Metabolomic mass spectrogram analysis

first steps in pattern classification of mass spectrograms in bioinformatics

Crombach, A.B.M.

Award date:
2003

TECHNISCHE UNIVERSITEIT EINDHOVEN

Department of Mathematics and Computing Science

MASTER'S THESIS

Metabolomic Mass Spectrogram Analysis

First steps in pattern classification of
mass spectrograms in bioinformatics

by
A.B.M. Crombach

Supervisor at TUE : prof. dr. P.A.J. Hilbers
Supervisor at UE : dr. Z.R. Yang

Eindhoven, August 2003

Abstract

This document is a report of a thesis project. The research field of the thesis project is called bioinformatics. It is a study in the analysis of metabolomic data. The task is modelled as a pattern recognition task.

The data set is derived from the fruits of tomatoes. Resulting from a mass spectrometry experiment, we have peak-tables from two parental lineages and four hybrids, so-called introgression lines. The high-level goal of the project is to explore the possibilities and obstacles of processing such peak-tables. In practical terms it means we attempt to extract features that reveal the structure of the data. By performing free, unsupervised classifications the features are examined and the data set is explored.

We encounter several problems. In the preprocessing stage of pattern recognition, we are faced with noise reduction. As we have replicate samples, the reduction is based on matching these replicate tables with each other to define what is noise. We determine the boundary between noise and real peaks with two methods: hand-made regressions and a data-driven approach. In addition, we design two methods to search for noise and real peaks in the tables, given a boundary. One is an exact method, that considers all possible combinations, and the other an approximation that is much faster.

During the feature extraction stage, three fundamental problems of peak-tables become obvious. The lengths of these tables varies across the data set, even for replicate samples. The accuracy of peaks is not perfect and distances between peaks are irregular. We propose several solutions that deal with these problems.

We explore three solutions furher in the free classifications. They have successfully been applied in combination with HCA or PCA on the two parental lines. They show there is information in the peak-tables that discriminates on the level of species (tomato lineages). They have not been successful in categorizing the special type of cultivars: introgression lines.

The current techniques provide clues about the information-rich parts of peak-tables. In the future, challenges lie ahead in each phase of pattern recognition. The noise reduction is not perfect: alignment procedures may help to automate the process. And new feature extraction methods may be designed that combine the information-rich parts, possibly leading to the clustering of introgression lines.

Contents

Chapter 1

Introduction

Traditionally, biological sciences view life on earth as a hierarchy of systems. These systems are seen as levels of organization that influence each other. In figure 1.1 one subsequently sees the earth as a system, a rain forest (ecosystem), a tomato plant (individual organism) and one of its cells.

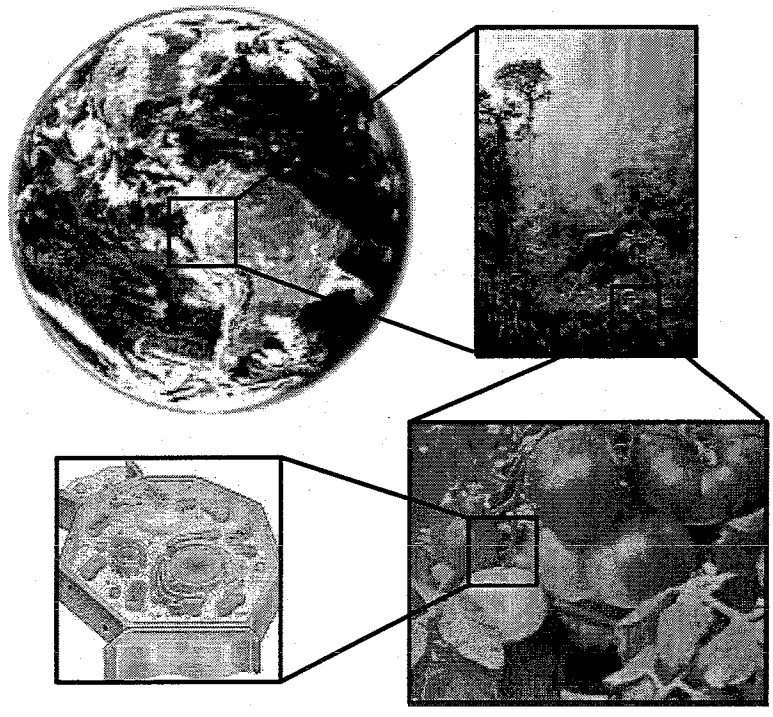

Figure 1.1: Levels of organization in biological systems.

The analogy continues within a cell. As depicted in figure 1.2, one may distinguish several systems as well. The levels of organization are slightly more abstract here. In figure 1.1 physical entities are the systems, each one contained in the other. In a cell, biomolecules form "hierarchical layers", not composed of another, but existing along side. It is the flow of information that determines the hierarchy. We introduce those layers together with the related scientific areas.

1.1 The *omic* disciplines

For long biology has been an area where reductionism and hypotheses-testing were the approaches to set up theories and to perform experiments. In the past couple of decades this has changed. With the advent of *genomics* in the late seventies and in its slipstream *transcriptomics*, *proteomics* and *metabolomics*, the underlying principles of life are researched in a holistic manner.

Genomics is concerned with the nucleotide sequences of organisms (DNA), the genomes. The genome is the bottom layer. It contains the hereditary information that, for instance, makes children look like their parents. The most famous achievement is the sequencing of the human genome (HGP) [55]. Sequencing organisms is becoming a routine job [42, 43], as the major technical problems have been solved. Methods are mainly being refined [56] and there exist standards for data format and storage. This maturity has caused scientists to shift their focus to the expression of genes and the mapping of genes to proteins [29, 46].

When genes are expressed, they are transcribed to messenger RNA (mRNA). *Transcriptomics* is the area, which researches such gene expressions. It has been growing quickly, since microarray experiments have become feasible [39]. Scientists are now able to perform high throughput experiments. They can measure the activity of whole genomes in one step [34]. Unfortunately, progress in this area is limited. Two causes are of technical and biological origin. The lack of data standards obstructs the combination of different sources to obtain new results, to verify conclusions or to test new hypotheses. The second, biological, reason is that regulatory networks of gene activation and inhibition appear to be a complicated area [33]. Two decades of research have passed, yet no complete network has been unraveled.

In turn, mRNA is translated into proteins. These biomolecules serve as enzymes, signal peptides, transporters and so on. Mapping genes to proteins and thus assigning a biological[1] function to one or more genes is a part of *proteomics*. Other aspects, such as structure [16], chemical properties or protein-protein interactions [10, 23] are also investigated in this omics field. At the moment most experiments are performed on a small scale only (2D gel electrophoresis) [9], but new technologies for high throughput and large scales are being developed [16, 41]. One should, for instance, think of an analogous tool to the microarray: the protein array.

[1] A *biological* function is (1) some chemical activity, (2) a time of presence during the life cycle of a cell and (3) a location in the cell.

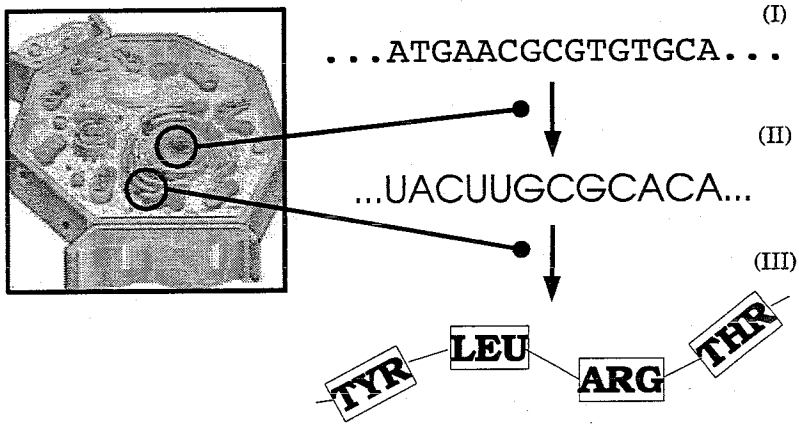

Figure 1.2: Levels of organization within a cell. (I) The genome level; a part of a DNA sequence, (II) The transcriptome level; a partial mRNA polymer, (III) The proteome level; the primary view of a protein. The arrows from I to II and II to III show the flow of information. The localization within a cell of these 'information transfers' is depicted too.

Proteins not only interact with each other, a large amount of them is dedicated to the regulation of the metabolism. The youngest field, *metabolomics* [8, 25] is the accompanying discipline. Its ultimate goal is the quantification of all metabolites, the fourth hierarchical layer. Metabolites are the biochemicals (passively) involved in the metabolism of a cell. The metabolism is defined as the chemical processes occurring within a living cell or organism that are necessary for the maintenance of life. Some biomolecules are broken down to yield energy while other substances are synthesized[2]. Examples of metabolites are sugars, lipids or amino acids. It is estimated that in the plant kingdom about 200,000 different metabolites exist. A well-known example of a chemical process is the break-down of glucose for gaining energy, called the Krebs cycle or citric-acid cycle. It is depicted schematically in figure 1.3. Scientists have realized that these cycles and pathways are in fact densely interconnected networks. Analogous to the regulatory networks in transcriptomics, the mapping of metabolic networks is an important goal in metabolomics [12, 35].

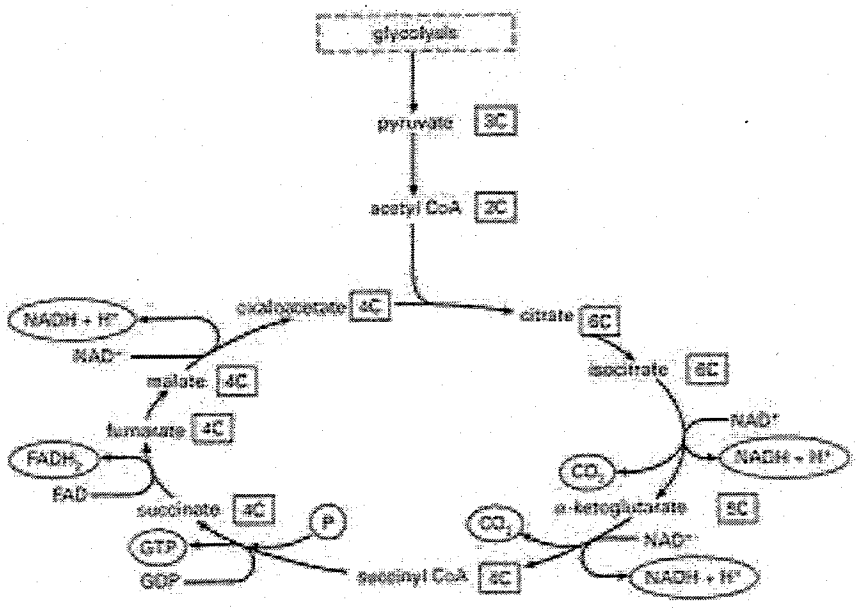

Figure 1.3: The Krebs cycle or citric acid cycle. The boxes indicate the number of carbon atoms in the biomolecule. Taken from "A Dictionary of Biology". Oxford University Press, 2000. Oxford Reference Online.

Four sub- or side-areas are recognized [26]. *Metabolic target analysis* aims at the detection of specific compounds. Methods have been taken from analytical chemistry and are so refined that even single molecules can be detected. For a comprehensive analysis of the metabolome, broader fields like *metabolite profiling* and metabolomics itself exist. In the first only a small class of biochemicals can be addressed at once, like only vitamins or only amino acids. Although the range is larger than for target analysis, there is a bias toward certain compounds. In metabolomics one tries to combine several instruments in order to accurately measure all the metabolites. The default acquisition method is mass spectrometry (MS), often preceded by either gas or liquid chromatography (GC/MS and LC/MS). When speed is important, one sacrifices accuracy over performance and throughput: *metabolic profiling*. Depending on the actual task, one can choose between these techniques varying in sensitivity, comprehensiveness and speed [30]. There are research groups that generate huge amounts of data, in the order of ten gigabytes per day [7]. However, the analysis is still in its infancy. Univariate statistics is applied in some cases REF, but scientists mainly use multivariate statistics. The default is to apply Principal Component Analysis (PCA) [51], possibly in combination with Linear Discriminant Analysis [48]. Hierarchical cluster

[2]Definition taken from *The American Heritage Dictionary of The English Language, Fourth Edition.*

analysis has been proposed [24, 26], but its use is still limited. An example is [51].

In the coming years, one can expect biologists to start with the integration of the different fields [24, 53]. It is probably the best way to build an all-encompassing model of life. Although there are still some technical hurdles in the data acquisition that need to be taken, the problems currently arise in the management and accessibility of data, as well as the analysis.

1.2 Bioinformatics

All omic disciplines have the same properties of producing vast amounts of data and troublesome analysis of the data. Here computers start playing an important role: *bioinformatics*. Computer scientists and biologists are working together in the storage and management of data, which means standardizing data formats and creating standard storage frameworks. They also collaborate in the area of analysis. For instance, knowledge on data mining, pattern detection and classification from other computer-science related fields can be applied in this biological context.

There are some ambiguities around the term *bioinformatics* [22, 31] and its precise definition. We define it as the area of data storage, analysis and modelling, where the data originates from experiments in one of the omics. An important area of analysis is called *pattern recognition* or *pattern classification* [50].

The classification and recognition of patterns is usually described as a five stage process, see figure 1.4. We shortly browse through the different phases with an example. Say we have performed microarray experiments on healthy and diseased peppers. The task is to see if we can classify the microarray data into these two categories.

First of all, data needs to be collected. In this case it roughly consists of growing tomato plants, inflaming some to create diseased plants and performing the microarray experiments. This may seem easy, but often the data acquisition is a time-consuming and expensive process.

After the data gathering, it is preprocessed. That means the removal of experimental artifacts, data transformations or for instance the interpolation of missing data points. We apply a ratio transform on our data, since microarrays only show gene expressions that differ from a standard expression level. We also normalize the data by subtracting the mean and dividing by the standard deviation.

It is followed by feature extraction and selection. The extraction provides a compressed representation of the original data. As much information as possible is preserved. Several statistical methods exist for this process, like Principal Component Analysis (PCA) or Sammon's mapping [36]. A connectionist approach can be taken by applying Self-Organizing Maps (SOM) [36] or neuro-fuzzy approaches[3] [19]. On microarrays people often apply PCA and SOM. Especially SOMs can result in nice visualizations of the data REF.

Since the *actual* data in this project is spectral data (see section 2.3) an alternative is to use techniques from Digital Signal Processing (DSP) [40] or Wavelet theory REF.

Tightly coupled to extraction is selection. Feature sets often contain a subset of redundant features, which is known to degrade the performance of classifiers. Although this project is not focused on this part of the pattern recognition, we mention that selecting the best feature set for a given classifier may be viewed as a combinatorial optimization problem. Thus one can apply methods like Dynamical Programming (DP), Genetic Algorithms (GA) [44] or Simulated Annealing (SA) [50]. For the sake of simplicity the phase is skipped for the pepper microarrays.

The final stage is classification of the features and evaluating the performance. Classifiers are usually categorized as supervised or unsupervised. The first can be described as teaching the algorithm which pattern belongs to which class. After this training phase, one can use the algorithm to classify new patterns. Some well-known methods are Artificial Neural Networks (ANN) [17], k-Nearest Neighbor (kNN), Support Vector Machines (SVM) [13], Hidden Markov Models (HMM) [20] and Discriminant Function Analysis (DFA). In the pepper example, an SVM may be trained to categorize the microarrays in two classes: healthy or diseased. One might imagine an application in some factory that processes peppers. The SVM can be a tool to ensure

[3]Though neuro-fuzzy approaches are not so well-known or well-spread.

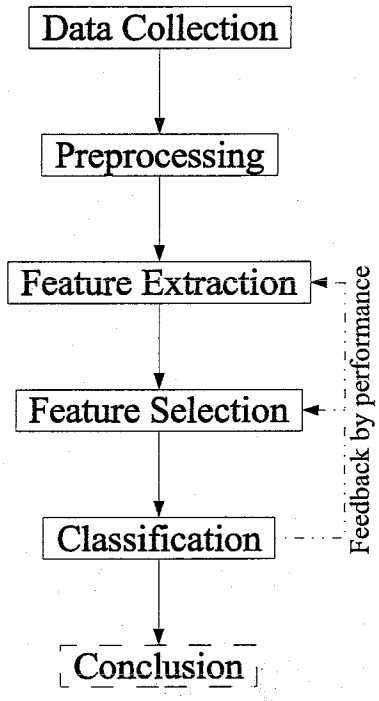

Figure 1.4: The five phases usually associated with pattern recognition. Feedback arrows are drawn to stress that the division in phases is in practice not always so obvious.

the quality of the peppers. After the training, the Support Vector Machine can judge newly arrived batches of peppers if they are healthy or not.

Unsupervised learning is used when one does not want to impose a predefined structure on the data; let the data speak for itself. One may view this as a sort of exploration. A few common methods are Self-Organizing Maps, Hierarchical Cluster Analysis (HCA) and fuzzy clustering. Say we wanted to explore the differences between several pepper diseases. We could have used HCA and see if it is capable of grouping the peppers by disease. In this manner we explore what information is in the data itself.

In this thesis project we use methods of the unsupervised category; we aim to uncover the structure that is already in the data.

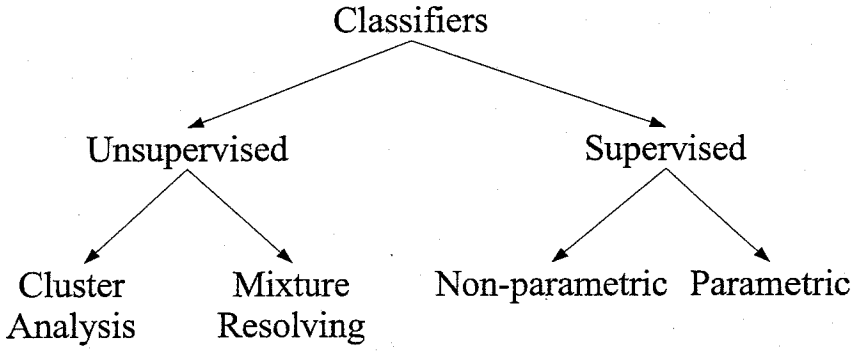

Figure 1.5: A categorization of classification algorithms. An example of a cluster analysis method is Hierarchical Clustering. Mixture resolving methods are k-Means, Principal Component Analysis (PCA) or Independent Component Analysis (ICA) [32]. Examples of non-parametric methods are Nearest Neighbor, Radial-Basis Neural Networks (RBNN) [54] and Back-Propagation Networks (BPNN). Examples of parametric (statistical) methods are Discriminant Function Analysis, Support Vector Machines. These and more methods are described in [50] as well.

Besides the area of pattern analysis, bioinformatics includes creating databases for the management of data and agreeing on standards of format to store the data. A closely related field, by some included in the area of bioinformatics as *dynamical* bioinformatics, is the modelling of evolution, regulatory networks [28, 49], immune-systems, whole hearts, ecosystems and so on. In [18] a more elaborate text on the omics and bioinformatic pattern analysis is found.

1.3 Outline of report

Above we have introduced the relevant fields of research. The thesis project is mainly a bioinformatic task, with the data originating from the metabolic level of tomato fruits (tomatoes).

As a computer scientist, one should be able to perform the project to some extend without background knowledge. But to be truly effective, to realize the difficulties and to be able to communicate with people from other involved research areas, it is important to have at least a general idea (of more profound knowledge where necessary) of the other sciences that are involved.

Therefore we not only present the areas of research in which the project is located, but we also describe the biological and experimental background of the data. After introducing tomato plants and mass spectrometry, the software engineering side of the project is highlighted.

Because the development of several methods for tackling the tomato data is tightly interwoven with the analysis, we present both next to each other. In the subsequent chapters an intuitive, exploratory approach to the analysis is documented; the preprocessing of the data, the feature extraction, the classification and the conclusions we draw from this work.

The first steps in mathematically formalizing the problems, which we have encountered and tackled, are presented next. Finally, an evaluation of the project is given and some possibilities for future work are presented. In appendix B a time schedule of the project is given.

Chapter 2

Preparation

2.1 Project goals

In plant research, metabolomics is an area of growing importance. It is partly due to agricultural reasons, but mostly by scientific ones. First of all, plants are relatively easy experimental subjects to grow and acquire data from. Secondly, the data acquisition has been driven by advances in mass spectrometry (MS) technology. And due to the goals of proteomics (sometimes called *functional genomics*), scientists have begun to explore the fourth level of organization in plant life.

2.1.1 Long term

One of the applications of metabolic research is the creation of libraries of metabolic profiles. You can think of profiles of mutants, diseased plants, wild types or cultivars. The main purpose of such a database is the (rapid) screening of plants against the library, like protein repositories are used for homologue searching. Often metabolomic research is combined with genomics or proteomics [7, 24, 48]. It is interesting to compare changes on, for instance, the genomic level with metabolic changes. A good example is [48]. Sometimes, there are mutations on the genome level, while those alterations do not seem to result in a different organism. In [48], they use metabolome data to reveal such silent mutations. They have been able to expose differences in mutants on the metabolic level, if one compares them to the wildtype. In this way, they have been able to link genes to proteins and to a biological function.

Closely related to the integration of the different omics technologies, is the creation of new cultivars[1]. New plant lines can be screened against the library for many purposes. An example is crossing wild tomatoes with the cultivated one. Cultivated tomatoes are very sensitive to diseases. The wild ones often have disease and pest resistances. One could check if the offspring has acquired disease resistance, while it retains the characteristics of cultivated tomatoes in a metabolomic library. The idea originates from a collaboration between the School of Biological Sciences and the Computer Science Department at Exeter University.

The creation of the libraries and the gathering of data is the first step. But for a library to be fully functional, one needs the ability to extract information from it or to compare new profiles with the stored ones. As the profiles are large sets of data on their own already, the question arises whether it is possible to capture the essential information in a small set of features. In other words, is it possible to develop unique *fingerprints* for each species or cultivar?

From a biological viewpoint, as we explained above, it is interesting to investigate metabolic changes in this manner. From a computational viewpoint, one wants to reduce the size of the dataset, as research groups produce vast amounts of data, see section 1.1. A second, equally important, argument is the analysis. It is the bottleneck of the omics disciplines, because it is

[1] A *cultivar* is a variety or strain of plant produced by horticultural techniques and is not normally found in wild populations. *Cult*ivated *var*iety.

hard to extract the essential information from the huge data repositories. A small set of features is often easier to understand than thousands of data points. A side-effect is that the smaller the data set that needs analysis, the faster a computer can do the analysis. It is also a case of classical reductionism.

2.1.2 This project

The above mentioned goals are long-term aims. The achievements we intend to reach in this project are more of a pioneering and exploratory nature. The objective is to perform a study in the possibilities, complications and solutions of processing metabolite mass spectrograms[2] and extracting meaningful features from them.

The question how to handle peak-tables effectively boils down to a pattern recognition task, with a preprocessing phase, feature extraction and selection and, in this case, a *free* classification. A free classification is a categorization with *unsupervised* learning methods. See figure 1.5 for some example techniques. We have restricted ourselves to unsupervised techniques, as we simply do not know what interesting features are. We do not impose any structure one the data, because we want to explore that information.

But first a few issues have to be addressed before the analysis is given. Some additional background information is presented. The data originates from tomato fruits that have been infused in a mass spectrometer. Biological information on tomato plants and a short introduction to mass spectrometry is given. Furthermore, we discuss some statistical properties of the data.

2.2 Tomato plants

The first stage is data acquisition. Large mass-spectrogram data sets of metabolic origin are rare. They are difficult and expensive to generate. At Exeter University no such data was available. The best alternative has been to ask for data in the scientific community. We started by searching the Internet for research groups or organizations.

After posting the question for metabolic mass-spectrogram data on the e-mail list of Platform Plant Metabolomics [5], we received replies from the following five institutes: University of Sheffield, Sheffield (UK); Institute of Food Research, Norwich (UK); The Noble Foundation, Virginia (USA); Plant Research International, Wageningen (NL) and Max Planck Institute of Molecular Plant Physiology, Potsdam (GER). Eventually only University of Sheffield remained, in the person of dr. S. Overy. The data we received from her is based on the fruits of tomato plants. In return we agreed to provide her department with the software developed for this project and a co-authorship in scientific articles based on this project.

2.2.1 Origins of tomatoes

In general, it is believed that the tomato originates from mountainous regions of the Andes. Relatives of this plant are found in a variety of environments; from high and moist areas in the mountains to dry, coastal regions. The domestication of the tomato probably started with the rise of the first civilizations in South and Central-America. European explorers spread it throughout Europe and Asia during the 16^{th} century.

The common tomato, *Lycopersicon esculentum*[3], is a member of the large family the Solanaceae, also known as the Nightshade family. Various well-known plants belong to this family; potato, pepper, tobacco, petunia, nightshade and thorn apple.

[2] Or their derivatives, peak-tables.
[3] lyco = wolf, persicon = peach, esculentum = edible

2.2.2 Genus *Lycopersicon*

The genus *Lycopersicon* is subdivided into eight species. These can be grouped into so-called complexes on several criteria. Based on the ability to cross with the common tomato, *L. esculentum*, a division of the eight species into two complexes has been established. All species in both complexes are diploid ($2n = 24$) and self-compatible, i.e. the species is capable of reproducing by pollinating its flowers with its own pollens. Together, they form an enormous gene pool, with a lot of variability. The exploitation of the genetic diversity could provide the cultivars with disease resistance or tolerance to harsh conditions, such as heat or salty environments. There are possibly many other opportunities in the agricultural context. Besides the commercial value of the tomato, there is a scientific value. Like wall cress[4] (*Arabidopsis thaliana*) is the main model plant for the omic technologies, the tomato is a model organism for some types of plant research [45].

2.3 Mass spectrometry

Nowadays, one of the commonly used analysis instruments of metabolomics is the mass spectrometer. It is an instrument capable of distinguishing chemical compounds from a sample based on their mass and, at the same time, detecting their relative abundance in the sample. The basic principle behind a mass spectrometer is the motion of a charged particle, ion, in an electric or magnetic field.

A nice introduction in the field is given by [11]. The paper covers the basics of mass spectrometry. An overview is given with respect to different sample introduction techniques, ionization methods and mass analyzers.

2.3.1 How does it work?

We shortly describe the general idea of a mass spectrometer. In figure 2.2 a schematic spectrometer is shown. A sample is introduced via the inlet into the vacuum of the spectrometer. In the source region the molecules of the sample are ionized and accelerated into the mass analyzer. A lot of different analyzers exist nowadays, but they all separate the ions in space or time, on basis of the ions' mass-to-charge (m/z) ratio. The ions arrive in the detector, which transfers that information to the data system. There the signals are combined into a mass spectrogram. For each of the stages, one has a large choice of techniques.

A mass spectrogram is essentially a 2 dimensional vector of m/z ratios and the number of ions measured at each ratio. We discuss both vectors.

An example is to infuse ammonia (NH_3) in the spectrometer. The resulting peak-table is shown in figure 2.3. A peak-table of a tomato fruit is found in figure 4.1 on page 34.

[4] *A. thaliana* is the first completely sequenced plant. The first version of the genome has been published at the end of 2000.

Figure 2.1: The mature red fruit of the common tomato, *L. esculentum*.

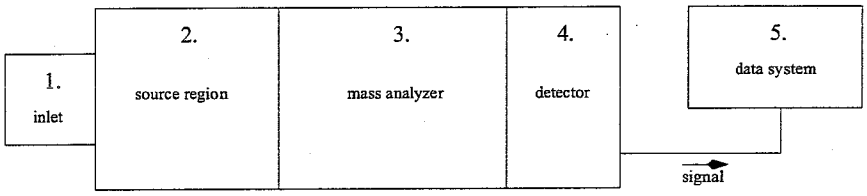

Figure 2.2: The schematic layout of a mass spectrometer. (1) sample inlet, (2) source region, (3) mass analyzer, (4) detector, (5) data system.

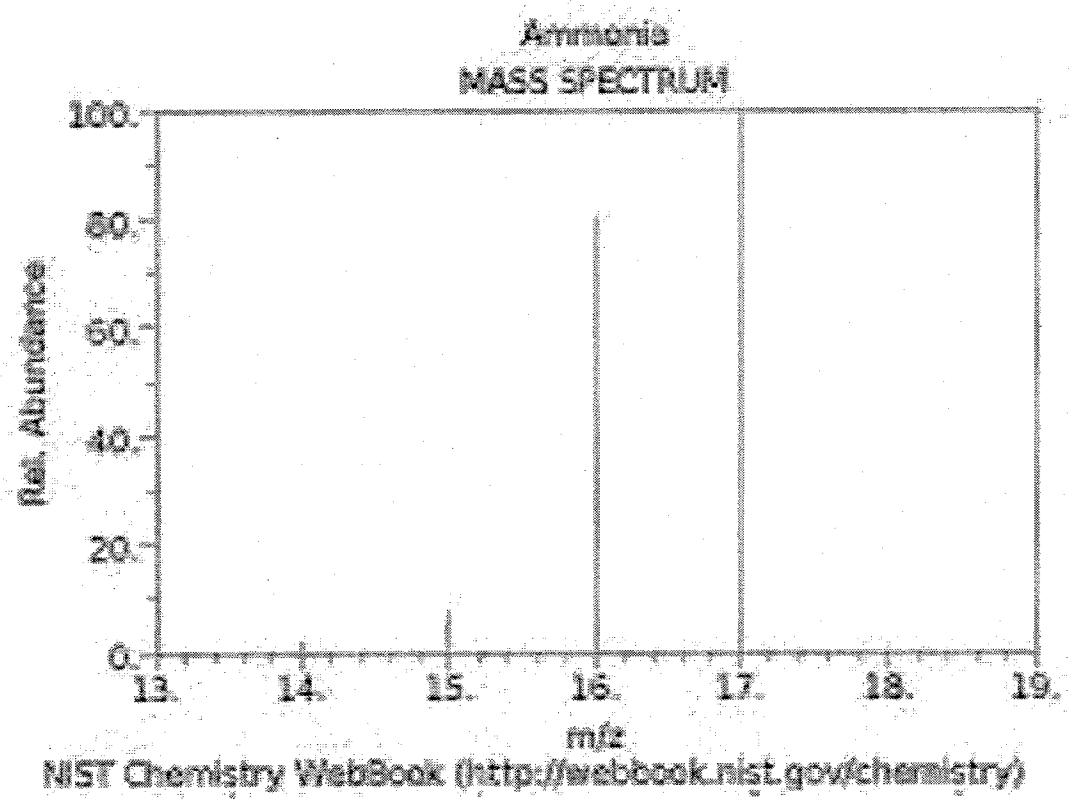

Figure 2.3: The peak-table of ammonia. Due to fragmentation of the ammonia molecules, we have multiple peaks.

2.3.2 Mass-to-charge ratio

The most important concept in mass spectrometry is the mass to charge ratio (m/z ratio) of an ion. The mass (m) is expressed in unified atomic mass units (u). Older, non-SI, units like dalton (Da) are also still in use[5]. The unified atomic mass is defined as 1/12 the mass of a ^{12}C atom. The charge (z) of an ion is the number of electrons gained (or lost) in the ionization process. In most experiments one electron is gained (or lost), which implies that the m/z ratio is equivalent to the mass of the ion. Therefore one often sees that the unit of m/z ratio is 'u' or 'Da'. Theoretically, this is not correct as the atomic mass unit and the charge are pure numbers without a unit.

2.3.3 Peak intensity

The ion count is called the *intensity* or *abundance*. Data processing software is used to locate peaks in the intensity vector. The result forms the peaks of the peak-tables, when we add the m/z ratio of those intensities. The peaks correspond to the relative abundance of the ions. The highest peak in a table is called the *base peak*. Normally, the intensities of the peaks are linearly scaled to the interval $[0 \ldots 100]$, where the base peak has the value 100. We discuss the motivations for such a scaling in the preprocessing stage, section 4.3.2 on page 38.

2.3.4 Possible problems

Complications arise in complex mixtures. First of all, different compounds are ionized at different rates. Thus the same concentration of two chemicals may produce different peak intensities. Second, in such complex mixtures the molecules compete for ionization. There is a dependence on what else, and how much, is present in the sample. Fortunately, the Animal & Plant Sciences Department of the University of Sheffield has performed several benchmark tests to verify that concentration differences between compounds are linearly related to peak intensity differences.

2.3.5 Properties

In addition we mention several properties of mass spectrometers that are relevant to this project:

- the machine was calibrated successfully before each run. Also, the masses are adjusted to an external standard (*lockspray*),
- the spectrometer is focused on an m/z ratio of 150. In the project, metabolites are measured. In general they are rather small molecules, therefore the spectrometer is set to be most sensitive in the lower part of the m/z range,
- S. Overy reports that the accuracy of mass-to-charge ratios decreases as one moves away from the focus ratio. The curve of the varying accuracy is not known,
- the m/z ratios should be accurate to ± 0.003. However, S. Overy indicates that one should allow for a larger ratio-deviation in the calculations,
- peaks have not been pre-selected. In between peaks there was nothing in sufficient quantity to be detected,
- one peak can be more than one compound. Structural isomers, such as glucose and fructose, are not separated,
- one compound can form multiple peaks. Low ionization energy should avoid fragmentation, but adducts with e.g. sodium have been found in the tomato data,
- if peaks have very low intensities, their m/z ratio may not be very accurate,
- one should not assume that every peak is a compound, especially peaks with very low intensity are questionable.

[5] Dalton is an alias for amu, mainly used in biochemistry.

2.4 Data set

We received our data from the Animal & Plant Sciences Department of Sheffield University. It consists of peak-tables from a pilot LC/MS experiment: a proof of concept of the processing and analysis of MS data.

Upon receipt of the data set from Sheffield University, the data is first converted from MS Word format into UNIX/ANSI text format. Technical details regarding the file conversion are found in appendix D. The description that we give in the coming paragraphs is based on information we have received from S. Overy.

2.4.1 Tomato lineages

The data is an ensemble of six different *tomato lineages*. The plants have been grown in a greenhouse at Sheffield University. The tomato lines consist of four introgression lines, as listed in table 2.1, and two parental lines. Several year ago, they have been produced as part of a larger project in Israel by D. Zamir. The parents are the cultivated *L. esculentum*, with line-id 3475 and the wild South-American *L. pennellii* with id 0716.

An introgression line is produced through successive backcrossing and self-fertilization. Backcrossing is repeated crossing of a hybrid with one of its parents. In the plant kingdom, self-fertilization boils down to pollinating the flowers of a plant with its own pollens. During these processes, lines are selected on basis of DNA markers that can identify specific parts of the genome. In this way a new tomato lineage is created, which has the genome of *L. esculentum*, except that it carries a single defined chromosome segment from *L. pennellii*. In figure 2.4 a schematic idea of an introgression line is depicted. See [14, 21, 57] for more information.

Line-id	Introgression line	Relevant chromosome
LA4033	IL 1-4	1
LA4057	IL 5-4	5
LA4067	IL 7-4	7
LA4082	IL 9-2-5	9

Table 2.1: The introgression lines used in this project. The information was obtained from the tomato database [14]. A note on line LA4082; the database mentioned "high soluble solids content", which means high sugar content.

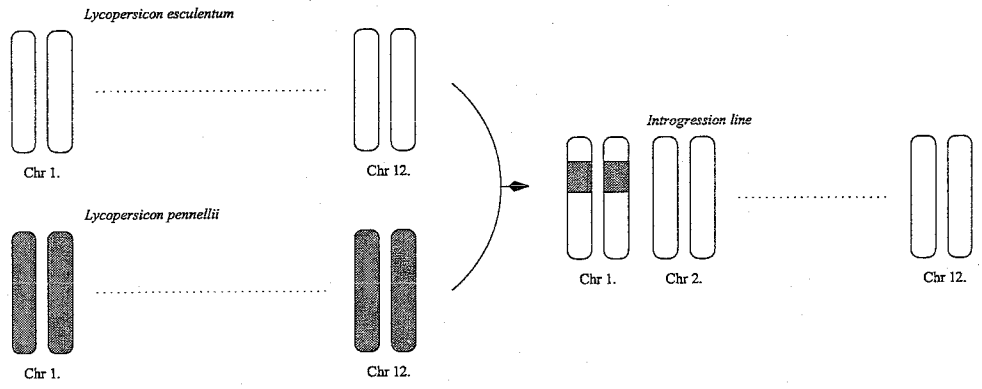

Figure 2.4: The idea of an introgression lineage. The recurrent parent *L. esculentum* and the donor parent *L. pennellii* are shown on the left. A possible introgression line, having a defined segment of the first chromosome pair from the donor, is shown on the right. Note that the introgression line is *homozygous* for the defined segment.

2.4.2 Origin of peak-tables

The peak-tables originate from a pilot LC/MS experiment, without chromatography. One sample is taken from the pericarp (flesh) of each mature green fruits[6]. The extraction technique removes[7] the proteins from the sample. Then the aqueous extracts are infused in the mass spectrometer for four minutes. The spectra are summed up over a three minute period. For each sample this was performed in triplicate, both in positive and negative ion-mode.

Each sample has six spectrograms[8] from which six peak-tables are derived with the software package *Micromass MassLynx* [3]. The format of a peak-table can be formulated in mathematical terms: a vector $(x, y)^N$, where $x \in \mathbb{R}$ is the m/z ratio, significant in four decimals, and $y \in \mathbb{R}$ the peak intensity or abundance with four significant digits. The number of peaks, $N \in \mathbb{N}$, differs per peak-table. As an example a part of a peak-table is shown in table 2.2.

m/z ratio	peak intensity
⋮	⋮
235.2619	7.167e+03
235.3237	5.899e+03
235.3876	7.242e+03
235.4689	7.642e+03
235.6123	2.273e+04
236.1219	2.120e+04
236.4693	2.356e+04
236.7020	3.471e+04
236.9025	1.547e+03
237.0810	1.853e+06
237.2856	3.773e+03
237.3507	2.571e+03
⋮	⋮

Table 2.2: A part of a peak-table. The m/z ratio is (practically) expressed in *Da*, the peak intensity in *counts*, as it is the total of *counts per second* over a three minute period. Peak-tables are ordered by their m/z ratio.

Although there are a few exceptions, one sample was taken per fruit, five fruits per plant and three plants per tomato line. As we have six lineages, a total of 90 samples per ion mode is expected. But some are duplicated, which results in a total amount of 101 samples for negative ion mode and 96 for positive ion mode. Each sample is infused in the mass spectrogram three times, hence the total number of peak-tables is 591.

2.5 Statistics

When one has acquired a dataset for analysis, in general one of the first issues is that one needs to investigate three questions:

- are the observations independent,
- has the data a normal distribution,
- are the residuals uniformly distributed.

The questions in this project are how these issues apply to the whole set of peak-tables and to the peaks in a single table.

[6] Hence the term *fruit* and *sample* are interchangeable.
[7] In chemistry the term *precipitates* is used.
[8] The original spectrograms are not available to us.

2.5.1 Peak-tables

For the set of peak-tables we assume that the tables are independent. According to mass spectrometry experts it might occur that one run (infusion of a sample) influences a next one, but every measure is taken to prevent it from happening (Z.R. Yang, personal communication).

As all plants have been grown under identical environmental conditions, we expect only Gaussian noise. Thus the sampling distributions are assumed to be normal on the sample, plant and line level. That implies uniformity of residuals as well.

We conclude that for peak-tables we are allowed to use statistical methods and models that require the three assumptions named above to hold.

2.5.2 Peaks

Within a peak-table the three issues are more troublesome to answer. Due to competition for ionization, the fact that one compound may cause multiple peaks, that peaks possibly consist of multiple chemicals and also because peaks, which are very close to each other, may influence each others intensity, we argue that theoretically not all peaks are independent. But, on the other hand, we do not have knowledge on these dependencies, nor can we unravel them. Further analysis is virtually impossible, or much more complicated, if we do not assume independence. The best we can do is to remove noise (low-abundance peaks) such that the average distance between peaks increases. That should eliminate the last mentioned problem.

Assuming independence, we can model the intensity of each peak as a Poisson process; the number of compounds hitting the detector at a certain m/z ratio. Thus the whole peak-table is an ensemble of Poisson processes. That implies non-normality of the distribution and non-uniform residuals. These two may be 'corrected' by applying proper transformations.

For peaks we conclude that it is possible to pretend that they are completely independent. But even if so, the other two assumptions do not hold.

2.6 Summary

The preparation of the analysis ends here. We have set our main objective: to study the possibilities, problems and solutions of analyzing metabolite mass spectrograms.

The data we have received from Sheffield University, consists of metabolic data acquired from a mass spectrometer experiment with tomato fruits. Since the original spectrograms are not available to us, we perform the analysis on peak-tables. The data is organized in lineages (or lines), which have plants. In turn, the plants have fruits and from each fruit one sample is taken.

Peak-tables consist of a set of peaks, where each peak has an m/z ratio and an intensity. We model a peak as a Poisson process and hence a table as an ensemble of Poisson processes. Two important properties are that due to the Poisson process, the abundance of a peak varies more as it is higher, and due to MS technology, the m/z ratio of a peak is less accurate as the abundance is lower.

Chapter 3

Software

In the rest of the report, we present the development of the software and the data analysis in an interleaved fashion. The two depend too much on each other to be separated. From that viewpoint, the previous chapter has been an introduction to the analysis side of the project and this chapter describes the first steps of the software engineering side.

We discuss the user - and software requirements, important design decisions and a general approach we have taken when we develop new algorithms. In addition, we provide an introduction to the used programming language, as it is not well-known at all.

In subsequent chapters we design several algorithms for a variety of problems. The detailed documentation of these algorithms is provided in the help-files of the software. The technical details, such as machine characteristics, are found in appendix C.

3.1 User and software requirements

It is hard to formulate user requirements. The problem is that we lack a specific end-user. It is the intention to distribute the software across TU/e, Exeter University and Sheffield University. In addition, it is possibly made available via the Internet. Hence we may view the scientific community as the end-users.

The project goals, both short and long-term, that are set by Sheffield and Exeter, as in section 2.1, are seen as a first version of the user requirements. In one sentence, the objective is to support the analysis in every aspect. The focus is on algorithms to automate analysis steps. In other words, the software is part of an exploration of the pattern recognition trajectory in metabolomic MS analysis.

These statements are not readily translated into software requirements. We distilled the following ones:

- software should be platform-independent, as parties are involved with different operating systems. The Animal & Plant Sciences Department of Sheffield uses mostly Microsoft Windows, while in Exeter and Eindhoven we prefer to develop and use the software in Linux,
- the software is expected to be a package or library of algorithms that facilitate or automate the analysis of mass spectrogram data,
- it is more important to explore the possibilities, than to develop a piece of software with, say, a nice graphical user-interface.

Because the field of research is still in its infancy, there are no standards for the data management. These factors are important in our design. They have resulted in two constraints:

- the *core* algorithms have to allow for easy changes in the future: extendability. The *core* algorithms are the functions that automate the analysis, not the auxiliary functions that, for instance, coerce two data-types or write to a file,

- abstract data types (ADT) should be kept as simple as possible. We do not want to speculate about a nice generalizing format or extra meta-data, as the representation is a volatile part.

The data is organized in a hierarchical fashion. As written in 2.4, the tomato data set consists of six lineages, each lineage of several plants, from each plant five fruits are taken and for each fruit there are two times three replicate tables. Besides this tree-like organization, the different entities have properties like an ion mode for the peak-tables or a Latin *genus - species* name for the tomato lines.

For a small project, such structured data can be saved directly on the hard disk (file system). The tree-structure can be expressed as a *directory* structure. But with more than 500 peak-tables, three hierarchical levels and some extra properties, the file system is not sufficient anymore. Relational database management systems are designed to store such data and provide easy access.

We decided to save the tables themselves in the file system and the meta data in a database, with references to the tables. In this manner, we exploit the fact that file systems are an efficient way for storing large amounts of data and that databases are the best solution for structuring the data. The database also facilitates in the loading, selecting and storing of the data during the analysis. As a requirement it is:

- a relational database management system (DBMS) for the storage of meta data is required,
- a file system for the storage of peak-tables is required,
- a database interface and a file system interface to the (yet unknown) programming language are needed.

In addition to the above motivation, we mention that although the workstation is capable of handling the whole data set, future data sets are expected to be much larger. Hence those sets are not likely to fit into memory as a whole, necessitating a database.

3.1.1 High-level design

The software requirements are captured in a high-level design. We distinguish three main components, as one sees in figure 3.1. They are rather straightforward in their meaning.

- *Library of Algorithms*: it contains the *core* algorithms, ADTs, interfaces to the other components and the auxiliary functions. The *core* algorithms consist of methods for the pattern recognition phases and for visualization. The auxiliary functions are used to glue everything together. One may think of coercion functions, data inspection and so on,
- *DBMS*: this component is responsible for the meta-data database. It provides an interface with the library component, such that meta-data can be selected, inserted and deleted,
- *File system*: it takes care of the storage of peak-tables. The environment provides this system.

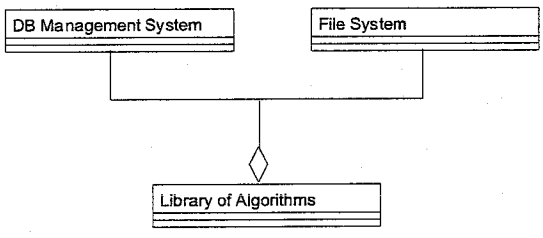

Figure 3.1: High-level design of the software.

With these high-level software requirements, we have selected a programming environment, a language and a database system. They are discussed in the following section.

3.2 Architectural choices

3.2.1 Environment

The open-source platform *Linux* is chosen as the programming environment, to be precise Red Hat Linux 7.3. The main reason is that Linux has proved to be a very stable platform. We have to keep in mind, that the final software needs to be platform independent.

3.2.2 Programming language

Requirements

Throughout the text we have stated some of our criteria already, below a list of all requirements for a programming language is given.

- the language should be as platform-independent as possible; we develop the software under Linux, but the biologists from University of Sheffield use mostly Microsoft Windows,
- the language and its additional packages need to be well-documented; we do not want to focus on the language, but on our own project and its problems,
- there need to be standard, well-documented, ways of creating one's own sets of methods and packages,
- we require a database interface and a file-system interface,
- to aid the analysis, to enhance our understanding and to make the results easier interpretable there should be good facilities for plotting the data graphically,
- for the analysis we require that traditional techniques are available (re-use of algorithms). Although we probably do not use all of the following algorithms, most of them should be available:
  - descriptive statistics; moments of data (mean, variance, skew),
  - n-dimensional projection; PCA, Sammon's mapping,
  - density estimation; spline interpolation (B-spline, cubic splines),
  - signal processing; DFT, wavelets,
  - neural networks; SOM, ANN (feed forward, multilayer),
  - cluster analysis; HCA, k-means, fuzzy-c means,
- high performance is necessary to process all data in short period of time; algorithms need to be fast.

A first choice

The first candidates are from the category "very well-known", general-purpose languages: C, C++ and Java. They seem to fulfill all requirements, although visualization is not their strongest characteristic. Our initial choice for C/C++ instead of Java is based on the fact that for C/C++ there are more libraries and toolkits available. The trade-off between performance and portability is a second reason to choose this language. We expect that the need for speed is more important.

There is one thing we had not fully realized. While creating a support framework for loading and storing the data, it became clear that the amount of work necessary to create certain functionality is large. Also the edit-compile-debug-test cycle is relatively long. And a compiled program does not provide the kind of interaction to test and alter every aspect of an algorithm. It all made us realize that we were focusing on the language, not on the analysis.

Reconsidering the language

So we have reconsidered our choice and formulated a few extra requirements:

- flexibility of the programming environment is important; interactive, interpreted languages provide such flexibility,
- the length of the development cycle must be as small as possible.

With these extra requirements the priority of high performance is decreased, as there is a trade-off between speed and interaction. But we probably gain time with the altered programming approach. And when performance is truly necessary, most interpreted languages offer interfaces with C or Fortran to accomplish this. Two interactive languages are considered: *R* and *Python*.

- R [47] originates from statistical areas. It is an open-source implementation of the S language [15]. R is designed for "computing with data". The developers describe it as a functional, object-based language. It has very good support for graphics. Many users have contributed to R and its commercial equivalent S-PLUS by providing extra packages. These include database interfaces, multi-variate analysis, clustering methods, PCA, Projection Pursuit (ICA) and many more.

  We highlight the package BioConductor [1]. It is a bioinformatics package that provides as most important items, visualization and analysis methods for micro arrays. Because it is focused on transcriptomics and functional genomics (which is a subfield of proteomics), it is not directly of any use to us.

- Python [6] has been developed at Stichting Mathematisch Centrum in Amsterdam, the Netherlands. It is also open-source software. The language is object-oriented and general-purpose. Python is accompanied by numerous packages. The packages range from Internet applications to components for parallel computing. Like R, Python has characteristics of functional languages. One of its strengths is component-based programming, i.e. gluing together several pieces of software. That explains why the majority of packages is related to the Internet, networking and other high-level programming tasks.

  For the scientific community Python provides several chemistry packages. It supports basic geometry (vectors, tensors), elementary statistics, 3D visualization and so on. There exists a bioinformatics package, BioPython [2]. But it is focused on the genomic level: among others it supports sequences, alignment algorithms and BLAST searches.

As a last remark, we mention that an interface between Python and R or S-PLUS exists.

Final choice

Both languages meet virtually all requirements. They provide interactive programming environments, database and file-system interfaces, statistical methods, visualization methods and classification techniques.

We have chosen *R*. Two reasons underlie this choice. First of all R is tailored to data analysis and exploration, whereas Python is a general-purpose language. With R come packages that provide a wide variety of techniques, an even larger array of methods than we required. Within R we have everything we need immediately at our command.

Secondly, it seems easier to create (nice) visualizations in R than in Python. We realized visualizing the data is important in order to get feeling with the data.

Besides these positive reasons, there is a slight disadvantage to Python. Python is a *true* general-purpose language, which means that gluing together all the packages that are needed, takes time. At least more than in R.

3.2.3 Database

As mentioned in section 3.1, we have chosen a relational database management system for implementing the database. It is *MySQL*, an open-source light-weight DBMS. The choice is based on the fact that it has an interface with R and that we do not need more than a light-weight system. The database is implemented with a SQL script, which can also be used to reset the database. The database is purely a support facility of the project.

3.3 The S language

There are several decent introductions into R and the language it implements, S. For the sake of completeness, we describe the most important concepts of the language and provide a few small examples. A full evaluation of the language is beyond the scope of this document. For further information (documentation), we suggest the Internet site [47].

The S language originates from the seventies. John M. Chambers lead the development of the first version at Lucent Technologies (formerly AT&T Bell Labs). During the years the language has been revised and extended. Nowadays it is a fully-equipped language with a focus on statistics and data analysis. Since 1998 there exist two implementations of this language: the commercial S-PLUS and the open-source R.

3.3.1 Characteristics

S is a language for computing with data. The developers claim that every aspect of its design supports you to explore, summarize, visualize and model the experimental data.

Readability

The S language is rather small, in an afternoon one may learn the basics. Most operations can be applied in only one particular fashion, which enhances the understanding of code.

Its readability is also enhanced by the fact that it is vector-based. Often loops are not necessary to perform some action on all data items, a single statement suffices. Unfortunately, with the many scalar, vector and matrix operations that S provides, those single statements sometimes become hard to understand.

The object-orientation of S is well-suited for data abstraction. There is a clear division between data and the associated methods. This is already visible in the syntax. A class definition only contains the data fields. In that sense it is similar to structures or records. The methods are defined separately. One can observe that in the examples of the next section.

The syntax itself is influenced by FORTRAN, C/C++ and functional languages. We find the mixture esthetically not very appealing. A good example is the definition of a function. In the examples of the next section, you can see the binding of the name to the function with the binding operator <-, which is like in a functional language. But the same line also contains { for indicating the beginning of the function body, alike imperative languages as C.

We conclude with the remark that readability of the language heavily depends on it use. We noticed that expressing a computation that is not suited for the language, often looks unnatural[1]. That decreases its readability. On the other hand, data analysis, such as linear regressions, are expressed with certain elegance.

Writability

Because the S language is rather small, it is syntax is quickly learned. The following learning curve is steep, though. In our experience, mastering the language well takes quite some effort in comparison to, for instance, Python. The main theme is that everything is considered to be an

[1]It may also be caused by a lack of experience in the S language

object. This does not mean S is object-oriented. It is better to call it *object-based*, as all entities encapsulate properties. Inheritance has only been introduced in the last two revisions.

As mentioned before, all primitive data types are vector-based. In a language that is aimed at data analysis, this is a natural design. A potentially dangerous feature is *vector recycling*. It means that if two vectors differ in length, the smallest one is repeated to fit the length of the longer vector. It is a very powerful characteristic of S, but it can also lead to hard-to-find bugs or strange results.

The primitive data types are clearly designed for data analysis. The language provides many high-level data types: vectors, arrays, matrices, data frames and lists. The `dataframe` is a type one does not see often in programming languages. It is a simple version of a spreadsheet. It is similar to a two-dimensional array, except that each columns of elements can be of a different primitive type. Say we have 20 documents and we want to register for each document what the most occurring word is and how often is occurs. The result can be saved as a `dataframe` with two columns. The first column contains the most occurring word and is of the type `character`. The second one stores the count and is of the type `integer`. Thus both columns have a size of twenty elements.

All primitive data types support the usage of names to label rows, columns or separate scalars. This is a valuable feature and we have used it extensively in our algorithms. In practice, experiments consist of measurements of some properties of an entity and it is done for many entities. The properties usually have names, such as *weight*, *mass* or *peak intensity*. The entities themselves often also have unique identifiers. In the tomato data, each sample (fruit) has a unique identifier, and so do the plants and lineages (see table 2.1 on page 17).

The usage of labels makes analysis results much easier interpretable, visualizations easier to understand and builds relations between different data representations. In addition, lists can be accessed as associative arrays in this manner (also known as *maps* or *dictionaries*) and many Perl-like facilities for string manipulation exist. The support of labels clearly shows that the S language is focused on data.

Creating your own extensions is easy. Functions are the workhorse of S. There is a vast amount of them already. If you cannot find a function that does not satisfy your needs already, it is usually little effort to extend one that was close to what you were looking for.

The development of new functions often proceeds in an incremental manner. Using the interactive environment (in R it is implemented as a command-line) you can explore some operations on your data. If you decide to group them into a function, you dump the sequence of operations in a file. In a small amount of time, you have turned that into a first version of a function. In the following steps, code may be changed to enhance performance or for generalization. Eventually, you may write some documentation and distribute the function with others in a package. All of these stages are supported by R and S-PLUS.

Object orientation can be used in the form of *classes* and *methods*. As described before, classes only define the data. Methods are used in combination with any number of classes. One defines an abstract method to define the names of the formal arguments. Then individual methods can be defined. They have the same method name as the abstract one and the same formal arguments. They only need to fill in the data types of the arguments. This is illustrated with an example on page 27. The loose coupling between data and code, strengthens us in our idea that S does not support object orientation as in C++ or Java. But that is not what is needed in S, the current design is the best solution for supporting analysis. Data objects are more likely to change per project, methods have a longer life-cycle.

The graphics facilities in S are very good. With a few lines of code one creates publishing-quality plots on any device. The plotting functions provide both high-level procedures to create plots fast and low-level functions for tuning the final appearance of a plot. The first is important in the exploration of data, the second to present your analysis results.

Every aspect of S and R is well-documented. From the basics of the language to more advanced topics such as interfacing with C or FORTRAN, you can find some document that discusses it. In addition, R provides a mailing list that is monitored by the developers and experienced users.

Reliability

The S language is not the most reliable language. It is essentially typeless. That is an error prone feature. Programmers have to realize with each assignment (or binding) that they perform, what the objects are. Additionally, there are some coercions that are applied automatically. That makes it harder to catch type errors.

The last revision of S, introduced a new system of object orientation, called S4[2]. In S4-methods, it is possible to define the types of the arguments, in contrast to functions. This is a first step in type checking.

The main reason for the lack of typing, is performance. The S language is an interpreted language and the execution speed is greatly decreased by type checking. We already observe that when we compare ordinary functions to S4-methods.

On the other hand, one may argue that experienced programmers can handle the freedom. And as S is not intended for enormous intricate software packages, this lack of reliability should not be given too much weight.

3.3.2 Some examples

In an interactive session you give S a task via the command prompt (>) and continuation prompt (+). The tasks may range from simple expressions to definitions of functions. We provide some small examples of tasks.

An expression

The language is vector-based. We create a vector `a` with the concatenation function `c()` and a scalar `b`:

```
> a <- c( 1, 4, 7, 8 ); b <- 2
> a * b
[1]  2  8 14 16
```

A function

We define a function for scaling a *vector* linearly to the range [0...100]. Then we create a vector `a` and scale it. The dot (`.`) has no special meaning in S. It is often used to compose function names.

```
> scale.linear <- function( x ) { x / max( x ) * 100 }
> a <- seq( 1, 10 ); a
[1]  1  2  3  4  5  6  7  8  9 10
> scale.linear( a )
[1]  10  20  30  40  50  60  70  80  90 100
```

The result of the last expression, which is executed by the function, is the return value. One may also use the `return()` function to accomplish this. In case you do not want the result to be displayed, you use `invisible()`. Passing no argument to `invisible()` is returning nothing, i.e. the function is used for its side-effect.

A class

The best way to handle experimental data is to encapsulate it in a class, such that you can give meaning to the representation. We define a class in S4 format:

```
> setClass( "peaktable", representation(  mzratio = "numeric",
+ intensity = "numeric", ionmode = "character" ) )
[1] "peaktable"
```

[2]There also exists an older class mechanism, called S3. It is only available for back-compatibility.

The data fields, in this example `mzratio`, `intensity` and `ionmode` are called *slots* in S. Because S is vector-based, all slots are vectors. `ionmode` is not a vector of characters though, but a vector of strings. Of course, a vector may contain one element, as can be seen below.

If you have an instance of the class, you access the slots with the `@` operator.

```
> testtable@ionmode
[1] "+"
```

A method

Classes are accompanied by methods. Basically, methods are functions, with a dispatching mechanism behind the scenes. The proper method is selected by looking at the classes of the arguments.

```
> setGeneric( "scale",
+             function( x ) standardGeneric( "scale" )
+ )
> setMethod( "scale",
+            signature( x = "peaktable" ),
+            function( x ) { scale.linear( x@intensity ) }
+ )
```

The `setGeneric()` function defines the formal arguments, in this case only `x`. It is an abstract function. Each `setMethod()` must have the exact same formal arguments and can define which classes it handles. Here, the method is only executed if the argument `x` is of the class `peaktable`.

3.3.3 Conclusion

The overview was short, but one should have a general idea of the S language. The language gives the programmer a lot of freedom. That should not be a problem for experienced programmers, but can cause bugs that are hard to trace.

In R and S-PLUS there is a *base* package providing a large range of functions. One can find functions for describing a data set (mean, variance, median), functions for visualization (histogram, density plots, quantile-quantile plots) and a range of functions for modelling your data. If these functions do not satisfy your needs, they are easily adapted or extended. With the same ease, you may add your own functions, classes and methods.

Although it has a rather steep learning curve, it is a versatile and mature language that enables you to do a lot of work in a short time.

3.4 Design

In this project, the design of software for handling metabolomic data is not a clear-cut case of software engineering. Only a small part of the software can be designed beforehand. Namely, the data-objects that represent the data, accompanying methods for coercion to other data-types, simple (console) visualization functions and the database - and file interface.

The design and implementation of several algorithms in such a new field as metabolomics is rather ad-hoc. As we do not know yet what purpose the algorithms will have, what their arguments or results are and how that should be implemented, we can only set up general guidelines or "rules of thumb" to guide our design: a (partial) design template.

Hence we divide the section in two: first the designed software is described, second some guidelines for and observations of the development of new algorithms are given.

3.4.1 Designed software

Database

Together with the peak-tables, two documents are provided that contain extra information about the tables: their properties. This meta data structures the set of tables. The design is straightforward and shown in figure 3.2.

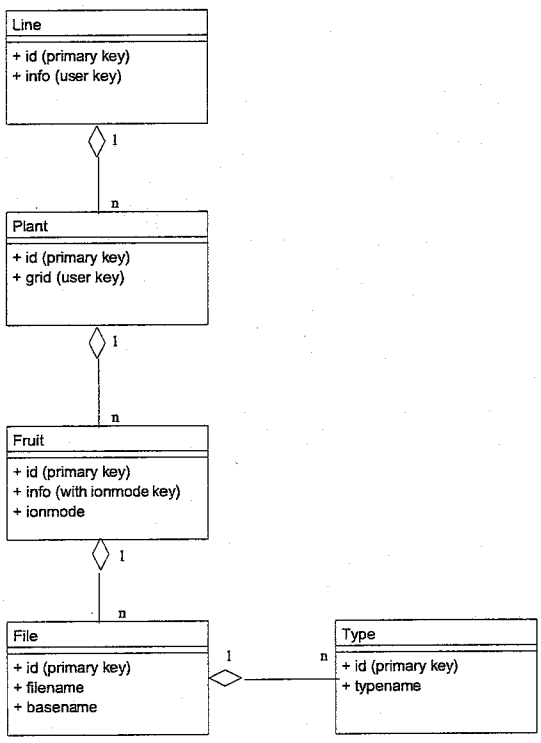

Figure 3.2: UML diagram of the database. The model has been implemented in a relational DBMS.

The UML diagram depicts the hierarchical ordering of the data. Each of the six tomato lines consists of several plants. From each plant, five fruits have been taken, ignoring some exceptions. And each fruit has been infused in the mass spectrometer for three times, resulting in six peak-tables per sample. See also section 2.4. In the database each object has a unique id, the primary key, generated by the database. Additionally, for a line the property `info` is unique, for a plant the `grid` is a key and for fruits the `info` combined with the `ionmode` and the type, is a key as well. These keys are more user-friendly than the database-generated identifiers, when one wants to select certain lineages, plants or files. Therefore the interface of the *Library of algorithms* with the database is based on the user-friendly keys.

Typename	Description
`peak`	The *raw* peak intensities.
`sqrt`	The square-root transformed peak intensities.
`log`	The (natural) logarithmic transformed intensities.
`rank`	The rank transformed intensities.
`sqrtlin`	The square-root and linear transformed intensities.

Table 3.1: The types of peak-table that the database currently supports. The exact meaning of these types will become clear in the following chapter.

The peak-tables are stored in the file system. In order to retrieve them, the location of the files can be created from the fields of `File` and `Type`. The location is formed by concatenating `basename` and `typename` to create the path, followed by `filename` as the file itself. Several types of peak-tables are supported. The object `Type` indicates the type of a peak-table. See table 3.1.

We have already motivated our choice for using a DBMS. By choosing R as the programming environment, we have several extra reasons. According to R documentation, one should use a database if the following holds. During execution of statements, in R all data is resident in memory. And as several copies of the data can be made while R operates on it, the language is not well suited to vast amounts of data. Our data set spans several tens of megabytes, which may cause R to run out of memory.

The strengths that R lacks, are complemented by database management systems (DBMS). They allow access to selections of the dataset and have more capabilities than the (simple) spreadsheets R can handle. In R documentation the strategy of selecting and loading data group by group to perform separate analysis is given as a typical example of database use. It is this kind of tactics that we intend to deploy in this project, given hardware limitations and the size of the data set.

We have restricted the database to the storage of peak-table meta data and the general organization of the data. Because at the time of design we only had a faint idea of which feature - and classification methods we were going to use, we did not know what the meta data of the resulting features or classifications would be. Therefore they are not saved in the database. For these intermediate data objects we have decided to rely on the persistence of objects in R. Each object may be *marshalled* or *serialized* by saving it to disk. Such an object can be loaded in any other R session.

Data objects

The abstract data types have been kept as simple as possible. We introduce two data objects: peak-table and tomatoes.

- *peaktable*: the class *peaktable* is the most important class of the library. It encapsulates the concept of a peak-table: the vector of m/z ratios, the vector of peak abundances, the ion mode and additional information indicating the type of the table. One may think of non-scaled tables, square-root transformed ones and so on. These properties are all essential data.

- *tomatoes*: in this class, the concept of a "whole object view" becomes important. The class *tomatoes* represents the whole data set *with* its hierarchical ordering. Hence a tree-like structure is expected. Yet we design it as a collection of lists. One list contains all peak-tables, the others impose the data structure. As shown in figure 3.3 the additional lists operate as associative arrays. The list `fruits` indicates which peak-tables belong to a particular fruit. In the same way, plants refer to fruits and lines to plants. Because we can use the labels of elements to index the list, we can operate without having to bother about translating indexes to fruits or to plant names and so on.

  As J.M. Chambers describes in chapter 5.1 of [15], the advantages are that we can re-use any method that can be applied on lists and we do not need to traverse trees. That often leads to complicated (and challenging) algorithms, but we do not focus on the language in this project. The re-use of methods is valuable in R, since there is a vast amount of them available. A programming problem that occurs, has probably been solved by someone else already.

In addition to these self-designed classes, we use database connection objects, file connection objects and specialized classes that result from third-party feature extraction or classification functions. One should think of distance matrices or ADTs encapsulating principal components. The third-party packages are listed in appendix C.

Some functions

In R it is good practice (and very useful) to extend common visualization methods. Three such basic methods have been extended by us:

- `show()`, which displays the object on the text console,
- `plot()`, which graphically displays the object (on any device),
- `summary()`, which prints a statistical summary of the object on the text console.

We have also added several *coercion* methods, that enable us to easily change the representation of the data. It eases the use of feature extraction - and classification functions that originate from other packages.

We provide several extra statistical visualizations for peak-tables: one and two-dimensional histograms, box-whisker plots and cumulative frequency plots. An overview of all methods is given in appendix E.

Interfaces

The database and file system interfaces are designed as a set of methods and functions. For the file system, R already provides a platform-independent interface. We extended functions, such that peak-tables may be read or written to disk in a transparent manner. The tables are saved in ANSI/UNIX format.

An additional package, `RMySQL`, provides objects and methods that interface the DBMS. We designed two methods, `import()` and `export()`, that use the meta-data in the database to load or save parts of the tomato data. For instance, we can load the two parental lines by specifying their lineage id, 0716 and 3475. In the same manner, we can load specific plants or even separate fruits.

All the classes, methods and functions are listed in appendix E and described in detail in the help files of the software. Once you have installed our package in R, the help files are automatically added to the off-line HTML help-pages of R.

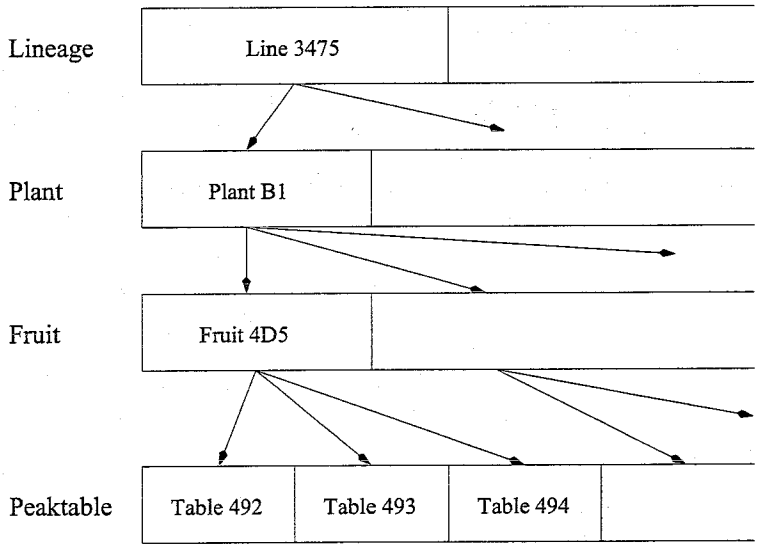

Figure 3.3: The "whole object view" of the tomatoes data set. One sees a set of peak-tables, that belong to fruit 4D5. The fruit belongs to plant B1 as indicated by the arrow. The plant is from lineage 3475. In this manner a tree structure is defined on the set of peak-tables.

3.4.2 New algorithms

As we mentioned before, we can only present some guidelines and reflect on the process. Some of the guidelines have originated from the R documentation and some are consequences of the software requirements.

The core idea of R is to "turn ideas into software, quickly and faithfully". The design of a new algorithm can be described in similar words. When a new idea occurs, we can quickly implement it in R. If the idea appears viable, a series of refinements follows. During this process, we have the following rules.

- The first rule is not to worry about performance. In the R documentation, one is advised to focus on solving the problem. The algorithm must work before it has to be fast.

  Once we are satisfied with the working of the algorithm, it is often fast enough. If not, R provides a profiler that shows where the bottleneck is located in the source code. More than once, the profiler showed different results than we expected. It is a valuable tool.

  If an increase in performance is necessary, there is a well-documented interface with C/C++ or Fortran. The "heavy computing" may be transfered to a C routine,

- The second rule is an implementation template. If an idea is tested and we want to add it to our toolkit, we implement it as a set of methods where the default method uses only native data types. It means *peaktable* or *tomato* objects are not allowed to occur in the signature of the core method.

  The motivation is that in the future data representations are likely to change, but primitive types stay the same. A nice side effect is that if other packages want to use the algorithm, they do not need to construct peak-tables. One can imagine that an algorithm is used for data that is not related to peak-tables at all.

- As a last 'guideline', we elaborate on how to implement a design pattern. In object-oriented programming (OOP), a well-known pattern is a *strategy* [27]. It encapsulates certain functionality and hence makes it re-usable for other objects. The pattern is often used to implement families of algorithms, such that other objects can choose one of them or easily change to another. In our project this pattern would be used a lot, as we investigate different ideas or solutions to the same problems.

  The strategy pattern is implemented quite easily. In R, everything is an object. That includes functions. So we may pass any function F as an argument to another function G. What we have to do, is to agree on a signature such that the function G can use F, even though G does not know exactly how F does its task. The common signature can be enforced partially by using methods. The problem is that if two methods need the same signature, the dispatching mechanism cannot tell the difference between the two. Hence, an error occurs.

  Currently, we implement the strategy pattern with ordinary functions. We do not have the signature problem, but we have to take extra care with checking the argument types.

In addition to these design guidelines, we have made several interesting observations. Two of them have been exploited to reduce the amount of code and to re-use functionality, the other is worth noting.

- Because we have several hierarchical levels, we often want to apply a function on one of them. An example is to find all common peaks of a plant or a lineage. The built-in function `tapply` has a similar purpose, but it cannot handle our data objects. Hence we have implemented a function, `.internal.apply`, which applies a given function on the tables grouped by a given level.

  Re-using the functionality provided by `.internal.apply` has saved time and caused only minor generalization troubles.

- In R every element of a vector or list and every row and column of an array or matrix can be named. We underestimated the usefulness in the beginning. During the project, we realized such name vectors are a powerful way to associate similar data in different objects with each other. If the name vectors are used consistently, the algorithms can work with the data in an almost human-like fashion. One can speak of a row of a matrix, being an index vector of peak-table p, because the identifier of the table is equal to the name of the row.

  Although it is not the most efficient way for a computer, the interpretability is high and it makes programming easier. Imagine we have to think about a mapping from peak-table identifiers to some set of integers, when to translate between the two and so on. One realizes, again, that we would be focusing on the language and the particular implementation, instead of the problem at hand.

- The last interesting observation concerning software design is that we re-use patterns for implementing methods. This is not completely unexpected: methods are implemented according to the second rule we described above. What we do find interesting, is that we only have a few different implementations for the methods that accept a *peaktable* or *tomatoes* object. Unfortunately, for most methods their signature is unique. Hence we cannot generalize this.

3.5 Summary

The data analysis is performed in an interactive programming environment called R. It is an open-source implementation of the S language, a language designed for visualizing, summarizing and exploring experimental data.

The software design has been divided into two groups. The custom-made data representation and loading or storing facilities is one. The other is the set of algorithms, implemented in a general way.

We intended to design and implement algorithms such that they are as data independent as possible. Looking at our guidelines and observations, we conclude that we have done what is reasonable to ensure future maintainability. It is up to others to see if that is truly the case.

We do not expect that the peak-tables or the tomato set object, nor the importing and exporting functions, are useful in future projects.

Chapter 4

Preprocessing

The raw data needs some sort of preprocessing, before we can extract any features. It is necessary to remove artifacts caused by the mass spectrometer and the peak-detection software. In general, the preprocessing consists of a sequence of data transformations, like removal of erroneous results, reducing experimental noise, the reduction of any biases and making data numerically more stable.

When mass spectrograms or chromatograms are preprocessed, we find that the methods are not applicable to peak-tables [37]. This is due to the irregular, discrete format of peak-tables, whereas spectrograms behave as continuous data. We assume commercial software packages that process mass spectrograms reduce the level of noise in their data. But as the software is commercial, the exact working of their algorithm remains hidden. Peak detection software provides options for reducing the level of noise of mass spectrograms, when peaks are located [3]. Yet we do not have spectrograms, we have their derivatives. Even if the level of noise has been reduced in the peak detection process, we are still faced with artifacts of that process: peaks that do not represent any compound, i.e. *noise* peaks.

We have a rather rare opportunity of reducing noise on basis of replicate samples. Unfortunately we have not been able to find any literature on the removal of noise in peak-tables. It seems we have to develop techniques ourselves.

Before we describe the preprocessing steps, we discuss a novel visualization method. Without a proper way to visualize your experimental data, it is hard to get any feeling with the data. The enhanced understanding of the data, combined with some general mass spectrometry knowledge and discussions with S. Overy has resulted in two preprocessing steps. First we scale the peak intensities, followed by noise reduction on the level of fruits (samples).

The noise reduction consists of two stages. First we determine the boundary between signal and noise. The actual noise reduction is described after. It is viewed as a search and we have investigated two types. After the preprocessing, ideally we have one peak-table per fruit (instead of three replicates) that has scaled intensities and hardly contains noise.

4.1 Visualization

Traditionally, mass spectrograms and the derived data, peak-tables, have been displayed as two-dimensional figures. An example is shown in figure 4.1. On the x-axis the m/z ratio is displayed. In the figure it ranges from 50 to 800. Along the y-axis the relative abundance of the peaks is shown. Usually, the highest peak, the base peak, has a value of 100 and the others have been scaled relatively to it.

4.1.1 Novel view

In metabolomics, mass spectrograms are numerous. The pilot experiment alone already results in almost 600 tables. It is difficult for the human observer to detect patterns among so many tables,

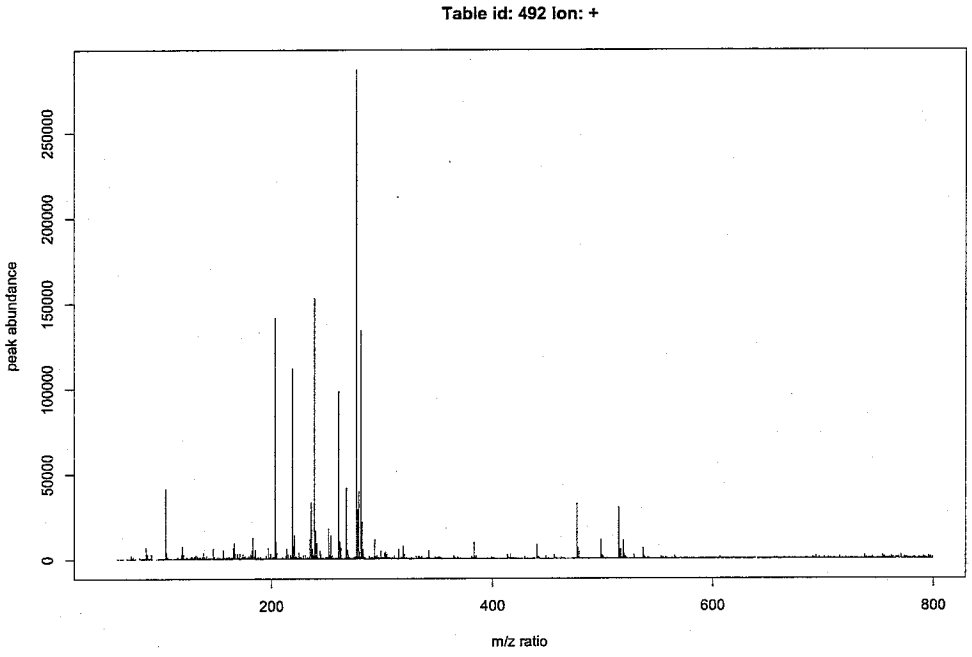

Figure 4.1: A traditional view of a peak-table.

especially when the first ten to twenty highest peaks appear to be the same. Therefore a novel idea for displaying large sets of peak-tables has been developed. The main idea is to view the tables "from above", when you regard the traditional view as "from the side".

We describe the core idea of the method in more detail. Each peak-table is represented as a horizontal bar, along the x-axis. That axis still displays the m/z ratio. The height of a peak determines the color of the bar. An example plot is shown in figure 4.2. It looks rather specific for the tomato data set. That is inevitable, as there is no standard in this field for storing the meta data yet. The core algorithm does not depend on that, but the annotation on the axis does.

A schematic implementation of the core algorithm looks as follows: the input consists of a peak-table t, the number of colors c, the number of intervals n and a range on the m/z axis r.

1. divide the range r into n equal intervals: $b_0, \ldots, b_{n-1}$ Each bin b_i, $(0 \leq i < n)$ is an interval $[l_{b_i} \ldots l_{b_{i+1}})$ on the m/z axis. l_{b_i} is the inclusive lower bound of the interval and $l_{b_{i+1}}$ the exclusive upper bound of interval b_i,
2. generate a color map of c different colors. Let us call this map col,
3. calculate the scaling factor s, such that all peak intensities are mapped to the correct color. $s := \frac{\text{base peak intensity of } t}{c}$,
4. for each b_i, $(0 \leq i < n)$ do:
   (a) select the peaks in t with $l_{b_i} \leq \text{m/z} < l_{b_{i+1}}$. In other words, the peaks with an m/z ratio within the interval-bounds are selected. Let us call the resulting set of peaks P,
   (b) compute the average intensity of P: $\mu(P)$ and round it, such that we have an integer avg. If $P = \emptyset$, then $avg := 0$,
   (c) the color of interval b_i is determined by $col_{b_i} := col[\frac{avg}{s}]$,
   (d) draw a box from l_{b_i} to $l_{b_{i+1}}$ and fill it with color col_{b_i},

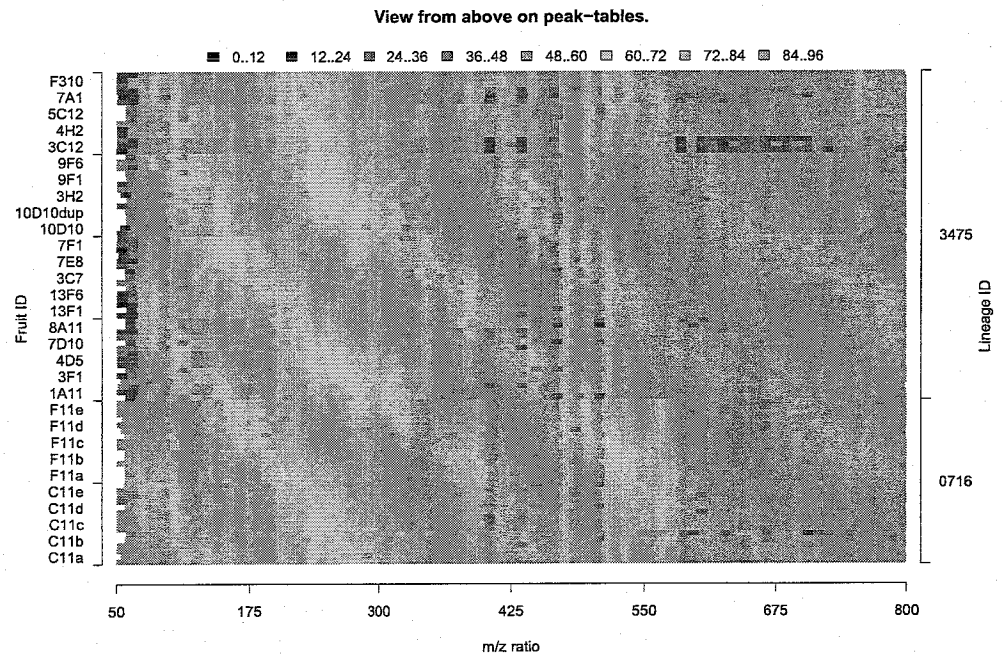

Figure 4.2: A novel view on peak-tables. The two parental lines (*L. pennellii* (0716) and *L. esculentum* (3475)) are shown here. On the left the sample-id is printed, grouped by plant. On the right the line identifier shows which samples belong to which lineage. The number of colors is 100, although the legend on top of the plot shows a maximum of 96. The number of intervals 200.

5. the result is a horizontal color-coded bar.

There are two parameters to adjust the view: the number of colors and the number of intervals. The first one is straightforward, it determines the number of different colors. These are equally distributed from zero to the maximum peak-height. The average intensity in an interval specifies the color of the bar-interval. The latter parameter is a way of granulating the peaks in the table. If you want to investigate trends on a larger range than just one peak, you take a smaller number of intervals. On the other hand, viewing the peaks "on their own" is possible by specifying more intervals.

The intention is to use the new visualization for a better understanding of and view on the data set. Properties like similar peaks specific to plants or patterns across several lineages are easily detected. If we look at figure 4.2, the vertical bands are caused by common peaks. And one can see that there is a significant difference between the two lineages, as they split into two horizontal bands.

The method is available as `plot.heat` (appendix E). The name originates from the fact that the first implementations used a heat color map, where the colors range from dark blue to yellow and white. If one compares the novel view to having several sheets of traditional figures, one realizes the advantage.

4.2 Pre-preprocessing

Before we start the preprocessing stage, we check the data for obviously faulty or corrupted peak-tables. If a fruit contains a corrupt table, i.e. one of its replicate peak-tables is erroneous, the whole fruit is removed from the data set.

After our first view on the data, we categorize two different types of 'obviously faulty' for the positive ion-mode. S. Overy had already warned us for these errors. The first type is that only a subrange of the m/z ratio is present. The second that the intensities have been cut-off at a certain level. Both faults can be detected unambiguously. We assessed each sample by both looking at the "from above" view of the samples, and the individual peak-tables. Such errors occur sometimes, due to wrong parameter settings of the spectrometer or due to the peripheral devices of the machine.

It appears that in negative ion-mode each first sample of a plant has a corrupt peak-table; it is caused by accident. There are no small peaks (noise), because a threshold has mistakenly been set on the spectrometer. Eliminating those tables, would mean removing 20% of the negative ion mode data. We intend to postpone the decision whether or not to remove the corrupt tables. Instead we decide to perform the analysis in this project on the positive ion mode data. In the rest of the report, if there is no ambiguity, we omit to mention we only use positive ion-mode tables. The names and ion-mode of all corrupt fruits are given in table 4.1.

Ion-mode	Names
+	3D10, 7C12, 14A1, 11H7, 14H2, 9D10
−	3F1, 3C7, 9F6, 4H2, 1A6, 3D10, F216, 11H2, 7A6, F229, 3C2, F288, F6, F11, 3A6, 2D10, 1B9, F253, F264, F140, F264Dup

Table 4.1: Flawed samples. The positive ion mode samples have been removed from the set prior to the analysis.

After these pre-preprocessing issues the analysis of tomato lineages is started. During the preprocessing we are only concerned with the fact that we are dealing with peak-tables. At this stage knowledge of the biological side of the data, i.e. that the data represents metabolites of the fruits of different tomato lines, is not needed.

The most important characteristics of a peak-table are repeated for convenience. A peak-table is defined as $(x, y)^N$, with $x \in \mathbb{R}$ the m/z ratio, $y \in \mathbb{R}$ the peak abundance (intensity) and $N \in \mathbb{N}$

the number of peaks. In other words, a peak-table is a tuple of two N-dimensional vectors: $(\bar{x}, \bar{y})$.

4.3 Transforming the data

The first views on the tables make us realize that two transformations are appropriate. We discuss both of them and our motivations.

4.3.1 Non-linear scaling

We start by exploring the data set with descriptive-statistical techniques. In the leftmost plot of figure 4.3 we notice that the range of the peak intensities is rather large. As one can see the minimal abundance often differs from the maximum with a magnitude of 10^4. Also, one observes that most peaks are rather small: half of the peaks are smaller than 10^2. Only a few "outliers" are of the order 10^4 or 10^5.

The problems are that the large range makes computations less numerically stable, for instance when the data is linearly scaled, and that the "outliers" are not outliers. The last statement means that although the tiny fraction of peaks with a high intensity may look like outliers in the traditional statistical sense, we know the contrary is true. These peaks are probably the most accurate ones regarding their m/z ratio, see section 2.3. Hence they are important as reference points between the replicate tables, and, in the feature extraction, they may be useful for creating fingerprints.

In chapter 2 we have written that peak intensities are modelled as independent Poisson processes: the number of compounds hitting the detector at a certain m/z ratio. That implies a non-Gaussian distribution. The peak intensities are basically counts of compounds, and therefore have a non-uniform standard deviation [4]: it is higher for higher counts. Therefore a non-linear scaling of the peak abundances is desirable for three reasons:

- To make the computations numerically more stable,
- To "pull outliers back" into the main cloud of peak heights,
- To remove the variable standard deviation from the intensities.

A nice implication is that the scaling simplifies our analysis. We are allowed to apply methods that assume a normal distribution, instead of 'special' methods for the Poisson or binomial distributions.

The disadvantage is that the moments of the data set change as well and this effect is irreversible. The mean of a scaled variable is (in general) not equal to the original mean, when one back-transforms that value.

In table 4.2 some possible transformations are shown. The square-root transformation is a standard one applied on Poisson distributions [4]. The logarithmic transformation is also allowed. The rank transformation is a rather crude scaling for the peak intensities. We suspect there is information in the relative heights of the peaks, as is explained in the section 4.3.2. Yet the rank transformation removes all that information. Hence we regard it as a last resort; only if the other techniques fail, we apply it.

Name	Transformation
square-root	$(\bar{x},\ \sqrt{\bar{y}})$
logarithmic	$(\bar{x},\ {}^{10}\log(\bar{y}))$ or $(\bar{x},\ \ln(\bar{y}))$
rank	$(\bar{x},\ rank(\bar{y}))$

Table 4.2: Possible non-linear data transformations. $\bar{x}$ is the m/z ratio, $\bar{y}$ is the intensity.

We show the result of the proposed transformations in a series of box-whisker plots, see figure 4.3. One can see on the y-axis that the difference in magnitude is reduced in all cases. The

square-root transform does "pull back the outliers" slightly, but not a good as the logarithmic and rank transformation. In addition we observe that the logarithmic-scaled table does not show the typical distribution of the peak intensities as in the original and the square-root plot. Because it is more similar to the rank transformation, we fear that it may result in information loss as well.

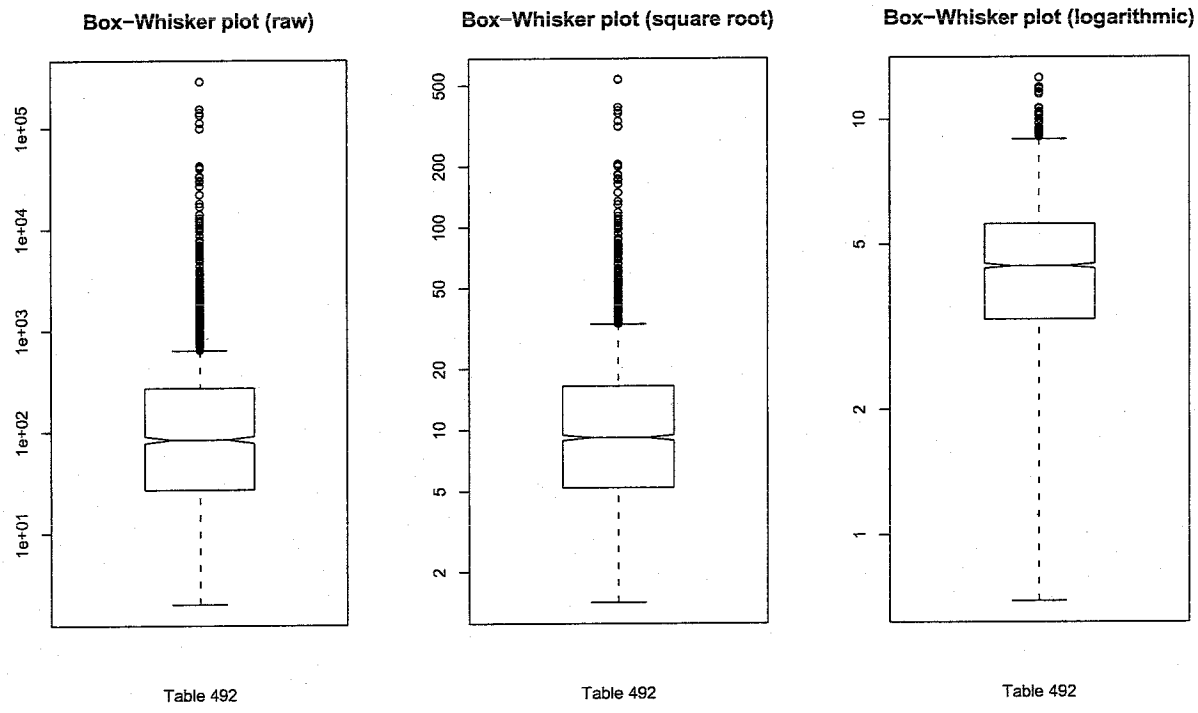

Figure 4.3: Box-whisker plots of a raw, square-root (sqrt) scaled, logarithmic-scaled and a rank-scaled peak-table. In all plots, the whiskers extend 1.5 times the interquartile range from the box by definition. The tables are taken from fruit 4D5. Note that the y-axis is logarithmic in the plots.

4.3.2 Linear scaling

It is common in the field of mass spectrometry to scale the intensities of the spectrograms or peak-tables linearly to the range $[0 \ldots 100]$. The highest peak, called the *base peak*, is set to an intensity of 100 and the others are scaled accordingly. This transformation is performed, because the absolute value of a peak is not important. It depends on the sampling, the ionization and the detector. It changes from one spectrometer experiment to the next, even if the sample is the same. But the relation among peaks in a table does not change; if a peak is half the height of the base peak in one run, the same relation holds in a replicate run. Therefore one looks at the peaks relative to each other and not at their absolute values.

Additionally, the linear scaling makes it easier to compare tables. For instance when one wants to test the reproducibility of the experiment. In this project comparison of tables also occurs between different samples (fruits), as we are looking for patterns in the overall data set. Any classification one wants to make with this data set forces you to compare tables of, for example, different lines. The difficulties of comparing such different tables are reduced by the linear scaling. An example is given in table 4.3. Two samples α and β both have two peaks, of which one is twice as high as the other. It is quite possible that the absolute values in the two samples differ

by a factor two or three, which makes it harder to compare them. Especially when you realize that their m/z ratios may differ slightly too. The ambiguity if these peaks are the same or not is reduced by the linear scaling.

Conclusion

We summarize the results of the non-linear and linear data transformations. First a non-linear scaling is performed on the peak intensities. In this report we work with the square-root transformation.

The motivation is that we re-evaluated the weight of each of the three motivations. At first "pulling back the outliers" was thought to be important. We realized the loss of information by applying the logarithmic or rank scaling is more important. The square-root transformation retains the shape of the original data set better than the other transformations. We assume that the typical distribution of peak intensities harbors information that can be exploited in later stages. Second, the peaks are linearly scaled to the range $[0 \ldots 100]$.

The final result is depicted in a traditional peak-table plot, see figure 4.4. Except for the annotation on the y-axis, the square-root plot in figure 4.3 shows the intensity distribution. As a last remark we note that the order in which these two scalings are applied, does not matter.

We have used the following methods (appendix E): `scale.linear`, `scale.sqrt`, `scale.log`, `scale.rank` and some descriptive statistics functions.

4.4 What is noise?

As with all experiments, our data contains noise. Originally caused by the mass spectrometer, the noise in the peak-tables is mainly an artifact of the peak detection software. The software[1], Micromass MassLynx [3], not only detects *real* peaks, but also *noise* peaks.

Micromass gives an idea of the amount of noise. It is stated on their website that the number of noise peaks is often in the order of ten times the number of real peaks. In the tomato data set, this means with an average of 3000 peaks per table, about 300 of them are real peaks. They suggest to select the 50 to 150 peaks with the highest intensity. A optimal cut-off intensity that separates noise from real peaks has to be established by *trial-and-error*.

The last statement is not satisfactory. We investigate whether there are better methods than trail-and-error. Taking the scaled peak-tables, we define the concept of noise peaks better. The accuracy of the m/z ratio of peaks varies along the m/z axis and it appears that noise is dependent on that accuracy (see also section 2.3). We present two different approaches for the calculation of this accuracy curve. How the elimination of noise proceeds, given an accuracy curve, is described in section 4.6.

As written before, each fruit consists of three replicate peak-tables. We use the replicates in the noise reduction process. Hence, the final step is to combine the replicates into one peak-table.

[1] Because it is a commercial software package, the detection algorithm remains unknown to us.

sample α		sample β	
m/z ratio	intensity	m/z ratio	intensity
⋮	⋮	⋮	⋮
110.01	0.5e+02	110.02	0.9+02
130.05	10e+02	130.03	18+02
⋮	⋮	⋮	⋮

Table 4.3: Example of comparison problems. In practice there can be low-intensity peaks (noise) closely around these peaks, making the problem harder.

4.4.1 Cut-off intensity

According to Micromass, the low intensity peaks are suspected to contain the noise. In section 2.3 we also mentioned this: one should not assume that every peak is a compound, especially peaks with very low intensity are questionable.

Hence we assume noise is restricted to the low-intensity part of a peak-table. We have plotted cumulative frequency graphs of the peak intensity for several tables. A typical one is shown in the left plot of figure 4.5. In the graph, the 95 percentage point is marked by a circle. In statistics, it is common to regard the data points with an intensity below the mark as noise and the upper five percent as signal (Z.R. Yang, personal communication). We name the 95 percentage point the *cut-off intensity* or p_c.

We examined the cut-off intensities for all tables using the method `ecdf.tomatoes` (appendix E). In the box-whisker plot of figure 4.5 we show the distributions per tomato lineage. One can observe that in the whole data set peaks with an intensity below $p_c = 6$ are probably noise. A naive way of removing noise would be to simply remove all peaks below p_c, which could be done with a simple call to `pass.high` (appendix E).

4.4.2 Defining noise better

Fortunately, we have not yet exhausted all the information that can help us to design a more sophisticated algorithm. A hint is provided by the fact that we have three replicate runs. Biochemicals that are present in the sample, should occur in all three runs[2], and at the same spot on the m/z ratio axis.

[2]Theoretically, there is the possibility of genuine peaks not occurring in all three replicates. When you realize that there is competition for ionization (see section 2.3), a biochemical that occurs only in very low concentrations, may be out-competed by other compounds in one or two of the runs. We do not have any way to distinguish this case from noise.

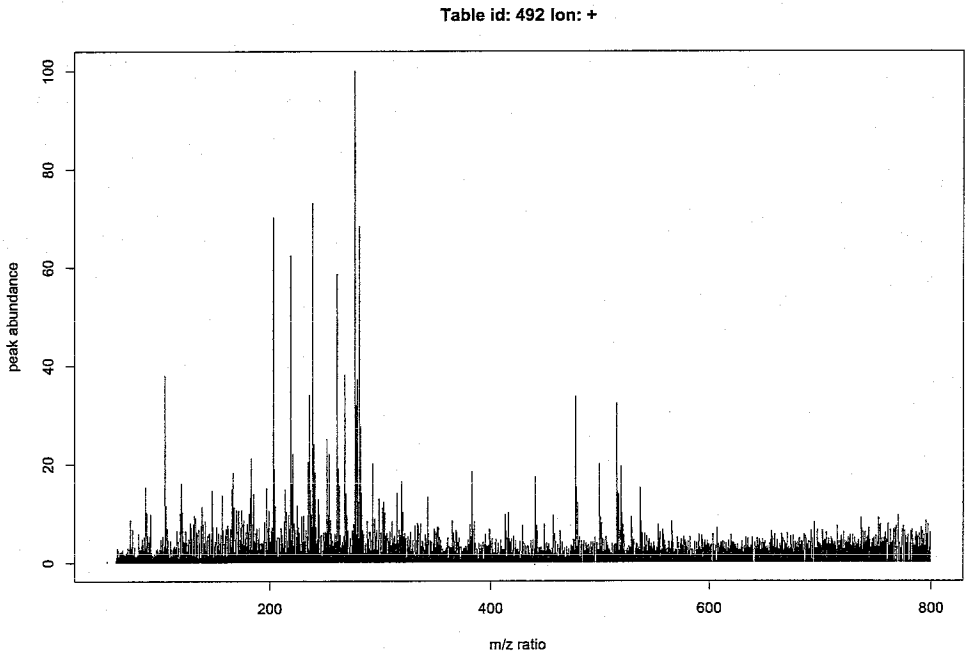

Figure 4.4: A traditional peak-table plot. If one compares the plot to the traditional plot on page 34, one can see the influence of the square-root and linear transformation.

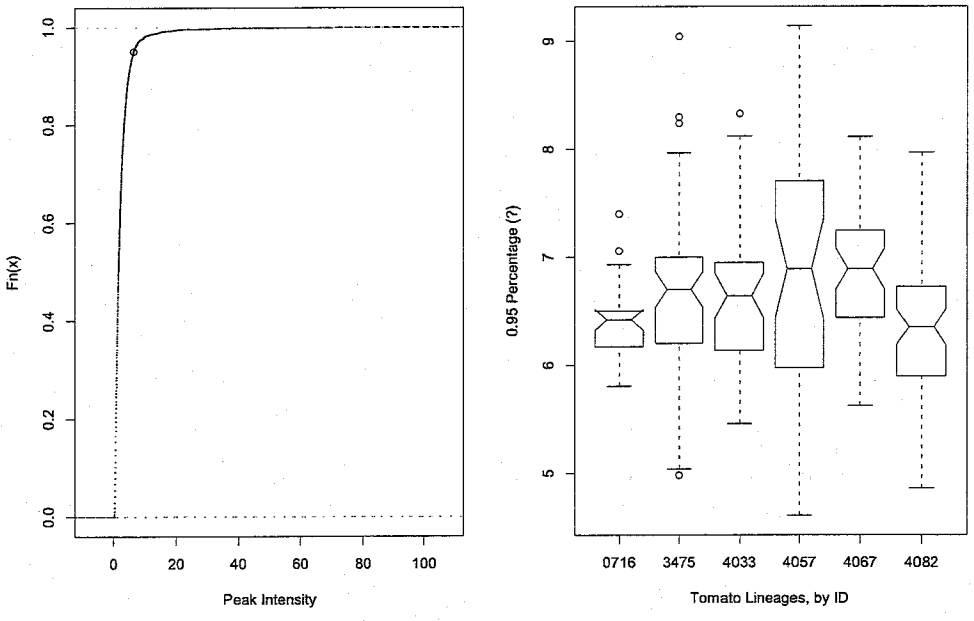

Figure 4.5: The left graph shows the empirical cumulative frequency of the scaled peak intensities of fruit 4D5. The right graph shows the distribution of 95 percentage points, aggregated by lineage. The whiskers extend 1.5 times the interquartile range. According to the documentation of the R function `boxplot`, if the notches of two plots do not overlap then the medians are significantly different at the 5 percent level.

First attempt involving the replicates

When we combine the above with the properties that lower peaks have a less accurate m/z ratio and low peaks are not guaranteed to be compounds, a new definition of what is noise and what is real, is:

- *real peaks*: peaks that occur in all replicate runs at the same m/z ratio,
- *noise peaks*: not real, i.e. peaks that occur either not in all runs at the same m/z ratio or peaks that occur in all runs, but not at the same m/z ratio.

A varying accuracy margin

A problem is that, although mass spectrometry is a very precise way of measuring , there is an accuracy margin. Additionally, the characteristics of mass spectrometers include that the accuracy margin varies along the ratio axis. It is focused at $m/z = 150$, hence the margin is at its minimum there. The rest of the curve is unknown.

So we have to alter the definition of real peaks and noise, to accommodate for a variable accuracy margin.

- *real peaks*: peaks that occur in all replicate runs within, for their m/z ratio, an accuracy margin given by function f_α,
- *noise peaks*: not real, i.e. peaks that occur either not in all runs but within the margin f_α or peaks that occur in all runs, but not within the margin.

Say we have three peaks p, q, r, each from a replicate. Then they are regarded as a real peak if $g_\alpha(p, q, r) \leq f_\alpha$, where g_α is a function to compute the accuracy margin (error) of a set of peaks on basis of their m/z ratios. Noise, then, is every peak p' for which no peaks q' and r' in the other replicates exist such that $g_\alpha(p', q', r') \leq f_\alpha$.

With the above definition of real peaks and noise two (intuitive) approaches to the computation of an accuracy curve have been taken. The first is based on approximating the accuracy margin function f_α with a regression. The second is an incremental technique that uses the fact that if lower peaks have a less accurate m/z ratio, higher peaks must have a more accurate ratio value.

4.5 Boundary between noise and real peaks

4.5.1 Regressions

The initial work on estimating a curve for the accuracy f_α has been done by Sheffield University. Because the accuracy provided by the manufacturer was unsatisfactory, they established their own accuracy margin with a linear regression. We describe their work (S. Overy, personal communication) in combination with the methods that we have used to extend it.

Methods

The regressions are based on manually selected combinations of peaks that are *real* peaks. We have assumed the actual accuracy curve does not depend on the data items. So we have taken five fruits, without replacement, from the data set. They are listed in table 4.4.

Assuming the accuracy curve interpolates smoothly along the m/z axis, we manually search for combinations of peaks around the ratios $75 + 100n$, where $n = 0, 1, \ldots 7$. That spans the range $[50 \ldots 800)$ quite well.

The decision if some combination consists of real peaks, is slightly intuitive. The peaks are judged on their margins in the m/z ratio and on their intensities, both with respect to each other (similarity) and to the neighboring peaks (dissimilarity). This is different from the definitions of *real* and *noise* peaks we present in the previous section. These definitions only addressed the m/z

ratio. We have included the intensity here, as it is much easier for the human observer to judge if certain combinations are *real* peaks, if he is allowed to take a clue from the intensities. An example is given in table 4.5

Then the accuracy is calculated for each combination of peaks by taking the standard deviation of the m/z ratios. This calculation of the accuracy has been taken from Sheffield. They define the accuracy margin of a set of peaks, g_α (introduced in section 4.4), as $g_\alpha(\bar{x}) = \sigma(\bar{x})$, where $\bar{x}$ is a sequence of m/z ratios and σ the standard deviation function. In the noise reduction, $\bar{x}$ is a triple, as we have the replicates in three-fold.

The average of these m/z ratios determines where along the m/z axis the accuracy holds. So we have five accuracy margins around each of the eight ratios. The open dots in figure 4.6 are those combinations. These 40 points are the input of the linear regression and the second and third order polynomial regressions. The regressions are shown in figure 4.6 as well.

We manually shift the regression in the y-direction by adding 0.003. The regressions are a weighting of the manually determined accuracies. All these accuracies are from real peaks. Because we want to define a boundary between the accuracy of real peaks and noise, the boundary must be higher than most of the open circles in figure 4.6. We have attempted to shift the boundary even higher than depicted in figure 4.6, but that resulted in the inclusion of a large amount of noise peaks. So the value 0.003 has been empirically determined, with as an informal lower bound that we should have at least more than half of the regression accuracy points below the boundary. Only at the high end of the m/z ratio, the lower bound is violated.

Any combination of peaks below the boundary, i.e. with an accuracy margin *smaller* than f_α, is a *real* peak. The rest is noise. We compare the regressions to each other and to the linear regression of Sheffield in their ability to discriminate between *real* peaks and noise.

Sheffield's work

The starting point of Sheffield for estimating the accuracy curve f_α has been the accuracy deviation provided by the manufacturer of the spectrometer: $f_\alpha^1 = 0.003$. This accuracy margin appeared to be rather tight for comparing tables of replicate runs. Increasing the margin up to $f_\alpha^2 = 0.006$ has given better results in the sense that more combinations of peaks (real peaks) are found. According to S. Overy it performs well for the lower part of the tables ($m/z < 400$), but its ability to find peaks quickly decreases as the m/z ratio increases over 500 or higher.

Fruit ID	Lineage ID
3G4	4082
10D10	3475
F11b	0716
9A6	4033
F131	4067

Table 4.4: The five fruits that provide the data for the regressions.

m/z ratio	intensity	intensity	intensity
222.9265	-	0.8284	-
223.0240	-	-	2.6090
223.0901	-	*7.3280*	-
223.0937	-	-	*6.3150*
223.0943	*6.8230*	-	-
223.1362	2.8620	-	-
223.1432	-	-	2.9440
223.1712	1.7810	-	-

Table 4.5: A combined view of the replicate tables of fruit 4D5. The combination of peaks, forming a *real* peak is in italics.

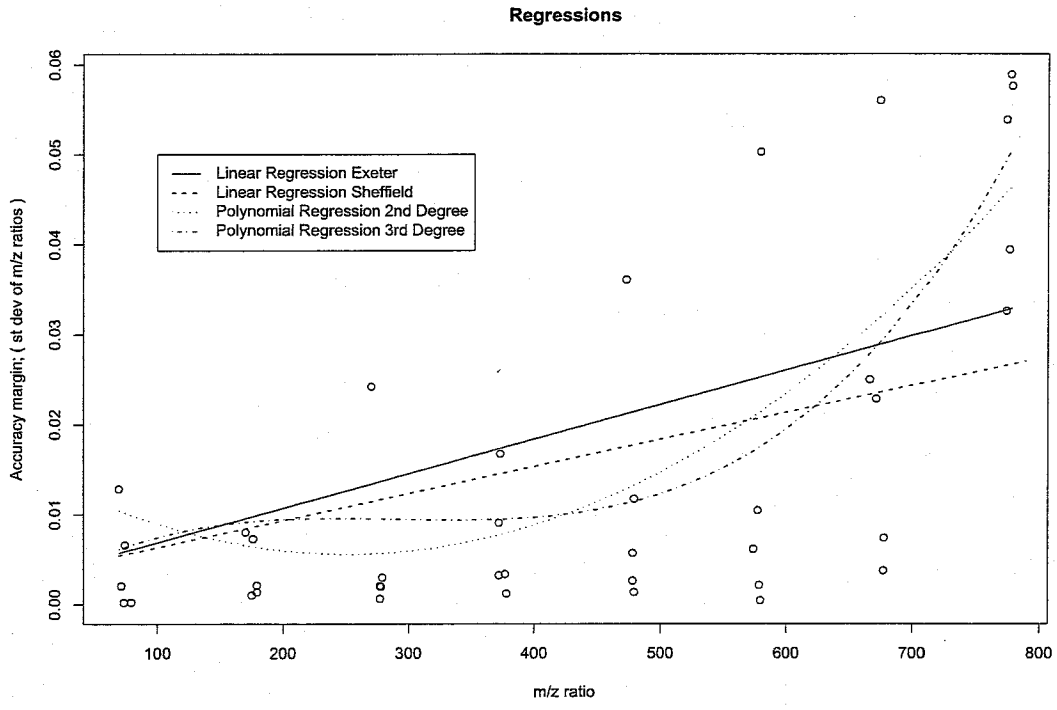

Figure 4.6: The input data (open dots) and the resulting, shifted regressions. Each data point has as m/z value the mean of three peaks and the y-value is determined by the accuracy of the peaks (standard deviation of the m/z ratios).

Therefore Sheffield University (empirically) established a more suitable margin. A linear regression of accuracy margins, that have been manually identified along the m/z ratio axis, has been calculated: $f_\alpha^3(x) = 3.000 * 10^{-5}x + 3.300 * 10^{-3}$, where x is the m/z ratio.

The linear curve performs much better for high ratios, but it has disadvantages too. Sheffield reports it is a little too strict in the lower end of the range ($50 \leq m/z < 100$) and for some real peaks too strict at the high end ($700 \leq m/z < 800$).

We have attempted to tackle the disadvantages of the linear curve by extending the work of Sheffield. Thus we have performed regressions including higher order terms.

Given the regressions, we have taken the scaled fruit 4D5. We have searched the fruit for real peaks with the different regressions (see also section 4.6). Two-dimensional histogram tables are used to visualize the result.

The regressions are available as functions starting with `criterium` (appendix E). We have used `criterium.exeter` throughout the project.

Results

In appendix F we show the results. A histogram of the original tables is found in table F.1, the regression results in table F.2 through F.5. We observe that all models perform equally well for the 10% highest peaks. Peaks with intensities inbetween zero and one are almost all left out. The differences occur in the intensity range $[1 \ldots 10)$, exactly the area around the cut-off intensity p_c. One interesting observation we have made, is that all regression models retrieve at least twice as many peaks as Micromass suggested there are.

Conclusion and discussion

It is not possible to draw strong conclusions from the results. The main theme is that there is a fuzzy area around the cut-off intensity, where some peaks are real and some are noise. It seems that the best regression would follow the maximal accuracy margin in figure 4.6, such that it includes even the peaks with high accuracy margins. Yet, a lot of noise peaks are included if we do so. The incorporation of peak intensities or the mass spectrometer's sensitivity may produce better results in the future.

We have examined several tables by plotting them. Some peak-tables looked unusual: they were missing several of the highest peaks. After searching for the cause, we found out that some of the highest peaks in one of the replicates had shifted m/z ratios in relation to the other replicates. This implies that the accuracy error for these peaks is abnormally high. Hence, they are classified as being noise[3].

We cannot identify one of the current regressions as the best. If we need to use one of them during the analysis, we assume the impact of including a small amount of noise is less harmful than excluding information (real peaks). Hence we choose the simplest one: a linear regression. Because there is no qualitative difference between the two linear regressions, we decide to use the Exeter regression. The motivation is that we know exactly how it has been constructed. The result of such a noise reduction is depicted in a box-whisker plot, figure 4.7.

Sample extraction techniques, in combination with the characteristics of the mass spectrometer and the peak detection software, influence the accuracy curve. This implies a disadvantage for all the regressions: they need to be computed and crafted manually for each data set. In other words, regressions are a rather ad-hoc technique. Data is produced at a fast rate, so the need for a more algorithmic approach becomes obvious.

[3]The decision was made to remove the fruits with such tables from the data set. There are only a few: C11c, C11e, 5A6, 9A1, 11C7 and 2D10. If we keep them, they act as outliers in the true sense of the word. We have assessed classifications with these fruits (unpublished results). It showed they do not influence the techniques qualitatively, they only show up as outliers.

4.5.2 Data-driven approach

As we explain at the end of the regression analysis, we want to be able to give the data to an algorithm without having to bother about an accuracy curve. In other words the objective is to design an the algorithm that figures out an accuracy curve itself. When there is no human intervention, the phase of data preprocessing may be fully automated.

Methods

The problem is how to determine accuracy curve automatically and use it to remove the noise peaks. Our idea is not to approximate the curve with a regression or other function, but to use two properties of peak-tables. The first is that noise peaks are low-intensity peaks. The second is that for low peaks the m/z ratio is not very accurate. Therefore we expect it to be more accurate for higher peaks, see section 2.3.

Assume we have three replicate peak-tables. We start with taking the highest peaks (*base* peaks). That is equivalent to taking the peak with the maximum intensity of the table. We can calculate their accuracy as in the regression analysis: take the standard deviation of the m/z ratios. That accuracy is used as the accuracy curve, i.e. $f_\alpha := g_\alpha(\bar{x})$, where $\bar{x}$ is the set of base peaks. We may use a search algorithm of section 4.6 to find all (real) peaks that have an accuracy at most f_α. It is guaranteed that the base peaks are found.

The idea is to iterate that procedure. After the search, we remove the found peaks and take the base peaks of the remaining peaks in the tables. The accuracy error has to be greater than the previous, otherwise these base peaks would have been found already. If the accuracy margin is not *very high*, it is probably a real peak. On the other hand, noise is characterized by not having any structure in its m/z ratio or intensity. Hence if noise peaks are taken as the base peaks, the accuracy error will be very high. Namely, there is no relation between these base peaks, they originate from anywhere in the table. So we can stop the iterations, if the accuracy margin becomes *very high*.

The important question is: what is *very high*? Given the number of significant digits of the m/z ratio and the mass deviation according to the manufacturer (± 0.003), we estimate the value 0.1 should already be *very high*.

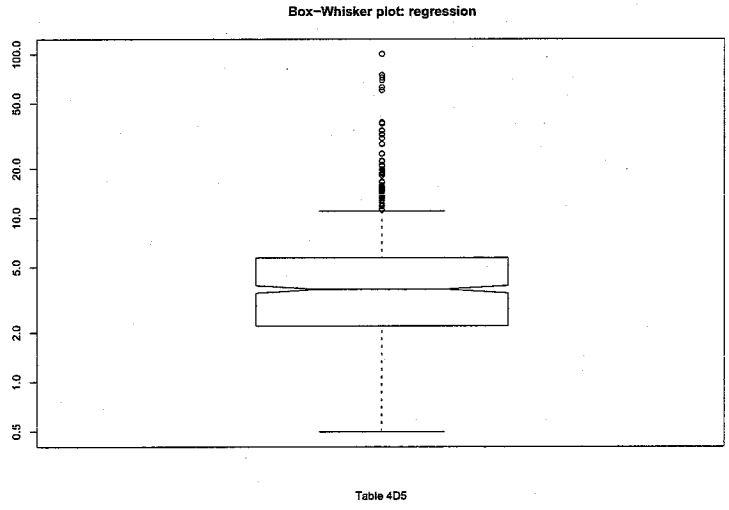

Figure 4.7: A box-whisker plot of the noise-reduced fruit 4D5. The noise has been reduced with the 'Exeter' linear regression. One observes an upward shift of the box. The whiskers extend 1.5 times the interquartile range from the box by definition. Note that the y-axis is logarithmic.

In a pseudo-algorithm the above looks as follows. Consider a set of peak-tables T and a maximal allowed margin $g_{max} = 0.1$.

1. take the base peaks of T. Let us call that set B,
2. calculate the accuracy margin $f_\alpha := \sigma(B)$,
3. if $f_\alpha \leq g_{max}$, continue with the next step, else stop,
4. search the tables in T for common peaks with accuracy margin f_α. The result is a set of combinations of peaks P,
5. remove all peaks in P from the tables in T,
6. go to step 1.

Besides being completely automatic, the algorithm never includes noise, if g_{max} is set to a suitable value. One may argue, we can set f_α to g_{max} immediately and skip the iterations. But, g_{max} is not a tight upper bound on the accuracy margin of real peaks. It is merely a value which is crossed, when the base peaks switch from real to noise. The algorithm relies on the fact that noise peaks have random intensities. When noise peaks in the replicates are accidentally close to each other, it is not likely that they all are the base peaks of their table. It cannot be proved that it will never happen that noisy base peaks have an accuracy error less than g_{max}.

Finally, we remark that the method also relies on the fact that tables are already similar in their m/z ratio and intensity: they are replicates. In the implementation of the algorithm, the real peaks that are found, are saved. We left that out of the pseudo-algorithm, such that we focus on the algorithm itself. Also, we have implemented a generalized version that can handle any number of replicates. A detailed description with some other implementation issues, like maintaining the invariants *uniqueness* and relatively ordered, is given in the help files of the software. The function is `search.common.incremental` (appendix E).

We have taken fruit 4D5 to test the viability of this technique. We have searched the fruit for real peaks.

Results

In appendix F table F.6 is the two-dimensional histogram of the incremental approach to the accuracy curve. We observe that the incremental technique finds nearly all high-intensity peaks, many of the mid-intensity peaks $[1 \ldots 10)$ and none in the low intensity range. As the m/z ratio increases, less peaks are found. The total number of peaks is almost six percent of the original number of peaks per table.

In figure 4.8 we observe that the change of accuracy margin from real peaks to noise is very abrupt. In addition, the number of peaks that are found is decreasing.

Conclusion and discussion

The incremental technique finds an even smaller amount of peaks as Micromass expects there to be, i.e. only one real peak per ± 18 noise instead of ten noise. So the incremental technique finds one fourth of the number of peaks that the regressions find. The differences are located mostly in the intensity range $[1 \ldots 10)$, which is around p_c.

The main cause for these results is that the algorithm is sensitive to anomalies in the peak-tables. If the variance in intensities is large among the replicates, the base peak can start to differ (too) soon. Also, we observe that in some samples one of the base peaks is shifted relative to the other two. During the initialization that causes a large margin to be generated and used, resulting in the inclusion of noise.

Our final conclusion is that the idea seems elegant, but the current algorithm is too sensitive to small "errors" in the tables. We think it is worth enhancing the algorithm to overcome the above

discussed problems, as it is an approach that can be fully automated. Due to the above mentioned problems, we do not use the incremental technique in our analysis. For completeness, we do show the result of noise reduction in fruit 4D5 with the incremental technique in a box-whisker plot, see figure 4.9.

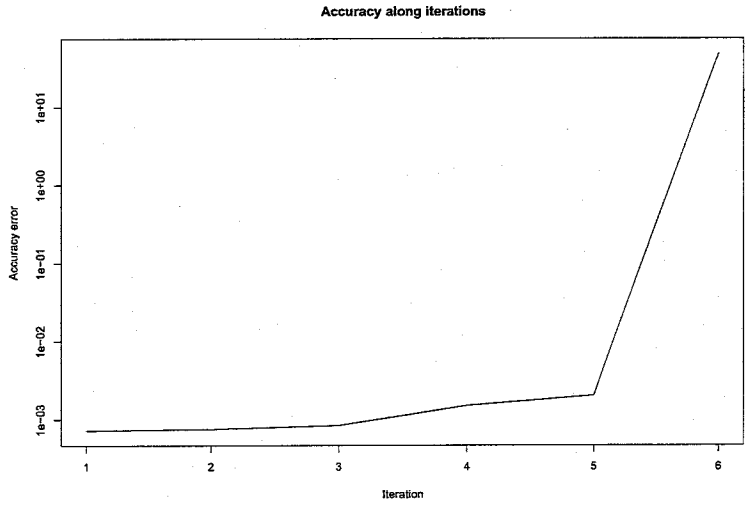

Figure 4.8: The plot shows the accuracies of each iteration and the accuracy of the first noisy base peaks.

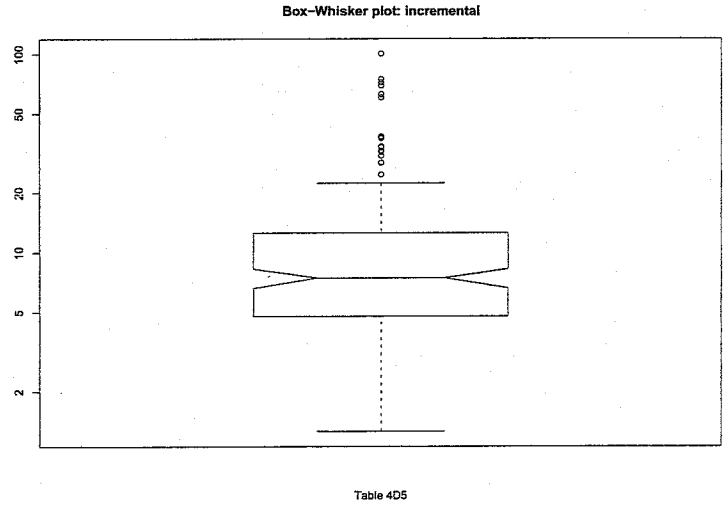

Figure 4.9: A box-whisker plot of the noise-reduced fruit 4D5. The noise has been reduced with the data-driven approach. The whiskers extend 1.5 times the interquartile range from the box by definition. Note that the y-axis is logarithmic.

4.6 Searching for peaks

We have introduced two techniques for establishing an accuracy curve. Assuming we are using one of them, how do we discriminate between noise and real peaks? Below we present two algorithms that separate noise from information. They search the tables for combinations of peaks, such that these combinations give us the real peaks. In a more general setting, the combinations of peaks are the peaks that the tables have in common.

The first algorithm is a so-called *exact* algorithm. It considers all possible peak combinations within accuracy margins. It is designed to perform well even with low-quality peak-tables (noisy, less accurate m/z values).

The second algorithm is an *approximation* method. In peak-tables with noise it does not consider all combinations of peaks. If the tables do not contain noise and have the property that accuracy $\ll$ sensitivity, it practically considers as many peak combinations as the exact version. Because the approximation method simplifies the required computations, it is the faster of the two.

Below we start with two additional complications that arise in searching for peaks. It is followed by a description of both algorithms. We introduce the concepts that are associated with the algorithms and explain their working. Finally, we describe the precise difference between the two with an example and the implications for their performance.

4.6.1 Complications

The current definitions of real peaks and noise (see section 4.4.2) are not complete. As an example, one can imagine that combinations of peaks may overlap. It is likely to happen with noise peaks when these are close to real peaks. In such a case, the current definitions of real peaks and noise categorize all combinations as real peaks. But we know, only one of them can be the *true* real peak.

Because we have to solve this issue, we introduce two invariants that must hold during the search for common peaks. Additionally, we propose a strategy for dealing with the overlapping combinations.

Invariants

The definition of real peaks and noise needs to be complemented by two invariants. One expresses that all combinations of peaks are unique, the other that the combinations may not violate the relative ordering of the peaks along the m/z axis. For both we show a violation in figure 4.10 and figure 4.11.

- *uniqueness*: each combination of peaks has no peak in common in the same table with any other combination of peaks,
- *relatively ordered*: if p, q, r and t, u, v are combinations of peaks and for the m/z ratio holds $p < t$, then for the m/z ratios of the other peaks holds $q < u$ and $r < v$.

Dealing with overlaps

The current definition of noise (section 4.4.2) does not refer to the peak intensities. Hence if there is some ambiguity in the form of overlapping combinations of peaks, we still have some information that we can use.

We introduce a strategy for dealing with overlaps. The problem is illustrated with an example. Say we have two combinations p, q, r and p, q, s. Both are real peaks. Then only one of them, for instance r, is a true *real* peak and s is probably a noise peak that accidentally fits in as well.

In such a case, we assume a combination of peaks is more likely to be *real* peak, if their peak intensities differ only little. In operational terms, if the intensities of p, q, r have a lower standard

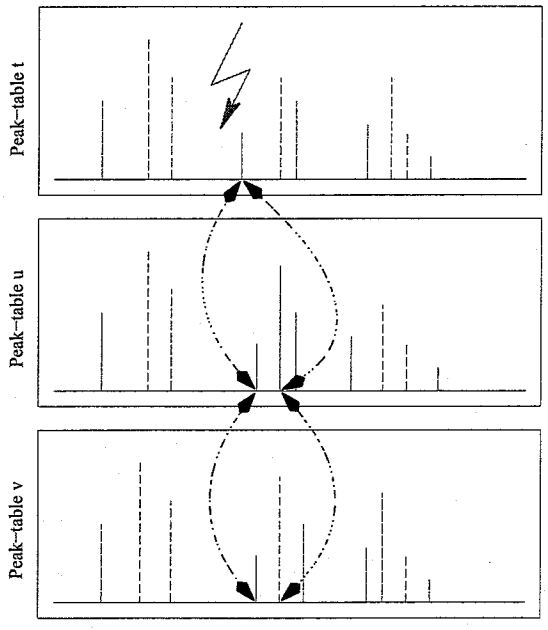

Figure 4.10: An example of the violation of the invariant *uniqueness*.

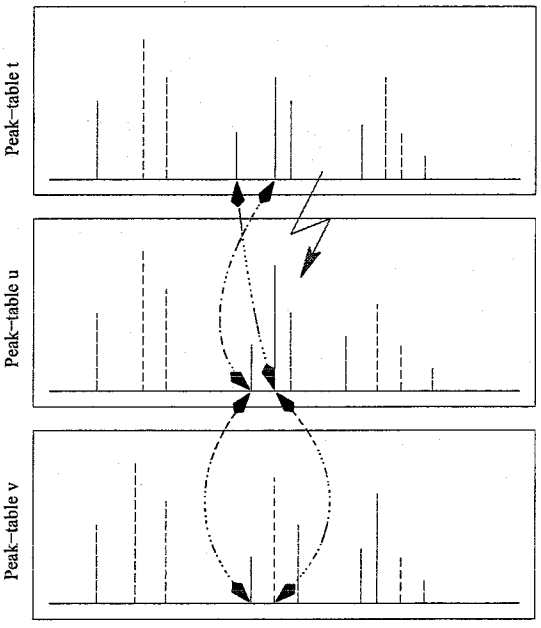

Figure 4.11: An example of the violation of the invariant *relatively ordered*.

deviation than the intensities of p, q, s, we select p, q, r as the *real* peak. In this manner, we can select the real peaks of fruits, while we keep the above introduced invariants.

We generalize to any number of overlapping combinations. If we have a set of overlapping combinations O, we adopt a greedy procedure. We start by selecting the combination with the lowest intensity difference from O. Let us remove that combination from O and store it in set R. As we have to maintain the invariants *uniqueness* and *relatively ordered*, we eliminate all combinations from O that would violate one of the invariants in relation to any of the combinations in R. This is repeated until $O = \emptyset$.

This greedy algorithm is available as `select.peak.greedy` (appendix E).

4.6.2 An *exact* algorithm

Concepts

The input consists of a set of peak-tables T, an accuracy margin f_α and a function g_α that computes the margin of a set of peaks.

- *window*: a set of peaks, where each table in T has contributed at least one peak. The window is defined by its lower and upper bound. An example is given in figure 4.12. The window consists of the nine solid peaks. A typical variable name for a window is w,
- *accuracy*: a window w has an accuracy. It is determined by taking all peaks in w and computing $g_\alpha(w)$. Hence it is the accuracy of the peaks in w,
- *increment*: the increment of the lower bound of a window w may be phrased as excluding the peak with the lowest m/z ratio from w. The increment of the upper bound of w is defined as including the peak with the next occurring m/z ratio (across all tables in T),
- *movement*: a window w is moved along the m/z axis. The movement consists of either incrementing the lower or upper bound (or both),
- *successor*: when a window w can increment its upper bound, it has a successor or next window. The successor is the window that results from the increment of the upper bound of w,
- *combination*: a combination of peaks is a set of peaks. There is one peak from each table in T.

The increments may violate the "each table in T has contributed at least one peak" part of the window definition. We solve the violation by performing a bounded linear search (BLS) along the m/z axis to find the next *valid* window. The BLS consists of incrementing the upper bound until each table in T *does* contribute at least one peak, while keeping in mind that the end of the peak-table can be reached before a valid window is found. Therefore, the algorithm may not reference the windows, for instance to compute the accuracy, inbetween an increment and the bounded linear search.

Description

The goal of the algorithm is to find combinations, such that the peaks of a combination can be regarded as being the *same* peaks, i.e. *real* peaks.

The exact method initializes the window at the beginning of the m/z axis, with the first peak of each table in T. Then it moves the window along that axis. Each current window w_c is saved if its accuracy is maximal.

The window accuracy is maximal if it is as large as possible, but not larger than f_α. This way the number of windows that is checked for combinations is minimal and so is their possible overlap.

The movement of the window consists of increments of the lower or upper bounds. It depends on the accuracy of w_c and the successor window w_n.

After iterating through the tables, we have a set of windows for further processing. These windows contain peaks, from which we have to filter out the most likely combinations. Before we describe the filtering, we give the algorithm in pseudo-code. The input consists of a set of peak-tables T and an accuracy f_α.

1. initialize window w_c with the first peak from each table in T,
2. initialize the set of windows W that need further filtering. $W := \emptyset$,
3. while w_c is not at the end of each table in T do
   (a) the successor window w_n is determined,
   (b) if $g_\alpha(w_c) \leq f_\alpha$ and $g_\alpha(w_n) \leq f_\alpha$ then the accuracy of w_c is too small, so its upper bound is incremented,
   (c) if $g_\alpha(w_c) > f_\alpha$ the accuracy of w_c is too large, hence its lower bound is incremented,
   (d) if $g_\alpha(w_c) \leq f_\alpha$ and $g_\alpha(w_n) > f_\alpha$ the accuracy of w_c is maximal. w_c is saved for further filtering, $W := W \cup \{w_c\}$. Both bounds of w_c are incremented,
   (e) bounded linear search on w_c to the next valid window,
4. the result is a set of windows W that need filtering.

The exact algorithm groups the overlapping windows in W and considers all possible combinations of peaks in these groups. It chooses the best by filtering the combinations through the greedy strategy `select.peak.greedy`.

The current implementation of the *exact* algorithm returns a set of indexes, referencing the common peaks of the tables in T. It is a matrix, where each row corresponds to a table from T and each column to a combination of peaks of these tables. The columns are strictly increasing in each element, due to the invariant *relatively ordered*. The algorithm is called `extract.all` (appendix E).

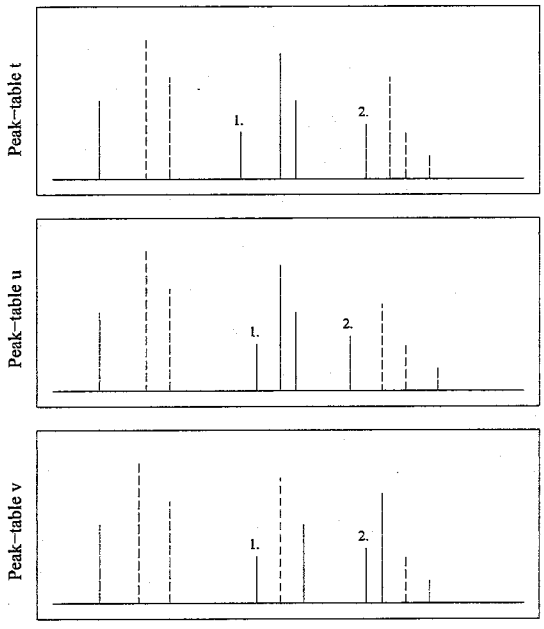

Figure 4.12: Three replicate peak-tables. The window consists of the solid peaks. The peaks with a '1' are the lower bound. The peaks with a '2' form the (exclusive) upper bound.

4.6.3 An *approximation* algorithm

Concepts

The input is a set of peak-tables T, an accuracy margin f_α and a function g_α that computes the margin of a set of peaks. The concepts of the previous section still hold, if they are not re-defined below. We introduce some additional terms.

- *front*: a set of peaks of size $|T|$. Each table in T contributes exactly one peak. A typical variable name for a front is f,
- *increment*: a front f is incremented by taking the table t with the first next occurring peak p (along the m/z axis). The current peak of t in f is removed and the new peak p is added,
- *movement*: the movement of a front f is incrementing f.

Description

The algorithm has the same goal as the *exact* version: to find combinations of peaks such that the combinations form *real* peaks.

The approximation technique starts at the beginning of the tables in T. A front f is initialized by taking the first peak from each table in T. f is moved (unconditionally) along the m/z axis, as described in the concepts above.

For each front f the accuracy is computed, $g_\alpha(f)$. This is analogous to computing the accuracy of a window in the exact version. If the accuracy is within the allowed margin f_α, the front is saved.

The intermediate result is a set of fronts F, with possible overlaps between the fronts. Hence, although fronts are in principle equivalent to combinations, violation of the *uniqueness* invariant forces us to filter F. We apply the greedy strategy `select.peak.greedy`, as we do in the *exact* procedure.

The above explanation in pseudo-code looks as follows.

1. initialize front f with each first peak from the tables in T,
2. initialize the set of fronts that need further filtering, $F := \emptyset$,
3. while f is not at the end of all tables in T do
   (a) if $g_\alpha(f) \leq f_\alpha$, then add f to F. $F := F \cup \{f\}$,
   (b) increment f,
4. return the (intermediate) result F for filtering.

After the greedy procedure, the final result is a set of combinations of peaks, which are *real* peaks. Currently, the result is a matrix of indexes, in which the rows represent the tables and the columns the common peaks (strictly increasing), equivalent to the format of the exact algorithm. The approximation search is named `extract.approx`.

4.6.4 Differences

Skipping combinations

We stated that the difference between the *exact* and *approximation* algorithm is that the latter does not consider all combinations of peaks.

It is best explained with an example. Figure 4.13 shows three replicate tables, together with a window and a front.

The window contains combination p', q, r. The current front is given by p, q, r. If we increment the front, the next is p, q', r. When we increment the front again, the result is p', q', r. The approximation clearly skips a possible combination, namely p', q, r.

Although in the figure it may not look so bad, we imagine it can be problematic in low-quality peak-tables. In the tomato data, this does not occur. The two algorithms give *exactly* the same results. The histograms in appendix F could have been produced with either technique.

But let us suppose it does happen in worse quality tables. In that case, missing a combination is still not so bad. It is not unreasonable to assume that even within the accuracy margin, more accurate combinations should be preferred over less accurate ones. Then for each missed combination x, the approximation method considers an overlapping combination y with an even lower accuracy margin.

Performance

The different style of searching results in a different performance. We estimate, based on the number of assignments and the fact that a window is a more complex data structure than a front, that the approximation algorithm is at least twice as fast as the exact algorithm for the same input.

We do not compare the algorithms by execution time, as currently the approximation method is implemented in R and C, whereas the exact algorithm is implemented only in R. Obviously, the current implementation of the *approximation* algorithm is the fastest. Hence we have used that peak searching technique in our analysis.

4.7 Creating one table

After noise reduction, we have combinations of peaks. We do not wish to keep three replicates, which essentially contain the same information. Instead we want one peak-table per fruit.

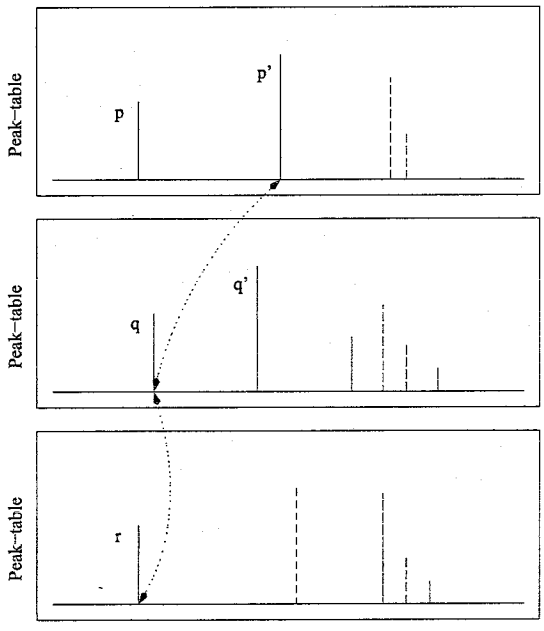

Figure 4.13: Three peak-tables with a window and a front. The window is displayed as the solid peaks. The front is given by p, q, r. The arrows point at the skipped combination.

The combining of these peak-tables consists of deciding how we create one peak from a combination of peaks. We have considered two options:

- *mean*: we may take the mean of the m/z ratios and intensities,
- *median*: taking the median of the m/z ratios and intensities may be more robust if the m/z values or intensities vary a lot.

Both options have been implemented. The corresponding functions are the standard R functions `mean` and `median`. One of these functions is passed to the method `combine.peaktable`.

We have chosen to take the *mean*. The m/z values do not vary much at all (otherwise it would be noise), nor do the intensities.

4.8 Summary

The view on the tomato data is enhanced with a novel visualization method. Using the novel view and traditional plots, we found and subsequently removed several corrupt peak-tables.

The preprocessing stage is constituted of two steps. First we scale the peak intensity. For numerical stability and to remove a variable standard deviation, we apply a square-root transformation. Then we scale the intensities linearly to the interval $[0 \ldots 100]$. That is a standard procedure in mass spectrometry.

The second step is noise reduction. Usually, in mass-spectrometry experiments the samples are not replicated. We have such replicates. Because noise and real peaks are mostly characterized by their m/z ratio accuracy, we have designed two techniques for approximating the boundary of accuracy between noise and real peaks. In this project we approximate the accuracy curve with a linear regression, `criterium.exeter`.

Given a boundary, the tables are searched for peaks that they have in common: *real* peaks. Unfortunately we have not been able to eliminate all noise from the tables. It is caused by the fact that the boundary between noise and real peaks is fuzzy.

We have chosen to use the approximation search for the analysis. Although the current focus is on searching on the level of fruits, i.e. we take the replicates and determine which peaks they have in common, the algorithms are generalized such that we can investigate the common peaks of plants, lineages or the whole data set.

Eventually, the common peaks may be combined into one table by taking the mean or median of the m/z ratios and the intensities. The main application is to combine the replicate tables into one, such that a fruit has one table with a reduced noise level. The broader picture is, for instance, that we may create tables containing the peaks of a plant. We combine the replicates by taking the mean.

The result of these preprocessing steps is shown for fruit 4D5 in figure 4.14. At the end of preprocessing we have 84 fruits (samples), each consisting of one peak-table. The tables' intensities are in the range $[0 \ldots 100]$, and their level of noise is reduced.

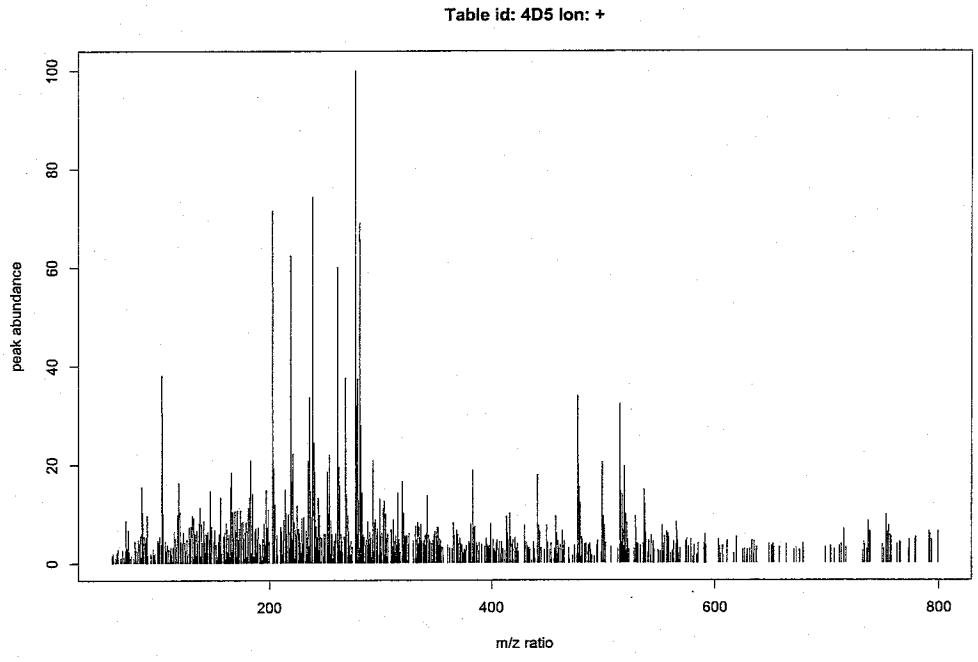

Figure 4.14: The traditional view of fruit 4D5. One can clearly observe that most low-intensity peaks are removed.

Chapter 5

Features

The third phase in pattern classification is *feature extraction.* The aim of feature extraction is to capture all essential information in a few variables, the features. So we define features as condensed representations of the original (preprocessed) data. Ideally, they still contain all relevant information. With *relevant*, we mean that we are able to discriminate between categories or that the features uniquely represent the original data. However, in practice there is some loss.

It depends on the classification task, what kind of information-loss you tolerate. An example is that if we intend to classify pigs being male or female, we may settle for a lot of information-loss. It suffices to keep as a feature the answer to the question "Has the pig got nipples?". With this feature we are still capable of discriminating between males and females, but we lose all other information, such as the size of a pig or if it has had any diseases during its lifetime.

First we discuss the philosophy behind our feature extraction stage. Then we restrict the project to the structure of our tomato data and we define the main criterion. It is followed by a search for feature extraction methods in other fields. We describe the problems, that we have run into during the search and propose solutions. We have applied most of the solutions on our data.

5.1 Objectives

Although the preprocessing stage has taken much time, the current stage has the primary focus of the project. The long-term aims of Sheffield University and the project objectives that we derived from these aims (section 2.1) hint at the use of features. The two goals are:

- rapidly screen sets of plants (against a yet-to-build library),
- develop a unique fingerprint for each species or variety.

Since spectrograms and peak-tables are large data items, the feature extraction is more or less compulsory. It is true that computers have enough resources nowadays to handle full spectrograms, but high-throughput metabolomics and metabolic profiling require faster ways of handling the data.

There are few previous attempts in metabolomics to handle sets of spectrograms. [51, 52] tackle the data by labelling the peaks, i.e. extracting the compounds by screening the spectrograms against spectral databases. The result is a list of identified compounds with their relative abundance in the data. We take one step back. We look at the tables and explore what information is in the tables themselves. We take the tables as one entity, not a collection of compounds. One may call this a 'weak' approach, we expect the least from the data and let it show us its structure. In this setting, our data analysis can be very fruitful for one of the side areas of metabolomics, *metabolic fingerprinting.*

Our aim for feature extraction is of exploratory nature. We do not rule out possible features, because they might not suit the above goals. Instead, we are looking for all kinds of features and the accompanying feature extraction techniques. This is simply because we do not know what features are suited for classifying peak-tables.

The main advantage of the above-described approach is the processing speed. We do not need to deconvolute the peak-tables, which is a computational intensive task. A nice side-effect is that we do not need a profound knowledge of mass spectrometry. On the other hand, the drawback is that, if we are able to discriminate some interesting categories in the data, it is more difficult to point out the metabolites that cause the dissimilarities.

5.2 Constraints

We have constrained our project explicitly on two topics. First of all the feature extraction is performed on the level of fruits. There are several reasons to do so. First of all, we have the most (84) data items at this level. On the level of plants, we only have 19 items. And on the lineage level only six.

The other reason is that the fruits consist of one peak-table. As the feature extraction of peak-tables seems an unexplored area (section 5.3), we start with the basic entity, one table. Taking features from collections of tables is a future challenge.

The second constraint concerns the m/z ratio and intensity. One of the areas we would like to explore, is which of the two vectors in a peak-table contains the most information.

Our goal on the feature extraction level is assessing the different techniques on their ability to expose the structure of the data. We immediately realize that the separation of the m/z and intensity vectors can have a negative effect on the unravelling of the structure. One can imagine that the two vectors together are the basic building block of the data structure.

We mention that the final array of extraction techniques depends on the available ones and our creative process in designing new ones. The results and conclusions on this matter are discussed in the classifications.

5.3 Other research fields

In metabolomics there have been few previous attempts in clustering or classifying metabolic data [48, 51, 52]. Most cases we have found in in the literature, apply supervised methods or concern mass spectrograms or NMR spectrograms. That is different from our unsupervised approach on peak-tables. Processing, analyzing and modelling such tables is a new area for metabolomics. So finding existing techniques for feature extraction is hard.

As reported before, one approach in metabolomics is labelling the peaks, i.e. extracting the compounds by searching spectral databases [51, 52]. Our motivation not to use this technique has been documented in section 5.1.

The next step we have taken, is to look for other scientific fields with similar data, either conceptually or mathematically. If we find such research areas, we may be able to take their analysis techniques and apply them to our data. This is a common tactic in bioinformatics. During the past ten years, HMMs have been taken from speech recognition, sequence comparison algorithms from character-string methods, SVMs from machine learning and so on. We have looked for fields with spectral data or time-series data that are analyzed on a large scale.

The search has proved to be difficult. We have examined possibilities in analytical chemistry and nuclear-magnetic-resonance research for spectral data. Although the data often looks similar, in all cases we find it has a so-called implicit sampling. It means that the data is sampled at regularly spaced points in time or space. Peak-tables have an explicit sampling: the peaks occur at irregular distances from each other. Therefore the techniques cannot be applied. In addition, when methods are used that especially suit our type of data, these fields use supervised methods[1]

[1] They may still be useful in the future.

[37, 38].

We have looked into fields with time-series data, like financial analysis and the analysis of cardiograms. In the financial area, when one talks about time-series data, the main objective is the fore-casting of stock markets. Although fore-casting can be easily rephrased in terms of classifying time-series in different categories, the common method that is used ("moving window") is hard to apply in our case. First of all, because of a conceptual difference. Where the moving window assumes a strong dependence between the data points, we have data where the points, peaks, are assumed to be (more or less) independent. Second, time-series are implicitly sampled, like the spectral data sets. In cardiogram analysis also most pattern recognition is based on implicit sampling, yet we have been able to find a technique that can be applied.

The literature research has not been fruitful. We have found a few methods that may be applied in the future, but directly applicable techniques are rare. Apparently, the unsupervised classification of peak-tables without labelling peaks seems an unexplored area.

5.4 Problems

In the first attempts to reveal some possibly hidden structure in the peak-tables, several problems became clear. They boil down to two properties of peak-tables that render direct application of feature extraction methods on the tables impossible.

- the length of the tables is variable; the number of peaks differs between tables. For example, fruit 4D5 has a length of 855, but within the same plant we observe that fruit 7D10 has only 702 peaks,

- the distances between m/z ratios of peaks are irregular. One can observe that, for instance, in the peak-table depicted in table 2.2 and figure 4.14.

As an extra problem, these two imply that the peaks of different tables are not aligned. Hence comparison of tables is complicated. We noticed that issue already in the preprocessing stage. From a different viewpoint, one can say that peak-tables are, by nature, explicitly sampled.

The implications depend on the feature extraction method, which one wants to apply. In total, we have three objectives before any traditional method, like PCA, can be applied.

- the length must be constant across the data set. Virtually every feature extraction technique expects an equal length of all items in the data set. One can only apply descriptive statistics methods, i.e. calculating the mean, variance or skew, if the number of peaks differs from table to table,

- the distances between the m/z ratios may be irregular for some feature extraction methods. Then a prerequisite is that the irregularities appear at the same m/z ratio. Such an *alignment* of the peaks may suffice for, by example, the 'raw' clustering of the tables on their peak intensities,

- other methods, like DSP techniques, need a constant, fixed distance both in between the peaks and across the data set.

In addition we mention that the objectives mostly deal with the m/z ratios of the tables, not the intensities. This means the m/z ratios are more likely to be changed, creating a possible information loss. Hence our conclusion (section 6.2.5) if information resides mostly in the m/z ratio or the intensity may be biased.

The problems are simple to formulate, so are the objectives. But the problems are fundamental characteristics of peak-tables, which makes it difficult to tackle them.

5.5 Solutions

We search for solutions in two complementary directions. First, if we accept the properties, we may view them as potential sources for features. An example is that the length of a table might indicate from which tomato lineage that table is. We do not use this particular idea, because we know the preprocessing algorithm includes noise. Then differences in length are not necessarily linked to differences in the metabolome of lineages. The result may also originate from artifacts of the preprocessing technique.

Second, we can try to modify the properties, such that traditional feature extraction techniques are applicable. We have to be careful with this approach. Because we do not have much knowledge of the data (yet), it is hard to estimate the effect of "molding" the data. Some information loss or the introduction of biases are the expected side-effects. The first is not too bad. Our classifications may fail, but it points us at the information-rich parts of the peak-tables. The introduction of biases is truly unwanted. We might impose a structure on the data that we expect or hope for, like clustering according to plants.

Below a list of solutions is presented. We have captured the working of these solutions in a figure, see figure 5.4 on page 65.

5.5.1 Inter-peak distance

This feature technique is inspired by cardiology research (dr. Z. R. Yang, personal communication). In ECG time series the time differences between certain events in the heart have appeared to be a good indicator for the type of heart deficiency. We apply the same idea in peak-tables. We consider peaks as events and the differences in m/z ratio between neighboring peaks as the features. Such distances may be characteristic for a plant or lineage.

However, we have a varying number of distances, because the lengths of tables differ. Our first solution has been to compute two statistical moments of the inter-peak distances: the mean and the standard deviation. An example is shown in figure 5.1.

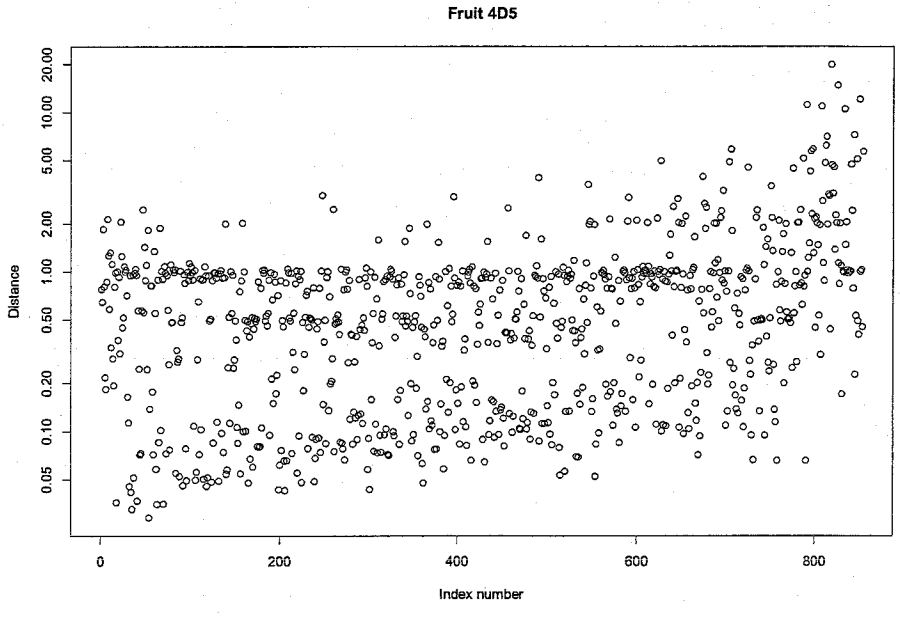

Figure 5.1: The inter-peak distances of fruit 4D5. The mean and standard deviation of the inter-peak distance are 0.8661 and 1.4109 respectively. The index number refers to order of occurrence of the inter-peak distances along the m/z axis.

The method is easy to implement and apply. It focuses solely on the m/z ratios. Hence it can be used to investigate the amount of information in the m/z component of a peak-table. However taking the moments of a dataset is a source of information loss. In the cardiology research, this technique is capable of discriminating between heart deficiencies, as long as they are 'not too similar'. In other words, the resolving power of the inter-peak distance technique is probably limited.

The method is available as `interpeak.distance` (appendix E).

5.5.2 Select-n-peaks

The first attempt to mold the tables, has been (a crude way of) fixing the length of the tables at a constant n. There are many possibilities to accomplish this, any predicate on the m/z ratio or intensity that selects the same number of peaks in each table, suffices.

Here we have a nice relation with the labelling approach [51, 52]. It selects a collection of compounds (peaks) that are found by screening against spectral databases. That approach has given good results: [52] has discriminated between different *A. thaliana* (wall cress) lineages.

We have restricted ourselves to predicates on the intensity. A simple example is to select the ten highest peaks in each table. A more advanced predicate is to select $m, m > n$ peaks, followed by discarding the n highest. These n peaks are the same across the data set. The examples show the possibility of zooming in on selections of peaks. Please note that the last example is already exploiting extra knowledge of the data. It is not a supervised technique, but it is 'guided' in the right direction.

The method is easy to apply, just as *inter-peak distance*. It corrects only one of the three problematic properties, namely the length is fixed. The success of taking predicates on the intensity depends on how much information is located in the order of occurrence of the peaks along the m/z axis and their height. In that respect it is related to the *inter-peak distance* method, but it evades the information loss that occurs in taking the moments of the data.

At the same time, it creates another problem. We silently assume that if we select n peaks from one table, we select the same peaks in another table, i.e. the peaks are aligned. that is truly the case, is questionable. There are solutions to this problem, see section 5.6.

The idea has been developed as one of the first attempts to extract features. The two proposed predicates are naive and produce only 'random' clusterings. The select-n-peaks method has not been implemented as such. We decided that it is useless to create a general method that captures all possibilities. In R it boils down to applying some function on the `tomatoes` object. This, in turn, is equivalent to using `lapply`, the function for applying any function on a list.

5.5.3 Binning

During the implementation of the novel visualization (section 4.1) we thought of using the same principle as a feature extraction method. In the visualization we divide the m/z range into a number of intervals and let the average intensity in the interval determine the color. The basic idea of *binning* is a generalization of that approach.

Suppose we have a peak-table t, a number of intervals or *bins* n and a function f that operates on a collection of peaks. The m/z axis of t is divided into n equally large bins and f is applied on the set of peaks in each bin. Three examples of f are taking the number of peaks, the average or maximum intensity.

In a pseudo-code algorithm it looks as follows: the input consists of a peak-table t, the number of bins n, a range on the m/z axis r and a function f that computes a value from a set of peaks.

1. divide the range r into n equal bins: $b_0, \ldots, b_{n-1}$ Each bin b_i, $(0 \leq i < n)$ is an interval $[l_{b_i} \ldots l_{b_{i+1}})$ on the m/z axis. l_{b_i} is the inclusive lower bound of the interval and $l_{b_{i+1}}$ the exclusive upper bound,

2. for each b_i, $(0 \leq i < n)$ do:

(a) select the peaks in t with $l_{b_i} \leq \text{m/z} < l_{b_{i+1}}$. In other words, the peaks with an m/z ratio within the bin-bounds are selected. Let us call the resulting set of peaks P,

(b) apply f on P: $f(P)$. An example of f is a function that computes the maximal occurring intensity in P,

(c) a new peak is created in the mid-point of the bin b_i, representing the whole bin: $p := (\frac{l_{b_{i+1}} - l_{b_i}}{2},\ f(P))$, where the tuple is interpreted as (m/z ratio, intensity),

(d) replace the peaks P with p in peak-table t

3. return a *binned* table t

An example of the binning technique is shown in figure 5.2. Because of its high number of bins, the plot of 4D5 looks very similar to the original fruit (figure 4.14).

There are some remarks to be made. First, the range r enables us to focus on parts of the whole peak-table. Thus creating a more versatile algorithm.

Second, the function f has to be able to handle the empty set as input. It is possible that in a bin no peaks occur: the postcondition $P = \emptyset$ can be true. In that case, we have to assign a value to that bin ourselves. The possibilities are numerous. It is best to let it depend on the objective one has in mind. The first idea is often to give the bin the value zero. If we want to apply a hierarchical cluster analysis with euclidean distances, zero is indeed a good value. But there are more options. For instance, if we are going to apply a ratio transform after *binning*, a value of 1 may be chosen because one may always divide by one. A more exotic idea is to assign empty bins a value that causes algorithms to perform a special action on those bins.

As a third remark, the algorithm focuses on intensities. The m/z ratio is not important, as it is "molded" into a regular sequence. Therefore, the *binning* technique complements the *inter-peak distance* method.

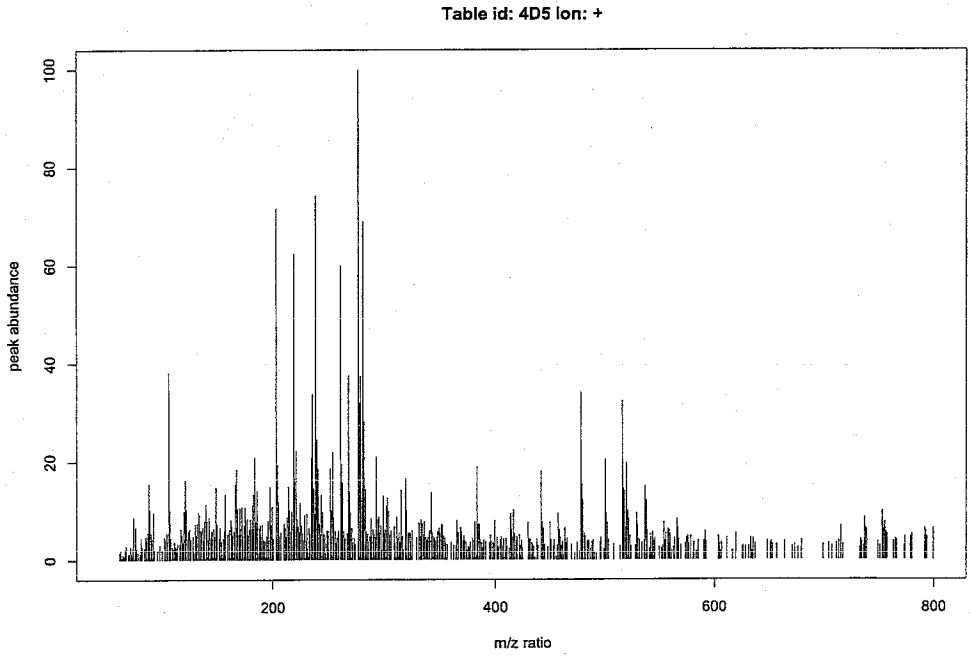

Figure 5.2: Traditional plot of fruit 4D5 after applying binning with $r = [50 \ldots 800)$, $n = 750$ and $f =$ 'maximum intensity of bin'.

The technique satisfies all three objectives. The length is constant and equal to the number of bins n. The distances between the peaks are fixed and regular, because in each peak-table the bins are located at the same m/z values. Additionally, all bins have equal width and the new peaks are created at the mid-points of the bins. So if this technique is applied on a set of tables, the result is a collection of peak-tables that are aligned and have a fixed length. Any traditional classification technique is applicable.

The binning technique is implemented as `align.binning` (appendix E).

5.5.4 Interpolation

After developing the *binning* technique, we realized the term *bin* is used in histograms as well. The binning approach has the similarity to histograms that it puts the data in bins. Closely related to histograms are density estimations. These estimations are often performed with splines. Hence we explored the possibility of viewing a peak-table as a density distribution. Generally speaking, splines are piece-wise polynomials of low degree that are used to interpolate between (data)points. They are used a lot in computer visualization and as function approximations.

We have implemented the spline interpolation using the R package `modreg`. It supplies us with a function `smooth.spline` that calculates a cubic smoothing spline through a given set of data points. The function `predict.smooth.spline` evaluates the spline interpolation at new data points.

There are two parameters: the smoothing parameter `spar` and the sequence of m/z ratios `new.peaks`. The smoothing parameter has the range $(0 \ldots 1.5]$ and it determines how smooth the resulting interpolation is. The higher, the more smoothed. The resampling of the splines is done with `new.peaks`. It determines at which m/z ratios the splines are evaluated. As we can pass any sequence of m/z ratios, the result is not necessarily a fixed inter-peak distance. The result is a new peak-table, with the m/z ratios determined by `new.peaks` and intensities by the interpolating smoothing splines.

We have not explored this area deeply, because the problem of interpolation is that splines smooth the data. In the following chapter, we show that the *binning* technique performs less well, if you take the mean of the intensities in a bin. That is a smoothing operation. Hence we expect that the smoothing nature of splines is detrimental to the discriminatory power in this method. The smoothing effect is observed in the plots of figure 5.3. We expect the classifications to perform better for small values of `spar`, as these values retain the original profile of the peak-table better.

The technique fulfills the three objectives, just like *binning*. The length is constant and the tables are aligned. The objective of creating a fixed, constant inter-peak distance can be accomplished if we pass a sequence of m/z ratios that has the property of a constant inter-element distance. Also, just like *binning*, the spline interpolation focuses on the peak intensities. Therefore it complements *inter-peak distance* as well.

The method in the R package is called `interpolate.spline` (appendix E).

5.5.5 Others...

There are other ideas that have never left the drawing board. We shortly describe two of them. We have observed that feature and classification techniques require a set of one-dimensional vectors. Our peak-tables are two-dimensional. One attempt has been to combine the two vectors in one. A *complex-number encoding* does that: m/z ratio $+ I *$ intensity. Unfortunately we have not been able to find any feature extraction method or classification that easily accommodates for complex data. Additionally, also in complex numbers, we are faced with two dimensions: complex numbers are tuples too.

The second idea has been to add peaks of some value to the tables such that the lengths of the tables become equal. We had intended to place these *phantom* peaks such that the result is a table with a fixed regular inter-peak distance. That would enable us to use Fourier Transforms or Wavelets. Our algorithm was based on the greatest common divisor (GCD) of all inter-peak distances. A first trial showed a practical problem. The GCD was equal to the significance level

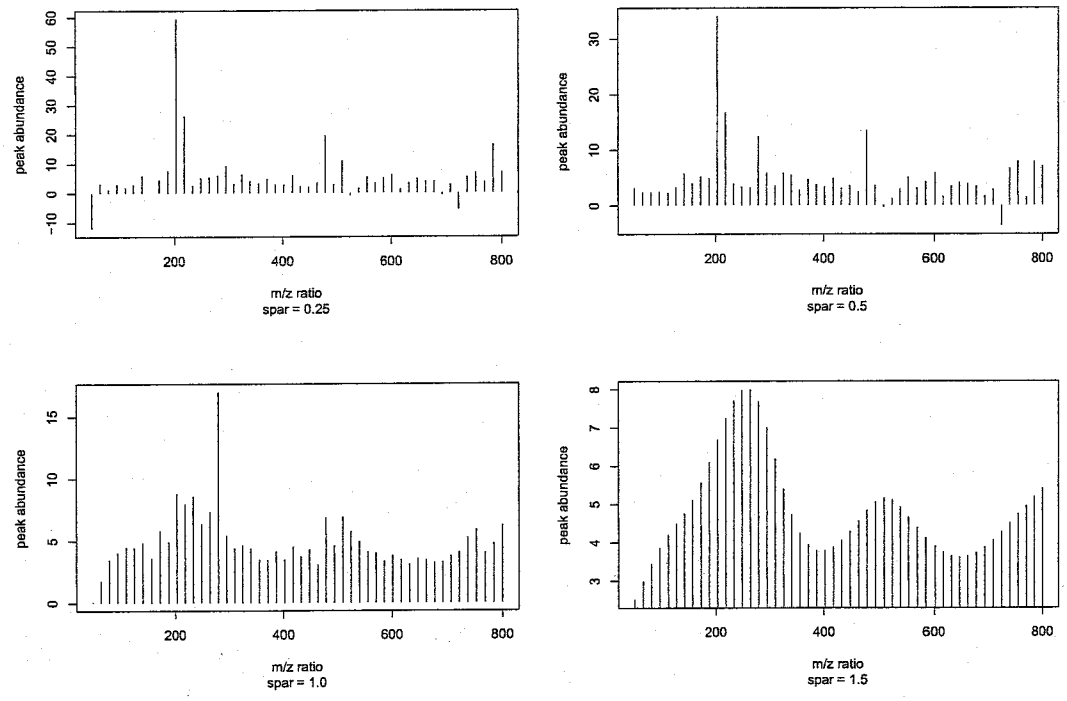

Figure 5.3: Four traditional plots of fruit 4D5 after applying a spline interpolation with different values for the smoothing parameter *spar*. Resampling has been done in the range $[50 \ldots 800)$ at 50 equally spaced points.

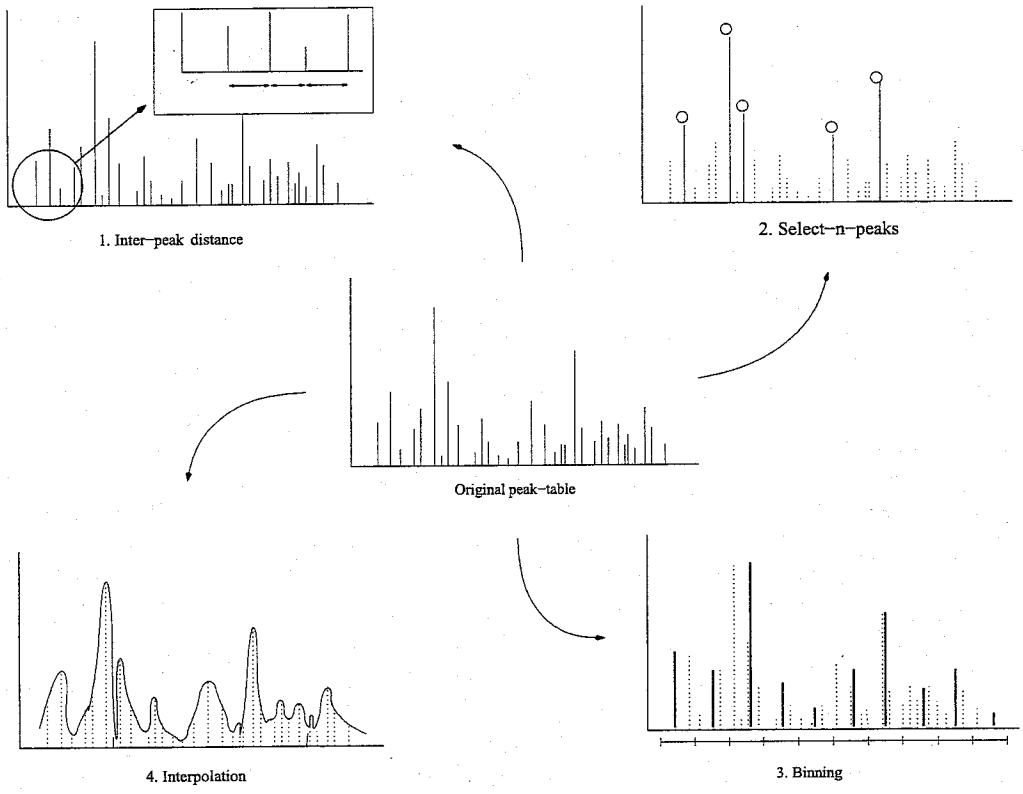

Figure 5.4: The four feature extraction techniques we developed that are applicable on peak-tables. In clockwise fashion one sees (1) *inter-peak distance*, (2) *select-n-peaks*, (3) *binning* and (4) *interpolation*. In the middle, the original peak-table is shown.

of the m/z ratio, 0.0001. That results in the creation of millions of phantom peaks in the data set, which could not be handled by the workstation.

5.6 Improvements

As we describe in the next chapter, some of the solutions have successfully categorized the fruits of parental lines into these two lineages. The application of the same techniques on the introgression lines has not resulted in such a categorization.

Although it is future work, we propose several improvements or additional procedures in this section. We have two goals. First of all, we may extract more information by taking both the m/z ratio and the intensity into consideration. The solutions of section 5.5 take either the m/z ratio or the intensity, but never both of them. Another objective is amplifying the differences between the samples. That should increase the chances that classification algorithms can, for instance, cluster the fruits in the introgression lines.

Remove common peaks

The motivation is that if two samples have differences, but these are very small, the common factors can cloud these dissimilarities. Then chances are that the information loss in the extraction process results in the loss of the differences. In the case of *raw* classifications, the dissimilarities do not have enough 'strength' to expose the structure of the data to the algorithms. Therefore we imagine that removing the common parts of the tables, results in exposure of the differences and that may lead to more structure in the classifications.

There is one possible drawback. If everything looks different from everything, there is no common ground to base the classification on. Then we end up with no categorization as well.

In the preprocessing stage we have developed two algorithms for locating the common peaks of peak-tables. The general approach we have taken in the design of these algorithms, makes it possible that we re-use them to search for the common peaks in the whole tomato data set. The algorithms for keeping or removing common peaks of peak-tables are available as `common.keep` and `common.remove` (appendix E).

Alignment of tables

Our *select-n-peaks* approach has the disadvantage that the peaks are not aligned. That results in the failure of this method to lead to any other classification, but the random one.

The improvement we suggest is, given a set of tables, to align the peak-tables as an extra preprocessing step. The result is a multiple alignment. We are interested in the gaps of such an alignment, because they point at differences in the peak-tables.

We describe two options of processing peak-tables further. We can can fill the gaps with *phantom* peaks. In this manner, we create a set of tables that have a constant length and a regular, fixed distance between the peaks. Note that the inter-peak distance can still vary along the m/z axis. We can also directly use the alignment for calculating some distance measure. These distances can be used in cluster analysis.

It seems that the alignment is somewhat related to *binning*. If one drops the fixed bin size and lets it vary along the m/z axis, the optimal granularity is given by the proposed alignment. With the same line of thought, *interpolation* is closely related too. The improvement over these two techniques is that we are combining the m/z ratios and intensities. We expect that we have less information loss in this manner.

Ratio transformation

This idea takes us back to the preprocessing phase. A common approach to amplify differences is to use a ratio transform. Here, that means taking the (average) peak-table intensities of the

recurrent parent *L. esculentum* and divide the introgression lines by it. A precondition is that the tables are aligned, otherwise one does not know which peak intensity to divide by which.

The ratio transform is beyond the scope of the project. Prior knowledge of the data is used: a set of samples is labeled as the parent, and another set as the introgression lines. Nonetheless, the idea looks fruitful. And one may argue that introgression lines are a special type of cultivars, requiring a special treatment.

Plant level

At the moment we have been working on the level of samples. Also in the classifications, we categorize on the level of fruits. In an attempt to rule out more of the environmentally caused peaks (and thus amplifying the genomic differences) we may move to the level of *plants*.

A possible way is to take the common peaks of a plant. These originate from the same genome, and the same organism. Thus those peaks represent the 'average' state of the organisms fruits. Because each fruit has been growing at a different position on the plant, there may be environmental influences causing different peaks. These are now ruled out.

Currently, the disadvantage is that we have only a few plants. We can interprete the results of classifications involving only plants as hints. These categorizations cannot serve as the basis of conclusions.

5.7 Summary

In this chapter, we have performed feature extraction on the tomato data. We investigated several related research fields for viable extraction techniques without success. The extraction of features from peak-tables, without identifying compounds from the tables, seems to be an unexplored area.

The first attempts of extracting features have uncovered three fundamental problems. In the tomato data, peak-tables differ in length and have variable distances between the m/z ratios of their peaks. These two imply that the peaks are not aligned when one compares two tables.

We propose several solutions for tackling the problems. Three of them are selected for further investigation in the classification phase. These are: inter-peak distance, binning and interpolation. The first is based on the m/z ratios of peak-tables, the other two on the intensities.

In addition to the first solutions, we propose several improvements. The two objectives are combining the m/z ratio and intensity such that we maximize the information content of features and amplifying differences among the fruits.

Chapter 6

Classification

After feature extraction one usually performs a feature selection. We skip the selection stage, because we do not have any criteria for a selection, i.e we do not know what the good features are. In fact, it is our goal to determine what good features are.

So we proceed with the actual classification phase. Data classification is the classification of patterns into predefined classes or into maximally disjunct clusters. As stated in section 2.1 the categorization we apply, is called a *free* classification. It means that we do not impose any structure on the data and that we only use unsupervised classification techniques.

The classification techniques are applied on the features. These features result from inter-peak distance, binning and interpolation. Our toolkit of classifiers consists of Hierarchical Cluster Analysis (HCA), k-means clustering, fuzzy-c clustering and Principal Component Analysis (PCA). HCA is the mainly applied technique, because of its good visualizations (dendrograms) and hence they are easier to interpret. Further investigations are made with PCA and conclusions may be verified with k-means or fuzzy-c.

Further, all classifications are performed on the level of tomato fruits. The motivation is that the features are at the level of fruits. Also, all reasons that apply on the choice for extracting features on the level of fruits (section 5.2), are valid here.

First we shortly describe the classification techniques. The following sections are devoted to the classification on parental lineages and on introgression lines. We describe methods, results and conclusions. We elaborate on the parental lines, but keep the exploration on feature sets in introgression lines short. Finally the classification results are summarized.

6.1 Toolkit

We rely on several cluster algorithms to explore the features. They tackle the problem of finding distinct categories in a set of samples in different ways, but the underlying principles are equal. The inner-cluster distances are minimized, while the dissimilarity between clusters is maximized.

6.1.1 Hierarchical Cluster Analysis

It is a commonly used method in bioinformatics. The most important variants take an *agglomerative* approach. The samples are iteratively joined into clusters according to their distance from each other.

In the project we take the euclidean distance as the distance. It has the disadvantage that it is very sensitive to differences in scales among the samples. That is not an issue in our data set, since the m/z ratios are all in the same range and the peak intensities are all scaled to the interval $[0 \ldots 100]$. For the joining, also known as *linking*, several rules exist. We have used *complete linkage*, which means that the distance of clusters equals the distance between the two furthest

neighbors of different clusters. When the data is distributed in distinct clusters, the rule performs well. There are other linkage rules, but usually the resulting dendrograms differ only slightly [18].

The dendrograms we produce in this project have an axis *height*. The linkage distance between two clusters is displayed in that manner. We do not pay attention to the absolute value, we only interprete the dendrograms qualitatively. We do not derive any conclusions from observing a certain height, because it depends on the specific distance measure, the number of features and the linkage rule.

We use the hierarchical clustering algorithm `hclust` of R package `mva`.

6.1.2 k-means

Here the number of clusters one wants to distinguish is a parameter of the method. Say we want to classify our data in k classes. At first the samples $s_i \in S$ are randomly distributed over the k classes. By moving a sample s_i from one cluster to another only if it increases inter-cluster dissimilarity and decreases inner-cluster distance, the algorithm calculates an optimal distribution of the samples over the clusters.

The fuzzy version of this algorithm is called *fuzzy-c*. It is known to have some advantage over k-means with small data sets, because of the soft classifications. A drawback is that when the number of clusters is specified incorrectly (i.e. not in accordance with the data), the clustering membership is virtually un-interpretable [50].

The application of k-means and fuzzy-c equals cutting the HCA dendrogram at a certain height. The k-means technique is provided by `mva`, fuzzy-c by the R package `cluster`.

6.1.3 Principal Component Analysis

Although strictly speaking PCA is not a clustering technique, we use it as well. It is a well-known mathematical technique, described in many books [50]. It projects n dimensional data onto a lower dimensional subspace in an optimal way. It is optimal in sense of the sum-squared error. We use it as a method for projecting the peak-table onto a 2D plane.

Its limitations are that it performs best for data with zero mean and normalized standard deviation. And the data transformation is linear, i.e. non-linear relations in the high dimensional space are not discovered.

The principal component analysis is provided by the package `mva`.

6.2 *L. esculentum* and *L. pennellii*

The first classifications have been performed on the two parental lineages. The motivation is that it is the easy case: if we look at the view "from above", figure 4.2, the parents are already distinguishable. This is not the case for the introgression lineages. They look almost equal to their recurrent parent *L. esculentum*. We assume if it is hard for us to discriminate them, the same holds (to some extend) for the algorithms.

Although a *free* classification means we do not impose any structure on the data a priori, we do have criteria for interpreting the clusterings. Given the goal of "developing a fingerprint for each species or variety", we primarily look for classifications on the lineage level. A second criterion is to look for plant-clusters. But this is not likely to occur: the plants have been grown in the same environment (a greenhouse). Hence, if we do not find such clusters it is a compliment to the people in Sheffield.

6.2.1 Methods

We have applied three feature extraction algorithms and studied them in more detail. For each of them we discuss the parameter choice, the classifications and the conclusions.

In all the figures we have labelled the fruits according to the plant they come from. It eases the interpretation. The mapping from plant to lineage is given in table 6.1

6.2.2 Inter-peak distance

The extraction method has been introduced in section 5.5.1. There are no parameters to set. Therefore we start the classifications with a HCA. The resulting dendrogram is shown in figure 6.1. The lower the height is at which two fruits are linked together, the more similar they are to the cluster algorithm.

We observe one cluster with four fruits from *L. pennellii*. The rest is seemingly randomly clustered. We can not report a clear categorization in plants or lineages.

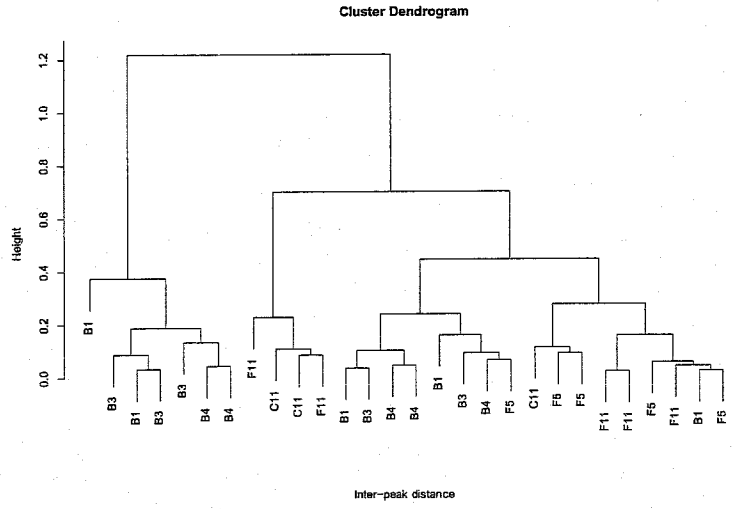

Figure 6.1: Hierarchical Cluster Analysis of inter-peak distances.

A second method for classification, PCA, is used to get a better view on the resolving power of inter-peak distance. Before the principal components are extracted, the data is normalized for optimal performance. The normalization consists of subtracting the mean and dividing by the standard deviation.

Because we have only two features per fruit, namely the mean and standard deviation, we plot the untransformed and PCA-transformed data. See figure 6.2.

The left plot shows a slight separation of the two parental lineages. After a PCA, this separation is observed even more clearly in the right plot. The first component captures 89.9 percent of the data. The second 10.0%. Due to the floating points, the total is not 100%.

One might draw a separating line from the bottom-left corner to the middle of the top plot-border. It shows that the lineages are almost separable. One exception is a sample F11 around $(-0.1, -0.025)$.

The results of the HCA show that the resolving power of the inter-peak distance is not so large, as we expect (section 5.5.1). But the PCA shows not all information of the tables is lost. Perhaps better noise reduction and larger data sets can result in better categorizations.

Lineage	Plant
L. pennellii	C11, F11
L. esculentum	B1, B3, B4, F5

Table 6.1: The parental lineages *L. pennellii* (0716) and *L. esculentum* (3475) with their plants.

6.2.3 Binning

The extraction technique *binning* is described in section 5.5.3. From a feature extraction viewpoint it looks promising. The method is capable of keeping the characteristics of the original tables, see for instance figure 5.2 on page 5.2. Since we could classify the original tables ourselves in the view from above as being from one of the two parents, we expect *binning* can do so as well.

We have assessed the technique for a range of parameters. We take three functions, *minimum intensity*, *mean intensity* and *max intensity*. And to develop an idea of the behavior on different scales, i.e. different interval sizes, we cluster the data for $25, 50, 125, 250, 500$ and 750 bins.

For the minimum-intensity function we expect no interpretable clusterings as the bins probably contain noise peaks. In terms of the cut-off intensity (section 4.4), the peaks are below that value. We know the mean-intensity function acts as a smoothing function. It is expected to be better when the bins are smaller. The maximum function keeps the traits of the original peak-table. Therefore we expect a classification, at least for a high number of bins and on lineage level. In figure 6.3 and 6.4 we show the results of performing HCA on these different feature sets.

A first view on the dendrograms of the minimum function, shows no interesting or obvious clustering in plants or lineage. When we look better, we see that for 25 and 50 bins we can identify two clusters. The left cluster mainly contains *L. pennellii* fruits, while the right cluster has *L. esculentum* fruits. This observation is more profound for the 25 bins. The classification vanishes into uninterpretable (random) clusters when we increase the number of bins. In other words, there is a general trend in the data that points at lineage membership, but it disappears when the number of bins increases. Then noise takes over. A possible scenario for this behavior is that a few of the *lowest* peaks are in fact *real* peaks. They are selected in all cases, but when the number of bins increases, the number of bins with noise increases. After a while the noise clouds the signal of the real peaks and clustering algorithms do not detect the difference anymore.

The mean-intensity function shows a distinct categorization in lineages, although it is not perfect. We have not observed cluster on the level of plants. In contrast with our expectations, HCA does not perform better when the number of bins is increased, it performs worse. We

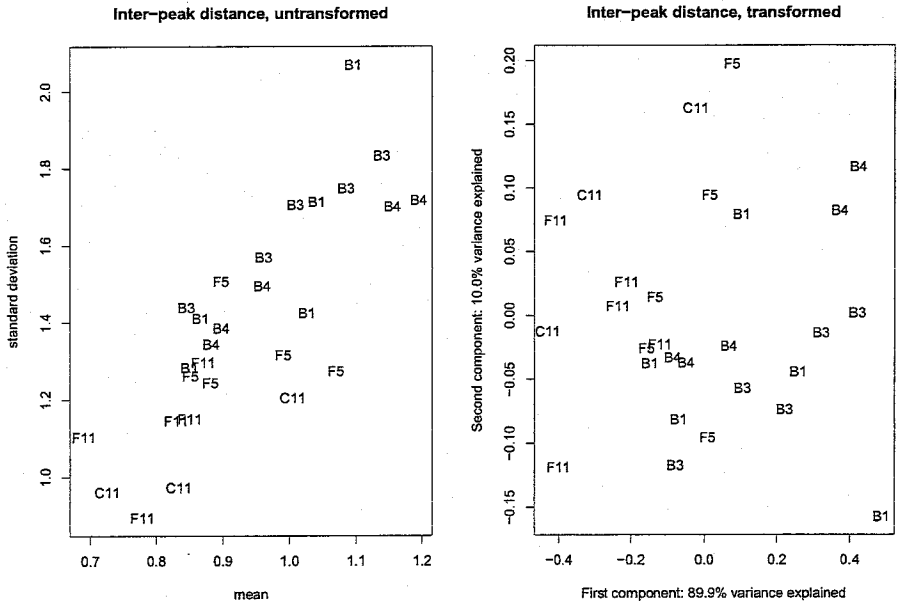

Figure 6.2: The left plot shows the inter-peak features that result directly from the parental data. The right plot shows the PCA-transformed version.

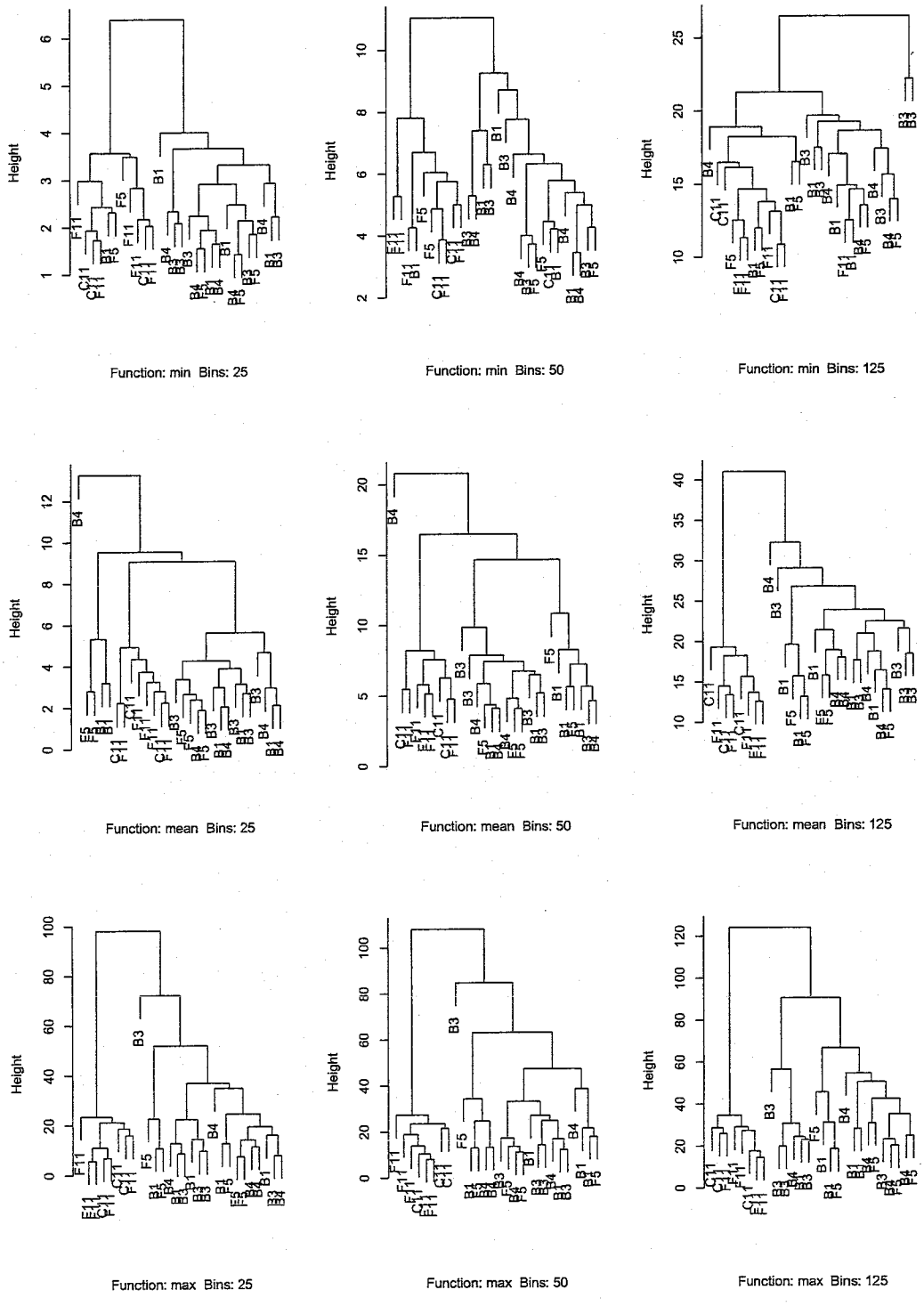

Figure 6.3: Hierarchical Cluster Analysis of binning. The rows differ in number of bins, the columns in the function that is applied on the peaks in each bin.

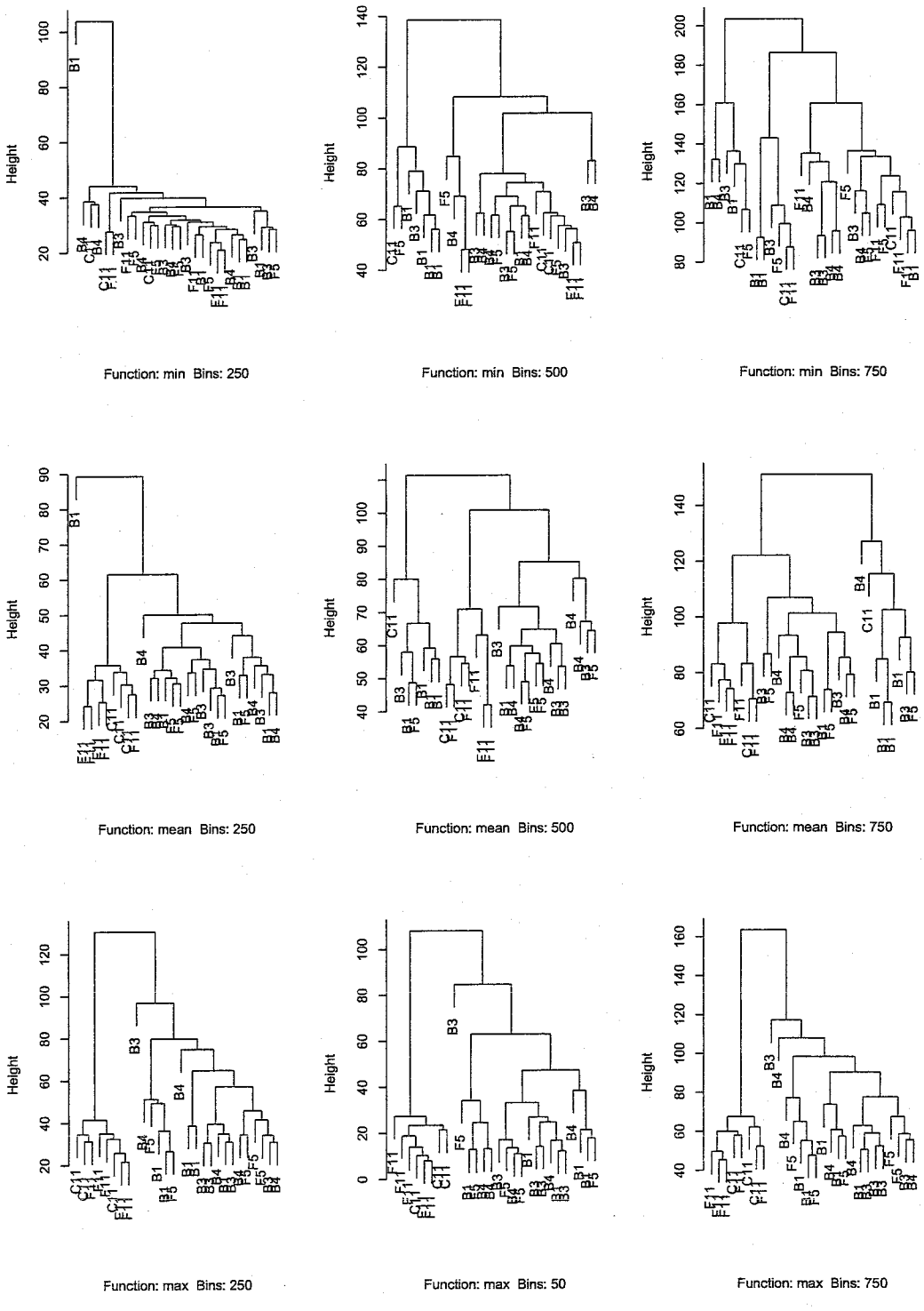

Figure 6.4: Hierarchical Cluster Analysis of binning (continued).

realized that there must be a few *high* peaks unique to one of the lineages. These peaks have a lot of influence on the mean. It is a commonly known characteristic of taking-the-average that very high or low values have a large impact on the final mean value. The same principle as in the minimum function is at work here. When the number of bins is small, the bins are large and hence the high peaks have a larger area of influence. If we increase the number of bins, the influence decreases, which results in a less well-structured clustering. But unlike the minimum function, here the clustering in lineages does not disappear completely. This indicates there is information in the medium high peaks: around the cut-off intensity.

The last function we assess, is the maximum-intensity function. Across the whole range of bins, there is a clear clustering in tomato lines. In that sense there is not much else to observe. Increasing the number of bins, increases the differences between the lineages: their clusters become more compact and the joining happens at a relatively higher height. Even the outliers that occur in the dendrograms of the mean-intensity are included in the correct cluster. So the strongest signal, i.e. the most information that leads to a discrimination on lineage level, is in the high intensity part of the peak-table. This holds regardless of the number of bins.

We verified the clustering in tomato lineages with k-means. If we assume the average *raw* peak-table contains two vectors of 3500 values and we compare that to the maximum-intensity function with 25 bins, we have practically reduced the data set to less than 0.36% of its original size. This is worse than the inter-peak distance (which keeps all information in two variables), but given the discriminatory power of *binning*, that is not such a drawback.

6.2.4 Interpolation

Interpolation is designed and described in section 5.5.4. The interpolation is done with smoothing splines. We expect that smoothing less is beneficial for an interesting clustering, as the shape of the peak-table is preserved better.

We have varied two parameters. The smoothing parameter is set to the values 0.5, 1.0 and 1.5. For each of these parameter values, we take the number of resampling points equal to the number of bins in the above classifications: $25, 50, 125, 250, 500$ and 750. Like in wavelet theory, we can explore trends on different scales.

The expectation is that from a smoothing value of 0.5 to 1.5 the results become worse. We mean that the dendrograms show their clusters becoming less interpretable. Because the smoothness is maximal at 1.5 and is a truly smooth curve along the m/z axis, we expect the clustering to be more or less invariant to the number of resample points. The results of a HCA are depicted in figures 6.5 and 6.6.

The results show an unexpected outcome. The dendrograms become less interpretable in the direction exactly opposite to our prediction. We observe clusters on the level of tomato lineages in the bottom row and random clustering in the top. We expected it vice versa.

A possible explanation is that the smooth curve in the bottom row is strongly determined by the highest values. Then, just like in mean-intensity binning, if the parents differ in a few high peaks (and they do!), then it results in very different interpolations.

On the other side, splines that do not smooth so much, destroy information. Apparently, interpolating through the intensities and resampling on other m/z ratios destroys the original peak heights and results in information loss. The observation that in the middle row the number of bins needs to increase to accomplish a clearer clustering in tomato lineages, is in accordance with the above explanation. This holds at least for the usage of cubic smoothing splines.

The predicted saturation in the number of bins to get clearer clusterings has been observed. The smoothness effectively eliminates the need to increase this number. We conclude that interpolation behaves quite differently then expected. In future work tuning the smoothing parameter in combination with resampling in a more sophisticated manner may be useful.

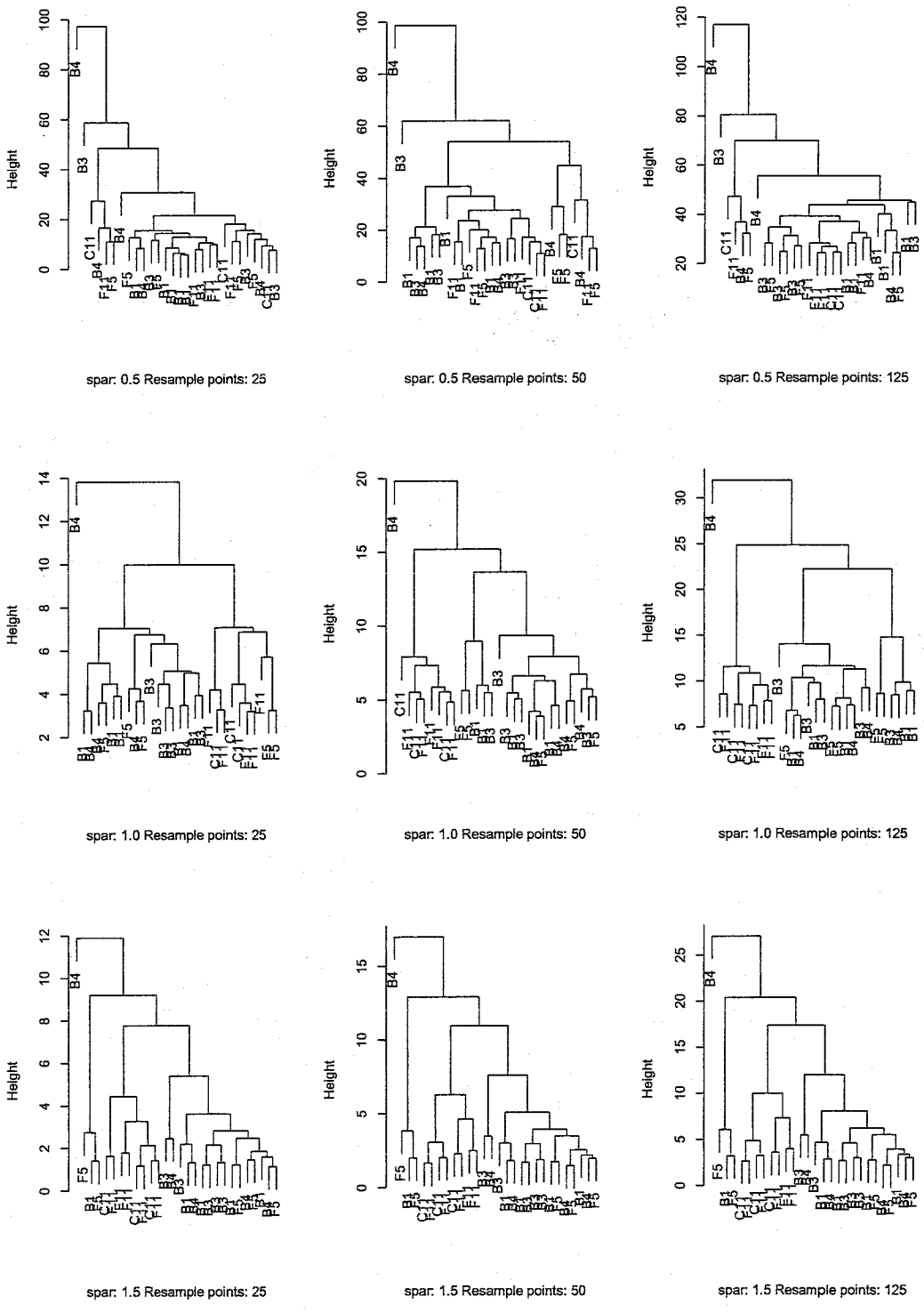

Figure 6.5: Hierarchical Cluster Analysis of interpolation. The rows differ in number of resampled m/z ratios. The columns differ in the value of the smoothing parameter.

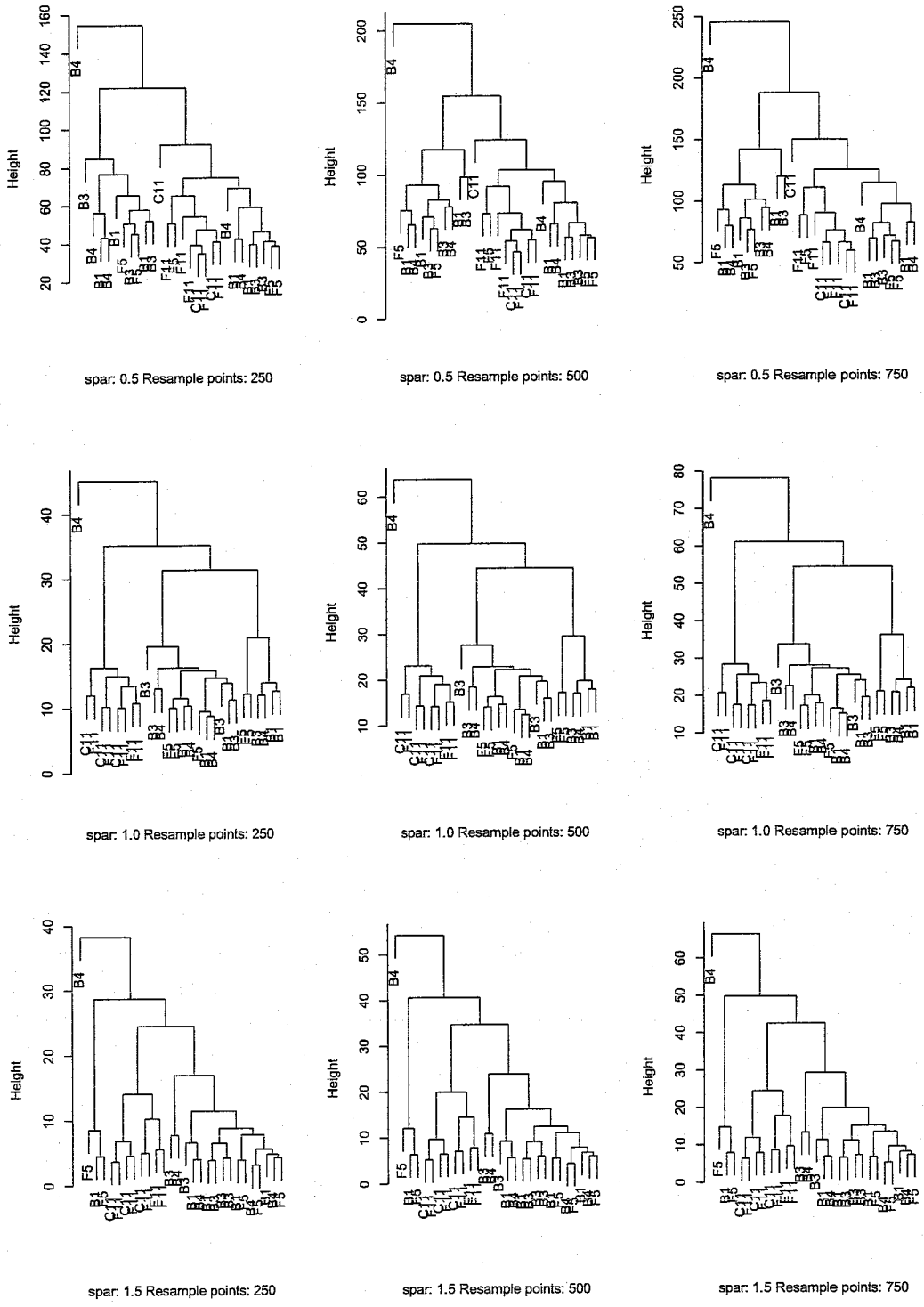

Figure 6.6: Hierarchical Cluster Analysis of interpolation (continued).

6.2.5 Conclusion

Summarizing the results and conclusions, we state that there is a (strong) signal in the tomato data regarding the parental lineages. The m/z ratios have some information, even if we take statistical moments. The intensities seem to contain information on every scale. A crude technique as *binning* is capable of separating the tomato lineages into more or less distinct clusters on several levels of intensity. The interpolation has shown that the exact values of intensities are important characteristics of peak-tables. They are needed in the categorization of tomato lineages.

With the successful unraveling of some peak-table structure, we turn our focus to the introgression lines. We have taken a few steps in that field, but time limitations have stopped us. In the evaluation we propose several possibilities for future work.

6.3 Introgression lines

We have successfully found structure in the parental data. The extraction methods provide us with features from the data such that unsupervised classification algorithms categorize the data into two clusters, one for each parent. The next step is to hard case: clustering introgression lines.

We do not know *if* there is a difference between the introgression fruits, as the function of the *L. pennellii* chromosome segments is unknown. It is possible that the genes on this segment are only expressed in other tissues or at some other stage of development.

We have randomly chosen two of the four introgression lines for the clustering. Their line-ids are: 4057, 4067. In table 6.2 the mapping from plant to line is given.

Lineage	Plant
IL-4057	D9, E9, F2
IL-4067	A8, E4, E11

Table 6.2: The introgression lines and their corresponding plants.

6.3.1 Binning and k-means

Because introgression lines are so much alike, we have decided to apply the best technique we have. This means we extract features with *binning*. As in the classification of the parents we vary the two parameters of binning when we perform hierarchical cluster analysis.

We have assessed the binning technique for three functions, *minimum intensity*, *mean intensity* and *max intensity*. Again, we aim to develop an idea of the behavior on different scales. Therefore we cluster the data for 25, 250 and 750 bins.

We expect no categorization into the two original lineages for both the minimum and mean function. The results of these two functions in the parental case showed only that there is some information, but not much, in the low-intensity part of the table. We might find a categorization when the maximum function is applied. The result is shown in figure 6.7.

In the HCA no clusters on either plant or lineage level are observed at all. In some cases lines group together, but it happens throughout the dendrogram. The algorithm does find two clusters with the maximum function. We cannot interprete these as clusters that reveal any hidden structure of the data.

We perform a k-means clustering on the data in an attempt to apply a mixture resolving technique on this discrimination task. The result is shown in table 6.3. As one can see, the algorithm does not cluster the fruits in the 'correct' categories.

6.3.2 Conclusion

Although it could be there is no biological difference in the mature fruits of introgression lines, we think it is more likely that the extraction methods have failed. The binning technique performed

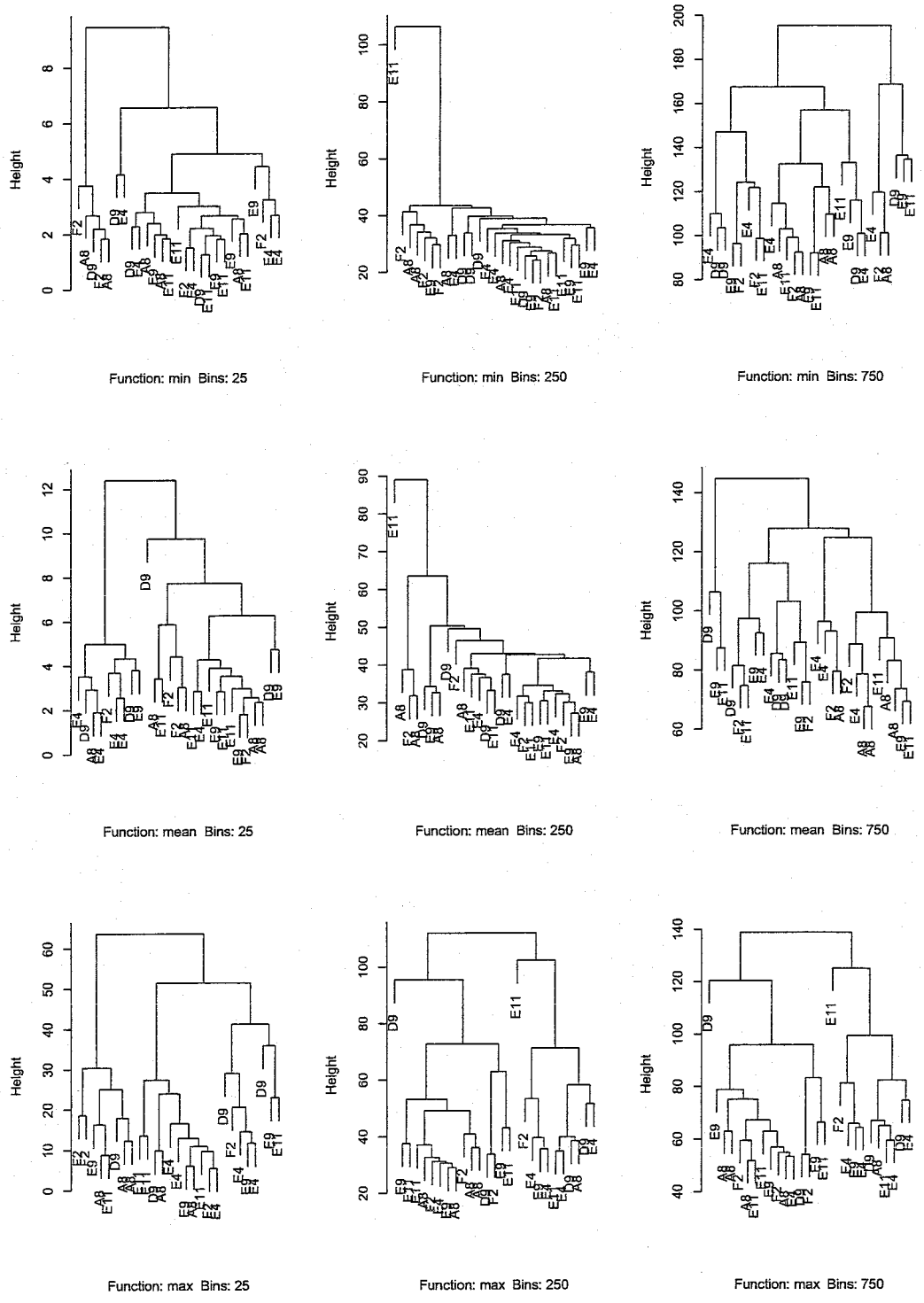

Figure 6.7: Hierarchical Cluster Analysis of binning on two introgression lines. The rows differ in number of bins, the columns in the function that is applied on the peaks in each bin.

so well for the parental lines, we assumed it would give at least a hint of a clustering in lineages.

We realize the attempts for finding an understandable classification have been kept at a minimum. On the other hand, we do apply the best technique. If it fails, we assume the rest of the extraction techniques will fail as well. Therefore improvements are necessary. In the next, and last, chapter we propose several of them to enhance the discriminatory power of the classifiers.

Unfortunately, we do not have an explanation for the clustering that does occur in both classifiers.

6.4 Summary

In this chapter we have successfully applied three extraction techniques on the parental tomato lineages. To some extend they all separate the lineages, given the right parameter settings. No clustering on plant level has been observed.

The inter-peak distance shows that there is information in the m/z ratios of peaks. Their distribution differs not enough for a clear hierarchical clustering, but principal component analysis on two components shows a nearly linear separable data distribution.

The binning technique has been valuable. Our exploration of the structure of peak-tables seems to reveal that information concerning lineage identity is found among peaks of all intensities. Even the lowest peaks indicate from which lineage a fruit is, although it is not a perfect classification.

Interpolation has shown that it is not a good idea to alter the peak intensities. Information loss is the result. Together with the mean-intensity function in binning, interpolation has shown that smoothing is probably only a viable approach if there are several high peaks that are unique for one lineage.

Applying the same extraction techniques on introgression lines has not given such results. No categorization on the level of plants of lines has been observed. Improvements are necessary for this type of data.

	IL-4057	IL-4067
Cluster I	14C7, F216, 11D10, 11E3, 14F6, 14G9, 14H7, 13H2	2F1, 3C2, 6G4, F229, 11F1, 9G7, 9D5, 10C2
Cluster II	11H2, 7A6, 8H7, 8G4	1C2, 11E8, 9C12, 10C7, 6G9, F126, F131

Table 6.3: The result of the clustering by k-means. Cluster I and II give the k-means categorization. The categorization by lineage is in the columns.

Chapter 7

Discussion and future work

7.1 Discussion

From a biological viewpoint we have gained several insights and we can articulate many more questions. First of all, one has to realize the metabolome is where the hereditary information of the DNA meets the environment. How large are the influences of the one process and the other? We already concluded there is never grouping on plants. Given that the environment of the plants is virtually the same, this is not entirely unexpected. But it also means that we are viewing genetic differences in the tomato dataset, at least different reactions to the same environment.

The aims of Sheffield University are to rapidly screen samples and develop fingerprints of species. Both are viewed as metabolic fingerprinting tasks, where the actual differences are not related to specific metabolites. That is also the drawback of our approach. If speed is essential and, for instance, one wants to screen samples if they look different from some standard, the direction we have taken suits the needs. But it is virtually impossible to explain the difference in the biological sense, i.e. pointing out a change in metabolic activity, which metabolites are involved and ultimately which genes are responsible. If one wants to unravel a part of the metabolic network, the compounds in a peak-table need to be determined.

The categorization on the level of lineages has been reported before in metabolomics [51, 52]. But many other aspects we seem to observe, such as that very low peaks seem contain information (at least in the tomato data), are not solid facts. We realize a better verification of such results is necessary.

Another aspect is the question how much of what we observe is specific for tomatoes and what is a general trait of metabolic mass spectrograms. We have attempted to abstract from the specific data set. A definitive answer requires patience before this matter is resolved: the more plants are examined on the metabolic level, the more generalizations researchers can make.

The fact that we cannot discriminate among the introgression lines, may have several causes. Besides that our techniques are just not good enough, it may be the case that there are no obvious differences in the relative concentrations of biomolecules. Mass spectrometry does not measure the absolute differences between tomato fruits. It could be there are no relative differences. Another explanation is to suggest the chromosome segment is not expressed in the fruit (pericarp) tissue. Or only in an other stage of development and now no longer visible. And these are only a few and the most straightforward explanations. At least we are not alone, in [52] unsupervised classification of similar nearly-isogenic hybrids does not result in a interpretable clustering either. From genome to metabolome is a long way, especially when science just starts to understand each one of them and the influences they have on each other.

7.2 Future work

Our work in the new field of bioinformatics for metabolomics showed us there are numerous challenges ahead. Obviously, they mainly concern the actual analysis, but also data management.

The data set itself caused several hurdles. First there is the data format problem. It is only a technical issue, but still one that needs to be dealt with. We believe plain text (f.i. XML) would be a good vehicle for future data exchange between research groups.

Second, the organization of metabolic data has to be improved in the future as well. We designed a small, project-tailored, database, because there is no standard for the storage of meta-experiment data (where samples originate from and so on). The problem is the exchange of the data and the results. In transcriptomics the data format and storage is obstructing progress, so we must handle the issue professionally.

Besides the data management, we mention that the current implementation of our methods in R is a first start. We have used R more as a prototyping language. The quote 'software is never finished' certainly seems applicable.

In the future, there are improvements possible across all stages of pattern classification. A better method than regression noise reduction must be developed, if metabolomics is to be automated further. We see two immediate improvements. First of all, noise reduction may be simplified and enhanced with the use of alignments of the replicate tables. We are already investigating the subject using the formal framework of pairwise and multiple sequence alignments.

As a second case, we see possibilities in more heuristic approaches to the determination of the accuracy boundary. It is a delicate balance between the accuracy of m/z ratios and accuracy of intensities. We attempted to search three replicates of sample 4D5 manually for real and noise peaks. We derived the following rules from our experiment:

- some peaks are easy to distinguish; their m/z ratios are almost the same and their intensities are dissimilar from the surrounding intensities,
- some peaks are more difficult, but their m/z distances to the previous and next peaks are so much larger (10^2) than the distances among themselves.

Building an algorithm around these two rules may be very fruitful in noise reduction. Once noise is removed from the tables, the problem of analyzing the peak-tables could be modelled as a more classical alignment problem.

Additionally, the peaks are aligned as well in the most successful feature extraction techniques *binning* and *interpolation*. HCA then calculates distances between these alignments. Regular peak distance is only a side-effect and not directly needed, as we mentioned in section 5.6. We may drop that effect and achieve much better alignments (and hence better distance measures) if we can apply pairwise or multiple alignments.

The feature extraction chapter provides a range of improvements (section 5.6. We shortly repeat them. We may remove the common peaks to enhance differences among fruits. The level of fruits is perhaps not the best to work on, moving to the plant should eliminate some environmentally caused peaks. If we want to separate the introgression lines, a ratio transform with their recurrent parent may be fruitful.

The incorporation of labeled peaks, i.e. extracting the compounds from the tables, may give us the extra information that is needed for the discrimination of introgression lines. If we accomplish that classification, we show that our extraction techniques are sensitive enough to detect genomic differences of only a few genes. Then working on the metabolic level truly complements and enhances the other omic disciplines.

Appendix A

Abbreviations

2D	Two Dimensional
3D	Three Dimensional
ADT	Abstract Data Type
ANN	Artificial Neural Network
BLAST	Basic Local Alignment Search Tool
BPNN	Back-Propagation Neural Network
DBMS	Database Management System
DFT	Discrete Fourier Transform
DNA	Deoxyribonucleic Acid
DSP	Digital Signal Processing
DP	Dynamic Programming
ECG	Electro Cardiogram
EST	Expressed Sequence Tag
FASTA	Fast All; fast protein or nucleotide comparison
FT-IR	Fourier Transform - Infra Red
GA	Genetic Algorithm
GC	Gas Chromatography
GP	Genetic Programming
HCA	Hierarchical Cluster Analysis
HMM	Hidden Markov Model
ICA	Independent Component Analysis
kNN	k Nearest Neighbor
LC	Liquid Chromatography
LVQ	Learning Vector Quantization
MLP	Multi Layer Perceptron
mRNA	Messenger RNA
MS	Mass Spectrometry
NMR	Nuclear Magnetic Resonance
ORF	Open Reading Frame
PCA	Principal Component Analysis
QTL	Quantitative Trait Loci
RBNN	Radial Basis Neural Network
RNA	Ribonucleic Acid
SA	Simulated Annealing
SOM	Self-Organizing Map
SQL	Structured Query Language
SVM	Support Vector Machine
tRNA	Transfer RNA
TU/e	Eindhoven University of Technology

UE	University of Exeter
UML	Unified Modelling Language
US	University of Sheffield
XML	Extensible Markup Language

Appendix B

Planning

The duration of a thesis project at the faculty of Computing Science at TU/e is an academic year, which is effectively nine months. The project is performed at the University of Exeter (UE), in the School of Engineering and Computer Science. Because the academic year starts a month later in the United Kingdom than in the Netherlands, the timetable starts in October, instead of September.

During the project, several subgoals have to be fulfilled. For each of them we give guidelines on the date that they should be finished. One could call it *soft deadlines*. In table B.1 the planning is shown.

Date	Task	Description
7 Oct '02	Start of project	
12 Dec '02	Literature review	Research literature on the subjects of *omics* and bio-informatics.
16 Dec '02 10 Jan '03	Christmas vacation	
6 Feb '03	Data acquisition	Collect data (mass spectrograms) from metabolic research.
27 Feb '03	Feature extraction	Extract features from the data.
24 Mar '03 25 Apr '03	Easter vacation	
15 May '03	Modeling	Select suitable algorithms for detection of patterns and (free) classification of the data.
26 Jun '03	Documentation	Write a report on the thesis project. Give a presentation at UE and TU/e.

Table B.1: Initial planning.

Delay

In the phase of data acquisition we incurred a delay. Though contact with University of Sheffield (US), which provided the data (see section 2.4), had been established quickly, the actual receipt of the data was at the beginning of March. This has resulted in a delay of a month.

Another event caused an extra delay, though only small; in the order of a week or two. It is the switch from programming environment. In section 3.2.2 additional information is given and reasons are explained.

The two delays overlapped. So finally, there was a total delay of four to five weeks. Therefore the schedule has been shifted with four weeks. Also, the Easter vacation has been shortened by a week.

Appendix C

Technical details

All software is developed, tested and used on a workstation at University of Exeter with the following characteristics.

Architecture:	Pentium III 600 MHz 128 Mb RAM 20 Gb Hard Disk
Platform:	RedHat Linux 7.3
Programming Language: with packages:	R 1.7.0 (2003-04-16) base, ctest, DBI, gregmisc, MASS, methods, modreg, multiv, mva, RMySQL, stepfun
Database:	MySQL Ver 11.18 Distrib 3.23.55 for pc-linux (i686)
Text Editor: Text Processor:	XEmacs 21.4 (patch 6) TeX (Web2C 7.3.1) 3.14159
Presentation Editor:	OpenOffice v 1.0.1

Appendix D

File conversion

The Animal & Plant Sciences Department of Sheffield University provided the data for this project in Microsoft Word Format. This format is not desirable for analysis with R, as R can only handle some statistical package formats (for instance SPSS), plain text (ASCII or ANSI/UNIX) and the native, binary, R format. We decided to convert the peak-tables from Word format to ANSI/UNIX plain text. Below the steps taken to accomplish it, are described.

1. *Microsoft Word* to *OpenOffice Writer*: for some unknown reason all the available MS Word to text conversion applications failed in the conversion of the peak-tables. Only OpenOffice Writer is capable of converting the tables in a batch job. Unfortunately, the only output format is the native Writer format, a compressed XML format.
2. *OpenOffice Writer* to *ANSI/UNIX plain text*: the OpenOffice website[1] provides a Writer-to-plain-text conversion script. After eliminating a bug from the script, we have been able to convert the peak-tables to plain text.

The University of Sheffield has noticed that during the creation of the Word documents, a little error is introduced in some peak-tables. An example is shown in table D.1.

m/z ratio	peak intensity
⋮	⋮
81.8772	2.200e1
81.9752	2.400e1.9952 1.000e2
82.0360	2.180e2
⋮	⋮

Table D.1: An example of the formatting error in Microsoft Word documents. In the middle line the faulty peak intensity is in the second column, the correct intensity is in the third column.

Sometimes an extra "strange" peak abundance with a decimal number in the exponent occurs. Using regular expressions and the stream editor *sed*, a script, that eliminates this anomaly from the tables, is written.

After these conversions and the error correction, the data is ready for analysis.

[1] http://www.openoffice.org

Appendix E

Function overview

Here a list of the function and methods is given, together with a short description of their working. We have categorized the functions by pattern recognition stage and supportive task. Detailed descriptions with argument listings and usage examples are given in the help files of the software package.

Name	Description
`peaktable`	The encapsulation of the m/z ratio and peak intensity, together with ion-mode and type of table.
`tomatoes`	The container of the data set. It stores peak-tables and imposes a structure of fruits, plants and lineages on the tables.
`common`	The base class encapsulating the results of a noise reduction. Its children are mainly used internally and for debug purposes.
`common.regr`	Inherits from `common`. The encapsulation of a noise reduction result with the accuracy curve of a peak-table set by a regression.
`common.incr`	Inherits from `common`. The encapsulation of a noise reduction result from the incremental search.

Table E.1: Classes.

Name	Description
`id`	Returns the identifier (not the database-generated one) of a `peaktable`.
`length`	Returns the length of a `peaktable`.
`type`	Returns the type of a `peaktable`.
`ion`	Returns the ion-mode of a `peaktable`.
`show`	Prints a peaktable on the console in a shortened format.
`summary`	Returns a summary of the m/z ratio and the intensity (recursion).
`plot`	Returns a two-dimensional traditional plot of a peak-table.

Table E.2: Standard methods for classes.

Name	Description
`hist`	Plots a histogram of a `peaktable` using the square-root of the length as the number of bins.
`hist2d`	Prints a table (matrix) containing a 2D histogram of a `peaktable`.
`boxplot`	Plots a box-whisker plot of a `peaktable`.
`plot.heat`	Plots a novel view "from above" of all tables in a `tomatoes` object.

Table E.3: New visualization methods.

Name	Description
`read.peaktable`	Given an absolute filename, the ion-mode and the type of table, a table is read into R.
`write.peaktable`	Given a `peaktable` and a `file` connection object, the table is written to the file.
`tomatodb.connect`	Open a connection with the database.
`tomatodb.query`	Query the database with an SQL query.
`tomatodb.disconnect`	Given a database connection, close it.
`import`	Given a fruit, plant, lineage or only an ion-mode, it loads the corresponding peak-tables into an `tomatoes` object.
`export`	Given a `tomatoes` object, its peak-tables are stored in the file system and the database is updated. `[untested]`

Table E.4: Database and file system interface.

Name	Description
ecdf.tomatoes	Calculates an empirical cumulative distribution frequency graph of a peaktable or a tomatoes object.
count.peak	Return number of peaks in a peaktable.
max.x.peak	Return the peak with the maximum m/z ratio of a peaktable.
max.y.peak	Return the peak with the maximum intensity of a peaktable.
mean.peak	Return the mean m/z ratio and mean intensity of a peaktable.
median.x.peak	Return the median m/z ratio and corresponding intensity of a peaktable.
median.y.peak	Return the median intensity and corresponding m/z ratio of a peaktable.
min.x.peak	Return the peak with the minimum m/z ratio of a peaktable.
min.y.peak	Return the peak with the minimum intensity of a peaktable.
sd.peak	Return the standard deviation of the m/z ratio and intensity of a peaktable.
scale.sqrt	Square-root scaling of the intensities.
scale.log	Logarithmic (ln) scaling of the intensities.
scale.linear	Linear scaling of the intensities.
scale.rank	Rank scaling of the intensities.
scale.ratio	Ratio transformation, peak-table is divided by other peak-table.
criterium.sheffield	Sheffield's regression.
criterium.exeter	Exeter's linear regression.
criterium.poly2	Second degree polynomial regression.
criterium.poly3	Third degree polynomial regression.
accuracy.window	Calculates accuracy error of a given window.
extract.all	Search the peak-tables for common peaks (slow).
extract.approx	Search the peak-tables for common peaks (fast).
search.common.regr	Search with a regression for common peaks.
search.common.incr	Search with a data-driven approach for common peaks.
remove.peaks	Given a set of indexes, remove the peaks that are indexed in a peak-table.
keep.peaks	Given a set of indexes, keep the peaks that are indexed in a peak-table.
combine.peaktable	Given a matrix of common peaks and a set of peaktable, combine the peak-tables into one table.

Table E.5: Preprocessing.

Name	Description
interpeak.distance	Features are the distance between m/z ratios of neighboring peaks.
interpeak.moments	Takes the mean and sd of the inter-peak distance.
align.binning	'Molds' a peak-table into a sequence of bins.
align.interpolation	Interpolates and resamples the peak-table.

Table E.6: Feature extraction.

Appendix F

Regression results

All tables are 2D histograms. The first table shows the fruit 4D5. It is followed by the Sheffield regression. The next three tables give the results of searching for combinations of peaks with the accuracy margin defined by a regression. The last table gives the results of the incremental technique.

All searches for combinations of peaks are performed with the approximation algorithm, see section 4.6. In the tables, the horizontal direction equals moving along the m/z axis. We have seven intervals, spanning the m/z range from zero to 800. In the vertical direction, we have increasing intensity intervals. The bottom row "Columns" gives the total number per m/z interval.

Original	0..100	100..200	200..300	300..400	400..500	500..600	600..800	Rows
0..1	131	330	128	201	139	14	2	945
1..4	279	417	389	257	247	285	400	2274
4..10	11	85	117	84	40	55	141	532
10..30	2	18	24	7	5	6	0	63
30..101	0	2	10	0	1	1	0	14
Columns	423	851	668	549	432	361	543	3828

Table F.1: A 2D histogram table. The average number of peaks of the replicates of fruit 4D5.

Linear	0..100	100..200	200..300	300..400	400..500	500..600	600..800	Rows
0..1	5	14	3	6	3	0	0	31
1..4	35	75	72	64	44	39	24	353
4..10	11	68	64	62	29	33	20	287
10..30	2	18	24	8	5	6	0	63
30..101	0	2	10	0	1	1	0	14
Columns	53	177	173	140	82	79	44	748

Table F.2: 2D histogram table. The number of peaks located by the linear regression of Sheffield.

Linear	0..100	100..200	200..300	300..400	400..500	500..600	600..800	Rows
0..1	5	19	5	13	6	0	0	48
1..4	36	85	87	77	56	50	37	428
4..10	11	68	65	64	31	35	27	301
10..30	2	19	24	8	5	6	0	64
30..101	0	2	10	0	1	1	0	14
Columns	54	193	191	162	99	92	64	855

Table F.3: 2D histogram table. The number of peaks located by the linear regression of Exeter.

Polynomial	0..100	100..200	200..300	300..400	400..500	500..600	600..800	Rows
0..1	11	14	0	0	1	0	0	26
1..4	64	71	42	23	26	34	45	305
4..10	11	69	48	50	25	33	28	264
10..30	2	19	24	8	5	6	0	64
30..101	0	2	10	0	1	1	0	14
Columns	88	175	124	81	58	74	73	673

Table F.4: 2D histogram table. The number of peaks located by the polynomial regression of the second degree.

Polynomial	0..100	100..200	200..300	300..400	400..500	500..600	600..800	Rows
0..1	5	19	3	0	1	0	0	28
1..4	36	85	69	33	23	27	43	316
4..10	11	69	58	54	25	32	28	277
10..30	2	19	24	8	5	6	0	64
30..101	0	2	10	0	1	1	0	14
Columns	54	194	164	95	55	66	71	699

Table F.5: 2D histogram table. The number of peaks located by the polynomial regression of the third degree.

Data-driven	0..100	100..200	200..300	300..400	400..500	500..600	600..800	Rows
0..1	0	0	0	0	0	0	0	0
1..4	7	16	11	1	1	0	0	36
4..10	10	43	24	21	7	5	1	111
10..30	2	16	22	8	5	5	0	58
30..101	0	2	10	0	1	1	0	14
Columns	19	77	67	30	14	11	1	219

Table F.6: 2D histogram table. The number of peaks located by the data-driven approach.

Bibliography

[1] Bioconductor, http://www.bioconductor.org.

[2] Biopython, http://www.biopython.org.

[3] Matrix science ltd (micromass masslynx), http://www.matrixscience.com/help/instruments_masslynx.html.

[4] A new view of statistics, http://www.sportsci.org/resource/stats.

[5] Platform plant metabolomics, http://www.metabolomics.nl.

Recent website started by the Platform Plant Metabolomics. The site is a portal to the plant metabolomics community.

[6] Python, http://www.python.org.

[7] Spectroscopy now, http://www.spectroscopynow.com.

[8] C. Beecher. Metabolomics: a new "omics" technology. *American Genomics/Proteomics Technology*, 2002.

Article providing a view on the current state of metabolomics and its potential. .

[9] W. Blackstock and M. Mann. A boundless future for proteomics? *TRENDS in Biotechnology*, 19, 2001.

[10] J.R. Bock and D.A. Gough. Predicting protein-protein interactions from primary structure. *Bioinformatics*, 17, 2001.

[11] S.E. Van Bramer. An introduction to mass spectrometry. 1998.

This article is a nice tutorial on mass spectrometry, its advantages and problems.

[12] A. Buchholz, J. Hurlebaus, C. Wandrey, and R. Takors. Metabolomics: quantification of intracellular metabolite dynamics. *Biomolecular Engineering*, 19, 2002.

The article describes a new experimental procedure to obtain metabolomic data as well as a theoretical model that describes the metabolic network.

[13] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. 1998.

[14] Tomato Genetic Resource Centre. http://tgrc.ucdavis.edu/.

[15] J.M. Chambers. *Programming with data: a guide to the S language.* Springer, 1998.

[16] D. Christendat, A. Yee, A. Dharamsi, Y. Kluger, M. Gerstein, C.H. Arrowsmith, and A.M. Edwards. Structural proteomics: prospects for high throughput sample preparation. *Progress in Biophysics and Molecular Biology*, 73, 2000.

[17] P. Crochat and D. Franklin. Back-propagation neural network tutorial. 2002.

[18] A.B.M. Crombach. Literature review on omics and bioinformatics, 2002.

[19] R.K. De, J. Basak, and S.K. Pal. Unsupervised feature extraction using neuro-fuzzy approach. *Fuzzy Sets and Systems*, 126, 2002.

[20] S.R. Eddy. Hidden markov models. *Current Opinion in Structural Biology*, 6, 1996.

[21] Y. Eshed and D. Zamir. An introgression line population of *Lycopersicon pennellii* in the cultivated tomato enables the identification and fine mapping of yield-associated qtl. *Genetics*, 1995.

[22] D.A. Fell. Beyond genomics. *TRENDS in Genetics*, 17, 2001.

[23] M. Fellenberg, K. Albermann, A. Zollner, H.W. Mewes, and J. Hani. Integrative analysis of protein interaction data. 2000.

[24] O. Fiehn. Combining genomics, metabolome analysis, and biochemical modelling to understand metabolic networks. *Comp Func Genom*, 2, 2001.

Review article on all aspects of metabolomic analysis. Excellent with respect to exposing (current) bottlenecks in sample preparation and subsequent data gathering.

[25] O. Fiehn. Metabolomics - the link between genotypes and phenotypes. *Plant Molecular Biology*, 48, 2002.

[26] O. Fiehn, S. Kloska, and T. Altmann. Integrated studies on plant biology using multiparallel techniques. *Current Opinion in Biotechnology*, 12, 2001.

It gives an overview of all four omics sciences, their interdisciplinary nature, the need for standards in the data management and a quick view on the current analysis software.

[27] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison Wesley, 1994.

[28] T. Genoud, M.B. Trevino Santa Cruz, and J. Métraux. Numeric simulation of plant signaling networks. *Plant Physiology*, 126, 2001.

[29] R.J. Gilbert, J.J. Rowland, and D.B. Kell. Genomic computing: explanatory modelling for functional genomics. 2000.

[30] N. Glassbrook, C. Beccher, and J. Ryals. Metabolic profiling on the right path. *Nature Biotechnology*, 18, 2000.

Short article on metabolite profiling and its prospects.

[31] N. Goodman. Biological data becomes computer literate: new advances in bioinformatics. *Current Opinion in Biotechnology*, 13, 2002.

[32] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13, 2000.

A new (statistical) approach for the feature extraction and selection problem is presented in this article.

[33] L.M. Jakt, L. Cao, K.S.E. Cheah, and D.K. Smith. Assessing clusters and motifs from gene expression data. *Genome Research*, 11, 2001.

[34] R. Jansen and M. Gerstein. Analysis of the yeast transcriptome with structural and functional categories: characterizing highly expressed proteins. *Nucleic Acids Research*, 28, 2000.

[35] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, and A.L. Barabasi. The large-scale organization of metabolic networks. 2000.

[36] T. Kohonen. The self-organizing map. *Neurocomputing*, 21, 1998.

An overview of the self-organizing map algorithm. It is a powerful analysis technique and may be usefull in the thesis project.

[37] W.J. Krzanowski. Orthogonal canonical variates for discrimination and classification. *Journal of Chemometrics*, 9, 1995.

[38] W.J. Krzanowski, P. Jonothan, W.V. McCarthy, and M.R. Thomas. Discriminant analysis with singular covariance matrices: methods and applications to spectroscopic data. *Applied Statistics*, 1, 1995.

[39] B. Lemieux, A. Aharoni, and M. Schena. Overview of dna chip technology. *Molecular Breeding*, 4, 1998.

[40] R.G. Lyons. *Understanding Digital Signal Processing.* Prentice Hall, 1997.

[41] E.M. Marcotte. Measuring the dynamics of the proteome. *Genomic Research*, 11, 2001.

[42] D. Meldrum. Automation for genomics, part one: preparation for sequencing. *Genome Research*, 10, 2000.

[43] D. Meldrum. Automation for genomics, part two: sequencers, microarrays and future trends. *Genome Research*, 10, 2000.

[44] M. Mitchell. *An Introduction to Genetic Algorithms.* The MIT Press, 1996.

[45] I. Paran and D. Zamir. Quantitative traits in plants: beyond the qtl. *Trends in Genetics*, 2003.

[46] P. Pesaresi, C. Varotto, E. Richly, J. Kurth, F. Salamini, and D. Leister. Functional genomics of *Arabidopsis* photosynthesis. *Plant Physiol. Biochem.*, 39, 2001.

[47] The R-Project. http://www.r-project.org.

[48] L.M. Raamsdonk, B. Teusink, D. Broadhurst, N. Zhang, A. Hayes, M.C. Walsh, J.A. Berden, K.M. Brindle, D.B. Kell, J.J. Rowland, H.V. Westerhof, K. van Dam, and S.G. Oliver. A functional genomics strategy that uses metabolome data to reveal the phenotype of silent mutations. *Nature Biotechnology*, 19, 2001.

Breakthrough article describing the analysis of a NMR spectrogram using PCA and DFA.

[49] T. Reil. Dynamics of gene expression in an artificial genome - implications for biological and artificial ontogeny. 1999.

[50] D.G. Stark R.O. Duda, P.E. Hart. *Pattern Classification (2nd edition).* Wiley Interscience, 2001.

[51] U. Roessner, A. Luedemann, D. Brust, O. Fiehn, T. Linke, L. Willmitzer, and A. R. Fernie. Metabolic profiling allows comprehensive phenotyping of genetically or environmentally modified plant systems. *The Plant Cell*, 13, 2001.

Another example of spectrogram data suitable for analysis in the thesis.

[52] J. Taylor, R.D. King, T. Altmann, and O. Fiehn. Application of metabolomics to plant genotype discrimination using statistics and machine learning. *Bioinformatics*, 18, 2002.

[53] B.H. ter Kuile and H.V. Westerhoff. Transcriptome meets metabolome: hierarchical and metabolic regulation of the glycolytic pathway. *Federation of European Biochemical Societies*, 2001.

Interesting article on the combination of modelling and experiments in metabolic analysis. Also interesting for its conclusions on different types of regulation.

[54] R. Thomson and Z.R. Yang. A novel basis function neural network. 2002.

[55] R.H. Waterston, E.S. Lander, and J.E. Sulston. On the sequencing of the human genome. *Proceedings of the National Academy of Sciences*, 99, 2002.

[56] V.I. Zabarovska, R.Z. Gizatullin, A.N. Al-Amin, R. Podowski, A.I. Protopopov, S.Löfdahl, C. Wahlestedt, G. Winberg, V.I. Kashuba, I. Ernberg, and E. Zabarovsky. A new approach to genome mapping and sequencing: slalom libraries. *Nucleic Acids Research*, 30, 2002.

[57] D. Zamir. Improving plant breeding with exotic genetic libraries. *Nature Reviews Genetics*, 2001.