The
Radial Projection Explorer

Group 4

Lukas Neuhold, Ridvan Aydin, Georg Regitnig

706.057 Information Visualisation SS 2020
Graz University of Technology

29 Jun 2020

Abstract

An explanation and introduction to the Radial Projection Explorer application developed by the authors of this paper. A tool developed with the purpose of visualizing data with radial projection techniques. The techniques included are RadViz, Star Coordinates, and Dust and Magnet. With this tool one can interact with all three of those techniques in a single application. This allows one to levy all the advantages they individually grant whilst negating disadvantages they have.

Contents

List of Figures

Chapter 1

Introduction

Data visualization is an important field in science and buisness. More and more data, especially multidimensional data, is gathered every day. With this multidimensional data one can not analyze it by just looking at a spread sheet of the thousands of records. To allow one to still competently argue about multidimensional data, visualizatuion techniques exist. One group of these techniques are Radial Projection Techniques.

With our tool, the Radial Projection Explorer, we want to combine three prominent radial projection techniques in one application. These are RadViz, Star Coordinates, and Dust and Magnet, as seen in figure 1.1. Combining these three in one allows one to exploit the individual strengths of the technique whilst simultaneously removing weaknesses.

In this report we will first introduce radial projection techniques in short detail. This is followed by an equally baseline introduction to the three techniques implemented. This should build an understanding of the techniques and how to effectively use them in a general, non application specific, sense. Afterwards an exploration of the development environment and the tools used is presented. Detailing how and why certain tools were used. The next chapter focuses on the application itself. How the application was laid out, and the purposes of all the separate parts of the application are explained. Finally the focus rests on future improvements, as well as how to convert the applications from an electron specific application to one that can be used as a web based application.

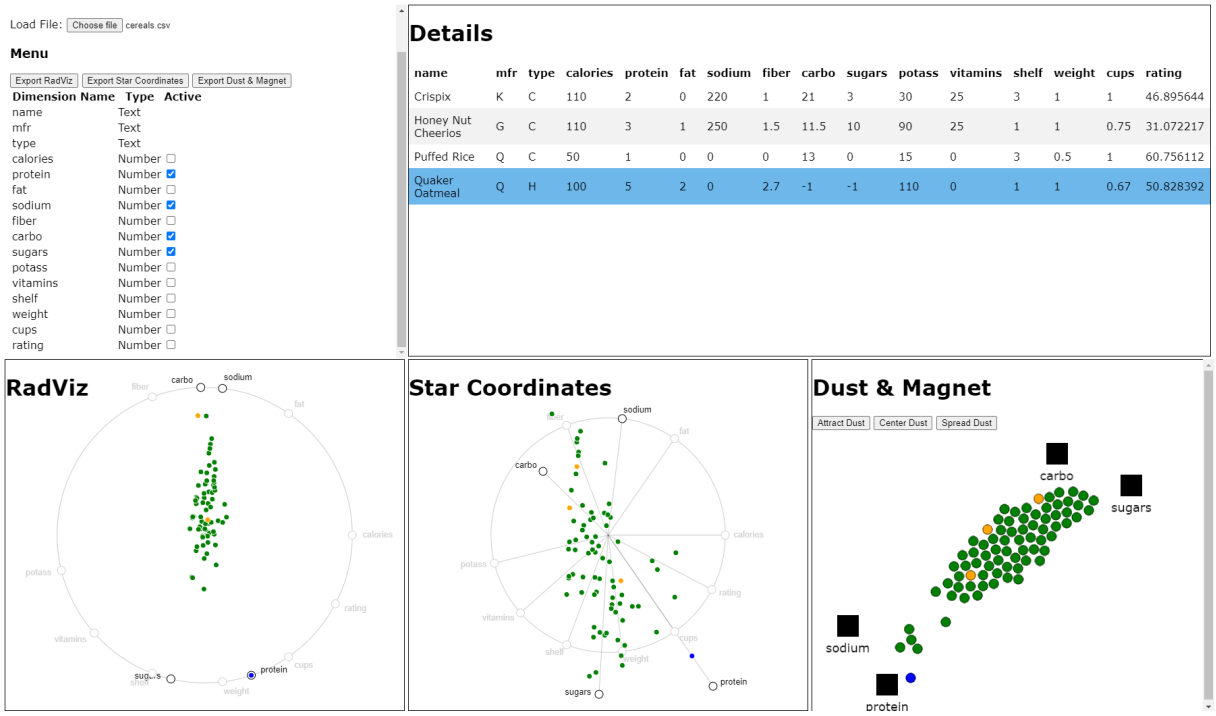

Figure 1.1: The Radial Projection Explorer in use, analysing the cereals data set. [Figure created by the authors of this paper.]

Chapter 2

Radial Projection Techniques

In this chapter a baseline understanding of all radial projection techniques used is established. At first for the general concept of these techniques, and afterwards for the implemented techniques themselves.

2.1 Basics

Radial Projection techniques are coarse multidimensional data visualizations. Coarse indicates that from this projection ambiguities are introduced about the actual values the data records have, as they are remapped to a 2D plane. However with the projection one can argue better about features of the data, be it clusters, outliers, or correlation.

The common feature all radial projection techniques have is that dimensions of the data are laid out radially in the 2D plane, as seen in figure 2.1. Data records are then mapped onto that plane using these dimension vectors. In the end a data record is represented by a single symbol in this 2D plane.

Different methods differ in how they approach this projection. Some techniques have a normalization step for the mapping, whereas others do not normalize the 2D coordinates of the data record. Other techniques improve upon the initial mapping with different optimization techniques. All these approaches try to improve upon different goals, for example the clustering of data.

2.2 Dust and Magnet

Dust and Magnet is a multidimensional data visualization tool that operates with the principal idea of ferrous dust and magnets in mind. Data features are magnets, and data records are represented as dust. This dust is attracted to the magnets over time, and from their movement and final placement one can reason about the data, as showcased in figure 2.2.

This very basic metaphor grounded in the physical world makes the interaction between data and dimensions very easy to understand and significantly lowers the barrier of entry for laymen in the field [Soo Yi et al. 2005]. Dust and Magnet can be seen as an extension to Star Coordinates. Magnets are the base vectors laid out freely by the user. The mapping of the data records differs by having an additional factor that describes the attraction of magnet and dust particles [Cheng and Mueller 2015]. An important part of the tool is animation over time. Magnets can be placed freely in the scene and after initial placement can be dragged around, to see how they influence the dust particles. During this dragging process the dust particle positions are continuously updated. From this movement one can argue about the data as well [Soo Yi et al. 2005].

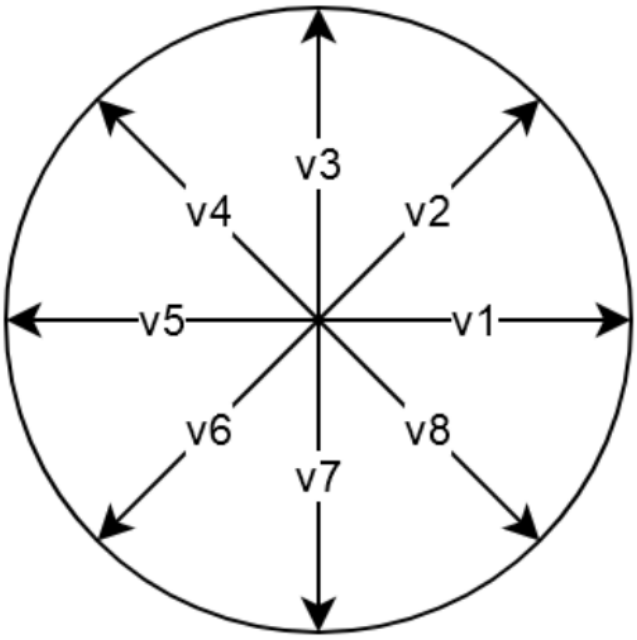

Figure 2.1: A basic radial layout of dimension base vectors. [Figure created by the authors of this paper.]

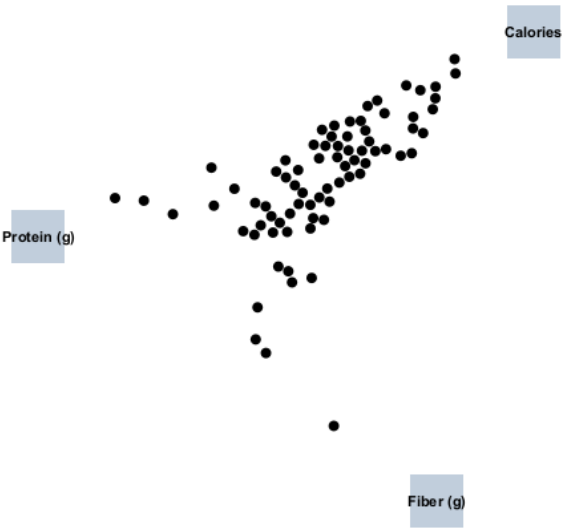

Figure 2.2: A basic scene in the dust and magnets tool. [Figure taken by the authors of this paper using the tool developed by Ji Soo Yi.]

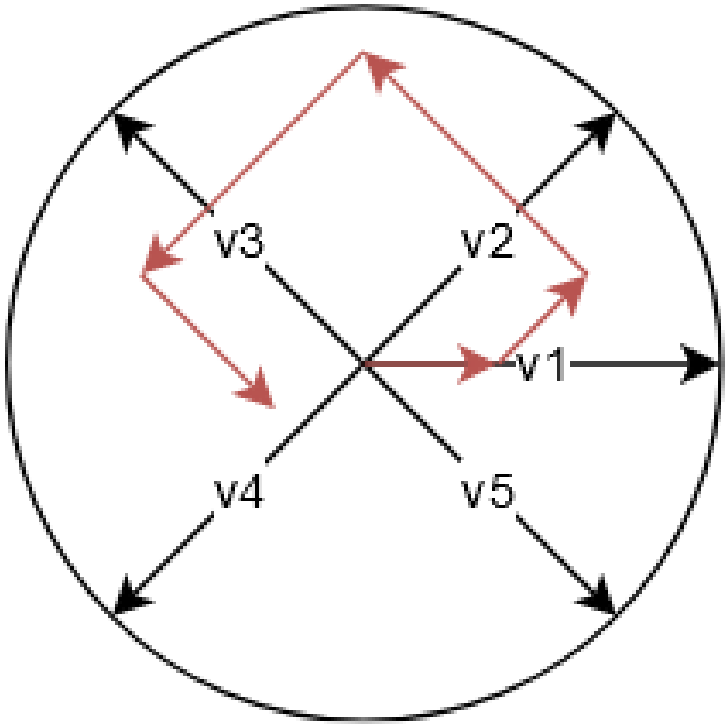

Figure 2.3: A data record is mapped by summing up its values in each dimension. [Figure created by the authors of this paper.]

2.3 Star Coordinates

Star coordinates were introduced by Kandogan [2001]. They extend the idea of two-dimensional scatter plots, where the data values of a certain point can easily be determined by checking the corresponding distances to the origin on the two coordinate axes. To allow a similar procedure using higher dimensions, the coordinate axes are not laid out orthogonal to each other, but in a circular shape. The amount of these coordinate axis is defined by the dimensions of the data. The origin of the axes defines the center of the circle. Before mapping data points, their dimensions have to be scaled to allow a meaningful visualization. The distance from the origin of data point, in regard to a certain dimension is linearly mapped between zero and the length of the corresponding axis. Values close to the minimum are closer to the origin, whereas values close to the maximum are mapped closer to the full extent of the dimension base vector. The final placement of a data record is determined by a vector sum of all scaled dimension base vectors, as seen in figure 2.3. Further analysis of the data is done via user interaction with the dimension bases vectors. This repositioning of the vectors updates the layout of all the mapped data records, creating new mappings that might reveal more information about the data.

2.4 RadViz

RadViz is a visualization technique which projects a n-dimensional dataset into a 2D plane. The goal is to be able to interpret the influence of each dimension as a balance between the influence of all dimensions. This is done by a physical spring model [Hoffman et al. 1997]. RadViz uses a set of n-dimensional vectors v_i $(i = 1, ..., m)$ where the number of these vectors define the dimensional anchor points of n springs. The coordinates of a n-dimensional datapoint are represented by a point p which then is connected to these anchor points. The attributes of the datapoint determine the location of p on the graph and also its stiffness. The location of p lies at the position where the sum of the spring forces equals 0. There is also to mention that the attributes of this datapoint must be non-negative but they are usually normalized that the range of each variable is between 0 and 1 [Rubio-Sánchez et al. 2015]. An example visualization can be seen in figure 2.4.

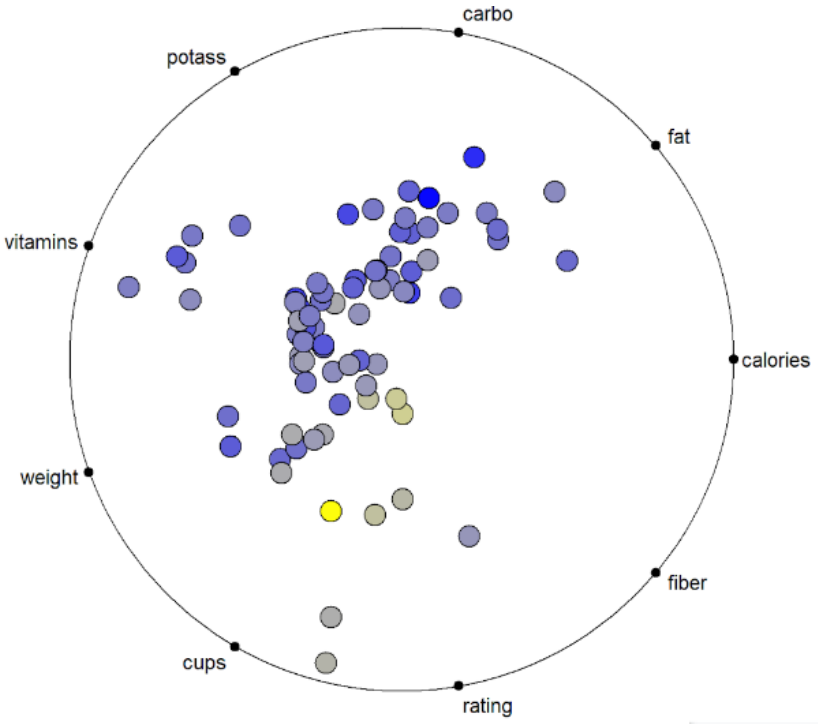

Figure 2.4: RadViz visualization of the cereals dataset. [Figure created by the authors of this paper using the tool RadViz-X.]

Chapter 3

Development Environment

3.1 Electron

Electron [1] is a framework for developing standalone applications using HTML, CSS and JavaScript. It provides several important features we need for our application like using multiple windows, communicating between those windows and exporting the application to an executable binary. The only drawback that comes with using the inter-window communication provided by the framework is the additional work required to run our application on a web server: It needs to be converted to a single-window application or use inter-tab communication when run in a web browser, but the multi-window approach has several significant advantages: Increased performance since each window is run in a separate process, easy reuse of the individual components and a separation in the source code.

3.2 TypeScript

The core of an Electron application is usually written in JavaScript, but we decided to use TypeScript [2] instead. TypeScript is a language built upon JavaScript that offers several additional features that make development easier. The most important addition we made use of is the static type checking added by TypeScript. This moves a lot of the debugging effort to compile time, making errors easier to find and to fix. Many of the external libraries we used provided type addons for TypeScript, so integrating them into the project was straightforward and we could also benefit from type checking when working with external tools.

3.3 Gulp

Developing always involves repetitive tasks, like compiling source files, removing old artifacts, copying files or exporting the results. Doing these things manually every time can get quite frustrating, therefore we used the Gulp [3] task runner to automate these steps. Gulp allows defining tasks using simple JavaScript files and running them sequentially or in parallel. We created Gulp tasks for pretty much every step required to build our project, like compiling the TypeScript files, starting the Electron application, exporting an executable and cleaning up the build directory. Another advantage Gulp provides compared to shell scripts that could perform the same tasks is its platform independence. Gulp tasks work on every platform, while shell scripts written for the Windows command line will not work on a Linux system.

[1] https://www.electronjs.org/

[2] https://www.typescriptlang.org/

[3] https://gulpjs.com/

3.4 Libraries

Our application uses several external libraries for loading and displaying the data records. The following sections will briefly explain the most important libraries we selected and why we use them.

3.4.1 Two.js

Two.js [4] is a JavaScript library that serves as a drawing API. It is designed towards drawing two-dimensional shapes, making it a perfect fit for our needs. The API allows rendering the shapes using either Canvas, WebGL or SVG without changing the drawing commands. Our application uses SVG as a render target since the figures are simple enough to maintain good performance and the graphics can be exported to a SVG file at all times without any additional conversion. The Star Coordinates and RadViz visualizations were implemented using Two.js.

3.4.2 PixiJS

Similar to Two.js, PixiJS [5] is another API for rendering two-dimensional graphics. It was used for implementing the Dust and Magnet visualization since it provides more advanced functionality for animation and user interaction. The most important difference compated to Two.js is the missing SVG render target. PixiJS can only use WebGL or Canvas, exporting the current image to SVG requires an additional conversion step.

3.4.3 D3.js

Before our application can visualize data records, they have to be loaded from somewhere. We support importing data stored in files using the CSV format. For parsing these files and deducting the data type of each column, we use functionalites provided by the D3 [6] library. Additionally, we make use of DOM-interaction features provided by D3 when creating the data table in the Details window.

[4] https://two.js.org

[5] https://www.pixijs.com/

[6] https://d3js.org/

Chapter 4

The Application

In this chapter a closer look at all parts of the final application is taken. It starts off with a short discussion of the layout, which is followed by an indepth look at all the different windows. The explanations focus on the features of the implementations, why they were done a certain way, and if applicable, how we differ from other implementations.

4.1 The Views

The basic premise of our applications was to unify three different radial projection techniques. The simplest approach, as well as the one approach that allows one to most easily compare them, is to have a different window for all techniques. However, doing it like that also means one needs to have some central control or overview window. Finally, an additional window was needed to display the detailed data values of any data records. As mentioned previously radial projections are coarse visualization techniques. From the final placement of the symbol representing a data record it is hard or impossible to read the actual data values. This brings the total count of differing views we have up to five.

Our final layout can be seen in figure 4.1. In the top half of the layout, the overview and detail windows are located. They are grouped together as they don't directly visualize data, and therefore can be seen as one group of views. On the bottom we have all the techniques right next to each other. With this layout one can easily compare the final data point positions between techniques, with a simple horizontal sweep of the window. No other window breaks up this comparison by being sandwiched in between techniques.

As our application is Electron [1] based, all our different views are their own processes. Electron works by having one main process, and every window being its own render process. One major issue that needs solving with this setup is window synchronization. All our techniques are quite independent from one another, however there are certain key areas where synchronization is needed.

The main causes for synchronization are twofold. Data dimension enabling and disabling, and data record selection. When a dimension is removed or added from consideration all the radial projection windows need to be updated. Secondly when a data point is selected in any window, all other technique windows need to be informed of that, to be able to highlight that specific point. Furthermore the detail window needs to be informed as well to be able to show the exact data values.

Electron already provides a powerful inter-process communication tool. With this one can name channels via a simple string, and send arbitrary data through them. Any window can listen to this channel and react or ignore any messages sent. Utilizing this, it was quite simple afterwards to keep all our windows synchronized over any action taken in any window.

[1] https://www.electronjs.org/

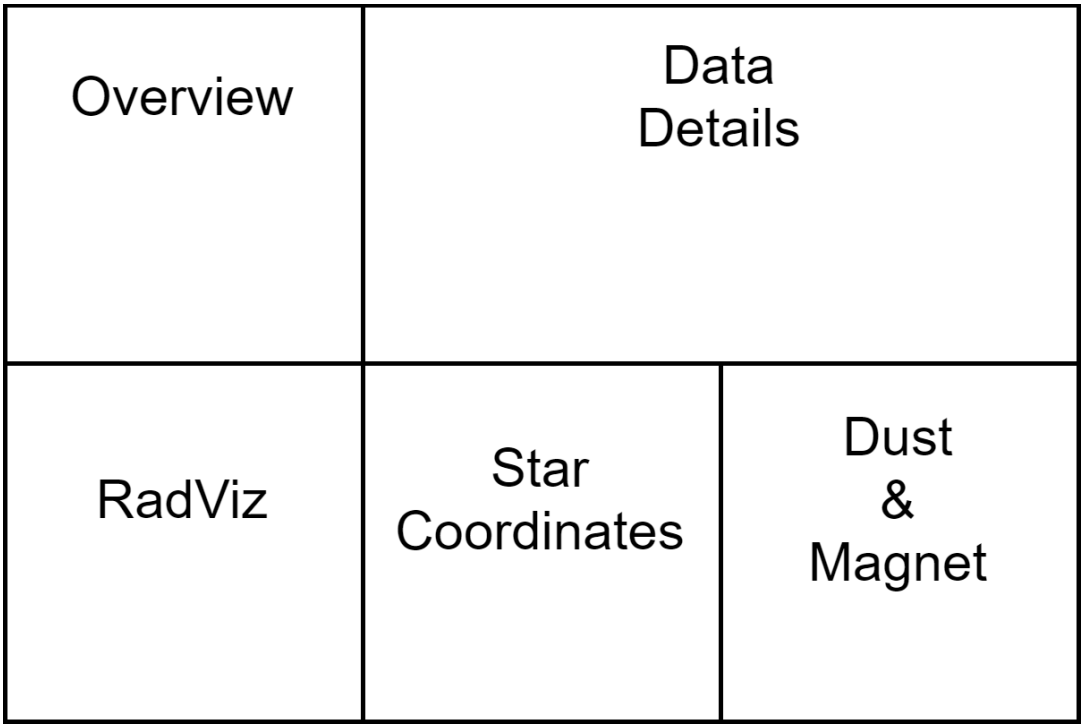

Figure 4.1: A visualization of the layout the Radial Projection Explorer has. The proportions are accurate to the proportions in the actual application. [Figure created by the authors of this paper.]

4.2 Overview Window

The overview window is the general control panel for the application. In this window, you load your CSV for your visualization. After loading the data, a dimension control panel appears, where one can activate and deactivate dimensions. Inactive dimensions do not influence the visualization in any view anymore. As mentioned previously all views are automatically synchronized with this window. The state of the dimensions is controlled by simple checkboxes, as seen in figure 4.2.

Another important feature in this window is the ability to export views as SVG. Exporting is achieved by pressing the correspondingly labeled button. Every technique is exported independently and not as one combined SVG, because not all generated layouts may be informative during data exploration. With independent exports the user can decide which views have valuable layouts, and does not need to manually edit the resulting SVG afterwards.

4.3 Detail Window

In the detail window you see the exact values of all selected data records displayed in a simple table. The application allows for multiple data points to be selected at the same time. This allows you to easily compare all the selected data records by their exact values.

However allowing for multiple selected points has a negative side effect. All selected points have a different color than unselected ones in the projection views. This selected color does not vary between different selection points. In practice, when a single point is selected one can now clearly see it in all the projection views, compare the location whilst simultaneously knowing the exact data values. After adding multiple selections however this clear distinction is lost. It becomes difficult to remember the correlating points in the view as well as what table entry it is.

To alleviate this problem hover based color changing was implemented, seen in figure 4.3. Hovering over an element in the detail window table will highlight this specific element, in all visualizations in a

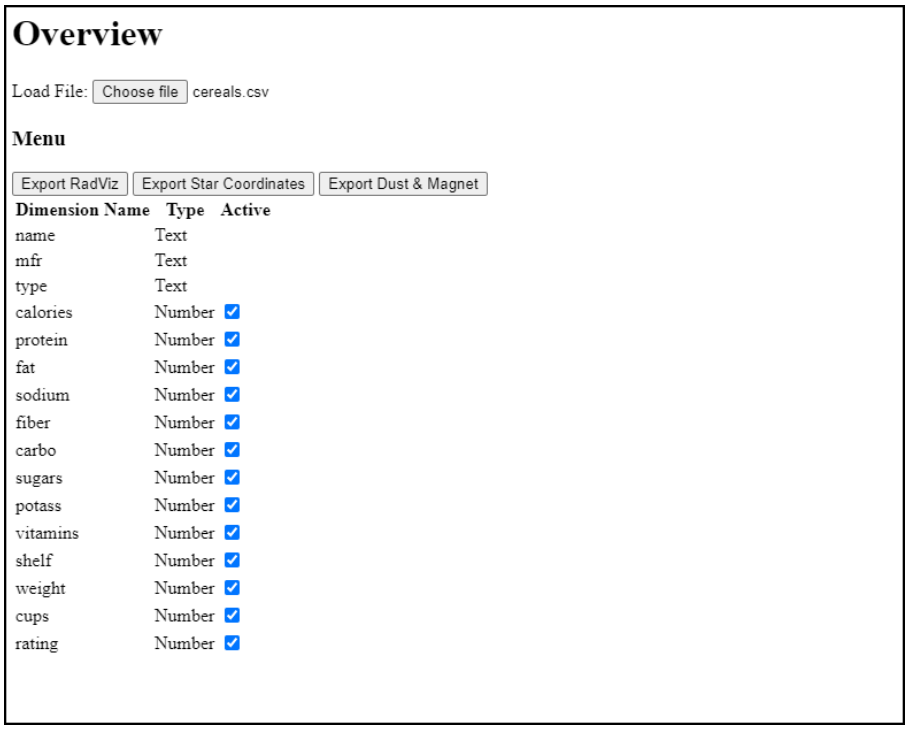

Figure 4.2: An example overview window with the cereals data set loaded. [Figure created by the authors of this paper.]

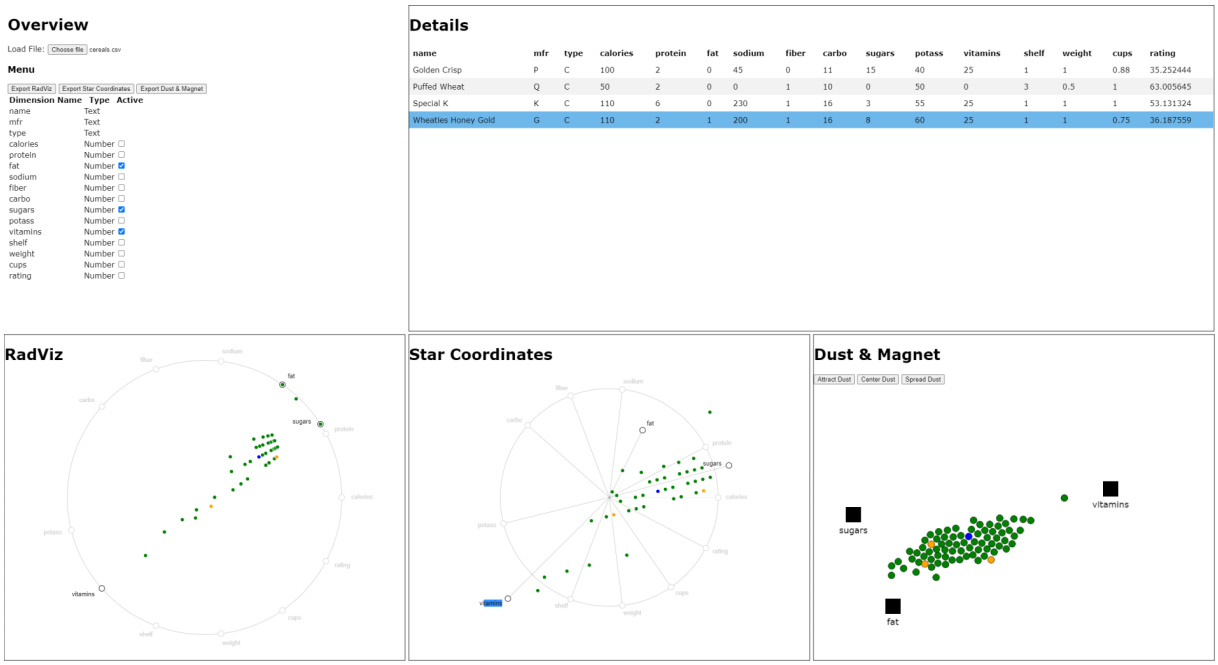

Figure 4.3: Multiple points selected and shown in the detail window, as well as colored in the projections. The user hovers over one entry in the detail table, and therefore marks the point in every view with a different color, highlighting it.[Figure created by the authors of this paper.]

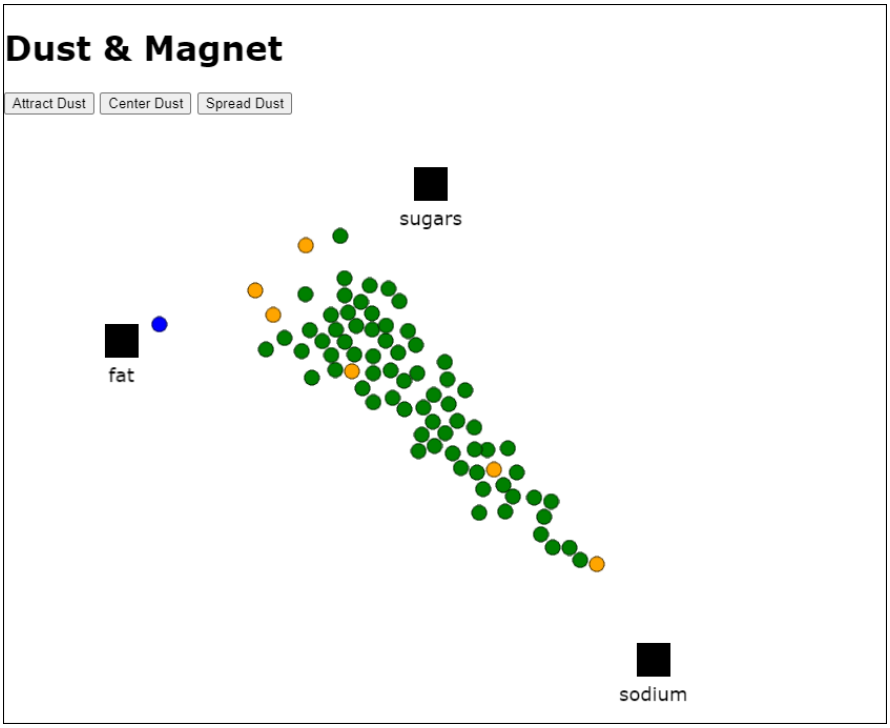

Figure 4.4: A basic Dust and Magnet scene created with the Radial Projection Exlporer. [Figure created by the authors of this paper.]

different color to both selected and unselected points. Using this one can once again focus on a single data record, but still have the possibility to highlight other records as well. This creates a sort of tiered focus system for the user. With multi selection a user can focus on a region of data points, and with hover based coloring the user can focus more on very specific data record.

4.4 Dust and Magnet Window

Our implementation of Dust and Magnet is based upon the implementation of Soo Yi et al. [2005]. We implemented the most useful features from their implementation in our application. Some features in the original are not present in our implementation. A couple of those features are left out, because after extensive testing, for a previous use case study the authors did, it was concluded that they were not that important. Others, like dust particle coloring where deliberately left out because it needs more synchronization between all the different views and can not be implemented the same way. Compared to the original implementation we have one additional feature which will be elaborated when talking about the dust particles. A basic visualization using our tool can be seen in figure 4.4.

4.4.1 Magnets

Every Magnet can be moved by left clicking with the mouse and dragging the sprite. Moving a magnet will automatically animated the dust particles attraction to this magnet. With a control key modified left or right click one can change the magnitude of that magnet. A right click decreases the magnitude, whilst left clicking increases it. The magnitude is an overall description of how influential the magnet is in the simulation. Lowering means it has a lower influence attracting dust less, the opposite happens upon increasing it. The influence of a magnet is represented by the size of its respective sprite, as seen in figure 4.5. Smaller size indicates lowered magnet magnitude. The amount of magnets is controlled by the overview windows dimension control panel. Updates in the control panel are automatically reflected in Dust and Magnet, enabling or disabling a magnet in the scene.

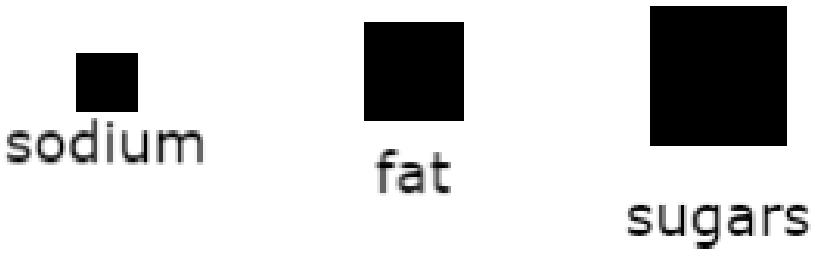

Figure 4.5: Differently influential magnets, and how they are visualized in the scene. [Figure created by the authors of this paper.]

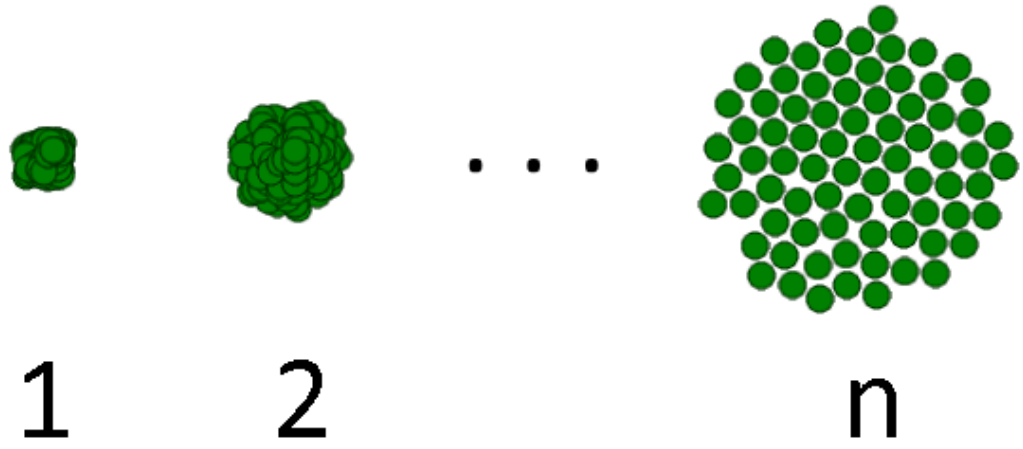

Figure 4.6: Spreading Dust at different stages. [Figure created by the authors of this paper.]

4.4.2 Dust

Similar to the other two visualizations any dust particle can be selected, highlighting it and letting one see the detailed values of it in the detail window. There are three more interactions with regard to dust particles. These are:

- Spread Dust
- Center Dust
- Attract Dust

The first ensures that dust particles do not overlap. As dust has no interaction with other particles, it is natural that some will end up at similar positions over time. To counter this occlusion spread dust moves the particles, in small steps, to positions that are non overlapping, as seen in figure 4.6. This can

be achieved with the keyboard key S or holding down the button Spread Dust at the top of the Dust and Magnet window. This feature is implemented via a simple adjacency matrix for all dust particles. For every iteration the distances between the particles are calculated. Based upon that the closest neighbors are moved into somewhat randomized opposite directions in case they overlap. After every particle movement the adjacency matrix has to be recalculated.

The second, Center Dust, allows one to reset the simulation to the beginning. Centering dust can be achieved by pressing the key C or clicking the button Center Dust at the top of the window. This centers the dust particles on the screen. This feature is most useful if you want to restart your simulation with a clean slate.

Lastly, Attract Dust, a feature to allow for static magnet setups. Without this feature particles would only move when a magnet is moved. However if one wants to see how a specific magnet setup influences the particles, one needs to use the Attract Dust feature. Pressing A or holding pressed the button at the top of the screen labeled Attract Dust, one can animate all particles with regard to the current magnet setup.

Compared to the original implementation, dust can not hide behind magnets in our application. When dust particles can hide behind magnets you lose intractability with them, as well as visibility. In the use case study previously done by the authors, this provided a constant sore spot. Because of this the original implementation was only used with a static magnet setup and the attract dust feature, caused by the constant worry to misinterpret the final result because one was unaware of a hidden dust particle. A focus of our implementation was to counter this.

Two different attempts were made, one with a more sophisticated placement correction and the second with a more simplistic approach. The first approach was to detect when a particle was hidden behind a magnet after any move and find the closest point on the magnet's bounds and place it right outside. This was done with simple line-point distance calculations and moving the particle along the direction from the particle to the closest point on the border of the magnet. However problems started occurring. With a static magnet setup the problem was the fact that if a particle moved behind a magnet in an update step and the distance moved was into the opposite half of the magnet. With this strong attraction the particle would suddenly start to appear on the opposite sides of the magnet constantly, leading to a very disorienting teleporting effect. However more problematic was the dragging action. In case of the user deliberately dragging the magnet over a dust particle the teleporting of the particles would make even less sense. However one can not restrict the dragging movement, as this would make for very unnatural maneuvers the user would need to complete. Moving the magnet from one side of the screen to the other without being able to cross a single dust particle is next to impossible.

The second approach was simpler. If the movement would hide this particle, just do not move. This has the effect that when a particle is right next to a magnet, it just stops moving. With this there is no random teleportation with neither the static magnet setup and the dragging interaction. One thing to consider is the fact what should happen if the dust particle is behind a magnet and in the next move would still remain behind a magnet. In our implementation the particle just does not move. The only time a situation like this can occur, is when a user deliberately placed a magnet on top of a particle. Which is not something we can forbid as a magnet placement. However if this occures, the responsibility lies with the user. They either had a good reason to do it, or are deliberately trying to mess with the the projection. As teleporting dust particles are unwatned, to keep the simulation consistent, no movement is the best option in this case. It is the users responsibility to not drag magnets over dust particles and let them rest in this position.

4.5 Star Coordinates

Our implementation of star coordinates is based upon the base idea as introduced by Kandogan [2001]. Every anchor point can be moved to a different position freely using drag & drop. Moving an anchor

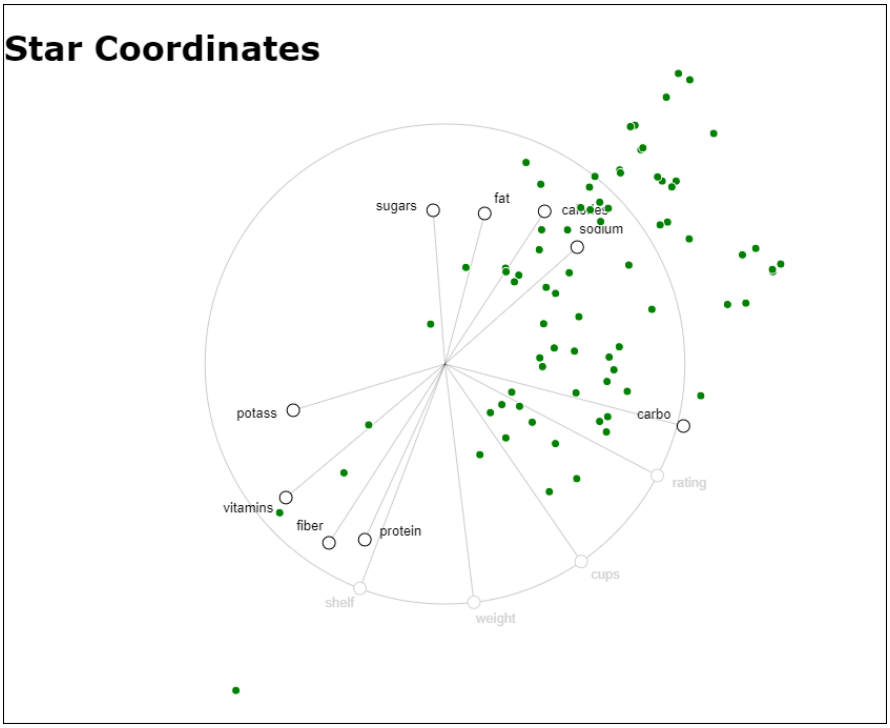

Figure 4.7: A star coordinates visualization of the cereals data set. The dimensions are split along opposite directions depending on if they represent healthy attributes or not. The influence of all dimensions has been modified by moving them different amounts closer to the center. [Figure created by the authors of this paper.]

away from the center will increase the impact the respective dimension has on the data point, moving it closer to the center has the opposite effect. If a dimension is disabled from the Overview menu, it will appear in a gray color and will not affect the mapping at all, regardless of the anchor position. By rotating the axes, you can affect the direction that will describe the value of the dimension. Figure 4.7 shows all these things combined in one potential layout to split the data into healthy and unhealthy cereals. In the same way as both RadViz and Dust and Magnet allow one to select data points, the same interaction can be taken in the Star Coordinates view.

4.6 RadViz

The implementation of RadViz in this application is based upon the base idea introduced by Hoffman et al. [1997]. Similar to Star Coordinates, the anchors can be moved via drag & drop, but in the case of RadViz, the anchors can only be positioned on the radial. The mapping for RadViz is a bit different than in Star Coordinates. The projection of the data points follows a physical spring model. The normalized values between 0 and 1 for each dimension define how strong the datapoint is pushed towards the anchor of this dimension. This kind of normalization leads the data points only to be mapped inside the radial. Same as with Star Coordinates, deactivated dimensions will not affect the mapping of the data points. Figure 4.8 showcases a visualization of the cereals data set. Every data point in the scene can be selected and will be highlighted in a different color. This will furthermore make the datapoint appear in the details window.

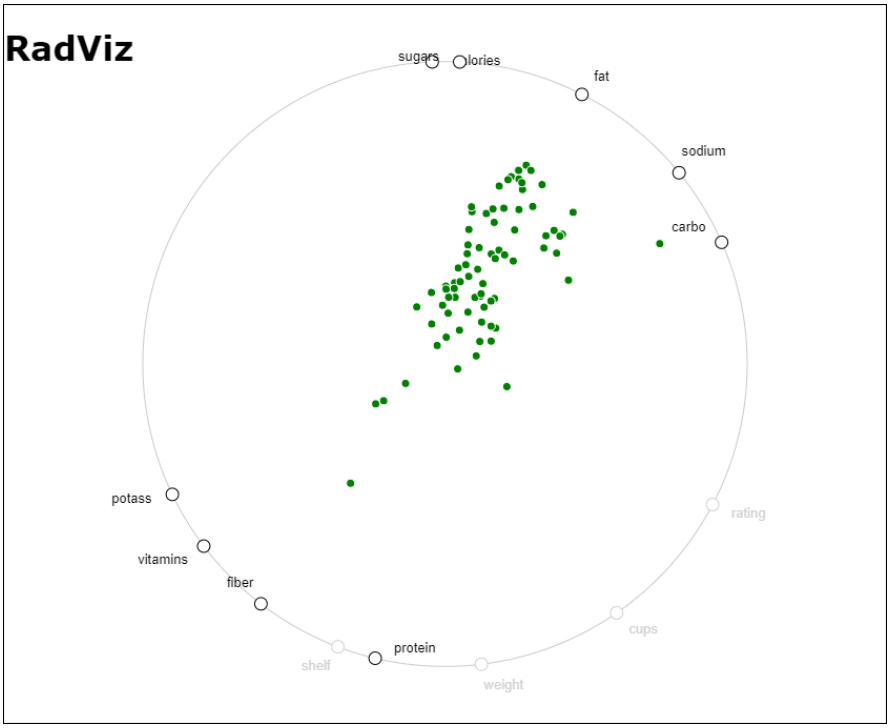

Figure 4.8: A RadViz visualization of the cereals data set. The dimensions are split along opposite directions depending on if they represent healthy attributes or not. [Figure created by the authors of this paper.]

Chapter 5

Future Work

5.1 Removing Electron If Unwanted

In case of Electron being unwanted as a basis for the applications a few steps have to be taken to be able to convert the project.

First of, the individual views are their own HTML pages being synchronized via the electron inter-process communication module. Any reference to this module has to be removed. Starting with the individual HTML pages. These need to be instead of full pages, only containers inside a single page. After doing that, one needs to use CSS to recreate the layout. When this is completed one can move on to the code base.

The basic structure of the inter-process communication for all windows in our application, is the Main process/class is being used as a relay station. Any communication flows through the main class. To remove the module any mention of the inter-process main *on* function and *webcontents.send* in the main process need to be replaced with static functions of the main class, and a send function, all other views have, that takes the same arguments of channel name and data. This send function could be an interface function, or a base class all other classes in the project implement. Doing it like that makes it easy for the individual classes to change their inter-process renderer *on* functions to ones that do not use the electron module.

The main class keeps references to all other objects and uses this to inform the others of a message. If the main class is kept static, other objects can simply send it data via specific static function calls. Of course there is the further possibility that all view objects are singletons and the main class does not need to keep references to them, but uses this static instance.

One other necessary change would be the Gulp task bundling. In its current setup every window has its own javascript file that it loads. With a single HTML page all the functionality should be bundled into a single javascript file instead of five different ones.

In conclusion there are multiple possible ways to remove the reliance on the inter-process communication. However if the structure of the main class being a relay class is kept, none of them are very work intensive as the individual views were kept very independent by us this way.

5.2 Coloring

Currently there are three different colors a data record can have in any visualization. Green as a default, orange if selected and blue if hover highlighted. An extension would be to allow coloring based on dimension values. This could be done by defining gradients for numeric dimensions, or color palettes for dimensions that define categories. Interactions need to be provided in the overview window, where the user can create these gradients, or color palettes. The views should most likely have synchronized pallets

to keep the comparison more meaningful. However one can argue against this synchronization need. The best approach is a question for testing.

5.3 Data Preprocessing

As mentioned in the previous section some dimensions could represent categories of values. In the current application the only delineation between data is, is it text or is it a number. A future iteration of the app could improve upon this classification. An approach could be whilst loading the data automatically try and figure out what are categories, what are identifiers and what are number valued dimensions. However the user should be able to have influence over this pre-processing and change it completely if so desired.

5.4 Dust and Magnet - Magnets

In the original implementation of Dust and Magnets, Magnets could also be repellent. However after not using it at all during our use case study, we decided against making it a priority of our implementation. This does not however mean that it is a useless feature, as there are a multitude of situations we might never account for in our initial design decisions. Therefore integrating this feature into the Dust and Magnet implementation should be a future extension of the implementation.

5.5 Application Window Size

As the current implementation stands, on starting of the app the window size is quite large. In case of the user resizing the window, this resized window dimensions should be stored and on reopening the application at a later time point should revert to this size.

5.6 Application Layout

The layout of the application is static, as it was defined by the authors of the project. Overall the split into non projection windows up top, and projection windows at the bottom is quite satisfactory and has a clear division of labor, and screen space. However users might want to rearrange the individual views to their liking. Supporting this would make the application much more user friendly and allows the user to find the setup with which they can work the most efficient.

5.7 Radial Projection Technique Optimizations

For both RadViz and Star Coordinates, multiple optimizations to the base implementation exists. As an additional add on to the base implementation one could allow the user to activate these different mapping techniques. One such optimization for Star Coordinates is orthographic projection, as described by Lehmann and Theisel [2013]. This aims to reduce the distortions of the cluster shapes from high dimensional space to the 2D plane. An applicable RadViz optimization might be FreeViz, as described by Demsar et al. [2007]. FreeViz aims to separate the distances of different classes in the data with a gradient optimization algorithm.

Chapter 6

Conclusion

The prospect of having multiple different approaches to visualize multidimensional data is very exciting. The fact that these techniques offer different advantages and disadvantages whilst analyzing data allows them to work together very well without too much overlap. As a user not having to install or switch between multiple applications to analyze one data set is desirable and eases the difficulty of work.

The three chosen techniques work well together. In a previous survey of the authors it was concluded that Star Coordinates and RadViz had an easier time finding correlation in data, whilst this was harder to accomplish with Dust and Magnet. On the other hand, clustering of data is extremely simple to achieve with Dust and Magnet, whereas RadViz had some minor inconveniences associated with these tasks.

One prospect for the future development of the project, would be implementing more radial projection techniques and letting the user mix and match. Dust and Magnet sometimes feels like the odd one out of the three techniques chosen, just by the nature of its interactivity compared to the others. This is exaggerated by the fact that Dust and Magnet works best when analyzing smaller subsets of the data dimensions. It is not impossible to do analysis with all dimensions of the data affecting the simulation. However after extensive testing on the authors part it is a lot simpler to understand what is going on in the simulation with fewer active dimensions. The opposite is true for RadViz and Star Coordinates. The data starts to overlap more and more the fewer dimensions are active. One can draw conclusions from the overlapping data points, however one can achieve better readability without losing the intractability of some data points with smarter anchor layouts instead of complete deactivation. In general allowing the user to choose between more radial projection techniques seems like a logical step forward.This enables them to use the techniques they are most comfortable analyzing with, as well as comparing the layouts of these easily.

The development environment chosen allows for fast iteration and a huge set of publicly and freely available modules to boost development speed. The barrier of entry is low making it easy to integrate new developers into the team.

In conclusion, the Radial Projection Explorer allows one to easily analyze and compare various radial projection techniques. The project as it stands now already offers effective tools that allow for intuitive and powerful data analysis. However there still are a lot of possibilities open to improve further on the current state of the project.

Bibliography

Cheng, Shenghui and Klaus Mueller [2015]. *Improving the fidelity of contextual data layouts using a generalized barycentric coordinates framework*. 2015 IEEE Pacific Visualization Symposium (PacificVis) (Hangzhou, China). IEEE, Apr 2015, pages 295–302. doi:10.1109/PACIFICVIS.2015.7156390 (cited on page 3).

Demsar, Janez, Gregor Leban, and Blaz Zupan [2007]. *FreeViz—An intelligent multivariate visualization approach to explorative analysis of biomedical data*. Journal of biomedical informatics 40.6 (Dec 2007), pages 661–71. doi:10.1109/TVCG.2015.2467324 (cited on page 18).

Hoffman, Patrick, Georges Grinstein, Kenneth Marx, Ivo Grosse, and Eugene Stanley [1997]. *DNA Visual and Analytic Data Mining*. Proceedings. Visualization'97 (Cat. No. 97CB36155) (Phoenix, AZ, USA). IEEE, Oct 1997, pages 295–302. doi:10.1109/VISUAL.1997.663916. https://cs.uml.edu/~phoffman/dna1/ (cited on pages 5, 15).

Kandogan, Eser [2001]. *Visualizing Multi-Dimensional Clusters, Trends, and Outliers Using Star Coordinates*. Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '01. ACM. San Francisco, California: Association for Computing Machinery, Aug 2001, pages 107–116. ISBN 158113391X. doi:10.1145/502512.502530 (cited on pages 5, 14).

Lehmann, D. J. and H. Theisel [2013]. *Orthographic Star Coordinates*. IEEE Transactions on Visualization and Computer Graphics 19.12 (2013), pages 2615–2624 (cited on page 18).

Rubio-Sánchez, Manuel, Laura Raya, Francisco Diaz, and Alberto Sanchez [2015]. *A comparative study between RadViz and Star Coordinates*. IEEE transactions on visualization and computer graphics 22.1 (Aug 2015), pages 619–628. doi:10.1109/TVCG.2015.2467324 (cited on page 5).

Soo Yi, Ji, Rachel Melton, John Stasko, and Julie A Jacko [2005]. *Dust & magnet: multivariate information visualization using a magnet metaphor*. Information visualization 4.4 (Dec 2005), pages 239–256. doi:10.1057/palgrave.ivs.9500099 (cited on pages 3, 12).