# Abstract

The research in proteins and their functions (proteomics) is a promising field in clinical research. Since proteins are considered a fingerprint of biological functions and are vastly available in bodily fluids and tissue, proteomics can be used, for example, as diagnostic tool for the early detection of cancer. This would improve the chance of survival of a patient considerably. This kind of research is characterized by the cooperation of different highly specialized groups. Due to the interdisciplinary nature of proteomic research, it goes hand in hand with difficult communication and different interests. It proves to be no easy task to get all noses pointing in the same direction. Furthermore, the sizes of datasets are growing, due to more sensitive technology and growing sample collections. This has resulted in a raising demand for bioinformatics, to form a bridge between specializations and to create routines for fast data analyses. This thesis describes the development and implementation of SPECTRA: a data analysis program for protein patterns. It encompasses a loading procedure to extract sample information from complex structured data, exported by the mass spectrometer. Furthermore, it provides different data preprocess and dimension reduction algorithms, extensive visualization options and a classification algorithm.

# *Contents*

# *Acknowledgements*

Many people are involved some way or another in the realization of SPECTRA and this thesis. Though this section is prepared with care, I would like to apologize in advance to the ones who should have been named here, but are not. Evidentially, all mistakes and erroneous utterances in this thesis are entirely due to misconception on my side.

I would first of all like to thank my friends and colleagues at the Leiden University Medical Center, who made it possible for me to graduate in a stimulating environment: doctor Mirre de Noo, for being a terrific boss. Her guidance and comments on my work were very professional and helped me a great deal. Furthermore, she has been a great roommate, together with doctor Lee Bouwman. Professor Andre Deelder from the department of Parasitology and professor Rob Tollenaar from the department of Surgery, whose knowledge, professionalism and cooperation with each other is an inspiration for all involved in translational research. The great statistic minds, Bart Mertens and Paul Eilers from the department of Medical Statistics, for their scientific input in SPECTRA's algorithms and without whom the proteomic research would not be possible.
Thank you all, I am looking forward to work with you during my PhD at the LUMC!

I would like to thank the graduation committee from the Delft Technical University: Charles van der Mast and Frans Ververs for taking the time and effort to assess my work and especially Leon Rothkrantz who put me on the right track, when I got silt up in tiny details.

I thank my parents, for their encouragement during my education, for setting an example and for their support in so many more ways than they probably realize themselves. Coen Zimmerman for putting his grand artistic skills at my disposal and designing the professionalism beaming SPECTRA logo. Besides that, needless to say actually, for being a great friend.

I thank Nanda for her enduring love and support. She put up with my whining, when things were not working out as I wanted them to be, and looked after me.

Martijn van der Werff
Leiden, January 2006

# 1 Introduction

Nowadays, the central dogma of molecular biology states that proteins are closer to actual biological functions of cells than mRNA or DNA is [1]. Therefore, proteomics (i.e. the research of proteins and their functions) is a rising field of interest in clinical research. It is possible to generate a protein fingerprint of samples, based on unique protein mass information. The analysis of these protein patterns promises to be very valuable for a wide range of clinical applications. For example, in oncology, the detection of cancer in a preliminary stage would improve the chance of survival considerably [2]. Since there are no reliable noninvasive and few invasive diagnostic tests to detect certain types of cancer at an early stage, a noninvasive diagnostic test would be of considerable value. The discovery of disease specific biomarkers (in this case proteins or peptides) could contribute considerably to this. In 2002, an article by Petricoin was published in the Lancet, concerning the use of proteomic patterns in serum to identify ovarian cancer [3]. The results of this research were promising according to the authors. With a sensitivity of 100%, specificity of 95% and a positive predictive value of 94% the described method, they argued, appeared acceptable to serve as screening tool regarding the detection of ovarian cancer. However, in 2004, an article was published in Biostatistics, in which the reproducibility of the obtained data from Petricoin was criticized [4]. The authors reported findings of structural feature difference that were not the same across experiments. Currently, the department of Oncology, in cooperation with the department of Parasitology of the Leiden University Medical Center and the department of Medical Statistics, is involved in the research of colon carcinoma detection by means of protein pattern analysis using very well standardized sample collection and serum preprocess methods [5]. This research project is known as DIPSTICC.

## 1.1 Abbreviations

The following abbreviations will be maintained throughout this document:

| | |
|---|---|
| SPECTRA | Software for Protein-pattern Exploration in Clinical Trials and Research Applications |
| DIPSTICC | Differences In Protein Spectra Tested in Colorectal Cancer |
| LUMC | Leiden University Medical Center |
| MALDI | Matrix-Assisted Laser Desorption Ionization |
| MS | Mass Spectrometry |
| SELDI | Surface-Enhanced Laser Desorption Ionization |
| TOF | Time Of Flight |

## 1.2 Translational proteomic research

At first hand, DIPSTICC may seem a sole biological occasion. However, when looked at it more closely, the opposite appears to be true. For example, due to the high dimensional datasets (a protein pattern dimensionality can reach over 100.000) and relatively sparse datasets (the number of colon cancer patient samples is around 200 in the LUMC), an extensive part of the research consists of complex statistical computations. Furthermore, the samples are retrieved in a clinical setting and the implications of the research definitely have to be assessed by a clinician. Of course this is just a tip of the iceberg. The proteomic research at the LUMC can be tracked from fundamental research (the examination of protein structure) all the way to the highest level of application (the clinic, where the presence of cancer in a patient is assessed). In fact, the fundamental research needs to be translated into an applicable protocol in the clinic. Hence the name translational proteomic research. Three groups in the LUMC, the department of Parasitology, the department of Surgery and the department of Medical Statistics, each use their own expertise to contribute to DIPSTICC on their part of that track. The fact that these different areas of expertise all are engaged in this research, adds to the interdisciplinary research characteristics of applied proteomics in clinical situations. Akin to all research, but to interdisciplinary research in particular, the interfaces between the different specializations are susceptible to errors, due to communication discrepancies or lack of knowledge. A great challenge lies in the creation of automated interfaces that are lucid at all sides for the concerning specializations.

## 1.3 DIPSTICC

To establish the scope of the problem description, we first take a closer look at the DIPSTICC project. The objective of DIPSTICC is to find a unique protein pattern in human serum, associated with colorectal cancer. As is mentioned before, no simple diagnostic test yet exists to determine whether a person has

colorectal cancer or not. The finding of a specific protein pattern would be a great addition to the complicated and expensive arsenal of diagnostic methods. To find such a protein pattern, a series of accurate actions must take place (figure 1.1). First, human blood from patients with colorectal cancer and from healthy persons must be collected. It is important to retrieve them under the same conditions, so no artefacts other than the biological ones based on the cancer can occur in the protein pattern. This is done in the polyclinic of the oncology department. The information about the patient is written down on his file. This record is entered in the database of the researcher. Next, the serum must be extracted using a centrifuge and stored in -80°. When the actual pattern research begins, the samples are thawed and processed with magnetic beads (to isolate only the certain proteins in the serum). After a matrix solution is added to the serum, they are analyzed with a machine that generates the protein patterns. The collection of protein patterns is then statistically analyzed, to discriminate the colorectal cancer related pattern from the healthy related one. For this, the information in the database is used, to assign the correct group numbers to the associated protein patterns.

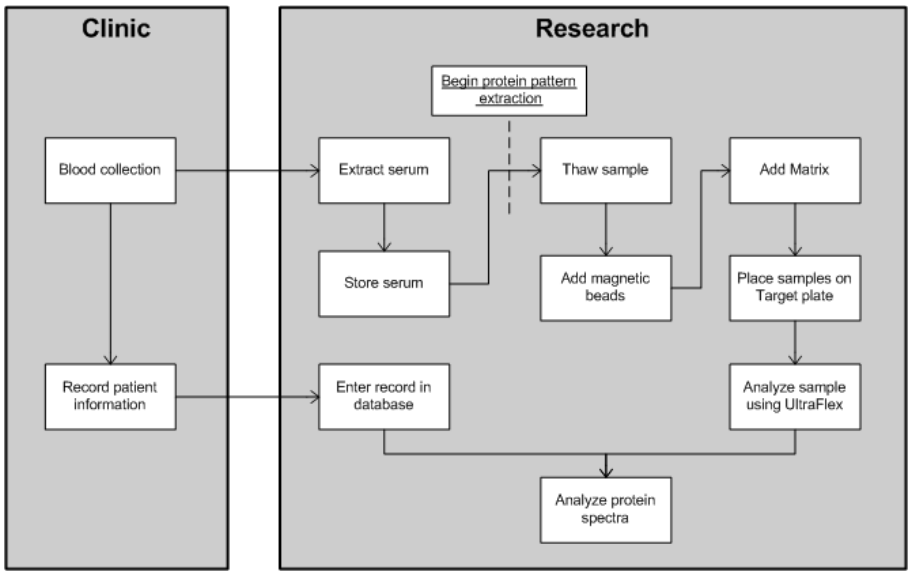


Figure 1.1: Overview of the sample handling sequence.

## 1.4 Problem description

During the project at the LUMC several aspects surfaced, that contributed to severe delay of DIPSTICC. Three of them were identified as solvable by creating a software tool. The first is the somewhat unusual way of data export. The mass spectrometer, exports the information concerning the pattern in folder names. To identify the spectrum, one of the upper folder names has to be used. The code of the statistician had to be revised whenever the folder structure changed.

Second, even small data analyses had to be performed by the statistician, due to the complexity of the data. This increased the workload of the statistician unnecessary. Last, the visualization possibilities of the data were insufficient. For a while, the data visualization was done using a software tool. However, this was an alpha version, and therefore contained several annoying bugs and was missing some most wanted features. When participating in translational research, dealing with different parties is per definition unavoidable. This induces a lot of problems. Of course this is not an unfamiliar problem. Many examples exist where an interdisciplinary environment has been the cause for disastrous errors, simply due to deficient communication. It is therefore important that the common mistakes concerning interdisciplinary projects are not repeated. Frequent communication and feedback to the user and employer will diminish the chance on recurrent errors. In the LUMC, together with the different research groups, emphasizing the translational nature of the project SPECTRA (Software for Protein pattern Exploration in Clinical Trials and Research Applications), the three aspects have been encapsulated in the following objective:

*To design and implement a software tool that automates and facilitates the analysis, visualization and classification of protein patterns.*

Before continuing with the actual design and implementation, some research has to be done, to gather material for the construction of SPECTRA. This brings us to the next chapter.

## 1.5 Research assignment

Since SPECTRA will be integrated as part of the DIPSTICC project, a thorough understanding of the current research situation is necessary. Furthermore, to contribute to the scientific development in the field of bioinformatic tools, the pros and cons of current comparable software must be analyzed.

***Target 1:*** *Analyze the current research situation and comparable tools.*

In order to come to a reliable foundation on which the program can be build, a clear overview of the functionalities of the tool is necessary. In close collaboration with the user, his ideas about the functionalities must be made concrete and molded into a good basis for a software design.

***Target 2:*** *Analyze and record the functionalities of SPECTRA.*

For the implementation of a tool that has to incorporate several data modification and analysis algorithms, it is imperative to look deeper into the required algorithms. The literature must be consulted for the algorithms that are suitable for this job. Furthermore, some in-depth study in the mathematical background of these algorithms is necessary, to implement the algorithms accurately.

***Target 3:*** *Create a theoretical overview of the required algorithms.*

Based on the requirement analysis, the design and implementation of the software tool is the next step in the process. Subsystem decomposition and class diagrams are two of several sub-targets that have to be resolved, in order to form a detailed platform on which the implementation will take place. The more accurate the design is done, the easier the implementation phase will be. The rise and fall of SPECTRA depends heavily on its appearance and the system interaction with the user. Through analysis of the user capacities and the development and evaluation of screen mockups and prototypes, a user interface has to be designed that is optimal adapted to the future users.

***Target 4:*** *Design the system architecture and user interface.*

After the design, SPECTRA will finally gain shape. The design will be processed into a fully functional prototype. The greatest challenges will be the interpretation of the user's characteristics and desires in the user interface and the conversion of the theoretical algorithms into operational and fast routines.

***Target 5:*** *Implement a working prototype.*

Finally, the assessment of the prototype will result in the final application. This must be thoroughly evaluated by the client, to assure the program fulfills all the wishes of the user (regarding this application that is). This must be done in a structured way, to ensure all situations are tested and reliable feedback is obtained.

***Target 6:*** *Test and evaluate the final application.*

## 1.6 Thesis structure

This document describes the development and implementation of SPECTRA. In chapter 2, the current research situation is analyzed, to get a good idea of the place of SPECTRA in DIPSTICC. The findings in this chapter will lead to a purpose of the system. Furthermore, user requirements will be analyzed through a questionnaire, which will result in a consented set of system functions. Chapter 3 presents a theoretical overview of the implemented algorithms. Accurate descriptions of the algorithms are required, in order to implement them in the system. The results of this research will be demonstrated through formulas and pseudocode. The analysis forms a sturdy platform to perform a system and user interface design. This will lead to some well-defined guidelines, specific and unambiguous enough to implement SPECTRA. Chapter 4 describes the design procedure and the results are expressed in use case diagrams and communication models. Implementation aspects and decisions are described in chapter 5. The result of the implementation chapter is a fully operational application. The user and performance tests of this application are described in chapter 6. Finally, the results of the entire project are discussed in chapter 7.

# 2

# Analysis

Clincial proteomic cancer research is still in its infancy. After the publication of Petricoin [3], the current research focus is mainly on the standardization of sample treatment and logistics, to exclude contribution of artefacts to the spectrum other then changes caused by cancer. Next to that, the application of discriminating algorithms on proteomic data is considered a vital aspect and point of discussion. The focus is slowly sliding towards the classification. This situation is important for the development of SPECTRA, because it excludes the possibility of one commonly agreed perfect classification algorithm. Later on, the subject of multiple algorithm integration will be treated in more detail. The necessary data for this chapter is extracted from multiple sessions with the user. After the initial intake conversation, a rough set of requirements and a problem statement was formulated. This formed the basis for a questionnaire (see appendix D) in which the user commented on the requirements and problem statement. Furthermore, the user was asked questions about the software tool that was used for data analysis. The filled-in questionnaires were signed and the answers were evaluated during an oral discussion. This resulted in the requirements and existing software descriptions in this chapter. Let us first take a closer look at the current research situation at the LUMC.

## 2.1 Current research situation

Currently four groups that work together in the DIPSTICC project are generally distinguishable as the researcher, the physician, the analyst and the statistician. Note that although the names of these groups are of singular form, it is likely that they consist of several people. For example, the group 'physician' is defined as the people that collect samples (they can be e.g. nurses or surgeons etc.) but also the physicians that assesses the medical condition of the patients. The groups are further described in this chapter. Although the research situation description is based on the DIPSTICC project at the LUMC, it can

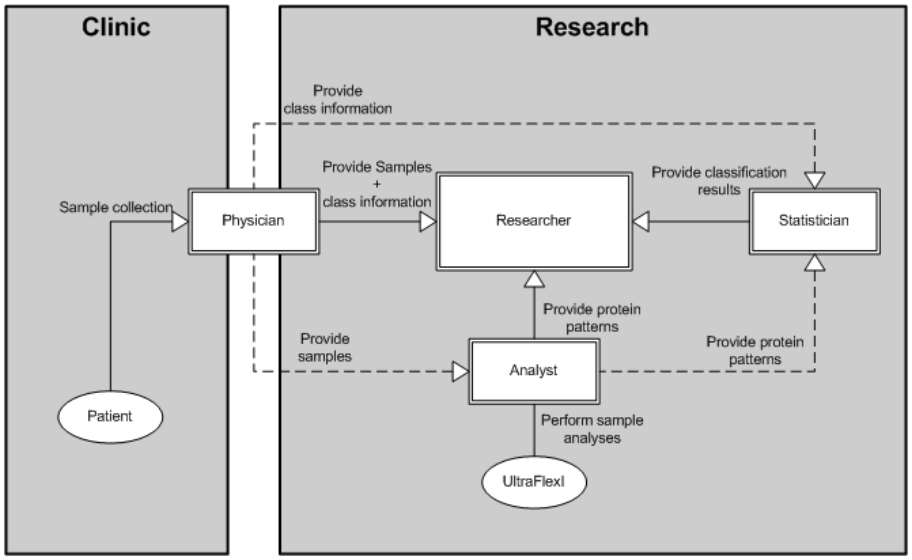


Figure 2.1: Current research situation

easily be extended to more general situations, in which different techniques and methods are used. The general principle will remain the same over all cases. The research situation structure is represented in figure 2.1. The solid lines in the figure represent direct contact between groups. The dashed lines represent inexplicit interfaces between the different groups. These groups do not communicate directly with each other, but indirectly through the data that flows between both parties.

The researcher is the supervisor of the entire study. She needs to know everything about the project at all times. Therefore, the supposed course of events is that the other groups communicate with each other through the researcher. This is represented in figure by the solid lines. Besides managing the process, the researcher is responsible for managing the data. Since in the LUMC the researcher is also part of the physician group, she has full access to all patient records. This facilitates the process considerably, because between the researcher and the physician, the chance of mistakes is fairly low. It can be argued that a fifth group, the datamanager, should be created. However, in the DIPSTICC situation, the database is maintained by the researcher. Moreover, desired is to prevent mistakes so the number of groups needs to be as small as possible. If an activity is not necessarily specialization-based (like database-maintenance), the activity is preferably appointed to the researcher. Besides data, the samples also have to be managed. This involves an accurate registration of location of samples and the responsibility that the samples are actually positioned according to that registration. Another important job of the researcher is to supply the statistician with the right sample identification number/mass spectrometry number/class number/protein spectrum tuples. This contains a considerable amount of data retrieval. The activity is very sensitive to errors and needs to be done accurately, since a wrong class assignment would severely affect the results. The final activity of the researcher is to interpret and discuss the results, produced by the statistician, together with the physician and

to write an article that will be submitted for publication in a scientific magazine. Again, the fact that the researcher also is part of the physician group, simplifies the situation considerably.

The physician collects the data, e.g. serum or blood samples, from her patients. She is responsible for a clean data collection. This implies that strict protocols for sample collection have to be maintained so all samples are collected according to a standardized protocol. The reason is that the chance of protein pattern discrimination based on artifacts, other than biological properties of the groups that are to be researched, is diminished. This is a major criticism in nowadays proteomic research. The results, first obtained by Petricoin et. al. [3] were criticized by Baggerly et. al. [4] based on the variation of patterns in one group. De Noo et. al. [6] described that some logistic and environmental issues do have influence on the protein pattern. It is therefore crucial for the research that the physician does the sample collection accurately according to protocol. Another important task is the assessment of the medical condition of the patients and the interpretation of the results, received from the statistician. Implicit communication with the analyst occurs through the supply of samples. These have to be processed by the analyst to obtain protein patterns. Implicit communication with the statistician occurs through the supply of sample identification number/class number association. Because of privacy regulations, the physician is the only person who, on account of her occupation, has access to patient information. The class numbers specify the group to which the sample is assigned (e.g. patient group, control group etc.). Obviously, these assignments are needed for classification and validation. The explicit communication with the researcher is the actual sample and data supply.

The analyst performs the actual protein pattern measurement with the samples. The samples are processed with MALDI-TOF using the UltraFlexI(figure 2.2) that records the protein pattern of the samples (see section 3.1). Her main


Figure 2.2: Bruker's UltraFlexI

responsibility is to execute and supervise the process, again to guarantee that the reliability of the samples is maintained (so no artifacts can occur in the spectra) and the right mass spectrometry number is associated with the right spectrum. However, this will usually be done by the mass spectrometer. The

implicit communication with the statistician is to provide him with the protein patterns and the associated mass spectrometry number. The only reason that the researcher has no view of the data is because of the privacy issues concerning the consult of patient records. This is overruled in the current situation, since the researcher belongs to the physician group. However, in the current situation at the LUMC, the analyst also produces the sample identification number / mass spectrometry number combination. This situation is not ideal, because of error sensitivity and vagueness of responsibility. As is mentioned in the previous note, the researcher (or the datamanager) should be responsible for all data issues. This includes the supply of a correct association between protein spectrum, and identification number and mass spectrometry number. Some improvement of the current situation is achievable on this point.

The statistician performs the actual data analysis. The protein spectra and their sample identification number from the analyst, together with the identification sample number and their class number from the physician are enough for classification and validation of the spectra. First, he preprocesses the data to obtain a normalized representation of the data. Then he performs a classification algorithm on the data. Results presented to the researcher manifest themselves in a model and statistical factors, like sensitivity, specificity and recognition rate.

Although the researcher is the supervising party (indicated by the solid line), all parties communicate with each other through the data they need and generate (indicated by the dashed lines). All communication lines are vulnerable to misunderstandings. It is vital for the project that these mistakes are minimized. Therefore, all parties need to be forced to maintain a fixed protocol, regarding data handling and regarding their data representation. Another way to diminish the chance of mistakes is to integrate the different groups. One way to do this is to create a software tool that incorporates the specialized knowledge of the groups so protein pattern analysis is facilitated and automated.

## 2.2 Currently used mass spectrometry technique

After the blood-samples are obtained by the physician according to standardized protocols, the serum is extracted, using centrifugation and preserved at -80° C. Just before protein spectra are measured, the samples are thawed and distributed over tubes. These tubes can be stored again if feasible. From here, the next steps are achieved using a Hamilton distribution robot (figure 2.3).

Isolation of peptides is done by adding magnetic beads with a peptide-affinitive surface to the serum. These beads are extracted using a magnet and then prepared with a matrix solution. In the LUMC, an UltraflexI (Bruker, Bremen, Germany) is used for protein-pattern extraction. The device is operated by an analyst and is based on the MALDI-TOF mass spectrometry technique. This technique is thoroughly described in chapter 3.1.3. The exported data format is a binary ASCII file consisting of two columns, the first being the $m/z$ values and the second the associated protein intensity values.

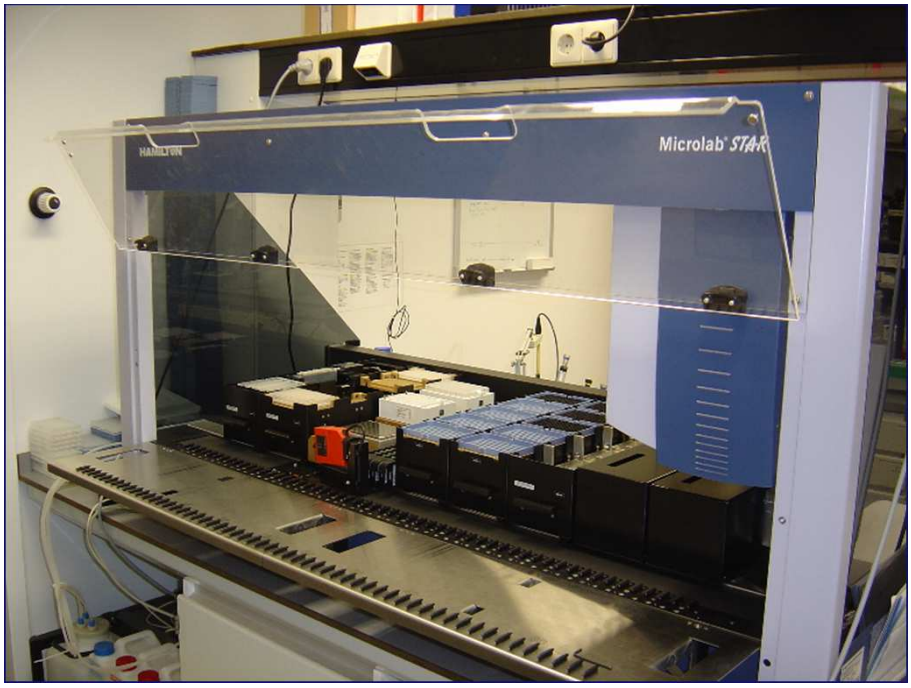


Figure 2.3: Hamilton robot

## 2.3 Purpose of the system

This software tool will be designed according to the wishes of the researchers of the LUMC that participate in proteomic related research. They will also be the future users of the system. Different possibilities can be explored and they can express their practical needs through the use of prototypes that will be presented to them throughout the process of the development of SPECTRA.

- The data retrieval part of the researcher will be automated to facilitate his work. This means she does not have to manually extract all information from the data.
- The visualization possibilities of the data will be elaborated, to provide the research more insight in the protein patterns.
- The preprocess part of the statistician will be automated to facilitate his work on the data. Nowadays the statistician programs all modifications of the raw data himself. A push on the button will simplify this time-consuming activity considerably. The preprocessed data could then be used for research in, for example, other classification algorithms. The preprocessing is also necessary to come to a reliable classification.
- The classification part of the statistician will be automated to facilitate the work of the researcher. The researcher can now perform small analyses of the data without the statistician.

In fact, the purpose of SPECTRA is to automate part of the interface between the researcher and the statistician, thereby reducing the risk of mistakes. The

entire process of data analysis is integrated in one program. Figure 2.4 represents the desired situation.

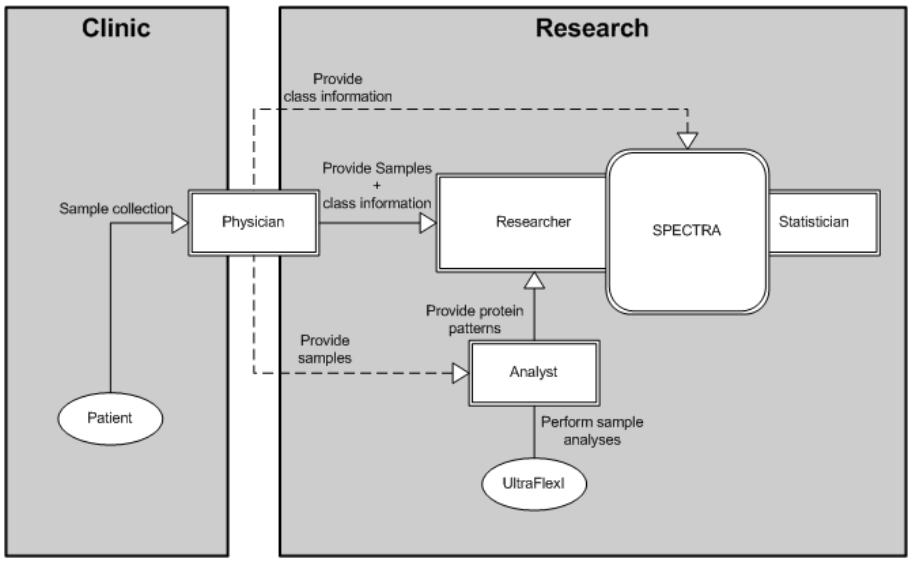


Figure 2.4: Desired research situation

## 2.4 Existing systems

To create an application with the maximum amount of advantages of today's comparable systems and without inventing the wheel all over again, research hast to be done in systems that are currently used to deal with protein patterns. A major predicament on this subject however, is the pioneering nature of the research. The consequence is that not many (commercial) available software tools yet exist. Actually, only *one* is available on large scale, provided by Bruker Daltonics, the leading manufacturer of MALDI-TOF devices, like the UltraFlexI and UltraFlexII, called ClinProTools. Another reason for the absence of reviewable tools is the fact that each research group develops its own tool, for internal use only. Since a research group often encompasses its own bioinformatics, they rather program their own algorithms they consider the correct ones then use existing tools, which are inflexible and not yet thoroughly validated. Two other commercial tools that exist, ProteomQuest from Correlogic and ProPeak from 3ZInformatics are more specifically aimed at the identification of peaks than on classification and validation. Few specifications of these programs exist and answers from the concerning parties to documentation requests were not satisfying enough to give a good description.

### 2.4.1 ClinProTools

In the LUMC, ClinProTools is used, because of the compatibility with the UltraFlexI. A short description is given, along with its advantages and disadvantages

in use. The information is based on the experience of the researchers in the LUMC and is derived partly from appendix D. ClinProTools has a one-screen interface, which means that all actions (data retrieval, preprocessing, classification and visualization) are present in one screen (figure 2.5).

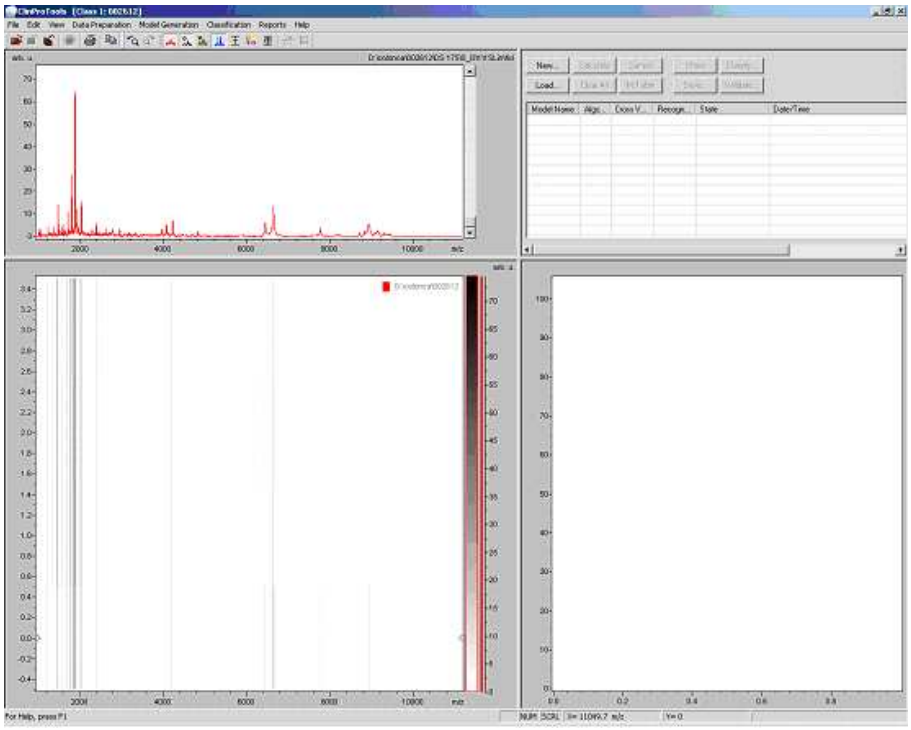

Figure 2.5: Screenshot of ClinProTools

Since ClinProTools is developed by Bruker Daltonics, it is primary suitable for exported UltraFlexI files. Therefore, data retrieval is fully implemented for UltraFlexI files and far easier than to manually extract the information of the spectra. It offers various visualization options, e.g. ClinProTools displays averaged spectra, compared spectra and single spectrum with intuitive visualization features such as trace, virtual gel, contours and stacked views. The locations of the biomarkers can be highlighted and it allows users to visually inspect individual spectrum to verify their results. The parameters for data processing, like baseline subtraction, peak definition, calibration, and normalization, can be manually defined. ClinProTools generates and validates pattern recognition models using different sophisticated mathematical and bioinformatic algorithms. The obtained results for each analysis can be stored.

The heat map and biomarker highlighting were considered advantages of the system, by the users of the program at the LUMC. Several disadvantages of ClinProTools were identified. The most important is the fact that the used algorithms were not clearly described. This black-box approach makes it less suitable for research applications. Another issue is that ClinProTools has no options for multiple spot handling implemented. Only one spectrum of one spot can be loaded into the software. In order to correct for technical variation, multiple spots on a plate are used to represent one sample. Since ClinPro-

Tools integrates all function options in one screen, there is little space left for the visualization of the spectra. The user therefore considers the visualization slightly messy by the user. The last recognized disadvantage is the several bugs in the program, which causes system instability and occasionally shuts down the computer.

### 2.4.2 Conclusion

The analysis of ClinProTools results in several considerations regarding the development of SPECTRA in relation to ClinProTools. Although ClinProTools is a commercially available product, on which several professional programmers have worked for years, it still has some aspects that can be improved according to the user. SPECTRA can use these points as well as the advantages of ClinProTools to its benefit, by paying extra attention to the implementation of them. Since the users are excited about the heat map function in ClinProTools, it should be implemented in SPECTRA. The transparency is poor in ClinProTools, so emphasis should be laid on providing a detailed description of the underlying functions of SPECTRA. The help function and detailed user manual will fulfill this requirement. Furthermore, the users miss a function, handling one sample distributed over different spot on one target plate. A function should be implemented in SPECTRA that copes with this issue. Finally, the visualization options of ClinProTools are not sufficient. SPECTRA needs to supply extra visualization options, like a correlation coefficients plot. These conclusions contribute to the set of requirements of the user regarding SPECTRA.

## 2.5 Requirement analysis

Now we have a general idea of the functions of SPECTRA, a detailed requirement analysis will provide a framework for further design of the application. As mentioned before, the requirements were established by analyzing and discussing a structured questionnaire.

*Modular approach for easy modification and adaptation to other datasets.*
Because of the innovative character of the proteomic field of research, the protein pattern extraction techniques and analysis methods are heavily subjected to alterations. It is therefore desired that SPECTRA, that is preordained to be used for this field of research (hence the name SPECTRA) is flexible and adaptable to other datasets and preprocess- and classification algorithms. To enhance this flexibility, SPECTRA preferably has to be build from different segments, that each interact with each other using a transparent interface. As additional requirement, SPECTRA will emphasize and elucidate the segmented approach, by displaying different screens for the different modules.

*Easy-to-learn, vivid and intelligent interface.*
Because of the expert nature of the system, interaction between the system and user is vital. The user must be instantly informed when wrong input occurs

about the meaning of the associated fields and buttons. Furthermore, it must be challenging and colorful. However, more important is that the user can intuitively use the interface in a rightful way. Screen mockups and prototype analysis and evaluation will be used for this goal.

*Must run on a Windows OS machine.*
Because the systems in the LUMC function on a Windows platform, it is desired that SPECTRA does also.

*Compatibility with Microsoft Excel and Access for class assignment.*
In the current situation, the data is managed, using Microsoft Access and Microsoft Excel. To prevent errors, made during class assignment to samples, it is desired that SPECTRA is compatible with class tables, directly imported from Excel.

*Data must contain: Unique mass spectrometry number, unique sample number, class-label.*
To facilitate data retrieval, it should be possible to export some information in a desired file format. The following information should be preserved per spectrum, and be possible to export: A number which is unique for every spectrum, a number which is unique for every sample and a label, representing the class to which the sample belongs.

*Data export possible in Matlab- and Microsoft Excel-format.*
SPECTRA should include two types for export: a Matlab .mat file, since the statistic department uses primary Matlab for their analysis and a Microsoft Excel .xls file, since the research department uses Microsoft Excel for their analysis. This is also very convenient for rapidly creating class assignment files as will be shown later on.

*Preprocessing functions available must be: binning, smoothing, normalization, baseline correction.*
To preprocess the data, several functions must be included in SPECTRA. The implemented algorithms are thoroughly described in chapter THEORY. Reduction of the data dimensionality using binning must be possible. Noise reduction of the data using smoothing must be possible. Correction of a distorted baseline must be possible.

*Transparency of used algorithms through help function, user parameter definition and well-documented algorithm information.*
Since a major criticism on ClinProTools is that it is not clear what algorithms it uses, SPECTRA must be fully transparent. An extensive help function contributes to the transparency of the system. The algorithms must be described in detail and this description must be available on request. For full transparency, all parameters may be defined by the user, so the user has optimal control over the data processing and analysis.

*Quick data analysis possible with default parameter values.*
When data needs to be analyzed quickly, and the current methods of data processing and analysis are considered correct, SPECTRA must include the

possibility to use default values for preprocessing and classification. This saves the user from entering every single parameter value. Furthermore, a user who is not familiar with certain algorithms is still able to use the system.

*Spot selection possibilities: all spots and spot average.*
Another criticism of comparable systems is that they cannot handle spectra on different spots, from the same sample. This feature is important for reproducibility issues, e.g. to eliminate plate and batch variances. Therefore, different spot handle functions should be integrated in SPECTRA. It should be possible for all spots to be treated as different samples and the average of different spectra on different spots should be possible to be computed.

*PCA and LDA must be integrated.*
To reduce the high dimensionality of the proteomic data, Principal Component analysis should be integrated in SPECTRA. Dimensionality reduction is necessary for a reliable classification. For this classification, Linear Discriminant Analysis must be integrated in SPECTRA. Since the users are familiar with these algorithms and are currently consider this the most correct algorithms when applied on proteomic data these specific algorithms should be implemented and research must be done to these algorithms only. They are described in detail in chapter THEORY.

*Leave-one-out validation and external validation must be implemented.*
To validate the model, obtained by the LDA, two possibilities should be implemented: external validation, using an independent validation set, and leave-one-out crossvalidation, using the entire data minus one sample to calibrate the model and then use the left-over sample for validation of the model.

*All data information must be visible.*
The information about the data: class, ms-number, sample number, spot should be printed on the screen. Next, the data manipulations should also be visible on the screen.

*Classification results must be visible.*
It should be clear which spectra have been classified incorrectly and correctly. Furthermore, the classification parameters, like recognition rate, must be shown.

*Visualization options must be: heat map, averages, single spectrum, correlation weight plots, multiple plots, mirror plots.*
Another criticism on comparable systems is the limited and chaotic way visualization is done. Additionally, more options of visualization are desirable. It should be possible to display a heat map of the data. A visualization of the averages of multiple selected spectra should be possible. A single spectrum should be possible to visualize. The weights of all $m/z$ correlation values after dimension reduction using PCA should be able to be displayed. It should be possible to plot multiple spectra in one plot. Last, a negative plot of spectra should be possible, in order to compare spectra in a intuitive way.

*A zoom option must be included.*
To view an image in more detail, the user asks for a zoom option to be included.

## 2.6 Assessing the system

The level of success of the final application will be measured through the following methods.

*Extensive application evaluation by the user.*

The final implementation of SPECTRA will be assessed by the users. They will try different actions while thinking aloud. The whole procedure will be logged and reviewed in an interview.

*Qualitative requirement evaluation.*

The final implementation of SPECTRA will be assessed by critically evaluating the requirements. A short comment will be written on all stated requirements and the final result.

*Reliability tests and comparison with ClinProTools.*

The reliability of SPECTRA will be assessed by comparing results obtained from the data analysis of colon cancer patients versus controls, with the results of the written paper (see Appendix A). To investigate SPECTRA's performance in comparison with ClinProTools, an analysis will be performed on both systems. The results will be compared.

*Robustness tests and benchmarking.*

The program will be tested for robustness by putting the application through a series of extreme situations. A description of the reactions of the system on the situations will be described.

# 3

# Theory

In order to proceed with the design and implementation of the actual application, first some background theory needs to be provided. This section describes the mass spectrometry methods used in the LUMC and the algorithms, used for the arithmetic operations on the data and that are implemented in SPECTRA. The implementation details of the algorithms can be found in chapter 5. For a detailed description of the statistical analysis of the DIPSTICC project and further references for this chapter, consult appendix A and the statistical validation paper [7].

## 3.1 Protein spectrum extraction techniques

SPECTRA analyses data in the form of protein-patterns. To fully understand the functionality of the program, it is essential to have some knowledge of the techniques used to establish these protein-patterns. Therefore, this section is dedicated to some theory of mass spectrometry techniques.

### 3.1.1 Mass spectrometry

Mass spectrometry is an analytic technique that measures molecules by their mass-to-charge ratio. Nowadays, mainly two techniques are used in the research of protein patterns, namely Surface Enhanced Laser Desorption/Ionisation-Time Of Flight (SELDI-TOF) and Matrix Assisted Laser Desorption/Ionisation-Time Of Flight (MALDI-TOF). A sample, usually tissue or bodily fluid, is treated with a protein-affinitive solution. This way, only proteins remain in the eluate. The prepared sample is irradiated with laser pulses and the molecules are subjected to a process of desorption and ionization accompanied by fragmentation. The mass spectrometer measures the mass-to-charge ratio of the protein, peptide or peptide fragments. This can be achieved in a flight tube by

the measuring the time between the ionization of a molecule and the arrival at a detector.

The mass of a peptide or protein, divided by its charge ($m/z$) is considered unique for every peptide. A typical TOF spectrum has the $m/z$ values on the horizontal axis and the intensities on the vertical axis (figure 3.1).

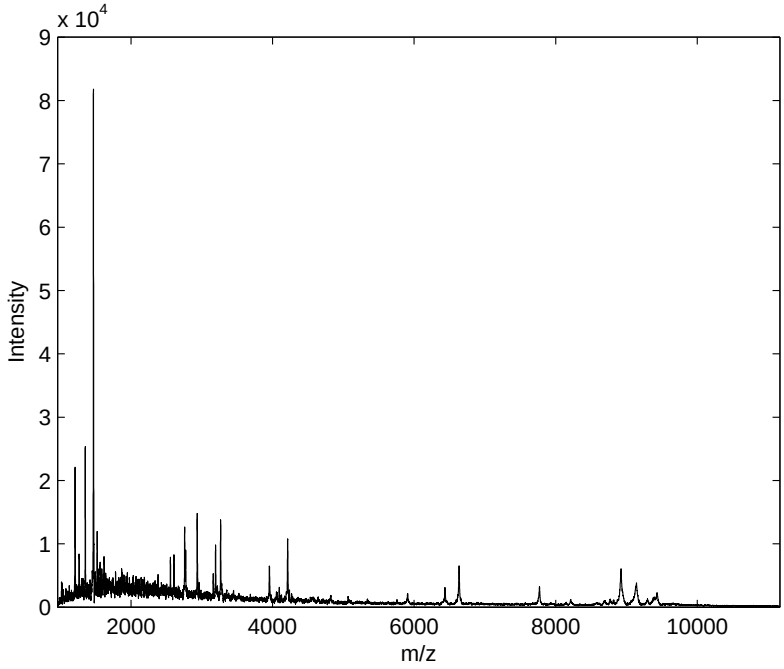


Figure 3.1: Typical protein spectrum

### 3.1.2 Theory of TOF

Once the ions are detected by the sensor, the mass per charge value can be calculated from the several established parameters of the MS instrument. Since the ions all receive the same initial kinetic energy, the mass of the ions is the only factor that determines the time of arrival at the sensor while drifting through a linear tube. The $m/z$ values can be calculated through:

$$\frac{m}{z} = D = 2eEs\left(\frac{t}{d}\right)^2 \tag{3.1}$$

With $m$ being the mass of the ion, $z$ the specific charge of the ion, $e$ the elementary charge, $E$ the extraction pulse potential, $s$ the part of the tube over which $E$ is applied, $d$ the length of the drift free zone through which the ion moves and $t$ the determined time-of-flight of the ion. Figure 3.2 shows a schematic representation of TOF.

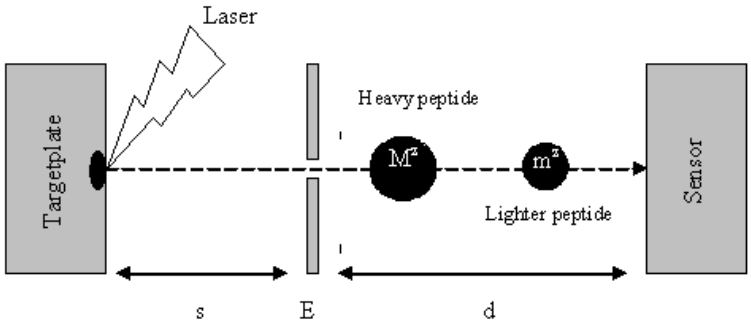


Figure 3.2: Schematic overview of TOF

### 3.1.3 MALDI-TOF

With MALDI-TOF, the sample is prepared with a matrix that causes the mixture to crystallize when it dries. The sample is distributed on a target plate. The plate containing the crystallized solution is placed in a vacuum chamber and is irradiated by a pulse-laser. This causes the crystals to desorb and to ionize. The ions then accelerate through a flight tube. The sensor detects small ions possessing a relative higher velocity than the large ions, earlier. The addition of a matrix to the sample prevents the fragile peptides from degeneration, resulting in a high resolution output [8] [9].

### 3.1.4 SELDI-TOF

SELDI-TOF is a simplified protein isolation method of MALDI-TOF. On the basis of the work of Hutchens et.al. [10], Ciphergen Biosystems, Inc. developed the surface enhanced laser desorption/ionization ProteinChip [11]. This is a prepared commercial chip, available with different surfaces with molecule-affinitive solutions. Only molecules that have affinity with this solution will attach to the chip. After the addition of serum, energy-absorbing molecules are added to abridge the ionization of molecules. The molecules are also ionized using a pulse-laser, but the acceleration tube is shorter than MALDI.

## 3.2 Preprocessing

Due to capacity issues, data reduction is needed prior to classification. Eventually, a dataset is pursued, which represent the protein pattern as best as possible, is clean from noise and is small in size. This is even more important, since the intensities of protein peaks are relative to each other. No assumptions can be made concerning the amount of proteins in one sample, based on consecutive peaks. Therefore, the *pattern* is the one thing that is interesting. In the preprocessing phase of the application several data-manipulation options are available. The following data manipulations are derived from the requirement

analysis: Smoothing, binning, baseline removal and normalization. The terms are extensively described below.

### 3.2.1 Smoothing

As is frequently the case with measured data, the protein patterns are liable to noise. Because a protein spectrum consists of discrete protein intensities, a good way to reduce this noise is to use a smoother. Generally, a smoothing algorithm reduces the local variability of the data, resulting in a less coarse representation of the data. Several methods for smoothing are available in literature, varying from the more commonly known Fourier filters and wavelets, to the less famous Savitzky-Golay filter [12]. Because it is beyond the scope of this report to give a detailed description of all smoothing methods, this section is restricted to describe only the algorithm that is integrated in SPECTRA.

### 3.2.2 SPECTRA smoothing algorithm

Whittaker described [13] a series $y_s$ to be smooth if its third difference:

$$\Delta^3 y_{si} = y_i - 3y_{i-1} + 3y_{i-2} - y_{i-3} \qquad , i = 4, 5, \ldots, N. \tag{3.2}$$

is small. Series can be used, since protein patterns are represented by discrete intensity values (namely, the number of molecules detected by the sensor). The variable $y$ is used here, because it concerns a smoothing of the protein intensities. The smoothness of a series $y_s$ is defined as

$$S = \sum_i (\Delta^3 y_{si})^2 \tag{3.3}$$

With $S$ is zero representing perfect smoothness (a parabola). Take heed that the term 'perfect' used here, refers to the absolute value of smoothness. It does not necessarily mean that the smoothed data is ideal for the representation of the original data. Beside smoothness, a second parameter defines the quality of the smoothed data $y_s$, namely the fitness of the smoothed data to the original data:

$$F = \sum (y_i - y_{si})^2 \tag{3.4}$$

In fact, a consideration must be made between the smoothness and the fitness of the data to obtain maximally reliable smoothed data. This reliability can be optimized using:

$$Q = F + \lambda S \tag{3.5}$$

A series $y_s$ must be found so $Q$ is minimized. It is obvious that the two features are contradictory. A high smoothness factor will per definition result in a low fitness factor, provided the original data is coarse. The user-defined parameter $\lambda$ defines the influence of the smoothness on the quality (i.e. how important the smoothness is considered). If $\lambda$ is chosen too high, the similarity of the smoothed data on the original data will suffer severely. If $\lambda$ is chosen

too low, the original data will not be smoothed at all. To minimize $Q$, the first derivative of equation must be determined and set to zero. To simplify calculations, equation 3.5 can be converted in the notation of matrices:

$$Q = (y - ys)^2 + \lambda(\mathbf{D} \times y_s)^2 \tag{3.6}$$

$D$ is a matrix such that $\mathbf{D} \times y_s$ is the accumulated third order difference $\Delta^3(y_s)$. Thanks to the matrix notation and Matlab, the formula can easily be extended to a general form of $n$-order differences, and $\mathbf{D}$ being the accumulated $n$-order difference matrix. For an increasing $\lambda$, the data will approach a polynomial of order $n - 1$. This is because it will move towards the least-square regression line (see equation 3.3). Experience shows that an order of 2 is usually sufficient. The vector of partial derivative can be found using the results of matrix calculus:

$$\frac{\delta Q}{\delta y_s} = -2(y - y_s) + 2\lambda\mathbf{D}'(\mathbf{D} \times y_s) \tag{3.7}$$

If set to zero, this results in the linear equation:

$$y_s = \frac{y}{I + \lambda\mathbf{D}'\mathbf{D}} \tag{3.8}$$

The optimal smoothed series $y_s$ can thus be derived from this equation, and associated $\lambda$ value. In this case it is assumed that the data is distributed at equal intervals. However, because of the non-linear character of the TOF equation 3.1, this is not the case in the MALDI-TOF protein patterns (chapter 3.2.3, 3.1.2). Therefore, the algorithm must be extended to non-uniform sampling. The $n$-order difference is now dependent on the intervals (bin-width) of the data. The following recursive equation, the so-called $n$-order divided differences, represents this:

$$\Delta^n y_{si,n} = \frac{\Delta^n y_{si,n-1} - \Delta^n y_{si-1,n-1}}{y_i - y_{i-n}} \tag{3.9}$$

After the accumulated $n$-order difference matrix is obtained using a recursive function, the smoothed spectrum $y_s$ is not differently produced than in the linear case (see equation 3.8). The algorithm is extensively described in [14]. A graphical representation of the effect of the smoother is presented in the figures 3.3 and 3.4.

Figure 3.3 is a part of a large protein spectrum, zoomed in on the region 940(Da[1]) $\sim$ 1040(Da). Figure 3.4 shows the result of a applied smoother with $\lambda = 100$ and $n = 2$. Since these values are used in the DIPSTICC project, they will be used as default values.

### 3.2.3 Binning

The raw data, exported from the UltraFlexI is high dimensional and contains a tremendous amount of information. For example, a raw protein spectrum

[1] 'Dalton' or 'Da' is a unit of mass very nearly to that of a hydrogen atom, named after John Dalton (1766-1844). It is interchangeably used with 'molecular weight' and, since we assume 'z' constant, with 'm/z' [15].

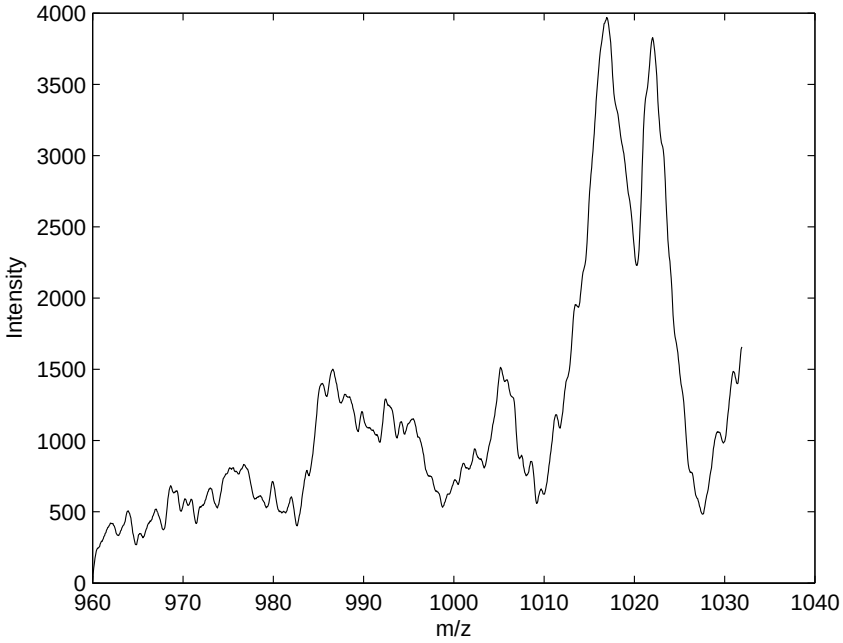


Figure 3.3: Unsmoothed protein spectrum

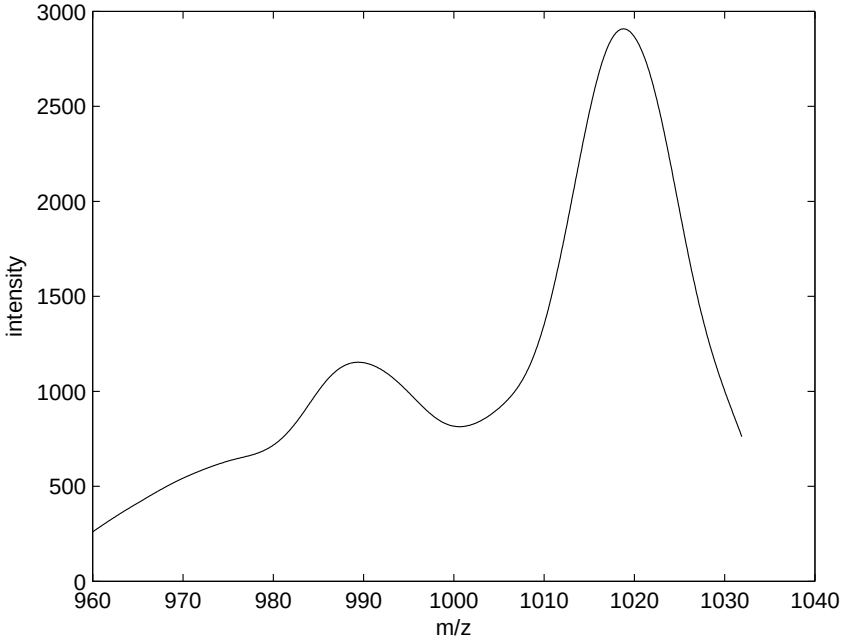


Figure 3.4: Smoothed protein spectrum

consists of 65363 m/z values. For large datasets, capacity becomes an imperative constraint. It is possible to decrease the size of the spectra, without loss of data, because the protein peaks generally consists of larger Dalton variability than the resolution the UltraFlexI has. This is explained in detail in [16]. It is therefore possible to summarize the data of various Dalton-values into one characterizing value. Figure 3.5 shows a zoomed region of Dalton values between 960 - 1000 Da.

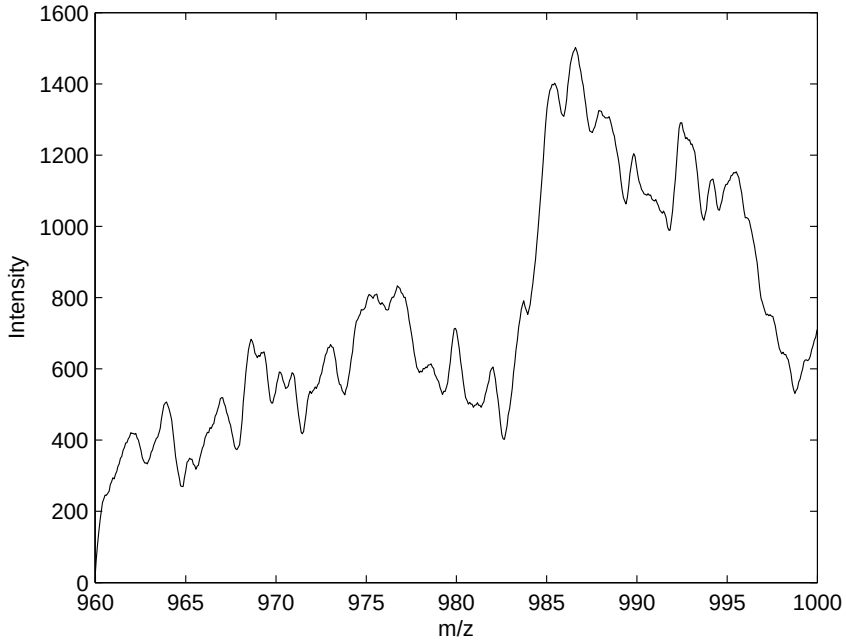


Figure 3.5: Unbinned protein spectrum

The original spectrum contained 561 Dalton values. Depending on the size of the bin, this spectrum can be reduced. For example, to decrease the spectrum to 38 Dalton values, a bin size of 1 can be chosen, resulting in figure 3.6. Next to data reduction, some initial noise reduction is a welcome side effect of this operation. This is not surprising, because essentially the objective of binning is to select the most representative Dalton value in a specific domain. This uniform binning works very well in practice. However, theoretically, the obtained spectrum has a nonlinear character (see equation 3.1). Binning should occur in a linear growing way, because of the first derivative of this equation:

$$\frac{\Delta D}{\Delta t} = -l \times 4eEs(t/l) \tag{3.10}$$

This coincide with the exported Dalton values of the UltraFlexI, which are increasing linear from a bin size depending on the resolution. The $m/z$ values in a bin width $\beta$ can be added together and divided by the number of $m/z$ values. In the DIPSTICC project, the interesting regions are between the 500 and 15000 Da. These limits are therefore default. A bin of 1 Da. is maintained as default $\beta$ value.

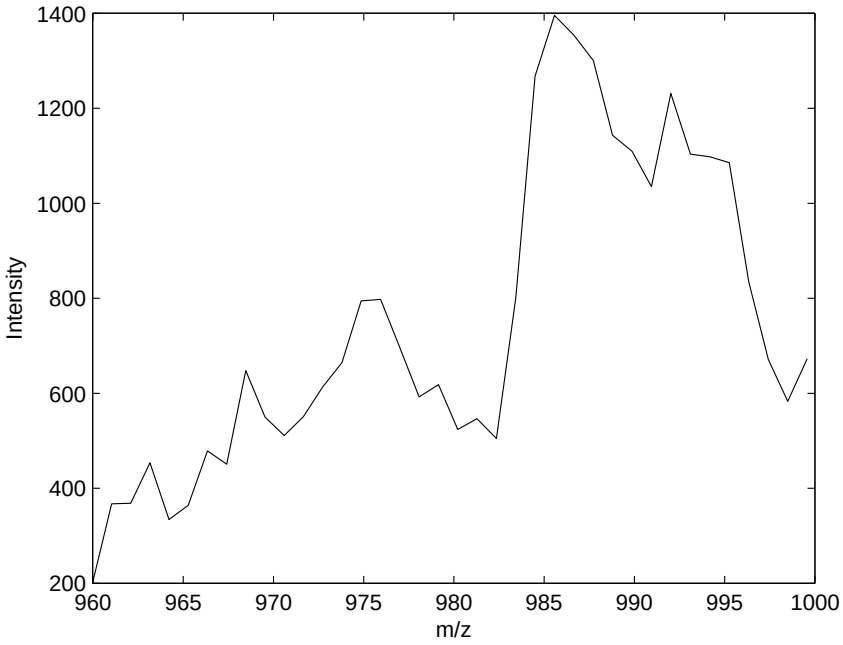


Figure 3.6: Binned protein spectrum

### 3.2.4 Baseline correction

Due to initial blast bias of the laser, the spectra might show severe baseline irregularities. It is a given fact that low molecular mass peptides ionise better than high molecular mass peptides or proteins. Since these are sole technical artifacts and they have little to do with the biological properties of the data, the baselines are preferred to be normalized along all spectra. Furthermore, classification based on the baseline must be avoided. To correct this baseline, it first has to be estimated and can then easily be removed by subtracting it from the original spectrum. For baseline estimation, the Whittaker smooth algorithm is used, described in chapter 3.2.2. Recall equation 3.6

$$Q = (y - ys)^2 + \lambda(\mathbf{D} \times y_s)^2 \tag{3.11}$$

To this equation, a weight parameter $w$ can be included to manipulate the extent of influence of spectrum resemblance. This results in:

$$Q_w = w(y - ys)^2 + \lambda(\mathbf{D} \times y_s)^2 \tag{3.12}$$

With $w$ a mapping of weights on $(y - y_s)^2$ . To obtain the smoothed function $y_s$ (compare with 3.7 and 3.8):

$$y_s = \frac{wy}{\mathbf{W} + \lambda(\mathbf{D} \times \mathbf{D})} \tag{3.13}$$

With $\mathbf{W}$ a diagonal matrix with on the diagonal $w$. To find the baseline, a highly smoothed representation of the spectrum has to be made, and this has

to be descended to the base of the spectrum. This is done using the mapping $w$. Values of $y$, which are lower than the smoothed function $y_s$, and therefore have a lower base than the smoothed spectrum actually represent, should be weighted lower in function 3.13 than values of $y$ which are higher than $y_s$. Now consider if $w = p$ if $y_s < y$ and $w = 1 - p$ if $y_s \geq y$ with $0 < p < 1$. A high value for $\lambda$, representing a very smooth baseline, and a low value for $p$, representing a close resemblance to the spectrums baseline, results in a series akin to the baseline of the spectrum(figure 3.7).

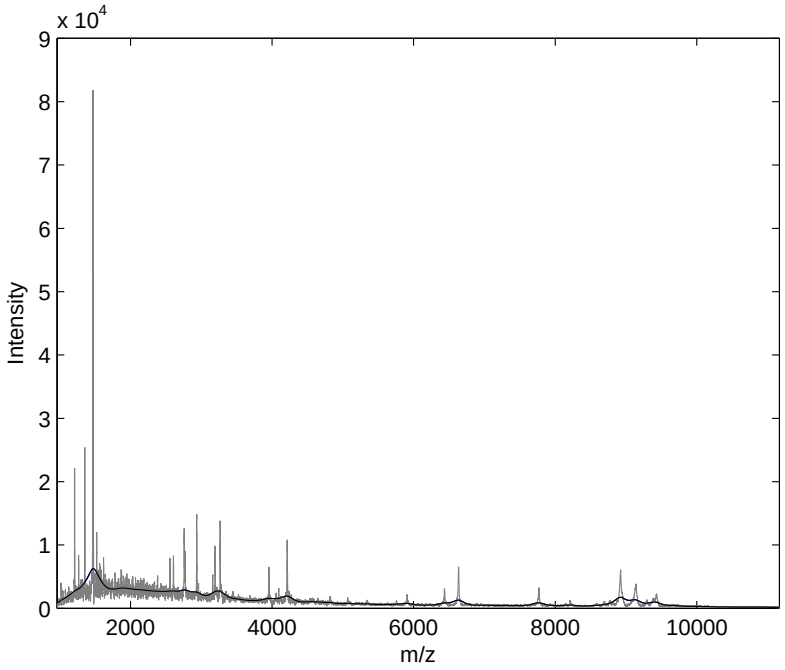


Figure 3.7: Protein spectrum with estimated baseline

The black line indicates the baseline, obtained by applying the method, described above, and the gray line indicates the spectrum. If the baseline is subtracted from the spectrum, the result (figure 3.8) is a fairly clean spectrum (i.e. a spectrum with baseline approximately zero). The default values are $p = 0.01$ and $\lambda = 10^7$, since there is strived to an optimal resemblance to the baseline of the data.

This procedure is called asymmetric least squares baseline estimation, and is thoroughly described in [17] and [18].

### 3.2.5 Normalization

Next to the previously described data processing, another important one is normalization. Since MALDI-TOF is a relative technique, the intensity says nothing of the concentration of the corresponding protein. For classification, the relation between peaks is the most important. The point of normalization is to reduce other effects than disease-specific artefacts. Several options to

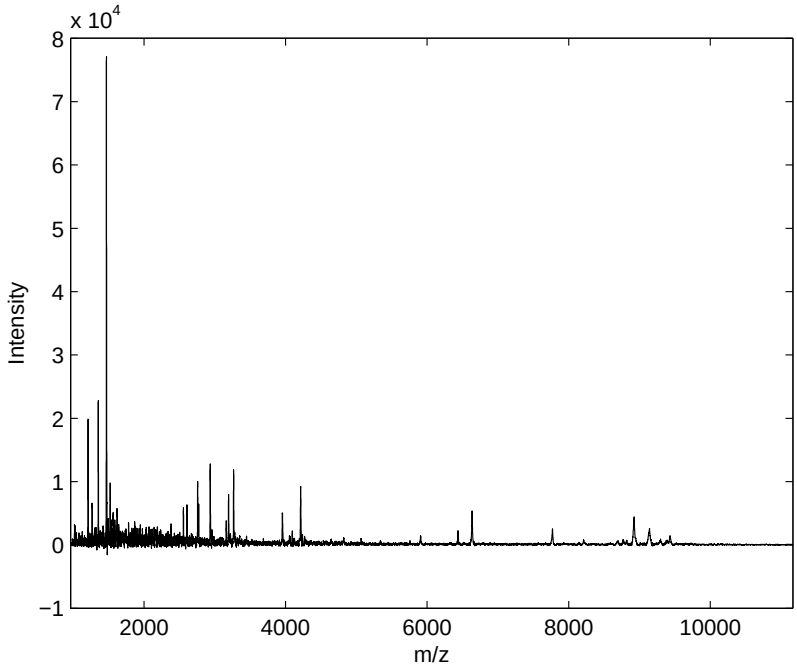


Figure 3.8: Protein spectrum with corrected baseline

this end should be implemented. The first is the vertical alignment issue. To compensate for overall intensity differences, the mean intensity value of the spectrum, or the median intensity value can be subtracted from this spectrum. Furthermore, the spectrum can be divided by its interquartile range. This is to reduce the influence of outlying samples on the classification. The possibility to take the ln transformation of the spectrum helps also to reduce the influence of rare high measurements, while retaining the form of the spectra, which is important for classification. The default values are: subtracting median, divide by the interquartile range and transform to ln.

## 3.3 Classification

To distinguish two different classes that are subjected to research, based on the information in the protein patterns, a model has to be trained, using unbiased and reliable example-data. This section gives a thorough description of the algorithm used for dimension reduction, Principle Component Analysis, and the algorithm used for pattern recognition in SPECTRA, Linear Discriminant Analysis. Furthermore, a description of Support Vector Machines is given.

### 3.3.1 Dimension reduction

Two interrelated properties of the data are the cause for the need of dimension reduction: the curse of dimensionality and the curse of data sparsity. The terms are explained below as is the algorithm used in SPECTRA, principal component analysis.

### 3.3.2 Curse of dimensionality

The resolution of protein pattern retrieval devices (e.g. the UltraflexI) reaches up to 20.000 peptides on a domain of 5000 Da. [16]. The domain of reliable protein detection of the UltraflexI is 0-300.000 [19], so data dimensionality can reach up to 1.200.000. Although in practice, the domain will be ± 500-15.000 Da, the dimensionality is still too high. For robust classification, the number of samples per feature needs to be at least 5-10 [20] [21], depending on the data and complexity of the classifier [22], while for MS data it is more in the range of 1/20 - 1/500. The phenomenon that there are too many features, considering the number of samples, is known as the curse of dimensionality [23]. To prevent unreliable data and ensure statistical significant classification, feature reduction is necessary.

### 3.3.3 Curse of data sparsity

Collecting data in medical research is often a complex activity with a lot of considerations to be made. Ethical (e.g. reporting cases of positive cancer diagnosis in control group subjects) and funding issues make it a delicate and time-consuming activity. Samples have to be accurately stored and documented. In order to retain the reliability and reproducibility, it is essential to standardize the sample acquisition and documentation and to maintain protocols [6]. All these factors contribute to the fact that data, collected in one study, cannot be merged with data collected in another study, without losing statistical integrity. This data sparsity is a foundation for the occurrence of the curse of dimensionality. Validation is more difficult because of this curse. No data can be wasted on a test set. Rules of evidence for cancer molecular-marker discovery and validation [24] even stated that half of the dataset should be used as test set, to achieve reliable results.

### 3.3.4 Principle Component Analysis

The foundations of PCA were developed by Pearson in 1901 [25]. It uses the principle, that most information is contained in the most variance. PCA derives linear combinations from the data that removes correlation and retain as much variance as possible [26]. Consider:

$$y_i = \sum_{j=1}^{d} a_{ij} x_j \tag{3.14}$$

or

$$\mathbf{y} = \mathbf{A^T x} \tag{3.15}$$

$a_i$ is a coefficient that maximizes the variance of $y_i$, with $y_i$ being the new variable, $x_j$ the original data, $d$ the dimension of the original data and $\mathbf{A}$ an orthogonal transformation of $\mathbf{x}$ so that $\mathbf{y}$ has maximum variance. Since the data may have a non-zero mean, and zero-mean is preferred, because of standardization perspectives, the mean must be subtracted from $x$ resulting in:

$$x_m = x - \mu \tag{3.16}$$

and

$$y_i = \sum^{d} a_{ij} x_{m_j} \tag{3.17}$$

We seek the set $a$ that maximizes the variance of $y$. To do this, we can use the equation from probability theory:

$$\begin{aligned} Var(y) &= E[y^2] - E[y]^2 \\ &= E[a^T x x^T a] - E[a^T x]E[x^T a] \\ &= a^T \Sigma a \end{aligned} \tag{3.18}$$

With $\Sigma$ the covariance matrix of $x$. Since we strive to find the orthogonal transformation $\mathbf{A}$, the additional requirement $a^T a = 1$ forms:

$$f(a) = a^T \Sigma a - \nu a^T a \tag{3.19}$$

It is now clear that to maximize the variance of $x$, the derivative of equation 3.19 has to be set to 0:

$$\Sigma a - \nu a = 0 \tag{3.20}$$

To solve this equation for nontrivial solutions, $\nu$ must be the eigenvectors of $\Sigma$. We can now conclude that to obtain $\mathbf{A}$, we simply need to fill the eigenvectors of $\Sigma$ in its columns, the first representing the first principal component and so on.

## 3.4 Linear Discriminant Analysis

Since SPECTRA is currently concerned with two-class problems, only binary LDA is treated in this section. SPECTRA's LDA is based on the approach of Fisher. The principle of Fisher searches for a vector $\mathbf{w}$ such that the within-class distance is a maximum, with respect to the between-class distance:

$$f = \frac{|\mathbf{w}^T(\mu_1 - \mu_2)^2|}{\mathbf{w}^T \Sigma_W \mathbf{w}} \tag{3.21}$$

where $f$ is Fisher's ratio, $\Sigma_W$ the pooled within-class sample covariance matrix, $\mu_1$ and $\mu_2$ the mean of the class $\omega_1$ and $\omega_2$ respectively. We can present $\Sigma_W$ in its biased form as:

$$\Sigma_W = \frac{1}{n-2}(n_1\Sigma_1 + n_2\Sigma_2) \tag{3.22}$$

with $\Sigma_1$ and $\Sigma_2$ the maximum likelihood estimate of the covariance matrix of the class $\omega_1$ and $\omega_2$ respectively, $n_1$ and $n_2$ the number of samples in $\omega_1$ and $\omega_2$ respectively and $n = (n_1 + n_2)$. If we want to maximize equation 3.21 to obtain a solution for $\mathbf{w}$, we have to differentiate the equation and solve it for 0:

$$F = \frac{\mathbf{w}^T(\mu_1 - \mu_2)^2}{\mathbf{w}^T\Sigma_W\mathbf{w}} \left\{ 2(\mu_1 - \mu_2) + \left( \frac{\mathbf{w}^T(\mu_1 - \mu_2)}{\mathbf{w}^T\Sigma_W\mathbf{w}} \right) \Sigma_W\mathbf{w} \right\} = 0 \tag{3.23}$$

Since we are only interested in the direction of the vector $\mathbf{w}$, we can leave out the scalars, proportion it to:

$$\mathbf{w} \propto \Sigma_W^{-1}(\mu_1 - \mu_2) \tag{3.24}$$

Now we have found a vector, that transforms the data in a lower dimensional space to obtain better linear separability, we must form a discriminator rule. We assign an independent observation $\mathbf{x}$ according to:

$$\mathbf{w}^T\mathbf{x} + \omega_0 = \begin{cases} > 0 \\ < 0 \end{cases} \Rightarrow \mathbf{x} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases} \tag{3.25}$$

We are left with estimating the threshold value $\omega_0$. Fortunately, proteomic data is normally distributed. This key characteristic is decisive for the solution, since we can now solve $\omega_0$ with:

$$\omega_0 = -\frac{1}{2}(\mu_1 + \mu_2)^T\Sigma_W^{-1}(\mu_1 - \mu_2) - log\left(\frac{p(\omega_2)}{p(\omega_1)}\right) \tag{3.26}$$

where $p$ is the prior probability of the class. Equation 3.26 can be derived from Bayes' rule. The linear discriminant analysis approach is thoroughly described in [27] and [28].

### 3.4.1 Classifier considerations

It is possible to create a model, that will fit the data better than LDA. Considering the high dimensionality, the data is probably better separable using for example a Quadratic Discriminant Analysis (QDA). The problem is that a more parameters have to be estimated, since the model will become more complex. This results in a reduction of the reliability. The classification results may improve, but they will be less reliable than the ones achieved with the LDA. Therefore, LDA is favored as it is the most simple (i.e. requires the fewest parameter estimation) case. It must be mentioned that no assumptions about data distribution is made in the algorithm described here. Since only the mean and covariance matrix are used as parameters, it does not matter what distribution the data has. One complex classifier is worth looking at in some more detail, since it proves to be very promising in proteomic data: the Support Vector Machine.

### 3.4.2 Support Vector Machine

The Support Vector Machine (SVM) is an expansion of the linear Support Vector Classifier (SVC). The SVC finds linear boundaries in the input feature space, so an optimal separating hyperplane is formed between two classes. The hyperplane is defined as

$$H(x) = x^T\beta + \beta_0 = 0 \tag{3.27}$$

Where $\beta$ is a unit vector. We consider the class of sample $x_i$ to be $y_i \in \{-1, 1\}$. To find the maximal separating hyperplane, we need to optimize the distance $C = \frac{1}{||\beta||}$ between the closest point from either class with respect to $H$:

$$\begin{gathered} \max_{\beta,\beta_0} C \\ \text{Subject to } y_i(x^T\beta + \beta_0) \geq C||\beta||, i = 1, ..., N \end{gathered} \tag{3.28}$$

This is a convex optimization problem, so we can minimize the Lagrange function with respect to $\beta$ and $\beta_0$:

$$L_P = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N} \alpha_i[y_i(x_i^T\beta + \beta_0) - 1] \tag{3.29}$$

We can derive that if $\alpha_i > 0$ then $x_i$ is on the boundary of the hyperplane. $\beta$ is defined in terms of a linear combination of the points $x_i$ that are on the boundary of the hyperplane. These points are therefore called *support vectors*. Using a kernel function (e.g. a polynomial kernel) on these support vectors, this case is easily extended to a non-linear one. This approach is also described in [29]. An additional advantage of this approach is that no need for data reduction (e.g. PCA) exists, since the algorithm finds the most important features itself.

## 3.5 Validation

To assess the model, obtained by the linear discriminant analysis, we need some kind of validation. Of course, the ideal form of validation would be to present an independent external dataset to the model. However, as is mentioned before, the data sparsity implies that another form of validation is sought after. The validation paradigm SPECTRA incorporates, is the Leave-One-Out Crossvalidation method. This algorithm is quite simple: training the model on $n - 1$ samples, with $n$ the number of samples and assessing it on the left-over one. The strength of the algorithm lies in the fact that no additional test set is necessary, since the model is validated on its own data. Furthermore, it reduces the chance of overfitting, since the maximum amount of data is used for training and testing the model. Although the method is quite computational expensive, it is implemented, since robustness of the model is preferred greatly over the time it takes to compute the model.

# 4

# System design

In this chapter, the design phase of the system is described. The information gathered in the analysis phase is transformed in models and directions. These will be detailed and defined enough to directly derive the implementation from.

## 4.1 Subsystem decomposition

SPECTRA is preordained to be used for modern research purposes (hence the name SPECTRA). This implies that the general scientific attitude towards the processing and classification of the input of SPECTRA, the protein patterns, is subject to alterations. No one is yet certain of the best way to process and classify these patterns. It is therefore feasible that SPECTRA supports the implementation of possible other and newer preprocessing and classification algorithms and datasets. This is a revolutionary feature in protein pattern analysis tools, since most tools are of a static nature and it is not possible to alter them. Users are dependent of beta releases and revisions to use improved algorithms. SPECTRA will encompass the ability to change to the users' benefit and desire. A segmented approach of the system forms a clearer interpretation of the modifications, performed on the data, which is also a requirement. The interfaces of these delimited components create an excellent basis for the use and application of other algorithms on the (modified) data. The components are denoted with a capital to emphasize their unique roll in SPECTRA. The four main components are directly derived from the requirement analysis.

*The Data retrieval part*
The Data retrieval part is responsible for the extraction of information from the folder structure and the assignment of the classes to the samples. Furthermore, it is responsible for the loading of saved data-files. The component is strictly bounded by containing only functions that are concerned with extracting this information. This is not trivial, because of the fact that the information of

exported data (see chapter 4.2.1) has to be extracted from folder structure. It concerns data aspects on metadata level, i.e. the aspects that define the data itself.

*The Preprocessing part*
The Preprocessing part is responsible for all spectra processing actions, prior to classification. These actions include: normalizing, smoothing, binning and correcting the baseline of the spectra and handling the spots. Furthermore, it is responsible for adding the preprocessing information to the data. It is bounded by containing the sole purpose of preparing the data for optimal and reliable classification.

*The Classification part*
The Classification part is mainly responsible for the classification of the spectra. Furthermore, it encompasses reducing the dimensionality of the spectra, using a principal component analysis and for performing a linear discriminant analysis on the spectra, to form a model that distinguishes the different classes.

*The Visualization part*
The visualization part is responsible for the visualization of the spectra. Different visualization modes are covered; heat plotting, normal plotting and correlation weight plotting. These modes can be combined with several options, being several plots in one graph, a negative plot and a average plot of several spectra.

Besides the functionalities described here, there are several more that do not specifically fall into one of the subsystems. Since they are used by two or more of the main subsystems and contain a specific functionality, they are separately grouped.

*The Menu part*
The menu part handles all menu activities, archetypal for Window menus. Exit functions, data export and import functions and help options fall under this functionality.

*The List actions part*
The List actions part covers all functions involved in the representation of the data on the screen. Besides that, data manipulations using the data list on the screen, like data entry deletion and a 'Select all' options are also covered.

*The Max window part*
The Max window part is the only functionality that communicates directly with the system. For interface purposes it is desired to run SPECTRA in a screen size as large as possible. This implies information on the current resolution of the system is needed. Since this counts for all separate screens of SPECTRA, Max window is a separate functionality.

*The Navigation part*
The Navigation part is responsible for the switching between the different SPECTRA main parts. Since this involves the managing of the interfaces between the subsystems (described in the next section), it is a functionality on its own.

*The Error handling part*
The Error handling part is responsible for managing errors. If an error occurs, it must be clear to the user what exactly went wrong. Error handling is concerned with the notification and the aftermath of the error.

*The Dialog part*
With SPECTRA, large quantities of data are concerned The technical limits of the system on which SPECTRA runs causes a delay from pushing the button until the results are obtained. Meanwhile, the user can become uncertain whether the system still responds, wonder how much longer it will take, or discover an error on his side. Cancelation options and progress visualization must be integrated in SPECTRA to handle these situations. The Dialog part takes care of this.

*The Dimension reduction part*
The Dimension reduction part is a bit hard to categorize. One can argue that it belongs to the Preprocessing part, since dimension reduction usually takes place prior to classification. However, it is a separate component in SPECTRA, since the visualization of the correlation weights of the different m/z values is required by the user. For that, a principal component analysis is needed. Since two main modules, namely Classification and Visualization use this algorithm, it is a separate functionality.

### 4.1.1 Subsystem interfaces

The subsystems, depicted in the previous chapter, interact with each other according to the diagram, presented in figure 4.1.

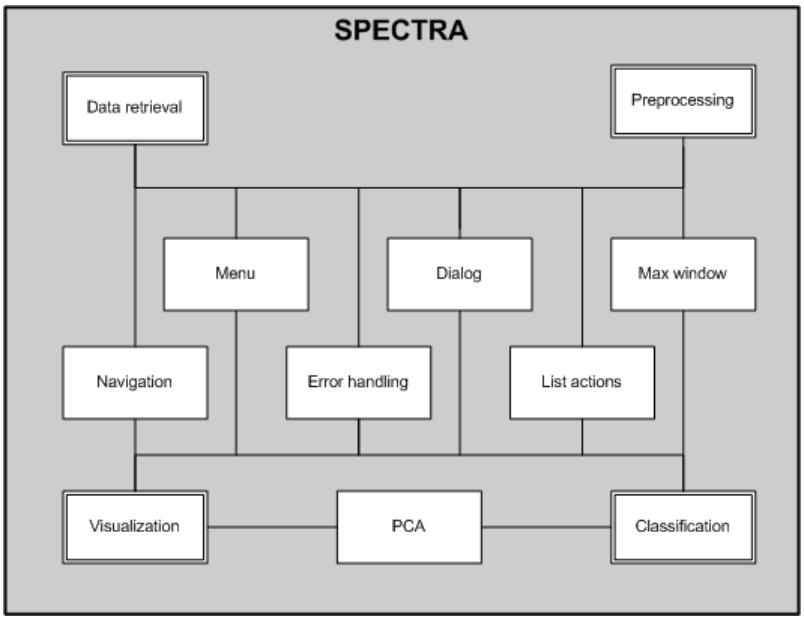


Figure 4.1: Subsystem decomposition diagram

The solid lines represent direct communication. The communication between the four main modules exists through the Navigation part, as described in the previous chapter. To realize this kind of dynamic program, described in the previous section, a simple interface between the subsystems is necessary. The dynamicity of the system is manifested in two ways. They are described in this section. Since only the main four modules are concerned with the interfaces (the other modules are not affected by new procedure integration) they are left out of the interface design. Another important note is that this section concerns only the interface *between* the main modules. It says nothing of interfaces between the user and the system.

### 4.1.2 New procedure integration

For the integration of other algorithms, a clear interface must be created for each module. This is an essential condition for the modularity requirement to work. When the interfaces are compatible and well described, it is easy to attach a new procedure, for example, an additional data manipulation before the preprocessing module, to the program, without knowledge or modification of the current system. Figure 4.2 clarifies this.

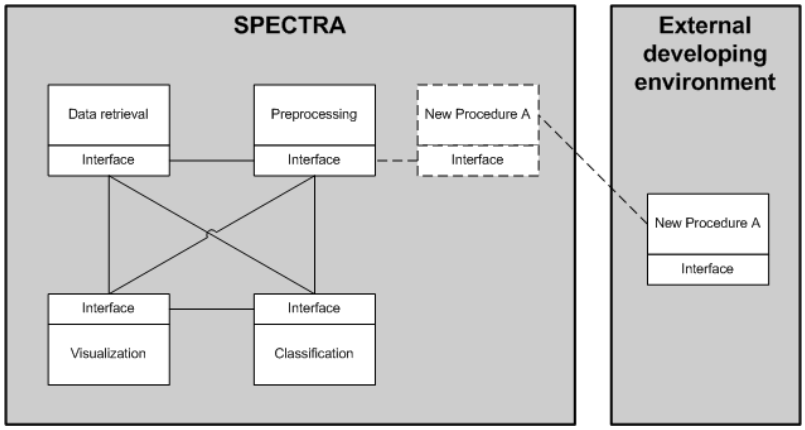


Figure 4.2: New procedure integration diagram

The communication between all modules, represented by the solid lines, occurs through fixed interfaces, represented by the 'Interface'-blocks. Since the interface is transparent, a new procedure, developed with an external application can easily be integrated in SPECTRA with the internal modules. This integration is represented by the dashed module 'New procedure A'. The new communication between the new procedure and the internal modules is represented by the dashed lines.

This kind of ingenious and highly compatible segmented design has a downside. Because of the clear interfaces, the user is not required to have any knowledge of the internal working of the modules, but only of the interfaces. This is disastrous for the requirement of transparency of the system. Great care has to be taken that this facet of the modularity of the system does not get in the way of the transparency of the system.

### 4.1.3 External procedures

Besides the compatibility of the system with procedural integration, through the segmented approach, the application has another, more implicit interface. Because of the requirement that data may be exported in Matlab or Excel format, the data can be modified externally from the application. After processing the data in an entirely different environment, the altered data can again be reloaded into the application, provided that the data has the required form (see chapter 4.2.5). This way, it is not necessary to implement a procedure in the application itself as is previously described.

## 4.2 Data management

Data management is an important issue in SPECTRA, since data export and saving and loading of previous work are requirements of the system. This chapter describes several key aspects of the data management in SPECTRA.

### 4.2.1 UltraFlexI data export

How does the UltraFlexI actually export the data? We know from chapter 2.2 and 3.1 how the device works physically, but the data export is yet (intentionally) unexplained. Before the protein patterns are exported, all locations on the target plate where the samples are applied (the so called 'spots'), are linked to the sample number, which are linked to an MS number on their turn. The MS-number is an automatically generated number that is unique for all samples *processed on the UltraFlexI*. Be aware of the fact that this is not the same as the sample identification number. There are different groups that use the UltraFlexI. The MS-numbers are the primary keys for the mass spectrometry database and the sample numbers are the primary keys for the researcher's database. The UltraFlexI generates the folder structure automatically, according to the inserted links, using the program AutoXecute. Generally, such a folder structure will look as follows:

> UpperFolders\MS-Number\Sample-Number\Spots\Proc-num\Mode\spectra.ext

UpperFolders can be any valid path under which the data is saved, e.g. 'C:\MS\myNobelPrize1'. 'MS-Number' is the automatically generated unique number from the mass spectrometry database. 'Sample-Number' is the unique number from the researcher's database, matched manually to the MS-numbers. 'Spots' refers to the location on the target plate, where the specific sample is applied. This can be more than one folder, depending on the number of spots one sample is applied to. For example, if one sample is applied on the first two spots on the target plate, 'Spots' contains two folders '0_A01' and '0_A02'. 'Proc-num' is the number of batches one sample is analyzed, (e.g. if one sample is analyzed twice, 'Proc-num' is '1' and '2'). 'Mode' is a predefined folder, stating the mode (linear of non-linear) of the UltraFlexI. It is beyond the scope of

this report to explain this further, but it concerns the way of ion flight propagation. We assume that 'Mode' is always '1Slin' or linear mode, although it is not of influence for the data retrieval. The final spectrum is located in the 'Mode' folder. Usually this will be the file 'xy.dat'. To induce flexibility, SPECTRA will implement a search algorithm that is independent of the name and extension of the file.

The observant reader must have raised a fundamental question after reading the above-mentioned state of affairs. Why go through all the trouble of retrieving the information concerning the spectra, when the information has all been manually inserted? Can this information not be used in SPECTRA, in the form of Microsoft Excel or Access files? This is certainly a point of interest, that has to be described in further detail. In the current state of affairs, the information is extracted from the folder structure, as is described in chapter 2.1. A huge advantage is that this saves a communication line between the analyst (who currently links the MS-numbers with the Sample-numbers) and the researcher and therefore reduces the chance on errors considerably. The researcher is not dependent on the possession of a list of MS and Sample-numbers. This brings us immediately to the next advantage: the data can be analyzed at any time. Data analysis can occur more quickly. These two advantages are the decisive factors to continue with the current way of data retrieval. Although this is not the most 'straightforward' manner, good automatization can overcome this problem.

### 4.2.2 Storage management

A consideration between speed and security must be made, determining what kind of storage management is preferred. Two possible options are flat file storage and database storage. The main advantage of flat file usage is the low abstraction level, which increases the speed of data transactions. Furthermore, it facilitates the interface and compatibility with other programs and users, since no extra actions have to be taken to extract the data from the database or link it to another database. The low abstraction level, however also requires security issues like concurrent file access and loss of data in case of system crashes. While database storage requires less security actions, the data transfer is slow compared to flat file storage. Since the capacity of the hardware at the LUMC is limited and there is no need for concurrent accessibility to the program, the flat file storage approach is preferred to database storage. To improve flexibility of the program the user must be able to store the work or export the data on a location of his choice. Therefore, an option must be included to browse through his available drives.

### 4.2.3 Current database structure

Currently, there are several databases involved in the process from taking the blood sample to analyzing the protein patterns. First, the location of the stored blood sample is entered in the 'Physical sample database' together with the LUMC patient number 'Pnr' and the number of the tube in which the sample is stored. This tube number and patient number are stored in the main database of the researcher. Each patient number is linked to a sample number 'Snr' that

is unique for every patient in the study. Multiple tube numbers can occur with one 'Snr', since more than one tube of sample can be acquired from one person. However, only one 'Pnr' can occur with one 'Snr', since the sample number represents a patient in the research database. This relation is one to one. The third database involved is the database from the mass spectrometry institute. Since they store every pattern from every study that makes use of their service, they have an own unique number for every sample, 'MS', that communicates with 'Snr'. Figure 4.3 clarifies the situation.

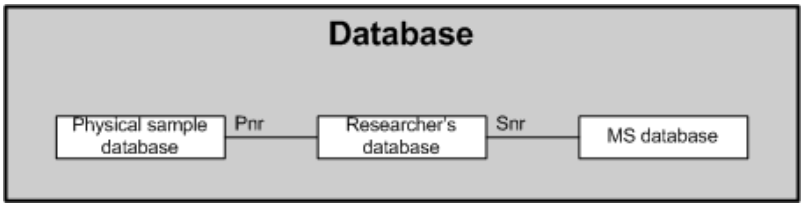


Figure 4.3: Relation diagram of the current database structure.

### 4.2.4 Ethics

There are two reasons for the somewhat elaborate database situation. The first is the interdisciplinary nature of the research. Since every department has its own database, and they are not interrelated, different unique numbers play a part. The second is the ethics involved in clinical research. A physician is bound by oath to privacy regulations. Therefore, he has access to private patient records. This information is important to the research, because the medical conditions of the patient determine to which group his protein profile is assigned. Furthermore, the chance is at hand that a control group person that was considered healthy, proves ill by way of being wrongly classified. This is a state of affairs to deal with by the physician only. Since above information is not accessible to every person involved in the research, different databases are developed, with different access privileges. It is vital for privacy regulations that the class assignments and study numbers are not traceable to patient names. A somewhat positive side-effect of this situation is, that all other parties operate without knowledge of the real condition of the sample, which improves reliability of the research, with respect to researchers bias (i.e. the situation where the control samples are treated differently than the patient samples, or manipulation of the outcome).

### 4.2.5 Internal data management

The execution time of the loading of spectra is expected to be the bottleneck of the waiting time. Since most preprocessing will be done on spectra with reduced dimensionality (i.e. after binning) the preprocessing time will be considerably less than the loading time. To verify this, a comparison was made between the loading time and preprocessing time of 31 samples. The results are presented in table 4.1. The loading time with larger datasets is expected to be even larger

in relation to the preprocessing procedures (chapter 6).

Table 4.1: Execution time comparison of 31 samples between different procedures.

| Procedure | Execution time (s) | Execution time on binned spectra (s) |
|---|---|---|
| Loading data | 18.427 | - |
| Binning data | 3.505 | - |
| Smoothing data | 25.326 | 1.362 |
| Baseline correction | 306.771 | 6.419 |

Since the size of the datasets is usual considerable with proteomics (see chapter 3.3.1), speed and capacity issues are very important for usability (e.g. waiting time) and the available hardware. The internal representation of the data must therefore be carefully considered. The specifications are described in the chapter 5.3. While a flat file approach is selected for the data storage, a possible future database approach is not unlikely. Concurrent accesses and increase of capacity would make the use of databases more appealing. Therefore, it is agreed that the internal data management must have a database interface. Based on the requirement analysis the following information must be extracted (partly from the UltraFlexI files, partly from class assignment and SPECTRA data manipulation). The MS-number, the unique sample number as it exists in the mass spectrometry database. The Sample-number, the unique sample number as it exists in the research data base. The Spot, the location where the sample substance is placed on the target plate. The Class, the class-number that is assigned to the protein pattern. The Location, the path to the location where the UltraflexI-file is stored. The Spectrum, the protein pattern, represented in binary ascii file. The Information, the modifications applied on the data, e.g. if the pattern is smoothed, this will be represented in the information, containing the smooth parameters. The following ER diagram can be constructed based on this information. It must be emphasized that the structure 'DATA' (note the capitals, they are used to distinguish the Matlab attribute from the original concept of data) also forms the interface to SPECTRA, described in chapter 4.1.1.

| **MS** | Sample | Spot | Class | Location | Spectrum | Information |
|---|---|---|---|---|---|---|

The MS-number forms the primary key. Although theoretically the same, it can be argued that the Sample-number should be the primary key, since that would represent the sample that is researched. However, in the current situation the MS-number is used for data analysis by the statistician, which is the decisive factor.

All data above can be extracted from the directory structure and the inner program functions. The 'Class'-field, however, must be assigned by the user. After the data, the class information must be loaded. According to the requirements, the class file is an Excel-file, with the following two-column structure:

| **MS** | Class |
|---|---|

Obviously, the MS-number is the primary key in this diagram. The data is represented in a Matlab cell array. Next to the DATA, there is another cell array that forms the interface together with the DATA, namely a cell array called 'HISTORY'. HISTORY has the following structure:

| x | y | contrast |
|---|---|---|

With 'x' being the m/z value vector, 'y' being the intensities and 'contrast' the contrast of the heatmap (1-1000) or 0 if the figure was a normal plot.

## 4.3 Memory issues

The high dimensional datasets bring an somewhat annoying but obvious side effect: they occupy a lot of physical memory. For example, one exported spectrum, measured linearly and containing 65000 Dalton values, covers over 1 Mb of memory. When a dataset of 300 samples is loaded, each applied 4 times on the target plate, it can reach up to 1.2 Gb of memory. That is quite a lot to load in one time! Again we reach a point where an important decision has to be made: what implementation language do we choose for SPECTRA. Two options are relevant in this case, the language C++ and Matlab. Both have their advantages over each other of course. C++ is very efficient considering the memory, with respect to Matlab, since Matlab loads its data in one time. However, Matlab offers a lot of functions that could be of great value to SPECTRA. Furthermore, the statistical department at the LUMC uses Matlab. It would therefore considerably simplify the design and implementation of SPECTRA if Matlab were used. Still, a solution to the memory problem has to be found.

### 4.3.1 Memory Saving Mode

Our salvation, if you can speak of one in this case, lies in the fact that we simply do not need all the information that is present in the spectra. This is thoroughly described in chapter 3.3.1. There are just too few samples to perform a reliable classification. Dimension reduction is therefore essential and, luckily, this reduces the occupied memory. Unfortunately, the reduction can only take place *after* the data is loaded. If somehow data reduction could take place *before* the data was loaded, this would solve our problem, or at least reduce it considerably. There is a way to accomplish this. The reduction method used (we are speaking of the binning method, see chapter 3.2.3) is independent of other samples. This means that we can apply binning on one sample. We steer the middle course of first loading, but loading a part of the dataset, and then reducing the dimensionality. However, a downside to this method exists. The preprocessing will take considerably more time, when each spectrum has to be loaded first. In large datasets we have seen that this is the proper solution, but for small datasets it can be fairly annoying. The user would prefer waiting a long time to load the data and then perform preprocess algorithms as quickly as possible for small datasets. Therefore, the Memory Saving Mode (MSM) must be facultative. When initiated, the MSM will load all information, except the

actual spectrum. The location of the data file is stored anyway, so a reference to the spectrum remains.

## 4.4 Use case diagrams

Now that the design of SPECTRA is gaining in shape, let us take a look at SPECTRA from the user's perspective. The visible changes of SPECTRA, when reacting to actions of the user, can be visualized through use case diagrams. Of course, a lot more happens in the system than is visualized in the use case diagram. It is only an abstract way of presenting the various actions of the users and the visible effects on the system. In chapter 4.1.1, the interfaces between the subsystems are defined, so four separate models can be devised, representing each of the four subsystems.

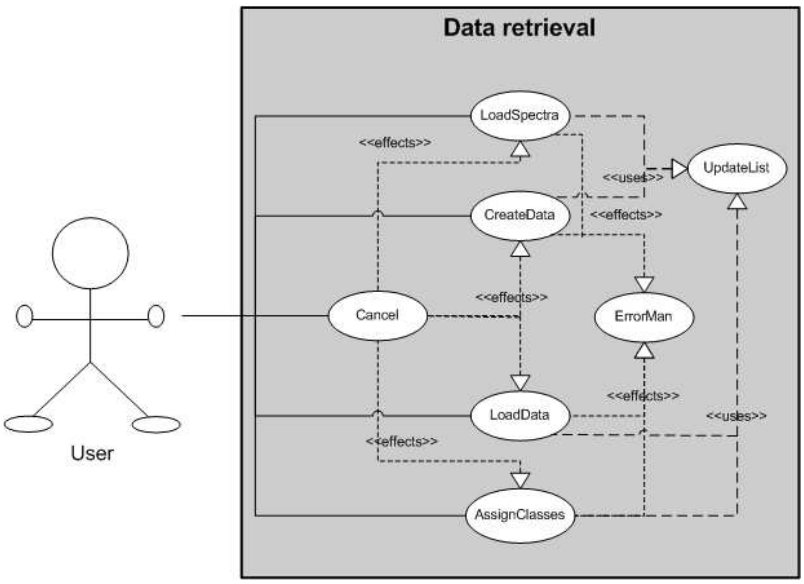


Figure 4.4: Use case diagram of the data retrieval subsystem.

Figure 4.4 represents the use case diagram of the data retrieval module. The different use cases show their communication with the user-actor. When the user initiates the 'LoadSpectra'-use case, the spectra associated with the retrieved data are immediately loaded. It might be feasible to yet load the spectra, when in MSM. Therefore, this action is only functional when the MSM is active and the spectra of the samples on which 'LoadSpectra' is applied are loaded yet. The result will be that the spectra are loaded for the selected samples. With the 'CreateData'-use case the user specifies the search criteria to collect all required information concerning the data and executes the search. This will result in a data list of the form described in chapter 4.2.5 (figure 4.5).

The user can choose whether the data has to be added to the current DATA, or the current DATA has to be replaced by the new data. In the 'Cancel'-use

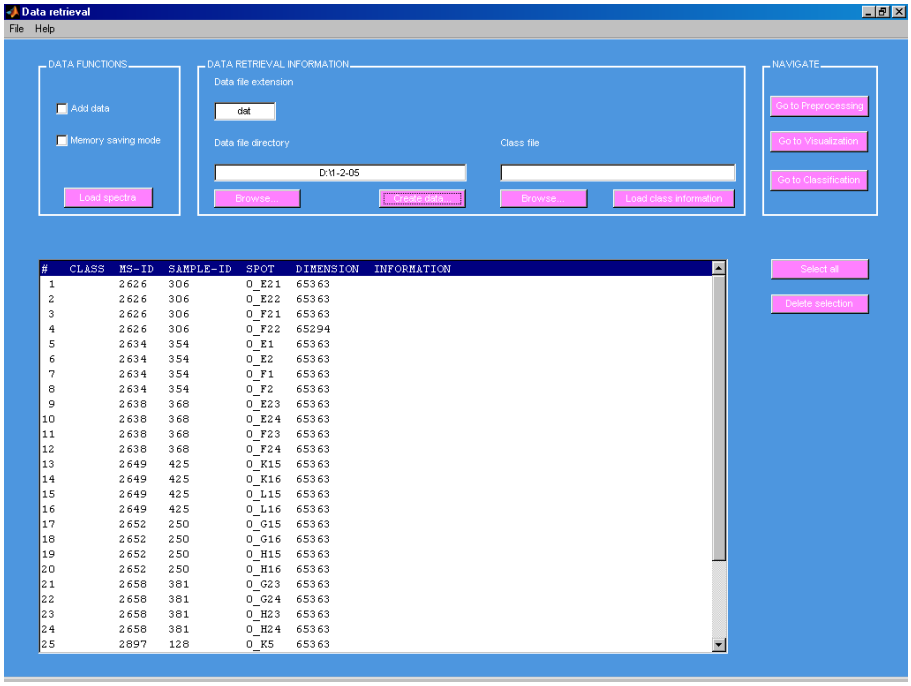


Figure 4.5: Screenshot of SPECTRA after data is created.

case, all current data retrieval processes are canceled (obviously). Since data retrieval is generally the most time-consuming operation, it is preferred the whole operation can be canceled in case of a slip instead of waiting the entire time until it is finished! Therefore, the 'Cancel'-use case effects all loading operations. The 'UpdateList'-use case refreshes the information concerning the data that is displayed on the screen. It is always addressed by the system after all loading operations. Therefore, it is used by all loading operations. Due to the fact that the system communicates with the user through this use case (by showing the data in the data list), it is present in this diagram. The 'ErrorMan'-use case handles error situations. In case of faulty input, e.g. incorrect parameters for the search criteria, the system explains to the user what went wrong through error messages and references to help files. Since all operations can throw an error, 'ErrorMan' is effected by all of them. The 'LoadData'-use case is invoked when users want to load previous saved DATA files, modified with SPECTRA. Contrary to the 'ExportData'-use case, which will be treated further on, the 'LoadData'-use case is the sole occasion for the data retrieval module, to draw a clear line around its responsibilities. The last use case in the data retrieval module is the 'AssignClasses'-use case. When the user wants to assign classes to the various (already loaded) sample entities, SPECTRA invokes this use case. The user has to locate a file containing sample-class tuples. SPECTRA assigns all classes to the different entities.

Figure 4.6 represents the use case diagram of the preprocess module. The 'ExecutePreprocess'-use case is invoked when the user wants to apply one of the implemented algorithms concerning data normalization on the selected DATA

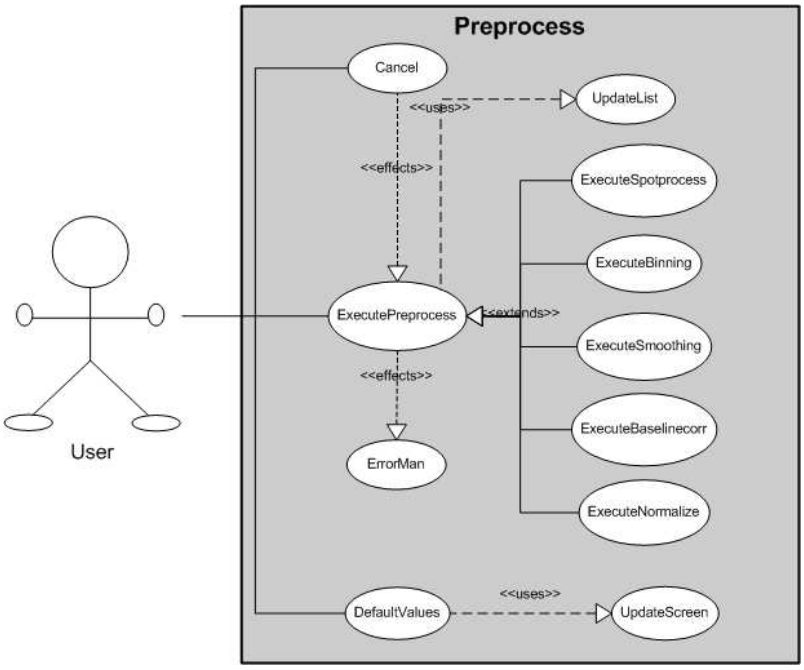


Figure 4.6: Use case diagram of the preprocess subsystem.

entries. Since several options are available, this use case is extended by the different preprocess options that follow. The 'ExecuteBinning'-use case is invoked when the user wants to perform a binning operation on the data, as is described in chapter 3.2.3. The 'ExecuteSmoothing'-use case is invoked when the user wants to perform a smoothing operation on the data, as is described in chapter 3.2.2. The 'ExecuteSpotProcessing'-use case is invoked either if the user wants to use all spots as different samples, or if the user wants to use the average of all samples, distributed over several spots, as one sample. The 'ExecuteBaselineCorrect'-use case is invoked if the user wants to remove baseline discrepancies from the data, as is described in chapter 3.2.4. The 'ExecuteNormalization' is invoked if the user wants to normalize the data, as is described in chapter 3.2.5. Since it is preferred to perform all desired actions with one push on the button (each action can take some time and the user decided that a long waiting time is better than several shorter ones), the user invokes the 'ExecutePreprocess'-use case to determine the sequence and actions. Figure 4.7 shows SPECTRA after the data is preprocessed.

The default sequence of these performance is: spot processing, binning, smoothing, baseline subtraction and normalization. Because of the flexible nature of SPECTRA, it must be possible to alter this sequence. This will be described more in-depth in the implementation chapter. After the preprocessing, the DATA displayed on the screen is updated, through the 'UpdateList'-use case. In case of error, the 'ErrorMan'-use case is invoked, handling all error issues. The 'ExecutePreprocess'-use case can be canceled with the 'Cancel'-use case. For fast data analysis, the use of default values can decrease the handlings needed for preprocess performance. To obtain the default values, the 'DefaultValues'-use case can be invoked. This use case resets all fields to prede-

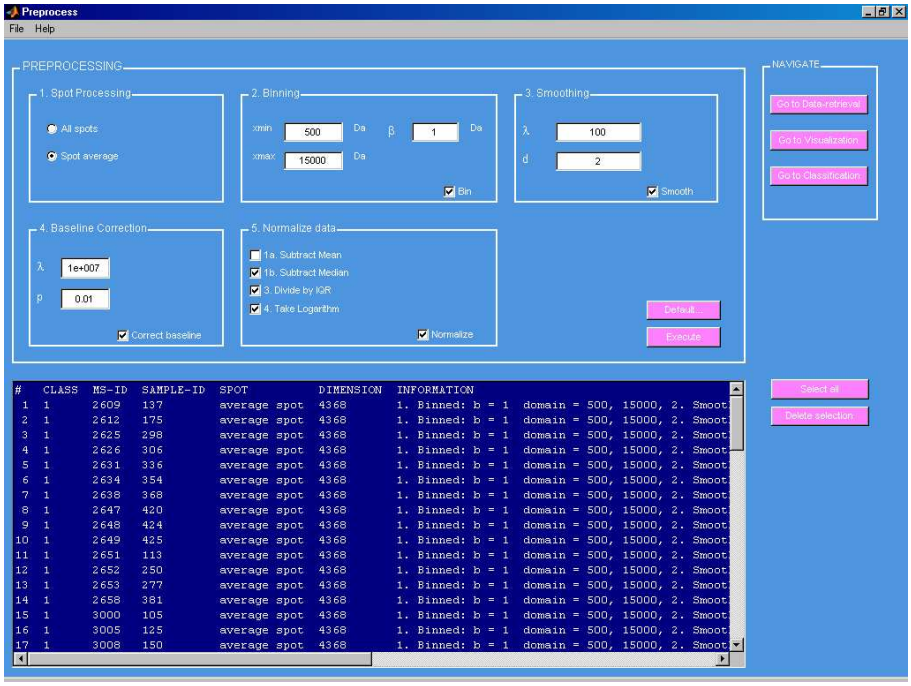


Figure 4.7: Screenshot of SPECTRA after data is preprocessed.

fined values. It calls the 'UpdateScreen'-use case to display these values on the screen.

Figure 4.8 represents the use case diagram of the classification module. The 'ExecuteClassification'-use case is invoked when the user wants to perform a linear discriminant analysis. Prior to this classification, the user has to choose whether SPECTRA has to perform an leave-one-out crossvalidation or use an external validation set. Furthermore, the user has the opportunity to perform a principal component analysis on the data. The 'ExternalValidation'-use case is invoked when the user wants to use a(n) (external) SPECTRA file for the validation of the model. The file must be loaded in the program. If the user wants to use leave-one-out crossvalidation, the 'InternalValidation'-use case is invoked. These two cases are a special case of the 'Validation'-use case. The results of the classification are displayed on the screen through the 'UpdateClassInf'-use case. Classification can take some time, especially when the 'InternalValidation'-use case is initiated. When the 'Cancel'-use case is invoked, the classification is interrupted. Since the classification is susceptible to errors, good error management is needed, through the use case 'ErrorMan'. When the user wants to use default values for the classification, the 'DefaultValues'-use case is invoked, displaying the values on the screen through the 'UpdateScreen'-use case.

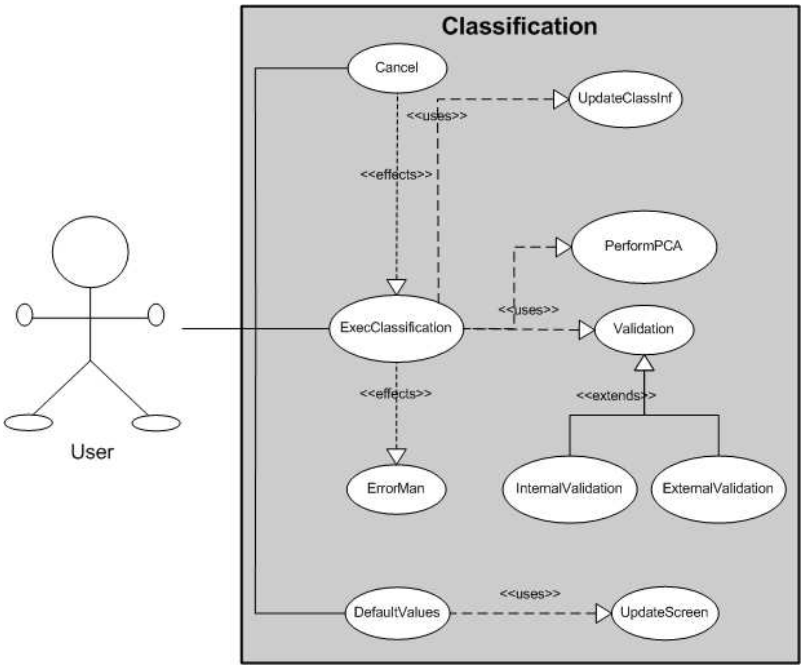


Figure 4.8: Use case diagram of the classification subsystem.

Figure 4.10 represents the use case diagram of the visualization module. The 'Plot'-use case is invoked when the user wants to view a graphical representation of the data. Since this can happen in three different ways, the 'Plot'-use case is a general case of three use cases. The 'NormalPlot'-use case displays a standard two-dimensional representation of the selected DATA entries, plotting the $m/z$ values against the intensities. With the 'HeatPlot'-use case a heatmap of the selected DATA entries is displayed. A heatmap is an image of $n$ equal thick lines with $n$ the number of $m/z$ values (figure 4.9). The lines vary in

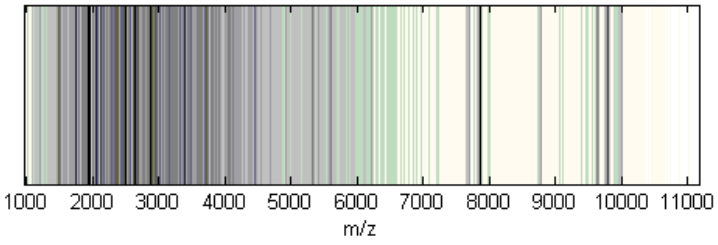


Figure 4.9: Heatmap of a protein pattern.

gray-value, according to the intensity (white being low and black being high in intensity). The 'CorrelationPlot'-use case is invoked when the user wants to visualize the weights of the correlations of the $m/z$ values after a principal component analysis, performed on the selected DATA entries. These forms of visualization can be combined with three options, that are therefore used by these forms. The 'MeanPlot'-use case is invoked if the user wants to view the average

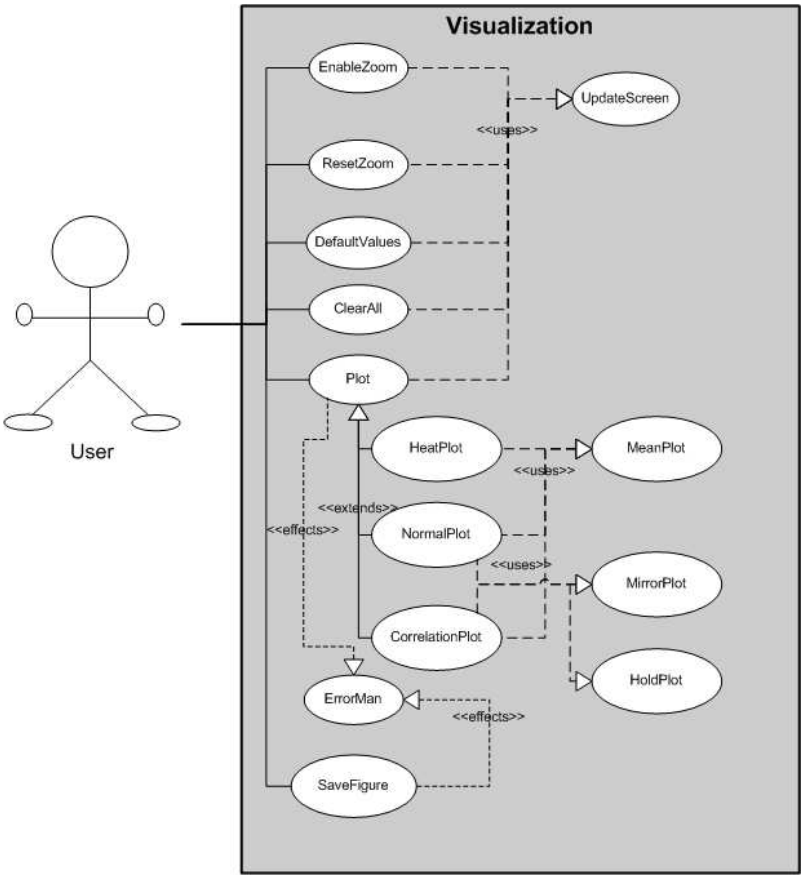


Figure 4.10: Use case diagram of the visualization subsystem.

of the DATA entry selection, or the average of the correlation weights. Instead of several plots in one figure at one time, the different spectra are represented in one average plot. The 'HoldPlot'-use case is invoked when the user wants to add another plot in the same figure as the previous plot. This can be feasible for data comparison. The 'MirrorPlot'-use case is invoked if the user wants to plot a negative representation of the selected DATA entries, i.e. the spectra is swapped around the x-axes. This too can be feasible for data comparison. Note that the 'MirrorPlot' and 'HoldPlot'-use cases do not hold for the heat plot, since this kind of image is not suitable for these actions. When the user wants to view a plot in more detail, the 'EnableZoom'-use case is invoked. When the zoomed representation is reset, the 'ResetZoom'-use case is invoked. Again, the 'DefaultValues'-use case is invoked when the user wants to return to the predefined values. All plots are erased with the invocation of the 'ClearAll'-use case. Since all these use cases affect the appearance of SPECTRA in the visualization screen, they all include the 'UpdateScreen'-use case. When the 'SaveFigure'-use case is invoked, the plot the user selected is saved to a designated location. If an error occurs during the visualization or saving, the 'ErrorMan'-use case is invoked to present the error details to the user.

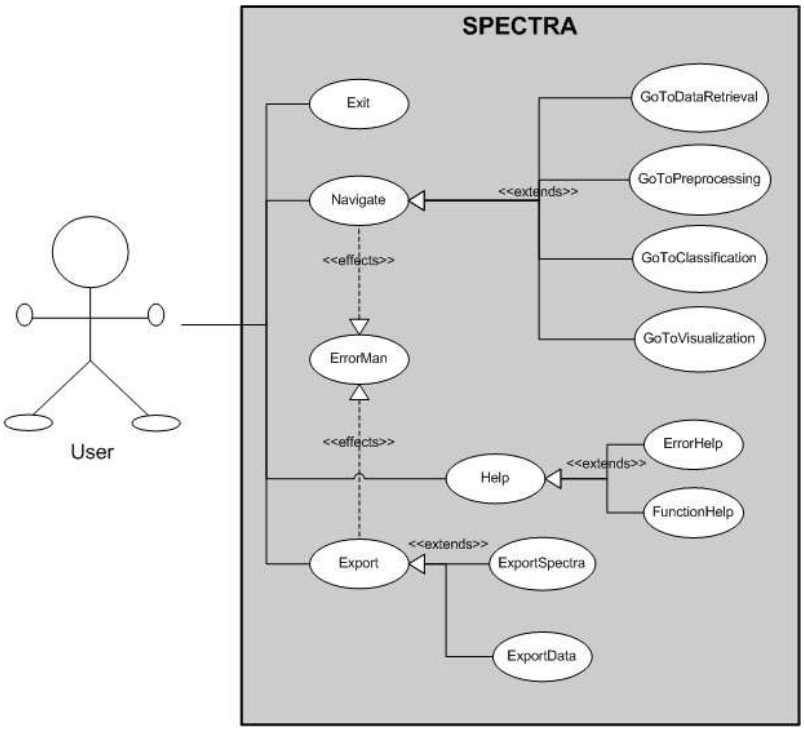


Figure 4.11: Use case diagram of the general actions.

After the assessment of these models, some functions seem to be missing. The rest of the modules, as described in the subsystem decomposition are encapsulated in one use case diagram. It presents all general use cases. 'General' in this context means that the diagram can be copy-pasted in two or more of the previous use cases. The exact sequence of performance is equal to the workflow, described in section 4.5. Figure 4.11 is separately developed, solely due to space considerations. The 'Exit'-use case is invoked when the user wants to close the application. The 'Navigate'-use case is a general form of four navigation use cases. When the user wants to access the data retrieval subsystem, the 'GoToDataRetrieval'-use case is invoked, opening the data retrieval subsystem and passing on the DATA and HISTORY. When the user wants to access the preprocess subsystem, the 'GoToPreprocess'-use case is invoked. The preprocess subsystem is opened and the DATA and HISTORY are passed on. The same goes for the 'GoToVisualization'-use case, when the user wants to access the visualization subsystem and the 'GoToClassification'-use case, when the user wants to access the classification subsystem. The transparency is proven to be very important in SPECTRA. Therefore, a help file with all the algorithms described has to be provided. When the user wants to view information on the implemented algorithms, the 'FunctionalHelp'-use case is invoked, displaying a the information about the algorithms. When the user encounters an error message, the code of this error is a reference to the help file. If the user wants to access the error explanation, the 'ErrorHelp'-use case is invoked, displaying the information about possible errors. Last, the matter of data export has to be raised. When the user wants to save his work in SPECTRA data format (so

that he can load the file again through the 'LoadData'-use case next time), the 'ExportData'-use case is invoked. The DATA will be saved on a user-specified location. To save only the spectra (to perform her own manipulations) in Matlab or Microsoft Excel-format, the 'ExportSpectra'-use case is invoked. The spectra will be saved in a user-specified location.

## 4.5 Workflow

To finish the functional design of SPECTRA, we are left with one thing: the sequence of procedures the user will execute with SPECTRA. Of course, the first thing the user will do is to load the data. Next the classes will be assigned to the various samples, using the appropriate Excel file. To visualize the data for quality control or general impression, the visualization screen is opened subsequently. Different visualization methods as have earlier been described are possible in this screen. To plot the correlation weights of a group of samples, the dimension reduction part will be called internally by SPECTRA. The preprocessing screen is opened next, to execute various preprocess algorithms. Usually the default sequence will be used, but , since SPECTRA is part of pioneer research, this sequence can be altered if desired. Finally, the classification screen is accessed to perform the actual classification. Prior to this classification, the user will perform a PCA, but this also is subject to alteration. Figure 4.12 represents the sequence on a high level.

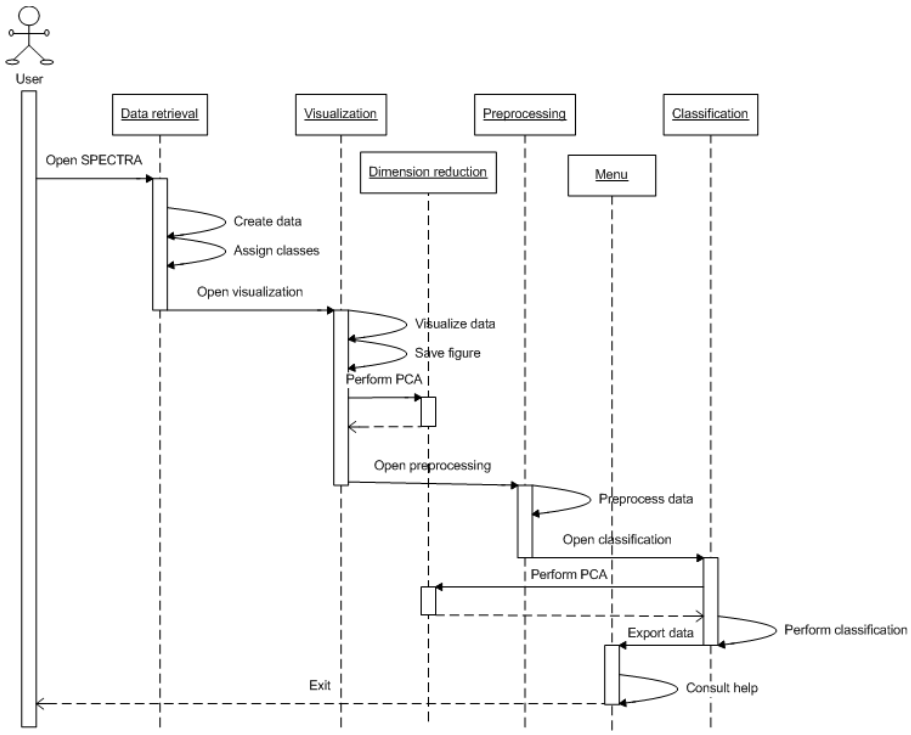


Figure 4.12: Diagram of the procedure execution sequence.

A side-note has to be made with this workflow. All screens can be accessed after the data is retrieved, since it might be feasible to, for example, first preprocess the data and then visualize it, or the other way around. Furthermore, classification must also be possible without preprocessing the data. No screen but the data retrieval screen can be accessed, however, when no data is loaded yet. Of course, the Help function can be consulted and data can be exported at any time. It must be mentioned that only the interactive parts of SPECTRA with the user are represented in the figure.

## 4.6 User interface design

Functionality of the system is obviously important, but it is also dependent on the interface of the application. In fact, it is one of the critical aspects, determining the frequency and user-satisfaction of the use of SPECTRA. The user interface is determined by the appearance of the system and the interaction of the user with the system. The two aspects are heavily related. The input methods of the system, e.g. determination of parameters, are for example a part of the appearance of the system, as well as a part of the interaction. This chapter describes SPECTRA's interface design, according to the procedure described by van der Mast [30]. All steps were thoroughly discussed with the user.

### 4.6.1 Metaphor

The first step to a well designed interface is the choice of a metaphor. In this context a metaphor is an already implemented device or existing situation on which the interface is based. The current data analysis in this case forms a profound basis for the interface. The experience of the user with Windows and Matlab was the decisive factor to maintain a metaphor of the Windows OS environment in combination with the interface of the Matlab environment. The appearance of the system and location of various implemented functions are similar to common Windows applications. The tool bar will be present in the top position. The figure manipulation interface is based on Matlab.

### 4.6.2 Interaction style

As is noted in the previous paragraph, the interaction style resembles that of the Windows OS. The point-and-click interaction method is intuitive and favored by the user. To reduce errors due to wrong input, typing possibilities are diminished and forced choices are preferred. To retain the ease-of-use and maximize the transparency of the system, the use of menus is discouraged, but unavoidable due to space considerations. To enhance the comprehensibility of the system, it is important that only the parameters of desired algorithms can be adjusted. If the user does not want to execute an algorithm, its parameter fields must be disabled, so the user is not bothered by them. Furthermore, the use of default values will increase the user friendliness and speed of use. If a user does not know which parameter to use for a certain algorithm, a default value will ensure the continuity of the analysis.

### 4.6.3 Mental model

The mental model of a user presents the way the user thinks the program will function. It is very important to take this into account, when designing the application. This way, the make-up of the program will be optimally adapted to the user's way of thinking and therefore decrease the time the user takes to get familiarized with the program. The mental model of the user of SPECTRA is exceptionally strong and clear, and is even a requirement for the program. As is already described in chapter 4.1, the program can be divided in four main parts. These parts roughly resemble the current way of analyzing protein patterns. The mental model of the user resembles the global structure of the program, i.e. first the data has to be retrieved, then the data has to be preprocessed and finally the data can be analyzed. Visualization of the data can occur at all times after the loading of the data. Of course, the Windows metaphor plays closely together with the mental model of the user. The user expects, for example, to find the 'Save' option under the 'File' menu item at the menu bar.

### 4.6.4 Error notification

To notify the user that an error occurred, error dialogues will be used. This kind of error management has several advantages. The error explanation instantly gives the user an active learning moment, since the user is instantly confronted with the nature of the error. A reference to the help file must be given, to supply further details. Furthermore, there must be some kind of reference to the input place where the error occurred, so the users knows directly where it all went wrong. If it is relevant, the designated input place will therefore become red, since this is considered the color of error. This approach is an alternative to, for example, disabling buttons until a proper input is supplied. That way, the user would be unaware of the nature of his error. Although the first requires substantially more work, it is absolutely worth the effort from the user's point of view. The error dialog consists of three parts: the error code, the error message and an 'OK' button for the user to confirm the notification.

### 4.6.5 Dialogs

Besides the error notification, there is another way SPECTRA communicates with the user. Through dialogs the system keeps the user updated on the current state of affairs. The use of dialogs is very important, especially in bioinformatic tools. Since the computation time can be extensive, the user might wonder whether the system has crashed. A dialog can prevent this, by informing the user of the progress of a procedure. In addition, a dialog can be used for cancelation of the procedure. When the user performs an (in retrospect) undesired action, it must be possible to cancel the action. Again, this is especially important when the action would take a long time to execute. Furthermore, it must not be possible to de-select or unfocus the dialog. This conveniently prevents the user from clicking in the current module screen out of frustration, and thus preventing system instability through subsequent interrupts. The dialog consists of three parts: a description of the current procedure, the progress of that procedure in percentages and the cancel button.

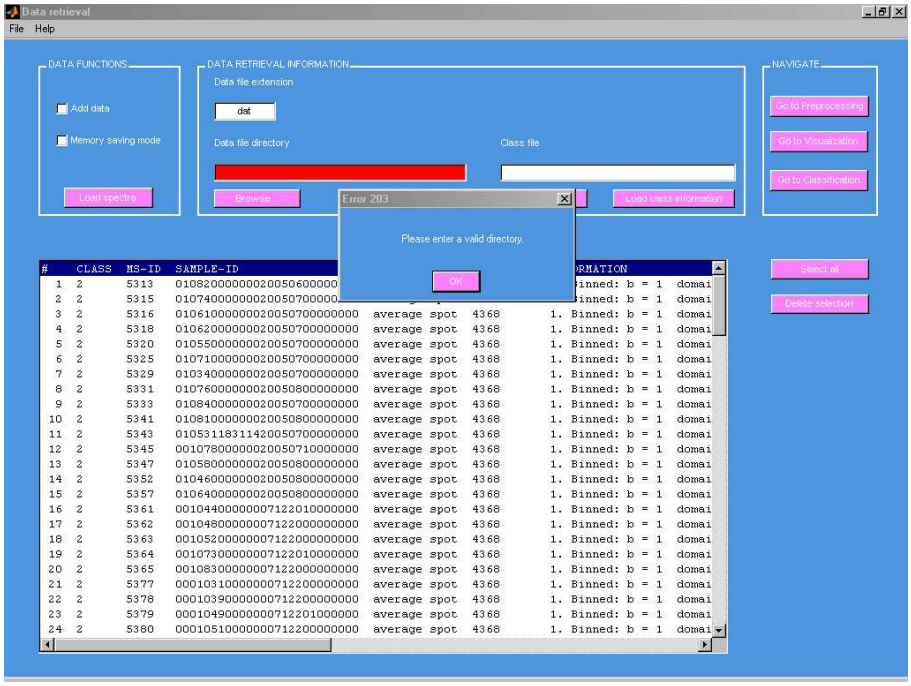


Figure 4.13: Screenshot of a SPECTRA error notification.

## 4.7 Help

For the sake of transparency and to provide an optimally user friendly system, special care has to be given to the help function, encapsulated in the Menu functionality. All algorithms must be explained here, and the user must get quick access to the information he desires. The information displayed in the help screen will be the same information as the User Manual provided, since both describe the entire functionality of SPECTRA. Since we are striving to get a dynamic program structure, we encounter one difficulty, or at least something that goes against our proclamation of the system being transparent. How can information on a new procedure be added to the help function? SPECTRA therefore contains a search function that seeks relevant SPECTRA help files. A help page can therefore simply be added. The only requirements are that the page must be in one of SPECTRA's help folders and preferably in an associated heading folder, that it must be in plain text format and that it must have the extension '.spe'.

## 4.8 User interface geography

Now that the way of interaction between SPECTRA and the user is analyzed, we can take a look to the actual appearance of the system. In close collaboration with the user, an appropriate layout for the four screens has been founded. The decision is made by evaluating various screen mockups of which several are

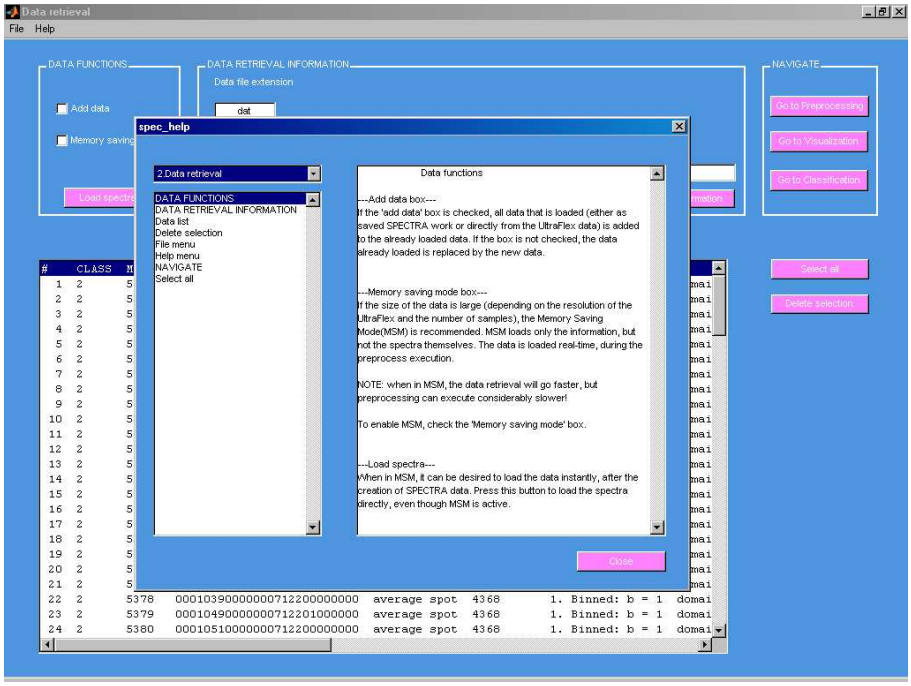


Figure 4.14: Screenshot of SPECTRA's help function.

presented in appendix B. Since the use cases offer enough information to derive the necessary interface controls (like buttons and lists), the only thing that distinguishes them is their geography. Note that according to the metaphor, the menu bar is consistent throughout all mockups. It is also important to keep in mind that these mockups were not created out of nothing. It was an iterative process, influenced by prototype evaluation. From the prototype analysis, special care has been taken for the color of the interface, the location of the classification lists and the appearance of the classification results.

## 4.9 The prototype

Though this section is placed at the end of the design chapter, the reader must realize that the location has nothing to do with the sequence of execution here. The development of SPECTRA is a circular approach. After the analysis and design phase, a prototype has been developed. This prototype has been evaluated with the user, and the conclusions have lead again to changes in the requirements and interface. The analysis, design and implementation aspects described in this document, are the final versions, formed after analysis of the prototype. If this document would be chronologically built, it would thus have to start with a prototype evaluation. This would have been a bit strange.

### 4.9.1 Prototype design

Based on the system design, the prototype presented here was an almost fully operational version of SPECTRA. It lacked the extensive help function (this was not yet entirely implemented), but the functionality of SPECTRA was entirely implemented. This approach has been chosen, so it could directly be used for two important analyses. One is the evaluation of the interface (the location of the buttons, the interpretable layout of the different screens etc.) and the other is the evaluation of the functionality.

### 4.9.2 Prototype evaluation

The prototype was analyzed using Camtasia from the TechSmith Corporation. Camtasia is a screen camcorder and video production tool and is ideal for monitoring the actions of a user on the screen. The user performed system tests, using an evaluation assignment (see appendix E). All user activities were recorded using Camtasia and the user was interviewed during the testing.

The user started by opening the program. When the data retrieval screen opened, the first remark was that the appearance was very dull. The loading of the protein spectra presented the first problem. It was not clear which button had to be used. This could probably be due to the ambiguous formulation of the task. The user canceled the operation after pushing the wrong button. After some explanation, the user was back in business. The rest of the data retrieval tasks went smoothly, except for the fact that it took some time, even in MSM. The user remarked, however, that it took only a short time compared to ClinProTools.

The next session covered the preprocess module. All tasks were accurately performed. During the execution, there was some discussion about the sequence of preprocess functions. To perform the functions in any other sequence than the default was considered somewhat laborious. However, it was decided that this was not critical issue, since most of the time, the default sequence will be used. The user was very satisfied with the progress dialog. It prevented her (as expected) from thinking the system had crashed. The export function was not easily found, even though the user's mental model was considered to extrapolate the Windows OS lay-out to the one of SPECTRA. This too was not considered a problem, provided that it is clearly described in the User Manual.

The user was very enthusiastic about the visualization options of SPECTRA. She performed the tasks effortlessly and was exalted by the various possibilities.

The classification module was considered clear and surveyable. One side remark was that the classification results were not emphasized enough. The user would like the results more on the foreground. Also, the data list took a more prominent place than the classification lists. This was considered a wrong order of importance.

The overall attitude towards the SPECTRA prototype was very positive. The user seemed very content with the prototype and its usability. It was considered fast regarding the current software. No functional changes were necessary. However, the interface was considered substantially too dull. Furthermore, analysis of the video log pointed out, that the user could not find the export and import functions, incorporated in the menu bar.

### 4.9.3 Conclusions

The evaluation of the prototype presented some directions for the eventual SPECTRA application:

- The primary point of criticism is the interface. The user considered it far too dull and required a more interesting color scheme.
- Little attention towards the functionalities is needed, since the user was satisfied with them, as they were implemented in the prototype.
- The location of the menu functions is not clear. However, the user determined they should not be altered, provided they are clearly described in the User Manual.
- The classification results must be more prominently presented.
- The classification lists (of the correctly and incorrectly classified results) must be more notably present than the data list.
- The use of dialogs to communicate with the user was applauded and considered a very good addition to the requirements.

# 5 Implementation

This chapter describes the actual implementation of SPECTRA. Since the design is ready, we now focus on the different functions and implementation aspects like memory and speed issues.

## 5.1 Implementing SPECTRA in Matlab

One of the advantages of programming in Matlab is that it offers a lot of predefined functions. Especially for the user interface and the visualization modules, this has saved a considerable amount of time. The entire user interface is created using the 'GUIDE' toolbox of Matlab. This is a visual programming environment, ideal for creating interfaces. The code of the interface component interactions is generated by Matlab. For example, the 'Callback' function interrupt is initiated by Matlab itself. Since the four modules are in essence the basis for the interface, the generated code provides a good framework to build SPECTRA on. Even though Matlab functions come in handy, a lot of additional programming work is needed to fulfill all requirements to the maximum. *It must be emphasized that from now on, all functions described are developed by the author of this report.* In case a Matlab function is used, this will be explicitly mentioned. The final program can be compiled to a stand-alone application. To deploy the program on multiple platforms, the Matlab program and Matlab Compiler license are still needed. Since the LUMC possesses these licenses, this present no problems.

## 5.2 SPECTRA architecture

Now the functionality of SPECTRA is determined and the user interface is also designed, we have all the ingredients for a detailed application lay-out. We

immediately encounter our first problem in programming with Matlab. The language is functional and the (my) preferred object oriented approach appears to end here. We cannot use the surveyability of class diagrams (one of the main advantages of object oriented programming) to gain insight in the application structure. Therefore, another way of structuring must be conceived. Fortunately, the issue of modularity has been stressed throughout the analysis and design. The attention given to modularity now bears its fruit. The four defined subsystems can now be used as basis for a more detailed structure. Previously, the global structure of SPECTRA was already based on the different functionalities. Consider the diagram in appendix F. The diagram represents different functionalities of the system, grouped in boxes. The first compartment of a box contains the name of the functionality, the second (if applicable) contains attributes used for communication between functionalities and the third contains the functions. The four subsystems, data retrieval, preprocess, classification and visualization, are located in the outer rims. The functions are grouped by functionality of the four subsystems. However, there are two cases in which two functionalities conflict. The 'Menu' functionality does not contain all menu functions in the application. Several saving and loading functions are encapsulated in the data retrieval and visualization modules. This decision has been made considering the functionality of the main modules, which is of more influence on the application and the lay out. Furthermore, the 'DimensionReduction' functionality is called by both the 'Visualization' and 'Classification' functionality, as is mentioned before. It must be mentioned, that the clear screen mockups provide a good basis for the interface. Therefore, the structure diagram provides only functional aspects of the program. All interface components are represented by the 'handles' attribute. Since these handles are only used within a functionality component, they are not present in the second compartments. The 'Callback' functions all concern the interrupt thrown by Matlab when the user activates an interface component (e.g. by pressing a button). The 'eventdata' argument of the 'Callback' function is an empty parameter, used by Matlab for future use. It will not be discussed in this report. Since it would be too elaborate to describe all functions in detail, only several interesting examples are discussed extensively. For further reference, please examine the source code of SPECTRA, which is available and thoroughly commented on the enclosed CD. The pseudo code of some functions can be found in appendix C.

### 5.2.1 The DataRetrieval functionality

The data retrieval functionality concerns all functions that retrieve information from the data, exported by the UltraFlexI as well as previously saved. When information is extracted from the folder structure, a bottom-up approach is needed. This way, the algorithm is independent of the directory where the exported folders of the UltraFlexI are. An additional requirement is that the spectrum files are found first. For this purpose, the 'foundlist = filesearch(rootdir, findfile, foundlist)' function is founded. To extract the information necessary for SPECTRA's data, the list of spectrum file locations is used through the function 'data = infExtract(foundlist)'. The output 'data' is of the internal data management form, described in chapter 4.2.5. To provide an overview of the DATA, it must be sorted. We chose to first sort on class

number and second on MS number, since SPECTRA will mainly be used to discern groups. The 'loadSpectra(hObject,handles)' function is concerned with loading the data when in MSM. Executing this function will yet load the data of the selected samples. The 'createData(hObject,handles)' function retrieves the data from the designated folder and presents the results to the screen. The 'loadClass(hObject,handles)' function loads the designated Microsoft Excel file and assigns the associated class labels to the samples. The conversion of the Excel file to a Matlab variable is performed using Matlab's 'xls-read' function. The 'browseSpecBut_Callback(hObject, eventdata, handles)' function displays an browse window (provided by the Matlab function 'uigetdir'), in which the user can browse to the designated data folder. The result is displayed on the screen. The 'browseClassBut_Callback(hObject, eventdata, handles)' function displays an browse window (provided by the Matlab function 'uigetfile'), in which the user can browse to the designated class assignment file. The result is displayed on the screen. Finally, the 'dataRetrieval' function creates the data retrieval interface. In the data retrieval screen, it closes the previous window, updates the list and calls the 'maximizeWindow' function.

### 5.2.2 The Preprocess functionality

The preprocess functionality incorporates all functions that concerns data preprocessing. The preprocessing is needed to remove noise and normalize the data. The 'Preprocess' function creates the preprocess interface. The 'err = errPrep(handles)' prepares the system for the preprocessing. It checks on errors concerning input parameters. The 'exPrep(hObject,handles)' executes the desired preprocessing algorithms on the data with the specified parameters and updates the DATA and the list. The '[data inf] = normHandle(handles, data)' function performs a normalization on the data if desired. The other preprocess functions work in similar way and are described in chapter 5.3.

### 5.2.3 The Classification functionality

The classification functionality includes all functions that are associated with the classification between two groups. The 'performLDA(handles)' function performs a linear discriminant analysis on the data. The results are printed on screen (figure 5.1).

It offers two possibilities of validation, namely using an external validation set, or an internal leave-one-out crossvalidation. The user can define the class priors and whether he wants to perform a PCA on the data prior to the classification. The number of principal components must be defined in this case. The 'browseBut_Callback(hObject, eventdata, handles)' function displays an browse window, using Matlab's 'uigetfile' function, in which the user can browse to the designated external validation SPECTRA DATA file.

### 5.2.4 The Visualization functionality

The visualization functionality integrates all functions that are involved with the different data visualization methods. The 'Visualization' function is used to create the interface. If the visualization screen has been accessed previously this

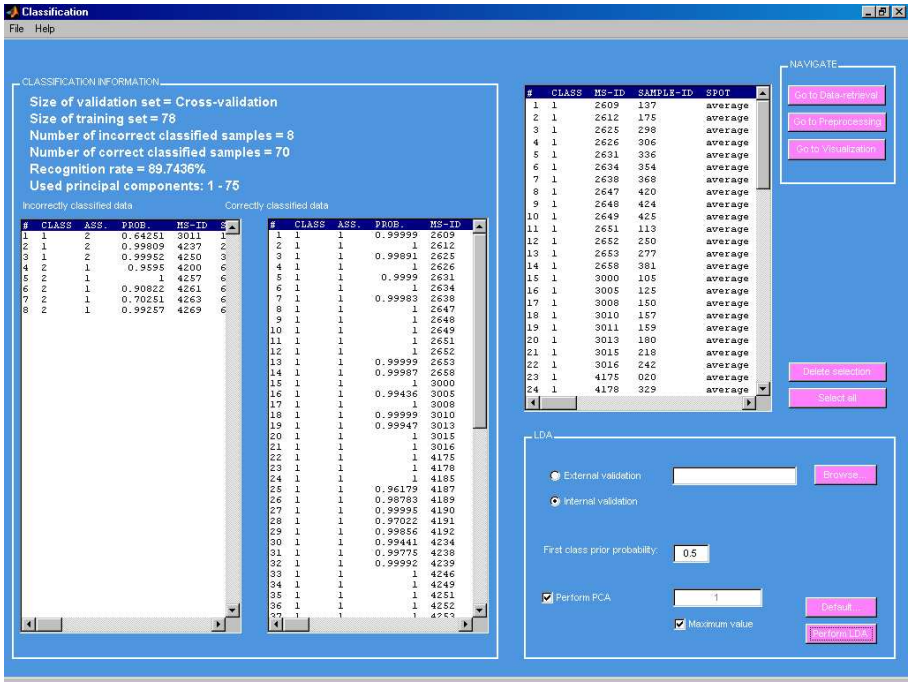


Figure 5.1: Screenshot of SPECTRA after data is classified.

section, it presents the 'history' on the screen. The 'history' are the (at most) last four images that are made with the visualization module. The '[errf specx specy vis way handles] = prepPlot(hObject,handles)' function prepares the data for the final plot function. It retrieves all data from the interface and checks on erroneous conditions. The heatplot cannot be used in combination with 'hold' or 'mirror'. Furthermore, graphs of different dimensionality cannot be displayed in one figure together. This has to be done consecutively. It also computes the mean if requested and calls the 'plotSpec(specx, specy, way, visop,hObject, handles)' function afterwards, with its retrieved parameters. The 'handles = plotHist(hObject,handles)' function moves the images a location up and delete the last one. This way, up to four images can be compared. This function plots the contents of HISTORY, so the figures are saved, when the visualization screen is closed. The 'plotSpec(specx, specy, way, visop,hObject, handles)' function executes the actual plotting on the screen. It calls the 'h = heatplot(x,y,con)' function when a heatmap is desired and the '[u,e,v]=spec_pca(x)' when the correlation weights are desired to be plotted. For the plotting, the Matlab 'plot' function is used. The 'h = heatplot(x,y,con)' function returns a heatmap image of the input spectrum, using 'con' as the contrast variable. The contrast values range from 0 to 1000. The function uses Matlab's 'image' and 'colormap' functions to achieve this. The figure is added to HISTORY. Figure 5.2 shows the visualization screen after from top to bottom a normal plot of one spectrum, a normal plot of five spectra, a plot of the first correlation coefficient vector of five spectra and a heatmap.

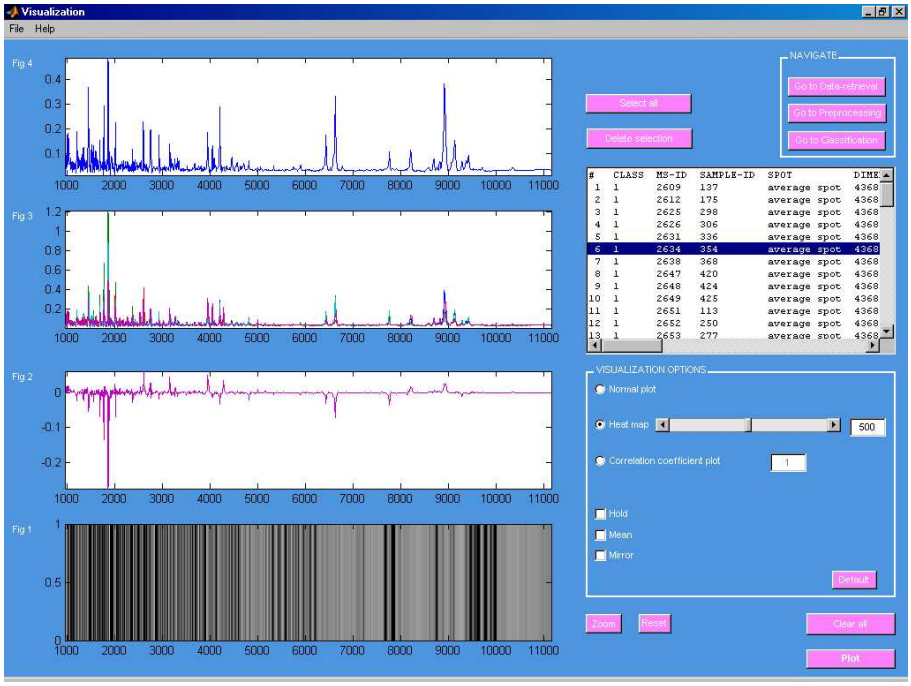


Figure 5.2: Screenshot of SPECTRA after data is visualized.

### 5.2.5 Miscellaneous functionalities

As is previously mentioned, the functions are grouped by functionality instead of e.g. interface or hierarchy (which does not really exist here). It is therefore quite possible that some functions are integrated in all four interface screens. Nevertheless, their functionality is the same, so they are described separately from the interfaces.

*Menu*

The Menu functionality is primary concerned with the data export, closing the program and the help function. The reason this name is given to the functionality is derived from the user's mental model. Since the metaphor of SPECTRA is based on the Windows OS, the user will expect these functions under the menu bar, where the functions are located in the Windows OS. SPECTRA incorporates two menu items: the 'File' menu item, containing import and export and program closure possibilities and the 'Help' menu item, containing the help function and error explanations. The function 'spec_menu(str,handles)' handles all menu possibilities and the 'spec_help' opens a new window. Furthermore, it retrieves the help information from stored files.

*DataListActions*

The DataListActions is concerned with all manipulations of the data, using the list. Deleting and selecting are main issues, but also the representation of data cells in an orderly fashion on the screen. Since the data is presented in a string, a monospaced font is used, to ensure surveyability.

*MaxWindow*
To ensure a maximal use of the screen space, without deforming the SPECTRA interface component organization, a separate function is devised. The 'maximizeWindow' function reads the resolution of the users monitor and organizes the window in this resolution. This way, the application is run nearly full screen.

*Navigation*
The Navigation functionality is concerned with the navigation between the four modules. This means correct propagation of the DATA and HISTORY cells and forcing the data to be loaded first, before the other modules (other than the data retrieval module) are accessed.

*ErrorHandling*
The error handling in SPECTRA is one of its strengths. Through the notification and labeling of different errors, the user is interactively learning about SPECTRA and how to use it. The error notification uses error codes to present the correct error message on the screen. This has the advantage that no redundant messages have to be created and that the error can be orderly displayed in the help screen.

*Dialog*
The dialog functionality consist of one function '[lab, rem)=spec_dialog'. This function produces a dialog screen with two label handles, 'lab' and 'rem'. These handles can be used to display the progress of a function in percentages, 'rem', and to display the current function that is executing, 'lab'.

*DimensionReduction*
As is mentioned before, the 'DimensionReduction' functionality is a separate function, since it is used by both Visualization and Classification. The implementation is described in chapter 5.4.

## 5.3 Implementation in depth

The functionality of SPECTRA is mainly determined by the implementation of the described algorithms. It is very important that the theory is implemented accurately, to ensure equivalence, and smartly, to ensure the procedure does not take more time than strictly necessary. Since a small deviation can cause considerable delays with such large data sets as with proteomics, the last aspect is important. Fortunately, the strength of Matlab among others is the support of matrix computations. This saves a considerable amount of time and working memory than using various iterations. In the Theory chapter we already adjusted the formulas in vector format. The pseudo code is included in appendix C. Especially the use of sparse matrices reduces the execution time considerably. Sparse matrices are very useful for adding similar values at one time, instead of looping through a vector, and for coping with vectors that contain few non-zero entries. The computational complexity of sparse matrices is proportional to the number of non-zero entries. Especially when eye matrices are used, or dimen-

sion reduction takes place, the data contains many zero elements, compared to non-zero elements. Matlab provides the function 'S = sparse(A)', that converts a full matrix A to sparse form S by squeezing out any zero elements.

Another matrix organization that reduces computing time considerably is the Cholesky factorization. This organization consists of only the upper triangle of the data matrix (and thereby reducing computation time and space). Matlab's 'C = chol(A)' function uses only the diagonal and upper triangle of A, while assuming the lower triangle is a transposition of the upper. However, a requirement is that X is positive definite. Fortunately, with mass spectrometry data this is the case, since no negative m/z or intensities can occur. In addition, if the Cholesky factorization of sparse matrices is used, no zero-elements can occur either. For the smoothing algorithm, this can be used for the differential matrix.

The last implementation aspect, is the implementation of the internal data management structure. We silently stepped over the fact that a Matlab cell structure is used. It is now time to justify this choice. In Matlab, two candidate structures are available: the Cell and the Structure. A structure is a high-dimensional Matlab array with data fields that can be accessed by textual name indexes. Cell arrays in MATLAB are multidimensional arrays whose elements are copies of other arrays. A huge advantage of a structure over a cell is that it is very transparent. One can easily access fields, using, for example, data(1).msnumber to retrieve the msnumber of the first sample in the data structure. This is a very unambiguous way of data storage and the chance an error occurs, is very low. Since the cell array is a very low-level form of storage, the retrieval is far less obvious. For comparison, to retrieve the same information from a cell, the code would be something like: data1,1 to access the actual msnumber, but data(1,1) to access the cell containing the data. This is far more susceptible to error. Furthermore, the computation time or size do not matter much. This is empirically tested, using a benchmark test. The creation and exporting of a dataset of 500 entries with 7 attributes was simulated 200 times. Table 5.1 represents the results.

Table 5.1: Benchmarking results cell versus structure

| | Cell | Structure |
|---|---|---|
| Average creation time | .417005 | .41976 |
| Average exporting time | .00380 | .00596 |
| Size in bytes | 2234000 | 2234480 |

Why choose a cell array as internal data structure then? The decision was made, based on the fact that the lower level structure makes it possible to use Matlab's cell functions to perform quick operations. This way, the cell structure *it* faster. Furthermore, it is easily converted in matrices, contrary to structures.

Another aspect that can help a great deal in the speed up of computation is the predefinition of cells. When the size of a future cell is known, it is best to define it in advance, instead of adding entries to a cell with size 0. In SPECTRA, this is possible, since the data is extracted from the data exported by the UltraFlexI. The number of cell entries therefore corresponds with the number of samples/files.

## 5.4 Data analysis functions

The actual data preprocessing and classification functions cover a large aspect of SPECTRA's functionality. They are therefore described in more detail in this chapter. Although the Theory chapter describes the mathematical nature of the algorithms, they still have to be implemented in Matlab. Since this chapter treats the implementation aspects of SPECTRA, it is decided to grab back from here to the described algorithms in the Theory chapter rather than discuss the implementation there. Let us start with the binning algorithm. From the Theory we can derive five parameters. The minimum m/z value, 'xmin', and the maximum m/z value, 'xmax', to determine the domain. The initial bin-width in Dalton, the rest of the bins can then be calculated. The m/z values of the spectrum, 'x', and the associated intensities, 'y'. After cropping the spectrum to the desired domain, the number of m/z values in the bin-width is calculated. Since the bin-width has to increase linear, the square root of 'x' is used to calculate a multiplier for the exact bin-length and the bins are then created. Next, all intensities are summed per bin. We use

```
s = nonzeros(full(sparse(k,1,y,m,1)));
```

with 'k' the new bins, 'y' the intensities and 'm' the number of bins. This instantly shows the power of the Matlab sparse function. The variable 's' is divided by the number of equal m/z values per bin to obtain the average m/z values per bin.

The smooth function has four parameters: the 'lambda' and 'order' parameters of the smoother function and the spectrum, represented as 'x' and 'y'. The complexity of the function depends on the order parameter. When a first order smoothing is desired, it will take less time to compute than a higher order, since it cannot be analytically computed, but a recursive function has to be used. First, the order-difference matrix 'D' has to be determined. A separate function Dd(x, order) has to be devised. If 'order' is zero, 'D' is a sparse eye matrix. Else 'dx' is the order-differential of 'x', 'm' dimension of the data and

```
V = spdiags(1 ./ dx, 0, m-order, m-order);
D = V * diff(Dd(x, order - 1));
```

Again the sparse diagonal matrix function spdiags expedites the process. After 'D' is calculated, the following piece of code creates the smoothed data (compare equation 3.8).

```
C = chol(E + lambda * D' * D);
ys = C \(C'\y);
```

The baseline procedure actually exists of the subtraction of a highly smoothed representation of the data. The besides the parameter 'lambda' the parameter 'p' represents the spectrum likeliness. The previous two lines of code, have to be integrated in a iteration, with an addition of the spectrum resemblance diagonal matrix 'W' (see equation 3.13). W is adjusted as long as the smoothed representation of the spectrum is outside of the tolerance factor 'p'.

```
while repeat > 0
   W = spdiags(w, 0, m, m);
   C = chol(W + lambda * D' * D);
   z = C  (C'  (w .* y));
   wold = w;
   w = p * (y > z) + (1 - p) * (y < z);
   repeat = sum(wold  = w);
end
```

The PCA procedure consists of two cases: the singular and regular case. The function has three output parameters: a vector of mean- -centered normalized principal components 'u', a column vector of eigenvalues 'e' and a matrix of principal component loadings 'v'. In both cases, first the intensity values must be normalized by subtracting the mean intensity value (see equation 3.16). In the regular case, 'v' and 'e' are determined using Matlab's 'eig' function. In the singular case, 'u' and 'e' are determined using Matlab's 'eig' function. The 'u' is calculated in the regular case using

```
u=ym*v./(ones(n,1)*sqrt(e'));
```

With 'ym' being the normalized intensity vector and 'n' the dimensionality of the intensity vector in the regular case. In the singular case, 'v' is calculated using

```
v=ym'*u./(ones(p,1)*sqrt(e'));
```

With 'p' being the dimensionality of the data in the singular case.

The LDA procedure has five output parameters. 'cls' is a vector with the classes assigned by the model to the test set, with probability 'prob' and density 'density'. 'coef' are the model coefficients, 'w' and '$w_0$', and 'dim' is the dimensionality of the data in which the classification took place. There are five input parameters. 'train' are the intensity vectors of the training set and 'test' the ones of the test set. 'class' is the vector of the sample classes, assigned by the user and 'priors' the prior class probability. 'dim' is the dimensionality in which the classification takes. First, the pooled within cross-products are calculated (compare equation 3.22)

```
Sw=Sw+(n1-1)*cov(train1);
Sw=Sw+(n2-1)*cov(train2);
Sw=Sw/(ntrain-2);
```

With 'Sw' is the pooled within-class sample covariance matrix $\Sigma_W$, 'n1' and 'n2' the number of samples in class 1 respectively class 2, and 'train1' and 'train2' the training samples in class 1 respectively class 2. The Matlab function 'cov' calculates the covariance matrix. We calculate the eigenvectors 'v' and eigenvalues 'e' from 'Sw' and compute the discriminant function coefficients (compare equations 3.24 and 3.26), with 'mn' the vector of intensity means.

```
coef=(mn(1,:)-mn(2,:))*v*diag(1./e)*v';
coef=[log(priors(1)/priors(2))-0.5*coef*(mn(1,:)
+mn(2,:))'  coef];
```

# 6

# System assessment

Now SPECTRA is fully implemented, it is time for the final tests. The assessment consists of several parts. First, the user satisfaction is the most important measure of success. Extensive testing in cooperation with the user is therefore required. Second, since the user does not want to test all extreme situations herself (that is to say, it is not efficient to do that), we can test some aspects ourselves. The functionality correctness, as well as the robustness, do not necessarily depend on the user himself, so they can be tested by the developer. This chapter describes several tests and results obtained by them. The user has tested the SPECTRA using the same task list as with the prototype (appendix E). The reason these tasks were used is to see whether the prototype has adapted satisfactorily to the user's wishes. For comparison, this is the most reliable method. Furthermore, since no large functional changes have been made, no additional tasks could be performed. The emphasis with the analysis of the prototype lay primary on interface aspects. In this evaluation, it lies more on the functional results. It must be remarked, that all tests were performed on a Pentium III with 480 Mb RAM and a 2.4 GHz CPU.

## 6.1 Comparison with prototype

The largest point of dissatisfaction with the prototype was the interface. It appeared very dull to the user. This time, the user was enthusiastic about the new interface. It was much more appealing and clear to the user. Though the locations of the export and loading options were not changed, the user could find them directly this time. This was of course due to the fact that she had learned since the last time. However, the user was satisfied to find the options thoroughly described in the user manual. Furthermore, the interface of the classification screen was considered far more clear than with the prototype. The user could find the results, after performing a classification instantly this time. The lists with correctly and incorrectly classified samples was prominently present (in contrast to the data list) and the bold and larger fontsize ensured a

very striking classification result area. The user had no further remarks on the eventual interface, compared to the prototype.

## 6.2 Functional test results

After evaluating the interface, the functionality of SPECTRA was tested. Of course the user is interested in the functionality, but she is less concerned about the actual correctness (i.e. she assumes it is correct). Therefore, the functional correctness of SPECTRA was partly assessed by the developer. Assessing correctness is not as easy as it seems, for what *is* correct? How can we be sure, the results presented by SPECTRA are the ones that we want them to be? For this end, we need some kind of standard, or correct answers to tasks SPECTRA has to perform. The golden standard used here are the results, as produced by the statistician, on the colon cancer versus control experiment [5]. It concerns data of the second week. First, a preprocessing on the data was executed. The mean of the spots was calculated, followed by a binning (domain 1160-11600 Da., linear increasing bin width, starting with 1 Da), a smoothing ($\lambda = 100$ and $n = 2$) and a baseline correction ($\lambda = 10^7$ and $p = 0.01$). The data was normalized by subtracting the median, dividing the result by the inter-quartile range and taking the natural logarithm. The results were exactly the same as the ones obtained by the statistician. Since the algorithms are deterministic with respect to their parameters (i.e. with the same parameters and input, the result will always be the same), the algorithms of the statistician are assumed to be correctly implemented. Finally, the results of the classification were assessed. They are displayed in table 6.1.

Table 6.1: Classification test results

| | **SPECTRA** | **PAPER** |
|---|---|---|
| **Sensitivity** | 82.6% | 80.6% |
| **Specificity** | 96.7% | 97.1% |
| **Total recognition rate** | 90.3% | 88.8% |

The results of SPECTRA are fairly comparable with the results in our manuscript, that is recently accepted for publishing in the European Journal of Cancer. The slight deviations are probably due to the fact that the validation methods are separately coded. There is no description of the (single) cross-validation method that is used by the statistician.

It is also exciting to see how SPECTRA performs on a new dataset. For this test, we used a set of 84 serum protein profiles of patients with benign bowel diseases[1]. It is interesting to investigate whether protein patterns from healthy persons can be distinguished from protein patterns of patients with a benign bowel disease. Figure 6.1 shows the mean protein spectrum of the healthy

[1]Benign bowel diseases are defined by all bowel disorders that are not cancer. It concerns a broad collection of diseases with different pathogenesis.

persons and the negative mean protein spectrum of the patients suffering from benign bowel disease for comparison. The colonoscopy, like with colon cancer, is currently the golden standard for diagnosing these diseases. This is an invasive and relatively expensive method for diagnosis.

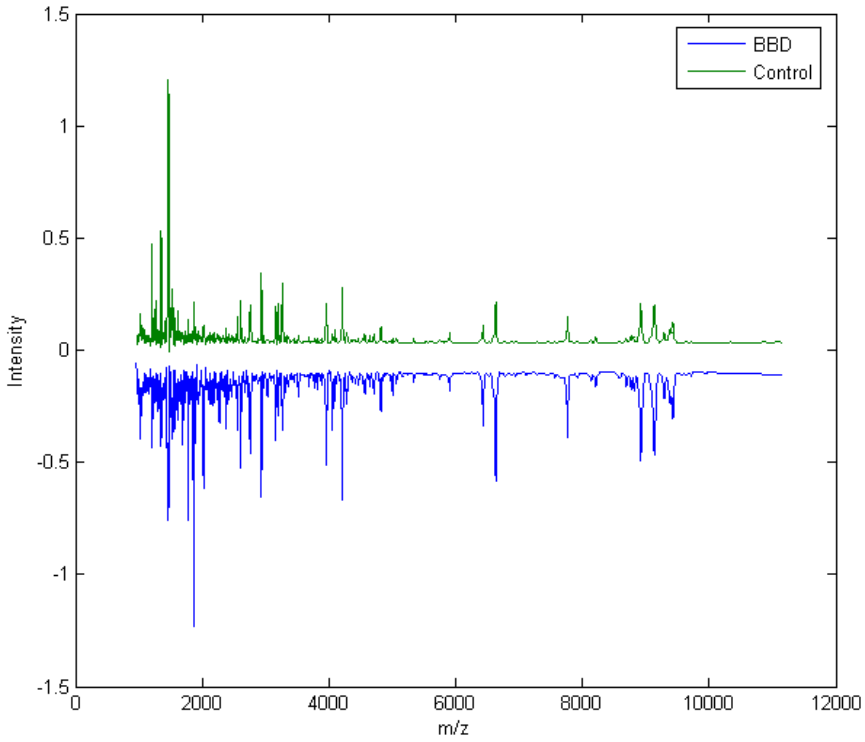


Figure 6.1: Mean protein spectrum healthy persons in comparison with the mean protein spectrum of patients suffering from benign bowel disorders.

If biomarkers specific for benign bowel diseases could be found, it would be possible to develop a cheap and non-invasive diagnose method. The discrimination can be prolonged even further. In some cases, it is hard for the pathologist to determine what kind of disorder the patient is confronted with. Biomarkers for specific bowel diseases can aid a great deal in the diagnosis of the pathologist. Furthermore, some types of bowel disease can develop a malignant form. If the disease is treated *before* this form is developed, the survival rate will increase even more than with early cancer detection. Although the research in this topic is still ongoing at the moment, the analysis with SPECTRA shows promising results, presented in table 6.2.

Table 6.2: Results of benign bowel diseases versus healthy subjects

| | **SPECTRA** |
|---|---|
| **Sensitivity** | 100% |
| **Specificity** | 97.9% |
| **Total recognition rate** | 97.0% |

## 6.3 Robustness, benchmarking and performance

Under these conditions, SPECTRA seems to function quite well. However, what happens when extreme situations are simulated? Though SPECTRA provides an extensive error handling, we do not have any hard evidence. We created several conditions to see how SPECTRA would perform.

The first condition was a systematically wrong input. Other types, negative values when not allowed and exceptionally high values were entered in all parameter fields. SPECTRA handled all these situations with an error notification. Furthermore, the cancelation option in all dialogs was tested. All running routines were neatly interrupted by SPECTRA. No situations could be created in which SPECTRA would crash.

The next condition was extensive datasets. These were expected to present some problems, since large datasets make a heavy demand on a system. The time of loading and the behavior of the system was compared with the current software tool. The results are presented in table 6.3. A set of 200 control samples and 316 benign bowel diseases samples was used. The total size was 738 Mb.

Table 6.3: Results of benchmark tests on ClinProTools and SPECTRA.

| | **SPECTRA in MSM** | **SPECTRA in normal mode** | **ClinProTools** |
|---|---|---|---|
| **Time to load data** | 5 min | - | 1 hour |
| **Time to execute analysis** | 40 min | - | - |
| **Crash** | - | 50% on data loading: 20 min | 3 hours after executing analysis |

## 6.4 Qualitative requirement evaluation

The last evaluation method we use to assess SPECTRA is a qualitative overview. Let us go over the requirements again to see whether they are fulfilled or not.

*Modular approach for easy modification and adaptation to other datasets.*
This requirement has eventually formed the backbone of SPECTRA and is therefore certainly fulfilled. The simple interfaces between the modules, the DATA and HISTORY cell arrays, make sure that external procedures are easily integrated.

*Easy-to-learn, vivid and intelligent interface.*
This presented the only bottleneck in the development of the system. It turned out that the user was not satisfied with the interface SPECTRA offered. Since this was one of the main conclusions from the prototype analysis, extra attention

was given to this requirement afterwards.

*Must run on a Windows OS machine.*
Since SPECTRA is entirely developed in Matlab, this requirement did not present much problems. The creation of a stand-alone application for Windows was quickly achieved using Matlab's 'Matlab to C\C++' Compiler. A side effect of this solution is that the Matlab and MCC must be present on the system where SPECTRA is deployed. For the LUMC this is no problem, since the group possesses all necessary licenses.

*Compatibility with Microsoft Excel and Access for class assignment.*
This requirement is partly fulfilled. Extensive import and export compatibilities with Microsoft Excel are possible, but not with Microsoft Access. Since queries and tables are easily exported in Excel (in fact, that is the current standard procedure), this was not considered a critical problem.

*Data must contain: Unique mass spectrometry number, unique sample number, class-label.*
The structure of the DATA cell speaks for itself. This requirement is fulfilled.

*Data export possible in Matlab- and Microsoft Excel-format.*
The combination of MS number and sample number, together with class assignment can be exported in Excel, using the function incorporated in the Menu functionality. The intensities and m/z values can be exported in Matlab using the appropriate function in the Menu functionality. This requirement is considered fulfilled.

*Preprocessing functions available must be: binning, smoothing, normalization, baseline correction.*
All these preprocess algorithms are implemented in SPECTRA. They can be executed in the Preprocess screen, using the homonymous functionalities.

*Transparency of used algorithms through help function, user parameter definition and well-documented algorithm information.*
All parameters as they are defined in the Theory chapter can be altered at the user's demands. The same chapter also provides an extensive explanation, including references, about the implemented algorithms. The help function is dynamically developed, to ensure full transparency, also of new procedures. A user manual is written to provide full insight in SPECTRA's possibilities.

*Quick data analysis possible with default parameter values.*
This requirement is fulfilled. When a screen is opened, the default values are initially presented on the screen. By pressing the 'Default' button, all parameter values are restored to these default values.

*Spot selection possibilities: all spots and spot average.*
SPECTRA is the first application that can handle one sample distributed over different spots. It is possible to compute the mean spectrum over different spots, or use all spots as separate samples. This requirement makes it unique in its kind.

*PCA and LDA must be integrated.*
These algorithms are the core of the Classification module. The implementation did not present any real problem, since the algorithms were described in detail in the Theory chapter.

*Leave-one-out validation and external validation must be implemented.*
The validation methods are coded independently of the statistical department. Again, the classification module provides these two types of validation.

*All data information must be visible.*
The data list contains all information of the data; ms-number, sample-number, class, spot information, dimensionality and the last column encompasses all information on preprocess procedures used on the data.

*Classification results must be visible.*
The two lists, one with the incorrectly classified and one with the correctly classified samples present these result. Extra attention to the presentation of these lists, as well as the results in text was given after the prototype analysis.

*Visualization options must be: heat map, averages, single spectrum, correlation weight plots, multiple plots, mirror plots.*
The visualization module contains these options. This requirement is fulfilled.

*A zoom option must be included.*
The zoom option is encompassed by the visualization module. This requirement is fulfilled.

# 7

# Conclusions

In this chapter, we discuss the entire project. The results of the tests are interpreted and the targets, as they are stated in chapter 1.5, are assessed. Furthermore, we discuss some recommendations and confer about some future aspects of SPECTRA.

## 7.1 Assessing the targets

SPECTRA is a stand-alone application that it is created out of nothing. It was therefore very important to accurately gather information on *what* the problem was. Furthermore, to prevent the time-consuming and redundant activity of exploring and designing factors of SPECTRA that are already implemented by someone else, it was important to get a good overview of available comparable tools. The first target:

> ***Target 1:*** *Analyze the current research situation and comparable tools.*

has been achieved in chapter 2. The current research situation at the LUMC is well explored, since it proved an excellent basis for SPECTRA's modularity. Furthermore, nearly all advantages of ClinProTools, like the heat map, are implemented in SPECTRA and special consideration is given regarding ClinProTools disadvantages, like its system instability.

Next to the context of SPECTRA, the attention given to what the user actually wanted was well spend. Learning from the examples of interdisciplinary research cases where things went wrong because of bad communication, frequent discussions and interviews with the user has been taken place. Therefore, target 2:

> ***Target 2:*** *Analyze and record the functionalities of SPECTRA.*

has been achieved. Chapter 2 gives an overview of a list of requirements approved by the user.

For the implementation of the statistical algorithms, we first needed to know what actually took place. All algorithms are mathematically described in chapter 3. These formulas were then implemented in chapter 5. Target 3:

***Target 3:*** *Create a theoretical overview of the required algorithms.*

is therefore achieved. Besides the required algorithms, an extensive literature survey has been done by the user, as is described in the literature research assignment "Dimensionality Reduction Methods for Mass Spectrometry Data Used in Oncology".

Prior to the implementation of SPECTRA, it had to be designed on a abstract level, so all requirements would be actually fulfilled in the end. Since SPECTRA has to be used also by non-experienced users, the user interface was a very important point of research. Target 4:

***Target 4:*** *Design the system architecture and user interface.*

is achieved in chapter 4. SPECTRA's modularity and unique Memory Saving Mode are designed in this chapter. Furthermore, a more detailed design of the functionality of SPECTRA is given here, to facilitate the implementation.

The influence of the user on the design of SPECTRA was large, thanks to the iterative approach of the development of SPECTRA. Target 5:

***Target 5:*** *Implement a working prototype.*

contributed considerably to this incremental design. By discussing various changes with the user and evaluating the prototype (section 4.9) gave the user insight in the development of SPECTRA, while providing enough opportunities to make alterations.

The quality of the final version of SPECTRA was thoroughly assessed through benchmarks tests, user evaluation and testing SPECTRA on a new dataset. SPECTRA proved able to create a model that discriminates patients with a benign bowel disease from healthy persons based on protein information in their serum. Target 6:

***Target 6:*** *Test and evaluate the final application.*

is achieved and described in chapter 6.

Additionally to these targets, a User Manual has been created to help a new user with the use of SPECTRA.

## 7.2 Discussion

The meta-requirement, namely whether the user actually considers the application a success is fulfilled. This is very important for further use of SPECTRA. It

would end up on the shelf with a load of dust on it, no matter *how* well the other requirements were met, if the user did not like it. Therefore, the fulfillment of this requirement is considered the most determining proof of SPECTRA's success.

The resemblance of SPECTRA's outcome to the one presented in the paper (see table 6.1) points to the fact that the algorithms used by the statistician are correctly implemented. The correctness we speak of here, is parity of SPECTRA's algorithms and the algorithms used by the statistician. This does not cover in any aspect the correctness of these algorithms regarding the data! Of course, these algorithms *are* correct, but since proteomic research is very new, they do not necessary have to be the *best*. For example, the literature describes an extensive discussion on the best way to reduce the dimensionality of protein patterns, and keeping the most important ones for the classification. For further details regarding this subject, please consult my literature research: "Dimensionality Reduction Methods for Mass Spectrometry Data Used in Oncology".

During the robustness tests, one of the first things that surfaced, was that the efforts on implementing cancelation options and error handling were certainly not wasted. Although they were no primary requirement, the addition appeared very valuable for the user's comfort when using SPECTRA. Since it could take over an hour to load a dataset, making an error would cost a lot of valuable time. The cancelation option prevented this. Furthermore, the clear error dialogs contributed to the user's understanding of the system and the use of special error codes made consulting the help function a lot easier. Since SPECTRA never crashed during the tests, error handling is considered extensive enough under these conditions.

The system *did* crash, when coping with large datasets. The only time in fact it did not crash, was when in MSM. As expected, the memory load was less when using this mode. Still, since the research field is expanding, the datasets are increasing in size evermore. Even 'worse', the technology is developing at light speed, producing larger and larger resolution protein patterns. It is therefore expected that memory issues will become a great problem in the future.

## 7.3 Recommendations and future of SPECTRA

The search for decisive protein patterns still lingers on. SPECTRA could contribute greatly to the speed at which this happens at the LUMC. However, it must be adapted to the newest findings. Eventually, the ideal situation would be to integrate SPECTRA in the clinic. The physician could use it to analyze protein patterns of patients, and quickly make a diagnosis. Figure 7.1 represent this state.

Before this state is reached, a lot of research has to be done. To enhance this research, several recommendation are stated here.

The first is the validation method. SPECTRA incorporates single cross-validation. It is recommended to extend this validation method to double cross validation, as is described in [7]. This would considerably improve the reliability of SPECTRA. Furthermore, SPECTRA should be enhanced, implementing new

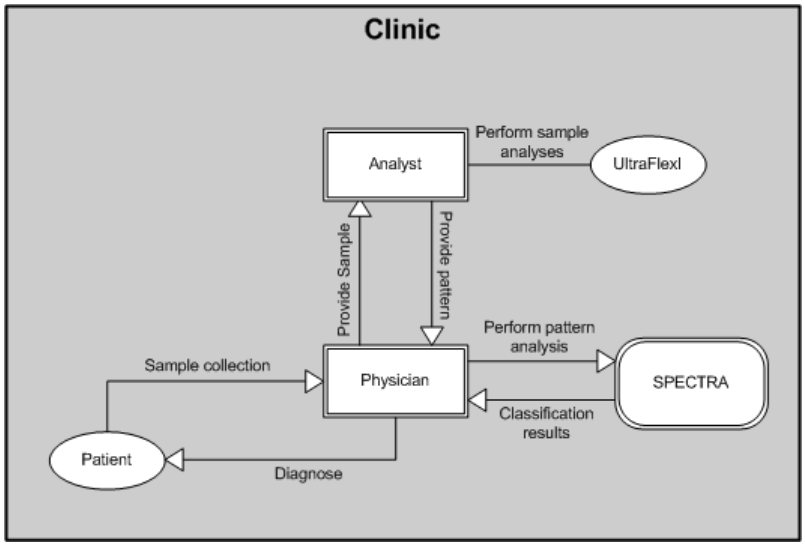


Figure 7.1: Future use of SPECTRA.

algorithms. Especially the Support Vector Machine proves to be very promising in creating reliable models for protein pattern classification. In addition, a multiple class classification algorithm should be implemented. This way, a more detailed diagnosis can be devised in the future.

An important recommendation is the integration of the three physical databases. Erroneous queries have resulted in a major setback in the DIPSTICC research. The export function of SPECTRA has solved this problem somewhat. When this integration would take place, another aspect concerning SPECTRA arises. A flat file storage has been chosen, but database storage will become more and more important in the future. The advantage of this is that multiple instances can use the data at the same time. Concurrent accesses will also make it possible for multiple groups to present their data to SPECTRA. This would be of enormous influence on the proteomic research. The datasets would grow far more quickly than in the current situation, thereby enhancing reliability of the resulting models. A webinterface of SPECTRA would considerably contribute to this. Thanks to SPECTRA's modular approach, the alterations that have to be made, will not be extensive. This way, even new procedures can be added, providing they have the same function interface. SPECTRA would function as a database portal for the rest of the world. With these changes, the computation capacity will be the bottleneck. It is therefore recommended that the system that uses SPECTRA will have a large memory capacity to load all protein patterns.

Although these visions are not yet (practically) possible to realize, proteomic research is already developing as we read. The future will most certainly prove that proteomic research boosts the quality of clinical diagnostic aspects. SPECTRA's role will become more clear in this prospect.

# A

# *Paper*

The following paper is partly constructed using SPECTRA's visualization options. Furthermore, it uses algorithms as they are implemented in SPECTRA. It is recently accepted for publication in the European Journal of Cancer.

## DETECTION OF COLORECTAL CANCER USING MALDI-TOF SERUM PROTEIN PROFILING

M.E. de Noo[1], B.J. Mertens[2], A.Ozalp[3], M.R. Bladergroen[3], M.P.J. van der Werff[1,3], C.J.H. van de Velde[1], A.M. Deelder[3], R.A.E.M. Tollenaar[1]

Corresponding author: M.E. de Noo, M.D. Department of Surgery, K6-R, Leiden University Medical Center P.O. Box 9600, 2300 RC Leiden The Netherlands Tel: +31-71-5261278 e-mail: M.E.de_Noo@lumc.nl

[1]Department of Surgery,
[2]Department of Medical Statistics and Bioinformatics,
[3]Biomolecular Mass Spectrometry Unit, Department of Parasitology,
Leiden University Medical Center, Leiden, The Netherlands

## A.1 Abbreviations

| | |
|---|---|
| CRC | Colorectal cancer |
| AUC | Area under the curve |
| MALDI | Matrix Assited Laser Desorption Ionisation |
| TOF | Time Of Flight |
| MS | Mass spectrometry |
| SELDI | Surface Enhanced Laser Desorption Ionisation |

## A.2 Introduction

Colorectal cancer (CRC) is among the most common malignancies and remains a leading cause of cancer-related morbidity and mortality. It is well recognized that CRC arises from a multistep sequence of genetic alterations that result in the transformation of normal mucosa to a precursor adenoma and ultimately to carcinoma. Given the natural history of CRC, early diagnosis appears to be the most appropriate tool to reduce disease-related mortality [31], [32]. Currently, there is no early diagnostic test with high sensitivity, specificity and positive predictive value, which can be used as a routine screening tool. Therefore, there is a need for new biomarkers for colorectal cancer that can improve early diagnosis, monitoring of disease progression and therapeutic response and detect disease recurrence. Furthermore, these markers may give indications for targets for novel therapeutic strategies.

Proteomic expression profiles generated with mass spectrometry have been suggested as potential tools for the early diagnosis of cancer and other diseases. Different protein profiles may be associated with varying responses to therapeutics. It has been postulated that on the basis of the presence/absence of multiple low-molecular-weight serum proteins using time-of-flight (TOF) mass spectrometry technologies, such as SELDI-TOF and MALDI-TOF, biomarkers can be identified [33]-[36]. Although the data from these studies are encouraging, critical notes have been made on both study design and experimental procedures for proteomic profiling[37]-[39]. In addition, the importance of avoiding confounding biological variables, as well as technological factors that may bias the results, have previously been stressed by several authors[40], [41]. Another recurrent topic for debate is the use of independent validation sets for the classification of diseased versus healthy individuals. A specific problem in the discovery-based research field of clinical proteomics is overfitting. Overfitting may occur in the analysis of large datasets when multivariate models show apparent discrimination that is actually caused by data over-interpretation, and hence give rise to results that are not reproducible [39], [42], [4]. The chance of overfitting, however, can be reduced by appropriate application of validatory estimation and assessment, such as through application of double cross-validation, when properly implemented.

The objective of this study was to assess the feasibility of mass spectrometry based protein profiling for the discrimination of colorectal cancer patients from healthy individuals. In addition to standardizing technical factors and biolog-

ical variations, we performed blinded tests and employed a randomized block design experimentation to minimize impact of potential confounding factors and to avoid bias. To minimize danger of overfitting, among other reasons, we used a fairly inflexible classification method based on first-and-second order statistics only. Specifically, Fisher linear discriminant analysis was employed with double cross-validatory integrated estimation and validation of error rate on the entire dataset to calculate an unbiased error rate assessment, which was the prime research objective of this study. Subsequent to this fairly critical evaluation of the data, which is geared towards exclusion of bias in the assessment, we then explored a post hoc evaluation of the proteomic profiles to explain the observed classification (error rate) results.

## A.3 Material and methods

***Subjects***. Serum samples were obtained from a total of 66 colorectal cancer patients one day before surgery. All patients with stage IV disease had synchronous metastatic disease confined to the liver. Colorectal cancer was histologically confirmed on surgical specimens and preoperatively assessed with abdominal CT scan and carcinoembryonic antigen (CEA) levels were determined. The extent of tumor spread was assessed by TNM classification based on histological examination of the resected specimen. All stages of colorectal cancer were represented in the patient group. The median age of the patient group was 62.8 years (range 32.6-90.3) and the male to female ratio was 41/35. Patients were included from October 2002 till December 2004 in our Center. The control group consisted of 50 healthy volunteers. The median age of the healthy symptom-free control group was 49.7 years (range 25.9-76.6) and the male to female ratio was 21/29. The controls were included in November and December 2004 (Table A.1).

***Study design***. Having identified plate-to-plate and day-today variation as important potential batch effects, we used a randomized blocked design [43], [44]. All the available 116 samples from both groups (controls and colorectal cancer) were randomly distributed across 3 plates in roughly equal proportions (Table A.2). For colon cancer, the distribution of stadia across plates was again in random fashion and in approximately equal proportions (Table A.3). The position on the plates of samples allocated to each plate was randomized as well. Each plate was then assigned to a distinct day, which completes the design. Analysis was carried out on 3 consecutive days, Tuesday to Thursday, processing a single plate each day. A duplicate of this randomized blocked study was performed in the following week.

***Serum samples***. Informed consent was obtained from all patients and the Medical Ethical Committee approved the study. All blood samples were drawn while the patients or healthy controls were seated and non-fasting. The samples were collected in a 10 cc Serum Separator Vacutainer Tube and centrifuged 30 min later at 3000rpm for 10 minutes. The serum samples were distributed into 1 ml aliquots and stored at -70C. After thawing on ice the 126 serum samples were randomised over 3 different 96-well microtititration plates (Greiner) and

then stored at -70C until the experiment.

***Isolation of peptides***. The isolation of peptides from serum was performed using the magnetic beads based hydrophobic interaction chromatography (MB-HIC) kit from Bruker, mainly according to manufacturers instructions, adapted for automation on a 8-channel Hamilton STAR®pipeting robot (Hamilton, Martinsried, Germany). Magnetic beads with C8- functionality (MB-HIC8) were divided in 5-$\mu$l aliquots in a 96-well microtiter plate, which was placed on the magnetic beads separation device (MPC®-auto96, Dynal, Oslo, Norway), with the magnet down. Ten-$\mu$l MB-HIC binding solution and 5-$\mu$l serum sample were added to the beads and carefully mixed using the mixing feature of the robot. The sample was incubated for 30 sec and the magnet was lifted, followed by a 30 sec waiting interval to settle the magnetic beads. The supernatant was removed and the magnet was lowered again. The magnetic beads were washed three times with MB-HIC washing solution (also provided with the kit) lifting and lowering the magnet as needed. The peptides were eluted from the beads using 10-$\mu$l 50% acetonitril and 2-$\mu$l of this eluate was transferred to a fresh 348-well microtiter plate (Greiner). Most of the remaining eluate (6-$\mu$l) was transferred to an auto sampler vial containing 54-$\mu$l water and stored for later use. 15-$\mu$l $\alpha$-cyano-4-hydroxycinnamic acid (0.3 g/l in ethanol: acetone 2:1) was added to the 1-$\mu$l eluate in the 348-well microtiter plate and mixed carefully. 1-$\mu$l of this mixture was spotted in quadruplicate on a MALDI AnchorChip$^{TM}$(Bruker Daltonics, Bremen, Germany).

***Reagents***. $\alpha$-cyano-4-hydroxycinnamic acid as well as the MB-HIC8 isolation kit was obtained from Bruker Daltonics (Leipzig, Germany). Acetonitril was obtained from Biosolve (Valkenswaard, The Netherlands), ethanol was purchased from Mallinckrodt Baker (Deventer, The Netherlands) and acetone was obtained from Nedalco (Bergen op Zoom, The Netherlands).

***Electronically regulated matrix container***. To prevent evaporation of the highly volatile matrix solution a specially designed matrix container was developed in house. This container consists of a PVC holder with a PEEK inlay to make the container inert and a Teflon$^{TM}$sliding cover with eight small holes (for the eight-channel pipeting robot). The opening and closing of the sliding cover is regulated using one of the three channels from the magnetic beads separation device. The programming of this device is performed with the same software that is used to program the Hamilton STAR®.

***Protein profiling***. Matrix Assisted Laser Desorption Ionisation Time-Of-Flight (MALDI-TOF) mass spectrometry measurements were performed using an UltraflexI TOF/TOF instrument (Bruker Daltonics, Bremen, Germany) equipped with a SCOUT ion source, operating in linear mode. Ions formed with a N2 pulse laser beam (337 nm) were accelerated to 25kV. With this specific serum preparation peptide/protein peaks in the m/z range of 960 to 11169 Dalton were measured. An independent mass spectrometer operator performed the experiments at 3 consecutive days after cleaning of the instrument. One week later the experiment was duplicated in exactly same order. Hereafter the entire process of capturing and concentrating serum proteins using C8 magnetic beads including the generation of readouts of the MALDI-TOF spectra will be designated as the protein profiling procedure.

***Data processing***. All unprocessed spectra were exported from the Ultraflex in standard 8-bit binary ASCII format. They consisted of approximately 45000 mass-to-charge ratio (m/z) values, covering a domain of 1160-11600 Dalton. To

increase robustness, the average of four spots was used to represent one serum sample. Subsequently, we lightly smoothed the spectra using the Whittaker [13] smoother. Due to the quadratic nature of the TOF-equation, the high-resolution spectra were binned using a linear scaling at the time scale, resulting in bin width of approximately 1 Dalton at the beginning of the spectrum and 3 Dalton at the end at the mass/charge scale. The resulting spectra generally showed strong baseline effects. These were removed using an asymmetric least squares algorithm. To normalize the spectra, we calculated the median intensity of every spectrum and subtracted it from the original spectrum. Each of the thus normalized spectra was then also divided by the interquartile range of intensity within that spectrum. We consider this more robust than normalization of the spectra on the average, as it is less sensitive to the most extreme intensities. Finally, prior to classification and evaluation of error rate, the logarithm was taken of all intensity measurements (predominantly to ensure numerical stability of computations).

***Statistical data-analysis***. Fully validated classification error rates were estimated based on a classical Fisher linear discriminant analysis through complete double cross-validatory joint estimation and assessment of class predictions.

Fisher linear discriminant analysis may be defined as assigning an observation to the group for which the smallest within-group distance:$D_g(\mathbf{x}) = (\mathbf{x} - \mu_\mathbf{g})\sum^{-\mathbf{1}}(\mathbf{x} - \mu_\mathbf{g})^T$ is found for the corresponding observed feature vector $\mathbf{x} = (x_1, \ldots, x_p)$ with respect to the $g^{th}$ group (g=1,2 here, for either cases or controls), where $p$ is the dimensionality of the problem, $\mu_\mathbf{g}$ denotes the population within-group sample mean for the $g^{th}$ group and $\sum$ is the (common) within-group dispersion matrix. We may estimate the population means through the within-group sample means. When the dimensionality of the problem is greater than the sample size, as is the case in this problem, the observed within-group pooled covariance matrix $\mathbf{S}$ will typically not be of full rank and hence special measures are called for before we can apply the above paradigm in this context. This can be achieved through an initial principal components decomposition of the observed within-group pooled covariance matrix $\mathbf{S} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$, where $\mathbf{Q}$ and $\mathbf{\Lambda} = diag(\lambda_1, \ldots, \lambda_r)$ are the matrices of principal component weightings and variances respectively ($r$ is the rank of the pooled covariance matrix). We then re-estimate the within-group covariance matrix by only retaining the first $k$ components only: $\mathbf{S_{(k)}} = \mathbf{Q_{(k)}}\mathbf{\Lambda_{(k)}}\mathbf{Q_{(k)}}^T$ , which account for most of the variation in the spectra. The discriminant rule may now be expressed as assigning an observation to the group for which we observe the smallest sample estimate $\bar{D}_g(\mathbf{x}) = (\mathbf{x} - \bar{\mathbf{x}}_\mathbf{g})\mathbf{S}_\mathbf{k}^{-\mathbf{1}}(\mathbf{x} - \bar{\mathbf{x}}_\mathbf{g})^T$.

In the two-group case, this is also equivalent to least-squares regression analysis using the Moore-Penrose inverse of the pooled covariance matrix when $k = r$ (all components kept, also known as shortest least squares regression), or else is equivalent to so-called shrunken least-squares regression [45], [46] for more details). When choosing $k < r$, the choice may be made through appeal to a (cross-) validatory evaluation of the performance of the respective possible choices for the parameter $k$. The above methodology has been described and compared to other methods in the recent paper by Mertens [27], which shows this method to be competitive in the closely related high-dimensional setting for classification with microarrays. Much similar and confirmatory experience has accumulated in related fields of application, which identifies this classifi-

cation method as reliable and stable in high-dimensional analysis, as has been described by Stone and Jonathan, among others [47], [48]. Instead of ordinary leave-one-out cross-validatory choice of $k$, we employ double cross-validation. This is an extension of leave-one-out cross-validation which combines validatory "choice of model" (the parameter $k$ in this case) with "predictive assessment" (of the same model, through use of error rate or other suitable summary statistic). The reason for this additional "technical complication" is that we do not wish to incur the bias inherent in the assessment, which would normally result from a model choice based on ordinary leave-one-out validation only (see the seminal paper by Mervin Stone [49] for full explanation of all the issues involved ). In a double cross-validatory evaluation, we remove each individual in turn from the data (just as in ordinary leave-one-out cross-validation), after which the discriminant rule is fully recalibrated and optimized for prediction on the leftover data (now of size $n-1$, where $n$ is the total initial sample size) and using the same procedure in each case. The choice of the calibration rule (i.e. choice of $k$ in this case) to classify the left-out observation is then again based on a leave-one-out cross-validatory estimation (hence the name "double-cross") within the leftover set of size $n-1$. The resulting classification rule is then applied to the left-out datum to obtain an unbiased allocation for this sample. This procedure is then repeated across all individuals and for each person separately, after which we can calculate a truly unbiased estimate of the misclassification rates on the basis of the thus validated (and calibrated) classifications. In other words, "double-cross" is actually "leave-one-out cross-validation within leave-one-out cross-validation" and it is precisely because of this that we can avoid bias in error rate estimation that an ordinary application of standard leave-one-out choice would imply.

## A.4 Results

In the first week three different randomized target plates were successfully measured on three consecutive days in the middle of the week. A duplicate experiment was performed in the second week on the same days. Figure 1 shows a raw data spectrum, directly obtained from the MALDI-TOF mass spectrometer. Before pre-processing and further analysis a mean spectrum of each sample was calculated over all four spots that were measured for each sample. In case all four spots from one sample showed spectra of poor quality due to a technical problem, the sample was left out of the analysis. This was the case for 3 CRC patients' samples. The above-described pre-processing steps resulted in a sequence of 4483 normalized m/z values ranging from 1160 to 11.600 Dalton, for each individual.

***Detection of colorectal cancer***. Double cross-validatory analysis and evaluation carried out on the protein spectra measured in week 1, correctly classified 45 of the 50 controls as not cancer. Sixty of the 63 cancer samples were correctly classified as malignant, including 9 of 10 TNM stage I patients (Table A.4). The remaining 2 misclassified patients had stage II disease. All patients with stage III and IV disease were correctly recognized as malignant within the double cross-validatory evaluation. These validated results thus yield

a total recognition rate of 92.6%, a sensitivity of 95.2% and a specificity of 90.0% for the detection of CRC (table 5). To analyze the actual discriminative power of the classifier, we produced an ROC-curve (again based on the double cross-validatory classification probabilities), visualizing the performance of the two-class classifier in figure A.2. The (double c.v.) AUC of the classifier was 97.6.

We repeated the entire double cross-validatory evaluation executed with the week 1 data using the duplicate measured spectra from week 2. This entire double cross-validatory procedure was identical to that carried out in week 1 and used the same calibration spectra. However, prior to classifying each left-out datum in the outer "shell" of the double cross-validatory procedure, we substituted the week 1 data with the corresponding measured spectra from the same sample in week 2. In this manner, we could calculate a double cross-validatory error rate, which takes the effect of replicate measurement of the spectrum (and thus also recalibration of the equipment) into account. The effect of classifying the remaining replicate data was that the recognition rate dropped to 88.8%. The sensitivity and specificity for the detection of CRC for the second week data was 80.6% and 97.1% respectively (Table A.4). The associated AUC of this repeat double cross-validatory estimation on week 2 was 96.8%.

It is of interest to evaluate bias of the double cross-validatory calculations. Hence, we performed a permutation exercise, which randomly permutes and reassigns the class labels across subjects and then repeats the entire double cross-validation procedure. Carrying out this procedure more than 600 times resulted in a median recognition rate of 50.0% (95% confidence interval is [36.3, 72.7]). The median AUC was 49.4% with confidence interval of [24.8,64.2]. As both median recognition rates and AUC's equal 50%, there is thus no substantial evidence of bias remaining within the cross-validatory calculation.

Having executed the above-described validatory evaluation, we can explore the nature of the classification through a post hoc analysis. We found that the first two principal components provide most of the between-group separation. Figure A.3 shows a plot of the correlation coefficients, with the class indicator, which can be calculated from the linear discriminant weightings [45], [46] in the region between 1160 and 11.600 Dalton. The remainder of the plot is not shown, as the coefficients are effectively zero in that range. As can be seen, the classification is achieved primarily through a contrast in peak intensities between the first and second principal component. This can also be seen from the scatter plot shown in figure 4: low intensities at the first peak for cases separates cases from controls. Likewise, a small contribution for controls at the second peak separates controls from patients. To illustrate these results further, we can simply calculate the contrast between the two peak intensities directly across all subjects and construct a simple one-dimensional summary of the data, as shown in the histogram displayed in figure A.5, which shows overlapping histograms of this (ad hoc) contrast for each group separately. The separation is clearly visible. We may also quantify the significance of this difference by performing a two-sample Student t-test on this contrast, which is $t = 14.0$ ($p < 0.0001$).

## A.5 Discussion

Our study supports the hypothesis that serum protein profiles can discriminate a normal from a malignant state of organs, in our case of the colon. Here we show that, based upon information in MALDI-TOF serum spectra, a classifier could be constructed for the detection of CRC. This classifier, calibrated and validated on spectra of one week (1) demonstrated a sensitivity and specificity of 95.2 and 90.0% respectively. Thirty-four patients out of thirty-seven with early stage disease (stage 1 and 2) and all patients with stage 3 or 4 disease were correctly classified as having cancer. For the misclassified control subjects it was not possible to retrieve the current physical state as it concerned anonymous healthy controls.

Sensitivity and specificity of 80.6 and 97.1% respectively was achieved when the entire double cross-validatory evaluation was repeated for the data of week 2. The latter evaluation, through use of replicate measurements within the double cross-validation, is likely to provide the more realistic assessment of true error rates and appears to better represent possible diagnostic potential as will be discussed further in this paper.

Although previous studies have reported similar high classification results for various solid tumors, we prefer evaluation though a thorough study design and double cross-validation of classification as proposed in this study3-6,12,23,24. As a great variety of different discriminating peaks for the same malignancy have been described [33], [34], [50] caution with proteomic data has been stressed before [37], [38]. The discrepancies in discriminating protein profiles, found by different research groups, lead to serious concerns regarding to biological variations and technological reproducibility issues. Therefore, we used a standardized and well-documented sample collection and a thorough study design, matching biological variables and pre-analytical conditions [6]. Still, patient samples from all stages of CRC were equally distributed over the different target plates, as was the male/female ratio between the two groups, excluding these factors as a discriminator in the detection classifier. Unfortunately there was significant difference in age; the control group being younger than the CRC patients. Ideally, the control group should consist of age-matched symptom-free individuals undergoing a colonoscopy showing no aberrations. However, due to the nature of the intervention, ethical legislation and the increasing disease burden with ageing this is difficult to realize in clinical practice.

A source of bias may be the presence of batch effects, such as day-to-day variation or plate-to-plate variation. The presence of batch effects is unavoidable and - rather than to eliminate them from the design - a better approach is to account for and accommodate these effects, in such a way that they do not lead to errors of artificially induced group separation. In addition, this makes experimentation more realistic as well. Randomized block design is the appropriate design choice in these circumstances. Consequently, we randomly distributed the available samples from each group across the batches such that proportions were equal across batches within group. The so-called randomized block design ensured that the batch effect - if it materialized - would not induce an artificial between-group effect [43], [44].

A crucial point of discussion in the evolving field of clinical proteomics is validation of classification [39], [51]. Given the sample size achievable within the experiment, use of a separate (possibly set-aside) validation set was pre-

cluded. The other problem is "predictive optimization". However, as evaluation of predictive performance of the classifier is our primary focus, it is crucial that calibration is not carried out on the same data used for validation, which in turn would require an additional tuning set. Again, this would greatly increase the burden of collecting sufficient samples. For these reasons, other studies often carry out predictive optimization on the full data in practice- which results in optimistically biased error rate evaluations, particularly with high-dimensional data such as in mass spectrometry proteomics [52]. As we have already suggested, another option is to reduce the available calibration data prior to optimization, so as to set aside data, both for a training and validation set. However, this "solution" is not as innocent as would appear at first sight, since it typically reduces the calibration set beyond the point of what is needed for reasonable calibration. Once more, this is particularly the case in high-dimensional cases such as clinical proteomics, where samples of malignancies are relatively difficult to obtain. Both problems may be avoided by carrying out a so-called double-cross-validatory approach, which avoids the need for separate test and validation sets to yield unbiased error rate estimates. The double validatory aspect of the procedure results from the fact that the discriminant rule constructed to classify the left-out data was optimized through a secondary cross-validatory evaluation within the first cross-validatory layer (i.e. full cross-validation again on each "leftover" set after removal of an observation). In this manner, we are able to integrate predictive optimization and predictive unbiased validation in the same procedure, without loss of data - which is a crucial requirement to get realistic estimates of error rate with high-dimensional data while reducing the risk of overfitting (first proposed by Stone [49]). Although the principle is sound and understood, this procedure has until recently not been applied in practice due to the considerable computational cost and (algebraic) complexity of the method.

Our classifier is based on Fisher linear discrimination, which is one of the oldest statistical allocation methods and certainly the most widely used and successful approach to statistical classification and pattern recognition this day. It has been derived and may be justified based on a variety of principles of inference, such as maximization of the between-group separation relative to within-group error in the two-group case or the likelihood principle for normally distributed within-group populations, among others. The methodology has been amply studied and has been established as reliable and robust form of classification and discriminant analysis. Furthermore, Fisher discrimination does not require an assumption of within-group normal dispersion [46], [53], [54]. Hastie et al. contains an up-to-date account of many new applications that demonstrate the continuing success of the approach [29], [27]. Much similar and confirmatory experience has accumulated in related fields of application [47], [48], which identifies this classification method as most reliable in high-dimensional analysis. For proteomic mass spectra, principal components are attractive as it provides a means of non-parametrically smoothing and pooling information across peaks.

The controversy about the use of protein profiles as a pattern diagnostic without analysis of the diagnostic biomarkers remains to be solved for its clinical application. Identification and functional analysis of these discriminating proteins/peptides might render new insights on tumor development and environmental responsiveness, which could eventually be translated in new diagnostic and prognostic insights for the clinician. Unfortunately, little success has been

booked so far in assigning reproducible discriminating biomarkers [42], [51]. Though this study showed two most discriminating mass values of MALDI-TOF based protein profiling analysis to be low molecular weight fragments, we have not identified these potential biomarkers yet.

In the present study we used patterns of proteomic signatures from high dimensional mass spectrometry data to generate a diagnostic classifier for the detection of CRC. To our knowledge, this is the first double crossvalidatory study in a randomized block design in this field of research. Although independent validation would strengthen the observations and follow up studies are now underway, we obtained maximal reliability in classification in this study while maintaining protection against overfitting. Due to the relatively small sample size we have chosen to use our entire dataset for a within-study validation to avoid optimistic biased (error) misclassification rates. To assess the performance of our classifier a further independent study will be necessary. Nevertheless, we are currently able to detect CRC accurately on the basis of differences in actual information in the serum protein profiles with a rigorously standardized approach and exclusion of batch effects. Thus, although introduction in a routine clinical setting may take longer than originally hoped for, this study is an initial proof for a successful evolution of the potentially great use of discriminating protein profiles in the detection of CRC.

## A.6 Figures

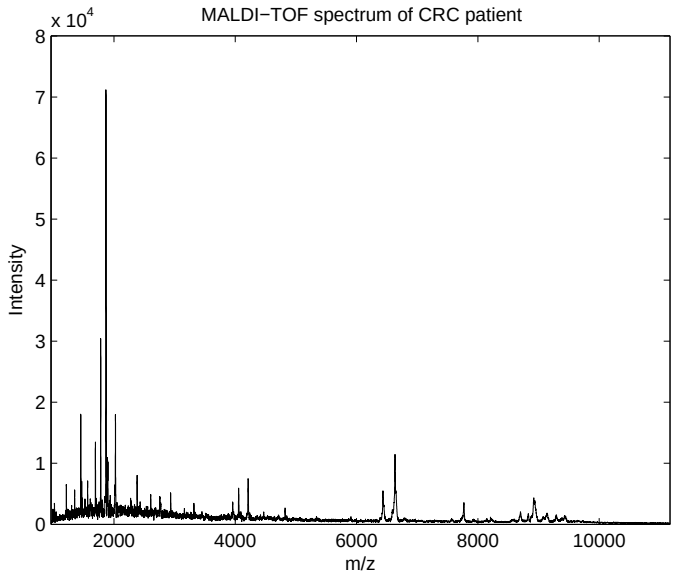


Figure A.1: MALDI-TOF spectrum of a colorectal cancer patient after peptide isolation with C8 magnetic beads. On the Y-axis the relative intensity is shown. The mass to charge ration (m/z) is demonstrated on the X-axis in Dalton.

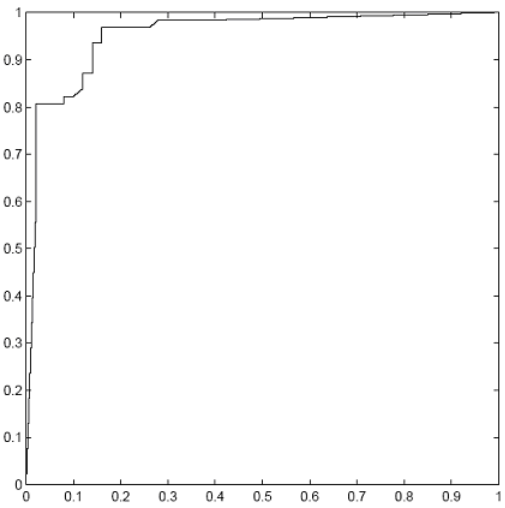


Figure A.2: ROC-curve for the double cross-validated two-group classifier. The true positive recognition rate (sensitivity) is demonstrated on the y-axis against the false negative recognition rate (1-specificity) on the x-axis of the classifier.

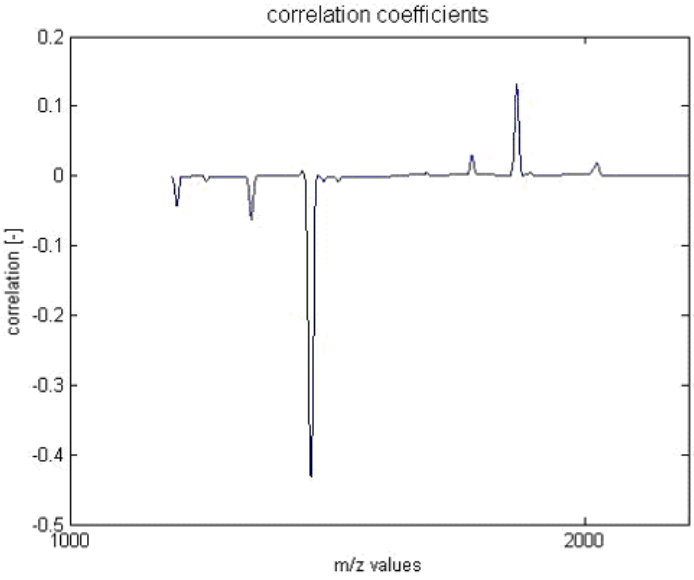


Figure A.3: Correlation coefficients of two first principal components with the class indicator. The correlation coefficients were calculated from the linear discriminant weightings. The negative correlation of the first peak is an indicator for the control group and the positive correlation of the second peak points out the cases.

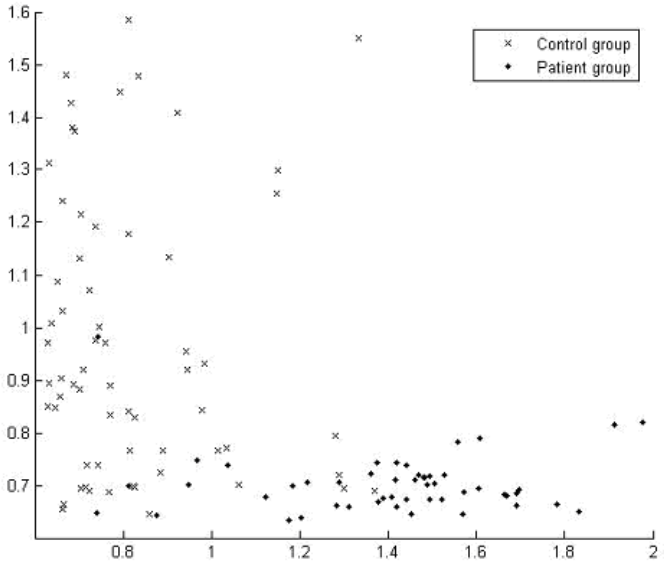


Figure A.4: Scatter plot of the first two principle components on basis of which the classification patient-control group was made.

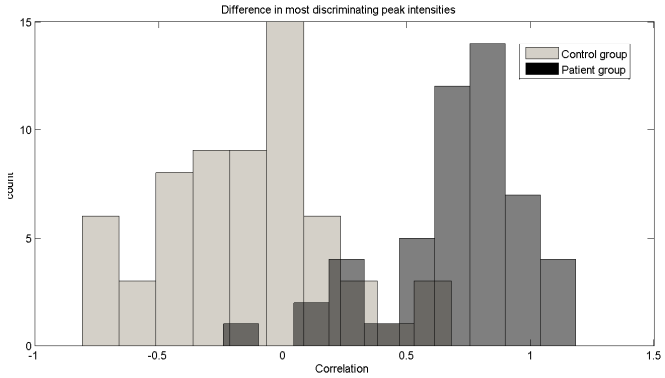


Figure A.5: Histogram showing the difference between the normalized intensities of the two most discriminating "peaks" (bins). The X-axis shows the difference between the normalized intensities of the peaks. On the Y-axis the number of subjects is displayed.

## A.7 Tables

Table A.1: Patient characteristics.

| | **CRC patients** | **Controls** |
|---|---|---|
| n = | 63 | 50 |
| Age (mean) | 62.8 | 49.7 |
| Age (range) | (32.6-90.3) | (25.9-76.6) |
| Male/female ratio | 32/31 | 21/29 |

Table A.2: Distribution and randomization of serum samples of colorectal cancer patients with different TNM stage before and after the MALDI-TOF experiment. The distribution of stadia across plates was performed randomly random fashion and approximately equal proportions.

| | **Plate 1** | **Plate 2** | **Plate 3** | **Total** |
|---|---|---|---|---|
| Colorectal cancer | 22 | 22 | 19 | 63 |
| Controls | 17 | 17 | 16 | 50 |
| **Total** | 39 | 39 | 35 | 113 |

Table A.3: Distribution and randomization of serum samples of different groups over the three MS target plates.

| | **TNM stage** | **Plate 1** | **Plate 2** | **Plate 3** |
|---|---|---|---|---|
| Inclusion | I | 4 | 4 | 3 |
| | II | 10 | 10 | 8 |
| | III | 4 | 4 | 4 |
| | IV | 4 | 4 | 4 |
| | 0 | 4 | 3 | 3 |
| | **Total** | 26 | 25 | 25 |
| Exclusion | I | 0 | 0 | 1 |
| | II | 0 | 0 | 1 |
| | III | 0 | 0 | 1 |
| | IV | 0 | 0 | 0 |
| | 0 | 4 | 3 | 3 |
| | **Total** | 4 | 3 | 6 |

Table A.4: Double cross-validatory classification of serum samples. A positive test results assigns subjects to the CRC group and a negative to the controls. In the horizontal plane the actual histologically confirmed diagnosis is stated.

| | Test results for detection of CRC | | |
|---|---|---|---|
| | **Pos** | **Neg** | **Total** |
| Controls | 45 | 5 | 50 |
| CRC patients | 3 | 60 | 63 |
| **Total** | 48 | 65 | 113 |

Table A.5: Cross-validated classification results for the detection of CRC. TRR is the total recognition rate, Sens and Spec are sensitivity and specificity respectively. AUC is the estimated area under the ROC curve.

| | First week | | | | Second week | | | |
|---|---|---|---|---|---|---|---|---|
| Method | **TRR** | **Sens** | **Spec** | **AUC** | **TRR** | **Sens** | **Spec** | **AUC** |
| PCA selection | 92.6 | 95.2 | 90.0 | 97.3 | 88.8 | 80.6 | 97.1 | 96.8 |

# B

# *Prototype screen mockups*

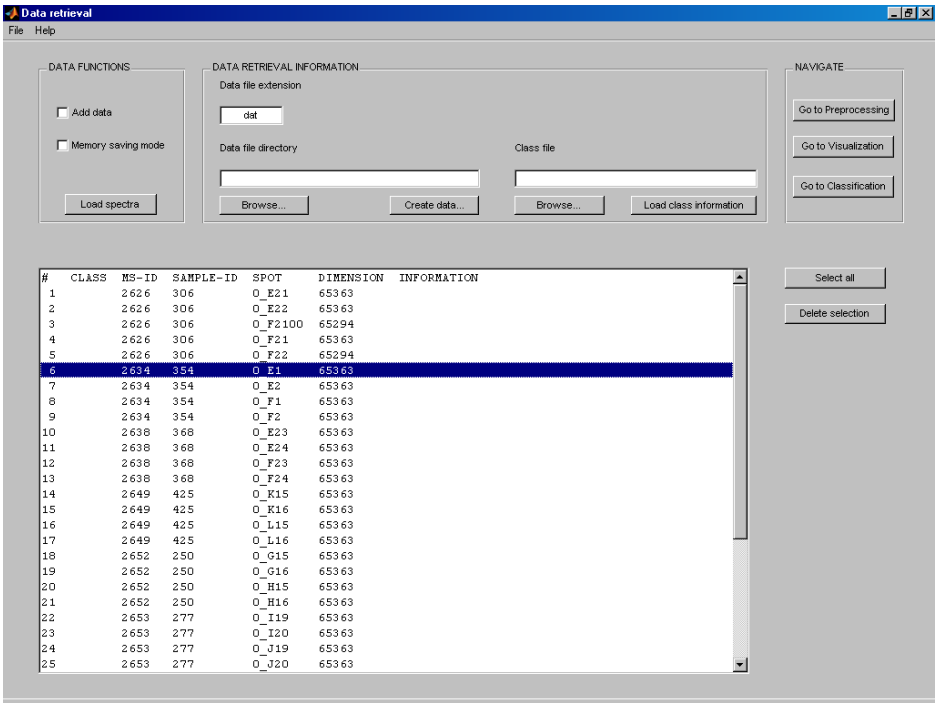


Figure B.1: Screenshot of data retrieval screen

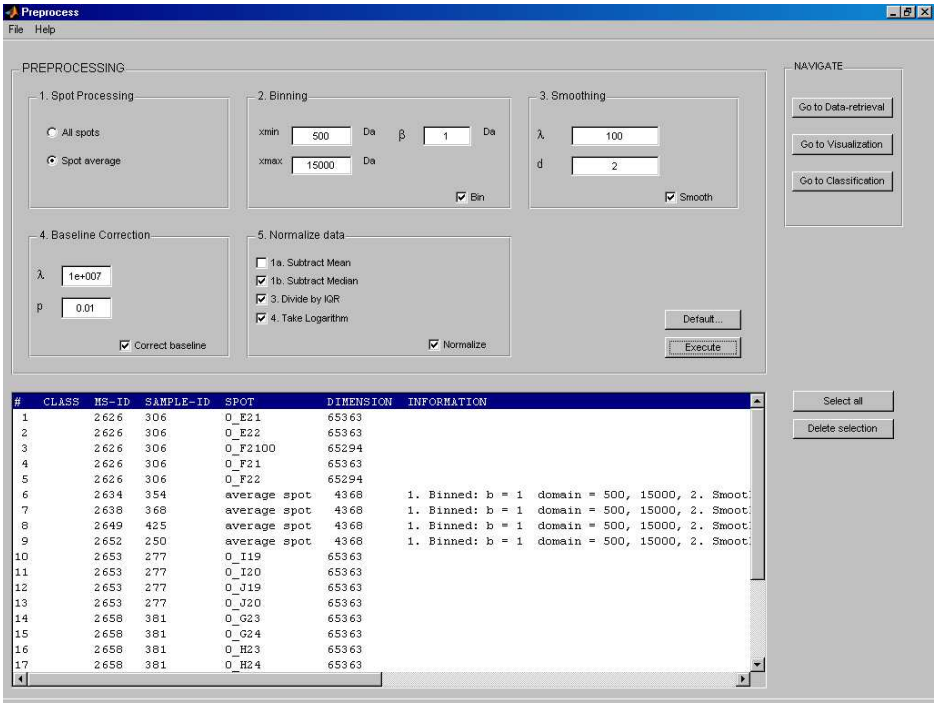


Figure B.2: Screenshot of preprocess screen

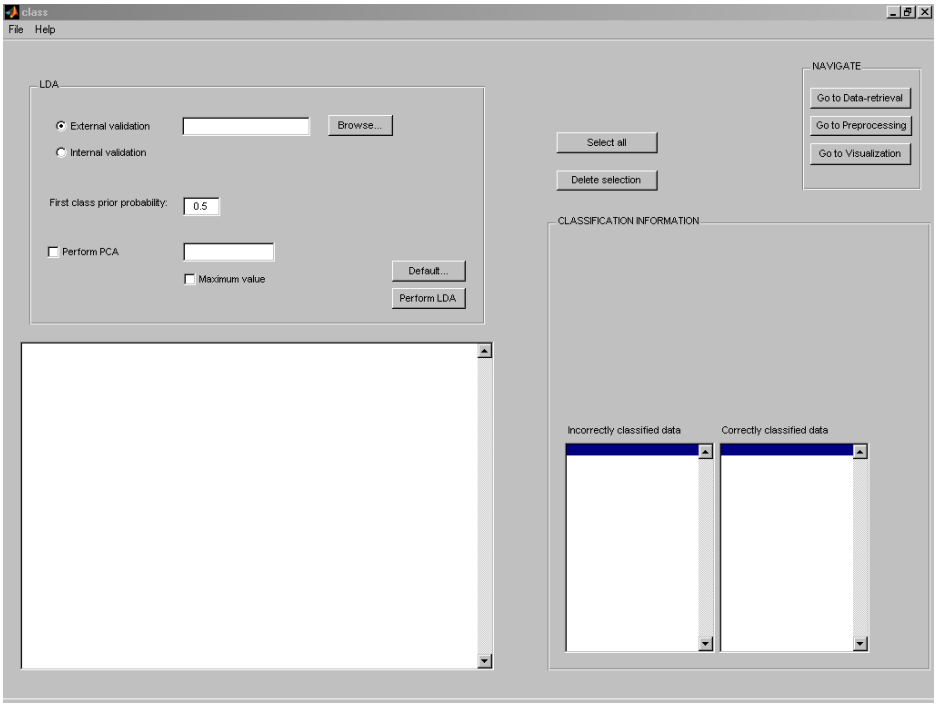


Figure B.3: Screenshot of classification screen

# C

# Pseudo code

## C.1 Data retrieval module

```
function [foundlist] = filesearch(rootdir, findfile, foundlist)
Search a file in the system, in 'rootdir' and subfolders
 INPUT:    -'rootdir' root directory, where to start the
           search
           -'findfile' the filename, of the file you want to
           find or empty, if all files are requested.
           -'foundlist' needed for the recursive function
 OUTPUT:   -'foundlist' an cell-array with the file path
           stored
Construct a cell array containing 'namelist' the names of files
and folders in 'rootdir'
Check whether the 'findfile' is in 'namelist'.
If 'findfile' is found, put its location in the result array
'foundlist'.
Remove '.'  and '..'  from the list to prevent upward seeking.
Perform the search again in the underlying directories.
If an entry in 'namelist' is a directory, search in it.
Sort the 'foundlist'.

   function loadSpec(hObject,handles)
Load the data of selected samples yet when in MSM.
 INPUT:    -'hObject' handle referring to data retrieval
           screen.  Needed for data saving.
           -'handles' list with interface component handles.
List samples, selected in the dataList.
If no samples are selected, go to error handling.
Display loading dialog.
```

```
For each selected sample:
   Display loading progress.
   Load data if data is not yet loaded.
   If file contains invalid data, go to error handling.
Close loading dialog.
Remove samples with bad files from list.
Update data and list.

   function createData(hObject, handles)
Retrieve the data from the designated folder in intern structure
and present the results on the screen.
 INPUT:     -'hObject' handle referring to data retrieval
            screen.  Needed for data saving.
            -'handles' list with interface component handles.
Retrieve the data file directory.
Retrieve the data file extension.
Retrieve whether the data must be added to the current data
or not.
Retrieve whether the data must be loaded in MSM.
When no valid dir or extension is submitted, go to error handling.
Display loading dialog.
Execute filesearch.
When no files are found, go to error handling.
Execute infExtract.
When folder structure is wrong, go to error handling.
Create empty data cell:


                            DATA:
[ms1],[ds1],[spot1],[class1],[location1],[spectrum1],[information1]
                               |
                              ...
         Information:  [type par1 par2 par3 par4]
               Binning:  [1 bin xmin xmax 0]
             Smoothing:  [2 lambda order 0 0]
              Baselinecorr:  [3 lambda p 0 0]
      Normalize:  [4 mn med iq lg] (1 = yes 0 = no)

For each sample:
   Display loading progress.
   Put extracted data in cell.
   When not in MSM, load spectra.
   If file contains invalid data, go to error handling.
Close loading dialog.
Remove empty data elements.
When dimensions are inconsistent, report.
Add new data to current data if wanted.
Execute sort_data
Update data.
Update list.
```

## C.2 Preprocess module

```
function err = errPrep(handles)
Prepares the system for the preprocessing.  It checks on errors
concerning input parameters.
 INPUT:    -'handles' list with interface component handles.
 OUTPUT:   -'err' flag.  Is 1 if an error occurred and 0 when
           no error occurred.
List samples, selected in the dataList.
Check whether an item is selected.
Check whether data is loaded.
Check on normalization errors.
Check on bin errors.
Check on smoothing errors.
Check on baseline correction errors.
If an error occurred return 1 else return 0.

   function exPrep(hObject, handles)
Executes the desired preprocessing algorithms on the data with
the specified parameters and updates the data and the list.
 INPUT:    -'hObject' handle referring to preprocess screen.
           Needed for data saving.
           -'handles' list with interface component handles.
List samples, selected in the dataList.
If spot average is desired:
   Display preprocessing dialog.
   For each unique sample:
      Display loading progress.
      For each sample instance:
         If in memory saving mode, load data this instance.
         If file contains invalid data, go to error handling.
      Else:
         List all data with same ms nr.
      When no error occurred:
         Create a matrix of the listed data.
         When dimensions are inconsistent, go to error handling.
      When no error occurred:
         Add location-info.
         If the spot info is not empty and
         If for all spots the inf is the same:
            Add information of the first sample instance
            to the new data instance.
         Add ms-nr to new data instance.
         Add sample nr to new data instance.
         Add 'average spot' to new data instance.
         Add class lab to new data instance.
         Add average spectra to new data instance.
         Execute [x y inf] = binHandle(handles, x, y).
         Execute [x y inf] = smoothHandle(handles, x, y).
         Execute [x y inf] = baseHandle(handles, x, y).
```

```
   Execute [data inf] = normHandle(handles, data).
   For each new data instance:
      If normalization is done, add normalization information.
      Delete old data.
   Close loading dialog.
If each spot is treated as separate sample:
   Display preprocessing dialog.
   For each selected sample:
      Display preprocess progress.
      If memory saving mode, load data this instance.
      If file contains invalid data, go to error handling.
   If no error occurred:
      Execute [x y inf] = binHandle(handles, x, y).
      Execute [x y inf] = smoothHandle(handles, x, y).
      Execute [x y inf] = baseHandle(handles, x, y).
   Execute [data inf] = normHandle(handles, data).
   When dimensions are inconsistent, go to error handling.
If an error occurred during preprocessing, go to error handling.
Close loading dialog.
Remove samples with bad files from list.
Add new data.
Execute data = sort_data(data).
Updata data.
Update list.

   function [data inf] = normHandle(handles, data)
Handles the normalization of the selected data.
 INPUT:    -'handles' list with interface component handles.
           -'data' cell with data on which normHandle is
           applied.
Create matrix of all spectra.
When desired, subtract median.
When desired, subtract mean.
When desired, divided by inter quartile range.
When desired, take logarithm.
Create information update data.
```

## C.3 Classification module

```
function performLDA(handles)
Performs an LDA analysis on the data.
 INPUT:    -'handles' list with interface component handles.
If external validation is wanted, and the file is bad, go to
error handling.
If no valid first class prior is submitted, go to error handling.
If no valid data is selected, go to error handling.
```

```
If not all samples are assigned, go to error handling.
If there are more or less than 2 classes, go to error handling.
If no valid number of pc's is submitted, go to error handling.
If more pc's are submitted than the size of the dataset -1,
go to error handling.
When external validation is wanted:
   Display loading dialog.
   Load external validation SPECTRA file.
   If file contains invalid data, go to error handling.
   Close loading dialog.
Display analysis dialog.
For each sample:
   If no data is loaded, go to error handling.
   Else:
      Display analysis progress.
      Create training set.
      When the dimensionality is inconsistent, go to error handling.
If training set dimension is over 500, recommend PCA.
Create class matrix.
Create class priors.
If external validation is desired:
   List validation data.
   List validation classes.
   Display analysis dialog.
   If validation data is not valid go to error handling.
   For each sample in validation data set:
      Display analysis progress.
      Create validation set.
      If val set has different dimensionality, go to error handling.
   If val set has different dim than training set, go to error.
   If desired, perform PCA.
      Display pca dialog.
      Execute [u e v] = spec_pca([trainset; valset]).
      Create principal components.
   Display lda dialog.
   Retrieve pc dimensionality if defined.
   Else, dimensionality is maximal.
   Execute [cl,prob,den,coef,dim]=spec_lda(tset,vset,c, p,dim).
   If class assignment is invalid, go to error handling.
   Assign correctly defined samples.
   Assign wrongly defined samples.
If internal validation is desired:
   List validation class assignments (== train classes).
   If desired, perform PCA.
      Display pca dialog.
      Execute [u e v] = spec_pca([trainset; valset])
      Create principal components.
   Retrieve pc dimensionality if defined.
   Else, dimensionality is maximal.
   Display lda dialog.
```

```
   For each sample in dataset:
      Display lda progress.
      Select one sample as validation.
      Use the rest as trainingsset.
      If there are 2 classes in trainingsset:
         Execute [cl,prob,den,coef,dim]=spec_lda(tset,v,c, p,dim).
         If class assignment is valid:
            Assign correctly defined samples.
            Assign wrongly defined samples.
   Check all samples are assigned.
Close dialog.
Execute wrongList(h,w,c,p,[],2) if there are wrong assigments.
Else display 'All data correctly classified'.
Execute corrList(h,c,p,[],2) if there are correct assignments.
Else display 'All data incorrectly classified'.
Display 'Cross-validation' when internal validation is done.
Display size of validation set when external validation is
done.
Compute recognition rate.
Display size of validation set, size of trainingsset, number
of correct assignments, number of incorrect assignments, recognition
rate and used principal components.
```

## C.4 Visualization module

*function plotSpec(specx, specy, way, visop,hObject, handles)*

```
Plots the spectrum in designated axes with the designated representations:
heat map, normal plot or correlation weight plot in designated
visualization (way).
 INPUT:     -'specx' (multiple) spectrum x-vector(s).
            -'specy' (multiple) spectrum y-vector(s).
            -'way' matrix with flags for different ways:
            [mean mirror hold].  1 = on, 0 = off
            -'visop' character for representation:  'H' = heat
            map, 'N' = normal plot 'P' = weight plot.
            -'hObject' handle referring to visualization
            screen.  Needed for data saving.
            -'handles' list with interface component handles.
If heat map is requested:
   Select figure 1.
   Retrieve and round contrast.
   Execute heatplot(specx,specy,con).
   Add image to history list.
If a weight plot is requested:
   Retrieve the principal components.
   If mirror plot is requested, swap spectra.
```

```
   Execute [u,e,v] = spec_pca(x).
   Select only the requested weights.
   If hold is on:
      Hold the image in figure 1.
      Plot the principal component(s) in figure 1.
      Set the x- and y-axis limits to extreme values within
the image.
   Else:
      Plot the principal component(s) in figure 1.
      Set the x- and y-axis limits to extreme values within
the image.
   Add image to history list.
   Unhold the image in figure 1.
If a normal plot is requested:
   If mirror plot is requested, swap spectra.
   If hold is on:
      Hold the image in figure 1.
      Plot the spectrum(s) in figure 1.
      Set the x- and y-axis limits within the image.
   Else:
      Plot the spectrum(s) in figure 1.
      Set the x- and y-axis limits within the image.
   Add image to history list.
   Hold the image in figure 1.
Update history.

   function h = heatplot(rx,ry,con)
Plots a heat map of the spectrum (rx, ry) with contrast value
'con'.
 INPUT:     -'rx' spectrum x-vector.
            -'ry' spectrum y-vector.
            -'con' contrast indicator.
 OUTPUT:    -'h' an image handle of the heat map.
If there are negative intensity values:
   Add minimal intensity to all values.
Invert the gray colormap (high intensities are now more black).
Normalize spectrum by dividing by maximum intensity.
Multiply spectrum by contrast.
Create an heatmap image.
```

# D

# *SPECTRA questionnaire*

Nowadays, experimental research in protein patterns is an interdisciplinary activity. In the current system, generally four groups can be defined, who take part in the research:

- The Physician
- The Analyst
- The Statistician
- The Researcher

In the current system, the research can be divided into generally four activities:

1. The sample collection (e.g. serum collection from patients, database maintenance), which is done in the clinic by Physician.
2. The protein-pattern generation (e.g. preparing MALDI plates, MALDI analysis execution), which is done in the laboratory by Analyst.
3. The data analysis (e.g. data preprocessing, classification of different protein patterns), which is done by Statistician.
4. The overall research (e.g. discuss and interpret results, writing articles), which is done by Researcher.

- This questionnaire is designed to verify and specify the current requirements and to specify the functionalities of SPECTRA.
- This questionnaire is used as a guiding principle for oral evaluation.

**Current requirements**
The current problem formulation is:
*To design and implement a software tool that automates and facilitates the analysis of protein patterns.*

Do you agree with this formulation? (if not, please comment)

The current requirements, established after the discussion with A. Deelder, M. de Noo, B. Mertens, A. Henneman and O. Mayboroda, are:

- Object-oriented approach for easy modification and adaptation to other datasets.
- Intelligent data retrieval system.
- Only two class classification is implemented.
- Data export possible after pre-processing phase.
- Exported data format must be numeric.
- Easy-to-learn and intuitive interface.
- Transparency of used algorithms.
- Comprehensible and extensive visualization options.
- Run on a Windows OS machine.

Do you agree with these requirements? (if not, please comment)

Do you have any other requirements? (if so, please write down)

**Current system**

What are the disadvantages of the current system (as described above)?

What are the advantages of the current system (as described above)?

What are the disadvantages of ClinProtools?

What are the advantages of ClinProtools?

**Data retrieval**
For the data retrieval part of SPECTRA, it is important that the method of class assignment and data file selection is specified.

How would you like to assign classes to the data?

**Preprocessing**

What options would you like to have regarding preprocessing?

What parameters would you like to be able to modify?

Regarding the exported data:

What other information would you like to be contained in exported data?

**Classification**

What algorithms would you like to have integrated in the classification part?

What parameters would you like to be able to modify?

**Visualization**

What visualization options of the classified data would you like?

What information would you like in the output?

What other visualization options through the application would you like?

Name Date

E

# *SPECTRA evaluation assignment*

(This is the SPECTRA evaluation assignment form, used for the evaluation of the prototype and the end product. The user's answers and comments on the prototype are presented in typewriter font. The questions are composed in such a way to optimally try all SPECTRA's functionalities and try all activities needed for the program (e.g. holding the 'Shift' of 'Ctrl' button to select multiple list entries) in as little time as possible.)

**NOTE: please refer to the 'Help'-menu for assistance!**

**Data retrieval**

1. Load The first group of protein spectra in Memory Saving Mode.
2. Add the second group of protein spectra in Memory Saving Mode.
3. Assign class information to the spectra using the Excel file.
4. Delete the unassigned samples.
5. What is the size of the data set? `228`
6. Check de MS-Sample number combinations. Do they comply? `yes`

   Comments: `It takes a long time to load the data, but short compared to the existing software.`

7. Go to the Preprocess screen.

**Preprocessing**

1. Execute in the following order on all data:
   (a) Average the sample applied on different spots.

(b) A binning starting with a bin of 1 Da. on the range of 1000-10000 Da.

Comments: `Comforting to see the progress of the process in percentages.`

2. Execute on the first sample a baseline correction.
3. Execute on all samples a normalization:
   (a) Subtracting the median.
   (b) Divide by inter-quartile range.
   (c) Take natural logarithm.
4. Verify whether the desired actions took place by looking at the sample information in the datalist. `O.K.`
5. Export the spectra. `Not clear where to do this...`
6. Export the SPECTRA data in Matlab format. `O.K. NOW it is clear.`
7. Export the data in Excel format.
8. Clear the list.

   Comments: `Wonderful!`

9. Return to the Data retrieval screen.
10. Load the previously saved SPECTRA data (Matlab format).

**Visualization**

1. Go to the Visualization screen.
2. Make a normal plot of the first spectrum.
3. Make a normal plot of the last spectrum.
4. Make a normal plot of the first spectrum:
   (a) Hold this plot.
   (b) Make a mirror plot of the second spectrum.
5. Make a normal plot of the first five spectra.
6. Make a normal plot of the mean of the first ten spectra.
7. Make a normal plot of the mean first and the last spectrum.
8. Make a heat map of the first spectrum using a contrast of 400.
9. Make a plot of the first 2 correlation weights of the first 10 samples in one figure.

10. Zoom in on the second figure.
11. Reset the view.
12. Save the second figure in .jpg format
13. Clear all figures.

    Comments: `Very good, I am impressed!`

14. Go to the classification screen.

**Classification**

1. Delete the first spectrum.
2. Perform a Linear Discriminant Analysis on the data using:
   (a) An internal validation.
   (b) The first two Principal Components.
3. Assess the results: `Clear and surveyable. Classification information must be more emphasized. Class list must be more emphasized than data list.`
4. Perform a Linear Discriminant Analysis on the data using:
   (a) An external validation with your previously saved data.
   (b) The maximum number of Principal Components.

Comments: `Very user-friendly and surveyable software. Works nicely and fast. Some points of attention: File → export and loading is not clear. Dull appearance. Classification information must be larger and more emphasized.`

F

# Functional diagram

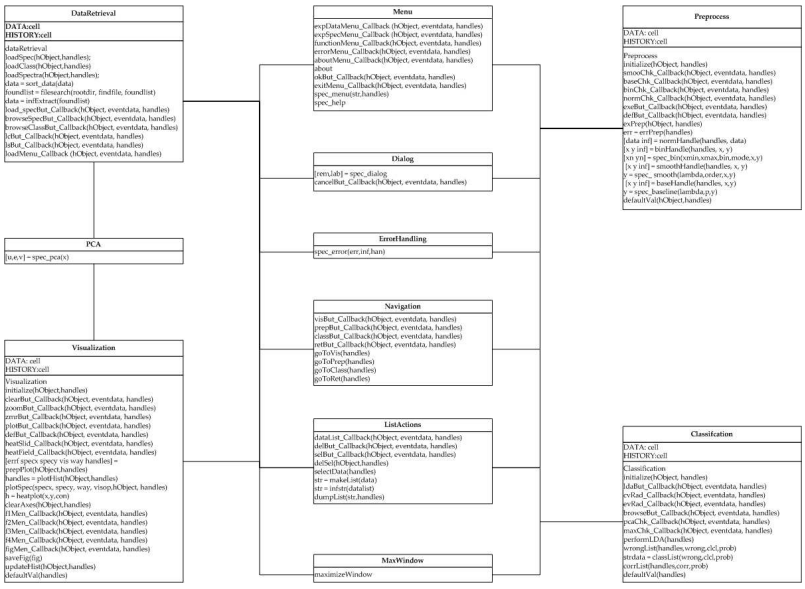


Figure F.1: Function diagram of SPECTRA.

# Bibliography


[1] Alberts B, Bray D, Lewis J, Raff M, Roberts K, and Watson JD. *Molecular Biology of the Cell*. Garland Publishing, 1994.

[2] Gloeckler Ries LA, Reichman ME, Lewis DR, Hankey BF, and Edwards BK. Cancer survival and incidence from the surveillance, epidemiology, and end results SEER program. *Oncologist*, 2003.

[3] Petricoin EF, Ardekani AM, Hitt BA, Levine PJ, Fusaro VA, Steinberg SM, Mills GB, Simone C, Fishman DA, Kohn EC, and Liotta LA. Use of proteomic patterns in serum to identify ovarian cancer. *Lancet*, 2002.

[4] Baggerly KA, Morris JS, and Coombes KR. Reproducibility of SELDI-TOF protein patterns in serum: comparing datasets from different experiments. *Bioinformatics*, 2004.

[5] de Noo ME, Mertens BJ, Ozalp A, Bladergroen MR, van der Werff MPJ, van de Velde CJH, Deelder AM, and Tollenaar RAEM. Detection of colorectal cancer using MALDI-TOF serum protein profiling. *European Journal of Cancer*, 2005.

[6] de Noo ME, Tollenaar RAEM, Ozalp A, Kuppen PJK, Bladergroen MR, Eilers PHC, and Deelder AM. Reliability of human serum protein profiles generated with C8 magnetic beads assisted MALDI-TOF mass spectrometry. *Analytical Chemistry*, 2005.

[7] Tollenaar RAEM Deelder AM Mertens BJA, de Noo ME. Mass spectrometry proteomic diagnosis: Enacting the validation paradigm. *Biostatistics, submitted*, 2005.

[8] Srinivas PR, Verma M, Zhao Y, and Srivastava S. Proteomics for cancer biomarker discovery. *Clinical Chemistry*, 2002.

[9] Morris JS, Coombes KR, Koomen J, Baggerly KA, and Kobayashi R. Feature extraction and quantification for mass spectrometry in biomedical applications using the mean spectrum. *Bioinformatics*, 2005.

[10] Hutchens TW and Yip TT. New desorption strategies for the mass spectrometric analysis of macromolecules. *Rapid Communications in Mass Spectrometry*, 1993.

[11] Merchant M and Weinberger SR. Recent advancements in surface-enhanced laser desorption/ionization-time of flight-mass spectrometry. *Electrophoresis*, 2000.

[12] Savitzky A and Golay MJE. Smoothing and differentiation of data by simplyfied least squeares procedures. *Analytical Chemistry*, 1964 36, 1627.

[13] Whittaker ET. *On a new method of graduation.* Proceedings of the Edinburgh Mathematical Society, 1923.

[14] Eilers PHC. A perfect smoother. *Analytical Chemistry*, 2003.

[15] Lubert Stryer. *Biochemistry.* WH Freeman and Company, New York.

[16] Bruker Daltonics. *MALDI Mass Spectrometry.*

[17] Eilers PHC. Parametric time warping. *Analytical Chemistry*, 2004.

[18] Newey WK and Powell JL. Asymmetric least squares estimation and testing. *Econometrica*, 1987.

[19] Lennon JJ. Matrix assisted laser desorption ionization time-of-flight mass spectrometry. Technical report, University of Washington, 1997.

[20] Foley DH. Considerations of sample and feature size. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1972.

[21] Jain AK. *Dimensionality and sample size considerations in pattern recognition practice.* North-Holland Publishing Company Amsterdam, 1982.

[22] Raudys S. *Statistical and Neural classifiers - An integrated approach to design.* Springer, London, 2001.

[23] Bellman RE. *Dynamic Programming.* Princeton University Press, New Jersey, 1957.

[24] Ransohoff DF. Rules of evidence for cancer molecular-marker discovery and validation. *Nature Reviews Cancer*, 2004.

[25] Pearson K. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 1901.

[26] Duda RO and Hart PE. *Pattern Classification and Scene Analysis.* John Wiley and Sons, 1973.

[27] Mertens BJA. Microarrays, pattern recognition and exploratory data analysis. *Statistics in Medicine*, 22: 1879-1899, 2003.

[28] Webb A. *Statistical Pattern Recognition.* John Wiley and Sons, 2004.

[29] Hastie T, Tibshirani R, and Friedman J. The elements of statistical learning. *Springer-verlag*, 2001.

[30] Van der Mast CAPG. *Mens-machine Interactie.* 2001.

[31] Ruo L, Gougoutas C, Paty PB, Guillem JG, Cohen AM, and Wong WD. Elective bowel resection for incurable stage iv colorectal cancer: prognostic variables for asymptomatic patients. *J.Am.Coll.Surg*, 196: 722-728, 2003.

[32] Gill S and Sinicrope FA. Colorectal cancer prevention: is an ounce of prevention worth a pound of cure? *Semin.Oncol.*, 32: 24-34, 2005.

[33] Adam BL, Qu Y, Davis JW, Ward MD, Clements MA, Cazares LH, Semmes OJ, Schellhammer PF, Yasui Y, Feng Z, and Wright Jr. GL. Serum protein fingerprinting coupled with a pattern-matching algorithm distinguishes prostate cancer from benign prostate hyperplasia and healthy men. *Cancer Res.*, 62: 3609-3614, 2002.

[34] Petricoin III EF, Ornstein DK, Paweletz CP, Ardekani A, Hackett PS, Hitt BA, Velassco A, Trucco C, Wiegand L, Wood K, Simone CB, Levine PJ, Linehan WM, Emmert-Buck MR, Steinberg SM, Kohn EC, and Liotta LA. Serum proteomic patterns for detection of prostate cancer. *J.Natl.Cancer Inst.*, 94: 1576-1578, 2002.

[35] Rai AJ, Zhang Z, Rosenzweig J, Shih I, Pham T, Fung ET, Sokoll LJ, and Chan DW. Proteomic approaches to tumor marker discovery. *Arch.Pathol.Lab Med.*, 126: 1518-1526, 2002.

[36] Yanagisawa K, Shyr Y, Xu BJ, Massion PP, Larsen PH, White BC, Roberts JR, Edgerton M, Gonzalez A, Nadaf S, Moore JH, Caprioli RM, and Carbone DP. Proteomic patterns of tumour subsets in non-small-cell lung cancer. *Lancet*, 362: 433-439, 2003.

[37] Hu J, Coombes KR, Morris JS, and Baggerly KA. The importance of experimental design in proteomic mass spectrometry experiments: some cautionary tales. *Brief.Funct.Genomic.Proteomic*, 3: 322-331, 2005.

[38] Coombes KR, Morris JS, Hu J, Edmonson SR, and Baggerly KA. Serum proteomics profiling-a young technology begins to mature. *Nat.Biotechnol.*, 23: 291-292, 2005.

[39] Ransohoff DF. Rules of evidence for cancer molecular-marker discovery and validation. *Nat.Rev.Cancer*, 4: 309-314, 2004.

[40] Boguski MS and McIntosh MW. Biomedical informatics for proteomics. *Nature*, 422: 233-237, 2003.

[41] J Villanueva, Philip J, Entenberg D, Chaparro CA, Tanwar MK, Holland EC, and Tempst P. Serum peptide profiling by magnetic particle-assisted, automated sample processing and maldi-tof mass spectrometry. *Anal.Chem.*, 76: 1560-1570, 2004.

[42] Diamandis EP. Analysis of serum proteomic patterns for early cancer diagnosis: drawing attention to potential problems. *J.Natl.Cancer Inst.*, 96: 353-356, 2004.

[43] Box GEP, Hunter WG, and Hunter JS. Statistics for experimenters. *John Wiley and Sons, Inc.*, 1978.

[44] Cox DR and Reid N. The theory of the design of experiments. *Chapmann/Hall CRC*, 2000.

[45] Ripley BD. Pattern recognition and neural networks. *Cambridge University Press*, 2005.

[46] Seber GAF. Multivariate observations. *John Wiley and Sons Inc.*, 2005.

[47] Stone M and Jonathan P. Statistical thinking and technique for qsar and related studies. *Journal of Chemometrics*, 7: 455-475, 1993.

[48] Stone M and Jonathan P. Statistical thinking and technique for qsar and related studies. part ii: Specific methods. *Journal of Chemometrics*, 8: 1-20, 1994.

[49] Stone M. Cross-validatory choice and assessment of statistical predictions. 36 ed. 1973.

[50] Qu Y, Adam BL, Yasui Y, Ward MD, Cazares LH, Schellhammer PF, Feng Z, Semmes OJ, and Wright Jr. GL. Boosted decision tree analysis of surface-enhanced laser desorption/ionization mass spectral serum profiles discriminates prostate cancer from noncancer patients. *Clin.Chem.*, 48: 1835-1843, 2002.

[51] Somorjai RL, Dolenko B, and Baumgartner R. Class prediction and discovery using gene microarray and proteomics mass spectroscopy data: curses, caveats, cautions. *Bioinformatics*, 19: 1484-1491, 2003.

[52] Baggerly KA, Morris JS, Wang J, Gold D, Xiao LC, and Coombes KR. A comprehensive approach to the analysis of matrix-assisted laser desorption/ionization-time of flight proteomics spectra from serum samples. *Proteomics*, 3: 1667-1672, 2003.

[53] McLachlan. Discriminant analysis and statistical pattern recognition. *John Wiley and Sons Inc.*, 2004.

[54] Hand DJ. Construction and assessment of classification rules. *John Wiley and Sons Inc.*, 1997.