, the changes of V could be followed by a simpler artificial hierarchy consisting of at most three hierarchical levels. If I had to choose between these two models, I would start with the version of TOL 2008.

Exp.2: (a) Very useful.

(b) Both versions are useful, but with the artificial hierarchy it would be extremely handy to be able to change the order of the features.

15. (a) **What do you think: Which one of the visualizations, the visualization by TOL 2008 or the visualization by the artificial hierarchy, would be easier to adopt by a non-technical end user?**

- (b) **Why?**

Exp.1: (a) The first one.

(b) With the artificial hierarchy, a user might think that he/she always has to drill down to the most exact hierarchical level. Moreover, since the order of the features in the hierarchy is predetermined, it does not necessarily respond to the needs of all users.

Exp.2: (a) The first one.

(b) Both visualization versions are implemented similarly. However, the idea of artificial hierarchy would require familiarization.

16. **How useful exactly these kind of prototypes are for following V and changes of V over time by the artificial hierarchy (for example, compared to a table containing the same information)?**

Exp.1: If there was a real business need to follow this kind of data, the presentation style would be absolutely excellent. A table would not work any better.

Exp.2: Quite good but utilizing the artificial hierarchy would require familiarization as already mentioned.

Part 5: Comparing $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$

17. (a) Which one of the prototype versions, $Sunburst_{(E_2)}$ or $Treemap_{(E_2)}$, would you select to be utilized in practice?
- (b) Why?

Exp.1: (a) $Sunburst_{(E_2)}$

(b) Equal reasons to the question 12(a).

Exp.2: (a) $Treemap_{(E_2)}$

(b) Mainly for the reason that it displays one hierarchical level more than the sunburst chart.

Part 6: Informal Feedback on $Sunburst_{(E_2)}$ and $Treemap_{(E_2)}$

18. Would you like to give informal feedback on $Sunburst_{(E_2)}$ or $Treemap_{(E_2)}$?

Exp.1: It would be very useful that the order of the features in the artificial hierarchy could be altered by a user. Then the same visualization would respond to more use cases.

Exp.2: It would be interesting that the order of the features in the hierarchy could be selected. However, this would mean more work at the data level.

8. Conclusion and Future Work

This thesis examined how to visualize changes over time in hierarchical customer data using the Plotly Python Graphing Library and was written as an assignment for a Finnish company. The motivation behind the thesis commission came from the commissioning company's desire to monitor changes of an important continuous variable over time by hierarchical customer segments. The commissioning company was interested in researching how this kind of complex customer data could be visualized in such a way that is clear enough for non-technical business users.

The thesis consisted of a literature survey and experimental part. The literature survey presented the most commonly used hierarchical visualization techniques and different possible encoding techniques for adding time dimension on top of these hierarchical visualization techniques. Moreover, the pros and cons of different techniques and encodings were discussed. In the experimental part, the gathered information was utilized in practice by designing experimental visualization prototypes visualizing changes over time in the commissioning company's hierarchical customer data. In addition, these visualization prototypes were validated in the commissioning company by arranging an end user study and expert review. The conclusions of the literature survey and experimental part are presented in their own sections. Finally, future work is discussed.

8.1 Conclusion of the Literature Survey

To conclude the findings of the literature survey: The hierarchical visualization techniques can be divided into two categories: the explicit methods, that clearly display the edges connecting the nodes, and the implicit methods, that do not draw visible links between the nodes but show the structure of the hierarchy by the arrangement of nodes. The implicit methods are more space-efficient than the explicit methods and therefore suit better for displaying target variable values related to a hierarchy. The implicit methods are further divided into the adjacency and enclosure diagrams according to whether the structure of the hierarchy is encoded by adjacency or containment respectively. The enclosure diagrams are more space-efficient than the adjacency diagrams, while the adjacency diagrams display the hierarchical structure more clearly. The explicit methods discussed

in the thesis are the traditional node-link diagram, dendrogram, and indented tree. The covered adjacency diagrams are the icicle chart and sunburst chart, and from the enclosure diagrams, the treemap, circular treemap, and bubble treemap are dealt with.

Different time encoding techniques that can be added on top of a hierarchical visualization technique are color, typography, interaction, animation, and small multiples. There can also exist other encodings, but the thesis concentrates on the aforementioned ones. These time encoding techniques are usually utilized in combination with the implicit hierarchical visualization methods rather than with the explicit methods since the implicit methods are more space-efficient. To summarize the main advantages and disadvantages of the different time encoding alternatives: Color gives a fast perception of in which nodes the values have increased or decreased, and what are the magnitudes of changes. However, color cannot display very subtle differences or exact numerical values. With typography instead, exact numerical values can be perceived, but a fast overall understanding of data cannot be obtained. Interactions are flexible, and they can be implemented in many different ways. With interactions, the limited display space can be expanded, but on the other hand, if all information cannot be observed from a single view, going over all possible combinations can take a lot of time and something important might go unnoticed. Animation can be a stylish and captivating encoding method in which time dimension does not clutter the view. However, in animation, detecting changes in data relies completely on a user's memory. With small multiples, a user can analyze the distribution of a target variable at distinct time points, but the distinct figures can become very small and difficult to distinguish as the display space is divided between them.

8.2 Conclusion of the Experimental Part

The commissioning company assigned two visualization problems to be solved in the experimental part of the thesis using the Plotly Python Graphing Library and requested to focus especially on the first problem and regard the second problem as more of an additional research problem. The data used in the visualizations of the thesis was the commissioning company's data set of their company customers. In the visualizations, this data was divided into hierarchical customer segments, and the changes over time in an important continuous variable V were visualized by these segments. The first problem was how to visualize changes over time in V by a hierarchical explanatory variable *Industry*. The second problem was how to visualize changes over time in V by an artificial hierarchy created by combining multiple categorical features. In cooperation with the commissioning company, five categorical features related to the customers were selected for this artificial hierarchy, and the customer base was divided into hierarchical customer segments by utilizing these five features one after another.

Two alternative hierarchical visualization prototypes were designed for the first research problem, and by changing the underlying data and making minor modifications, the same prototypes were also utilized for the second problem. To be more precise, the prototypes visualized the variable V in 2021 and the absolute difference of V from 2020 to 2021 by the aforementioned hierarchical structures. The commissioning company requested that the selected visualization techniques should highlight how the values of V and the absolute changes of V aggregate by summing from the lower hierarchical levels up to the higher hierarchical levels. Another point that had to be taken into account in the design process of the prototypes was that both the hierarchy of *Industry* and the artificial hierarchy were very large containing a lot of nodes and five hierarchical levels. The visualization prototypes were designed and validated by following the Munzner's nested model for visualization design and validation [22].

By exploiting the results of the literature survey, the sunburst chart and treemap were selected as the two alternative hierarchical visualization techniques for the prototypes due to their space-efficiency and ability to highlight the aggregation of data. The arc width in the sunburst chart and the area of a rectangle in the treemap were used to encode the 2021's values of V . The sunburst chart displays the hierarchical structure more clearly than the treemap, while the treemap is more space-efficient, and the purpose of the thesis was to examine how these alternative techniques compare in practice. Moreover, based on the literature survey of the thesis (which was not completely all-encompassing but contains a lot of examples from scientific papers), the treemap appears to be a very frequently used hierarchical visualization technique when visualizing time-dependent hierarchical data, while the sunburst chart does not appear to be as common. Thus, examining the usability of the sunburst chart in this context is relatively novel and less researched point of view.

Equal time encoding techniques were utilized for both prototype versions, the sunburst chart and treemap. By exploiting the results of the literature survey, altogether three different time encoding techniques: color, typography and interaction, were chosen to encode the absolute changes of V from 2020 to 2021. Different encodings were used together in order to utilize the advantages of each of them. Color was selected as the main time encoding technique since it enables observing the signs and magnitudes of the absolute changes very fast and effortlessly without cluttering the view. An intuitive red-white-green color scale was utilized because its connotations with decrease and increase. In order to encode also more detailed information into the view without taking any extra space, a hover tooltip, a combination of interaction and typography, was attached to each node. Each tooltip contained the node label and exact numerical values of the following features: V in 2021, V in 2020, the difference of V from 2020 to 2021, and the number of customers. Moreover, drill down and up interactions were added into the prototypes

so that only a couple of hierarchical levels were displayed simultaneously and information overload was avoided.

The validation focused especially on the prototypes for the first research problem as a request of the commissioning company. According to the end user study and expert review, the prototypes considering the first research problem are very useful and well-implemented in the domain context. Both the sunburst and treemap versions appear to work well in the explorative data analysis tasks they are meant for. The number of participants in the end user study was quite small so the study results are not statistically significant, but the results can be considered as an indicative of a broader user study. The participants of the end user study took separate tests on both prototype versions. The first hypothesis stated the time under which the tests had to be taken so that the prototypes could be proven to be clear and simple enough for non-technical end users, and that hypothesis was clearly supported regarding both prototype versions. In addition, the following two hypotheses were stated: the sunburst version would be faster to use and also the favorite prototype of the users because it shows the hierarchical structure more clearly. There was no significant evidence of which one of the prototype versions is faster to use, but the clear majority of the respondents regarded the sunburst chart as their favorite prototype version so the latter hypothesis was supported. The end users justified this preference by commenting that the sunburst chart displays the hierarchical structure more clearly, is more logical, and is easier to read. Furthermore, the results of the System Usability Scale (SUS) questionnaire measuring the usability of the prototypes were excellent. In SUS questionnaire, the end users were asked to consider their favorite prototype version.

The two experts who participated in the expert review gave also very positive feedback on the prototypes for the first research problem. They considered that the selected encoding and interaction techniques of both prototype versions work very well. The experts agreed that the value of V is easier to perceive with the sunburst chart, but the treemap is clearer and easier to use. However, they had different opinions on which one of the prototype versions they would select to be utilized in practice. Because of this disagreement and for the reason that there was no significant difference among the end users of which one of the prototype versions is faster to use, it could be inferred that there might not be any clear and objective answer to the question of whether the sunburst chart or treemap works better in this use case. Subjective preferences appear to have a strong effect on the user experience.

The end users only briefly commented on the prototypes for the second research problem because of the lack of time in the test situation. They considered these prototypes very useful and innovative, and reacted maybe even more positively to them compared to the experts. The experts were a little sceptic about the usefulness of the artificial hierarchy, but regarded the implementation very functional. They emphasized that a

possibility to change the order of the features in the artificial hierarchy is an extremely important development idea for the prototypes.

Both the sunburst chart and treemap suit well for visualizing large hierarchies, and also time dimension fits to the view by the combination of color, interaction, and typography encodings. All the prototypes of the thesis turned out to be clear and simple enough for non-technical end users. Even though the sunburst chart appears to be less utilized than the treemap when visualizing time-dependent hierarchical data, the sunburst chart proved to work in this use case at least equally well, or by some characteristics even better, compared to the treemap. If a clear business need exists, the visualizations like these prototypes could certainly be exploited when analyzing hierarchical customer data. The commissioning company is very satisfied with the visualization prototypes of the thesis, and will very likely utilize them in practice and further develop them. According to the commissioning company, the information acquired from the prototypes is extremely useful for the business, and the technical implementation is very successful.

8.3 Future Work

There are many possible directions for future work. When considering specifically the visualization prototypes of the thesis, if the Munzner's nested model for visualization design and validation had been exploited in full, more time would have been needed to validate properly the first two steps of the nested model. Validating these steps requires examining the adoption rates of the prototypes and conducting a long-term field study [22] for which there was not enough time during this thesis commission.

Regarding the prototype design, it would be interesting to test also other hierarchical visualization methods in addition to the sunburst chart and treemap. For instance, one alternative for the traditional sunburst chart could be the so-called *sundown chart* introduced in the paper [38]. It is a semi-circular version of the sunburst chart that suits better to the aspect ratio of a computer screen (16:9) compared to the traditional sunburst chart whose aspect ratio is 1:1 [38]. As an alternative for the traditional treemap, the bubble treemap could be experimented on since it displays the hierarchical structure more clearly than the traditional treemap. In addition, more different kinds of color scales could be developed and tested on without forgetting the importance of colorblind-friendly versions.

The visualization prototypes could also be expanded by adding the customer level, containing the individual customers, under the lowest level in the hierarchy. This could be very useful, and take the customer analysis to the more specific level. Moreover, the artificial hierarchy could be improved by developing interactive possibilities to tune the structure of the artificial hierarchy: how many and which variables are utilized in the

artificial hierarchy and in which order the variables are combined to create the customer segments. An analogous tuning interaction could be developed for changing which are the two time points that are compared in the prototypes.

When considering the visualization of time-dependent hierarchical customer data in general, it could be examined how longer hierarchical time series consisting of more than only two distinct time points could be visualized in a user-friendly manner. There exists some research on this topic but not very much from the perspective of business use or customer analysis. One important question related to this is whether the visualization of longer hierarchical time series is a too complicated and impractical concept for non-technical end users or is it a potential development idea in businesses. Furthermore, one very common type of hierarchy in customer data, geographical hierarchy, was not discussed in this thesis work. Geographical locations are often visualized using maps, and it would be extremely useful and interesting to examine different ways to visualize changes over time in hierarchically structured maps, such as drill down maps.

Bibliography

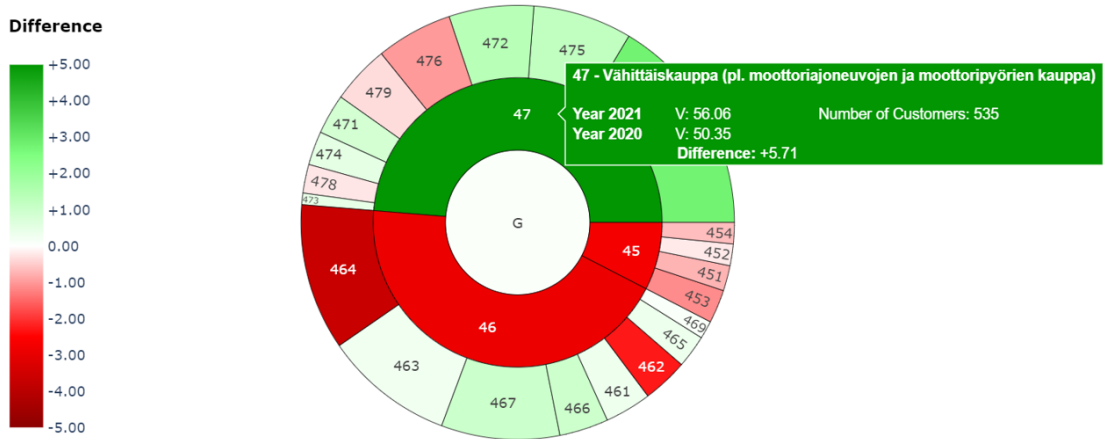
- [1] I. Bacher, B. M. Namee, and J. D. Kelleher. On using tree visualisation techniques to support source code comprehension. In *2016 IEEE Working Conference on Software Visualization (VISSOFT)*, pages 91–95, 2016.
- [2] A. Bangor, P. Kortum, and J. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *J. Usability Stud.*, 4:114–123, 04 2009.
- [3] M. Bostock. Animated treemap, 2019. D3. Observable. <https://observablehq.com/@d3/animated-treemap>, Accessed on 12th July 2022.
- [4] J. Brooke. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, November 1995.
- [5] M. Bruls, K. Huizing, and J. J. van Wijk. Squarified treemaps. In W. C. de Leeuw and R. van Liere, editors, *Data Visualization 2000*, pages 33–42, Vienna, 2000. Springer Vienna.
- [6] M. Burch. Interactive similarity links in treemap visualizations. In *2014 18th International Conference on Information Visualisation*, pages 34–39, 2014.
- [7] M. Burch, M. Hoferlin, and D. Weiskopf. Layered timeradartrees. In *2011 15th International Conference on Information Visualisation*, pages 18–25, 2011.
- [8] E. Cuenca, A. Sallaberry, F. Y. Wang, and P. Poncelet. Multistream: A multiresolution streamgraph approach to explore hierarchical time series. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3160–3173, 2018.
- [9] H. v. de Wetering, N. Klaassen, and M. Burch. Space-reclaiming icicle plots. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pages 121–130, 2020.
- [10] M. Ghoniem and J.-D. Fekete. Animating treemaps. In *Proceedings of 18th HCIL Symposium-Workshop on Treemap Implementations and Applications*, 2001.
- [11] M. Graham and J. Kennedy. Multiform views of multiple trees. In *2008 12th International Conference Information Visualisation*, pages 252–257, 2008.

-
- [12] J. A. Guerra-Gómez, A. Buck-Coleman, C. Plaisant, and B. Shneiderman. Treeversity: Visualizing hierarchal data for value with topology changes. *Proceedings of the Digital Research Society 2012: Bangkok*, 2(7):640–653, 2012.
- [13] J. A. Guerra-Gómez, M. L. Pack, C. Plaisant, and B. Shneiderman. Visualizing change over time using dynamic hierarchies: Treeversity2 and the StemView. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2566–2575, 2013.
- [14] J. Heer, M. Bostock, and V. Ogievetsky. A tour through the visualization zoo. *Communications of the ACM*, 53(6):59–67, June 2010.
- [15] M. L. Huang, T.-H. Huang, and J. Zhang. Treemapbar: Visualizing additional dimensions of data in bar chart. In *2009 13th International Conference Information Visualisation*, pages 98–103, 2009.
- [16] B. Johnson and B. Shneiderman. Tree-maps: a space-filling approach to the visualization of hierarchical information structures. In *Proceeding Visualization '91*, pages 284–291, 1991.
- [17] W. Köpp and T. Weinkauff. Temporal treemaps: Static visualization of evolving trees. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):534–543, 2019.
- [18] G. Li, Y. Zhang, Y. Dong, J. Liang, J. Zhang, J. Wang, M. J. McGuffin, and X. Yuan. Barcodetree: Scalable comparison of multiple hierarchies. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1022–1032, 2020.
- [19] K. Liu and P. Liu. Visual analysis of customer data in commercial banks. In *2009 International Conference on Business Intelligence and Financial Engineering*, pages 652–655, 2009.
- [20] J. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, April 1986.
- [21] A. Macquisten, A. M. Smith, and S. Johansson Fernstad. Evaluation of hierarchical visualization for large and small hierarchies. In *2020 24th International Conference Information Visualisation (IV)*, pages 166–173, 2020.
- [22] T. Munzner. A nested model for visualization design and validation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):921–928, 2009.
- [23] C. Plaisant, J. Grosjean, and B. Bederson. Spacetree: supporting exploration in large node link tree, design evolution and empirical evaluation. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, pages 57–64, 2002.



- [24] PyData. Pandas – python data analysis library. <https://pandas.pydata.org/>, Accessed on 20th July 2022.
- [25] H.-J. Schulz. Treevis.net: A tree visualization reference. *IEEE Computer Graphics and Applications*, 31(6):11–15, 2011.
- [26] M. Sondag, B. Speckmann, and K. Verbeek. Stable treemaps via local moves. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):729–738, 2018.
- [27] H. Song, E. Curran, and R. Sterritt. Flextree: visualising large quantities of hierarchical information. In *IEEE International Conference on Systems, Man and Cybernetics*, volume 7, 2002.
- [28] T. Tekusova and T. Schreck. Visualizing time-dependent data in multivariate hierarchical plots - design and evaluation of an economic application. In *2008 12th International Conference Information Visualisation*, pages 143–150, 2008.
- [29] E. W. Weisstein. Tree. Mathworld – a wolfram web resource. <https://mathworld.wolfram.com/Tree.html>, Accessed on 30th March 2022.
- [30] Plotly Graphing Libraries. Plotly python open source graphing library. <https://plotly.com/python/>, Accessed on 29th March 2022.
- [31] Statistics Finland. Standard industrial classification TOL 2008. https://www2.stat.fi/en/luokitukset/toimiala/toimiala_1_20080101/, Accessed on 15th March 2022.
- [32] Y. Tu and H.-W. Shen. Visualizing changes of hierarchical data using treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1286–1293, 2007.
- [33] D. Turo and B. Johnson. Improving the visualization of hierarchies with treemaps: design issues and experimentation. In *Proceedings Visualization '92*, pages 124–131, 1992.
- [34] Usability.com. System usability scale (SUS). <https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html>, Accessed on 19th August 2022.
- [35] J. van Wijk. The value of visualization. In *VIS 05. IEEE Visualization, 2005.*, pages 79–86, 2005.
- [36] E. Vernier, M. Sondag, J. Comba, B. Speckmann, A. Telea, and K. Verbeek. Quantitative comparison of time-dependent treemaps. 39:393–404, 2020.

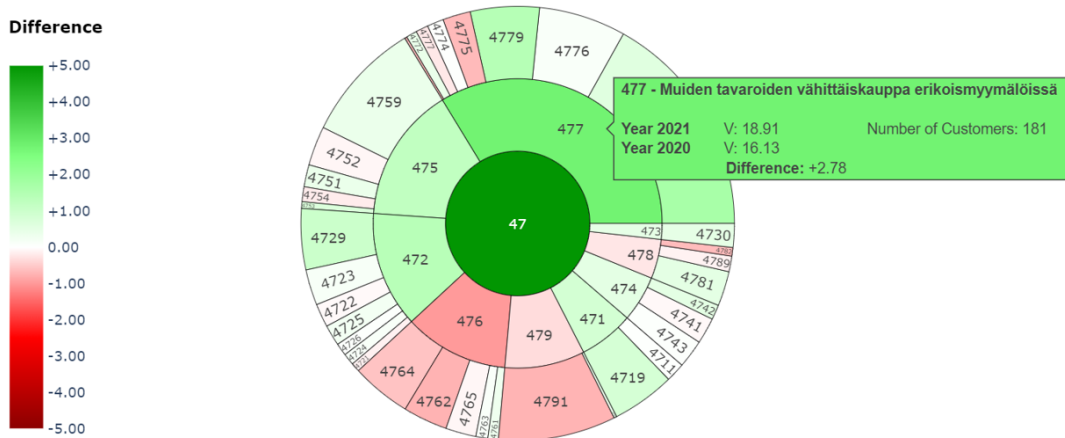
- [37] M. Wattenberg. Visualizing the stock market. *CHI '99 Extended Abstracts on Human Factors in Computing Systems*, pages 188–189, 1999.
- [38] L. Woodburn, Y. Yang, and K. Marriott. Interactive visualisation of hierarchical quantitative data: An evaluation. In *2019 IEEE Visualization Conference (VIS)*, pages 96–100, 2019.
- [39] J. Yang, M. O. Ward, and E. A. Rundensteiner. Interring: an interactive tool for visually navigating and manipulating hierarchical structures. In *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.*, pages 77–84, 2002.
- [40] J. S. Yi, Y. a. Kang, J. Stasko, and J. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007.

V in 2021 and the Difference of V from 2020 to 2021 by Standard Industrial Classification TOL 2008  



Phase 2.

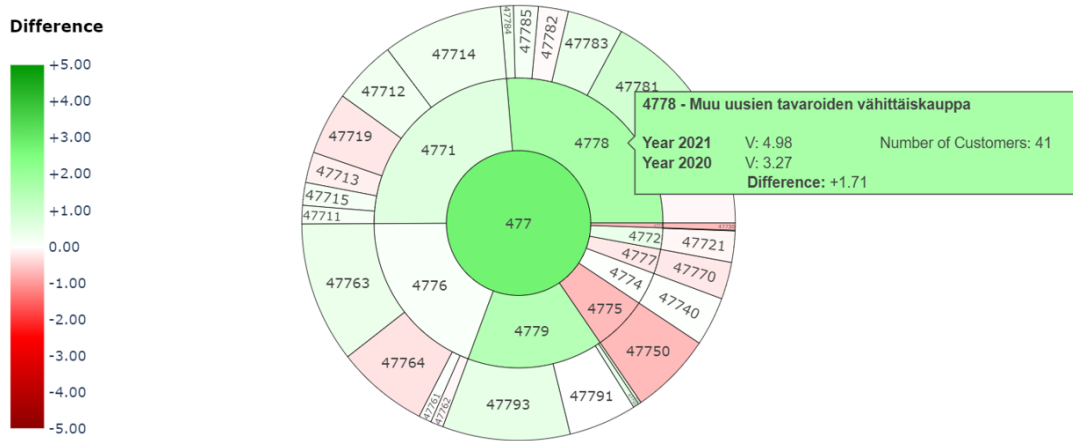
V in 2021 and the Difference of V from 2020 to 2021 by Standard Industrial Classification TOL 2008  



Phase 3.

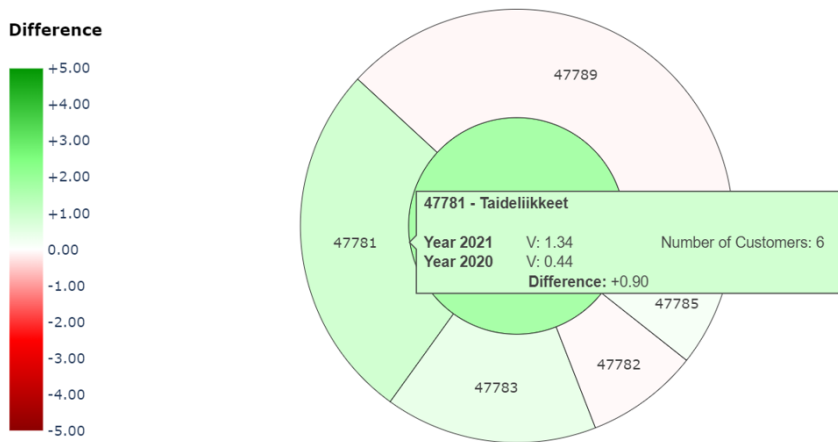
Figure A.1: A comprehensive drilling example of $Sunburst_{(E_1)}$ demonstrating all the possible phases from the initial view down to the most specific hierarchical level. The example path is $Total - G - 47 - 477 - 4778 - 47781$.

V in 2021 and the Difference of V from 2020 to 2021 by Standard Industrial Classification TOL 2008



Phase 4.

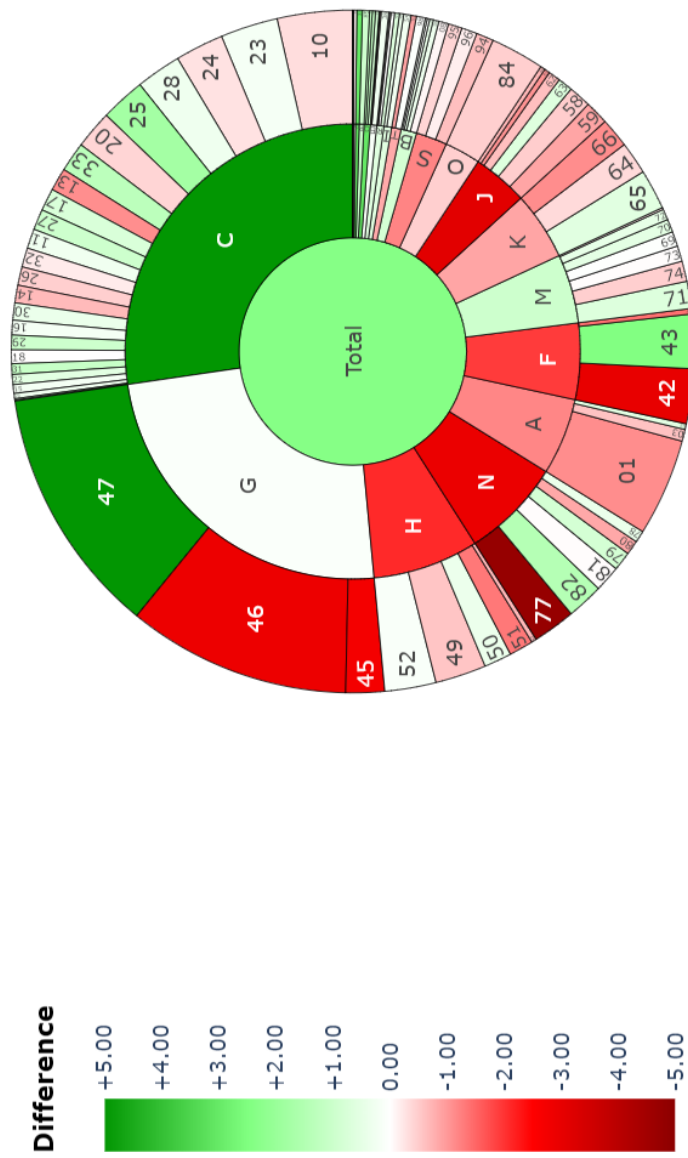
V in 2021 and the Difference of V from 2020 to 2021 by Standard Industrial Classification TOL 2008



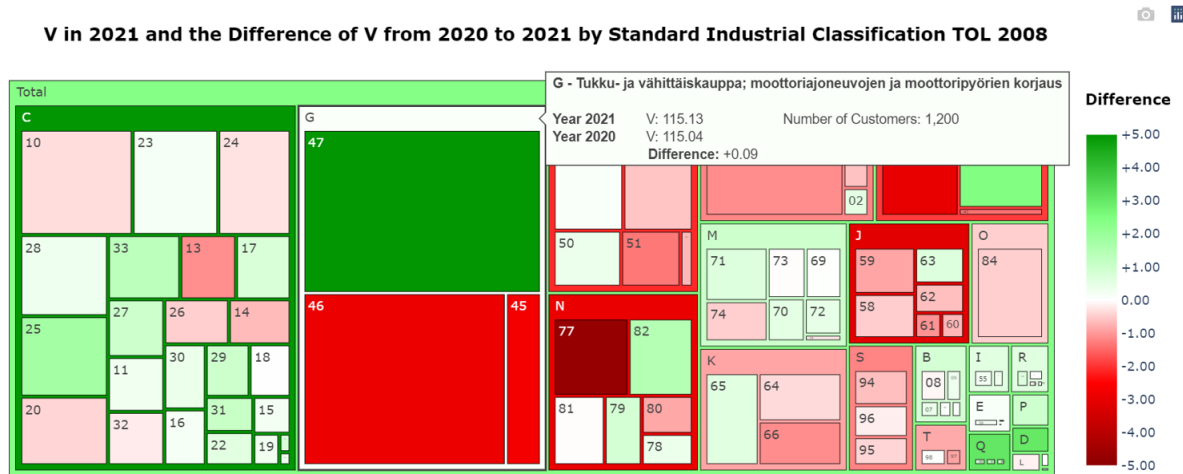
Phase 5.

Figure A.1: A comprehensive drilling example of *Sunburst*_(E₁) demonstrating all the possible phases from the initial view down to the most specific hierarchical level. The example path is *Total - G - 47 - 477 - 4778 - 47781*.

V in 2021 and the Difference of V from 2020 to 2021 by Standard Industrial Classification TOL 2008

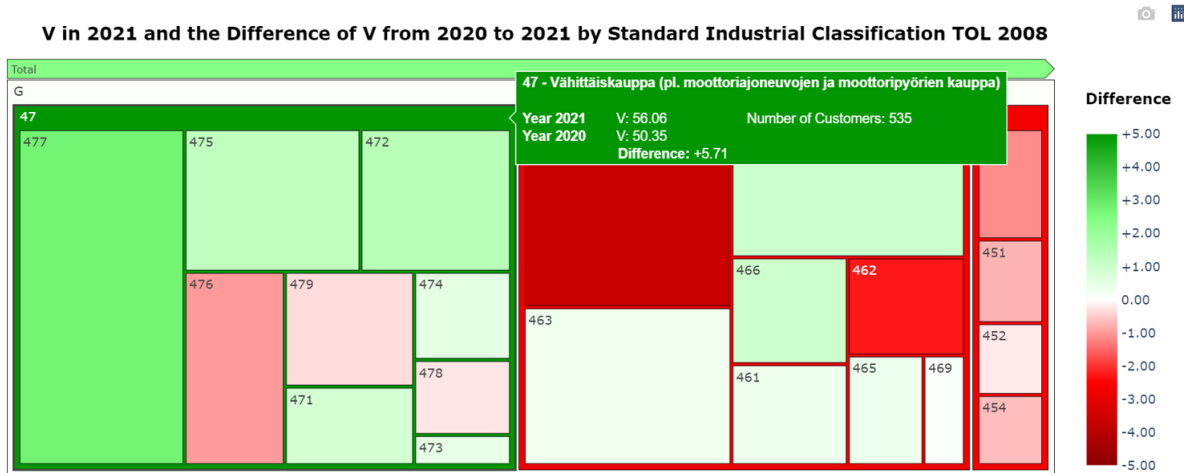


Appendix B. Figures: $Treemap_{(E_1)}$

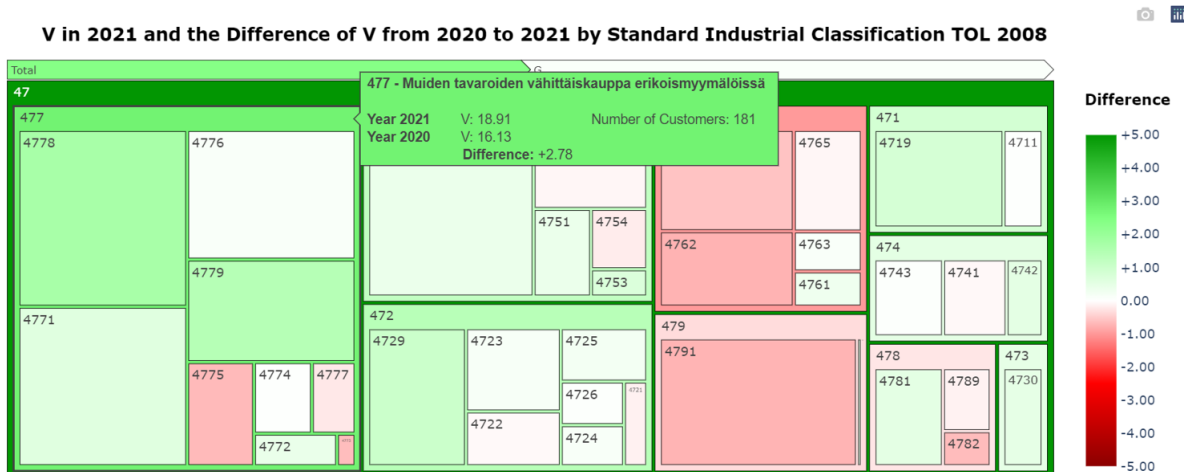


Phase 1.

Figure B.1: A comprehensive drilling example of $Treemap_{(E_1)}$ demonstrating all the possible phases from the initial view down to the most specific hierarchical level. The example path is $Total - G - 47 - 477 - 4778 - 47781$.

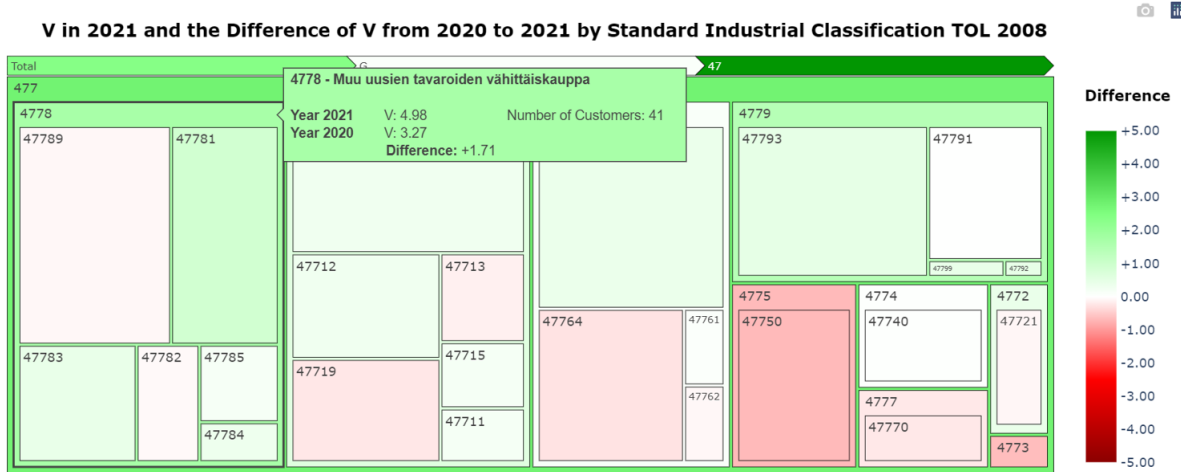


Phase 2.

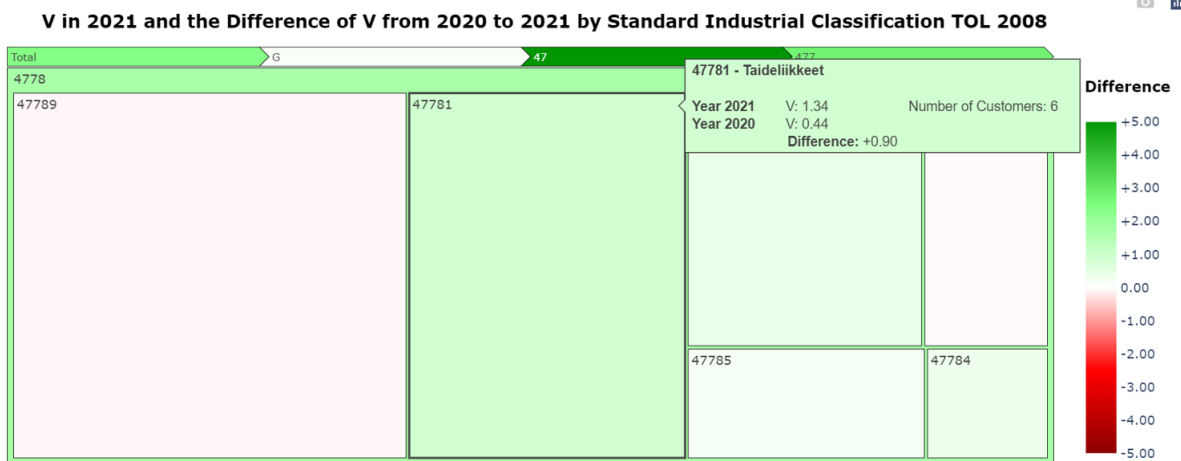


Phase 3.

Figure B.1: A comprehensive drilling example of $Treemap_{(E_1)}$ demonstrating all the possible phases from the initial view down to the most specific hierarchical level. The example path is $Total - G - 47 - 477 - 4778 - 47781$.



Phase 4.



Phase 5.

Figure B.1: A comprehensive drilling example of $Treemap_{(E_1)}$ demonstrating all the possible phases from the initial view down to the most specific hierarchical level. The example path is $Total - G - 47 - 477 - 4778 - 47781$.

V in 2021 and the Difference of V from 2020 to 2021 by Standard Industrial Classification TOL 2008

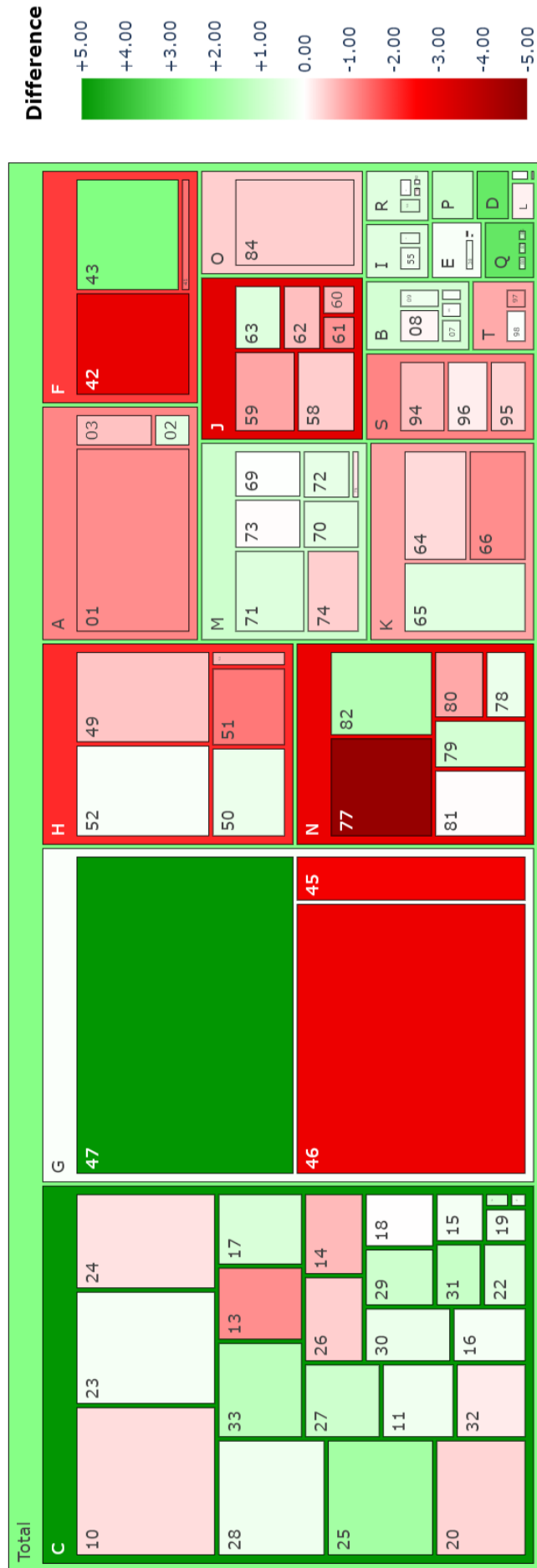
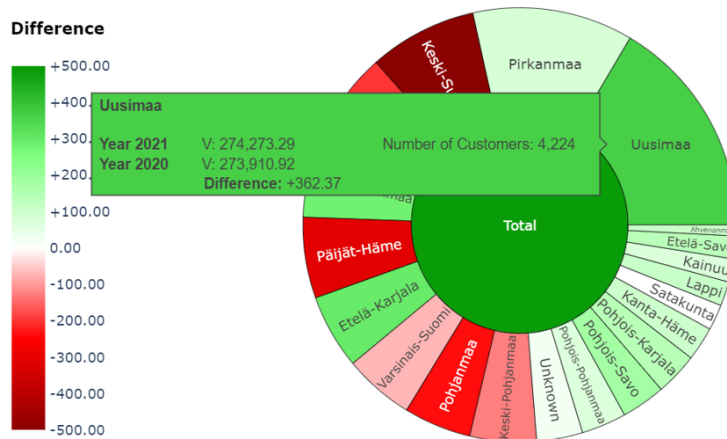


Figure B.2: The initial view of $Treemap_{(E_1)}$ as a full-page version.

Appendix C. Figures: $Sunburst_{(E_2)}$

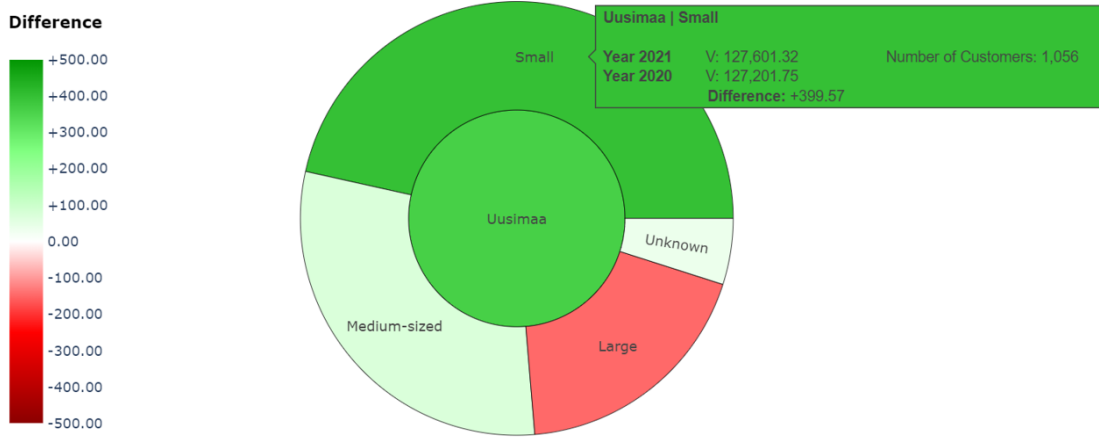
V in 2021 and the Difference of V from 2020 to 2021 by Region, Size, Industry, Turnover, and Business Entity



Phase 1.

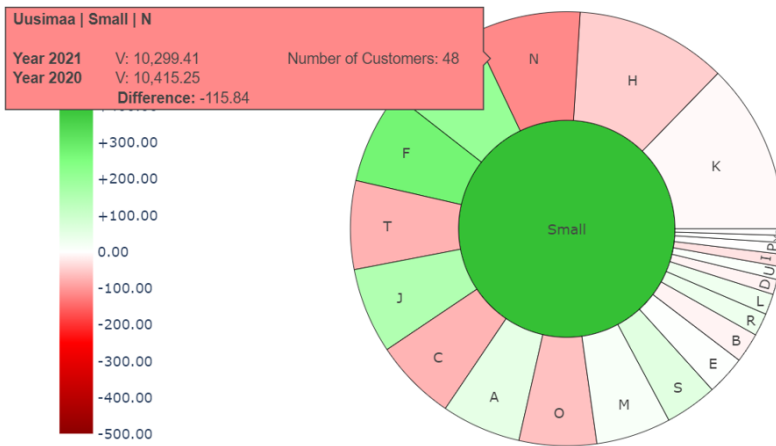
Figure C.1: A comprehensive drilling example of $Sunburst_{(E_2)}$ demonstrating all the possible phases from the initial view down to the most specific hierarchical level. The example path is $Total - Uusimaa - Small - N - <€0.5M - T:mi$.

V in 2021 and the Difference of V from 2020 to 2021 by Region, Size, Industry, Turnover, and Business Entity



Phase 2.

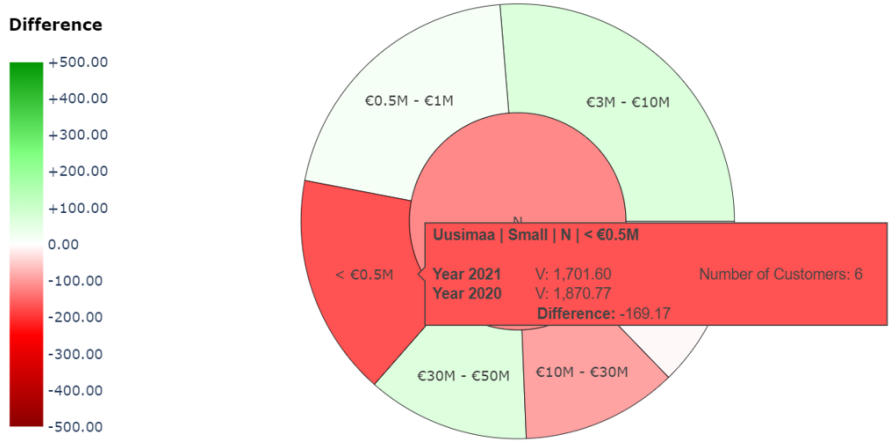
V in 2021 and the Difference of V from 2020 to 2021 by Region, Size, Industry, Turnover, and Business Entity



Phase 3.

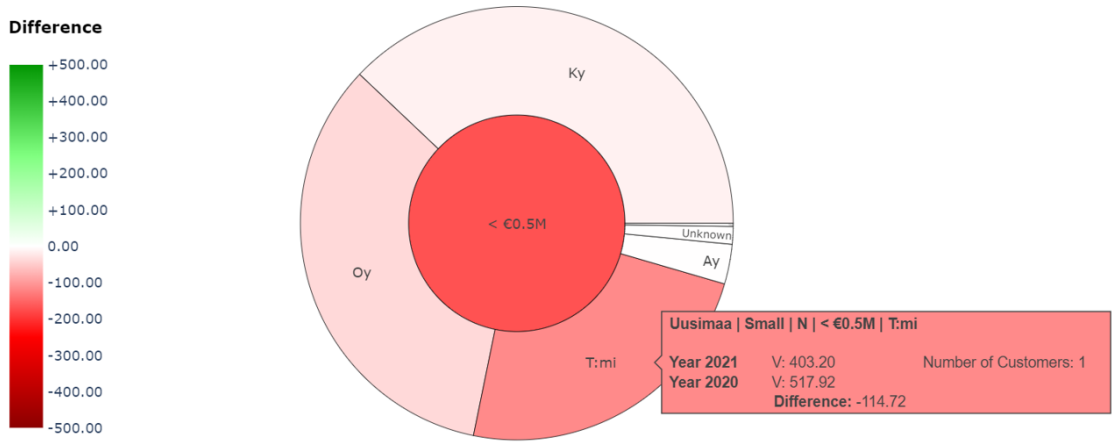
Figure C.1: A comprehensive drilling example of $Sunburst_{(E_2)}$ demonstrating all the possible phases from the initial view down to the most specific hierarchical level. The example path is *Total - Uusimaa - Small - N - <€0.5M - T:mi*.

V in 2021 and the Difference of V from 2020 to 2021 by Region, Size, Industry, Turnover, and Business Entity



Phase 4.

V in 2021 and the Difference of V from 2020 to 2021 by Region, Size, Industry, Turnover, and Business Entity



Phase 5.

Figure C.1: A comprehensive drilling example of *Sunburst*(E_2) demonstrating all the possible phases from the initial view down to the most specific hierarchical level. The example path is *Total - Uusimaa - Small - N - <€0.5M - T:mi*.

V in 2021 and the Difference of V from 2020 to 2021 by Region, Size, Industry, Turnover, and Business Entity

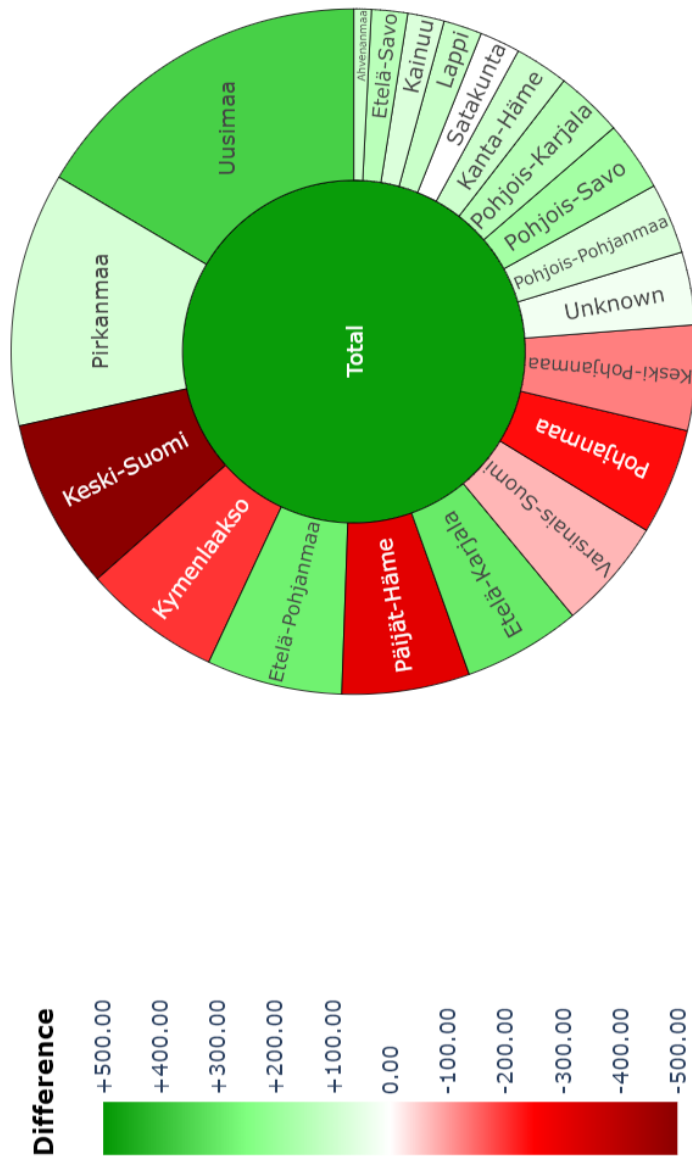


Figure C.2: The initial view of $Sunburst_{(E_2)}$ as a full-page version.

V in 2021 and the Difference of V from 2020 to 2021 by Region, Size, Industry, Turnover, and Business Entity

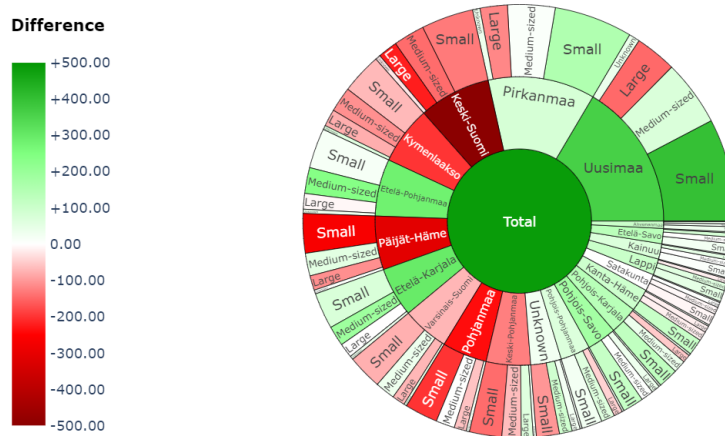


Figure C.3: The rejected version of $Sunburst_{(E_2)}$ with two hierarchical levels branching from the middle node was too cluttered and difficult to read for the purposes of the commissioning company.

Appendix D. Figures: $Treemap_{(E_2)}$

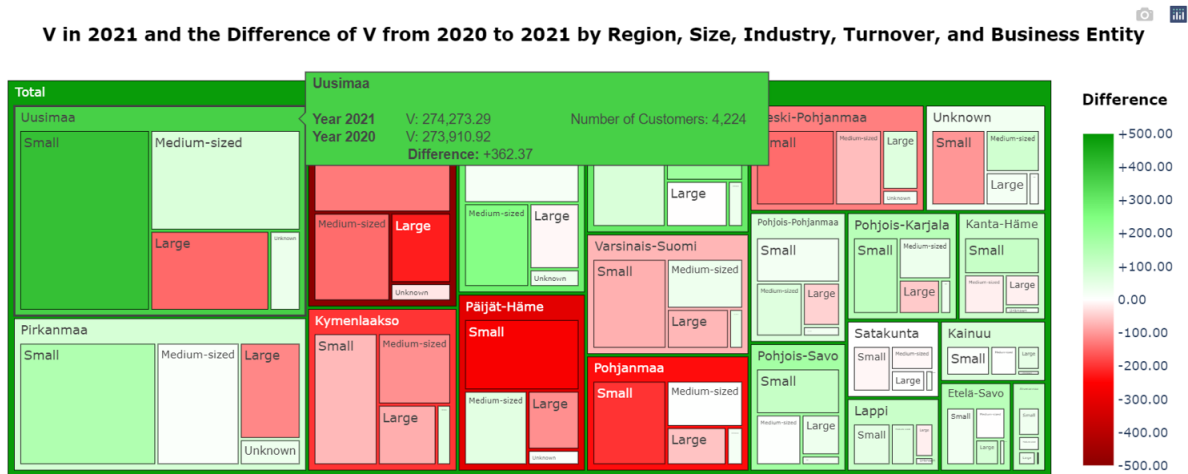
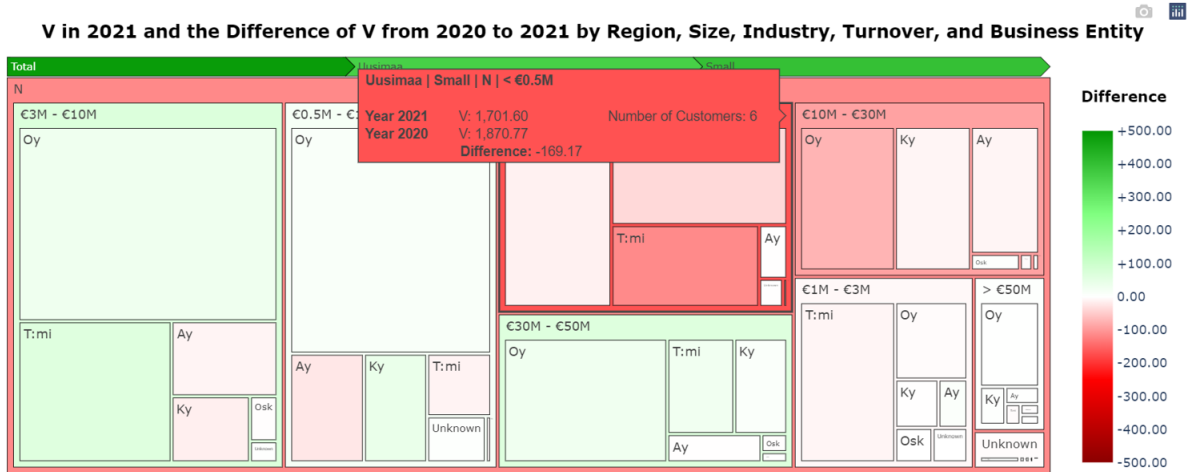
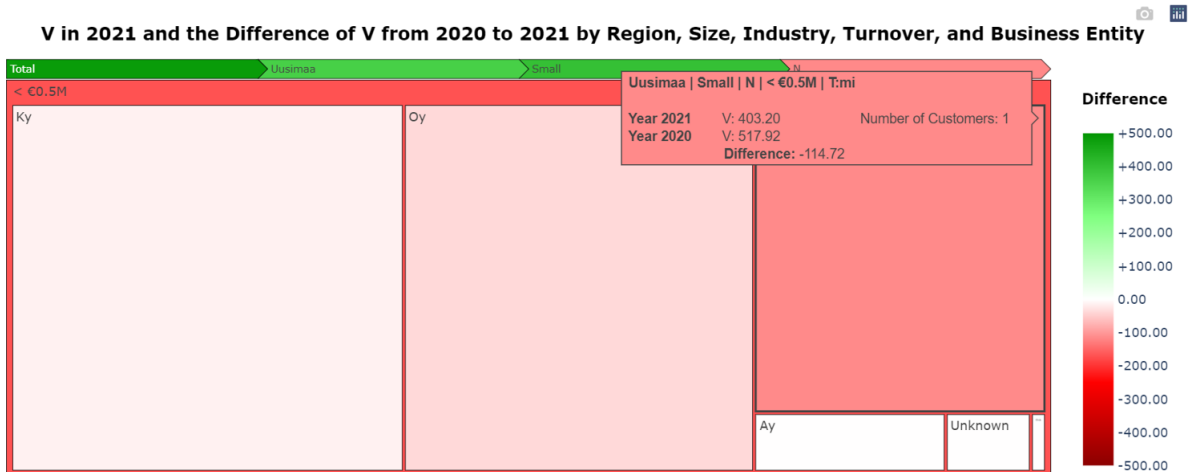


Figure D.1: A comprehensive drilling example of $Treemap_{(E_2)}$ demonstrating all the possible phases from the initial view down to the most specific hierarchical level. The example path is $Total - Uusimaa - Small - N - <€0.5M - T:mi$.



Phase 4.



Phase 5.

Figure D.1: A comprehensive drilling example of $Treemap_{(E_2)}$ demonstrating all the possible phases from the initial view down to the most specific hierarchical level. The example path is *Total - Uusimaa - Small - N - <€0.5M - T.mi*.

V in 2021 and the Difference of V from 2020 to 2021 by Region, Size, Industry, Turnover, and Business Entity

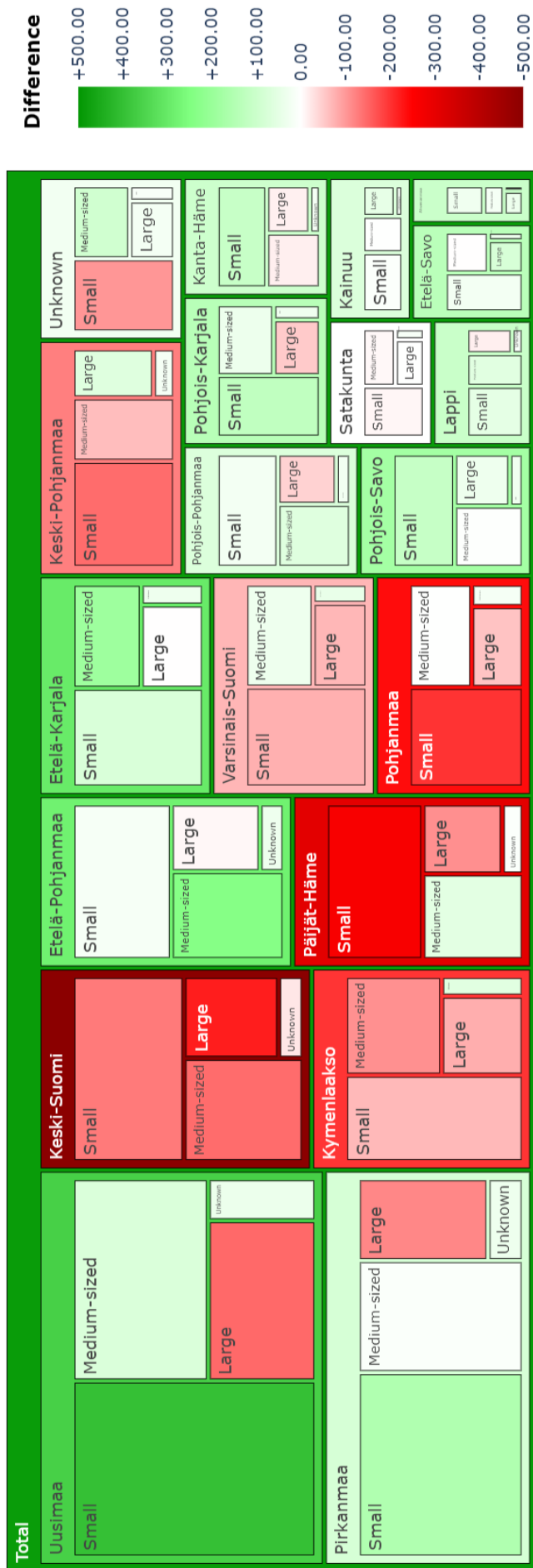


Figure D.2: The initial view of $Treemap_{(E_2)}$ as a full-page version.