School of Computing
FACULTY OF ENGINEERING

Maps, A GeoVisualization Domain Specific Language

Lucía Gómez Álvarez

Submitted in accordance with the requirements for the degree of
MSc on Advanced Computer Science, Intelligent Systems

Session 2014/2015

The candidate confirms that the following have been submitted*:*

<As an example>

Items	Format	Recipient(s) and Date
Deliverables 1, 2, 3	Report	SSO (15/09/15)
Deliverable 4	Software codes or URL	Supervisor, assessor (15/09/15)

Type of Project: Exploratory Software

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)

Summary

<Concise statement of the problem you intended to solve and main achievements (no more than one A4 page)>

Acknowledgements

I would like to thank my supervisor Dr David Duke for his valuable advice during the hole project. I would also like to thank to those who have been next to me this year, making me feel confortable and loved in Leeds, and showing me their support until the very end. Especially to Marta, who's the greater friend I could ever have.

Table of Contents

1. Introduction

1.1. Introduction

Context

Big Data comes, now a days, in a magnitude and complexity that challenges knowledge extraction from it. Together with Data Analytics and Machine Learning strategies, methods and tools for Data Visualization provide support to researchers from all the academic fields.

Furthermore, estimates suggest that 80 percent of all digital data generated today include geospatial referencing (e.g., geographic coordinates, addresses, and postal codes) [1]. In these cases, visualization and visual methods are specially relevant to study the new forms of geographic information, as they provide insight on the spatial relations of data. Therefore, quantitative approaches to geovisualization in GIScience offer productive ways of working with geoweb-based information in research [2].

Problem statement

The problem of generating meaningful geographic visualizations diverges significantly from the rest of information visualization areas. The reason being that, unlike most representations of discrete data (Section 3.2.1), geographical information is linked to spatial locations and therefore requires specific techniques, as it will be reviewed in Section 3.2.2.

In addition, within the context described in the previous section, some of the main challenges that arise for existing Geovisualization tools are the need of providing the flexibility and expressiveness needed to create appropriate visualizations within a wide range of contexts and extracting and linking information from heterogeneous sources.

1.2. Aim

The objective of the project is to provide a Geographic Visualization Toolkit in Haskell to address the problem stated on the previous section. As an alternative to traditional GIS which provide a limited yet complex set of visualizations, the aim of this work is to propose a way of dealing with visualizations through composition and transformation of basic features and facilitating the integration of heterogeneous data.

1.3. Report Structure

In the first place, a description of the planning and management approach that will be undertaken. Then, a review on the subject of Information visualization and specifically of Geographic Visualization and Multivariate Visualization

2. Planning and Project Management

This section contains a description of the project organization structure. It starts by the definition of the main objectives and requirements (2.1), followed by an outline of the deliverables of the project (2.2). Then, the methodology is described (2.3) and the initial and updated schedule are presented (2.4). Finally, the risk assessment is provided (2.5).

2.1. Objectives and Requirements

The key objectives and the corresponding requirements of the project can be summarized as follows:

1) To provide a review on the current status of geo-visualization of multivariate data. This includes both a broad overview on information visualization and a more specific look at geo-visualization. In addition, existing tools and current needs should be reviewed, specially with respect to multivariate data.
2) To develop an exploratory software toolkit able to generate meaningful visualizations and addressing some of the current challenges. These are described in section 3.5.
3) To produce meaningful visualizations for the study case, which consist of an analysis of human migrations. The reports elaborated by the World Bank will be taken as a relevant reference.
4) To evaluate existing visualization techniques, particularly those of the proposed solution. A critic analysis on the strengths, weaknesses and opportunities of different perspectives to tackle the visualization problem.

2.2. Deliverables

1, 2 – Written report of the project, including:

- An overview of the current status of geovisualization together with a summary of its problems, challenges and opportunities.
- An outline of existing scientific visualization techniques relevant for geographic data.

3 – Outputs of the Study Case: Set of visualizations showing insight on global human migrations.

4 – Software library containing the Domain Specific Language for Geographic Visualization.

2.3. Project Methodology

The project will be developed following an agile methodology. The architectural modularity of the software should benefit this approach, enabling the division of the process in multiple iterations. This organization guarantees our capability to manage the growing complexity without compromising the core functionality, which will be developed during the first iterations. Each iteration consist of well defined steps.

However, given the exploratory nature of the project, a longer (4 weeks) preliminary analysis phase is done, which is centered on:

- Study of the State of the art. A review on the state of the art is needed in order to detect the current challenges and to provide the necessary background to propose an adequate solution. This step corresponds to the Delivery 1
- Problem understanding. An analysis of the particularities of the problem is done and therefore the scope of the project is delimited within this section, with respect to the broader perspective of the prior. This step corresponds to the Delivery 1
- Design of the study case and data retrieval. A draft of the requirements of the study case is elaborated on this first phase of the project in order to detect interesting visualizations that should be provided by the toolkit. However, this section has evolved over the project. This step corresponds to the Delivery 3
- Functional Programming. Given that the project is developed in Haskell, it is necessary to estimate a reasonable time to learn a different programming paradigm. This is an essential precondition for any task of development or even analysis and design.

The next iterations follow a common scheme on agile methodologies and are approximately one week long:

- Definition of a set of requirements. With the knowledge acquired during the last iteration, some requirements may be updated, deleted or added.
- Planning. A subset of requirements is chosen to be completed within the current “sprint”.
- Development. This includes both the analysis, design, implementation and testing of the selected requirements.

The following picture (Figure 2.1) shows the cycle of tasks that are executed on every iteration. Within this project, there is no Delivery stage in the springs, given that the submission is unique.

This cyclic process provides great flexibility to the development. In addition, the meetings with the supervisor have been a good opportunity to discuss the requirements and directions that have been taken.

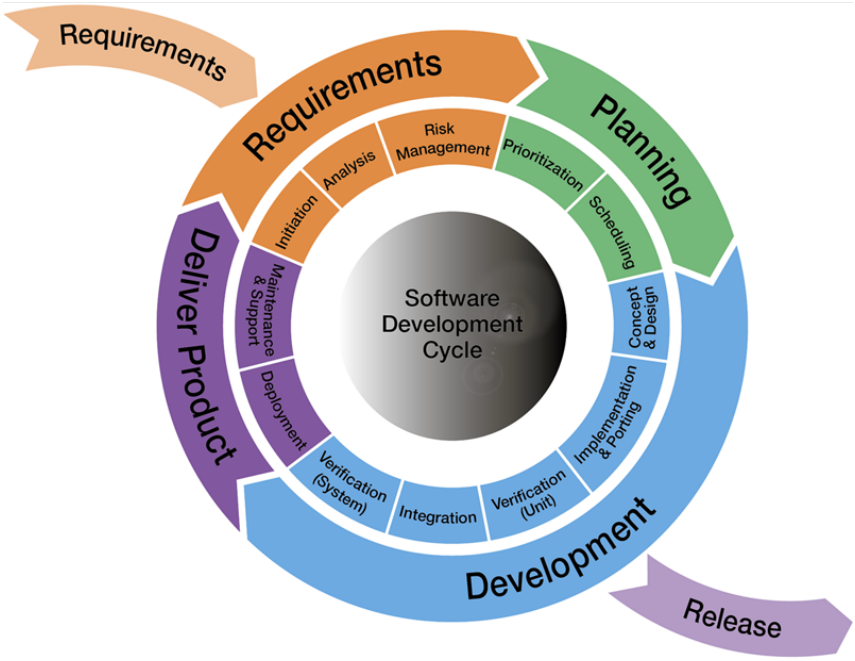


Figure 2.1: The cycle of Agile Programming

2.4. Schedule

The following diagram shows a scheme of the iterations undertaken during the project. By the nature of the methodology, the objectives for every iteration were not set since de beginning, but decided and planned during every sprint.

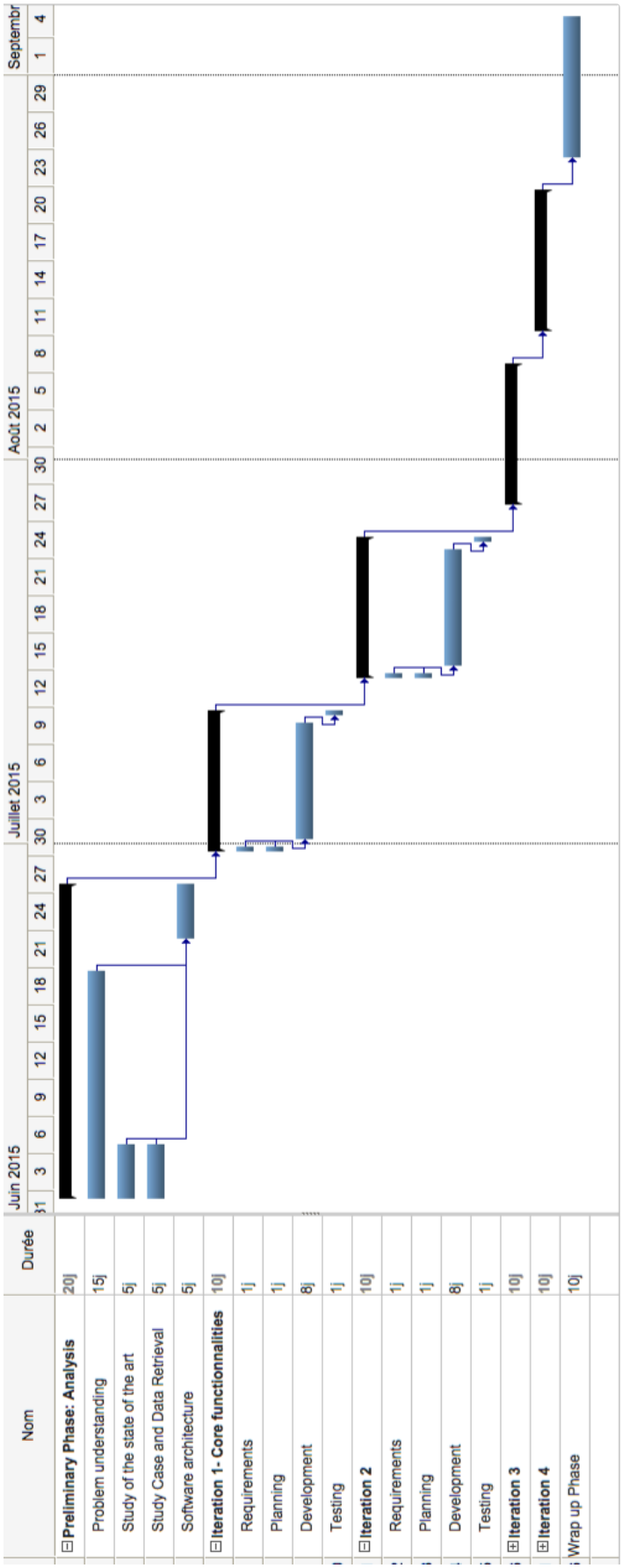


Figure 2.2: Gantt diagram of the first planification

2.5. Risk assessment

During the planification phase of the project, two main potential risks were identified as being possible obstacles for the achievement of the stated objectives.

In the first place, the complexity of the objectives can be underestimated. The exploratory nature of the project implies a certain degree of uncertainty and, even with the background acquired from the research on the field, it is difficult to estimate the time required to accomplish the tasks.

Secondly, the use of a Functional Programming Language to develop the solution. Given that I don't have prior background on functional programming, this becomes both an objective itself and a risk for the project. It implies not only learning a different language but a new paradigm, and therefore a new way to represent programs and to approach problems. Given that these languages are popularly known for having a steep learning curve, this is considered to be a risk that may affect to the quality of the software delivered.

Under the occurrence of any of these circumstances the scope of the project should be redefined, which can be done easily through the iterations of the project. This implies that the main repercussion of these risks is the accomplishment of a smaller number of requirements/objectives.

2.6. Conclusion

In this section the global strategy to address the problem has been described. In the first place by setting a list of specific objectives and requirements (Section 2.1), secondly by specifying the deliverables of the project. Then, the project methodology and planning have been described and the risk assessment is presented. All together sets the framework with which the work on the following sections is done.

3. Understanding the Problem

This chapter contains a literature review on multivariate geographical Information Visualization. First, the problem will be formally defined. Subsequently, a global analysis on the field of Information Visualization will be done, with individual chapters focusing on Knowledge Discovery and Geographical Visualization. Then follows a discussion on the typology of data and its implications for the project, together with a summary of the main visualization techniques. Finally, the main challenges are highlighted.

3.1. Problem Definition

As stated in the Introduction, the problem that this project tries to address is to provide an appropriate geographic visualization toolkit that enables research and knowledge discovery from multivariate spatial data.

This is framed within the broad area of Information Visualization (InfoVis), a research field that aims to aid users in exploring, understanding and analyzing data through progressive, iterative visual exploration [10].

InfoVis can be defined more precisely as the study of transforming data, information and knowledge into interactive visual representations, and it is specially important to users because it provides mental models of information [9]. Furthermore, the boom in big data analytics has triggered broad use of InfoVis in a variety of domains, ranging from finance to sports to politics [9].

This project is mainly focused on the particularities related to the use of Geographical Information, which is an important part of the field because both the volume and the diversity of geographical data are increasing [7, 8]. Spatial information is being collected through automatized systems: by the public administrations, by the traffic systems, by satellites, … and published as open data in a wide variety of formats. What is more, the spatial relationships, when available, have demonstrated to provide useful insights in many different situations, being a classic example of that a map of cholera outbreaks, by John Snow [8], which showed that the disease spreads by water.

Finally, we will undertake a research on proposed visualization techniques for Knowledge Discovery. We consider this to be a necessary step because traditional spatial analytical methods were proposed in a context in which the amount of available variables was reduced. They consist mainly on statistical methods, particularly spatial statistics, which are confirmatory and require the researcher to have a priori hypotheses [7]. However, the current availability of highly multivariate and diverse datasets makes it difficult for the

researcher to synthetize and glimpse the relationships among the parameters (to propose hypotheses). Therefore it is needed to address the issue of Data Analysis and Knowledge Discovery.

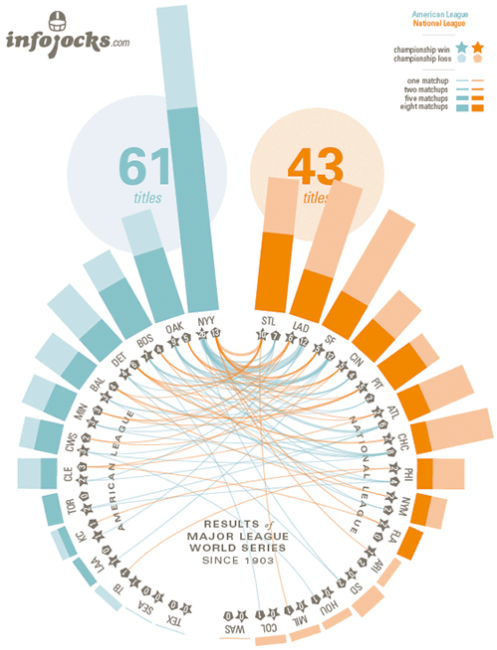
Consequently, within the area of Information Visualization, the problem is decomposed into three different aspects which will have the focus of the further research, which are (1) a global analysis of the state of the art of the field, (2) proposed visualization techniques for Geographic Information and (3) research on Knowledge Discovery and Data Analysis strategies.

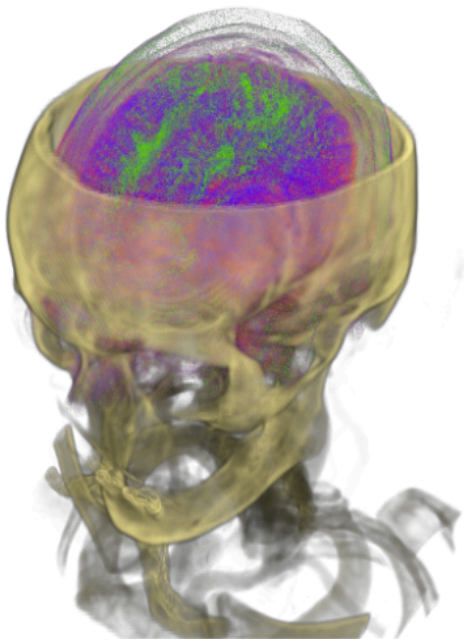
3.2. Information Visualization

Information visualization as a distinctive area of research has rapidly become a far-reaching, interdisciplinary research field [11]. It can be defined as the use of visual representations of abstract non-physically based data to amplify cognition [13], meaning that the visualizations themselves present the information in a way in which the human cognition can extract knowledge that would otherwise stay hidden within the data.

InfoVis with respect to SciVis and Visualization

Although both consist of generating visual representations of data, Information Visualization and Scientific Visualization are considered to be two independent fields of research [14]. The major difference among them is the kind of data that they deal with, consisting of observations of continuous phenomena made over a continuous domain in the case of SciVis and discrete relational records in InfoVis.





a) Results of the Major League Visualization b)Human skull visualization

Figure 3.1: Information Visualization from http://classes.engr.oregonstate.edu/eecs/winter2012/cs519-002/ and Scientific Visualization from http://scivis.itn.liu.se/publications/2009/LLHY09/ Fused Multi-Volume DVR of a human skull.

In the case of data that represents physical n-dimensional phenomena, visualizations typically emphasize on these natural constraints and focus the challenges on embedding visual representations of these data within the underlying physical space. An example of that can be observed in the figure 3.1(b), in which a human skull is visualized through volume rendering, providing insight on the areas in which more brain activity is detected [17]. On the other side, abstract information has no inherent perceptual form, and therefore the representations must rely on other properties like type or structure, rather than space [16]. It is therefore one of the challenges to define a space within which the representation of different properties can be embedded and linked. In figure 3.1(a) a visualization about the results of the Major League World Series through a Smith Graph provides a compact view of the results of all the matches together with the final score.

Additionally, general visualization problems (either SciVis or InfoVis) can be classified into three categories [15], depending on the purpose for which they are formulated. In the first place, Exploratory Analysis representations should enable the search for new hypotheses. Secondly, in a Confirmatory Analysis an already existing hypothesis is expected to be confirmed (or rejected) through the visualization. Finally, Presentations of facts are powerful ways to communicate facts that are fixed a priori to a public.

All this characteristics and constraints regarding the nature of the data and the purpose of the visualization are key factors that drive the selection of certain representation methods to address a particular problem, and therefore an understanding on that is crucial to provide appropriate options within the project's solution.

Geographic Visualization

Although the data used for Information Visualization purposes is by default considered to be abstract, it can be rooted in a number of closely related areas, particularly in geographical information. When geographical mapping is possible, information can be organized in association with geographical positions in a very natural and intuitive way. The influence of geographical and spatial metaphors is so strong that they can be found in most information visualization systems [11].

Indeed, Geovis sits between InfoVis and Scivis, comprising both visualization of continuous phenomena in continuous fields such as temperature, wind strength and direction, etc. and discrete fields such as governmental information.

Visualization works based on geographical maps appear to be simple, intuitive, and natural. This is largely due to the fact of having a geographical structure on which the information is matched and organized. Given that people understand and recognize maps, the visualizations based on them tend to be familiar, and therefore a large amount of information can be easily understood.

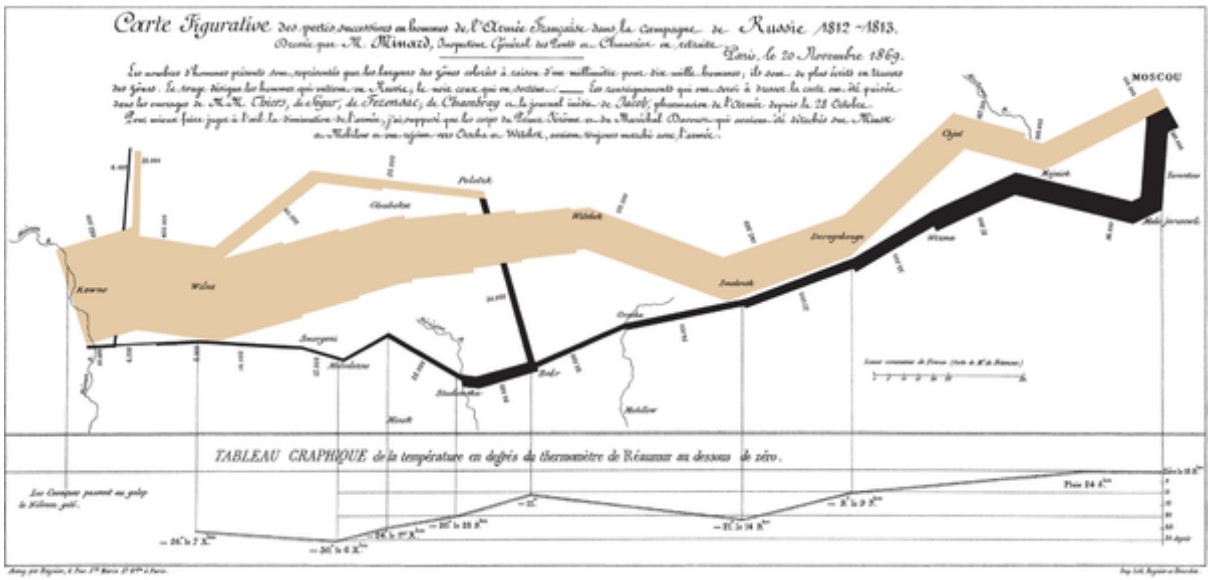


Figure 3.2: The Loss of Napoleon's Army Image by Charles Joseph Minard

A classic example on that is shown in the Figure 3.2, a visualization by Minard which is claimed to be one of the "best statistical drawings ever created." [18]. The visualization is interesting because of its ability to combine al lot of information gracefully: geographic view of the path, loss of life at a time (the width of the path), temperature (line chart), geography (rivers), historical context (color of the path), into one single graphic. In addition, it shows the drastic loss in life from Napoleon's decision in just a single corner of the diagram: through the width difference among the black and the brown lines at the departure point.

Summarizing, geographic information provides a spatial context suitable to map information over it. Visualizations usually combine a geographic/cartographic representations of a region with graphic figures such as areas, lines and arrows. Features such as colors, width or length of the elements represent the values of the information to be represented. This may be combined with linked generic Visualization strategies to amplify the information delivered.

Multivariate Visualization

As it is stated in the introduction, during the recent years the variety of information which may be of interest for making decisions has increased rapidly. The availability of a wide range of parameters potentially correlated implies new promising and important

opportunities. However, there are difficulties on finding the interesting links hidden in large amounts of indicators, which makes it hard to extract new insights from the data.

Regarding to that issue, 'Data Mining' may be defined as the (non-trivial) process of searching and analyzing data in order to find implicit but potentially useful information [21].

An approach to data mining through visualization aims at integrating the human in the data mining process and applying its abilities to the large data sets available in today's computer systems [20]. For this purpose, techniques which provide a good overview of the data and use the possibilities of visual representation for displaying large amounts of multi-dimensional data are especially important.

Conclusion

A global picture on Visualization enables us to have a broader view and to detect the opportunities in which the different fields can benefit from each other (information visualization of scientific data as well as SciVis goodies for spatial InfoVis).

At the same time, different visualization problems have different particularities and face different challenges, whose understanding is key to offer an appropriate set of tools in the proposed solution. This chapter leads to a subsequent discussion in 3.3 and 3.4 (Types of data and representations), in which the main representations for Information Visualization will be reviewed.

3.3. Topology of data

In this section we will propose different categorizations of information. In addition, we will introduce the concept of retinal variables, to see how to visually encode our data. All this should enable a discussion and analysis on the appropriate ways to represent information and properties. In the conclusion, the links with the proposed solution are specified.

Introduction

The quality of a visualization relies not only on the information that it contains, but also on its ability to optimize its representation to the human perceptual and cognitive capabilities. Indeed, the same data encoded in different ways can transmit significantly different amounts of information, or being very clear or misleading. Think of visualizing the temperatures of a country in a scale in which blue is the warmest and red the coldest.

By establishing broad categories of data and the way we perceive them, we should be able to generalize appropriate ways of representing it. However, the classification of data is a big issue. It is closely related to the classification of knowledge, and it is with great trepidation

that we approach the subject [8]. With no expectations of a profound classification, an informal set of data types, forms and categories will be discussed, while analyzing appropriate ways to visualize them.

As a basis, Bertin [22] suggested that there are two fundamental forms of data: data values and data structures, ideas which are also analogous to those of entities and relationships. These concepts have a long history in database design and have been adopted more recently in systems modeling [8]. Both entities and relationships are characterized by attributes.

Attribute Dimensions

One of the characteristics that can be used to establish a topology of data are dimensions. Here follows an overview on data classifications by their dimension, either of the field or the attribute.

N-dimensional fields

In visualization problems in which the data is spatially located, i.e. every value is associated to a point in space, the dimensions of that space determine to a great extent the kind of suitable visualizations. For example we may have 1-dimensional field to represent the position of a point on a line, a 2-dimensional space to represent the position of a point on a surface and so on. In these cases we typically create representations recreating this dimensional space, which is straightforward in two dimensions but can be challenging in other cases.

Scalars, vectors and tensors

Regarding to the value of the attributes, they can consist of single scalar quantities, such as the temperature of a particle (scalar field), quantities associated with a direction (vector field), such as the temperature and the direction of the particle, or higher order quantities (tensor fields), such as shear forces and diffusion. Again, visualizing a scalar value can be easy, trough color for example, but more elaborated representations are needed for tensors, like Haber glyphs or Tensorlines.

It is obvious that these characteristics of data have direct implications on the suitable visualizations for them. This is, indeed, an issue that impacts heavily to SciVis problems and that is been significantly discussed. However, within the area of Information Visualization, the data isn't usually in the form of spatial fields and therefore this problem rarely apply.

In most of the InfoVis problems we deal with the complication of a lack of any spatial dimension, which leads to different techniques to display a meaningful structure of data using its properties, as we can see in Section 3.4.1. Within geographical information, two dimensional coordinates are usually provided, simplifying the decision on the basic layout to display the data (Section 3.4.2). Vectors can be provided sometimes to represent directions or flows, which is the case of migrations for example. In addition, we usually deal with multivariate data.

Levels of measurement

Another of the areas in which we can establish a topology with direct implications in the visualization techniques is the nature of the values of the data. The statistician S.S. Stevens [23] purposed four levels of measurement for values: nominal, ordinal, interval, and ratio. This scale is widely adopted, although it is still being challenged by other theoreticians.

- **Nominal**: Refers to the labeling function. There is no natural order within this category, even if numbers are used as labels. An example of that can be the name of countries for which certain data is shown. Only membership, classification, categorical equality, and equivalence are operations which apply to objects of the nominal type. The mode is allowed as the measure of central tendency for the nominal type.
- **Ordinal**: Refers to sequential and comparable discontinuous values. Ordinal data includes dichotomous values such as “True/False” and non-dichotomous data consisting of a spectrum of values. An example of that could be a ranking on the quality of education (“Very good, good, bad, very bad”) in Europe. In addition to the nominal operations, we may want to calculate the median and perform ordering operations.
- **Interval**: Refers to sequential and comparable continuous values, which implies that the derivation of the gap between sample values is possible. However, in interval data, the origin is arbitrary, and therefore we can measure the *degree of difference* between items, but not the ratio between them. A common example of that is the Celsius scale, which sets the origin at the freezing point of water, which doesn’t mean that there’s no temperature at this point.
- **Ratio**: Refers to real numbers and possesses a meaningful (unique and non-arbitrary) zero value. With this kind of data it is possible to perform any mathematical operation, being the most common kind of measurement in the physical sciences. In this case we can actually say that an object of mass 0 has no mass, and that an object has the double of mass than another, for example.

This topology of data can be very useful in discussing visualization techniques. For example, Ware [8] mentions two generalizations:

(1) Using graphic size (such as the size of points) to represent nominal information is not recommended, as size tends to be interpreted as a quantity and can very easily be ordered. However, texture is almost only suitable for this kind of data and appropriate color scheme tend to be the most effective strategy.

(2) When using colors to map measurements we must take into account that human perception can easily interpret nominal or ordinal values on them. However, perceiving quantities and relationships in a precise way (e. g. “the double of”) is very difficult to achieve with colors, being more suitable size or direction.

These guidelines to represent correctly the available information provide us with opportunities to automatize certain processes and to generalize the ways to represent features within a map (or another representation) as it is discussed in Chapter 6.5.

Operations Considered as Data

When analyzing complex data relationships, different operations may be applied to the data. In his book Information visualization: perception for design, Ware [8] lists the following as the most common operations in data analysis:

- Mathematical operations on numbers-multiplication, division, ...
- Merging several elements on a single one.
- Inverting a value to create its opposite
- Bringing an entity or relationship into existence (such as the mean of a set of numbers)
- Deleting an entity or relationship (a country leaves the EU)
- Transforming an entity in some way (the regime of a country changes)
- Forming a new object out of other objects (A trade accord involves different countries)
- Splitting a single entity into its component parts.

Operations are usually not easy to visualize. Certain operations may have intuitive visualizations, such as creating grouping lines for ‘merging’ and performing opacity addition for ‘sum’. To have control on that may be very interesting for the analyst, which it is examined in Chapter 6.10.

Nevertheless, creating meaningful visualizations for all the operations is a challenging task that generally involves providing a legend to be able to interpret the coding of the representation.

Retinal variables

From the perspective of the representation itself, it is interesting to discuss which primitive variables are appropriate to construct visualizations, depending on the human perceptual capabilities. In the Semiology of Graphics [23], Bertin lists the retinal variables, which are claimed to be compared effortlessly by the perceptual processor (without additional cognitive processing). These are hue, value (i.e. brightness), texture, shape, position, orientation and size.

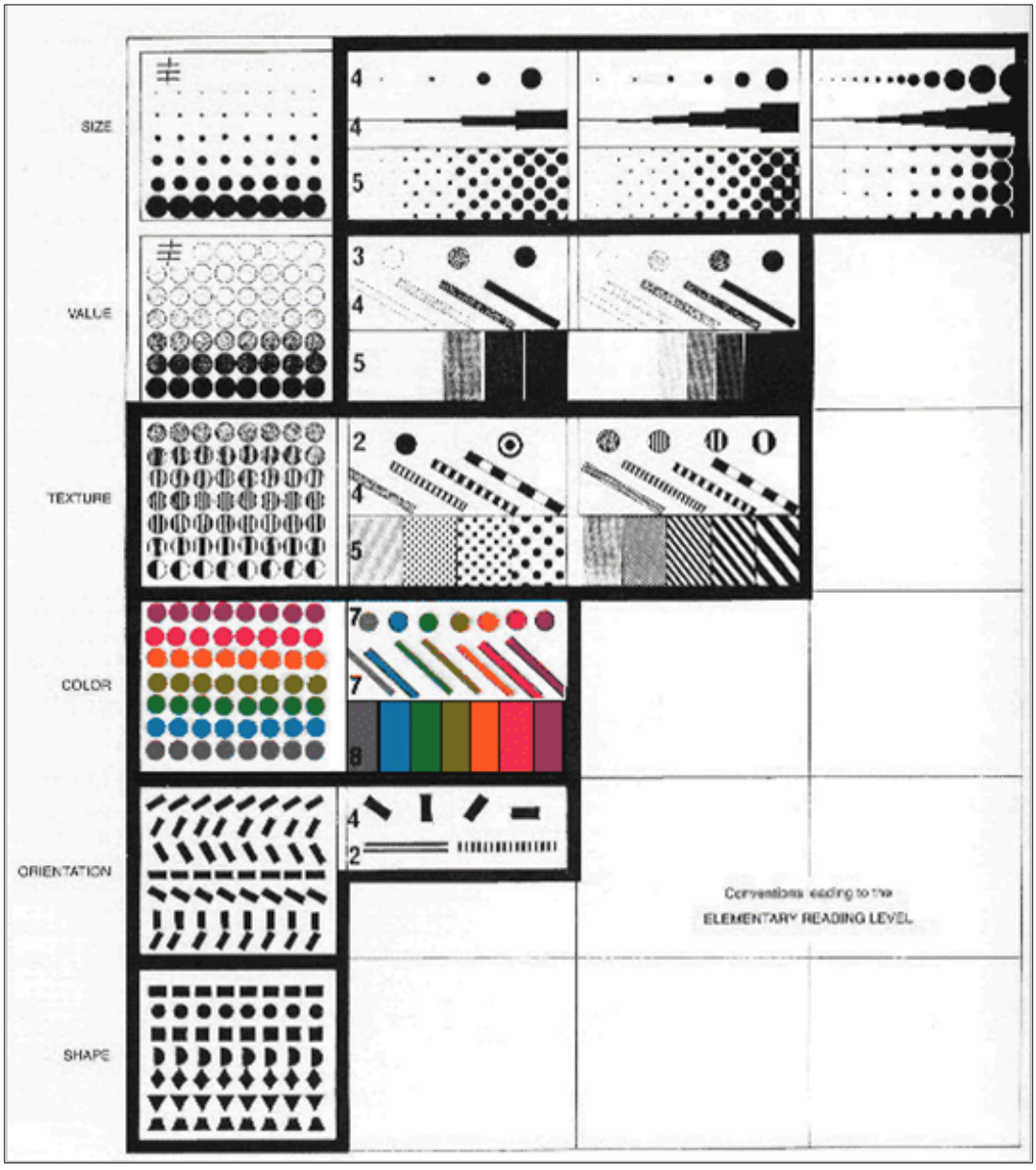


Figure 3.3: The retinal variables and the associated kinds of Data

The Figure 3.3, an extract from the Semiology of Graphics, serves us to wrap up this section, linking the retinal variables with the types of data that they can represent efficiently. Note that for ratio data the only purposed retinal type is the size, and that shapes are only recommended for nominal values.

Further on in the text (Section 6.10) it is discussed how to flexibly establish links between the retinal variables and the different types of data.

Conclusion

The discussed types of data and the available retinal variables to represent them provide us with a good reference to model our solution, as we will see in Chapter 6. Indeed, theorization on the field provides us with abstract ways of dealing with data that should generalize properly for different kinds of visualization. Although the subject has not been analyzed deeply, this section provided us with a good set of consolidated types within the InfoVis community.

3.4. Main representations

In this chapter we will provide a review on the main types of representations proposed to visualize information. On the one hand, we will cover general techniques. On the other hand we will focus on the particular cases that we want to cover, geographical and multivariate information. Finally, we will discuss appropriate ways to integrate all this for our specific problem.

General techniques

Depending on the underlying data and the specific application, specific visualizations may be used to represent it. Depending on that, certain patterns or characteristics will became visible.

Statistical Distributions

To gain insight on how data is distributed and on its statistical properties is a frequent objective for visualization problems. The classic two-dimensional charts such as the histogram, which shows the prevalence of values grouped into intervals, and the pie chart, which visualizes the relationship between the part and the whole, fall into this section. Both of them, have therefore the main aim of revealing how the data is distributed.

More complex visualizations have been proposed for assessing a distribution and examining interactions between multiple dimensions. Some of them are stem-and-leaf plots, Q-Q plots, SPloM (Scatter Plot Matrix) and parallel coordinates (Figure 3.4).

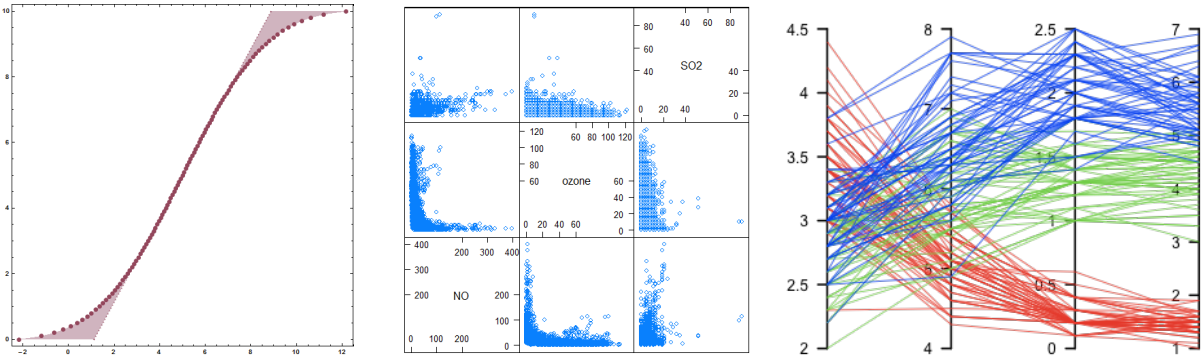


Figure 3.4: In order of appearance, Q-Q plot, SPloM and parallel coordinates

Time-Series Data

Time-varying phenomena is central to many domains and one of the most common forms of recorded data. By convenience is often considered as an extra dimension, and when available most of the insight of the data is expected to consist of the evolution and tendency through the time, like in immigration rates, economic rates, employment rates... Different visualization techniques enable the comparison of large numbers of time series, like Stacked Graphs, Small Multiples or Horizon Graphs

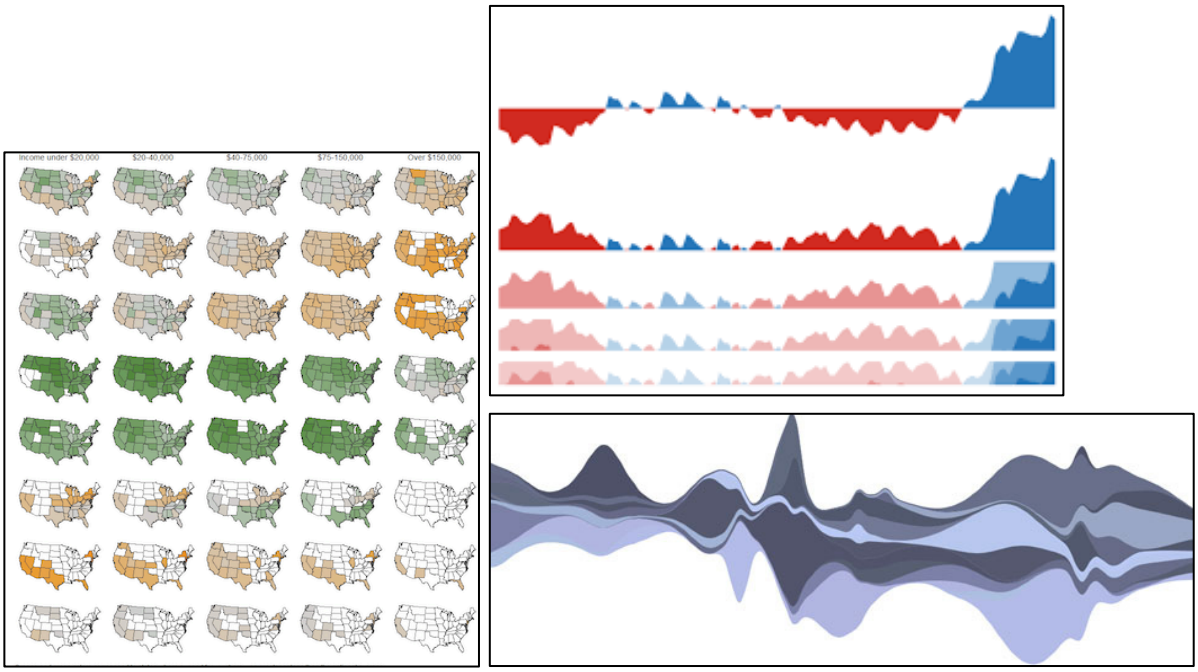


Figure 3.5: In order of appearance, Small Multiple, Horizon Graph and Stacked Graph

Hierarchies

Certain kinds of data can be organized into hierarchies. An example of that are spatial entities such as counties, states and countries. In other cases where the hierarchy is not evident, clustering methods may be applied to detect it there are classes that group the data.

Visualizing properly these connections it is easy to realize inferences part-hole through the hierarchical structure. Some available techniques are node-link diagrams, adjacency diagrams and enclosure diagrams.

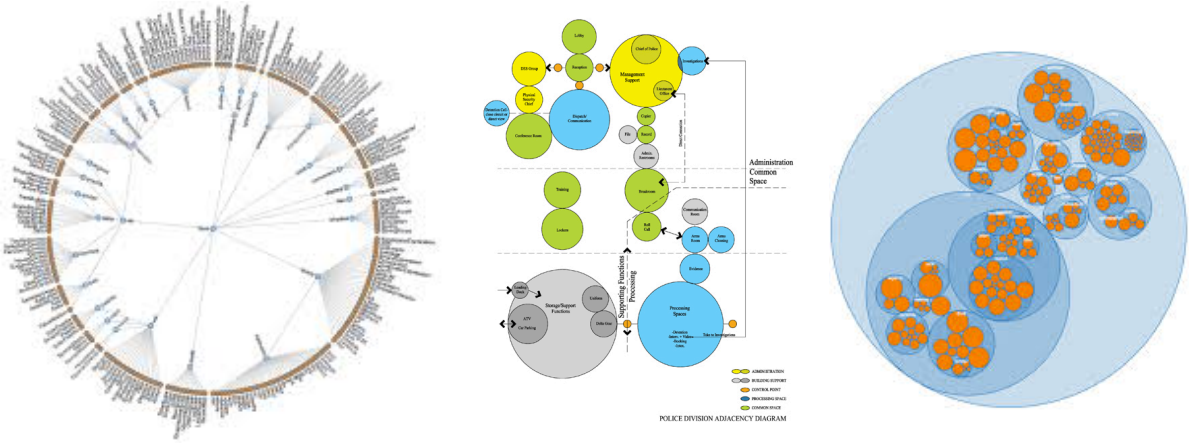


Figure 3.6: In order of appearance, a node-link diagram, adjacency diagram and enclosure diagram

Networks or graphs

Data with interconnections which are not hierarchical can be represented through graphs. In this cases, representations are usually focused on patterns related to topological structures and the structure context. For example, within a global marked, which country buys to which? And what does it buy? A graph is a powerful abstraction of data that consist of elements and connections between elements [9]. Social contacts [25], trajectories on maps [26], and electronic communications [27] can all be modeled as graphs. Some examples of graph visualizations are force-directed layouts, arc diagrams and matrix views.

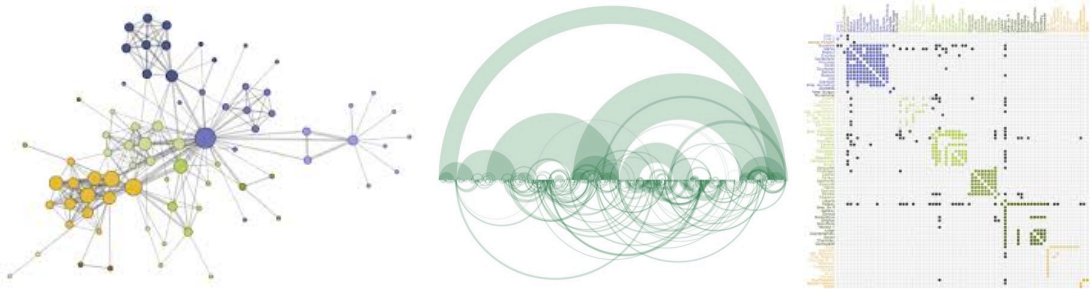


Figure 3.7: In order of appearance, a force-directed layout, arc diagram and matrix view

Map visualization

After reviewing the main techniques for non-spatial information, below we analyze the specific techniques for geographic information. In this case, the information is linked to a spatial location, which can either be a point or an area, and whose understanding is frequently the key to gain insight on the data. For example, to reveal patterns in trajectories, a common and straightforward approach is directly visualize them on the map [26].

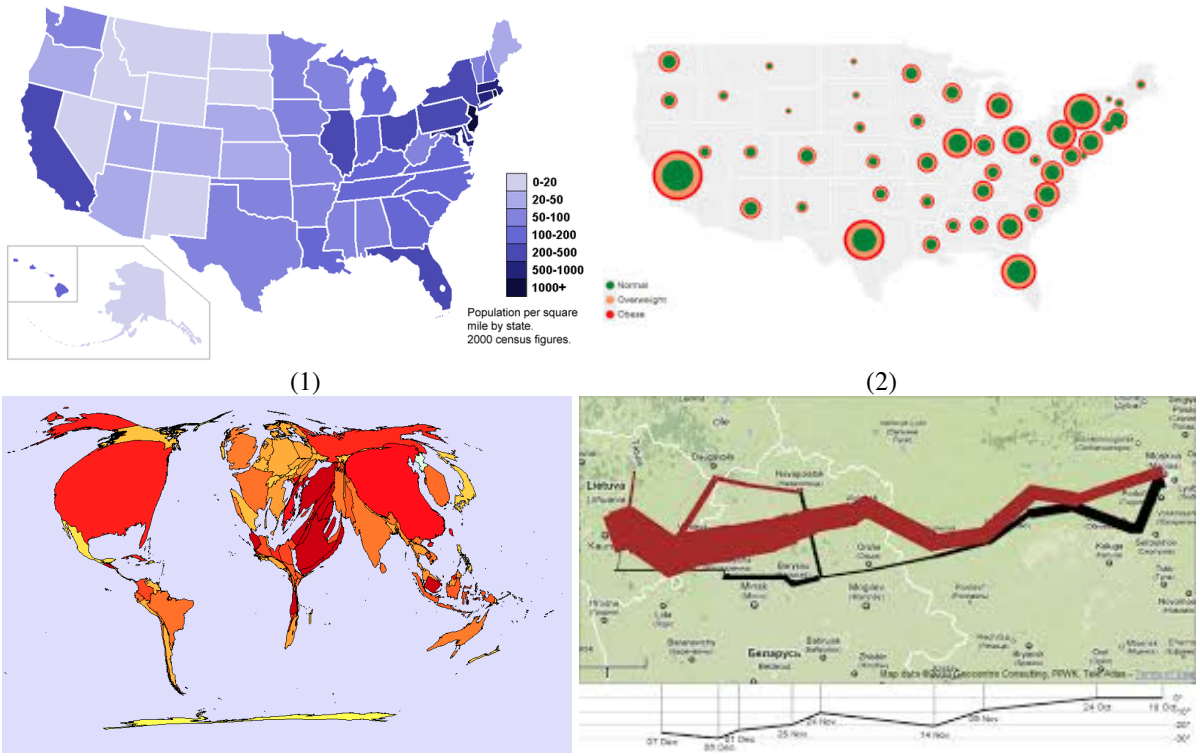
The development of geographic visualization has been strongly influenced by cartography, and benefited from its long history of visual language design and its knowledge of geographic information [9]. When undertaking the first steps to achieve a good visualization of a map, many problems have already been treated in cartography, such as map projection [29], map labeling [30], and map generalization [31].

Particularly map projections have significant implications in perception, given that they map the elliptical body of the Earth into a flat surface, which can't be done without distortion. Different projections are proposed depending on the characteristics that want to be preserved without distortion: area, shape, scale, direction, …

Common representations

Some of the most common representations of maps are:

- **Choropleth maps.** They consist of colouring or texturing geographical regions using different schemes to encode the information. See Figure 3.8(1).
- **Graduated symbol maps**. The information relates to a region, like in the choropleth map, but instead of the colour the data is encoded through glyphs or nested graphs. This strategy enables the visualization of complex data and avoids possible misleading relations between the area of the regions and the represented values. However, it is important to keep them simple as condensed representations may not be effective. See Figure 3.8(2).
- **Cartograms.** This charts modify the size and/or position of the regions in order to link these properties to the data that is been represented. Therefore, in a migration study, the size of the UK may depend on the number of immigrants coming since 2010 instead of the actual size. See Figure 3.8(3).
- **Flow maps**. They consist of placing stroke lines and arrows over a geographic map, symbolizing different relationships such as movement or trade. Lines and arrows can encode information through the color, width, style, direction, … See Figure 3.8(4).



(1) (2)

(3) (4)

Figure 3.8: In order of appearance, a choropleth map, a graduated symbol map, a cartogram and a flow map.

Innovations

The development of powerful Geographical Information Systems (GIS) enables not only the automated generation of basic graphs but also the experimentation on variations of these representations to improve the amount of transmitted data.

Some of these improvements are achieved through animation and user interaction. Other diverse approaches include providing richer information in support of sophisticated tasks, such as validating spatio-temporal distribution models (BirdVis, [32]) by visualizing computed operations.

At the same time, new geographic visualization techniques are being proposed. Scheepens et al. [33] created the composite density maps for multivariate trajectories (Figure 3.9), which support of 3D rendering capabilities. Tominski et al. [34] present maps drawn in two dimensions that contain the data in different planes of the third dimension, therefore generating 3D diagrams.

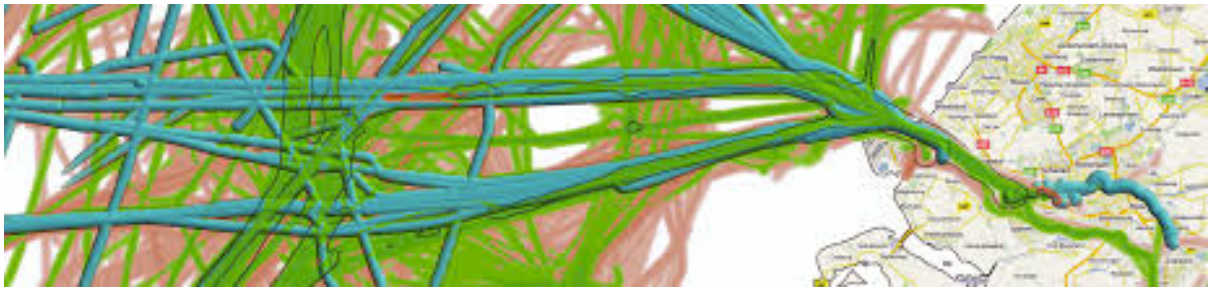


Figure 3.9. Composite density map for multivariate trajectories. An accident risk map of passenger vessels (turquoise), cargo vessels (orange), and tanker vessels (green) in front of Rotterdam harbor. The highest density at a location determines the color of the cell [35].

Multivariate data

Multivariate data can be considered itself as a general data type and it is commonly present in visualization and analytical tasks. Most of these problems consist of discovering or exploring complex relationships between many different attributes or dimensions. Consequently, various visualization techniques [19] have emerged to help analysts identify, locate, distinguish, categorize, cluster, rank, compare, associate, or correlate the underlying multivariate data.

Below is a review on different kinds of representation suitable for this problem, following the categorization proposed by Keim and Kriegel [20] except from graphs, who have already been discussed in Section 3.4.1.

Geometric

The Geometric Projection Techniques (GPT) consist on projecting the data of the different dimensions in the space, with the aim of finding interesting relationships between them. Most innovations on visualizations of high-dimensional data use GPT, and they can be considered an extension of the representations reviewed on Statistical Distributions (Chapter 3.4.1). The Parallel Coordinate Plot [36] is particularly suitable to represent multivariate data as it can display a larger number of dimensions than the other reviewed techniques.

The adaptation of those techniques to the high-dimensional space is done usually through statistical reduction techniques, such as principal component analysis or the ‘projection pursuit’ [37]. Far from limited to that, research on the field keeps exploring new projection techniques [38,39].

Finally, successful representations have been proposed based on integrating multiple geometric approaches to avoid limitations of using them individually [40, 41], like “Flexible Linked Axes”, which connects various geometry-based techniques, such as PCPs and scatterplots.

Icon-based

Another set of techniques proposed to visualize multivariate data consist on encoding the different parameters through iconic displays. Some examples on that are the well-known Chernoff faces [42], the stick figure technique [43] or the shape-coding approach [44].

The effectiveness of iconic-displays is usually limited to low-dimensional problems. Indeed, if too many variations can be observed in the icons, these become less readable and the visualization easily gets saturated. Nevertheless they are appropriate representations once several attributes of interest have been selected and can be easily integrated within spatial and cartographic visualizations as we can see in the Figure 3.10.

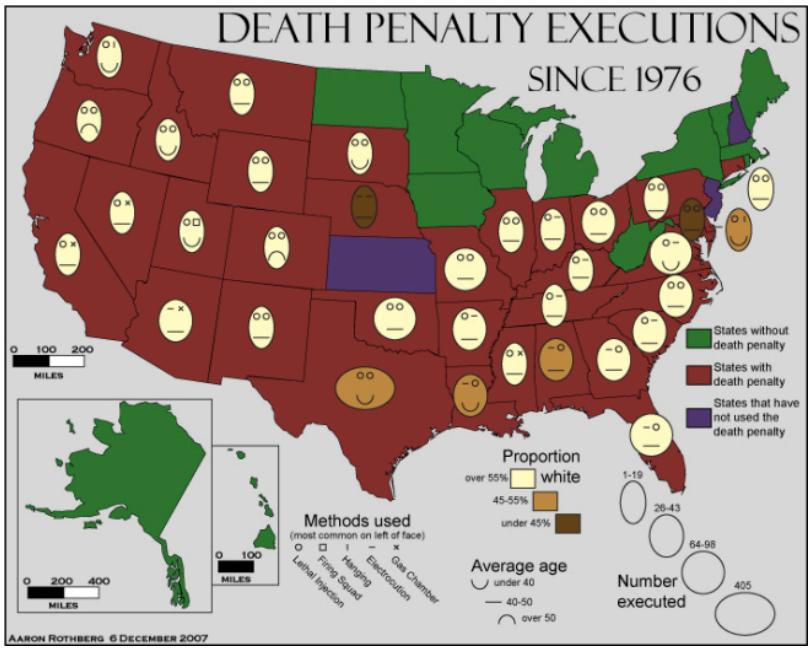


Figure 3.10. Chernoff faces are used to analyze the Death Penalty Executions in the USA. The color of the icon represents the predominant race, the smile the average age, the size the amount of executions and the eyes the methods used.

Pixel-oriented

Pixel-oriented techniques are based on the idea of mapping the every value of data with a pixel of the representation on a particular color, enabling therefore the direct visualization of a big amount of data. Every attribute is presented on a window and the whole representation is a mosaic of them, with the possibility of a global one (Figure 3.11). The arrangement of the data on each representation, which gives us the structure, may depend on some existing order (such as temporal data) or on query based information (interactive process).

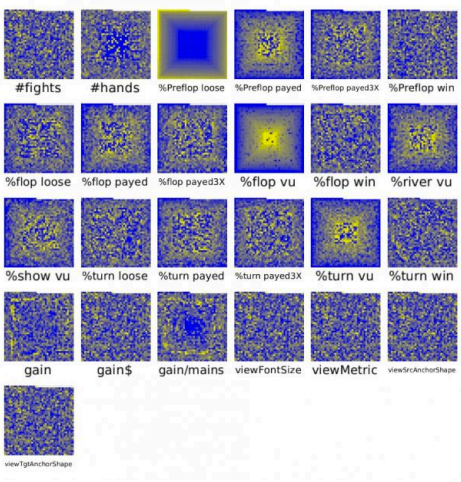


Figure 3.11. Pixel Oriented Visualization.

3.5. Challenges in Information Visualization

The generation of appropriate and meaningful visualizations is not a trivial task. Beyond the topics previously discussed, some major challenges within InfoVis will be considered below.

Generalization

Most of the improvements and complex visualizations have been developed to satisfy the requirements of a specific application or specific aspects of a visualization technique [8]. Although domain specific visualizations are useful and important, it is still a challenge to work on global strategies such that the advancements in particular fields can easily be extended to other domains.

Determining topologies of information and linking suitable representations to them is a step towards that.

Visual scalability

Visual scalability is defined as the capability of visualization tools to effectively display large data sets in terms of either the number or the dimension of individual data elements [45].

This is a challenging issue specially given the continuously increasing amount of available information. Indeed, to solve the same problems, analysts can use now a vast amount of indicators, records and other data.

This is a problem that we already treated when visualizing multivariate information, together with some data reduction techniques that have been developed to solve it, such as

sampling, filtering, clustering, PCA, and multidimensional scaling [45]. However, these techniques are still not satisfactory for this growing problem and more research on this field is being undertaken, exploring combination of existing techniques together with new ones.

Integrated analysis of heterogeneous data

Heterogeneous data is a problem that raises frequently in real world applications, and it is still a remaining challenge for most visualization applications and frameworks.

Indeed, with the rise of big data analytics, data comes frequently from multiple sources and in varying formats [8]. The automatized integration of this diverse information is key to be able to analyze it, but it is not clear how to solve the task in a global and abstract way.

An example of data integration would be to analyze the information about human rights provided by the world bank and compare it with the local data provided by the governments of several countries and the data from the African Union, all without a manual preprocessing of the information.

Assemblage of Visualizations

Finally, after reviewing a large amount of visualizations it is clear that there is not a single formula to highlight all the information. Therefore, in most of the cases, to solve a problem we need to provide different linked representations that provide insight in different aspects of the information visualized on them.

However, there is still work to do on assembling visualizations in flexible and easy ways, as most of the tools provided don't enable the user to change, nest and compose representations from more basic techniques. With respect to that, some work has been already undertaken as we can see in the chapter 3.4.1.

3.6. Conclusion

Through this section we have discussed the main theoretic points within Visualization in general and Information Visualization in particular, which have lead to the proposed solution. The design elaborated in the Chapter 6 tries to be close to the human perceptual process, in order to provide a natural language to generate visualization.

The reviewed techniques go lot further than the ones provided in this solution. However, the overview is necessary to create general structures that enable later integration of different representations. In addition, we will refer to them in the Chapter 7 to evaluate the quality of the solution.

4. Delivery - Proposed solution

In this section a review is done on the existing approaches to handle the problem. Following, a description of DSLs and Functional Programming is provided, which are the selected techniques to provide a solution. Finally, the proposed solution is presented.

4.1. Existing approaches

Current approaches to create Information Visualizations range from very simple and ad-hock solutions to complex and specialized software.

Beyond this first sector of ad-hock visualizations and basic representations generated with statistical and analytic software solutions, this chapter will focus at the existing frameworks and systems for InfoVis.

Different frameworks [46, 47, 48, 49] have emerged, modeling different aspects of visualization techniques. Among them we may find WebCharts [47], which provides a strategy for incorporating new kinds of representation to existing applications, and a framework described by Chen and Jänicke [46] which should be able to characterize the visualization process based in information-theory.

On the other hand, most of the developments consist of libraries or toolkits for developing visualizations. Some examples of that are *Improvise* [50], the *InfoVis Toolkit* [51], and *Prefuse* [52], characterized by generalizing steps of the visualization process, allowing to create linked views, using generic data structures and visualization algorithms or providing interaction and animation techniques.

Finally, *Protovis* [53] is a visualization system that uses Domain Specific Languages (DSL) to overcome the common difficulty of most of the previous systems on extending and tailoring the provided visualizations. It employs JavaScript and *Scalable Vector Graphics* (SVG). *Protovis* is no longer under active development, but its team works now on a new visualization library called *Document-Driven Documents* (*D*3) [54], which has become a very popular toolkit to construct interactive visualizations on the web. *D3* will be the golden standard that will be used to evaluate the solution provided by the present project.

4.2. Domain Specific Languages

A Domain Specific Language (DSL) is defined as a computer programming language of limited expressiveness focused on a particular domain [61].

DSLs, indeed, define the components of a particular domain and the operations that they can perform, thus enabling a flexible interaction. Therefore, this approach can provide great possibilities for composition, enabling the construction of complex and customized structures from basic ones.

In addition, the benefits of using DSL in the context of graphics have already been remarked, as it is described in the paper of Duke [62], where a summary of different DSLs on functional geometry, functional animation and image manipulation and graphical synthesis are presented.

There are two main approaches to implementing DSLs. The traditional one would be to develop a standalone language, with a customized syntax fitting the characteristics of the domain. The second approach is to build DSL embedded in a host language.

Some of the advantages of this approach are that there is no need to develop a compiler for the custom syntax and, mainly, that it emerges the possibility of exploiting the capabilities of the host language.

Within the context of this project, an embedded DSL can provide opportunities to handle the composition of representations through customized primitives adapted to the specific needs of the data to be displayed. Moreover, it will be built over a graphic DSL called Diagrams (discussed in section 5.2) which will enable the reutilization of basic primitives to build complex visualizations.

4.3. Functional Programming

Functional programming is the implementation of a formal system called lambda calculus, proposed by Alonzo Church. This system, which is equivalent in expressiveness to the Turing Machine [64], is based on function abstraction and application using variable binding and substitution [66].

A functional programming paradigm has multiple advantages, a review of which can be found in [65]. A major advantage is the lack of side effects, consequence of the fact that there’s no variables or assignments. This is known as referential transparency.

Furthermore, this paradigm is especially suitable for embedding Domain Specific Languages, as it is discussed in [63]. Among the reasons we can consider the expressive power that come with functional abstractions[65] such as algebraic data types, lazy evaluation and higher-order functions. These features together with a description of the chosen language for the development, Haskell, will be discussed in Section 5.1, being exemplified with the design of the visualization DSL.

4.4. Proposed Solution

The proposed solution for the problem presented in Section 3.1 is to develop an embedded DSL called Maps, focused on creating Geographic Visualizations.

Within the different challenges discussed in Section 3.5, and specifically in Section 3.5.4, the solution intends to tackle the problem of the assemblage of visualizations and the heterogeneity of data.

As discussed in Section 4.2, a DSL is a suitable approach to provide a fast and flexible way to generate customized visualizations, being able to create a big number of composed visualizations in few lines of code. This, within a process of Knowledge Discovery, turns to be a powerful capability. The research task can be, therefore, undertaken in an incremental way: starting through the generation of basic visualizations combining a big amount of parameters, and progressively filtering interesting cases and composing more specific views looking for insight.

The solution intends to construct a suitable model of Geographic Information, in order to provide support for heterogeneous data by providing functions to load and combine information from different sources, either maps, country structures or geographical data. In addition, basic functions enable transformations of the loaded data, which can be easily extended taking advantage of the underlying programming language (Haskell).

Indeed, Haskell provides a novel way of dealing with this kind of problem, specially through the use of Algebraic Data Types, Higher-Order functions and Type Classes. These concepts and their applicability to solve the problem are discussed in Section 5.1 and 6.

5. Framework

Within this section there is a summary of the framework in which the solution will be built, including the used programming language, Haskell (Section 5.1), the DSL on which relies the development, Diagrams (Section 5.2) and the environment (Section 5.3).

5.1. Haskell

Haskell is a non-strict, purely functional language. Purity consists on preserving referential transparency, described in Section 4.3. Non-strictness or Laziness refers to a delay on the reduction of function arguments. The returning values are not calculated until they are needed, which for example makes possible infinite lists. Laziness is possible thanks to referential transparency: because of the stateless characteristics of the programing paradigm, the order in which the functions are evaluated is not important, and therefore that operation is delayed or even not executed if it's not necessary.

Some of the most interesting features of Haskell are illustrated and discussed in Section 6, through the design and development of the DSL. Those are the following,

- Higher order functions and Lambda functions in Section 6.3
- Algebraic Data Types, in Section 6.3
- Pattern Matching and Recursion, in Section 6.6
- Types and Type Classes
- Monoids, in Section 5.2

5.2. Diagrams

This system is built over Diagrams, an embedded DSL that enables the creation of graphics in a powerful and modular way. Diagrams plays an important role on this system, not only for providing a powerful way of generating graphics with Haskell, but also by being an important design reference: Its declarative approach has been the main inspiration to design Maps.

Key concepts

The idea behind Diagram's approach is to mimic the way people actually think on spatial relationships. Thus, it enables the generation of complex graphics in a declarative fashion, by specifying *what* a diagram is (e.g. draw a triangle with a circle inside), not *how* to draw it (e.g. draw lines in the calculated vertices and a circle with center in x,y).

Relative positions

One of the main particularities of Diagrams is that it proposes the generation of graphics based on compositions, aggregations and transformations of simple primitives, whose positions and sizes are relative. This is in contrast to most of the existing libraries, which are based on a process of building the graphics directly over a coordinate system.

Briefly, there is never any global coordinate system to think about; all the calculations are done relative to diagrams' *local* vector spaces, specifically using the center of every diagram.

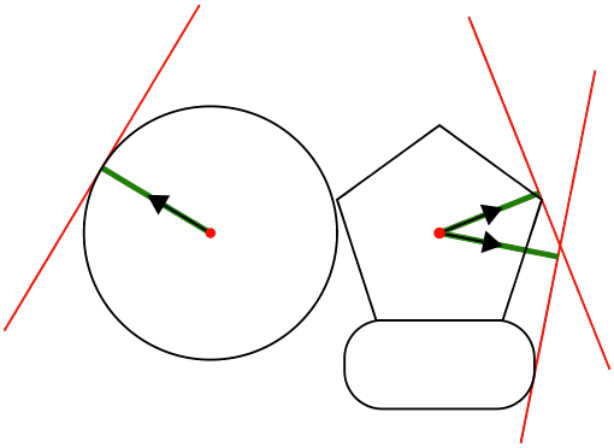


Figure 6.2. The envelope function.

In order to be able to position diagrams relative to one another, diagrams track bounds information using envelopes [], which roughly answer the question: "how far in this direction must one go before reaching a perpendicular (hyper)plane that completely encloses the diagram on one side of it?" and can be visualized in the figure.

Although this is an advantage to generate nested diagrams, mapping arrows and glyphs in the regions, it is less suitable to visualize the loaded SVGs. However a way to position the regions through coordinate positions (as they come in the SVG) is discussed on the design of Maps.Spatial.

Monoids

Diagrams does a wide use of monads, being most of the elements provided in the standard library implementation of the monoid typeclass, provided by the core library of Haskell. Among them, transformations, trails, paths, styles, colors, envelopes, traces...

A monoid has the following characteristics:

- Associativity: when adding several values, it doesn't matter how we group the terms.
- It has an identity element. An identity element can be added to any other number without changing its value.

These properties hold for many associative binary operations, being therefore examples of monoids. For instance, the operation sum operation within integers, with an identity equal to 0 is an example of a monoid.

5.3. Environment

Version Control

This project uses Git, a lightweight distributed revision control system. As it is very popular, its use facilitates future contributions from other Haskell hackers. This is done through GitHub, a Web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git.

Build system

The Haskell Cabal is the Common Architecture for Building Applications and Libraries. It is a system for building and packaging Haskell libraries and programs. It defines a common interface for package authors and distributors to easily build their applications in a portable way. Cabal is part of a larger infrastructure for distributing, organizing, and cataloging Haskell libraries and programs [56].

6. Delivery – Design and Implementation

The following section explores the whole design process, which is detailed on a deductive and progressive fashion, according to the real succession of events. The aim following this structure is to visualize not only the result but also the agile methodology followed on the process.

6.1. System Architecture

The System consist of three differentiated modules that can be considered as independent among them.

The first one, Spatial, creates a hierarchy of spatial areas. It enables operations of merging, grouping, deletion…, and it is intended to provide flexibility on the creation of a geographical structure, over which the data will be shown with respect to the different levels. Its code is found in the module

```
module Data.Spatial
```

The second one, DataType defines three types of data based on the topology proposed in Section 3.3. Basic functions to map these data into suitable retinal values (Section 3.3.5) are provided by default, allowing however to define a customized behavior. Data conversions are also facilitated within this module. Its code is found in

```
module Data.DataType
```

In the third place, Map provides the functionality and syntax to generate geographical visualizations, combining the elements defined with the prior modules. It is strongly inspired in the way Diagrams constructs composite elements, trying to maintain a great flexibility in that sense.

```
module Data.Map
```

Complementary, two additional modules are provided to facilitate the load of SVG maps and CSV data.

```
module DataLoader.CSV
module MapLoader.SVG
```

6.2. First Iteration: Risk reduction

During the first iteration the focus has been placed on reducing risks. This has been done by testing the main libraries and verifying that they provide enough capabilities to construct the solution over them.

During this phase the following simple operations have been achieved:

- Drawing a Polygon by its coordinates with Diagrams
- Loading a SVG file
- Drawing an arrow between two polygons
- Loading CSV files

6.3. Initial design issues

What is a region

We can consider that maps are visualizations consisting of a compound of regions. Regions are fundamentally defined by their area, their boundary and their position within the map. In addition, we consider that there are belonging relationships among regions. E. g. a region can belong to a group of regions, like West Yorkshire belongs to England and England belongs to UK. In addition, we can perform operations over regions, such as calculating the center or the areas. Regions are defined as follows,

```
data Region =  Base  {code :: Code, coor :: [Path V2 Double]} |
               Group {code :: Code, level::Int,
                       nested :: [(String, Region)], ids :: [String]}
               deriving (Eq, Show)
```

Algebraic Data Types

The prior code is an example of al Algebraic Data Type, which consist, somehow, of a composition of other different types.

The case of Region is an example of a sum type, also called disjoint union and characterized by the operator | separating the two components of Region, namely Base and Group. There are as well product types, like tuples and records.

Values of algebraic types can be analyzed with pattern matching (Section 6.6), which uses the constructor to identify the values, enabling that way the extraction of the data that they contain.

Are regions organized in a hierarchical fashion?

A simple approach drives us to organize regions in a pure hierarchical fashion. Indeed, if we look at the prior example, we could generalize it stating that towns belong to counties, counties to countries, countries to continents and so on (in the case of the UK among others

we would add the state in between the country and the continent, which wouldn't be a problem).

However, there are other possible organizations which we wouldn't support with this approach. If we are analyzing data about migrations, we may be interested on visualizing differences between the European Union, the Schengen Area and the countries of the G8 forum. As three different and potentially attractive groups of countries, we may want to get insight on the differences on the migration rates among them. However, it is evident that we can't represent this in a hierarchical structure, as many countries are part of the three groups.

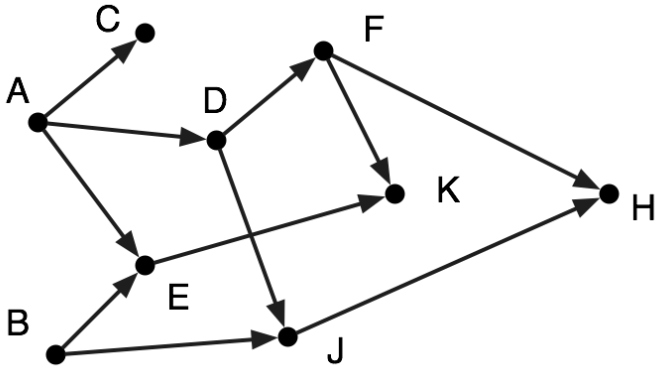


Figure 6.1. Directed acyclic graph

Therefore, it is decided to choose a more flexible approach, being a directed acyclic graph (DAG) (Figure 6.1) the most appropriate and flexible data structure to relate certain regions with others. Note that all the relationships are established from left to right, so no cycles are allowed. We may organize these classifications in different levels that will be, a priori, calculated adding one to the higher level of the descendants of a node.

The type synonym here contains the structure of the spatial tree, which consist of a sequence of lists of pairs (Code,Region). Every element of the sequence corresponds with a level, so therefore when a level is selected the whole list is extracted to be mapped. In addition, as we have seen in the Region definition, Group regions can define a set of nested ones.

```
type SPtree = S.Seq ([(Code, Region)])

addHierarchy:: [(String, [String])] -> Int -> SPtree -> SPtree
addHierarchy struc level tree = foldl (\ tree (code, list)-> groupRegions
code level list tree) tree struc
```

Higher order functions and Lambda functions

In the previous set of code we can observe two interesting features of Haskell. In the first place, we have a lambda function in

```
(\ tree (code, list)-> groupRegions code level list tree)
```

Lambda functions are declared on the row, prefixed by a \ . They can be used inline when the will be used just once.

In addition, higher order functions such as foldl, where the lambda is declared, are characterized by taking functions as a parameter or returning functions (or both). For example, fold takes a function, a list and an initial value and maps every element of the list through the function, updating the initial value provided. In the current code, this is done to map a received set of tuples (code, nested) into a grouped tuple containing all the elements (code, [elems]).

6.4. DataTypes DSL

Data types, discussed in the Chapter 3.3 and specifically the levels of measurement in Chapter 3.3.3, play an important role to enable the system to infer information at different levels of detail and to automatize the selection of appropriate retinal variables (Chapter 3.3.5), for both the selected representation and the attribute to be analyzed.

This is modeled using TypeClasses to specify the characteristics of data, together with the operations that can be realized with each one of them.

An initial set of Data Types will be designed following the Stevens scale, to provide a standard way of dealing with data. As an example, we present the code for the nominal type:

```
type StvNominal = String

data Context a = NominalContext {values :: [a], brewerset :: ColorCat} | …
```

In addition, the model should facilitate data transformation, by easily converting data from a different types to others. In the following example we transform the ratio data of the gbp of the different countries to nominal data, classifying the countries on a desired point to visualize a clear boundary among those with an elevate GBP from the rest.

```
dsToNominal :: DataSet a -> (a -> Int) -> [StvNominal] -> DataSet StvNominal
dsToNominal EntitySet{v=ds} f values =
        let list = [ (k, values !! (f d))|(k, d) <- (M.toList ds)]
        in EntitySet {v=M.fromList list, c= defNominalContext {values = values}}
dsToNominal RelationSet{rv=ds} f values =
        let list = [ (k, k2, values !! (f d))|(k, k2, d) <- ds]
        in RelationSet {rv=list, c= defNominalContext {values = values}}
```

Linking data categories, retinal variables and types of representation

Indeed, the use of data categories has repercussions all over the pipeline. As shown in the Figure 6.2 (in a non-comprehensive way) Data categories are associated with specific retinal variables. Thus, nominal data should only be visualized through the hue or shape (of an icon), and interval data with lightness and size (or width).

Then, these visual variables can be available for different basic shapes. Area should display hue and lightless, but not size because it would distort the shape itself. However, line can display size modifying its with. A compound shape is a compilation of basic shapes.

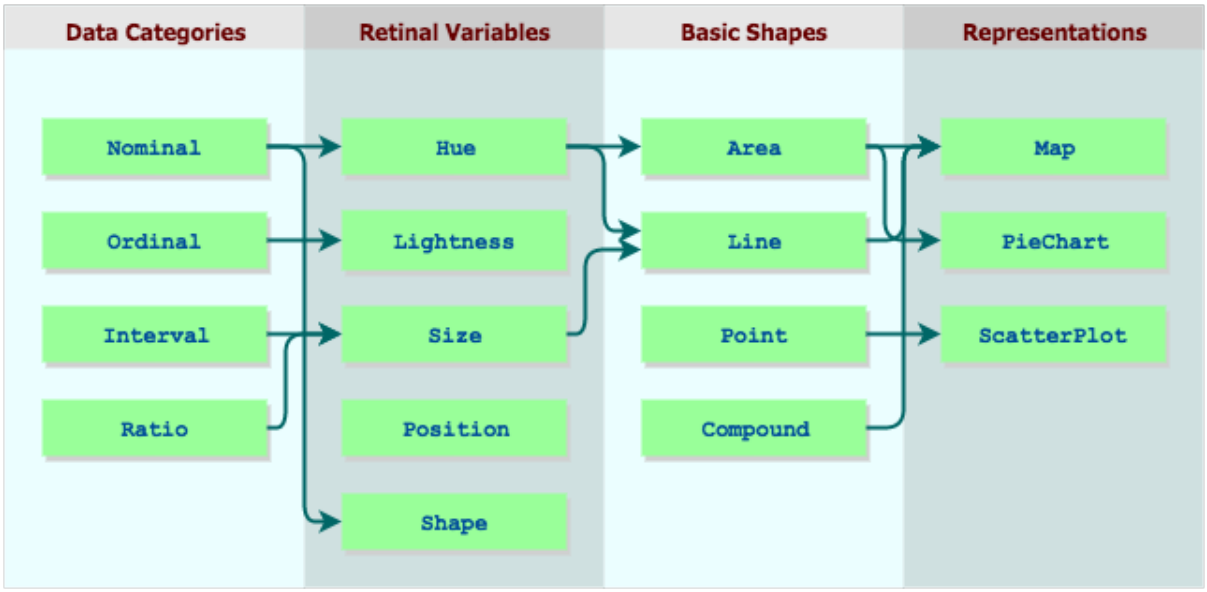


Figure 6.2. Non-comprehensive set of relationships between different levels of abstraction.

Although these relations are not restrictive, they can be used to obtain automatic inferences of the data and the appropriate ways to represent it.

The implementation of this feature is realized using Haskell's TypeClasses as we can see on the following example:

```
class RetHue a where
        hue :: Context a -> Maybe a -> Kolor
        hue' :: Int -> Context a -> Maybe a -> Kolor

instance RetHue StvNominal where
        hue ctx@NominalContext{values = vals} = hue' (length vals -1) ctx
        hue' _ _ Nothing = (brewerSet Greys 9) !! 0
        hue' size ctx (Just value) =
                let
                        vals = values ctx
                        index = ei (L.elemIndex value vals) 0
                        bs = brewerset ctx
                in (brewerSet bs 8) !! index
```

6.5. Spatial DSL

Spatial relationships and Design of the DSL

The first version of the object Region is the following:

```
data Region = Region { code :: String, coor :: [Path V2 Double]}
   deriving (Show, Eq)
```

This data type enables us to characterize the regions of the map and to associate them with the corresponding path which will be used by Diagrams to print it later.

However, this doesn't allow us to create a hierarchical structure, reason why we will create a recursive type:

Here, we need to define two different possible nodes, a Leaf, which contains information about the specific path of the region or a Branch, which contain a subgroup of Leafs. The area occupied by a Branch node will be calculated using its descendants.

```
data Region = Leaf { code :: String, coor :: [Path V2 Double]} | Branch
{code :: String, nested :: [Region]}
        deriving (Show, Eq)
```

Pattern Matching

Pattern matching consists of specifying patterns to which some data should conform and then checking to see if it does and deconstructing the data according to those patterns [60].

We apply pattern matching in many functions, like *containsRegion*. This function looks for a region within another one, returning true if it is contained or false if not. It takes two parameters, the code of the region to look for and the region that may contain it.

With pattern matching we first capture the base case, in which the container is a Leaf and therefore can't contain nothing. Then, we explore the nested attribute of the container (it will be a Branch) that we capture in '*b*'. If there is any region with *ncode* within *b*, then we return *true*, else we return *false*.

```
containsRegion:: String -> Region -> Bool
containsRegion ncode Leaf {} = False
containsRegion ncode Branch {nested = b} = any (isRegion ncode) b
```

Recursion

Recursion is actually a way of defining functions in which the function is applied inside its own definition. Definitions in mathematics are often given recursively [60].

In this module we have an example of recursion, in the function *deepContainsRegion*. This is a variation of the previous function in which we return true if the region is inside the provided container or inside any of the nested containers inside of it.

We use guards to check if the values passed to the function satisfy some requirements, in a similar way to an *if* expression. Again, the first line captures the base case, in which the region is contained in the current level (we use the previous function to know it). The second line analyzes the contents of the nested regions, using foldl (it reduces the list to some single value). There, it checks again wether any nested element contains the *ncode,* calling herself again in a recursive fashion.

```
deepContainsRegion :: String -> Region -> Bool
deepContainsRegion ncode region
  | containsRegion ncode region = True
  | otherwise = foldl (\acc x -> if deepContainsRegion ncode x then True
else acc) False (nested region)
```

Note that the fold function is very common and it takes three parameters: the accumulator function, the initial value and the list that will be folded.

6.6. Maps DSL

Maps, the representations that are being developed within this project, are composed of regions and drawn using different elements such as areas, lines, icons, …

Linked representations

Linked representations, also among the objectives of the project, come in two flavors:

- Nested representations. Already discussed
- Compound visualizations. One would imagine, for example, that we have a set of visualizations to analyze migrations. These consist of a colored map showing the migration rates, a pie chart showing the percentage of male and female migrants, a line chart showing the global rate evolution and another pie chart with age ratings. When we change the zoom from Europe to the UK, we expect that all the linked visualizations update to this particular subset of the data. This is done directly using Diagrams, which provides functionality to arrange and mix different diagrams in the desired way for the user, like this code that arranges four diagrams in a single visualization as a grid.

```
Compound =  mapArrows # lw none ||| Choropleth # lw none
   === mapGroupArrows # lw none ||| GroupChoropleth # lw none
```

Maps DSL design

The main purpose of the maps DSL is to provide a syntax which allows the user to decompose the process of elaborating a representation being therefore able to include functions and customize its subsections.

Below, through mock functions, it is shown the process of design of the chain to construct the visualization.

We start by this initial structure. Be noted that the types are not the real ones. It is intended to map the spatial structure / “($ 4.5)” / to the prior functions that will transform it.

```
f1 :: ( Int -> Float ) -> Int -> Float -> String
f1 _ _ _ = "Hello"

f2 :: ( Int -> Float ) -> Int -> Float -> String
f2 _ _ _ = "World"

fx = map ($ 4.5) [(f1 (\x -> 3.4) 4),(f2 (\x -> 3.4) 4)]
```

We incorporate from changing the order of parameters to increase the readability.

```
from :: [Float -> String] -> ((Float -> String) -> String) -> [String]
from = flip map

fx = [(f1 (\x -> 3.4) 4),(f2 (\x -> 3.4) 4)] `Main.from` ($ 4.5)
```

We propose a pattern to be able to chain methods, that means, to apply one property like color to a region and next, to the same region, brightness, for example.

```
fz :: Int -> Float -> Float
fz i f = f + fromIntegral i

fx2 = 1.2 # fz 1 # fz 1
```

In this case we are interested to map a function instead of a value

```
fz :: Int -> (Float -> String) -> (Float -> String)
fz i f = f

fx2 = (\x -> "3.4") # fz 1 # fz 1
```

The combination of all together results on that:

```
fx3 = [(\x -> "3.4") # fz 1 # fz 1, (\x -> "3.4") # fz 1 # fz 1]
`Main.from` ($ 4.5)
```

We include functions that enable us to filter part of the spatial structure, to apply the properties to subsets like the levels of detail.

```
--Design of all and sub functions
fAll :: ((Float, Float) -> String) -> Float -> String

fSub :: Int -> ((Float, Float) -> String) -> Float -> String
```

```
--redefine nestable funcs to accomodate:
fy :: Int -> ((Float, Float) -> String) -> ((Float, Float) -> String)

res = [ fAll ((\x -> "xx") # fy 1 # fy 1), fSub 1 ((\x -> "xx") # fy 1 # fy
1)] `Main.from` ($ xx)
```

Increasing the legibility of the previous, we have:

```
res = [(\x -> "xx") # fy 1 # fy 1 # fAll, (\x -> "xx") # fy 1 # fy 1 # fSub
1 ] `Main.from` ($ xx)
```

The resulting structure would be as follows:

```
--THis is the function from, which maps a list of operations
from :: [Trre -> Diagr] -> ((Trre -> Diagr) -> Diagr) -> [Diagr]

--Design of all and sub functions
fAll :: ((Trre, Trre) -> Diagr) -> Trre -> Diagr

fSub :: Int -> ((Trre, Trre) -> Diagr) -> Trre -> Diagr

--redefine nestable funcs to accomodate:
mapData :: Int -> ((Trre, Trre) -> Diagr) -> ((Trre, Trre) -> Diagr)

printRegions :: (Trre, Trre) -> Diagr

res = [printRegions # mapData 2 # mapData 3 # fAll, printRegions # mapData
1 # mapData 2 # fSub 1 ] `Main.from` ($ tree)
```

To conclude, this is the main syntax that makes possible the combination of all the prior structures. Examples of its use will be shown on the next section, using it to analyze migration data.

7. Evaluation and Study case

7.1. Methodology

The methodology followed to evaluate the solution provided within this project consists, firstly, on elaborating a Study Case about human migrations using the data provided by The World Bank and, secondly, on comparing it to a Gold Standard, namely D3js. This should enable us to both evaluate the usefulness on an arguably standard problem and to highlight its advantages and disadvantages with respect to other frameworks.

The World Databank

Over the years and like various other organizations, the World Bank has been collecting data about global development. This has been published, alongside many other indicators, in the so called “World Databank”. This data is public and open for research purposes as part of an initiative to promote knowledge acquisition from Big Data.

In addition, additional indicators coming from other entities such as the UN and Amnesty International will be used in order to evaluate the capability to deal with heterogeneous data.

Choosing the Gold Standard

With respect to the golden standard chosen, D3 is a popular JavaScript toolkit that enables the creation of different Information Visualizations. It counts on a big amount of plugins to generate different representations, being one of them specialized on generating geographic visualization.

Instead of focusing on the concept of a representation, the toolkit relies on existing web standards (DOM, CSS, SVG) and uses its primitive functions to map data on representations.

7.2. The Study Case

This study case consist of a visualization exercise to understand global migrations using the data provided by the World Bank. Through this section, different aspects of migrations will be represented. The code and concepts used to elaborate the visualizations will be discussed, together with a review of different alternatives.

Initial visualization of net migration

To start with, we generate a Choropleth (Section 3.4) with an indicator about the net migration.

To do so, we must load, on the one hand, the file with the indicators (line 1) and on the other hand the SVG map (line 2) where the information is going to be projected.

Next, the spatial hierarchy (SPTree) is built in line 5 and the DataSet in line 6. Note that we must specify which kind of data will be loaded, in this case Ratio (Stevens scale).

Finally the representation is composed in line 7:

- We build the diagram (M.buildDiagram) resultant of
- Printing the regions
    - Mapping the data (M.mapData) over the Hue (M.mapHue) with ds (the DataSet)
- Extracted from the base level (M.fAll)
- Of the previously generated tree (`M.from` ($ tree)))

We must note that the standard colors for ratio data are used to represent this information, but that it could be easily changed by setting the palette property of the dataset context.

```
1- main :: IO ()
2- main = do
3-   csvData <- BL.readFile "data/netMigration2013.csv"
4-   regions <- parseFile "source/worldmap.svg"
5-   let tree = buildTree regions
6-   let ds = parseCSV csvData :: DataSet StvRatio
7-   let res = M.buildDiagram ([M.printRegions # M.mapData M.mapHue ds #
8- M.fAll] `M.from` ($ tree))
9-   mainWith ( res )
```

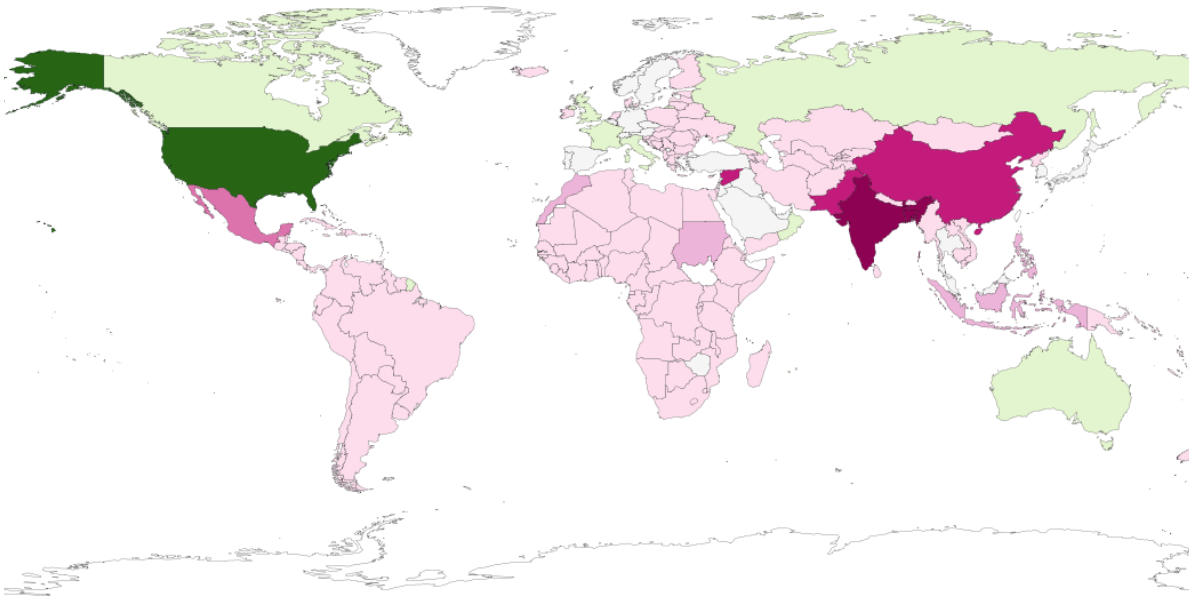


Figure 7.1. Choropleth showing the net migration, being the data 2013 the World Bank

As a result, the following map is obtained (Figure 7.1). It shows the countries with positive migration rates in green and those with negative rates in pink. USA is clearly over the rest welcoming migrants, while India, China y Syria are those with a greater amount of people emigrating. It is pretty clear, however, that the net measurement benefits countries with a large population over those who are smaller.

For this reason, on the next representation, a composed parameter will be created incorporating data about the population. The file and dataset for the population is loaded in the same way as the migrations (line 1 and 2).

In line 3 a parameter composition is realized just by stating the function which transforms the prior values into the new ones (\a b -> a / b) and the two databases implicated, m and p.

```
main :: IO ()
main = do
  migrations <- BL.readFile "data/netMigration2013.csv"
1-population <- BL.readFile "data/population.csv"
  regions <- parseFile "source/worldmap.svg"
  let tree = buildTree regions
  let m = parseCSV migrations 1:: DataSet StvRatio
2-let p = parseCSV population 1:: DataSet StvRatio
3-let mbyp = compositeDs (\a b -> a / b) m p
  let res = M.buildDiagram ([M.printRegions # M.mapData M.mapHue mbyp #
M.fAll] `M.from` ($ tree))
  mainWith ( res )
```

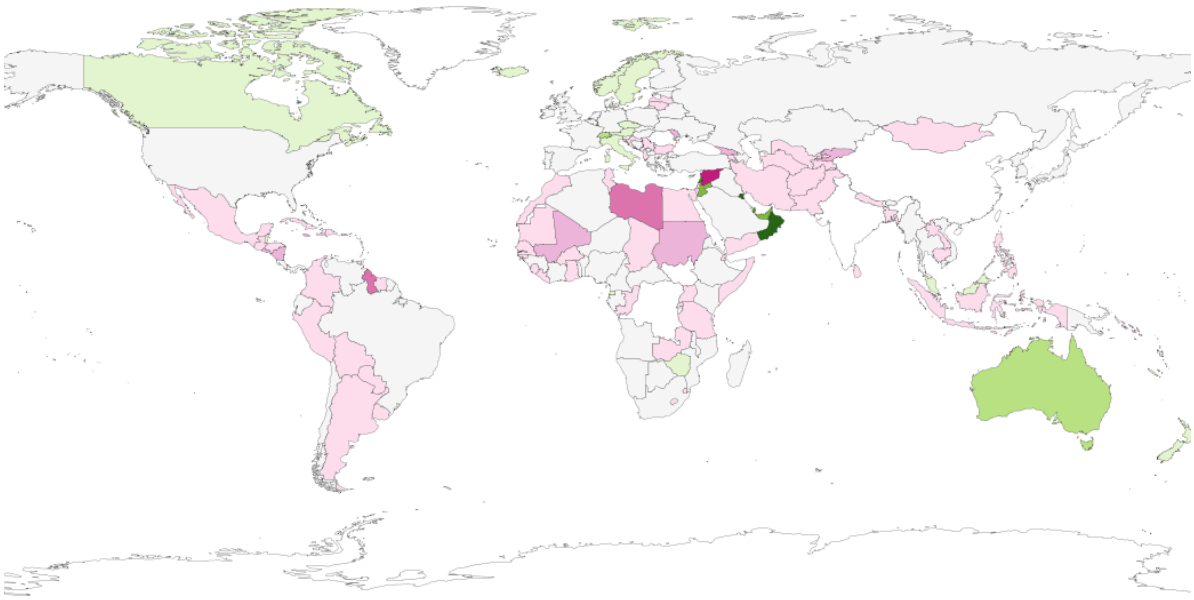


Figure 7.2. Choropleth showing the migration in function of the total population. 2013, the World Bank

In this map we can observe that big countries such as USA, China and India don't actually move that big amount of population if it is measured relatively.

It may be noted that Oman and other countries in the Persian Gulf have the highest migration rates. The reason is that they were countries with a reduced population which are now growing economically very fast, attracting therefore cheap labor from surrounding countries. In addition, the countries with a higher emigration rate are Syria and Libya (Figure 3.3), both of them in conflict.

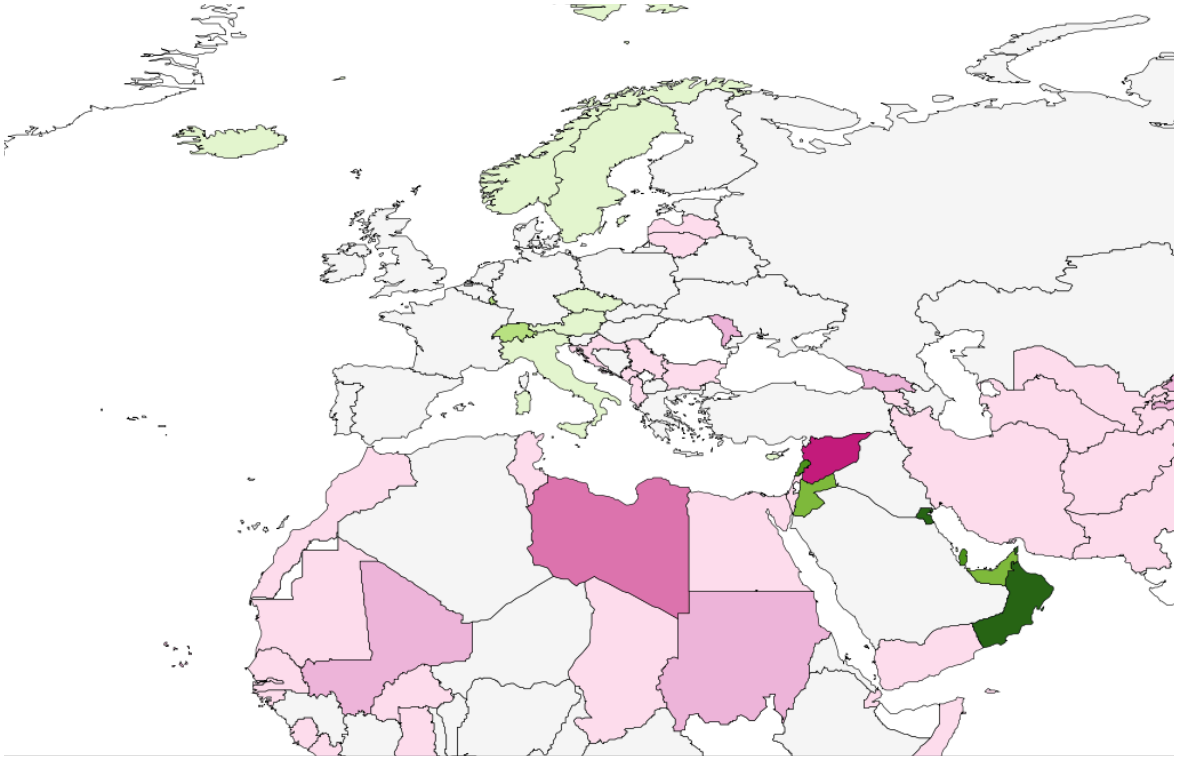


Figure 7.3. Choropleth showing the migration in function of the total population,

being the data . 2013, the World Bank

Visualization of migration flows

Next, we are interested on showing the migration flows, from one country to another. To implement this, we load the flows (line 1) and create the dataset (line 2) as before.

```
main :: IO ()
main = do
  migrations <- BL.readFile "data/netMigration2013.csv"
  population <- BL.readFile "data/population.csv"
1-flows <- BL.readFile "data/BilateralMigration2013.csv"
  regions <- parseFile "source/worldmap.svg"
  let tree = buildTree regions
  let m = parseCSV migrations 1:: DataSet StvRatio
  let p = parseCSV population 1:: DataSet StvRatio
2-let f = parseCSV flows 2:: DataSet StvRatio
  let mbyp = compositeDs (\a b -> a / b) m p
3-let res = M.buildDiagram ([M.printArrows f # M.fAll, M.printRegions #
M.mapData M.mapHue mbyp # M.fAll] `M.from` ($ tree))
  mainWith ( res )
```

In the line 3, we add “M.printArrows f # M.fAll” to add the arrows corresponding to the base level and defined in the dataset f.

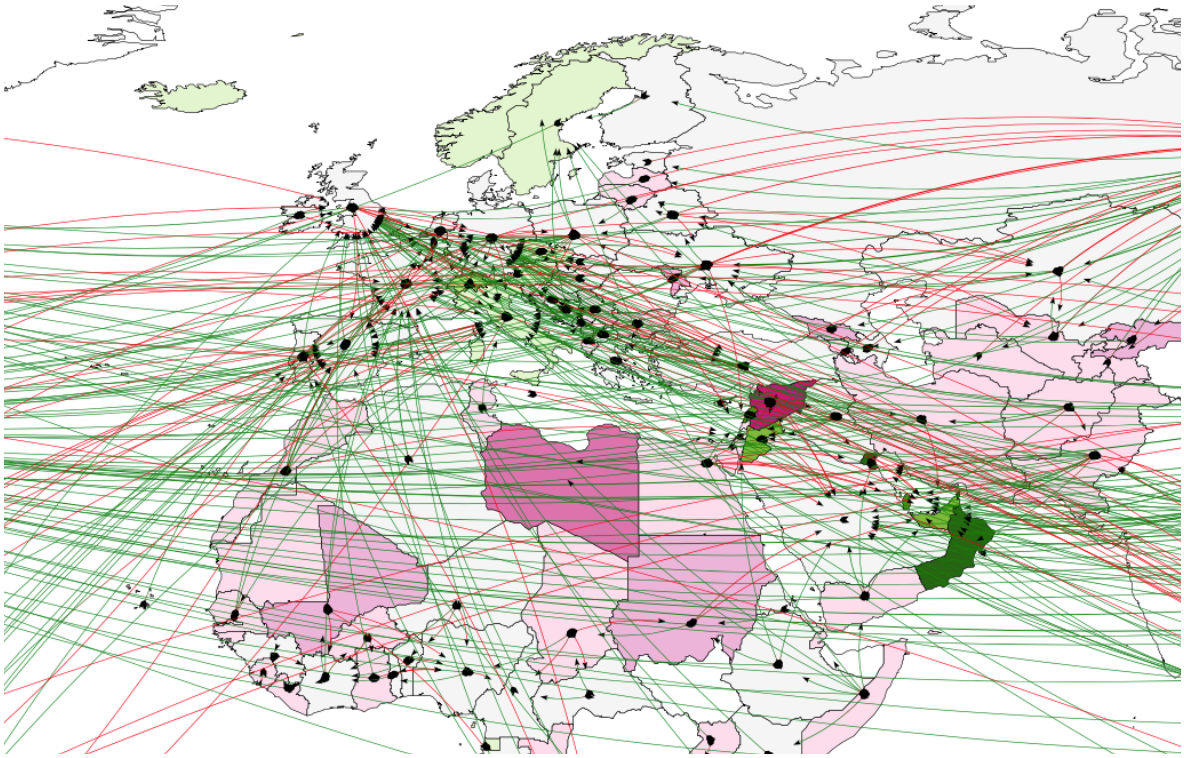


Figure 7.3. Choropleth showing the migrations. 2013, the World Bank

From this map we can extract interesting insight, such as the origin of the migration in European countries: In Spain it comes from Latin America and East Europe, while France concentrates the bigger amount of migration coming from Africa. The United Kingdom has few links with Africa (specially Nigeria) but most of the migrants come from the Middle East and India. In addition, there are a net of connections between the African countries placed under the Sahara desert.

A big help to detect the direction from which the arrows come from are the circles created around the center of every country. Also, the width of the arrows is related to the amount of migrants.

7.3. Comparison with D3js

In order to compare our visualization toolkit, we take the simplest example of a Choropleth diagram in D3js, published by one of it’s creators M. Bostock, encoding unemployment rates from 2008 in the counties of the USA.

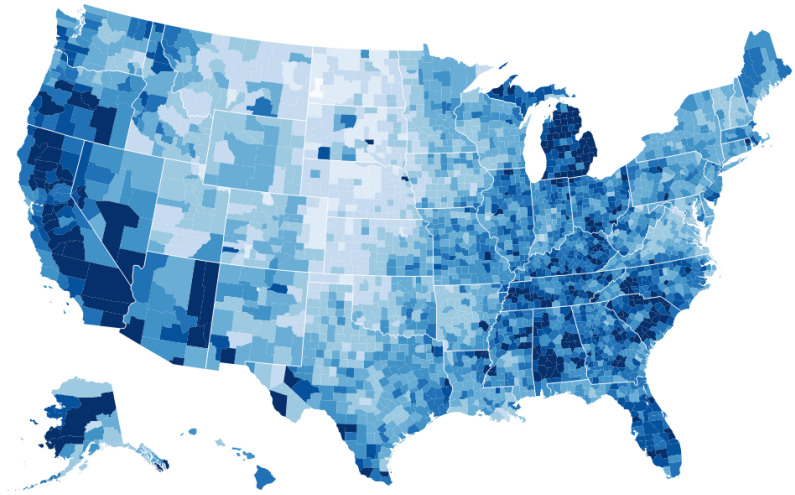


Figure 7.4. Choropleth showing unemployment rates in the USA. 2008, http://bl.ocks.org/mbostock

```
<!DOCTYPE html>
<meta charset="utf-8">
<style>

.counties {
  fill: none;
}

.states {
  fill: none;
  stroke: #fff;
  stroke-linejoin: round;
}

.q0-9 { fill:rgb(247,251,255); }
.q1-9 { fill:rgb(222,235,247); }
.q2-9 { fill:rgb(198,219,239); }
.q3-9 { fill:rgb(158,202,225); }
.q4-9 { fill:rgb(107,174,214); }
.q5-9 { fill:rgb(66,146,198); }
.q6-9 { fill:rgb(33,113,181); }
.q7-9 { fill:rgb(8,81,156); }
.q8-9 { fill:rgb(8,48,107); }

</style>
<body>
<script
src="https://cdnjs.cloudflare.com/aj
ax/libs/d3/3.5.5/d3.min.js"></script
>
<script
src="https://cdnjs.cloudflare.com/aj
ax/libs/queue-
async/1.0.7/queue.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/aj
ax/libs/topojson/1.6.19/topojson.min
.js"></script>
<script>

var width = 960,
```

```
import qualified
Data.ByteString.Lazy as BL
import qualified Data.Vector as V
import qualified Data.Csv as C

   main :: IO ()
   main = do
     csvData <- BL.readFile
"data/netMigration2013.csv"
     regions <- parseFile
"source/worldmap.svg"
     let tree = buildTree regions
     let ds = parseCSV csvData ::
DataSet StvRatio
     let res = M.buildDiagram
([M.printRegions # M.mapData
M.mapHue ds #
   M.fAll] `M.from` ($ tree))
     mainWith ( res )
```

```
    height = 600;

var rateById = d3.map();

var quantize = d3.scale.quantize()
    .domain([0, .15])

.range(d3.range(9).map(function(i) {
return "q" + i + "-9"; }));

var projection = d3.geo.albersUsa()
    .scale(1280)
    .translate([width / 2, height /
2]);

var path = d3.geo.path()
    .projection(projection);

var svg =
d3.select("body").append("svg")
    .attr("width", width)
    .attr("height", height);

queue()
    .defer(d3.json,
"/mbostock/raw/4090846/us.json")
    .defer(d3.tsv,
"unemployment.tsv", function(d) {
rateById.set(d.id, +d.rate); })
    .await(ready);

function ready(error, us) {
  if (error) throw error;

  svg.append("g")
      .attr("class", "counties")
    .selectAll("path")
      .data(topojson.feature(us,
us.objects.counties).features)
    .enter().append("path")
      .attr("class", function(d) {
return quantize(rateById.get(d.id));
})
      .attr("d", path);

  svg.append("path")
      .datum(topojson.mesh(us,
us.objects.states, function(a, b) {
return a !== b; }))
      .attr("class", "states")
      .attr("d", path);
}

d3.select(self.frameElement).style("
height", height + "px");

</script>
```

These two pieces of code are intended to generate the same output: a Choropleth mapping the values (loaded in both cases in TSV/CSV) using a map. This is loaded in SVG in the case of our DSL and in JSON in the case of D3js.

Simplicity or Easiness

As we can see in the previous comparison, the approach taken by the two systems is very different. Maps DSL models the visualization process and, therefore, the way of coding can be mapped to the way humans think of the task that is being undertaken. Literally:

- Load the files
- Transform them into Spatial Trees and DataSets of an appropriate Data Category
- Print a Visualization, mapping the values with the regions using a certain Retinal Variable

On the other side, D3js takes advantage of the prior knowledge of the developer of a Standard Language (HTML, SVG, …) and approaches the coding to the way in which the visual elements will be created. Therefore we define CSS class to set the properties of the elements, and we generate the visualization appending groups and paths to the SVG, setting the corresponding classes for the later manipulation. Data functions are provided to add the data easily all along this process.

Consequently, The Maps DSL creates the visualizations more straightforwardly, benefiting from the fact that it is a language exclusively designed for this purpose. It should be, for this reason, easier to learn as well, although it can be argued that the use of a functional programming background implies difficulties for many developers. This is where D3js is specially strong, by exploiting the prior knowledge that a big community of developers already has on SVG and other web standards.

In conclusion, the Maps DSL simplifies the process of visualization although it probably has a steeper learning rate.

Functionality

Projections

The main geographic related functionality provided by D3js is to handle projections in a very confortable way. The points of the map are loaded in JSON and translated to the desired projection through the lines:

```
var projection = d3.geo.albersUsa().scale(1280)
                       .translate([width / 2, height / 2]);
var path = d3.geo.path().projection(projection);
```

This enables the user to change the projection in a very efficient way.

The Maps DSL, however, doesn't provide any functionality to handle projections. The main reason being that the spatial data is loaded in SVG, and therefore already referring to a specific 2D projection. Given that there is a wide variety of open resources (SVG maps in different projections), the way in which a developer would change it in our system would be just changing the loaded SVG file by another with the desired projection.

Interactivity

A clear strength of D3js is the ability to deal with interactivity, which is provided through JavaScript, because SVG doesn't provide such possibilities.

The Maps DSL, however, doesn't provide interactivity at all, given that, to provide it, it would have been necessary to set a Client-Server architecture, with Haskell working on the Server and embedding JavaScript fragments within the generated visualizations to be able to provide a minimum interaction. This goes far beyond the scope of this project although it could be an interesting challenge to design a flexible and elegant way of dealing with it.

Data transformation, categories and retinal variables

In contrast, the Maps DSL provides a range of utilities to transform the data that will be visualized, which facilitates the management of multiple variables and combinations within the creation of a single representation. The type categories simplify the way in which the information can be correctly scaled and the retinal variables provide standard methods to visualize them by hue, lightless and size if appropriate.

All these capabilities are inexistent in D3js and JavaScript is not as expressive as Haskell to perform this kind of operations.

Spatial structure

Finally, the possibility of establishing a hierarchy of spatial areas enables the inference of the value of parameters which are set at different levels. For example, if I we use the first example provided in the study case, we can just change the level of information displayed to see the information at continent scale (in level 1) :

```
  let res = M.buildDiagram ([M.printRegions # M.mapData M.mapHue mbyp #
M.fSub 1] `M.from` ($ tree))
```

Specialization

The Maps DSL benefits from being built on a functional programming paradigm in order to provide specialization within tasks like data transformation. With Haskell it is very easy to create folds to reduce datasets, maps to modify them and so on. In addition, it takes just a few lines to add new instances of the defined type classes if there is a need for handling data that doesn't match the proposed topologies.

In addition, Maps DSL is been built over Diagrams, a very powerful graphic DSL, already discussed, and as a result, specializations on the representation module (Maps.Map) can be constructed and modified using it.

In the case of D3js, specialization comes with the availability of plugins that help to create different visualizations. The fact of being that close to the language in which the visualization is generated (SVG), makes it almost completely customizable, and in that sense goes clearly beyond the possibilities that the Maps DSL offers. However the process to do it can be a big task, because it requires a complete (re)implementation of the desired visual output.

To conclude, JavaScript doesn't provide natively high order functions like maps and folds. Nevertheless, there are several libraries available, such as JQuery, that provide a richer framework to perform data operations.

8. Conclusion

8.1. Overview

This project was intended to study the state of the art of Geographic Visualization and to propose a novel approach able to solve some of the remaining challenges. A Domain Specific Language called Maps has been built over Diagrams and within a functional programming paradigm, using Haskell. In addition, the DSL has been tested providing a study on Human Migrations, using data from the World Bank and it has been compared to a successful InfoVisualization (includind GeoVisualization) toolkit called D3js.

8.2. Main Challenges

The main challenges of the project have actually been a materialization of the risks stated in Section 2.

In the first place, rather than an underestimation of the objectives, it was difficult to decide how to tackle the problem. Indeed, it was intended to improve in some senses the current solutions, without planning a project too wide for the time constraints of the MSc Project. The direction to follow was not very clear until a late stage of the project, which limited the scope of the solution.

In addition, there were indeed difficulties to learn Haskell at the desired speed. Although I was used to learn different programming languages, the fact of being a completely different paradigm and learning by myself has been a great challenge. It has been particularly difficult to design the structure of the toolkit, due to two different factors. The first one is that I am used to model in a great detail any software before starting the development. UML diagrams are a specially great way for me to visualize and think about the architecture and design patterns. All this techniques and intuition to model had been acquired within an Object Oriented perspective and it was definitely difficult to think otherwise. At the same time, the fluidity with the programming language comes when the design decisions are already taken.

8.3. Achievements

It has been achieved to build a DSL that can handle the process of GeoVisualization, enabling:

- Loading the spatial structure and modifying it (nesting and grouping regions, creating new levels, …).

- Loading the data, characterizing it topologically, transforming it and providing built in functions to map it to retinal variables.
- Generating visualizations composing customized subsections of the spatial structure and mapping to them any characteristic through a dataset. Composition is done by adding as well other levels such as arrows and glyphs over the regions, enabling the representation of a multivariate data.

Overall the solution is satisfactory, specially with regard to the proposed architecture. The functionality on every module is limited due to the time constraints of the project, but it can be easily extended and, alternatively, the structure enables the insertion of user defined functions to manipulate and customize the different steps.

8.4. Future Work

Future work could be done in different areas of the project.

- In the firs place, the possibility of incorporating interactive capabilities, which could increase the usefulness of the visualizations and provide enhanced ways of dealing with multivariate information.
- At the same time, it would be interesting to enable the simplification of regions depending on the current zoom. By reducing the number of points to draw, the performance will increase significantly
- Other work can be done in minor aspects such as allowing to convert the points from one projection to another or providing a wider range of built-in techniques to map the data into the geoVis.

List of References

[1] Alan M. MacEachren & Menno-Jan Kraak (2001) Research Challenges in Geovisualization, Cartography and Geographic Information Science, 28:1, 3-12,.

[2] Anselin, Luc. (2012) "From spacestat to cybergis twenty years of spatial data analysis software." International Regional Science Review 35.2: 131-157.

[3] Jung, J. K. (2014). Code clouds: Qualitative geovisualization of geotweets. The Canadian Geographer/Le Géographe canadien.

[4] World Migration in Figures © OECD-UNDESA October 2013

[5] Shukla, Nishant. Haskell Data Analysis Cookbook. N.p.: Packt, 2014.

[6] Denckla, B., & Mosterman, P. J. (2006). Block diagrams as a syntactic extension to haskell. In Proc. Workshop on Multi-Paradigm Modeling: Concepts and Tools at the 9th Int. Conf. Model-Driven Eng. Lang. Syst (pp. 67-79).

[7] Miller, Harvey J., and Jiawei Han, eds. *Geographic data mining and knowledge discovery*. CRC Press, 2009.

[8] Ware, Colin. Information visualization: perception for design. Elsevier, 2012.

[9] Liu, Shixia, et al. "A survey on information visualization: recent advances and challenges." *The Visual Computer* 30.12 (2014): 1373-1393.

[10] Shiravi, H., Shiravi, A., Ghorbani, A.A.: A survey of visualization systems for network security. IEEE Trans. Vis. Comput. Graph. **18**(8), 1313–1329 (2012)

[11] Chen, Chaomei. *Information visualization: Beyond the horizon*. Springer Science & Business Media, 2006.

[12] Isenberg, Petra, et al. "Collaborative visualization: definition, challenges, and research agenda." *Information Visualization* 10.4 (2011): 310-326. https://hal.inria.fr/inria-00638540/document

[13] Schneiderman, S. C. J. M. B. "Information Visualization: Using Vision to Think." (1999).

[14] Card, S. K., Mackinlay, J. D., Shneiderman, B.: Readings in Information Visualization: Using Vision to Think Morgan Kaufmann (1999)

[15] Keim, D. A., Ankerst, M.: Visual Data Mining and Exploration of Large Databases Tutorial, Conf. on Principles of Knowlege Discovery and Data Mining Freiburg, Germany (2001)

[16] Fekete, Jean-Daniel, et al. "The value of information visualization." *Information visualization*. Springer Berlin Heidelberg, 2008. 1-18.

[17] Lindholm, Stefan, et al. "Fused Multi-Volume DVR using Binary Space Partitioning." *Computer Graphics Forum*. Vol. 28. No. 3. Blackwell Publishing Ltd, 2009.

[18] Tufte, Edward R., and P. R. Graves-Morris. *The visual display of quantitative information*. Vol. 2. No. 9. Cheshire, CT: Graphics press, 1983.

[19] Wehrend, S., Lewis, C.: A problem-oriented classification of visualization techniques. In: IEEE Visualization, pp. 139–143 (1990)

[20] Keim, Daniel, and Hans-Peter Kriegel. "Visualization techniques for mining large databases: A comparison." *Knowledge and Data Engineering, IEEE Transactions on* 8.6 (1996): 923-938.

[21] W. J. Frawley, G. Piatetsky-Shapiro, C. J. Matheus: 'Knowledge Discovery in Databases: An Overview', in: Knowledge Discovery in Databases, AAAI Press, Menlo Park, CA, pp. 1-27, 1991.

[22] J. Bertin, Graphics and Graphic Information-Processing. Berlin: Walter de Gruyter, 1977/1981.

[23] Stevens, S. S. "On the Theory of Scales of Measurement." *SCIENCE* 103.2684 (1946).

[24] Bertin, Jacques. "Semiology of graphics: diagrams, networks, maps." (1983).

[25] Heer, J., Card, S.K., Landay, J.A.: Prefuse: a toolkit for interactive information visualization. In: CHI, pp. 421–430 (2005)

[26] Selassie, D., Heller, B., Heer, J.: Divided edge bundling for directional network data. IEEE Trans. Vis. Comput. Graph. 17(12), 2354–2363 (2011)

[27] Sedlmair, M., Frank, A., Munzner, T., Butz, A.: Relex: visualization for actively changing overlay network specifications. IEEE Trans. Vis. Comput. Graph. 18(12), 2729–2738 (2012)

[28] Von Landesberger, T., Kuijper, A., Schreck, T., Kohlhammer, J., van Wijk, J.J., Fekete, J.-D., Fellner, D.W.: Visual analysis of large graphs: State-of-the-art and future research challenges. Comput. Graph. Forum 30(6), 1719–1749 (2011)

[29] Jenny, B.: Adaptive composite map projections. IEEE Trans. Vis. Comput. Graph. 18(12), 2575–2582 (2012)

[30] Afzal, S., Maciejewski, R., Jang, Y., Elmqvist, N., Ebert, D.S.: Spatial text visualization using automatic typographic maps. IEEE Trans. Vis. Comput. Graph. 18(12), 2556–2564 (2012)

[31] Haunert, J.-H., Sering, L.: Drawing road networks with focus regions. IEEE Trans. Vis. Comput. Graph. 17(12), 2555–2562 (2011)

[32] Ferreira, N., Lins, L., Fink, D., Kelling, S., Wood, C., Freire, J., Silva, C.: Birdvis: visualizing and understanding bird populations. IEEE Trans. Vis. Comput. Graph. 17(12), 2374–2383 (2011)

[33] Scheepens, R.,Willems, N., van deWetering, H., Andrienko, G.L., Andrienko, N.V., van Wijk, J.J.: Composite density maps for multivariate trajectories. IEEE Trans. Vis. Comput. Graph. 17(12), 2518–2527 (2011)

[34] Tominski, C., Schumann, H., Andrienko, G.L., Andrienko, N.V.: Stacking-based visualization of trajectory attribute data. IEEE Trans. Vis. Comput. Graph. 18(12), 2565–2574 (2012)

[35] Scheepens, Roeland, et al. "Composite density maps for multivariate trajectories." *Visualization and Computer Graphics, IEEE Transactions on* 17.12 (2011): 2518-2527.

[36] A. Inselberg: 'The Plane with Parallel Coordinates, Special Issue on Computational Geometry', The Visual Computer, Vol. 1, pp. 69-97, 1985

[37] J. Huber: *'Projection Pursuit',* The Annals of Statistics, Vol. 13, No. 2, pp. 435-474, 1985.

[38] Joia, P., Coimbra, D., Cuminato, J.A., Paulovich, F.V., Nonato, L.G.: Local affine multidimensional projection. IEEE Trans. Vis. Comput. Graph. 17(12), 2563–2571 (2011)

[39] Turkay, C., Lundervold, A., Lundervold, A.J., Hauser, H.: Representative factor generation for the interactive visual analysis of high-dimensional data. IEEE Trans. Vis. Comput. Graph. 18(12), 2621–2630 (2012)

[40] Claessen, J.H.T., van Wijk, J.J.: Flexible linked axes for multivariate data visualization. IEEE Trans. Vis. Comput. Graph. 17(12), 2310–2316 (2011)

[41] Lee, J.H.,McDonnell, K.T., Zelenyuk, A., Imre, D.,Mueller, K.: A structure-based distance metric for high-dimensional space exploration with multi-dimensional scaling. IEEE Trans. Vis. Comput. Graph. (2013)

[42] Block, F., Horn, M.S., Phillips, B.C., Diamond, J., Evans, E.M., Shen, C.: The deeptree exhibit: visualizing the tree of life to facilitate informal learning. IEEE Trans. Vis. Comput. Graph. 18(12), 2789–2798 (2012)

[43] R. M. Pickett, G. G. Grinstein: *'Iconographic Displays for Visualizing Multidimensional Data',* Proc. IEEE Conf. on Systems, Man and Cybernetics, IEEE Press, Piscataway, NJ, pp. 514-519, 1988.

[44] J. Beddow: *'Shape Coding of Multidimensional Data on a Mircocomputer Display',* Proc. Visualization '90, San Francisco, CA, pp. 238-246, 1990.

[45] Keim, D.A., Mansmann, F., Schneidewind, J., Ziegler, H.L.: Chal- lenges in visual data analysis. In: IV, pp. 9–16 (2006)

[46] Chen, M., Jänicke, H.: An information-theoretic framework for visualization. IEEE Trans. Vis. Comput. Graph. **16**(6), 1206–1215 (2010)

[47] Fisher, D., Drucker, S.M., Fernandez, R., Ruble, S.: Visualizations everywhere: a multiplatform infrastructure for linked visu- alizations. IEEE Trans. Vis. Comput. Graph. **16**(6), 1157–1163 (2010)

[48] Wu,Y.,Liu,X.,Liu,S.,Ma,K.-L.:ViSizer:avisualizationresiz- ing framework. IEEE Trans. Vis. Comput. Graph. **19**(2), 278–290 (2013)

[49] Wu, Y., Yuan, G.-X., Ma, K.-L.: Visualizing flow of uncertainty through analytical processes. IEEE Trans. Vis. Comput. Graph. **18**(12), 2526–2535 (2012)

[50] Weaver, C.: Building highly-coordinated visualizations in impro- vise. In: INFOVIS, pp. 159–166 (2004)

[51] Fekete, J.-D.: The infovis toolkit. INFOVIS. 167–174 (2004)

[52] Heer, J., Card, S.K., Landay, J.A.: Prefuse: a toolkit for interactive information visualization. In: CHI, pp. 421–430 (2005)

[53] Bostock, M., Heer, J.: Protovis: a graphical toolkit for visualiza- tion. IEEE Trans. Vis. Comput. Graph. **15**(6), 1121–1128 (2009).

[54] Bostock, M., Ogievetsky, V., Heer, J.: D^3 data-driven documents. IEEE Trans. Vis. Comput. Graph. **17**(12), 2301–2309 (2011)

[55] Fisher, D., Drucker, S.M., Fernandez, R., Ruble, S.: Visualizations everywhere: a multiplatform infrastructure for linked visu- alizations. IEEE Trans. Vis. Comput. Graph. **16**(6), 1157–1163 (2010)

[56] Haskell cabal - https://www.haskell.org/cabal/
[57] Haddock - https://www.haskell.org/haddock/
[58] HLint - https://www.haskell.org/hLint/
[59] QuickCheck - https://www.haskell.org/QuickCheck/
[60] Lipovaca, Miran. Learn you a haskell for great good!: a beginner's guide. no starch press, 2011.
[61] Fowler, Martin. *Domain-specific languages*. Pearson Education, 2010.
[62] Duke, David J., et al. "Huge Data But Small Programs: Visualization Design via Multiple Embedded DSLs." *PADL*. 2009.
[63] Gibbons, Jeremy. "Functional programming for domain-specific languages."*Central European Functional Programming School*. Springer International Publishing, 2015. 1-28.
[64] Turing, A. M. (December 1937). "Computability and λ-Definability". *The Journal of Symbolic Logic* (4): 153–163.
[65] Hughes, John. "Why functional programming matters." *The computer journal*32.2 (1989): 98-107.
[66] Verimag, Jean-François Monin. "Kuntal DAS BARMAN."

Appendix A
External Materials

All the external materials used within this project are extracted from The World Bank.

Appendix B
Ethical Issues Addressed

No ethical issues affect to the present project.