

MASARYK UNIVERSITY  
FACULTY OF INFORMATICS



# **Spectramosaic: an exploratory tool for the interactive visual analysis of magnetic resonance spectroscopy data**

MASTER'S THESIS

**Bc. Jakub Vašíček**

Brno, Spring 2020



MASARYK UNIVERSITY  
FACULTY OF INFORMATICS



# **Spectramosaic: an exploratory tool for the interactive visual analysis of magnetic resonance spectroscopy data**

MASTER'S THESIS

**Bc. Jakub Vašíček**

Brno, Spring 2020



*This is where a copy of the official signed thesis assignment and a copy of the Statement of an Author is located in the printed version of the document.*



## **Declaration**

Hereby I declare that this paper is my original authorial work, which I have worked out on my own. All sources, references, and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Bc. Jakub Vašíček

**Advisor:** doc. RNDr. Barbora Kozlíková, Ph.D.





## Acknowledgements

I would like to thank Laura Garrison, Stefan Bruckner, Helwig Hauser and the whole VisGroup and MMIV for the opportunity to work on such an interesting topic, and for the supportive and encouraging environment they provided for me during my stay in Bergen.

I would also like to thank my supervisor Barbora Kozlíková for connecting me to VisGroup, helping me organize my stay Bergen and advising me through the writing process of this thesis.

Most importantly, I want to express gratitude to my family who support me by all their means through my long years of study. Without their support, it would not have been possible for me to stay in Norway and work on this project, and the whole study would be much harder to manage.

## Abstract

Magnetic resonance spectroscopy is a promising method that allows a noninvasive analysis of the chemical environment of the brain. However, a visualization tool that would support an interactive visual analysis of such data, including comparison of different samples and capturing variance in ratios of tissue metabolites, was not available so far. Therefore, this problem was addressed by the research team led by Laura Garrison at the University of Bergen and Mohn Medical Imaging and Visualization Centre (MMIV), which focuses on developing a tool allowing intuitive and interactive exploration of tissue metabolite concentration ratios in spectroscopy clinical and research studies. This thesis has been conducted as a part of this research initiative.

Based on the already created prototypes of visual design and user interface, the goal of this thesis is to transfer these sketches into a working tool. This includes the tasks of selecting suitable technologies, proposing subtle design changes and procedures of algorithmization, and implementing the application. The result was used to validate design choices and propose changes and additional features for future development of this project. Feedback on this application was given by three domain experts from the hospital research environment, who confirmed the efficacy of the system. The project was presented at the 9th EG Workshop on Visual Computing for Biology and Medicine in September 2019 in Brno [1].

## Keywords

visual analysis, interactive visualization, medical visualization, data exploration, magnetic resonance spectroscopy, application design and implementation, p5.js



# Contents

<b>Introduction</b>	<b>1</b>
<b>1 Project Context, Task Specification</b>	<b>3</b>
1.1 <i>Magnetic Resonance Spectroscopy (MRS)</i> . . . . .	3
1.1.1 NMR Spectroscopy Basics . . . . .	3
1.1.2 MRS Terminology . . . . .	4
1.1.3 Data Processing Steps . . . . .	6
1.2 <i>Data Analysis</i> . . . . .	8
1.2.1 Visualization tasks . . . . .	8
1.2.2 Requirements . . . . .	9
<b>2 Related Work</b>	<b>11</b>
2.1 <i>Standard Tools to View MRS Data</i> . . . . .	11
2.1.1 Standard Processing of MR Spectra . . . . .	11
2.1.2 Advanced MRS Data Analysis . . . . .	12
2.2 <i>Related Visualization Research</i> . . . . .	13
2.2.1 Visualization Research Related to Spectroscopy	13
2.2.2 Multidimensional and Heterogeneous Data Vi-	
sualization . . . . .	14
2.2.3 Tabular and Unit Visualization . . . . .	15
<b>3 Interface and Visualization Design</b>	<b>17</b>
3.1 <i>Interface Components</i> . . . . .	18
3.1.1 Left Panel: Data Overview . . . . .	19
3.1.2 Right Panel: Metabolite Ratio Map . . . . .	21
3.2 <i>Interaction</i> . . . . .	23
3.2.1 Interactions in the Left Overview . . . . .	24
3.2.2 Interactions in the Matrix View . . . . .	25
3.2.3 Linking the Two Views . . . . .	25
<b>4 System Design</b>	<b>27</b>
4.1 <i>Benefits of a Web-based Application</i> . . . . .	27
4.2 <i>Single-page Application</i> . . . . .	28
4.3 <i>Client Application Components</i> . . . . .	30
<b>5 Computations and Data Processes</b>	<b>33</b>

5.1	<i>Input Format and Loading</i>	33
5.1.1	Required Format and Structure of Input	33
5.1.2	Loading Process	34
5.2	<i>Data Structures</i>	36
5.3	<i>Integral and Ratio Computation</i>	39
5.3.1	Peak Integral Ratios	40
5.3.2	Subset-aggregated and Individual Ratios	41
<b>6</b>	<b>Implementation</b>	<b>45</b>
6.1	<i>Source Files</i>	45
6.2	<i>Layout of the Page</i>	46
6.3	<i>Implementation of System Components</i>	47
6.3.1	Data Input	48
6.3.2	Main Panels as p5.js Canvas	50
6.4	<i>Interaction</i>	51
6.4.1	Selecting a Voxel for the Analysis	54
6.4.2	Adjusting the Matrix View	55
6.4.3	Highlighting System	57
<b>7</b>	<b>Conclusion and Future Work</b>	<b>61</b>
7.1	<i>Summary</i>	61
7.2	<i>Evaluation and Future Directions</i>	62
7.3	<i>Conclusion</i>	64
	<b>Bibliography</b>	<b>65</b>
<b>A</b>	<b>Low-fidelity Prototype: Design Sheets</b>	<b>73</b>

## List of Figures

- 1.1 A typical MRS spectrum visualized in a Siemens MR workstation 5
- 1.2 A spectral curve with a baseline and model fit 7
- 2.1 jMRUI Clinical Viewer plug-in: metabolite ratio map 13
- 3.1 Sketch of the interface layout 18
- 3.2 User interface of the final application 20
- 3.3 16 encoding scenarios of nested information in the matrix view 22
- 3.4 Spectral voxel selection workflow 24
- 3.5 Highlighting system when linking the two panels 26
- 4.1 Comparison of traditional and single-page web applications 29
- 4.2 The roles of system components 31
- 5.1 Directory structure of an exemplary dataset 34
- 5.2 Hierarchy of the main data structure 38
- 5.3 Class diagram of the matrix view data structure 38
- 5.4 Class diagram of the data structure for nested details 39
- 5.5 3D plot of the transformed symmetric ratio function 43
- 5.6 Illustration of computing all types of ratios 44
- 6.1 Diagram of the system components and source files 46
- 6.2 Gridster: page divided into five main elements 47
- 6.3 UML activity diagram for updating the left panel 52
- 6.4 UML activity diagram for updating the right panel 53
- 6.5 Layout of the left panel 54
- 6.6 Layout of the right panel 56
- A.1 Design sheet no. 1 74
- A.2 Design sheet no. 2 75
- A.3 Design sheet no. 3 76
- A.4 Design sheet no. 4 77
- A.5 Design sheet no. 5 78





# Introduction

Magnetic resonance spectroscopy (MRS) is a technique that allows medical experts to analyze the chemical environment of tissue, utilizing the differences of magnetic fields of atom nuclei. It produces a spectral curve, where peaks represent the presence and concentration of various tissue metabolites (products of metabolism). Recently, MRS witnessed a transition from a purely research technique to one that has a potential for clinical use. The fact that it can be obtained using a standard clinical MR unit and commercial or open-source software has made it available to use for early diagnosis of neurological diseases and other pathologies [2, 3].

As spectroscopy moved into clinical practice, researchers are now using it for exploring the variations of metabolite concentration in different individuals, over time, and in various locations of the brain [4], which can further expand the diagnostic potential of MRS. However, visualization research has paid little attention to spectroscopy and an interactive, intuitive visualization tool that would support such analysis is still missing.

This thesis is a part of the research project of Laura Garrison at the University of Bergen [1], which aims to fill this gap and develop a tool supporting the analysis of variations of metabolite concentration among various data samples acquired by MRS. Within this thesis, my goal is to process the initial sketches of the design of the application and its user interface provided by Garrison and her team, and implement a fully working prototype application.

In Chapter 1, I will describe the principles of MRS, the terminology used in the MRS research environment, and the standard process of analysis of spectroscopic data. I will also specify the requirements for our application. In Chapter 2, I will discuss the standard tools currently used to process spectroscopic data and the related work in the fields of visualization of spectroscopic data, heterogeneous data visualization, and table and unit visualization, that is relevant to this project.

In Chapter 3, I will focus on the interface and visualization design that Laura Garrison and her team iteratively developed in close collaboration with the domain experts in MRS, and I will explain

---

the changes in the design made in the implementation phase. Subsequently, in Chapter 4, I will address the choice of the web-based platform and specify four functional components of the system that will frame the further specification of data processes, computations, and implementation.

Chapter 5 is devoted to the underlying processes and computations that precede the visualization of the data. I will describe the data structures and transformations needed to successfully implement the proposed methods of visualization. Then, in Chapter 6, I will discuss some of the technical aspects of the implementation, mention the main obstacles I encountered during the process, and the solutions I proposed. Finally, in Chapter 7, I will summarize my work on this project and mention the feedback we received based on the testing of this prototype, and the possibilities for further improvements of the application.

I find it necessary to point out that even though my tasks were focused mainly on the implementation of this application, the project was developed in an iterative way, so the implementation process was intertwined with adjustments of the design. Therefore, I also had certain influence on the design choices, and the other team members had their influence on the choices with respect to the implementation. However, the final decisions with respect to the design were made by Laura Garrison, the decisions in the implementation were my responsibility and the code in the JavaScript files was written by me (the authorship is specified in each source file).

# 1 Project Context, Task Specification

In this chapter, I will discuss the premises of my work on the project. I will first briefly specify the principles behind MRS and clarify some of the terms that I will use in the text. Then I will describe the way acquired data must be processed before it can be used for the analysis, and I will finish this chapter by outlining the process of data analysis and exploration that our application aims to support.

## 1.1 Magnetic Resonance Spectroscopy (MRS)

Discovered in 1949 by research groups of F. Bloch and E. M. Purcell, the dependency of resonance frequency of protons on the magnetic field strength lies at the core of all magnetic resonance techniques [5, 6, 7]. Although originally used in physics and chemistry to analyze the molecular environment of a substance, the method called Nuclear Magnetic Resonance (NMR) Spectroscopy can be also used for the investigation of tissue of living humans and animals, i.e., *in vivo*. When speaking about *in vivo* NMR spectroscopy, the word “nuclear” has been dropped and the method is referred to as Magnetic Resonance Spectroscopy, or MRS [8, 4].

### 1.1.1 NMR Spectroscopy Basics

When a strong magnetic field is applied on a molecule, protons will align themselves along this magnetic field. To align them in a plane orthogonal to the magnetic field, a radio frequency (RF) pulse is applied — the protons gather the energy (proportional to the frequency of the pulse) which allows a transition to the aligned state. After the RF pulse is switched off, the protons return to their original state and release the energy, which is then recorded. The key observation is that the amount of energy needed to switch between the parallel and orthogonal alignment is dependent on the environment the proton is surrounded with [5, 7]. Therefore, by measuring the energy released by the transition, we get the information about the chemical environment of the substance in the MR scanner.

Since the energy and frequency of the pulse are proportional, after the Fourier transformation of the measured signal and a phase-correction (for details see [5]) we obtain a frequency spectrum with peaks around the resonant frequencies.

### 1.1.2 MRS Terminology

Nuclei of various atoms are used for MRS. The most commonly used nucleus is of the hydrogen atom, where we speak of **proton** or  $^1H$  spectroscopy (the nucleus of hydrogen consists of a single proton). Also phosphorus nuclei have been used frequently in clinical studies [3], and carbon, sodium, and fluorine spectroscopy is used less often [2]. The advantage of proton spectroscopy is that the signal can be obtained using a standard clinical MR imaging systems [4]. In the scope of this work, we take into account only proton spectroscopy since it is the one used by our collaborators.

In Figure 1.1 we can see a typical outcome of a MRS measurement. On the X axis, instead of a resonant frequency measured in Hz, there is a **chemical shift**. Chemical shift is computed as a frequency relative to a reference value, multiplied by  $10^6$ , and expressed in **ppm** (parts per million). By putting the resonant frequency relative to a reference value, the result is independent of the static magnetic field flux of the spectrometer [5]. By convention, the ppm values are decreasing from left to right, as opposed to a general intuition that values on an axis increase in this direction.

The peaks of the spectral curve characterize **metabolites**, end products of metabolism. Some of these, such as creatine (Cr), choline (Cho) and N-acetyl aspartate (NAA), are present in healthy tissue, others, such as lactate (Lac), indicate pathologies [3]. The peaks are characterized by their height and width at half-height. To quantify the **concentration** of a metabolite, we calculate the area under the corresponding peak [2].

Currently, there are two ways of acquiring MR spectra. Originally, it was possible to investigate only one spatial sample (voxel) at a time — a technique called **single voxel spectroscopy**, or SVS. At present, **magnetic resonance spectroscopy imaging** (MRSI), also known as chemical shift imaging (CSI), allows to capture spectra in a  $64 \times 64$  voxel grid with slices of 5 mm, covering the whole area of interest at once.

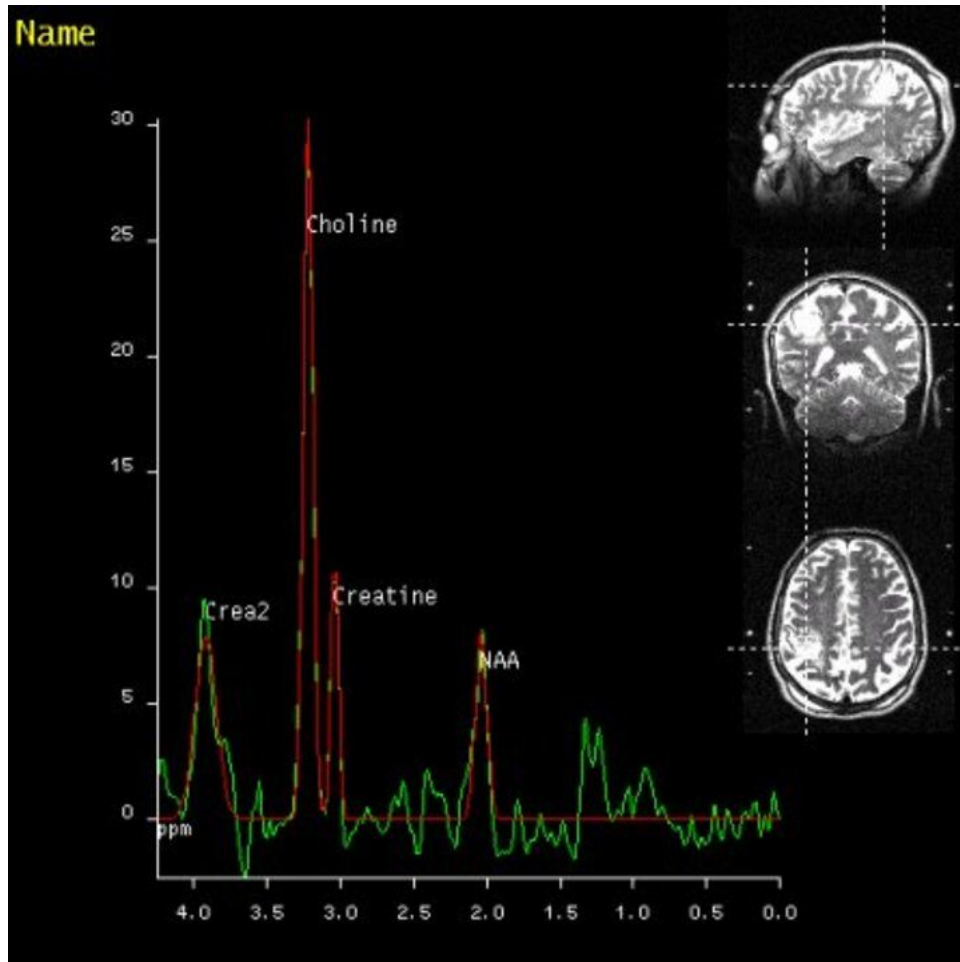


Figure 1.1: Typically, the result of a MRS spectrum acquisition is visualized on the MR workstation as a spectral curve with several identified metabolites and an anatomical image providing context [9]. Values on the x-axis represent chemical shift (measured in parts per million, ppm), metabolite concentrations are computed as integral values of respective peaks of the curve.

However, SVS still remains to be used due to a better signal-to-noise ratio [3], and since these provide a more detailed spectra, we focus on the SVS technique in our work [1].

An important parameter of a MRS measurement is **echo time** (TE), which is given by a delay between individual RF pulses during the acquisition. A shorter TE (down to 20–30 ms) allows for a more detailed signal, longer TE (often around 140 ms) is used to obtain a smaller amount of sharp peaks which are easier to analyze [4].

Typically, a MRS acquisition session captures a single moment in time. However, there are cases where the patient might be instructed to perform a task during which a spectrum is acquired, and then another acquisition follows when the patient is resting, resulting in a variance of metabolite concentration between those two acquisitions. In this context, we speak of the patient being in an **active brain state** or **resting brain state** [1]. This type of analysis may be used, for example, to assess the effects of a treatment of neurological disorders, e.g., dyslexia [10].

### 1.1.3 Data Processing Steps

The MRS experiment produces time-dependent data of damped oscillations mixed with noise, which is hard to interpret [11]. Therefore, a number of steps must be performed to obtain the desired spectral curve.

First, it is necessary to suppress the water signal as water has a much higher concentration in tissue than the metabolites of interest. The time-dependent data is usually multiplied by a window function to improve signal-to-noise ratio. The signal without water suppression can be stored separately and used as a reference to correct other artefacts, such as frequency shift or eddy current artefacts caused by switching magnetic gradients. After these steps, Fourier transformation is applied to obtain the frequency spectrum [11].

The spectral data is then processed using available MRS processing software which I will discuss in Chapter 2. It is necessary to estimate a baseline, which can be imagined as a background signal caused by experimental artefacts and resonances of large molecules. If not corrected, this can lead to errors in quantification of metabolites [11]. It is also common to have a prior expectation of the line shape, and

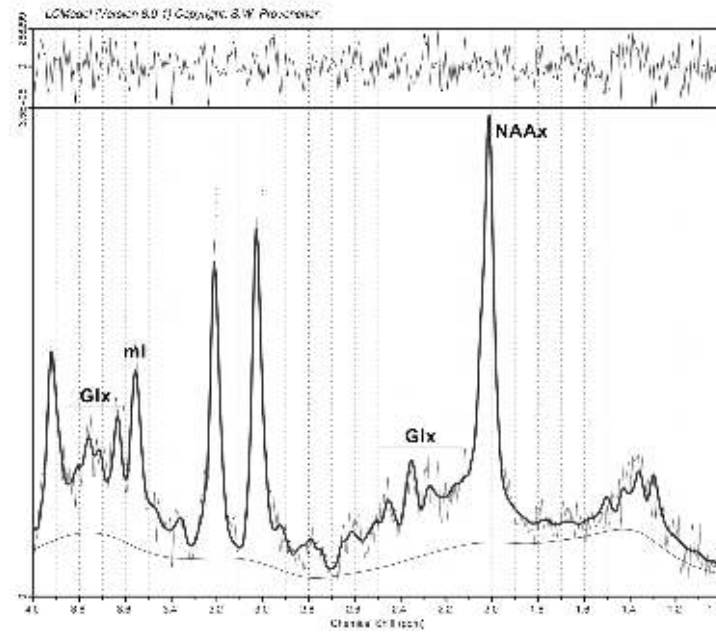


Figure 1.2: A typical visual output of the model fitting process [12]. The highlighted curve that represents the model fit is superimposed over the raw data, and the baseline estimate is displayed below.

so the peaks can be estimated with a fit using a model. However, it is always a matter of debate whether to use the model since the shape of in vivo MRS data is not always well known [11].

In Figure 1.2, we can see the baseline estimation below the curve. The smoother highlighted curve represents the model fit, the softer, more oscillating curve is the raw spectral data obtained by the Fourier transform.

Our application requires these steps to be performed independently, so that the data are available as a series of vectors — the raw data, baseline and model fit values to be plotted against the chemical shift (ppm).

In addition to the spectral data, additional information must be stored to provide context for the data analysis. The required metadata consist of:

- Anonymized patient information (identification, age, gender)

- Time of acquisition
- Echo time (TE) used during the acquisition
- Spatial location of the voxel, including an anatomical reference image
- Information whether the patient was in a resting or active brain state

### 1.2 Data Analysis

In the design study [1], we worked with six domain experts specializing in MR spectroscopy research in academic hospital setting. Based on individual and group interviews, we identified a set of core user tasks and design requirements for our application.

#### 1.2.1 Visualization tasks

We classify the visualization tasks according to the typology introduced by Brehmer and Munzner [13]: data discovery, production, search, and querying. These form the four key groups of tasks:

- **T1 (Data discovery):** In the first place, it is necessary to discover and verify key information about the data [1]. This includes visualizing the MRS spectral curve and providing the context by associating it with the metadata specified above.
- **T2 (Production):** New data elements are computed from the raw spectra. This includes computation of the peak integrals, and subsequently ratios of these integrals. Since the absolute metabolite concentrations vary significantly both between and within individuals, our collaborators, as well as papers focused on MRS research, repeatedly mention the significance of metabolite concentration ratios for understanding variability in the data [1, 2, 4, 10].
- **T3 (Search):** Often only a subset of the raw and computed data is of interest for the researcher. It may be desirable to look at



spectra from a given spatial location, examine gender variation, or explore a single time point in a longitudinal study. Another search task lies in identifying outliers: these can either indicate a pathology or a data quality issue [1].

- **T4 (Querying):** Finally, it is necessary to utilize the results of a search task for comparison and summary. This is the core discovery task that our application aims to support: to be able to identify, compare and summarize metabolite ratios across key dimensions of interest [1]. This will allow researchers to get a deeper insight into their data and track variances of these ratios across spatial locations, time points, or individuals.

### 1.2.2 Requirements

Based on the tasks, we formulated six requirements for the application as follows [1]:

- **R1:** Since our collaborators work in a hospital research environment and switch between workstations frequently, we have to provide a web-based solution to avoid the need of downloading third-party software and ensure that all security and privacy measures are met.
- **R2:** All the sensitive patient data must be properly anonymized before being loaded into the visualization tool. This involves culling all identifying attributes from the source files in the preprocessing stage.
- **R3:** To support tasks in T1, we should map chemical shift values to metabolite peaks, since this mapping is identical for all proton MRS spectra.
- **R4:** We must maintain the visual linkage between the spectral output and the metadata: voxel sample location and patient-specific information.
- **R5:** To support the central analysis in T2–T4, we must allow the computation of concentration ratios (individually and aggregated) for any combination of metabolite peaks in the spectra.

## 1. PROJECT CONTEXT, TASK SPECIFICATION

---

- **R6:** We must combine the aggregate ratio overview with the nested information on individual ratios by a layered approach.

Having described the nature of our data, the basics of the acquisition and processing, the tasks that have to be supported by our application, and the requirements that stem from these tasks, I will now discuss what tools are currently available in the field of MRS data analysis, and will provide the readers with a brief overview of other visualization research relevant to our project.

## 2 Related Work

In this chapter, I will explore currently available tools and methods for visualizing MRS data. I will first describe the standard methods of viewing the spectra in a MR workstation and in software dedicated to standard processing of MRS data, and I will address more complex visualization software for MRS data analysis. Then I will discuss other relevant fields of visualization research, linked to heterogeneous and multidimensional data and tabular and unit visualizations.

### 2.1 Standard Tools to View MRS Data

The tools described in this section are commonly used by MRS researchers for data processing and analysis. It is crucial to understand their functionality and design since these are the tools our collaborators are familiar with, and our application aims to complement these to create a complete working set.

#### 2.1.1 Standard Processing of MR Spectra

After the acquisition, the MR spectrum is usually visualized directly at the MR workstation. In the case of SVS measurement, the visualization usually consists of the spectral curve with an identification of the most significant metabolite peaks and a referential anatomical image showing the location of the voxel, as in Figure 1.1.

Subsequently, the process of model fitting and metabolite quantification takes place. From the commercial software tools, LCModel [14] is the most widely used for this step [4, 15]. It is mentioned as an advantage that for metabolite fitting, the entire spectral pattern of the metabolite is used [4], i.e., the case where a metabolite is represented by multiple peaks at different frequencies is covered. LCModel then visualizes the output in a way similar to that shown in Figure 1.2, with the original data and the model fit superimposed, and includes an error estimation for the quality of the model fit.

Several open source tools are available. Tarquin has been developed to be equally robust and accurate as LCModel for use in a clinical setting [16], offering both a GUI and a command line interface. Since

## 2. RELATED WORK

---

it is the tool most frequently used by our collaborators, one of the co-authors of the design study [1] has prepared scripts that use Tarquin to process the MRS acquisition output into the format required for our application.

Other open source tools for this level of MRS data processing include Matlab libraries Gannet [17] and OXSA [18], and also more advanced software, discussed in the next section, offers this type of functionality.

### 2.1.2 Advanced MRS Data Analysis

More advanced software packages allow for further manipulation and interpretation of MRS data. A widely used open source tool is jMRUI [19], which offers a similar functionality for quantification and model fitting as the tools mentioned above, but includes also features for database creation and advanced processing of MRSI data, such as mapping metabolite concentrations to an anatomical image. Besides these features, additional plugins are available for purposes of classification and diagnosis of pathologies.

The jMRUI Clinical Viewer plugin supports the visualization of metabolite concentration ratios (see Figure 2.1), which is also one of our requirements (R5). However, this plugin focuses on multi-voxel (MRSI) spectroscopy and allows to show only a single ratio at a time, whereas we aim for a comparison of various combinations of metabolite ratios in a single-voxel approach.

Similar functionality focused on MRSI is offered by another open source software, SIVIC [20]. A feature of interest here is the possibility to track the variation of a metabolite concentration in a time series, displayed as a time curve. This approach is also used by our application in the form of a nested sparkline to visualize changes in a metabolite ratio over time (see Chapter 3 for details).

From our review of the currently available software for MRS data analysis, it became clear that none of the standard tools known to us can support the visualization tasks we defined in the design study, and the comparison of metabolite concentrations can be done mostly by viewing the spectral curve(s), without proper quantification of the metabolite ratios.

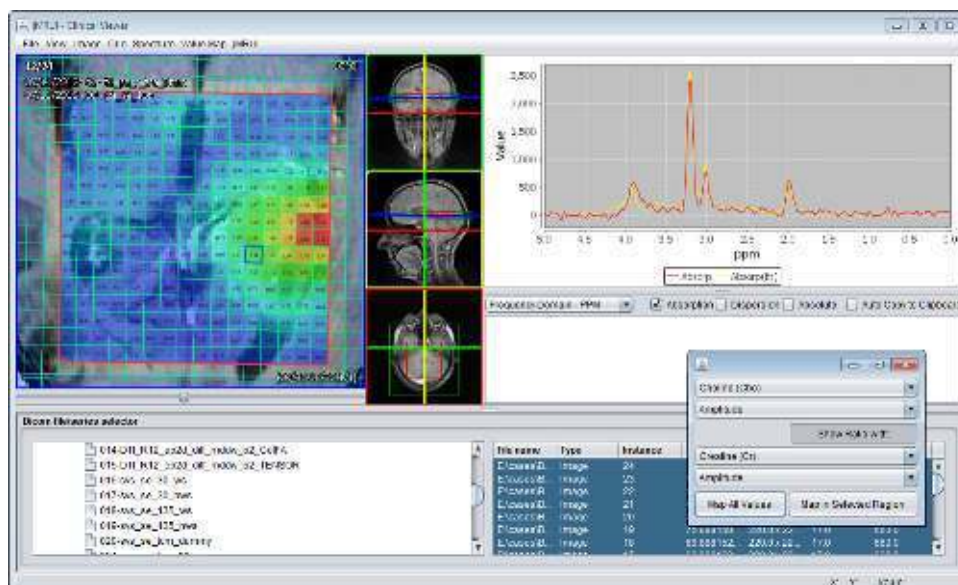


Figure 2.1: The jMRUI Clinical Viewer allows metabolite ratio computation and visualization as a heatmap over the anatomical image.

## 2.2 Related Visualization Research

In this section, I will provide an overview of visualization research that is relevant to our project. I will focus on the multidimensional and heterogeneous data visualization, and will also discuss methods of linking spatial and non-spatial information within the user interface.

### 2.2.1 Visualization Research Related to Spectroscopy

To the best of my knowledge, relatively few research projects have taken the visualization of MRS data beyond what the widely used above-mentioned tools are offering. Feng et al. have tested the effectiveness of Scaled Data-Driven Spheres (SDDS), a glyph-based visualization technique for multivariate three-dimensional data, applied to MRS [21]. The sphere radius represented data magnitude, the color of the sphere distinguished metabolites. The technique proved to be efficient especially for correlation identification, and was later extended by parallel coordinates plots and scatterplots, to include uncertainty

## 2. RELATED WORK

---

[22]. The focus here is mainly on the identification of relationships between anatomical MRI images and metabolite concentrations, but it relates to our project, as it allows for the comparison of metabolite concentrations in an abstract way.

A similar approach was used by Nunes et al. [23], who combine a visual analysis framework ComVis [24] with MITK [25], a framework for processing and display of medical images. Their goal is also mainly to provide linkage between metabolite concentration and anatomical image, however, this tool allows for the computation of metabolite ratios and a flexible comparison of one ratio to another. Similarly, Marino and Kaufman [26] link attributes acquired from MRS into the anatomical in a way of importance-driven direct volume rendering in the context of prostate cancer research.

Where our approach differs is that we focus on the abstract visualization, support the overview and exploration of all possible metabolite concentration ratios, provide context with additional data attributes, and allow for a flexible comparison, aggregation, and filtering based on these attributes.

### 2.2.2 Multidimensional and Heterogeneous Data Visualization

Related to the applications discussed above is InSpectr [27], a tool linking multiple views and multiple data sources (including spectroscopy) to provide information about the chemical composition of a sample. This solution works with pie-chart glyphs showing the main components of the spectrum superimposed over a structural image, and with transfer functions, defined over a spectrum, that map to the color in the structural image. InSpectr was later extended by isosurface similarity maps introduced by Bruckner and Möller [28]. As before, our application focuses on the abstract visualization part, but we take inspiration in combining and nesting multimodal information, and we apply the concept of isosurface similarity maps to ratio strength between metabolites.

Meyer et al. presented a way of analyzing the shape of time series curves in Pathline [29]. The time series data are shown in a matrix of small multiples the authors call a *curvemmap* (since it resembles a heatmap), each modality combination (in this case gene type and species) is represented by an element in the matrix showing the corre-

sponding time series curve. This approach was adopted by MulteeSum [30], again to visualize gene expression over time. Our visual encoding of variance of metabolite concentration ratio in time is inspired by this approach.

An idea combining graphical visualization primitives, superimposed over a structural image, and showing time series data in a matrix was introduced by Stoppel et al. in their primitive called *graxel* [31]. These are small units showing the variance of a value over time, which can be aggregated or resized dynamically.

### 2.2.3 Tabular and Unit Visualization

We can see that many of the examples I mentioned are visualizing data elements as units organized in a matrix or table. This is a crucial approach for our application as well as for visualization in general, and deserves more attention. As we formulated in our requirement R5 and the set of tasks T3, we need to compute a ratio of concentrations for all the combinations of metabolites and then allow the user to search among them. This will necessarily result in a number of units that we cannot visualize only as an aggregated value, but we need to be able to visually distinguish between these units.

Although the term *unit visualization* has not been used until recently, the principle of having a bijective mapping between data rows and visual marks can be traced through all human history, even in the Paleolithic era [32]. One of the first well-known systematic approaches to unit visualization is the “picture language” ISOTYPE developed by Otto Neurath in 1930s [33]. However, the repeating icons were used mainly to express quantity, which is not our goal.

An approach to visualizing units in a matrix was introduced in 1967 by Jacques Bertin [34], who developed a system of reorderable matrices to support finding patterns in tabular data. It was used and extended by Perin et al. into Bertifier [35] — an interactive web application for creating tabular visualizations. Our approach is inspired by the matrix view that allows a certain degree of reconfiguration, although it is not possible to reorder the rows and columns, since metabolite peaks on the spectral curve do have an ordering given by their chemical shift.

## 2. RELATED WORK

---

Within this tabular view, we add a nested layer of unit marks, inspired by Atom — a grammar for unit visualizations developed by Park et al. [32]. As the authors of Atom are mentioning in their work, unit visualizations have a weakness of limited scalability and high potential of creating visual clutter, which is why we chose these to be a secondary nested level, shown on demand in the tabular overview.

The focus+context technique for the tabular view was introduced by Rao and Card in Table Lens [36], who use three ways of interaction with the table: *zooming* — changing “the amount of space allocated to the focal area without changing the number of cells contained in the focal area [36]”, *adjusting* the amount of content viewed and *sliding* — changing the location of the focus area. We combine zooming and sliding in a single interaction that enlarges a cell of interest and shifts the area of focus.

The authors of Atom also discuss grammars and libraries available for the implementation of such unit visualizations and conclude that it is mostly low-level programming libraries, such as D3 or Processing, that are used [32]. We have reached the same conclusion for our implementation.



### 3 Interface and Visualization Design

This chapter is devoted to the design of the proposed tool that was created by Laura Garrison, based on the input from our collaborators, described in detail in our design study publication [1]. My role in the design process was to confirm the feasibility of the design with respect to its future implementation and suggest changes and possible improvements. I was also involved in interviews and presentations with our collaborators to better understand the requirements and goals of our application.

In this chapter, I will first describe the proposed design of the user interface and visual encodings, compare it with their final form, and explain the changes that have been made. Then I will discuss the interaction possibilities we decided to support in our application.

The process of creating the initial low-fidelity prototypes was guided by the Five Design-Sheet methodology developed by Roberts et al. [37]. It involves paper-based sketching of ideas for the first design sheet, combining those in three principle designs for sheets 2, 3, and 4, and finally converging to the final prototype in the fifth sheet, that is meant to be implemented. I attach the five design sheets, created by Garrison, in Appendix A. Since the design process is not the key part of this thesis, I will not describe in detail the previous design sheets but rather focus on the final design sheet (Figure A.5), and compare it with the interface of the final application (Figure 3.2).

To discuss the possible risks and drawbacks of the design, I will use the terminology introduced by Kindlmann and Scheidegger [38] who describe the relationship between three structures in the visualization process: mathematical structure in the underlying data, concrete representation of this data in a computer, and the resulting structure of the visualization. They define three design principles as follows:

**Representation Invariance** The visual representation should be invariant to any changes in the internal data representation. If changing the representation results in a change in the visualization, it has a *hallucinator*.

### 3. INTERFACE AND VISUALIZATION DESIGN

**Unambiguous Data Depiction** On the other hand, any change in the underlying data should be represented by a change in the visualization. Failing this principle creates a *confuser*.

**Visual-Data Correspondence** Related to the previous principle, a significant change in the underlying data should be represented by a similarly significant response in the visualization. If an important change is not displayed accordingly, the visualization has a *jumbler*.

## 3.1 Interface Components

The proposed interface is divided into two main sections, illustrated in Figure 3.1. The left panel shows an overview of the data samples that are available for the analysis, the right panel shows a heatmap matrix view representing metabolite concentration ratios in selected spectra. I will now describe the elements of both views in more detail.

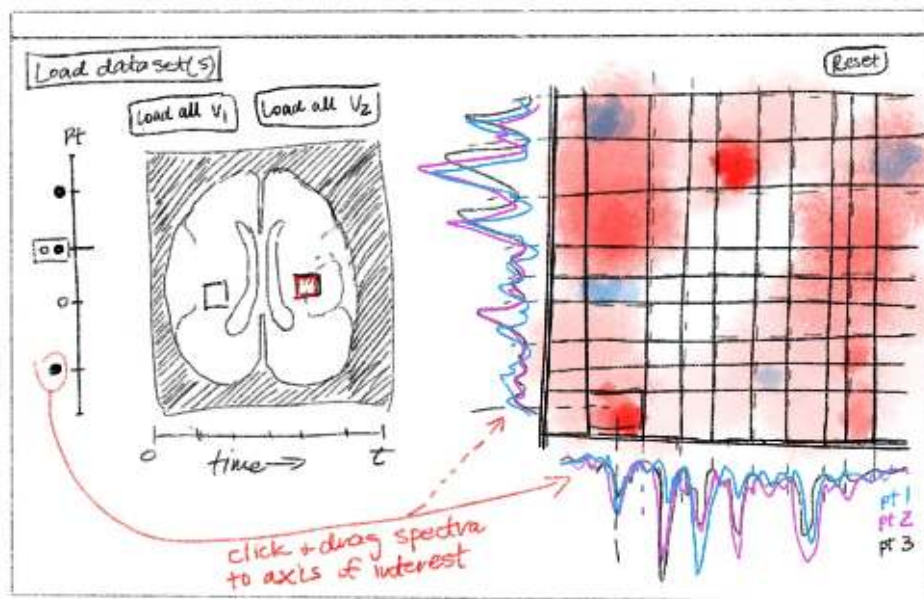


Figure 3.1: A sketch of the final stage of low-fidelity prototyping, showing the main components of the interface.

### 3.1.1 Left Panel: Data Overview

The left panel serves as an overview of the dataset that is currently loaded in the application. The main feature is an anatomical image showing the location of the currently selected spectral voxel. Next to it, there are widgets that allow the selection of a voxel based on the metadata (patient ID, time of acquisition, brain state).

In the prototype sketch (Figure 3.1), we can see two selectors: points on the vertical axis represent individual patients, the color of the point shows whether the acquisition was done with an active (black) or resting (white) brain state. Originally, the black and white color was meant to distinguish between proton and phosphorus spectra, but we decided to focus on proton spectroscopy exclusively, freeing up this channel for another modality. The horizontal selector contains the acquisition time points of the patient that is currently selected. The key part in the left view is an anatomical image of the brain showing the location of spectral voxels from the selected time point of the given patient.

Compared to the prototype, there are three noticeable changes in the implementation of the left panel (see Figure 3.2). First, the element to select between the active and resting brain state acquisition was added to the horizontal (time point) selector. The reason for this is that some acquisitions may have been performed only in one of the states and others in both states. If the brain state was included in the patient selector, individual acquisitions on the time point selector would appear and disappear (creating a possible hallucinator), and it would not be possible to easily see whether an acquisition was performed in both states.

Second, the anatomical image contains the location of only one spectral voxel at a time. After consulting and preparing the scripts for data preparation, it turned out that showing the location of multiple three-dimensional voxels on a two-dimensional image would be misleading. With a single viewing plane, a change in the third dimension would not be shown (i.e., would create a jumbler) and if we show the location in multiple viewing planes, such as in Figure 1.1, the view could become cluttered. Therefore, we moved the patient selector to the top left corner and the vertical selector contains individual spectral

### 3. INTERFACE AND VISUALIZATION DESIGN

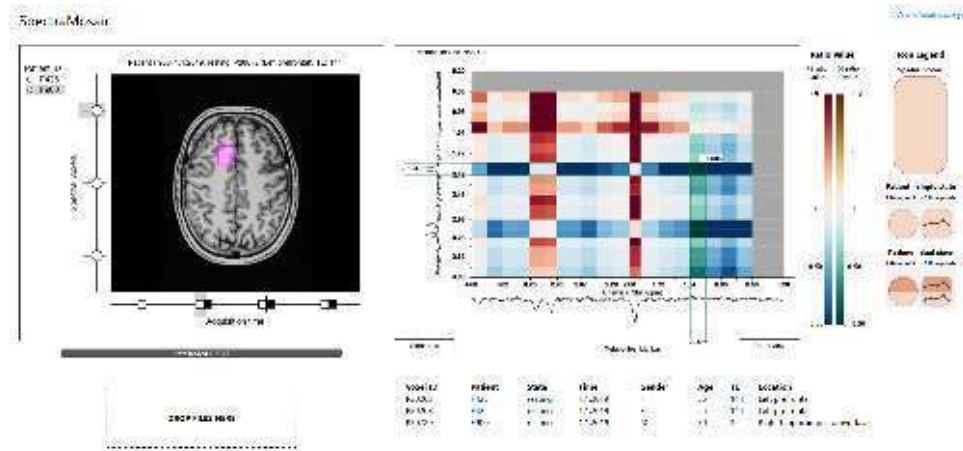


Figure 3.2: The user interface in the final stage of this version of SpectraMosaic

voxels of the given patient during the selected acquisition (time point and brain state).

Finally, the two buttons above the anatomical image in the proposed sketch were supposed to select all voxels at a specific location for the analysis. We have decided to omit this feature in the current version and prepare a more sophisticated functionality of this kind in the future instead.

Another possible drawback of the design could be the inability to see the spectral curves in the overview. This renders all the variability in the spectra invisible at the first glance, which we may call a confuser. However, this tool requires previous processing of acquired data (as described in Chapter 1), so we can suppose that the researcher has already seen the individual spectra and is mainly interested in their comparison. Therefore, this selection can be effectively performed even without seeing each of the spectral curves first.

Additional to the voxel selection and overview features, there is a simple drag-and-drop feature for data loading and a progress bar showing the loading status.

### 3.1.2 Right Panel: Metabolite Ratio Map

The key element of the right panel is the matrix view presenting the metabolite concentration ratios. Spectra are placed along an x- and y-axis (each axis also contains the chemical shift values, as specified in requirement R3). They are then divided into 20 adjustable regions — a spectrum typically contains at most 20 interesting metabolite peaks [1, 39] — producing a 20x20 matrix where each cell represents a ratio of spectral integrals (i.e., metabolite concentrations) of corresponding regions. The value of each ratio is mapped to a color on a diverging colormap to allow immediate identification of regions with major differences. This way we can easily compare ratios for any combination of metabolite peaks (requirement R5).

A change with respect to the initial prototype is a decision to plot all the spectral curves in black, and distinguish between individual patients only by dynamically highlighting the curves on interaction, as described in Section 3.2.

Within each of the matrix cells, we provide an additional nested structure available on demand. This structure contains a system of simple glyphs whose color is obtained from an aggregated integral ratio within a subset of the spectra currently displayed in the right view (for details about the calculations, see Chapter 5). This way we can see the overall aggregated ratio, as well as the ratios of individual voxels and subsets (requirement R6). We define the glyph structure as follows:

**Voxel location:** The highest level glyph (a capsule shape) represents all voxels in a specified location (e.g., left prefrontal region or right hippocampus-amygdala in our sample dataset).

**Patient:** If spectra from multiple patients in the same location in the brain are available, then each of the patients is represented as a disk shape within the pill glyph. If spectra of only one patient are available for the location, the disk glyph is not shown.

**Brain state** during the acquisition: Further on, if the acquisition for the given patient in the given location was performed in multiple brain states, we split the disk glyph (or the capsule glyph in the

### 3. INTERFACE AND VISUALIZATION DESIGN

















Case	individ.	region	time pt	state	visual	Case	individ.	region	time pt	state	visual
1	single	single	single	single		9	multiple	single	single	single	
2	single	single	single	dual		10	multiple	single	single	dual	
3	single	single	multiple	single		11	multiple	single	multiple	single	
4	single	single	multiple	dual		12	multiple	single	multiple	dual	
5	single	multiple	single	single		13	multiple	multiple	single	single	
6	single	multiple	single	dual		14	multiple	multiple	single	dual	
7	single	multiple	multiple	single		15	multiple	multiple	multiple	single	
8	single	multiple	multiple	dual		16	multiple	multiple	multiple	dual	

Figure 3.3: The 16 encoding scenarios for nested information in the matrix view, as presented in [40].

case of a single patient) in half, each half representing one brain state.

**Time series** for a given brain state, voxel location, and patient: Finally, if we have multiple acquisitions for a given patient in the same location and brain state, we show a sparkline showing the variation of this particular ratio over time. If the sparkline is nested inside a disk shape, the disk is transformed to a rounded square to provide more space for this nested information.

Based on whether these modalities are present in the current set of spectra, combinations of 16 different encoding scenarios are possible. These are illustrated in Figure 3.3, created by Garrison for a poster presented at the EuroVis conference in 2019 [40].

Next to the matrix view, there is a legend explaining the nested glyph shapes and a mapping of the ratio values to color. We use two separate diverging color maps to distinguish between a positive and a negative ratio value (a value on the spectral curve is considered

negative if it lies below the baseline). A red-blue map is used for positive and a green-gold map for negative ratios. Making this distinction, rather than computing just the absolute values of the peak integrals, was mentioned by our collaborators as important. The red-blue colormap was chosen as it is common in scientific visualization and familiar to our collaborators, the green-gold colormap is an analogy that can be easily distinguished but does not create too many disparate colors in the matrix [1].

Moreover, after a suggestion from the collaborators, we decided to leave the outer regions of the matrix (corresponding to the chemical shift values of 0.7 ppm and lower) inactive, as these regions do not contain any metabolite peaks. However, it was mentioned that we should still show these parts of the spectral curve as a way of data quality assurance. Therefore, we decided to show matrix cells in this area in grey color and disable the interactivity, while keeping the whole spectral curve displayed. Below the right panel, we placed a simple table showing the metadata for the spectra currently analyzed in the matrix view.

## 3.2 Interaction

To fully support the process of the exploratory analysis, as the title of our project states, it is crucial to provide a sufficient level of interactivity, and to visually link the left and right panels. To describe the exploration process, I will use Shneiderman's well-known Visual Information Seeking Mantra: *Overview first, zoom and filter, then details on demand* [41]. The process of information seeking in our application is not identical to this mantra: zooming is not necessary in the left view and in the matrix view, zooming and providing details on demand is performed in a single step, rather than as two separate actions. Still, the general workflow is inspired by Shneiderman's suggestions as they provide a reliable means to ensure efficiency of a visualization system.

In the sections below, I will explain how the data are selected for the analysis, what types of interaction are possible inside the matrix heatmap view, and how the two panels are linked.

### 3. INTERFACE AND VISUALIZATION DESIGN



Figure 3.4: The process of selecting a spectral voxel for the analysis consists of four steps: 1. selecting a patient, 2. selecting a specific acquisition (time and brain state), 3. selecting a voxel within this acquisition and 4. dragging the element over to the right panel.

#### 3.2.1 Interactions in the Left Overview

The left panel, serving mainly as an overview of the dataset, does not require much interaction. The main part of interaction is linked to the display of metadata. It is necessary to provide the overview in a simple and fast way, and since showing metadata for all available voxels at once would be inefficient and create visual clutter, we show these dynamically when the mouse is hovering over a selector element.

For the time and brain state selector on the x-axis, we show the corresponding date and brain state as a tooltip next to the mouse cursor. For the spectral voxel selector on the y-axis, we show full metadata as a heading over the anatomical image, and switch the anatomical image to the location of the voxel. In the context of this view, the metadata and switching image could be considered *details on demand* in the information seeking process.

The spectra are transferred to the right view for analysis by dragging either an element of the y-axis (a spectral voxel selector), or the square showing the voxel's location on the anatomical image, to a desired axis area in the matrix view. From the point of view of Shneiderman's typology, the *filtering* part would be selecting the voxels to include them in further analysis (as illustrated in Figure 3.4).



### 3.2.2 Interactions in the Matrix View

Once there are spectra on both axes of the matrix view, a heatmap (*overview*) appears and the user can explore the integral ratios for any combination of peaks. Initially, the spectra are divided into 20 blocks (tiles) of the same size, but these can be adjusted at any time. The tile borders are marked by ticks on the axis, which can be dragged left and right (or up and down on the y-axis) to align exactly with a peak of interest.

Since this gives the user a large degree of freedom when defining the tile regions, we provide a “reset” button in case it is easier for the user to start over with the adjustments, rather than re-adjust the current state of the interface. Once these regions are adjusted, the user can click on a matrix cell to expand it (*zoom*) and see the nested glyph structure (*details on demand*). Clicking on the expanded cell again will close it, and only one cell can be expanded at a time (i.e., clicking on another cell would close the one that is currently expanded).

By hovering the mouse cursor over a matrix cell, or over a glyph in the nested structure, the user can see the corresponding ratio value as a tooltip next to the cursor, to provide the exact information that is impossible to deduce from the element’s color. We also show the names of metabolites possibly located in the respective regions of a spectrum for each axis.

There are two ways to remove spectra from the right panel: First, we provide a button that will remove all the spectra from both axes. The second way is to double-click on the spectral curves at the axis and select the voxel to be removed.

### 3.2.3 Linking the Two Views

To maintain the linkage between the dataset overview on the left, the glyph structure in the right panel, and the metadata table below, we developed a system of highlighting (see Figure 3.5). This way, we can properly fulfill the requirement R4 and easily link the spectral curves to their corresponding metadata. When the mouse is hovering over a glyph in the matrix cell, the following elements will be highlighted:

1. Spectral curves of voxels in the subset represented by the glyph will highlight in fuchsia.



## 4 System Design

So far I have defined the problem domain, the requirements for our application, and the design of the user interface. In the following three chapters, I will address the technical aspects of SpectraMosaic, the choice of platforms and tools for its implementation, handling of the data inside of the application, and the computations performed during the visualization process.

In this chapter, I will discuss the choice of a single-page web application as the most suitable platform for our purposes. I will compare its advantages and disadvantages to other options and provide the rationale for this choice.

### 4.1 Benefits of a Web-based Application

Besides the advantage of not having to install any third-party software, as formulated in our requirement R1 in Chapter 1, there are more reasons why it is beneficial to create SpectraMosaic as a web application. First, since this project explores an area of research that is relatively new, we can expect frequent changes based on the input from our collaborators, so an iterative approach to the development will be necessary. It is known that websites are typically developed in such processes that support rapid prototyping and continuous change [42], but we can also reverse this statement and say that the web environment well supports frequent changes since there is no need to redistribute and reinstall new versions of the software, and the deployment is usually trivial, especially for JavaScript applications [43, 44].

Second, thanks to the rapid development of many popular web browsers, modern implementations of JavaScript are optimized to perform well even in tasks that require more intense computation [43]. It is not the case of SpectraMosaic at this stage, but if there was a decision to incorporate, for instance, dynamically rendered three-dimensional anatomical images, the usage of JavaScript would no longer be an issue, as it would have been in the past [44]. It is already relatively common to implement visualization tools that require intensive computation as web-based applications [45, 46].

Finally, as the authors of Bertifier mentioned, the possibility of online data import is another advantage of web-based tools [35]. In their case, it is possible to import data from a Google spreadsheet using a URL. In the case of our application, it would be beneficial to be able to connect to a server in the hospital to import data, rather than to download them and use them locally (for more discussion on possible future extensions, see Chapter 7).

## 4.2 Single-page Application

Typically, the client side of a web application that runs in the user's web browser would frequently communicate with a server, where the main application logic (in a commercial environment referred to as business logic) would be executed. This was also the approach for earlier web-based visualization tools, as in ParaWebView, where an image would be generated by the server, using its parallel data processing and rendering capabilities [47].

However, there is a tendency of moving more computation to the client side of the application and restricting communication with the server to loading the page and scripts. The main reason for this is that with a traditional model-view-controller server framework, an unreasonable amount of time is spent waiting for a page to load, resulting in high productivity costs [43] and a discontinuous experience for the user [44]. This is why a notion of a single-page application (SPA) has been introduced: in this approach, the entire application is delivered to the browser at once and is not reloaded during its use [43].

As illustrated in Figure 4.1, the only tasks where the interaction of the SPA with the server is required are authentication, authorization, validation, and data storage and retrieval. As none of these are a part of the prototype being developed so far, we can easily omit the server side from the system's architecture (a server that only delivers the web page would be trivial to implement at any point) and focus exclusively on the client application itself.

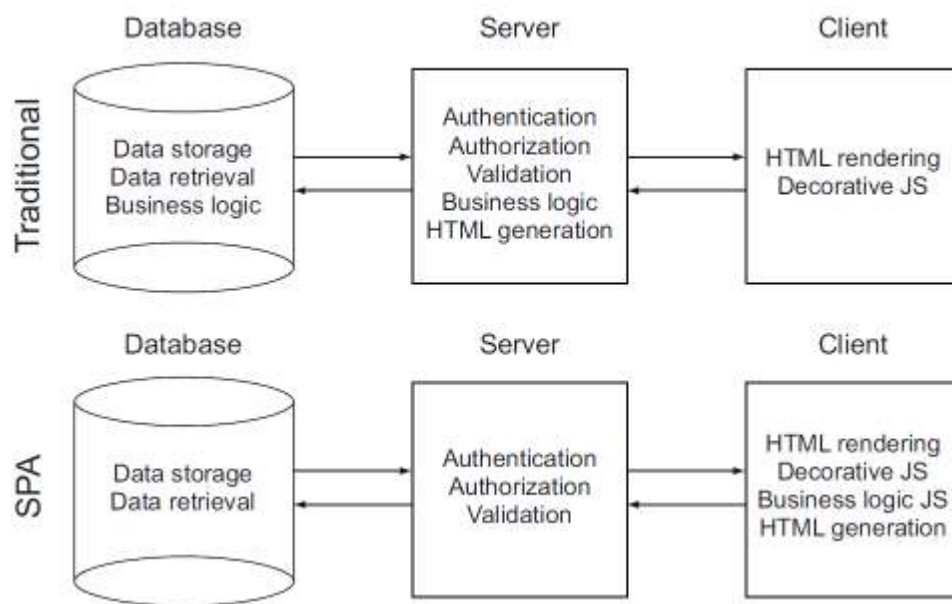


Figure 4.1: The comparison of a single-page and traditional approach to the web application architecture. In the single page application, all the logic and computation is executed in the client's browser, and the remote parts serve mainly for authorization and data storage and retrieval [43].

### 4.3 Client Application Components

The client application can be divided into four main functional components. Two of them correspond to the components of the visual interface, the additional two are responsible for the underlying file manipulations and computations. I describe the four functional components as follows:

1. **C1: Loading:** This component is responsible for handling the drag-and-drop loading operations and processing of the file structure.
2. **C2: Underlying Data Operations:** This component handles all the computations that are not directly tied to the state of the user interface. This includes normalization of values, up- and down-sampling of vectors, and any precomputation. Additionally, the main data structure is maintained here.
3. **C3: Left Panel:** This component is responsible for rendering of all visual elements and handling the interactivity in the left panel.
4. **C4: Right Panel:** Similarly, all the rendering and interactivity for the right panel is handled here. In addition, the computation directly connected to the heatmap visualization is performed in this component.

Before I describe the functionality and implementation of these four components in detail, I will provide a general overview of roles of these components in the data analysis and exploration process, as illustrated in Figure 4.2.

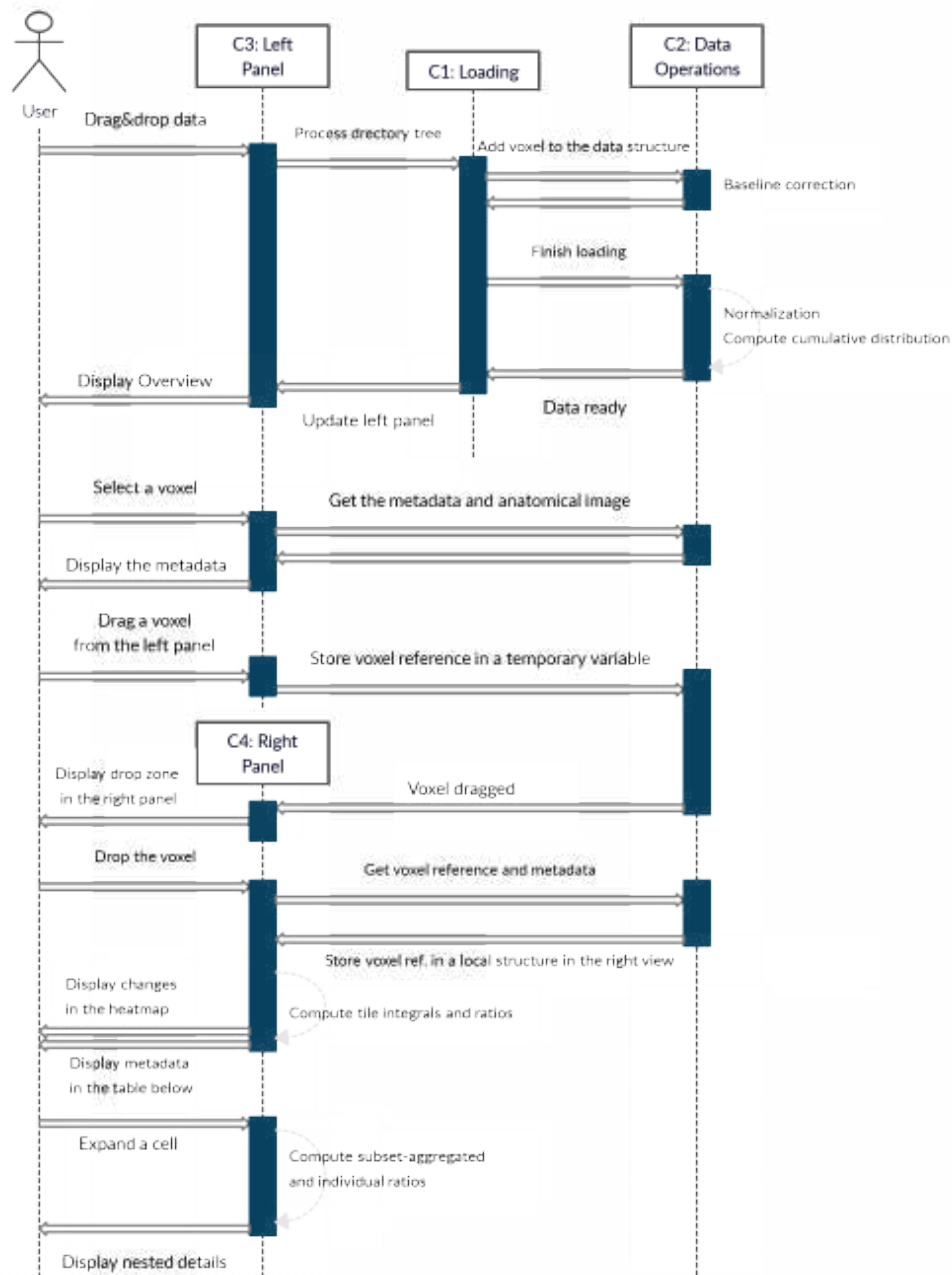


Figure 4.2: This diagram shows the roles of the components in the main events during the whole process of data analysis, from file loading to the display of nested glyphs in the matrix view.

#### 4. SYSTEM DESIGN

---

We can see that the components are interacting more intensively during the loading and preprocessing stage. Once the data are prepared and stored, the task of C1 is done completely, and C2 serves mainly as a data storage further on. As the left panel (C3) always displays the whole dataset, it does not have its own internal data structure and always takes the data stored by C2. In the right panel (C4), the user usually explores only a subset of the data, therefore, it is beneficial to maintain an internal data structure. The reason for this is that the computation of integrals and ratios is always performed only on the currently explored subset of data, since it needs to be performed dynamically during the user interaction with the matrix view. The actual spectral data in the local structure are referenced rather than copied, to maintain linkage and not to create any duplicities.



## 5 Computations and Data Processes

In this chapter, I will describe the data structures used in the application and the underlying computations that span from loading the dataset to the display of all visual elements of in matrix view. This includes loading the data from files into the application, preprocessing of spectral values, maintaining data structures, and computation of integrals and ratios.

### 5.1 Input Format and Loading

The data are processed using Tarquin (see Chapter 1 and Chapter 2) and stored in a directory structure containing all the required files that can be loaded into the application using a drag-and-drop feature. Subsequently, a data hierarchy is created corresponding to the hierarchy of information displayed in the left view, as described in Chapter 3.

#### 5.1.1 Required Format and Structure of Input

As a result of the data preparation process described in Chapter 1, we obtain a directory tree. The root directory contains subdirectories for each patient, named with the anonymized patient ID, and a header file. The header file contains the required metadata (patient's age and gender, echo time, brain state, voxel location, and time of acquisition for all the voxels in the dataset).

The patient's directory then contains subdirectories representing individual voxels. These contain a CSV file with the spectral data in four columns: chemical shift, raw spectral curve, model fit, and baseline (see Chapter 1 for details on spectral data processing and Figure 1.2 for the visual comparison of these types of data). It also contains anatomical images with a highlighted location of the spectral voxel in three orthogonal views: axial, coronal, and sagittal [7]. Additionally, the voxel directory contains an image generated by Tarquin that visualizes the spectral data. This generated image is not used by our application, but is left in the dataset as a standard output of initial spectral data processing. Figure 5.1 shows the directory tree of an exemplary dataset with one patient and one voxel.

## 5. COMPUTATIONS AND DATA PROCESSES

---

```
neuroinflammation_set .....root directory
├─ neuroinflammation_header_info.csv.....header file
├─ F425.....patient ID
│   └─ P29184.....voxel ID
│       ├── P29184.7.coo-output.csv.....spectral data
│       ├── P29184.7.png.....spectral data graph
│       ├── P29184_mask_spectramosaic_ax.png.....axial image
│       ├── P29184_mask_spectramosaic_cor.png.....coronal image
│       └─ P29184_mask_spectramosaic_sag.png.....sagittal image
```

Figure 5.1: Directory structure of an exemplary dataset

Not all the data and files are currently used by our application. After a discussion with our collaborators, we decided to use the raw spectral data rather than the model fit for the purpose of quality assurance. For keeping the simplicity of the left panel, we show only one of the three anatomical images, which is currently set to be the axial image. However, we keep the unused files and data in the dataset in order to be able to easily switch between showing the model fit and raw data or between the axial, coronal, and sagittal image in the future.

### 5.1.2 Loading Process

To make the data input as simple as possible for the user, we decided to use a drag-and-drop feature and process the whole directory tree automatically. The user can drop either the root directory, or all of its contents at once, and then all the files are read and connected with information in the header.

Once all the files are read, a number of calculations have to be performed to prepare the data for displaying. The following steps are essential in the preprocessing stage:

All the vectors need to contain the same number of values, so that the integrals over the same region are comparable. Therefore, it is necessary to set a fixed vector length and resize all vectors to this size. This is achieved either by linear interpolation, if the original vector is shorter than desired, or by binning, if the original vector is longer.

The raw data vector must be transformed to deviations from the baseline, i.e., values from the baseline column are subtracted from the

original raw data. As mentioned in Chapter 1, the baseline represents a background signal that would cause errors during quantification if not corrected.

The corrected signal is then stored separately in a normalized form. Even after subtracting the baseline values, the signal usually still reaches numbers over  $10^5$ , so the integral values of peaks would be extremely high. Therefore, the values are normalized with respect to peak height, so that the sign is preserved for negative peaks. Given  $N$  vectors of length  $M$  and denoting the  $j$ -th value in the  $i$ -th vector as  $v_{ij}$ , the normalized value  $n_{ij}$  is computed in the following way:

$$max\_peak = \max_{k < N} \max_{l < M} (|v_{kl}|)$$

$$n_{ij} = \frac{v_{ij}}{max\_peak}$$

In addition, a normalization that would be more appropriate for displaying the data must be performed, which returns values in the interval  $[0, 1]$ . This way, it is easier to scale the spectral curve height according to the panel size. Similarly as above, the normalized value  $n_{ij}^*$  is computed in the following way:

$$global\_min = \min_{k < N} \min_{l < M} (v_{kl})$$

$$global\_max = \max_{k < N} \max_{l < M} (v_{kl})$$

$$n_{ij}^* = \frac{v_{ij} - global\_min}{global\_max - global\_min}$$

The normalization is always performed with respect to all the spectra currently loaded in the application, so that they remain comparable. I am aware that changing the values that are already displayed will break the Principle of Representation Invariance [38], as the displayed spectra can suddenly “shrink” when new data are added. However, we chose to normalize the values in this way instead of dividing them

by a fixed factor as the absolute values can vary greatly depending on the acquisition details [1], and no such factor would be convenient for all cases.

In order to perform integral calculations efficiently, a cumulative distribution is computed over each data vector after the baseline correction. With  $n_i$  standing for the  $i$ -th value in the normalized vector, we can define the  $i$ -th value of the cumulative distribution as

$$d_i = \sum_{k=0}^i n_k$$

This idea is inspired by the approach of Summed Area Tables [48], simplified to a one-dimensional case. This way, we can compute the discrete integral over an arbitrary interval  $[n, m]$ , approximated by the Riemann sum, by a simple subtraction, instead of summing all values in the desired region, i.e.,

$$\sum_{k=n}^m n_k = d_m - d_{n-1}$$

So, for each spectral voxel, we have four different vectors of data (original data, normalized data, displayed data, and the cumulative distribution), an anatomical image, and the assigned metadata found in the header file. According to this metadata, the voxels are sorted into a data structure which will be described in the following section.

### 5.2 Data Structures

The main data structure contained in the component C2 (further referred to as *DataContainer*) reflects the hierarchy of information in the left panel, as described by the voxel selection workflow in Figure 3.4. Individual voxels are first sorted by the patient identification, then by time of their acquisition, and finally by the brain state during the acquisition. To illustrate this hierarchy in Figure 5.2, I am using a UML class diagram. I find this to be the most fitting illustration although I do not use JavaScript classes in the implementation, as it clearly shows the relationships between objects.

We can see that the data are divided into a nested hierarchy of objects that corresponds to the selector elements in the left panel (see

Chapter 3 for details): *Patient*, *Timepoint*, *BrainState* and *Voxel*. Each of these objects holds the metadata associated to it and an array of objects at the lower level of the hierarchy. In addition to the metadata, each object contains the property *highlighted*, which serves as a flag indicating that the corresponding selector element in the left panel should be highlighted. The *Timepoint* object does not have this property since it is not represented by a single selector element (the acquisitions are represented by brain state elements, sorted and grouped by time).

By contrast, the matrix view in the right panel (C4), without the nested glyphs, does not have any hierarchy of elements. Therefore, in the local data structure of the matrix view (*MatrixData*), the hierarchy is “flattened”, so that all the metadata are stored at the same level as the spectral data, as in Figure 5.3.

There are three vectors of spectral values: *scale*, which stores the chemical shift values, *values\_disp* with normalized spectral values for display and *c\_sum* with the cumulative distribution. These are stored as a reference to the corresponding arrays in *DataContainer*, so that any change (e.g., normalization when new data are loaded) is immediately visible in the right panel as well.

Two additional vectors of values are present in the matrix view: *tile\_values* and *tile\_integrals*. These represent the spectral curve divided into 20 tiles. *Tile\_values* serve for displaying the spectral curve, its values are taken from *values\_disp* and are split into 20 shorter vectors, which are resized according to the actual size of the corresponding tiles in pixels. *Tile\_integrals* are computed using the cumulative distribution stored in *c\_sum*.

Finally, another hierarchy of data must be constructed for the nested glyph structure in a cell of the matrix view. As described in Chapter 3 and Figure 3.3, the elements of this nested structure are aggregated first by the voxel location, then by the patient ID, then by the brain state, and finally by the time of acquisition. Therefore, we need to maintain a third data structure, containing the elements stored in the matrix view, which is constructed only when the nested glyph structure is displayed. I named this data structure *ExpandedData*, as the glyph structure is shown only when a matrix cell is expanded, and it is illustrated by a class diagram in Figure 5.4.

Here, only the metadata needed to identify the respective subset of voxels (i.e., location, patient ID, time, etc.) are stored. Besides that,

## 5. COMPUTATIONS AND DATA PROCESSES

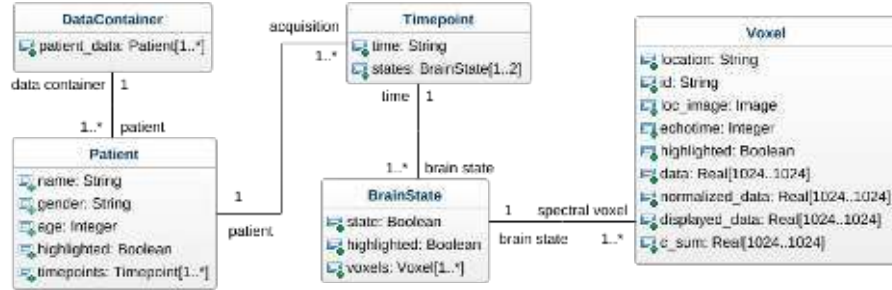


Figure 5.2: Class diagram representing the hierarchy of data stored in the component C2 (**DataContainer**). It consists of an array of **Patient** objects, each **Patient** object contains metadata associated with the given patient and an array of **Timepoint** objects, the **Timepoint** contains an array of **BrainState** objects (there will be two at most since we distinguish between active and resting state), and the **BrainState** contains an array of corresponding **Voxels**.

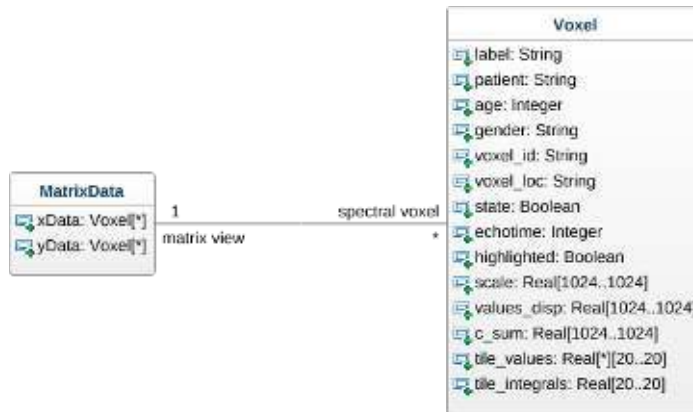


Figure 5.3: Data structure for the matrix view (**MatrixData**). Here, the voxels are stored without any hierarchical order. Each voxel can be assigned to one or both axes (the arrays *xData* and *yData* represent respective axes).

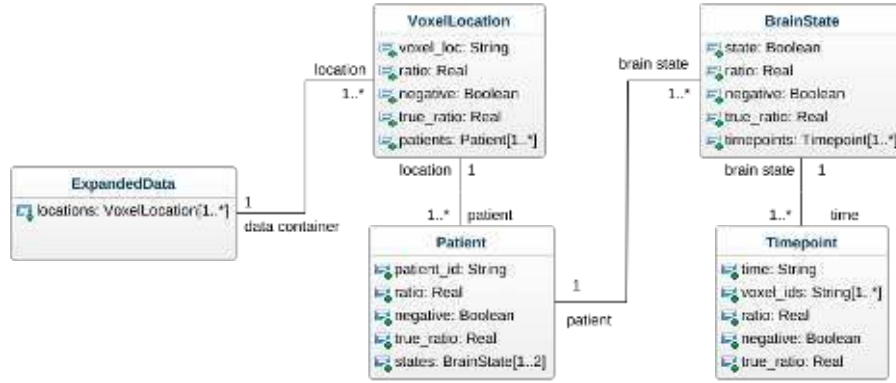


Figure 5.4: Data structure for the nested glyphs in the matrix view (ExpandedData). This data structure contains also the computed ratios for all the subsets of data. Because of issues with the color mapping, two different ratio values are stored (see Section 5.3).

already computed values of integral ratios are stored. The reason for the ratio being stored in the data structure, while the overall ratios are not stored in MatrixData, is that each level of the hierarchy has its own associated ratio, while there is only one ratio in each cell in the overview. Therefore, these ratios need to be stored along with the data hierarchy.

On the other hand, there is no need to store the spectral data here, as this data structure is reconstructed each time the nested glyph structure is demanded, and so the ratios can be computed directly. In the following section, I will focus on the computation of these ratios.

### 5.3 Integral and Ratio Computation

As mentioned in Chapter 1 and Chapter 3, exploration of peak integral ratios is the key task that our application aims to support. I will now describe how exactly the integrals and their ratios are computed and how they are mapped to the color of the matrix cell or the nested element.

Tile integrals for each spectrum are computed in the way described in Section 5.1 (using the cumulative distribution) and stored in *MatrixData* on these occasions:

- a voxel is dragged over to the right panel,
- a tile is readjusted by the user (an axis tick is moved),
- tiles are reset to the default size and position,
- new data are loaded into the application (all data are normalized again).

Once stored, the tile integrals are then aggregated over the spectra at each of the axes, and over all the subsets of spectra (more details in the following section) when the nested glyph structure is displayed, and used to compute ratios.

### 5.3.1 Peak Integral Ratios

Computing a ratio of integrals is very straightforward, once we have the integrals ready. However, the key problem with visualizing the ratio is assigning it to a color map. We chose two diverging color maps, where a white color in the middle represents the case where both integrals are the same in their absolute value (i.e., the ratio is equal to 1 or -1), and the colors diverge symmetrically to both sides.

However, the ratio  $r = \frac{x}{y}$  converges to zero when  $y$  gets larger than  $x$ , and diverges to infinity in the opposite case, so there is no direct mapping to the symmetric color gradient. Therefore, the ratio needs to be transformed in a way that will make it symmetric. I propose to compute the ratio in the following way:

$$r_{sym} = \begin{cases} \left| \frac{x}{y} \right| - 1 & x \geq y \\ -\left| \frac{y}{x} \right| + 1 & x < y, \end{cases}$$

$$r_{sign} = \text{sgn} \frac{x}{y}.$$



This way, we obtain a function that diverges in both directions, is continuous and symmetric, and the case where  $|x| = |y|$  is represented by  $r_{sym} = 0$ , as illustrated in Figure 5.5. However, we are no longer able to distinguish between the positive and negative ratio, so the sign must be stored separately as  $r_{sign}$  and then used as a parameter when choosing the color map. Moreover, in extreme cases, the function diverges to infinity (or minus infinity) so we need to define a cutoff positive and negative value, after which the color would be the darkest shade and values will not be distinguishable anymore. After a consultation with our collaborators, we have set these values of  $r_{sym}$  to -5 and 5, which are equivalent to values of  $r = [\frac{1}{6}, 6]$  or  $r = [-\frac{1}{6}, -6]$ .

In order to be able to show the color map in the legend with the original ratio values rather than  $r_{sym}$ , I have to define a transformation of  $r_{sym}$  to the original ratio value  $r$ :

$$r = \begin{cases} r_{sign}(r_{sym} + 1) & r_{sym} \geq 0 \\ (1 - r_{sign})(\frac{1}{r_{sym}-1}) & r_{sym} < 0. \end{cases}$$

In the case where multiple spectra are assigned to an axis, the ratios are computed using the mean of integrals of all the spectra at that axis in each tile, so the color of cells in the heatmap overview always represents the ratio of mean integrals. For the case where the user is interested in ratios within a specific group (subset) of voxels or an individual voxel, the glyphs inside an expanded matrix cell can be used.

### 5.3.2 Subset-aggregated and Individual Ratios

The nested glyph structure serves for exploring ratios of integrals in a specific subset of spectral voxels, or even in an individual spectrum in certain cases. These subsets are created according to the glyph hierarchy explained in Chapter 3 and represented by the corresponding data structure (ExpandedData). Here, we are interested in showing subsets of the dataset in the right panel as a whole, so we are not showing the ratio of integrals in a subset of spectra at the x-axis to integrals in a subset of spectra at the y-axis, but the ratio of integrals within the same subset. During our discussions with the collaborators,

this case proved to be non-intuitive. Therefore, it is best illustrated by Figure 5.6.

For a better illustration of how the subsets are created, we may also use the metaphor of “axis collapsing”. The data in both axes are “collapsed” to a single set, which is then subdivided into smaller sets, each representing a glyph shown in the nested structure. The ratio in each of these subsets is computed as the ratio of mean integrals in all the spectra in the subset. Therefore, if we explore the ratio of integrals in the same region (e.g., the ratio of integrals between chemical shift 1.5 to 2 ppm), the ratio value in all nested glyphs will necessarily be equal to 1, while the ratio in the overview heatmap cell may differ.

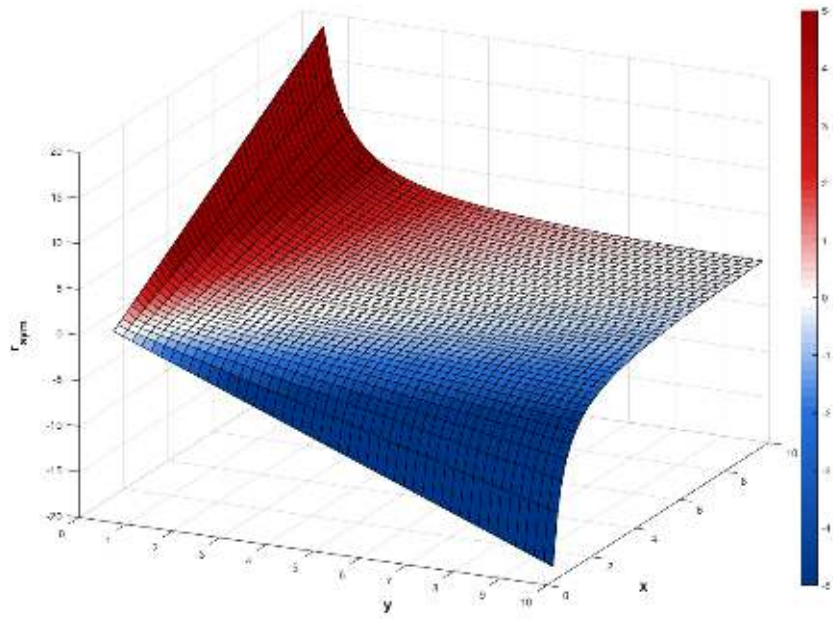


Figure 5.5: 3D plot of the transformed ratio function, which is more suitable to be mapped to a diverging color gradient. Here, the color mapping is illustrated by a red-blue gradient similar to the one used in our application for a positive ratio value. The cutoff values over and below which the color is no longer distinguished are set to -5 and 5. In the case where  $x$  or  $y$  is equal to zero, the color is set to gray.

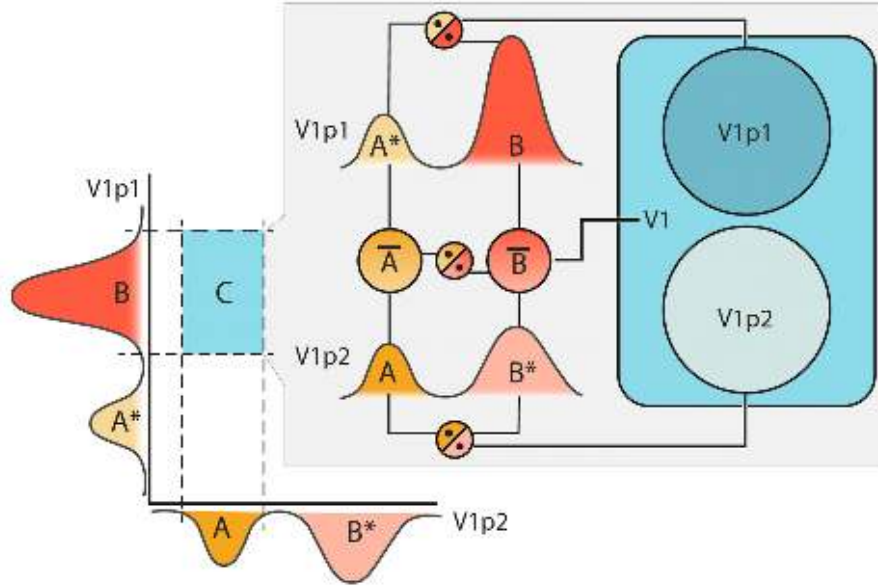


Figure 5.6: Illustration of computing all ratios in a simple dataset consisting of two voxels (created by Laura Garrison [1]): a voxel at a location  $V1$  of a patient  $p1$ , and a voxel at the same location  $V1$  of a patient  $p2$ . The user has chosen to explore the ratio of integrals of peak A to peak B. In the matrix cell of the overview heatmap, the ratio is simply the integral of peak A in  $V1p2$  divided by the integral of peak B in  $V1p1$ . Subsequently, there are three glyphs in the nested structure inside the matrix cell. The bottom-layer glyph denoted as  $V1$  represents all voxels located at  $V1$ . Therefore, its color maps to the ratio of the average of integrals of peak A in  $V1p1$  and  $V1p2$  to the average of integrals of peak B in  $V1p1$  and  $V1p2$ . In the top layer, there are two glyphs, representing voxels at the location  $V1$  of patients  $p1$  and  $p2$ . In our case, there is a single voxel in each of these subsets, so the color of the glyph denoted  $V1p1$  maps to the ratio of the integral of peak A in  $V1p1$  and the integral of peak B in  $V1p1$ .

## 6 Implementation

I have already outlined the specifics of MRS data analysis, described the design of the interface and visualization methods used in our application, and explained all underlying computations and processes. The remaining part is to address the implementation of the application itself, and the feedback we received from our collaborators during the brief user testing sessions.

For its description, I again need to cover the whole data analysis process, from loading the dataset to exploring the details in the matrix view, and then address the challenges I encountered during the implementation process. Before doing so, however, I would like to devote the following section to the structure of the source files and their assignment to the system components, as defined in Chapter 4 and shown in Figure 6.1.

### 6.1 Source Files

Since SpectraMosaic is a single-page application, there is only one main HTML file (`index.html`) that specifies the layout of the page and connects the JavaScript source files. Besides that, there is one more page (`help.html`) that serves as a manual for the application, created by Laura Garrison. This “help page” is always opened in another tab of the browser so that it does not interrupt the workflow in the main page. Associated to these pages, there are two stylesheets, which are used for formatting the elements of the two pages and are of little interest in the context of this thesis.

The key part are the scripts that implement all the computation, logic, and dynamic behavior of the application. These are assigned to the system components according to their role (see Figure 6.1). The one file that is not assigned to any component (`gridster_main.js`) serves just for creating the page layout (see the following section) and initializing some of the libraries and objects used. Now, I will discuss the implementation of the application, including its layout and system components.



Figure 6.1: Diagram of the system components and source files

### 6.2 Layout of the Page

To align the elements in the page in a suitable way, independent of the window size and stable when changing the size of the page, it is necessary to use a grid system. Initially, I used the grid system in the Bootstrap toolkit [49], which is the most commonly used library for handling front-end components. Later, we decided to use Gridster.js [50] for the placement of elements in the page while keeping the internal structure of these elements handled by Bootstrap. The reason for this change is the possibility of integrating SpectraMosaic into another project currently developed by Laura Garrison, which uses Gridster. This way, the main elements are compatible with the Gridster environment, allowing for a flexible integration into a more complex project, while keeping the original structure within the page elements as it was implemented before. The elements of the Gridster page structure are shown in Figure 6.2.

The original purpose of Gridster is to enable the page elements to be reordered in the page using drag-and-drop operations. This

behavior is not suitable for our application as we keep the page layout firmly given and already use drag-and-drop operations for moving data into the right panel. For this reason, it was necessary to disable this default behavior of Gridster and use it only in a static way.

Now, I will address the implementation of each of the system components, except for C2 (data operations), which is described in Chapter 5. I will begin with loading the data (C1) and continue with the description of the implementation of the two main panels (C3, C4).

47

### 6.3.1 Data Input

As I mentioned in Chapter 5, the intention was to make the data input as simple as possible for the user. More precisely, we wanted to be able to process the data exactly in the format provided by the preprocessing scripts using Tarquin, created by one of our collaborators. Therefore, I have decided that it is best to process the whole directory tree at once, and this can be achieved in JavaScript using the File and Directory Entries API [51]. The documentation page warns that this is a non-standard feature and should not be used on production sites, as it may not be supported by every browser and the behavior may change in the future [51].

In our case, SpectraMosaic is in a prototype stage and the future versions that could be considered a production site should ideally have the data preprocessing step integrated, rather than performed by a separate tool, so this type of file loading would no longer be necessary. So far, we have tested this feature in Firefox, Google Chrome, Safari, and Microsoft Edge in the implementation and testing phase, and encountered no issues with browser compatibility. However, with the latest update of the File and Directory Entries API (as of April 27, 2020 [52]), there were issues in compatibility with Google Chrome. After a consultation with the team at the University of Bergen, we have decided to address this issue in the next version of the application.

The issues I did encounter when implementing this feature were caused by the fact that this API, as well as other functions related to file processing, is asynchronous and uses callback functions. The problem with callbacks is the difficulty of managing their execution order [53], e.g., storing the voxel information only after all the files assigned to this voxel have been processed, and then normalizing data only after all voxels have been stored.

These issues have been solved by introducing *Promises*: objects serving as placeholders for a value returned by an asynchronous function, which can register callbacks that will run after this function succeeds or fails [53]. The main advantage is that it is also possible to store the Promise objects in an array and register a callback that will run only after all of these Promises are resolved. I use this approach as illustrated by the following code excerpt:



---

```

// chaining of loading voxel data and preprocessing
// taken from file_reader_master.js

var proms = []; // array of Promises for all voxels

// read all patient directories
foundFiles.forEach((patient) => {
  if (patient.isDirectory) {
    // read all voxel directories
    patient.contents.forEach((voxel) => {
      if (voxel.isDirectory) {
        proms.push( readVoxel( patient.name,
                               voxel.name,
                               voxel.contents ));
      }
    });
  }
});

// normalize after everything is processed
Promise.all(proms).then(() => {
  finishLoading();
});

```

The array *foundFiles* represents the contents of the root directory: the header file and patient directories (see Figure 5.1). The function *readVoxel* utilizes asynchronous functions to read the CSV and anatomical image files in the voxel directory. Therefore, it returns a Promise which is successfully resolved only after these two files are processed. The array *proms* contains Promises assigned to all the voxels currently being loaded, and the function *finishLoading* is executed after all of them are successfully resolved, i.e., all the files have been successfully read.

Another issue was caused by the fact that we need to display the anatomical image on a p5.js canvas (see the following subsection), therefore, it is necessary to store it as a p5 Image object. However, the p5 Image cannot be constructed from a JavaScript File object, into which the image file was loaded by the File and Directory Entries API. To solve this, it is necessary to convert the image to a *base64* string representation [54], which can be processed by the p5 *loadImage*

function [55]. The File object can be converted to a base64 string using the *FileReader.readAsURL* method [56].

### 6.3.2 Main Panels as p5.js Canvas

The first tool that usually comes into consideration when creating a web-based visualization interface is the D3 library [57]. While it is a suitable tool for most types of visualization, it did not prove to be optimal in our case. The main idea behind D3 is the manipulation of the standard document object model (DOM) by binding input data to the document elements [57].

In the earliest stages of our project, Laura Garrison sketched out the proposed interface and visual encoding on a blank page. The goal of this work was to transfer these sketches into a working application, so that the design choices could be properly validated. Using D3 to implement this design would need a substantially different approach to the visual elements: selecting from a set of standard visual encodings (although very broad) instead of drawing on a blank canvas. After an attempt to visualize the spectral curve using a parallel coordinate plot in D3, we have decided to use an approach more consistent with the initial “sketching”.

The idea of sketching was utilized by Casey Reas and Ben Fry in Processing [58], a programming language based on Java that is “built to act as a software sketchbook [58]” but started to be widely used in thousands of various projects [59]. Processing was later extended to p5.js [60], a web-based tool based on the same principles, which can be used to embed interactive graphics on a web page. This approach allows for simple and fast implementation of visual ideas, especially in projects that require the flexibility to make substantial changes in a short time [61]. Therefore, p5.js became our tool of choice for implementing the two main panels of the application’s interface.

The left and right panel (components C3 and C4) both contain a p5 object (sketch) nested in a *div* element in the corresponding Gridster cell. At the beginning, a blank canvas is created, and its size is given by the size of the Gridster cell. Normally, a p5 sketch behaves as an animation, updating its contents at a specified frame-rate. For a more static content, this behavior can be disabled, and the sketch is updated only when necessary. I have chosen the static approach and defined a

method *updateScene* for each sketch, that can be called whenever the contents need to change. These occasions include:

- change in the underlying data,
- interaction with the sketch,
- changing the browser window size.

Considering that any interaction with the sketch has already been handled and the properties of the view are set, the *updateScene* methods for the left and right panel are described by the UML activity diagrams in Figure 6.3 and Figure 6.4, respectively.

## 6.4 Interaction

In this section, I will describe how the interactions defined in Section 3.2 are handled, in the order of the workflow illustrated in Figure 4.2. The disadvantage of the sketching approach is that without standard visualization elements, I have to manually define even the trivial interactions, such as clicking on an element or hovering the mouse over it. I will mention the implementation of these basic interactions only in the first case, since the approach will be similar in other cases.

Handling of the interaction always takes place before the sketch is rendered. Properties of the view are updated according to the user's action and then the *updateScene* method is called. I will focus mainly on the description of these properties and the way they are computed or determined.

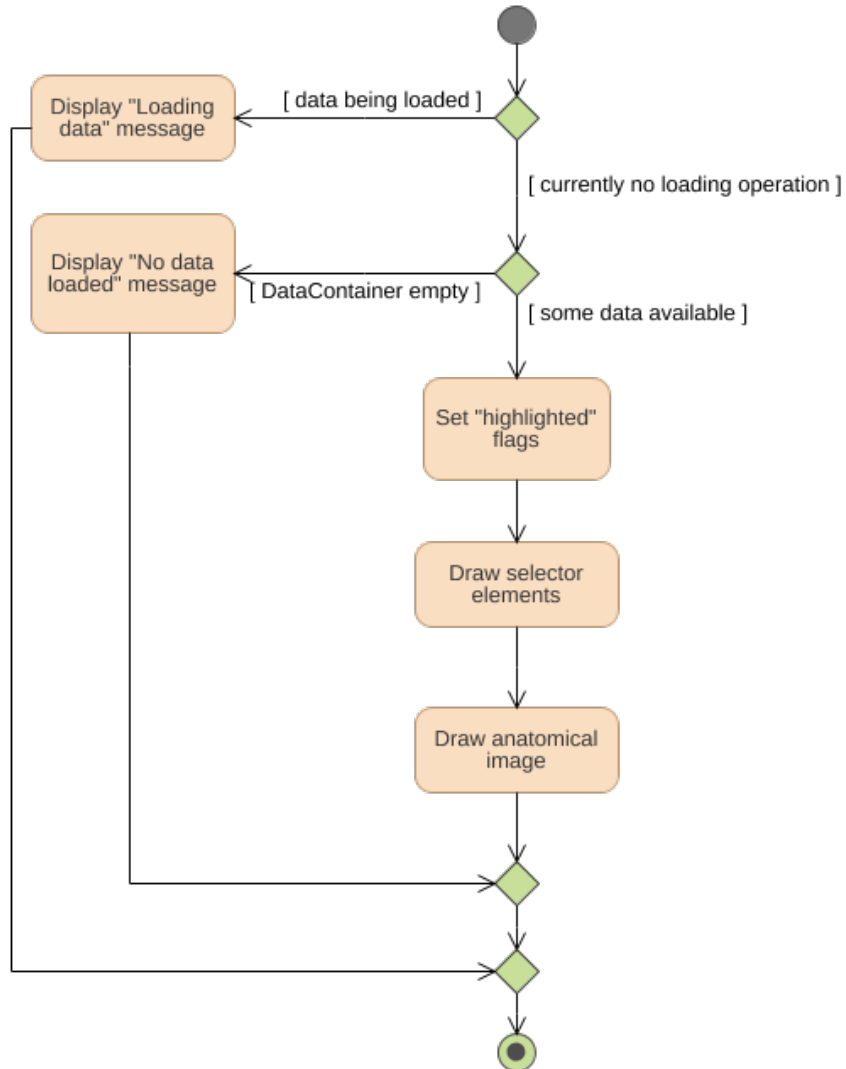


Figure 6.3: UML activity diagram for updating the left panel: If there is no ongoing loading process and there are data to be shown, we handle highlighting according to a message received from the right panel (more on highlighting in Section 6.4) and display the contents of this panel accordingly.

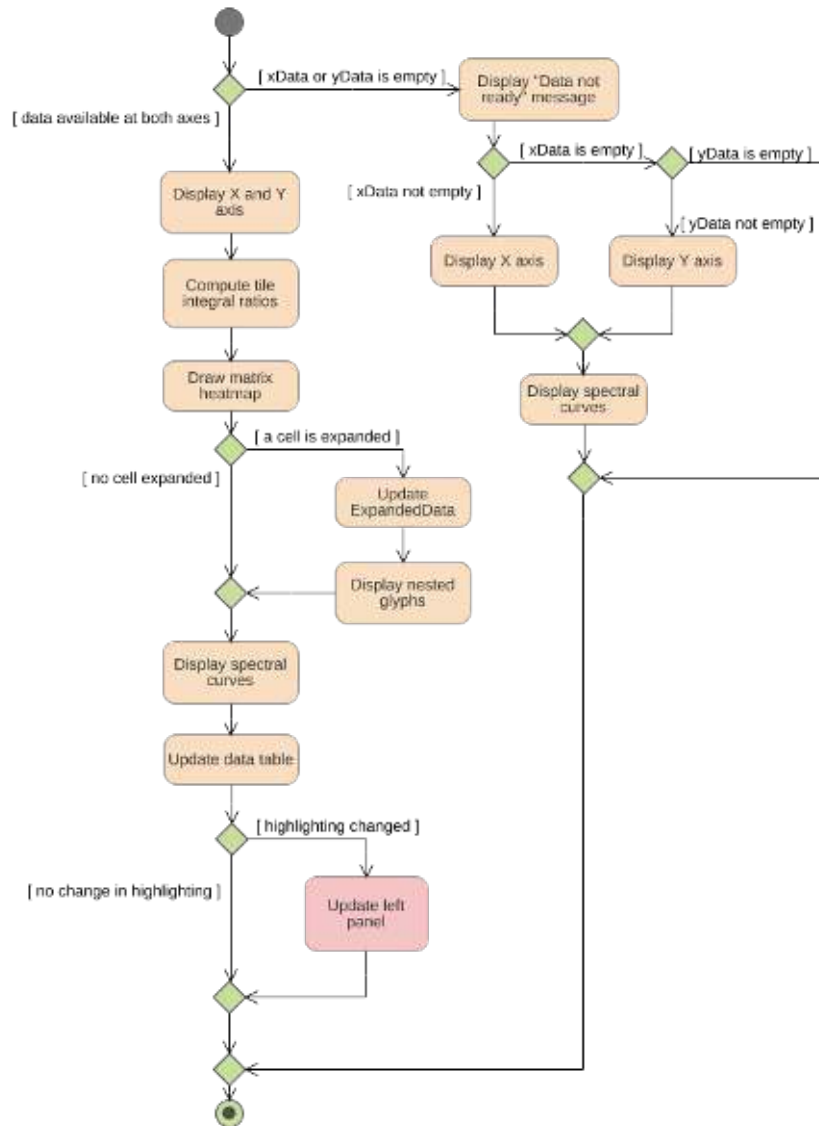


Figure 6.4: UML activity diagram for updating the right panel: If there are data associated only with one axis, we show the corresponding axis and spectral curve. If there are data at both axes, we compute integral ratios according to the current properties of the view (more on adjusting the matrix view in Section 6.4) and display the contents of this panel accordingly.

### 6.4.1 Selecting a Voxel for the Analysis

Two types of interaction in the left view are necessary for selecting a voxel for the analysis: exploring the metadata and viewing the anatomical image. Both of these are provided by the selector elements for the patient, time of acquisition, brain state, and voxels in the acquisition (see Chapter 3 for details). The elements and image currently displayed are determined by the properties of the view. For clarity, I show once again the layout of the left panel in Figure 6.5.

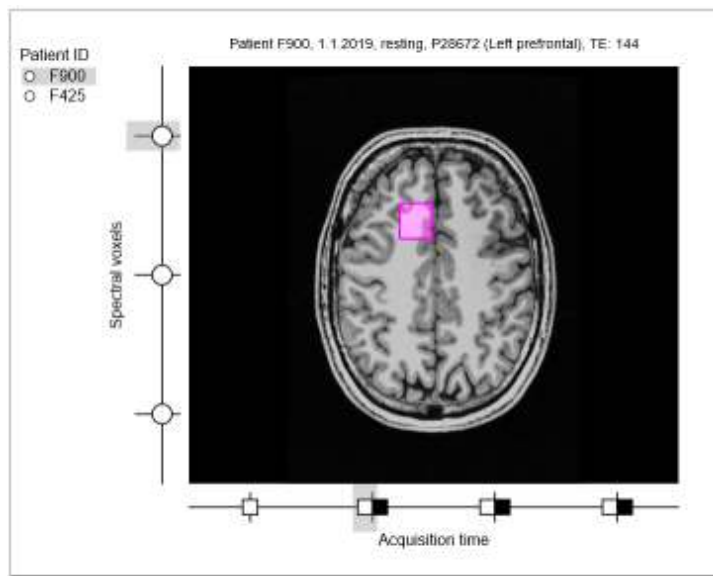


Figure 6.5: Layout of the left panel

Each of the selectors has three associated properties: *chosen*, *displayed*, and *active*. To provide an example, let us consider the time selector. Each timepoint is represented by a square or a pair of squares (depending on whether both brain states are available) on the horizontal selector. The *chosen\_timepoint* property stores the index of the timepoint that is currently selected by the user. *Displayed\_timepoint* determines the index of the timepoint from which the voxels on the vertical selector are taken. Normally, the *displayed* and *chosen* properties are equivalent (in Figure 6.5, *displayed\_timepoint* is set to 1). In the case where a voxel from another acquisition is highlighted, we

need to temporarily show this voxel instead of the currently selected one. Therefore, the *displayed\_timepoint* is changed to the index of the desired acquisition time. Once there are no more highlighted items, it is set back to the value of *chosen\_timepoint*. The *active* property represents the index of the element the mouse is currently hovering over (and is set to -1 otherwise). After clicking the mouse button, the *chosen* property is set to the value of *active*, if it is other than -1.

Moreover, the *active* property is used to show the metadata associated with the given selector element. The metadata of an acquisition (time and brain state) are displayed as a tooltip next to the cursor. The metadata of the voxel, connected with the patient information, are displayed as a heading above the anatomical image permanently, determined by *displayed\_voxel* when *active\_voxel* is unavailable. The anatomical image displayed is determined in the same way.

For dragging a voxel over to the right panel, it is possible to use either the voxel selector elements (circles on the vertical selector) or the square highlighting the voxel's location on the anatomical image. Whether the mouse is hovering over the square or another part of the anatomical image is determined simply by the color of the pixel at the position of the mouse (the voxel's location is set to be shown in fuchsia in the preprocessing stage). To determine which voxel to drag over, the *active\_voxel* property is used.

#### 6.4.2 Adjusting the Matrix View

Interactions in the right panel are linked to the adjustment of the matrix cells and displaying the nested glyphs inside a cell. The layout of the right panel with an expanded cell is shown in Figure 6.6.

At the start, all the 21 ticks of both axes, separating the tiles, are set to be equally distanced. In order to be able to align a tile with a peak of the spectral curve, it is necessary to move the ticks along the axis. In addition, when expanding a cell to display the nested glyphs, it is necessary to move the tick on the sketch but keep its position with respect to the chemical shift. To achieve this, these properties are associated with each axis:

*Relative tick positions*: positions of the ticks with respect to the chemical shift, where 0 represents the beginning of the axis (4 ppm)

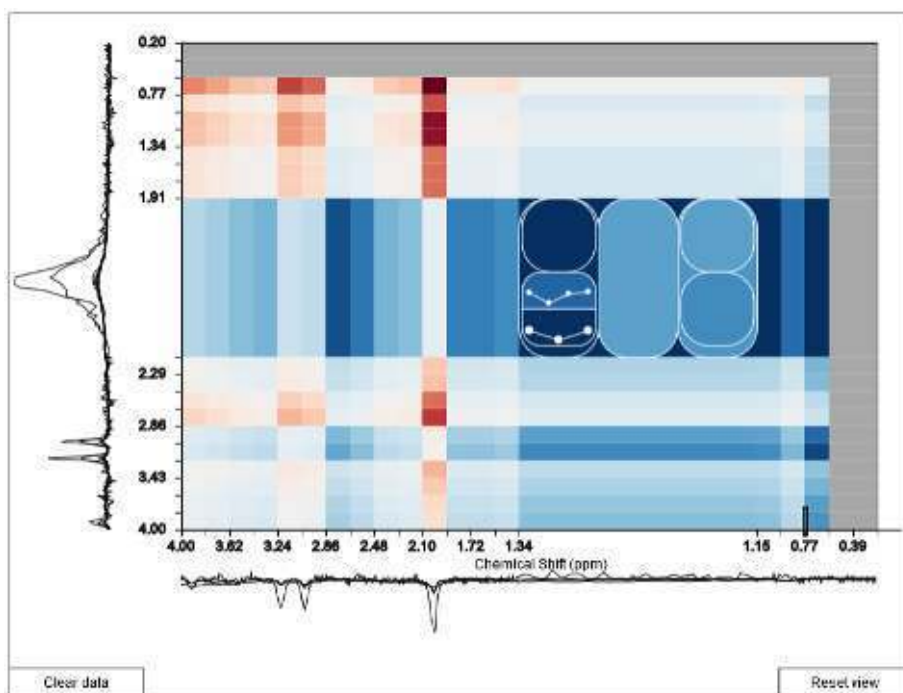


Figure 6.6: Layout of the right panel

and 1 represents the end of the axis (0 ppm). This value changes when a tick is moved along the axis.

*Tile scales*: magnification coefficients of the tiles used for cell expansion, where the value of 1 means no change of size. This property is changed when one cell is expanded and the others shrink accordingly.

It is possible to imagine the cell expansion as a zoom lens. Initially, all the scales are set to 1. When a cell is expanded, the scale of the corresponding tiles is set proportional to their current size, so that they take up at least  $\frac{1}{4}$  of the axis' length. Denoting by  $r_k$  the relative position of the  $k$ -th tick, by  $axis\_len$  the length of the axis in pixels and by  $i$  the index of the tile to be enlarged, the scale  $s_i$  is defined as follows:



$$area\_exp = \begin{cases} 0.25 \cdot (r_{i+1} - r_i) + 0.25 & \text{if } r_{i+1} - r_i < 0.5 \\ r_{i+1} - r_i & \text{otherwise} \end{cases}$$

$$s_i = \frac{axis\_len \cdot area\_exp}{axis\_len \cdot (r_{i+1} - r_i) - 1}$$

This way, I can ensure that no cell will be too small for the nested glyphs to be distinguishable but the cells that already take at least half of the heatmap's area will not get larger. Similarly, I can define the scale of the remaining 19 tiles  $s_j$  (where  $j \neq i$ ) as:

$$area\_shrink = \frac{1 - area\_exp}{19}$$

$$s_j = \frac{axis\_len \cdot area\_shrink}{axis\_len \cdot (r_{j+1} - r_j) - 1}$$

*Absolute tick positions:* positions of the ticks on the axis in pixels. These are used when drawing the heatmap matrix to determine the positions of the cells. We can compute the absolute position of the  $i$ -th tick  $a_i$  as follows:

$$\begin{aligned} a_0 &= 0 \\ a_i &= a_{i-1} + axis\_len \cdot (r_i - r_{i-1}) \cdot s_{i-1} \quad i > 0 \end{aligned}$$

The absolute positions are computed and stored when the axis is displayed. Therefore, the X and Y axes must always be the first elements that are drawn when the sketch is updated, as shown in Figure 6.4.

### 6.4.3 Highlighting System

When the user is hovering the mouse over a nested glyph in the expanded cell, we want elements corresponding to this glyph to be highlighted, as illustrated in Figure 3.5. The glyphs are drawn using values

## 6. IMPLEMENTATION

---

stored in *ExpandedData*. However, the elements to be highlighted are based on *MatrixData* or the object hierarchy in *DataContainer*. The key problem with implementing the highlighting system was to connect these data structures.

As a solution, I propose to use a string identifier of a voxel constructed according to the hierarchy of the nested glyph structure: *location\_patientID\_brainstate\_time\_voxelID*. The disadvantage of this approach is the fact that the metadata cannot contain the underscore character. However, this can be easily ensured when reading the header file. Using the string identifier, the subset of voxels to be highlighted can be easily determined simultaneously as the nested glyph structure is displayed.

When drawing the glyph element, its absolute position on the sketch is determined. Using this position, it is then trivial to find out whether the mouse cursor is hovering over it. If so, the only other glyphs that the mouse can possibly be hovering over are the ones nested in the glyph currently being drawn. For instance, when the mouse hovers over a glyph representing the patient P900 nested in the glyph representing voxels located in the left prefrontal area, the voxels in the corresponding subset will be represented by identifiers starting with “left prefrontal\_P900”.

Given the array of identifiers *ids\_to\_highlight* and supposing mouse interaction has been detected over the glyph representing the patient *pt* and voxels in the location *l*, the array can be simply reduced to the desired subset of voxels using the JavaScript Array method *filter* [62]:

```
ids_to_highlight = ids_to_highlight.filter(function(id){
    var ref_id = expanded_data[l].voxel_loc + "_" +
                expanded_data[l].patients[pt].patient_id;
    return id.startsWith(ref_id);
});
```

When such filtering is performed at every level of the nested glyph structure where mouse interaction was detected, we are left with the subset of voxels that will be highlighted. It is then simple to set the *highlighted* property of the corresponding voxels in *MatrixData*, based on their voxel ID (the last part of the identifier string). Using this property, the corresponding rows in the data table and the corresponding spectral curves in the matrix view are highlighted.

To set the properties inside `DataContainer` (see Figure 5.2), it is also beneficial to use the string identifier. For an illustration, let us consider the case where we highlight the voxels in the location *V1* of patients *p1* and *p2*, i.e., the mouse is hovering over the top-level glyph in Figure 5.6, but not over the glyphs nested inside of it. It is not possible to show data of two patients at once in the left panel, so the *displayed\_patient* property would be set to one of the two patients and the other's selector element will be highlighted.

Using the string identifier, it is simple to determine which patient will be displayed. Having the array of identifiers, a list of patient IDs, whose voxels are highlighted, is easily obtained. If the ID of the patient that is currently displayed is found inside the list, the *displayed\_patient* remains unchanged. Otherwise, it is changed to the first patient in the list. For all the other Patient objects, the *highlighted* property is set to *true* if their ID is found in the list.

Then, we can construct a list of acquisitions (identified by the time and brain state) of the displayed patient, containing highlighted voxels. In a similar way, the *displayed\_timepoint* and *displayed\_state* are determined, and the *highlighted* property of the remaining BrainState objects corresponding to the list is set as *true*.

Finally, we obtain a list of voxel IDs of the given patient (*displayed\_patient*) during the given acquisition (*displayed\_timepoint*, *displayed\_state*) that are highlighted, determine the *displayed\_voxel*, and set the *highlighted* property as *true* for all the other Voxel objects. Note that with the *displayed\_voxel* property, the anatomical image will change as well.



## 7 Conclusion and Future Work

SpectraMosaic is currently a fully functional prototype that allows for the exploration of variance of metabolite concentration ratios in data acquired by magnetic resonance spectroscopy (MRS). My work, focused on the implementation of this application, took place mainly during my Erasmus exchange semester at the University of Bergen as a part of the research project led by Laura Garrison [1], in cooperation with Mohn Medical Imaging and Visualization Centre in Bergen.

### 7.1 Summary

In Chapter 1, I have described the principles of magnetic resonance spectroscopy, the standard ways of processing spectroscopic data and the tasks our application aims to support. These tasks are centered around the exploratory analysis of metabolite concentration ratios in MRS data, and the subsequent tracking of the variance of these ratios across time, various locations in the brain, varying state of brain activity or across different patients. Subsequently, I described the requirements for our application, discussed the related work relevant to our project in Chapter 2, and came to the conclusion that no existing tool that we are aware of could fully support the tasks specified in this project, but many of the discussed work can serve as a source of inspiration for our project. These include not only previous work related to the spectroscopic data, but also other work in the realms of heterogeneous data visualization and unit and tabular visualization.

Next, in Chapter 3, I described the interface and visualization design provided mainly by Laura Garrison, and addressed the changes that were made during the implementation process. These changes were only minor as the functional prototype served mainly for the evaluation of the design choices and as a foundation for further development. In Chapter 4, I provided the rationale for the choice of a single-page web-based application as the platform suitable for this application. The web-based solution was requested as it does not require the installation of any additional software, and I have chosen the single-page approach as it is efficient in terms of network usage and response time for smaller-scale projects as this one.

Then, I have divided the application's functionality with respect to the implementation into four system components, which served as a framework for the description of the underlying data processes and computations in Chapter 5. These include the normalization of the data values so that they can be efficiently displayed and remain mutually comparable, computation of integrals of the spectral curves and their transformed ratio value that can be simply mapped to a symmetric diverging color gradient. Finally, in Chapter 6, I have addressed the specifics of the implementation of this application. I have discussed the choice of p5.js as the suitable tool for displaying the main interface components, as it is based on the idea of "sketching", which is consistent with the design process in this project.

The specifications of the tasks and requirements, along with the discussion of some of the related work, the proposed design, and the results of the evaluation of the implemented prototype (addressed here in Section 7.2) were included in the design study, presented at the 9th EG Workshop on Visual Computing for Biology and Medicine in September 2019 in Brno [1].

### 7.2 Evaluation and Future Directions

The final prototype was evaluated in two case studies conducted by Laura Garrison, using a dataset from the area of neuroinflammation research. Three volunteer participants from the hospital research environment took part, provided feedback and suggested possible improvements and additional features [1].

The feedback we received was mostly positive, highlighting the comfortable use of the drag-and-drop features, the aligned display of metabolite ratios in the matrix view, and generally stating that this application would augment their workflow, especially in the case of group comparison [1]. As a possible improvement, there was a suggestion to allow selecting between the axial, coronal, and sagittal anatomical image (see Chapter 5) to be shown in the left panel. Every user also indicated an interest in the possibility of exporting the metabolite ratios displayed in the heatmap into a CSV file that would be used for a subsequent statistical analysis [1].

A drawback of the current approach of quantifying metabolite concentrations as peak integrals is the fact that we are not able to properly quantify the concentration of a metabolite that is represented by multiple peaks [1]. Given that the standard software for processing MR spectra, such as LCModel or Tarquin, is capable of such quantification [14, 16], it would be beneficial to make the quantification of metabolite concentrations a step in the preprocessing stage, and use the resulting values instead of computing the peak integrals. However, the possibility to see the spectral curves was mentioned as important [1], and by simply replacing the spectral curves with quantified metabolite ratios, this possibility would be lost, so the improvement would require a more substantial change in the design.

Another drawback lies in the fixed limit values of the color mapping scale. So far, the diverging color mapping is effective for exploring larger differences, but even slight variations, represented by ratios close to 1, are meaningful to explore in certain cases [1]. Therefore, a user-defined color mapping system would be beneficial.

In addition, a suggestion was made that this tool would be useful for the visualization of large cohort data. For this purpose, it would be necessary to divide the data into groups, and then treat the groups in a similar way individual voxels are treated at the moment [1]. This provides the opportunity for a feature that would allow the user to create some groups manually, while other would be created automatically based on the associated metadata.

Finally, once the application is ready to be used in clinical practice or hospital research, it would be beneficial to load the data from a local server, instead of having to store them in a firmly given directory structure and then load them manually. This way, the privacy requirements would still be met, but the application would be well integrated into a set of tools available in the hospital environment.

### 7.3 Conclusion

SpectraMosaic proved to be a tool that is capable of augmenting the workflow of researchers working with magnetic resonance spectroscopy data. It is efficient for displaying the variance of metabolite concentration ratios in the spectra, and fully supports the tasks of the exploratory analysis. Within the scope of this thesis, I was involved in the iterative development and implementation of this application. The resulting prototype was fully functional, proved to be useful for testing the efficiency of the design choices, and is a good foundation for future development and a continued work on this project.



## Bibliography

1. GARRISON, Laura; VAŠIČEK, Jakub; GRÜNER, Renate; SMIT, Noeska; BRUCKNER, Stefan. SpectraMosaic: An Exploratory Tool for the Interactive Visual Analysis of Magnetic Resonance Spectroscopy Data. In: *Proceedings of 9th EG Workshop on Visual Computing for Biology and Medicine (VCBM)*. 2019.
2. CASTILLO, Mauricio; KWOCK, Lester; MUKHERJI, Suresh K. Clinical applications of proton MR spectroscopy. *American journal of neuroradiology*. 1996, vol. 17, no. 1, pp. 1–15.
3. YOUSEM, David M; GROSSMAN, Robert I. *Neuroradiology: the requisites*. 4th ed. Elsevier Health Sciences, 2017.
4. GRAAF, Marinette van der. In vivo magnetic resonance spectroscopy: basic methodology and clinical applications. *European Biophysics Journal*. 2010, vol. 39, no. 4, pp. 527–540.
5. FRIEBOLIN, Horst; BECCONSALL, Jack K. *Basic one-and two-dimensional NMR spectroscopy*. 3rd ed. VCH Weinheim, 1998.
6. MACOMBER, Roger S. *A complete introduction to modern NMR spectroscopy*. Wiley New York, 1998.
7. PREIM, Bernhard; BOTHA, Charl P. *Visual computing for medicine: theory, algorithms, and applications*. Newnes, 2013.
8. LINDON, John C; TRANTER, George E; KOPPENAAL, David. *Encyclopedia of spectroscopy and spectrometry*. Academic Press, 2016.
9. *Single Voxel Spectroscopy* [online]. Erlangen: Siemens Healthineers Global, 2020 [visited on 2020-03-27]. Available from: <https://www.siemens-healthineers.com/magnetic-resonance-imaging/options-and-upgrades/clinical-applications/single-voxel-spectroscopy>.
10. RICHARDS, Todd L et al. Effects of a phonologically driven treatment for dyslexia on lactate levels measured by proton MR spectroscopic imaging. *American Journal of Neuroradiology*. 2000, vol. 21, no. 5, pp. 916–922.

## BIBLIOGRAPHY

---

11. DER GRAAF, Marinette van; HEERSCHAP, Arend, et al. Common processing of in vivo MR spectra. *NMR in biomedicine*. 2001, vol. 14, no. 4, pp. 224.
12. VAN DER GRAAFF, MM et al. MR spectroscopy findings in early stages of motor neuron disease. *American journal of neuroradiology*. 2010, vol. 31, no. 10, pp. 1799–1806.
13. BREHMER, Matthew; MUNZNER, Tamara. A multi-level typology of abstract visualization tasks. *IEEE transactions on visualization and computer graphics*. 2013, vol. 19, no. 12, pp. 2376–2385.
14. PROVENCHER, Stephen W. Estimation of metabolite concentrations from localized in vivo proton NMR spectra. *Magnetic resonance in medicine*. 1993, vol. 30, no. 6, pp. 672–679.
15. WAGENINGEN, Heidi van; JØRGENSEN, Hugo A; SPECHT, Karsten; HUGDAHL, Kenneth. A <sup>1</sup>H-MR spectroscopy study of changes in glutamate and glutamine (Glx) concentrations in frontal spectra after administration of memantine. *Cerebral Cortex*. 2010, vol. 20, no. 4, pp. 798–803.
16. WILSON, Martin; REYNOLDS, Greg; KAUPPINEN, Risto A; ARVANITIS, Theodoros N; PEET, Andrew C. A constrained least-squares approach to the automated quantitation of in vivo <sup>1</sup>H magnetic resonance spectroscopy data. *Magnetic resonance in medicine*. 2011, vol. 65, no. 1, pp. 1–12.
17. MULLINS, Paul G; MCGONIGLE, David J; O’GORMAN, Ruth L; PUTS, Nicolaas AJ; VIDYASAGAR, Rishma; EVANS, C John; EDDEN, Richard AE, et al. Current practice in the use of MEGA-PRESS spectroscopy for the detection of GABA. *Neuroimage*. 2014, vol. 86, pp. 43–52.
18. PURVIS, Lucian AB; CLARKE, William T; BIASIOLLI, Luca; VALKOVIČ, Ladislav; ROBSON, Matthew D; RODGERS, Christopher T. OXSA: An open-source magnetic resonance spectroscopy analysis toolbox in MATLAB. *PloS one*. 2017, vol. 12, no. 9.
19. STEFAN, DDCF et al. Quantitation of magnetic resonance spectroscopy signals: the jMRUI software package. *Measurement Science and Technology*. 2009, vol. 20, no. 10, pp. 104035.

## BIBLIOGRAPHY

20. CRANE, Jason C; OLSON, Marram P; NELSON, Sarah J. SIVIC: open-source, standards-based software for DICOM MR spectroscopy workflows. *International journal of biomedical imaging*. 2013, vol. 2013.
21. FENG, David; LEE, Yueh; KWOCK, Lester; TAYLOR, Russell M. Evaluation of glyph-based multivariate scalar volume visualization techniques. In: *Proceedings of the 6th Symposium on Applied Perception in Graphics and Visualization*. 2009, pp. 61–68.
22. FENG, David; KWOCK, Lester; LEE, Yueh; II, Russell M. Taylor. Linked exploratory visualizations for uncertain MR spectroscopy data. In: PARK, Jinah; HAO, Ming C.; WONG, Pak Chung; CHEN, Chaomei (eds.). *Visualization and Data Analysis 2010*. SPIE, 2010, vol. 7530, pp. 33–44. Available from DOI: 10.1117/12.839818.
23. NUNES, Miguel; ROWLAND, Benjamin; SCHLACHTER, Matthias; KEN, Soléakhéna; MATKOVIC, Kresimir; LAPRIE, Anne; BÜHLER, Katja. An integrated visual analysis system for fusing MR spectroscopy and multi-modal radiology imaging. In: *2014 IEEE conference on visual analytics science and technology (VAST)*. 2014, pp. 53–62.
24. MATKOVIC, Kresimir; FREILER, Wolfgang; GRACANIN, Denis; HAUSER, Helwig. Comvis: A coordinated multiple views system for prototyping new visualization technology. In: *2008 12th international conference information visualisation*. 2008, pp. 215–220.
25. WOLF, Ivo; VETTER, Marcus; WEGNER, Ingmar; NOLDEN, Marco; BOTTFGER, Thomas; HASTENTEUFEL, Mark; SCHOBINGER, Max; KUNERT, Tobias; MEINZER, Hans-Peter. The medical imaging interaction toolkit (MITK): a toolkit facilitating the creation of interactive software by extending VTK and ITK. In: *Medical Imaging 2004: Visualization, Image-Guided Procedures, and Display*. 2004, vol. 5367, pp. 16–27.
26. MARINO, Joseph; KAUFMAN, Arie. Prostate cancer visualization from MR imagery and MR spectroscopy. In: *Computer Graphics Forum*. 2011, vol. 30, pp. 1051–1060. No. 3.

## BIBLIOGRAPHY

---

27. AMIRKHANOV, Artem; FRÖHLER, Bernhard; KASTNER, Johann; GRÖLLER, Eduard; HEINZL, Christoph. InSpectr: Multi-Modal Exploration, Visualization, and Analysis of Spectral Data. In: *Computer Graphics Forum*. 2014, vol. 33, pp. 91–100. No. 3.
28. BRUCKNER, Stefan; MÖLLER, Torsten. Isosurface similarity maps. In: *Computer Graphics Forum*. 2010, vol. 29, pp. 773–782. No. 3.
29. MEYER, Miriah; WONG, Bang; STYCZYNSKI, Mark; MUNZNER, Tamara; PFISTER, Hanspeter. Pathline: A tool for comparative functional genomics. In: *Computer Graphics Forum*. 2010, vol. 29, pp. 1043–1052. No. 3.
30. MEYER, Miriah; MUNZNER, Tamara; DEPACE, Angela; PFISTER, Hanspeter. MulteeSum: a tool for comparative spatial and temporal gene expression data. *IEEE transactions on visualization and computer graphics*. 2010, vol. 16, no. 6, pp. 908–917.
31. STOPPEL, Sergej; HODNELAND, Erlend; HAUSER, Helwig; BRUCKNER, Stefan. Graxels: Information Rich Primitives for the Visualization of Time-Dependent Spatial Data. In: *Proceedings of VCBM 2016*. Bergen, Norway, 2016, pp. 183–192. Available from DOI: 10.2312/vcbm.20161286.
32. PARK, Deokgun; DRUCKER, Steven M; FERNANDEZ, Roland; ELMQVIST, Niklas. Atom: A grammar for unit visualizations. *IEEE transactions on visualization and computer graphics*. 2017, vol. 24, no. 12, pp. 3032–3043.
33. NEURATH, Otto. *International Picture Language. The First Rules of Isotype: With Isotype Pictures*. Kegan Paul & Company, 1936.
34. BERTIN, Jacques. *Semiology of graphics; diagrams networks maps*. University of Wisconsin Press, 1983.
35. PERIN, Charles; DRAGICEVIC, Pierre; FEKETE, Jean-Daniel. Revisiting bertin matrices: New interactions for crafting tabular visualizations. *IEEE transactions on visualization and computer graphics*. 2014, vol. 20, no. 12, pp. 2082–2091.

36. RAO, Ramana; CARD, Stuart K. The table lens: merging graphical and symbolic representations in an interactive focus+ context visualization for tabular information. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1994, pp. 318–322.
37. ROBERTS, J.C.; HEADLEAND, C.; RITSOS, P.D. Sketching Designs Using the Five Design-Sheet Methodology. *Visualization and Computer Graphics, IEEE Transactions on*. 2015, vol. PP, no. 99, pp. 1–1. ISSN 1077-2626. Available from DOI: 10.1109/TVCG.2015.2467271.
38. KINDLMANN, Gordon; SCHEIDEGGER, Carlos. An algebraic process for visualization design. *IEEE transactions on visualization and computer graphics*. 2014, vol. 20, no. 12, pp. 2181–2190.
39. POSSE, Stefan; OTAZO, Ricardo; DAGER, Stephen R; ALGER, Jeffry. MR spectroscopic imaging: principles and recent advances. *Journal of Magnetic Resonance Imaging*. 2013, vol. 37, no. 6, pp. 1301–1325.
40. GARRISON, Laura; VAŠIČEK, Jakub; GRÜNER, Renate; SMIT, Noeska; BRUCKNER, Stefan. *Proceedings of the EuroVis Conference - Posters (EuroVis '19)*. A Visual Encoding System for Comparative Exploration of Magnetic Resonance Spectroscopy Data [Poster presented at the EuroVis conference 2019]. 2019.
41. SHNEIDERMAN, Ben. The eyes have it: A task by data type taxonomy for information visualizations. In: *Proceedings 1996 IEEE symposium on visual languages*. 1996, pp. 336–343.
42. RICCA, F.; TONELLA, P. Analysis and testing of Web applications. In: *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*. 2001, pp. 25–34.
43. MIKOWSKI, Michael; POWELL, Josh. *Single page web applications: JavaScript end-to-end*. Manning Publications Co., 2013.
44. VORA, Pawan. *Web application design patterns*. Morgan Kaufmann, 2009.

## BIBLIOGRAPHY

---

45. GANGLBERGER, Florian; SWOBODA, Nicolas; FRAUENSTEIN, Lisa; KACZANOWSKA, Joanna; HAUBENSAK, Wulf; BÜHLER, Katja. BrainTrawler: A visual analytics framework for iterative exploration of heterogeneous big brain data. *Computers & Graphics*. 2019, vol. 82, pp. 304–320.
46. SCHLACHTER, Matthias; RAIDOU, Renata Georgia; MUREN, Ludvig P; PREIM, Bernhard; PUTORA, Paul Martin; BÜHLER, Katja. State-of-the-Art Report: Visual Computing in Radiation Therapy Planning. In: *Computer Graphics Forum*. 2019, vol. 38, pp. 753–779. No. 3.
47. JOURDAIN, Sebastien; AYACHIT, Utkarsh; GEVECI, Berk. ParaViewWeb: A Web Framework for 3D Visualization and Data Processing. *IADIS international conference on web virtual reality and three-dimensional worlds*. 2010, vol. 7.
48. CROW, Franklin C. Summed-area tables for texture mapping. In: *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. 1984, pp. 207–212.
49. *Grid system* [online]. Bootstrap, 2020 [visited on 2020-05-12]. Available from: <https://getbootstrap.com/docs/4.0/layout/grid/>.
50. *gridster.js* [online]. Ducksboard, 2020 [visited on 2020-05-12]. Available from: <http://dsmorse.github.io/gridster.js/>.
51. *File and Directory Entries API* [online]. Mountain View, CA: MDN Web Docs, 2013–2019 [visited on 2020-05-10]. Available from: [https://developer.mozilla.org/en-US/docs/Web/API/File\\_and\\_Directory\\_Entries\\_API](https://developer.mozilla.org/en-US/docs/Web/API/File_and_Directory_Entries_API).
52. *File and Directory Entries API* [online]. W3C Community Group, 2020 [visited on 2020-05-15]. Available from: <https://wicg.github.io/entries-api/>.
53. PARKER, Daniel. *JavaScript with Promises: Managing Asynchronous Code*. "O'Reilly Media, Inc.", 2015.
54. JOSEFSSON, Simon et al. *The base16, base32, and base64 data encodings*. 2006. Technical report. RFC 4648, October.
55. *p5.js reference* [online]. Processing Foundation, 2020 [visited on 2020-05-11]. Available from: <https://p5js.org/reference/>.

## BIBLIOGRAPHY

---

56. *FileReader* [online]. Mountain View, CA: MDN Web Docs, 2013–2019 [visited on 2020-05-11]. Available from: <https://developer.mozilla.org/en-US/docs/Web/API/FileReader>.
57. BOSTOCK, Michael; OGIEVETSKY, Vadim; HEER, Jeffrey. D<sup>3</sup> data-driven documents. *IEEE transactions on visualization and computer graphics*. 2011, vol. 17, no. 12, pp. 2301–2309.
58. REAS, Casey; FRY, Ben. *Processing: a programming handbook for visual designers and artists*. Mit Press, 2007.
59. FRY, Ben. *Visualizing data: Exploring and explaining data with the processing environment*. "O'Reilly Media, Inc.", 2008.
60. MCCARTHY, Lauren; REAS, Casey; FRY, Ben. *Getting Started with P5.js: Making Interactive Graphics in JavaScript and Processing*. Maker Media, Inc., 2015.
61. GROSS, Benedikt; BOHNACKER, Hartmut; LAUB, Julia; LAZZERONI, Claudius. *Generative Design: Visualize, Program, and Create with JavaScript in p5.js*. Chronicle Books, 2018.
62. *Array* [online]. Mountain View, CA: MDN Web Docs, 2013–2020 [visited on 2020-05-13]. Available from: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array).





## **A Low-fidelity Prototype: Design Sheets**

Below are the five sketches drawn by Laura Garrison during the low-fidelity prototyping phase, created using the Five Design-Sheets methodology [37].

# 1. Ideas

single pixel  
needs 4 responses  
from 4 different  
metabolites  
Viz per voxel (for 4 metabolites)

color heatmap

1H or 31P curve

linked graph visual  
voxel spectra graphs

columns = traditional metabolites

user draws peak segment of interest

for 1H  
time, space, individual data  
+ full rest

# 2. Filter

select discrete spectra to compare + visualize

1H or 31P curve

Metabolite ratio heatmap

# 3. Categorize

Blocks/matrices

lines/more interaction

# 4. Combine and Refine

Atlas selector image

user overlay with ratio mapped to atlas image

single stage used for a cohort

Layered full spectrum

# 5. Question

Do these show MRS data in a novel way?

Is the raw spectrum visible in end visualization?

Can we explore relationships between metabolites?

Can we look at metabolites in context of time/space/population?

74

## A. LOW-FIDELITY PROTOTYPE: DESIGN SHEETS

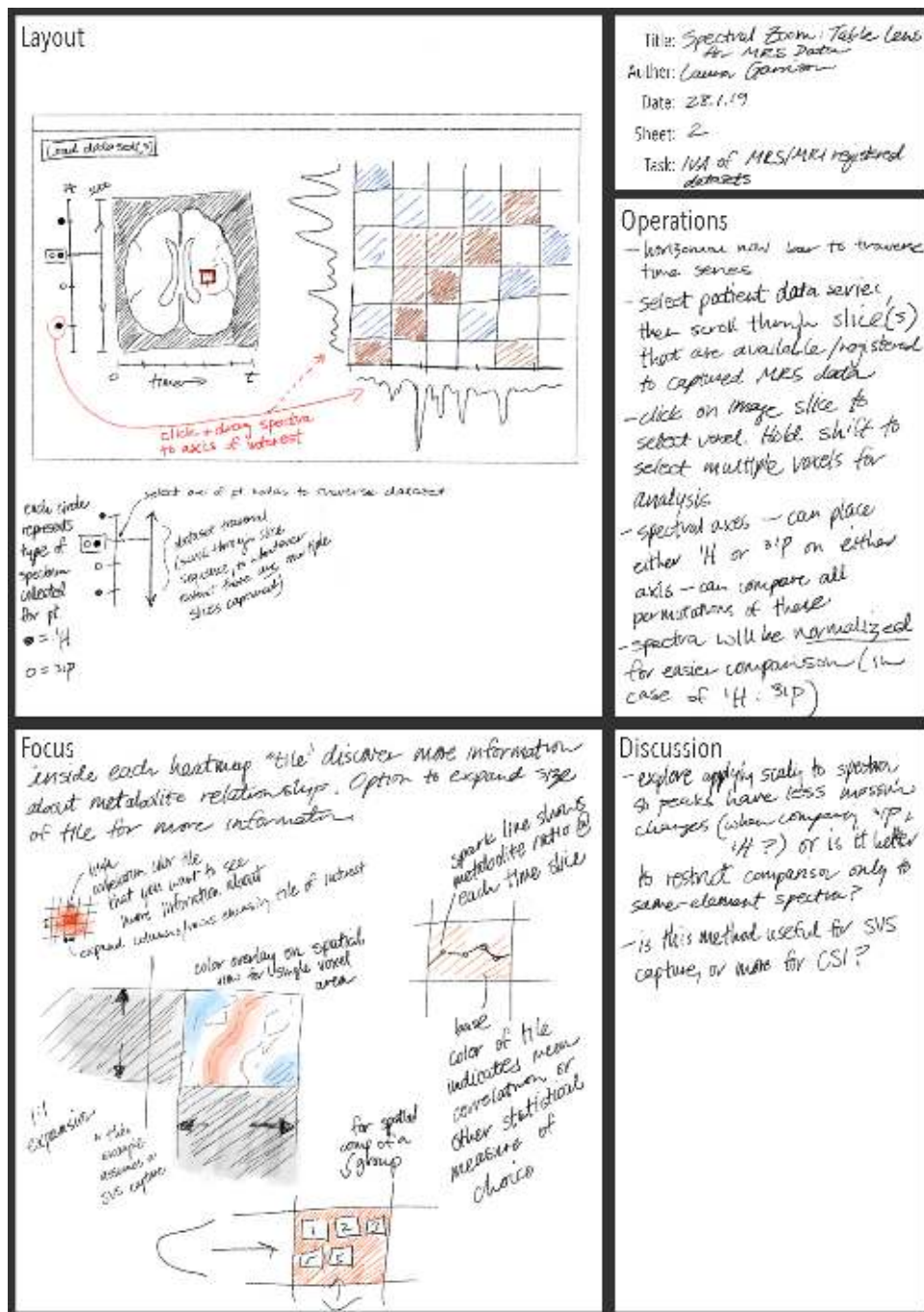


Figure A.2: Design sheet no. 2

## A. LOW-FIDELITY PROTOTYPE: DESIGN SHEETS

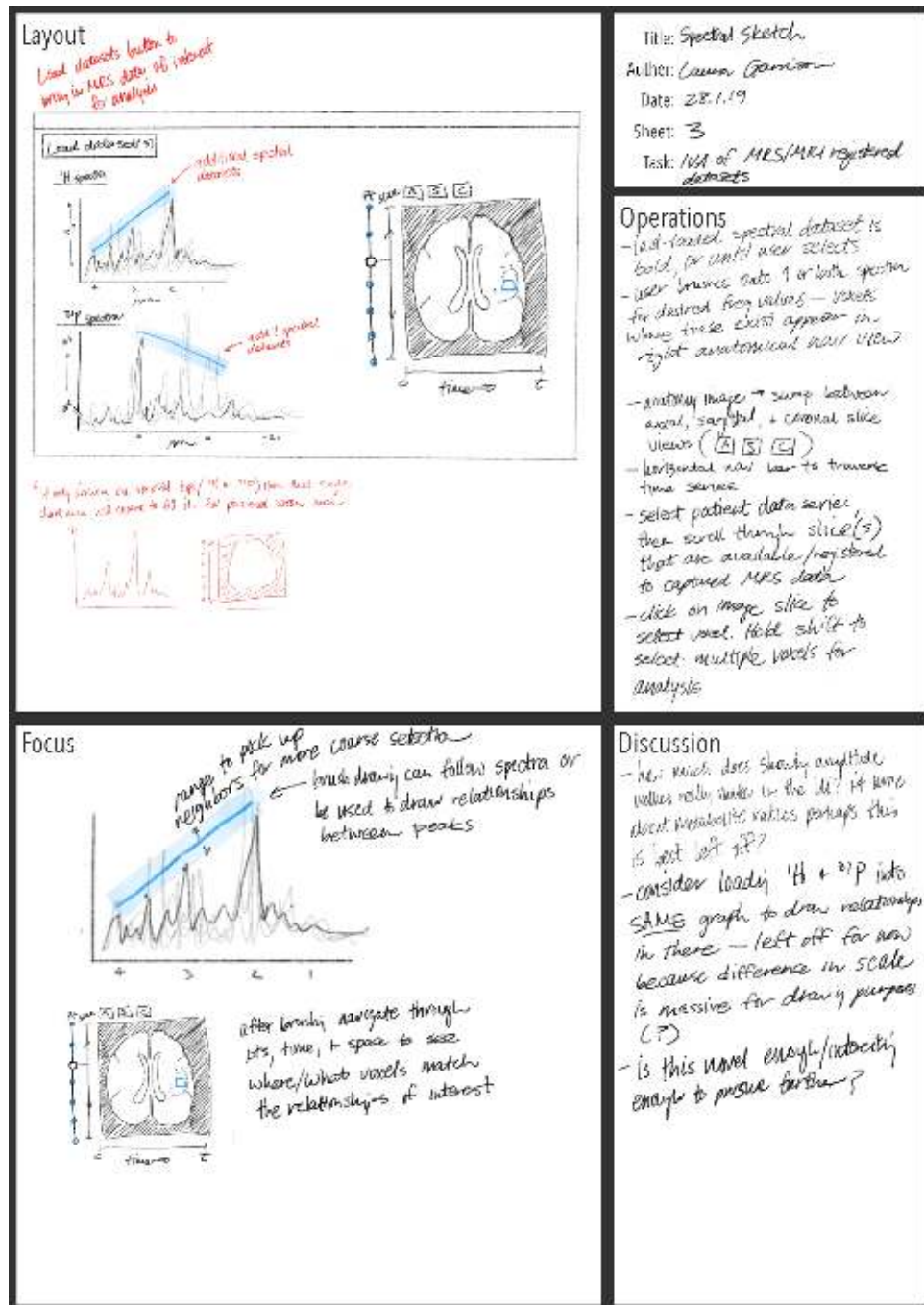


Figure A.3: Design sheet no. 3

## A. LOW-FIDELITY PROTOTYPE: DESIGN SHEETS

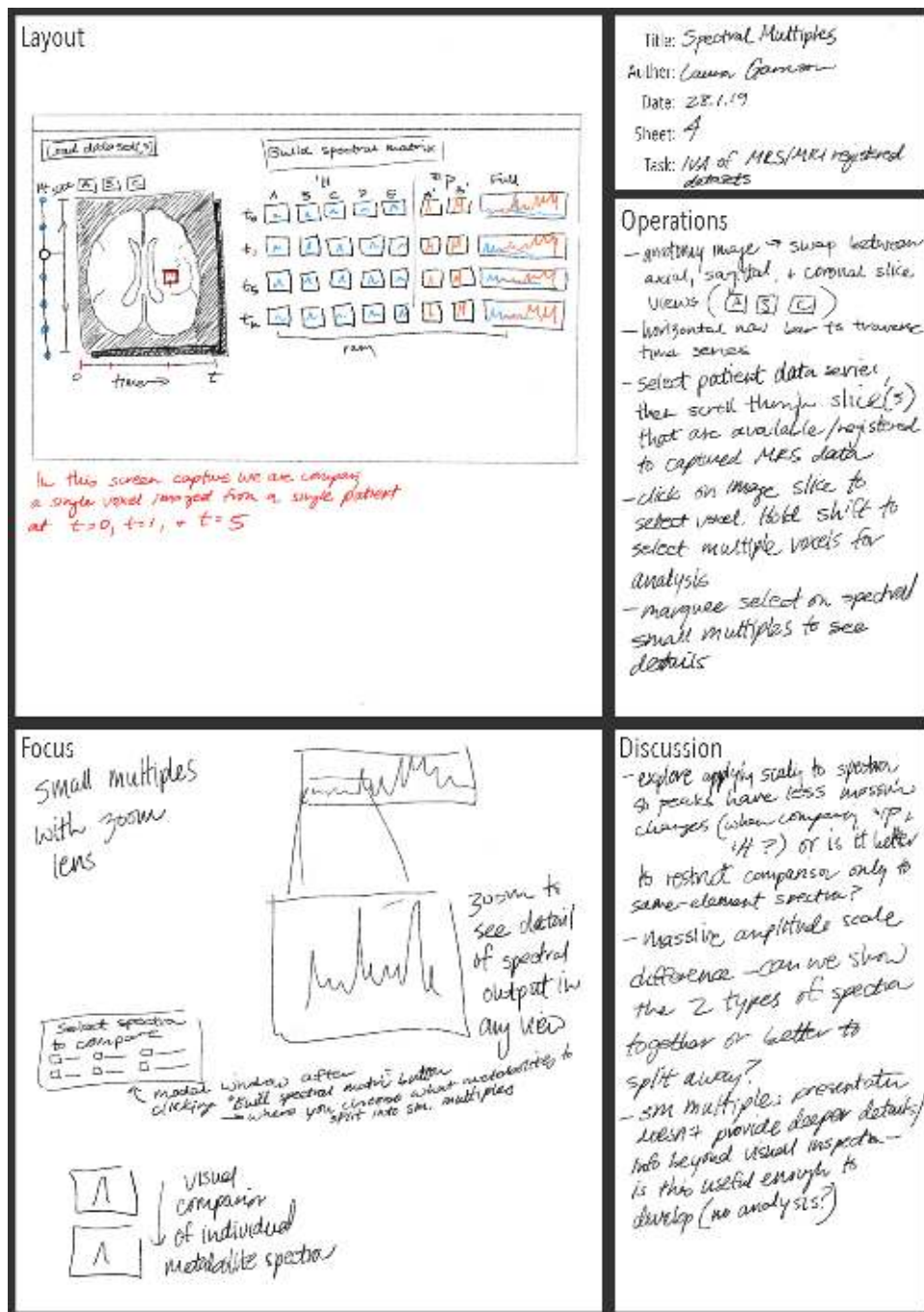


Figure A.4: Design sheet no. 4



## A. LOW-FIDELITY PROTOTYPE: DESIGN SHEETS

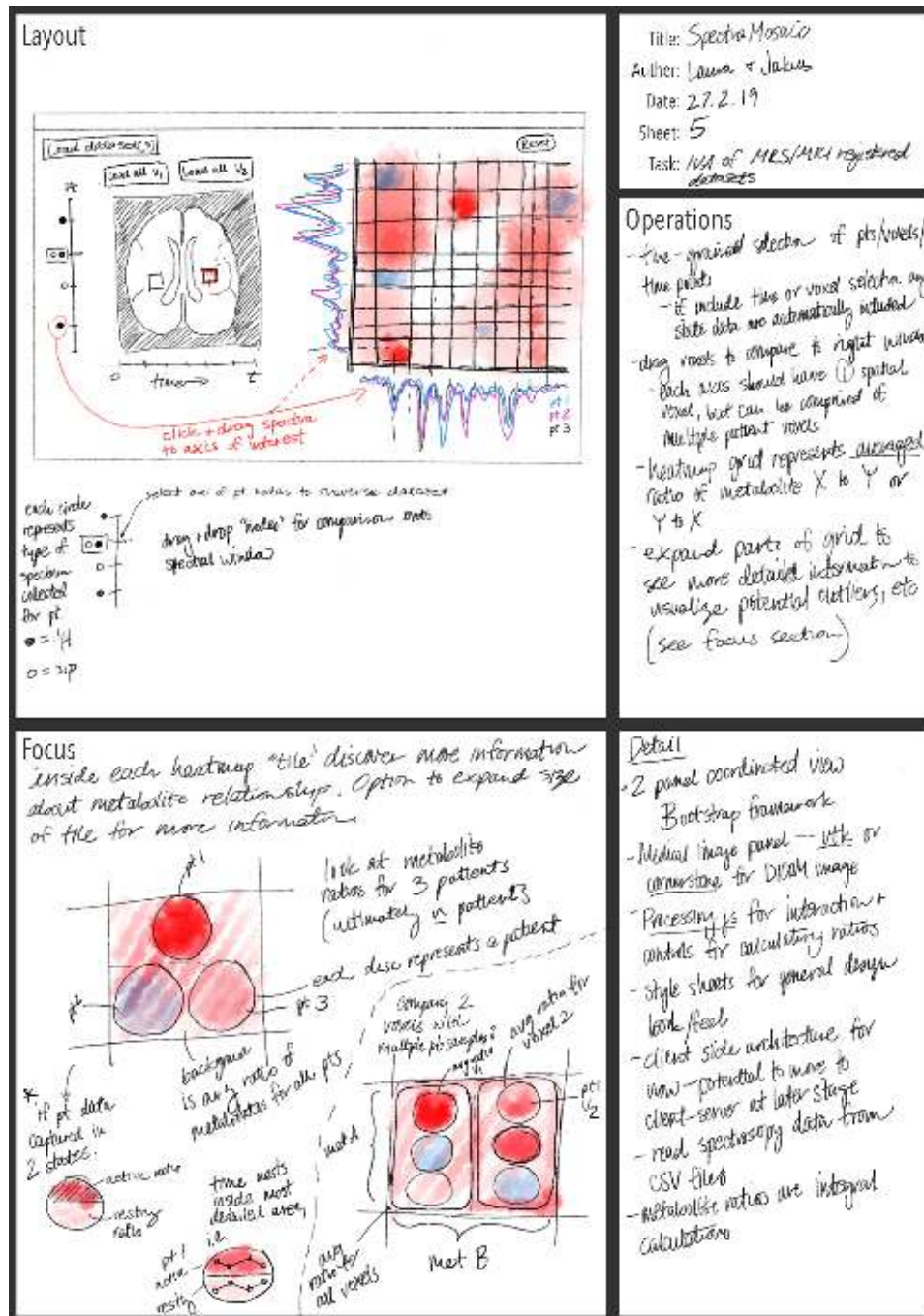


Figure A.5: Design sheet no. 5