Comparative Proteomics in the Absence of Tandem Mass Spectra

by

Bjorn L. M. Wielens

Submitted in partial fulfilment of the requirements
for the degree of Master of Science

at

Dalhousie University
Halifax, Nova Scotia
December 2013

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Mass spectrometry plays a significant role in many proteomics experiments owing to its ability to provide high quality, detailed data on complex samples containing proteins and/or their constituent peptides. As with any technology, the capabilities of mass spectrometers are constantly increasing to provide better resolution, faster data acquisition, and more accurate mass measurements. However, the existence and widespread use of previous-generation instruments is not negligible. While these instruments may not have the capabilities of their modern counterparts they are still able to collect useful experimental data, though their limitations can result in trade-offs between certain parameters such as resolution, sample run-time, and tandem MS experiments.

This work describes an alternative method of MS data analysis, dubbed Parallel Isotopic Tag Screening (PITS), which seeks to enable higher throughput and the collection of better quality data on such previous generation instruments. This is accomplished by altering the duty cycle of the instrument in favour of MS^1 scans for the purposes of quantitation, and reducing the collection of MS^2 data (for the identification of peptides) to a single MS run which is separate from the quantitation data. Detection of peptides in the mass chromatogram is achieved through the use of a stable-isotope labelling method (dimethyl labelling) to produce a pair of isotopic patterns in the raw MS data, where the signal from the “light” isotopic pattern corresponds to peptides from one sample, and the signal from the “heavy” isotopic pattern to a second sample. Through the fitting of reference isotopic patterns to the raw data, these peak pairs can be detected and quantified to determine the relative protein expression ratios between the two samples.

The algorithm was validated against the results of an established proteomics software component (SEQUEST) and was able to detect four times as many peptide pairs as the traditional tool. To demonstrate its ease of adaptability to different experimental methods and instrument configurations, the PITS algorithm was also shown to be successful in the quantitation of peptides for both low and high resolution data (*m/z* increments of 1/12 and 1/50, resp.), and for samples which contained a third isotopic label. Finally, the algorithm's ability to successfully detect differential protein expression was evaluated through the use of a sample containing proteins with light:heavy concentration ratios at 2:1 and 1:2, against a complex 1:1 background. PITS was able to detect groups of peptides that were present at each of the prepared concentration ratios above and below the background ratio. The viability of the algorithm in this range of situations demonstrates its potential for increasing both the throughput of proteomics experiments and the quality of the resulting data.

LIST OF ABBREVIATIONS USED

ΔCN	Delta Correlation Number
ASCII	American Standard Code for Information Interchange
BSA	Bovine Serum Albumin
DNA	Deoxy Ribonucleic Acid
DTT	Dithiothreitol
ESI	Electro-spray Ionization
HPLC	High-Performance Liquid Chromatography
IAA	Iodoacetamide
ICAT	Isotope-Coded Affinity Tagging
Inf	Infinity or infinite (MATLAB abbreviation)
iTRAQ	Isobaric Tag for Relative and Absolute Quantitation
LC	Liquid Chromatography
MALDI	Matrix-Assisted Laser Desorption Ionization
MS	Mass Spectrometer/Mass Spectrometry
MS/MS or MS^2	Tandem Mass spectrometry
NaN	Not-a-Number
PITS	Parallel Isotopic Tag Screening
RPLC	Reversed-phase liquid chromatography
SEQUEST	Protein sequence identification software using MS/MS scans.
SHA	Secure Hash Algorithm
SILAC	Stable-Isotope Labelling of Amino acids in Cell culture
SNR or S/N	Signal-to-noise ratio
TEAB	Triethylammonium bicarbonate
TFA	Trifluoroacetic acid
XML	Extensible Markup Language

ACKNOWLEDGEMENTS

I would like to thank all those who contributed to the creation of this work and the surrounding research, and made it possible. Notable contributions include the prior work (Joseph Boutilier for initial proof of concept and the *S. cerevisiae* data file, Mark Wall for preparing the 4 BSA samples and data files), experimental work in labelling methods and sample preparation for new experiments (Gemma Regan, summer research assistant), SEQUEST-related work and peak width code (Samantha Rudolph, summer research assistant), the members of the Doucette Group for the use of their lab (and assistance with instruments and procedures), and finally my supervisor, Dr. Peter Wentzell, for the opportunity to create this work.

CHAPTER 1 INTRODUCTION

The term “proteome” has its origins in literature by Wilkins *et al.*[1] in 1995. The term itself is a portmanteau of the words “protein” and “genome”. Analogous to the genome, the proteome is intended to encompass the entire protein complement of an organism. However, in contrast to the static genome of a given organism, and despite originating from the same, the proteome of an organism is dynamic in nature and there is no 1:1 relationship between the number of proteins and genes. This is because there are many biological processes involved in the transcription of DNA to messenger RNA (mRNA) and the translation of mRNA to proteins. As a consequence, there is no simple direct relationship between the expression of genes and the abundance of proteins.[2] Some genetic information is never translated into a protein, while other genes may give rise to several proteins, owing to post-translational modifications of the genetic information. Modern cell biology involves the study of genomics, transcriptomics and proteomics (as well as other “omics” fields), but the last is often regarded as the most challenging due to the number, diversity, and concentration range of proteins in an organism. Since proteins are ultimately the functional elements of a cell (catalysis, structure, signalling, etc.), the study of the proteome is of key importance.

Since its creation, the field of proteomics and its methodology have grown and improved significantly, but the definition by Anderson and Anderson[3] still holds true. In

1998 they defined proteomics as “the use of quantitative protein-level measurements of gene expression to characterize biological processes (*e.g.* disease processes and drug effects) and decipher the mechanisms of gene expression control”. They also note that, while ambitious to obtain, the knowledge of how the molecular regulation of a biological system functions is far more powerful than simply mapping proteins to the genome. Thus, the ultimate goal in proteomics is to understand the role that each protein plays and how it interacts with other proteins and molecules within the living system. There are also practical implications to this understanding from a medical perspective, and the search for protein biomarkers provides a strong motivation for the field of proteomics.[4],[5] The identification of proteins that are uniquely associated with a disease state allows for a better understanding of a disease, and potential drug targets to be found.

While proteomics can be broadly described to encompass the characterization (structure, properties, abundance, etc.) of any protein in a living system, a major emphasis of modern proteomics is the large-scale, system-wide description of the proteome in the study of systems biology. From a bio-analytical perspective, this means the use of high throughput biological methods to measure the expression of many proteins in a living system. Rather than static measurements, a better understanding of the roles and relationships of proteins is gained by examining *changes* in protein expression between two or more states. These states could consist of organisms in healthy and diseased conditions, or exposed to different stimuli (growing conditions, heat shock, etc.), or simply different time points in a longitudinal study. The objective is to examine changes in protein expression among the different states to gain a better understanding of the biological system. This is often referred to as “comparative proteomics” and has the goals of identifying the proteins that are differentially expressed (up- or down-regulated) and providing explanations for the observed behavior.

The objective of this research has been to develop methods to improve capabilities in comparative proteomics. The remainder of this chapter presents a brief overview of the most relevant high-throughput mass-spectrometry based methods currently used and outlines how the proposed method integrates with this framework.

1.1 Quantitative Proteomics

From an analytical perspective, the ideal methodology in proteomics would provide unambiguous identities and accurate concentrations for all of the proteins (including post-translational modifications and different isoforms) that make up the proteome of the sample. Since biological samples are complex and highly variable, the ideal methodology should also allow for rapid processing of many replicates. However, the realization of this ideal is not complete, and an insufficient number of samples is often cited as a reason for inconsistent results in the search for protein biomarkers.[6],[7],[8]

These ideal objectives often run into conflict with one another, so compromises are generally made. Proteins can exhibit a wide range of properties (e.g. size, hydrophobicity, charge), and so the extraction and purification of proteins from an organism is generally a complex process involving many steps. Solubilization can be difficult, especially for hydrophobic proteins such as membrane proteins, and the complexity of the proteome can mean that a variety of fractionation steps may be needed to simplify the sample prior to further analysis. These steps can include gel electrophoresis and various types of chromatography (e.g., ion-exchange, reversed-phase), and may be applied to the proteins themselves, or the peptide fragments obtained by protein digestion during sample work-up. These procedures may result a single sample producing multiple fractions for analysis. The final step in the analysis generally involves some form of liquid chromatography coupled to mass-spectrometry (LC-MS) with electro-spray ionization (ESI), but other approaches such as matrix-assisted laser desorption ionization (MALDI) MS are also used. MS provides an almost ideal detection method for proteomics, since it provides high sensitivity, mass selectivity, a wide dynamic range and the ability identify peptides and proteins through high-resolution measurements and tandem MS methods. Therefore, MS has become the workhorse of modern proteomics and a wide variety of platforms are used.

There are generally two approaches to proteomics, which are termed “top-down” and “bottom-up” or “shotgun” proteomics.[9],[10] In top-down proteomics, the proteins extracted from the sample remain intact throughout the analysis, whereas bottom-up analysis requires that the proteins are digested into constituent peptides prior to analysis by MS. Bottom-up methods have the advantage that peptides are easier to analyze by chromatography and mass spectrometry since they are smaller and have more homogeneous properties. However, the number of peptides will be much greater than the number of proteins, and identification of the protein associated with a given peptide requires sequencing of the peptide and comparison with a protein database. Furthermore, information on post-translational modifications is obscured. On the other hand, top-down proteomics provides more direct information on the protein, but the size and diverse protein properties make the chromatography and MS more challenging. Applications involving bottom-up proteomics are currently more widespread, but advances in top-down proteomics continue to improve the methodology. This present work will focus on bottom-up proteomics.

Regardless of the approach used, it is still necessary to obtain some measure of protein expression under a given set of conditions. Absolute quantitation of protein concentrations in an organism would be ideal, but is difficult as it necessitates the use of internal standards in a reproducible fashion. These internal standards are typically peptides with isotopic substitutions.[11],[12] This type of quantitation can be useful in circumstances such as cellular modelling, where knowledge of absolute concentrations of proteins is important, *e.g.* direct comparisons across organisms for metabolic pathway studies, but the determination of these concentrations poses an additional challenge. In many cases, the knowledge of the *relative* changes in protein concentration is sufficient to determine protein significance in relation to a biological condition, as the absolute concentration of a protein can have little correlation with its importance in the biological process (such as signalling proteins, which can trigger significant changes even at low concentrations). Within the context of a given biological system, it is therefore generally more practical to determine the change in protein expression for a “test” state (*e.g.* stressed, diseased, or after a fixed time) in relation to a “reference” or “control” state (normal, healthy, time zero). These relative measures are what give rise to quantitative proteomics.

The central thesis of this research is that the throughput and quantitation of current methods in comparative, bottom-up proteomics can be improved through the use of the methods employed herein. These proposed methods involve the analysis of MS data only, and although tandem scans may be available, they are not used for quantitation nor the location of peptide peaks. Instead, they are only used for peptide and protein identification. The remainder of this chapter will consist of an overview of current methods in comparative bottom-up proteomics, followed by a brief description of the proposed method and its perceived advantages.

1.2 Comparative Proteomics

While there are many methods that have been proposed and used in comparative proteomics, a comprehensive review of these methods is beyond the intended scope of this document. Most of the common methods can be separated into one of several categories, although specific variations are common. All methods start with the extraction of the proteins in at least two biological states to be analyzed, with the objective of comparing individual protein expression levels under those conditions. For simplicity, the discussion that follows will limit this to two states, termed “test” and “reference.” Sample preparation will typically include digestion of the protein into peptides (typically using trypsin), and a series of purification and separation steps. This is then followed by further cleanup and separation, potentially with multiple dimensions and/or fractions, and ultimately lead to detection of the peptides by mass spectrometry. The separation and digestion steps can vary considerably, depending on the nature of the sample, but are generally critical to its success. Ergo, the focus of this work is on the final separation and MS detection of the peptides.

The principal methods currently used to identify and quantify the proteins and peptides in comparative bottom-up proteomics can be classified in a number of ways. One classification is according to whether the method uses labels, or is “label-free”. When labelling is used, different chemical tags are added to the digested peptides for the test and reference samples, and the samples are combined prior to LC-MS analysis. The tags used allow for distinction of the mass spectra of the two samples, so that the relative concentrations of peptides can be determined within the same analysis. These methods have the drawback of the inconvenience, expense and complications of adding a label to the sample, but also have the advantage of fewer experiments and less susceptibility to variations between runs. For label-free methods, which include spectral counting and feature extraction, the digested peptides are analyzed directly without the addition of chemical tags. This can simplify the analysis, but the test and reference samples must be run separately, which can lead to issues of variability in retention time and mass spectral sensitivity.

Another distinction that can be made is whether or not the methods require the use of tandem mass spectrometry data, as modern proteomics makes extensive use of tandem mass spectrometry to identify peptides and proteins present. In a typical LC-MS analysis, this occurs as follows: as compounds elute from the column, the mass spectrometer will constantly cycle to obtain ESI mass spectra (referred to as MS^1 data). Peaks detected within the mass spectrum may correspond to peptides, or originate from other molecules that are not of interest. At the end of each MS^1 scan, algorithms built into the mass spectrometer determine whether peaks warranting further examination by tandem mass spectrometry are present. This algorithm is complex and considers a number of configurable factors in determining how many and which peaks to examine, such as the size of the signal, the charge state of the ion, and whether or not the same peak has been recently examined. On the instrument used in this work, this process is also configured to include a "zoom" scan, which is a localized higher resolution scan to assess candidate peaks in the MS^1 spectrum. For each of the peaks chosen from the MS^1 scan, the corresponding ion is selected and fragmented to provide the MS/MS data. These additional scans are stored and can later be used to determine if the selected ion is a peptide, and if so, to attempt to identify its amino acid sequence (discussed shortly). The number of tandem scans carried out in this manner is determined by the instrument configuration, and after these scans are complete, the instrument returns to the MS^1 scanning mode and the entire process is repeated.

Peptides, and their parent proteins, can be identified from MS/MS data in a variety of ways, including *de novo* sequencing from the fragmentation pattern, protein database matching, or hybrid methods[13], but database matching is the most common approach and will be described briefly here. Candidate peptides can be identified from the protein database based on parameters such as the mass of the parent ion and the digestion protocol. For each of these candidates, hypothetical fragmentation patterns based on the amino acid sequence can be computed and compared with the experimental pattern observed in the MS/MS data. If, according to a variety of criteria, a good match is obtained, then the peptide is considered to be identified. If the peptide sequence is sufficiently unique, then it can be assigned to the corresponding protein in the database. However, this process is not without pitfalls. First, a number of parameters for the matching algorithm must be optimized to ensure an acceptable level of false positives and false negatives for peptide detection[14]. Second, criteria must be established to rank acceptable matches within a number of candidates, and it is possible that the selected sequence is incorrect. Third, protein databases are continually evolving and protein identifiers may not be consistent across different sources or over time. Finally, it should be noted that the unambiguous association of a peptide with a protein in the database requires a sequence of sufficient length to be unique. Despite these limitations, protein identification through MS/MS data is generally quite reliable.

An important consideration in this work is the “duty cycle” of the mass spectrometer, which can be defined as the fraction of time that the instrument spends performing MS^1 scans. When the MS^1 spectra are plotted as a function of time, the result is a mass chromatogram (intensity *vs*. *m/z vs*. time). To obtain reliable quantitative information, the sampling interval along the time dimension must be sufficiently small to adequately capture chromatographic elution profiles. If the number of tandem MS scans between MS^1 scans is increased, the sampling interval is also increased, and under-sampling of the chromatogram may result in poor quality data. On the other hand, if the number of tandem scans is reduced, the sampling interval is decreased, but important peptides in the mass chromatogram may not be located and identified by MS/MS. Additionally, the sampling interval can be considerably variable over time, as not all MS^1 scans will contain the maximum number of signals meeting the criteria for tandem MS analysis. On the instrument used in this work, sampling intervals ranging from 1.4 up to 9.0 seconds have been observed in the same data file. The highest sampling frequency can be achieved if tandem MS scans are eliminated altogether, corresponding to a 100% duty cycle. A higher sampling rate has implications for throughput as well. As narrower chromatographic elution profiles can be tolerated with a reduced MS^1 sampling interval, shorter chromatographic runs and/or higher resolution separation methods can be employed, resulting in reduced analysis time and improved peak capacity. Furthermore, if MS/MS scans are eliminated while retaining the same sampling rate at 100% duty cycle, higher resolution MS^1 data can be obtained, again improving the quality of the data. Thus, while tandem MS scans are needed for peptide identification, their elimination can improve throughput, peak capacity and/or mass spectral resolution, providing advantages to methods that do not require tandem MS data.

The remainder of this section focuses on several general methods that are used for comparative proteomics. The first two of these are based on isotopic labelling, while the last two are label-free methods. All of the methods, except for the last, require tandem MS data.

1.2.1 Stable Isotope Labelling

Stable-isotope labelling refers to a method in which the samples in the different states being analyzed are labelled with isotopic tags. In the case of two states, one will be

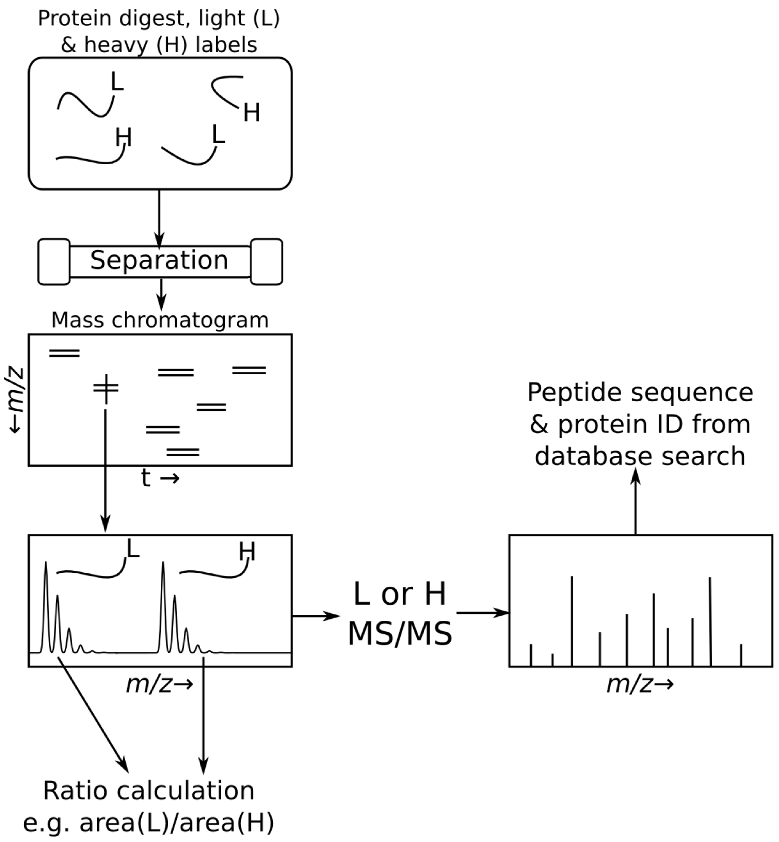


Figure 1: An overview of the steps involved in stable isotope labelling

labelled with a “light” tag, and the other state with a “heavy” tag. There are various strategies for the incorporation of these labels, and several types of labels from which to choose.[15],[16],[17],[18],[19],[20] Despite these options, the underlying strategies employed tend to remain the same, and an overview is illustrated in Figure 1.

Generally, the mixture containing the differentially labelled peptides is subjected to chromatographic separation, followed by introduction into a mass spectrometer to produce a mass chromatogram. During the MS run, peaks which meet the configured tandem MS scanning criteria are subjected to tandem MS analysis, as described previously. The data are then subjected to post-processing in which the tandem MS spectra and a protein database are used to determine the peptide sequence and identify the protein to which the sequence belongs. This is done using a software program such as SEQUEST,[21] MASCOT,[22] or TANDEM,[23] which can also integrate labelling information to provide information on the number and location of the light or heavy tags incorporated to the peptide. Once a peptide sequence has been identified in this fashion, the corresponding light or heavy peak can be located within the mass chromatogram and the intensity ratios calculated based on the MS^1 data.

There are several drawbacks to this approach, both instrumental and methodological, as has been noted in the previous section. First, the identification of a protein is not a standardized procedure, as there are a myriad of adjustable matching parameters which must be set, and a variety of databases to choose from. Furthermore, the duty cycle of the MS instrument is split between preliminary (MS^1), and tandem (MS^2) scans, such that it is only possible to gather a limited number of tandem scans for any given MS^1 sampling rate. While it is possible to increase either the sampling rate or the number of tandem scans, the other will be negatively impacted as either a degradation in the quality of the chromatographic data, or a reduction in the number of peptide peaks that can be detected and identified.

1.2.2 Isobaric Tagging

Isobaric tagging methods such as iTRAQ (developed by Ross *et al.*[24]) or Tandem Mass Tags (Thompson *et al.*[20]) are a special type of stable-isotope labelling used to achieve relative quantitation of proteins. The difference arises in how the tags themselves are designed. All of the tags employed have the same nominal mass, but contain different combinations of isotopes. An overview of the strategy is shown in Figure 2. Note that the order of the steps has changed somewhat from the previous example. The labelling step is as before, but incorporating the isobaric labels. Once again, mass peaks are selected for tandem MS scans, but in this method the peaks are not paired. In this case, a single MS^1 peak contains both the test and reference states for the peptide, but these two states give rise to different fragment ions ("reporter ions") in the MS/MS spectra. The fragment ion intensities in the tandem MS scans are used to determine differences in the intensity ratios of the samples. The tandem MS scans are also post-processed in a similar fashion as before.

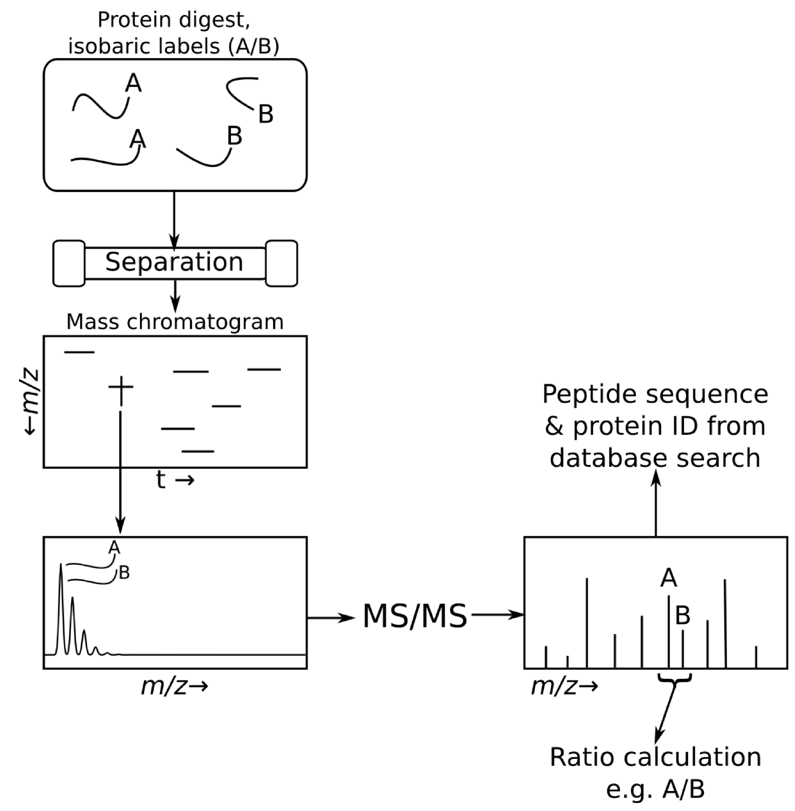


Figure 2: An overview of labelling using isobaric tagging methods

This approach resolves the necessity of locating both the light and heavy peaks in the sample as they occur simultaneously in the mass and time dimensions. The problems of database search parameters, duty cycle, and reagent cost still remain. Furthermore, one is limited to using only tandem MS scans to quantify the protein ratios, and this scan may not occur at the maximum of the peak itself. Integration of peak area for distinct labels is not possible either, as all tag variants are contained within the same mass peak. Thus, the calculated ratio may be less precise than that obtained by using multiple time channels in the MS^1 data.

1.2.3 Spectral Counting

Spectral counting,[25],[26] first validated by Liu *et al.,*[27] is a significant departure from the workflows described previously. The test and reference samples are processed

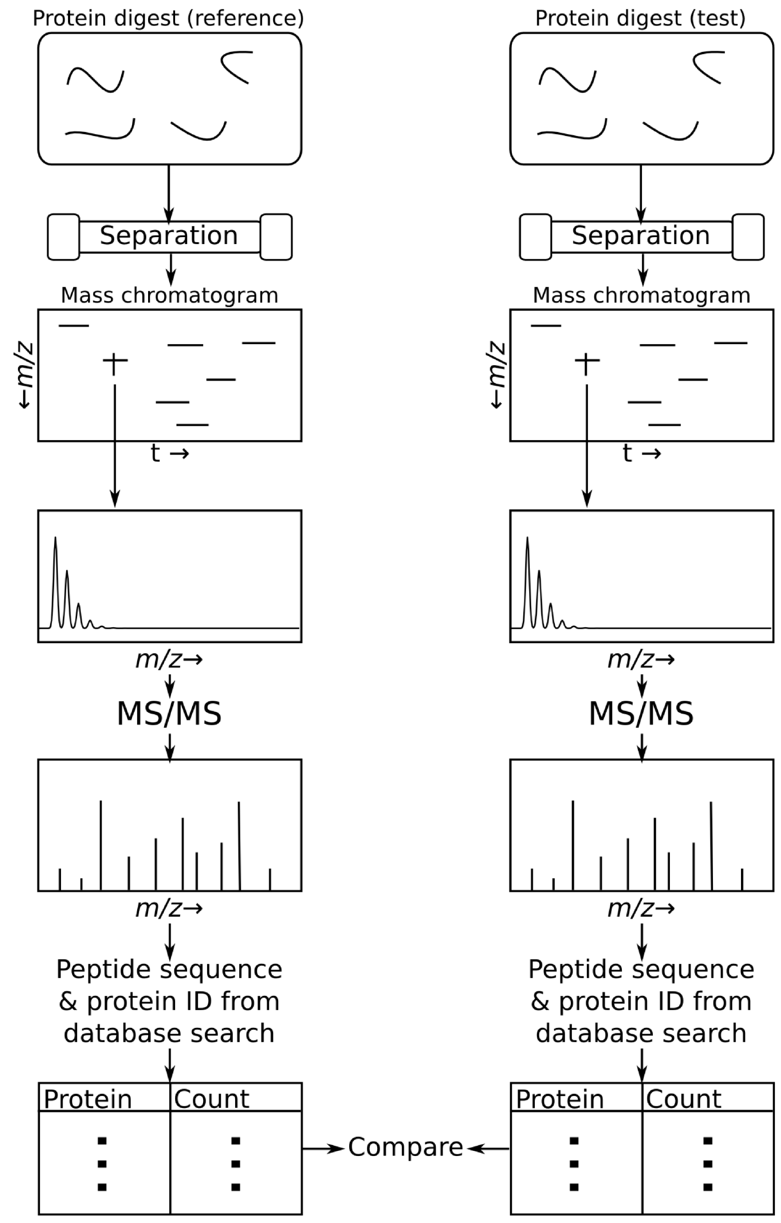


Figure 3: An overview of the spectral counting workflow.

separately, and therefore no labels are necessary to distinguish them. As before, peaks of interest are scanned using tandem MS, and the proteins identified from the composition of peptides detected. It is then assumed that the number of times a protein is detected from its constituent peptides is related to the amount of that protein present in the sample. Comparison of the number of counts of a protein in both the test and reference samples is used to establish the relative ratio and determine whether the protein is up-regulated or down-regulated. This is illustrated in Figure 3.

As no labels are required, this method is less expensive and procedurally less complex, but it is again limited in its reliance on tandem MS data and the correct identification of a peak. It is entirely possible that a peak is present but not detected, and because the test and reference samples are processed separately, it is imperative that the conditions of analysis remain consistent across all samples. Finally, the relationship between detection counts and the amount of protein present is likely to be non-linear and highly variable, since the detection of one protein is affected by the presence of other proteins in a complex sample[28].

1.2.4 Feature Extraction Methods

Feature extraction methods represent a substantial class of techniques for comparative proteomics that are characterized by their direct use of information from the mass chromatogram to obtain relative measures of protein expression.[29],[30],[31],[32] Although specific implementations of this approach differ in their algorithmic details, the general strategy is essentially the same and illustrated in Figure 4. As these methods do not make direct use of MS/MS data (unlike the other methods discussed so far), the issues surrounding protein identification and under-sampling due to mass spectrometer duty cycle are removed, and simpler instruments can, in principle, be employed to run the experiments. In addition, these methods are generally label-free approaches, further simplifying their implementation.

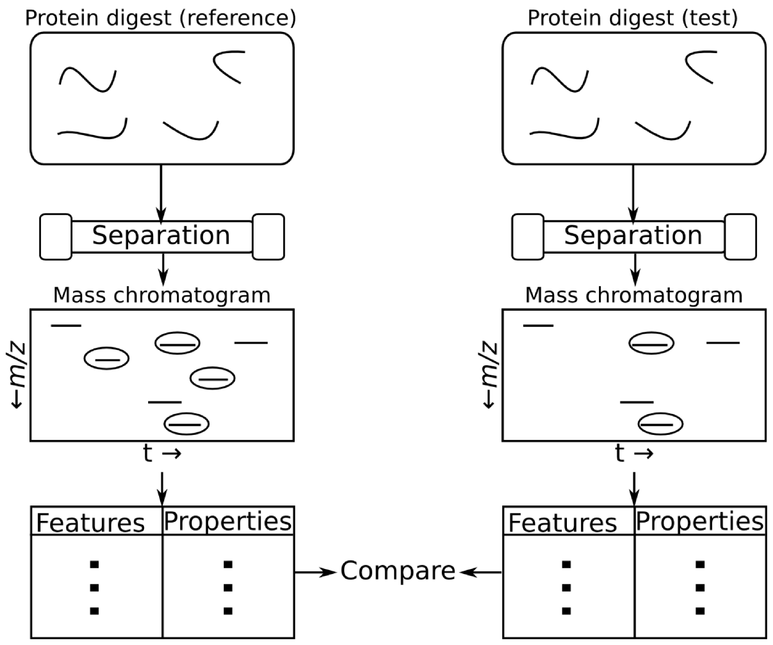


Figure 4: Overview of feature detection process.

Feature extraction methods can be considered to derive from earlier approaches to peptide finger-printing that existed even before mass spectrometry was routinely available.[33],[34],[35] In these approaches, the proteins in the sample would first be separated, typically using a method such as two-dimensional gel electrophoresis. Differences in the patterns observed for the test and reference samples could then be noted and proteins of interest could be extracted for further investigation and identification. This identification could be performed by traditional biochemical methods or, more recently, using mass spectrometry. This approach focuses on the identification of only those proteins where differences are observed and ignores the majority that exhibit no differential expression. In contrast, most strategies in MS-based proteomics seek to identify as many proteins as possible and then search for differential expression.

The transformation of traditional peptide finger-printing to MS based platforms involves replacing the 2D gels with the mass chromatograms and detecting differences in intensities of peptide signals (and thereby the expression levels of the parent proteins). Two elements are key to this process. First, chromatographic features within the mass chromatogram associated with the peptides must be identified. Since no tandem MS information is assumed to be available, association of peaks with a particular peptide is not possible and the location of peptides (or possible peptides) must employ other methods. This is a complex procedure, since signals within the mass chromatogram arising from peptides must be distinguished from background noise and irrelevant signals that may be present. A variety of strategies have been developed for this purpose in different software packages. Generally, however, algorithms employ a stepwise procedure involving signal smoothing, background removal, peak detection, detection of isotopic patterns, and conditional acceptance criteria. At the end of this process, the features identified form a map of typically hundreds of mass/time windows which contain the peaks of interest. The peaks within these windows can then be quantified through intensity or area measurement.

To detect differential expression through the peptides, it is necessary to compare the mass chromatograms of the test and reference samples. Therefore, a second key element of the process is mapping the features extracted from one mass chromatogram to another, which is also not trivial. Although the mass registration for peptide peaks detected should be fairly consistent, chromatographic variability, especially over extended periods of time, can lead to significant differences in retention times. Therefore, alignment of the mass chromatograms in the time direction is generally necessary before the two sets of results can be compared. Post-alignment, an algorithm with specified matching criteria can be used to pair the features in each data set and calculate ratios. Of course, issues with missing features or incorrect matching are still possible.

Once expression ratios have been obtained, the data can be analyzed to identify peptide peaks that exhibit differential expression or interesting behavior. In the original peptide mapping experiments, fairly simple methods, such as visual inspection, were typically employed to identify interesting features. As technology improved, it became possible to use more advanced chemometric tools, such as exploratory data analysis, to distinguish the more subtle aspects of a data set. At this point, one or more experiments can be carried out to target the peptides of interest for MS/MS identification. This way, time is not spent identifying peptides and proteins that are not relevant to the system under study.

The experimental simplicity of feature extraction methods, which are label-free methods that require only mass chromatograms, is their most attractive characteristic. In addition, the 100% duty cycle has the advantages of higher throughput, better mass resolution and higher peptide detection rates already noted. However, these methods are not without drawbacks. The algorithmic complexity of locating features and aligning them across multiple experiments can be a complicating factor, and these methods work best with high resolution MS data. From an analytical standpoint, it is also often more appealing to have features that are associated with an identified protein from MS/MS data from the beginning, rather than trying to discover these at the end of the experiment. As there are variations in chromatographic behavior and MS sensitivity across multiple experiments, the calculated expression ratios are also likely to be more variable than those obtained from a method based on isotopic labelling, and normalization of ratios may be required. It is possible to use feature extraction methods in conjunction with isotopic labelling experiments, but generally the isotopic pairs are identified after their extraction as individual features. It is this aspect that is addressed in the present work.

1.3 Proposed Method for Quantitation

The method proposed in this work strives to combine elements of stable isotope labelling and feature extraction methods to exploit the advantages of both. This new method, referred to as “Parallel Isotopic Tag Screening” (PITS), was originally investigated by J. Boutilier[36] in this research group. While this original work established a proof-of-principle for the underlying concept, many of the algorithms developed were unrefined, resulting in sub-optimal performance with regards to both efficiency and results. Furthermore, the original work did not fully validate the method for detection of differential expression, nor its full potential under different experimental conditions. The present work represents the author's contribution to address some of these issues in an ongoing effort to establish PITS as a useful tool for large-scale proteomics studies.

The advantages and drawbacks surrounding feature extraction methods were discussed in the previous section. In contrast to other methods discussed, feature extraction methods represent a fundamental shift in the strategy for comparative proteomics experiments, as represented in Figure 5.

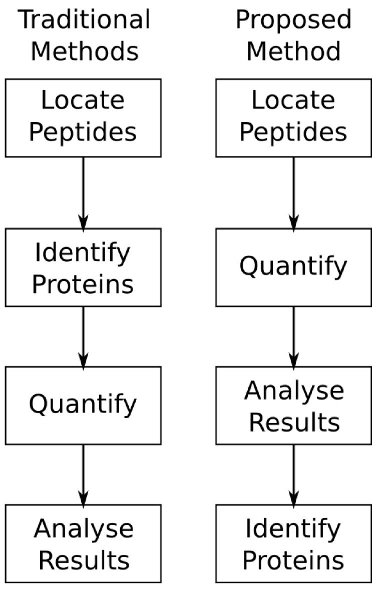


Figure 5: An overview of the order of steps in traditional methods *vs* the proposed method.

Whereas the sequence used in most methods is to locate the peptides, identify the peptides and proteins, quantify the relative expression levels and analyze the results for interesting behavior, feature extraction methods move the peptide/protein identification step to the end, making this a separate experiment. In doing so, the need for tandem MS scans is eliminated and the instrument can operate in MS-only mode. As already noted, there are numerous potential advantages to this that include the detection of a larger number of peptide features, greater throughput, higher mass resolution and less restrictive instrument requirements.

Perhaps the biggest drawbacks of feature extraction methods are the complexity of identifying legitimate features in the mass chromatogram and the variability associated with calculating expression ratios across multiple experiments. It is the premise of the work presented here that both of these issues can be mitigated by combining the principles of feature extraction to stable isotope labelling experiments. To address the first drawback of feature identification, the PITS algorithm simultaneously models the isotopic patterns of both light and heavy peptides in the pair to provide greater confidence in establishing the location of peptides. This is in contrast to other feature extraction methods that model isotopic patterns for individual ions, making it difficult to distinguish peptides from other contaminants or spurious signals. Additionally, this approach allows the detection of overlapping peptide pairs with relatively small separation in isotopic mass, a situation which is difficult to accommodate with conventional feature extraction methods. Such modelling also leads to direct quantitative assessment of expression ratios between the light and heavy peptides (test and reference), reducing the problem of quantitative variability across experiments.

It has been the goal of this research to improve the algorithms associated with the PITS methodology and to demonstrate their efficacy in a variety of situations. The intent is to be able to apply the method to mass chromatograms from stable isotope labelling experiments and to make it adaptable to a variety of protocols, but the target data are experiments with relatively low mass resolution. It may be argued that, as instrumentation continues to improve, providing higher resolution and faster scanning rates, issues surrounding the duty cycle and the collection of tandem MS data will become less relevant and algorithms based on the analysis of MS^1 data will become obsolete. However, such instruments, which are costly and of limited availability, will remain a bottleneck in the proteomics workflow and restrict higher throughput of samples. A trend in modern high throughput analysis, as evidenced by the human genome project, is towards parallel analysis using multiple, less expensive instruments. With the increased miniaturization of analytical instruments, including mass spectrometers,[37] there is likely to continue to be a role for the analysis of low resolution MS^1 data.

To demonstrate the PITS methodology, this research employs dimethyl labelling[38],[39], which uses isotopic variants of formaldehyde to differentiate the masses of the test and reference samples. The advantages of this method are that it is less expensive than commercial methods, and the labelling procedure consists of straightforward addition of reagents and reasonably short incubation times. During the process of dimethyl labelling, the digested peptides are labelled by dimethylation of any primary amines present in the peptide chain. Such primary amines are found at both the N-terminus of the peptide chain and also in the side chain of each lysine residue. The mechanism of this reaction is via a reductive amination pathway, and the overall reaction is shown in Figure 6.

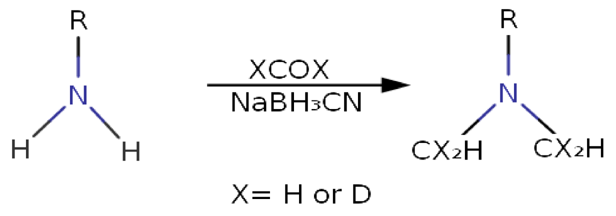


Figure 6: Reductive amination reaction which methylates primary amines.

The use of deuterium labels has minimal effects of retention time shifts in the chromatographic domain[40]. It is even possible to expand the number of possible labels through the additional use of deuterated sodium cyanoborohydride (replacing the remaining H atom in the added methyl groups) and/or ^{13}C deuterated formaldehyde (replacing the ^{12}C of the methyl groups with ^{13}C). Combinations of these reagents can produce labels with offsets of two, four, six, and eight mass units above the “light” label. Use of labels differing by only two mass units will result in overlap of signals within an isotopic cluster. While this work only utilizes labels with mass differences of 4 mass units, the proposed method will still work in situations with such an overlap. This is because the computed models would incorporate the overlap and de-convolute the contributions of each isotopic pattern during the model-fitting calculations.

As mentioned, tandem MS scans are not necessary for the quantitation of a detected peptide pair (though they may be present). This results in the possibility of detecting a greater number of peptide pairs, and allows the tailoring of the instrument configuration to match the intent of the experiment. Eliminating tandem MS scans improves the duty cycle of the instrument, which allows for the choice of more full MS scans for better quantitation, higher resolution for better detection, and/or faster separations for better throughput.

Of course, a disadvantage of any feature extraction method based on MS^1 data is that tandem MS is still necessary to provide identification of a peptide, but it is possible to collect this information in a few post-analysis experiments and cross-reference them to the quantified data, should a signal of interest be observed. Also, the location of peptides by the PITS method is susceptible to interferences that affect the isotopic pattern, unlike methods that identify peptides on the basis of tandem MS scans. However, quantitation of the expression ratios based on MS^1 data will be an issue for both approaches in such situations.

A further challenge in implementing the PITS algorithm on samples with only two isotopic labels occurs when one of the two isotopic patterns is completely absent. In this situation, the PITS algorithm is unable to detect a pair of peaks for a peptide which may be of high interest to the researcher. This is because the double-label method relies on the presence of both of the isotopic patterns within the spectral data, and the second isotopic pattern in the pair serves both as a reference for detection of peptides against other components in the sample, and as the signal to be quantified. The dual purpose of this signal can be eliminated through the use of a third isotopic label. One such experimental design could have two of the isotopic patterns correspond to reference sample, but labelled using two different labels. These patterns would then always be present to serve as a reference for the location of signals. The third isotopic pattern would correspond to the test sample, labelled using a third variant of the isotopic label. Should this signal be completely absent, it is still possible to detect the reference signals and make note of the absence of this third signal.

1.4 Thesis Outline

The theory and principles of the proposed method outlined above may seem straightforward, but there are substantial technical challenges which must be overcome during the implementation stages. Particularly, establishing the relationships between the detections at individual points on the mass chromatogram, processing time of the algorithm on larger data sets, and integration of SEQUEST results. The remainder of this work describes the re-written algorithms for data import and peptide detection, which reduced the performance bottleneck, the newly-developed clustering algorithm for grouping individual detections into a signal which persists over chromatographic domain, and the code to import SEQUEST detections and cross-reference them with PITS detections. It should be noted that while this work builds on the previous work of Joseph Boutilier,[36] the original implementations of those algorithms have also been re-written to provide additional features, performance, and flexibility.

The following chapter (Chapter 2) describes the experimental details of the methods used in the preparation of samples, including the digestion process, labelling procedure, and the instruments used. Chapter 3 covers the workflow and specific algorithms as implemented for each of the key stages of the workflow. Chapter 4 describes the results and validation of these algorithms on a variety of protein samples, and applications to higher-resolution data, and the use of multiple labels, with preliminary experiments that demonstrate this functionality. Finally, the overall conclusions are discussed in Chapter 5, along with in-progress and future work which would further validate the PITS method.

CHAPTER 2 METHODS AND EXPERIMENTS

2.1 Chemicals Used

The following materials were used in the digestion and labelling of the protein samples in this work. Catalogue numbers are provided in brackets after the chemical name. Ammonium bicarbonate (A1641), iodoacetamide (11149), trifluoroacetic acid (T5608), Tris (T1378-IKG), trypsin (T8802), triethylammonium bicarbonate (T7408), formaldehyde 37 wt. % in H_2O (as formalin, F1635), deuterated formaldehyde 20 wt. % in D_2O (49-262-0), 13C deuterated formaldehyde 20 wt. % in D_2O (5-96388), sodium cyanoborohydride (15615-9) and sodium cyanoborodeuteride (190020) were obtained from Sigma-Aldrich (St. Louis, MI, USA), dithiothreitol (161-0611) from Bio-Rad, (Hercules, CA, USA), formic acid (A119P-4) from Fisher Scientific, (Waltham, MA, USA).

Additional HPLC grade reagents were formic acid (94318-50ML) from Sigma-Aldrich (St. Louis, MI, USA) and acetonitrile (A996-4) from Fisher Scientific (Waltham, MA, USA).

Protein standards used were as follows: Bovine Serum Albumin (BSA) (A-8022), and Lysozyme C (L-6878 Sigma) from Sigma-Aldrich (St. Louis, MI, USA). *Escherichia coli* was grown as a culture per established methods[41], and was a gift from Dr. Andrew Roger of the Department of Biochemistry and Molecular Biology at Dalhousie University. All water used in sample preparation and HPLC was filtered on a Milli-Q purifier (Millipore EMD, Billerica, MA, USA) and purified to 18 MΩ cm, courtesy of the laboratory of Peng

Zhang in the Department of Chemistry at Dalhousie University.

2.2 Tryptic Digest

In preparation for tryptic digest as published by Wall *et al.*[14], the proteins to be digested (either BSA, lysozyme C, or the *E. coli* proteome) were portioned into 200 µg aliquots, either by dilution from frozen stock to 1 mg/mL protein in 50 mM Tris buffer (pH 7.4), or by weighing approximately 1 mg of solid protein standard and dissolving in equivalent volume of 50 mM Tris buffer (pH 7.4) to working concentration of 1 mg/mL. Dithiothreitol (DTT) was added to each sample to a final concentration of 9.5 mM, from a 200 mM stock (4.75 µL / 100 µL of sample). The samples were incubated at 60 °C for 20 min in a water bath, after which iodoacetamide (IAA) was added to a concentration of 19 mM (10.5 µL of 200 mM stock per 100 µL of sample). The samples were placed in the dark at room temperature for 20 min, after which 4 µg of trypsin (from 1 mg/mL stock in 1×10^{-4} M HCl) was added to the sample. The samples were then left to incubate at 37 °C overnight. Finally, 24 µL (10% of the total sample volume) of 10% TFA (by volume in H_2O) was added to each sample to terminate the reaction. The samples were evaporated to dryness in a Speedvac (Thermo Fisher Scientific, Waltham, MA, USA) rotary concentrator and re-suspended (by actuating with a micropipette 40 times, followed by vortexing of the sample) in 200 µL of water with 0.1% TFA, in preparation for cleanup as described in section 2.3.

2.3 Post-digestion Cleanup

After digestion, the samples were subjected to reversed-phase liquid chromatography

(RPLC) cleanup to remove leftover reagents, especially ammonium bicarbonate which interferes with the labelling process. Cleanups were run on a 1200 series Agilent (Santa Clara, CA, USA) HPLC system with a G1379B degasser, 1376A pump, G1313A auto-sampler, G1315B diode array detector, and G1364D fraction collector. Solvents used were nano-filtered water with 0.1% TFA (Solvent A) and acetonitrile with 0.1% TFA (Solvent B). The flow rate was set at 200 µL/min with 10 µL injections (10 µg protein/injection). The solvent gradient timing is as follows: 0 min to 5 min at 5% B, increasing to 95% B at 10 min, and re-equilibrating at 5% B until the 40 min mark. Fractions were collected from 8.5 to 11 min, depositing three collected fractions per vial to produce 30 µg aliquots of cleaned protein. Collected fractions were dried down in the rotary concentrator in preparation for labelling. Samples were cleaned on a self-packed column (5 cm x 1mm inner diameter) with Waters (Milford, MA, USA) Spherisorb S5 ODS2 beads (part no. 820019).

2.4 Dimethyl Labelling

In preparation for dimethyl labelling,[39] the 30 µg aliquots that were previously digested, cleaned, and dried were re-suspended in 100 µL of triethylammonium bicarbonate (TEAB). A 4 µL aliquot of 4 wt. % formaldehyde in H_2O (either normal or deuterated as appropriate) was added to the sample. The samples were briefly vortexed and spun down in a centrifuge. A 4 µL aliquot of 0.6 M sodium cyanoborohydride (normal or deuterated per the experimental design) was added to each sample, and the samples were vortexed and left to incubate for two hours in a fume hood, with an additional vortex step after one hour. The remaining reagents were quenched by the addition of 16 µL of 1% *w/v* solution of ammonium bicarbonate, vortexed, and spun down in the centrifuge. Finally, the samples were placed in a freezer-chilled rack to prevent heating during the final quenching, which was completed by the addition of 8 µL of 5% formic acid. The samples were evaporated to dryness in preparation for a final RPLC cleanup.

2.5 Post-labelling RPLC Cleanup

The reversed-phase cleanup procedure after labelling is identical to that discussed previously (section 2.3) in the post-digestion cleanup, except that fractions were collected so as to aliquot 10 µg of protein per vial. The amount of protein in the fractions was estimated based on the chromatographic peaks, and used to recombine the light and heavy labelled samples at a 1:1 ratio prior to LC/MS analysis. Samples were again dried down in the rotary concentrator and placed in a freezer until use.

2.6 LC-MS/MS HPLC

The LC-MS/MS and HPLC configuration were shortened variants of the method published by Wall *et al.*[14] Solvent A consisted of nano-filtered water with 0.1% formic acid. Solvent B was acetonitrile with 0.1% formic acid. Protein samples to be analyzed were re-suspended in solvent A and combined in the light:heavy ratios as appropriate for the experiment (these samples and ratios are detailed subsequently in sections 2.7 to 2.9). The pump flow rate was set to 250 nL/min with a 10 µL injection amount (injecting a total of 1 µg protein). The solvent gradient was as follows: 0 min: 5% B; 0.1 min: 7.5% B; 45 min: 20% B; 57.5 min: 25% B; 60 min: 35% B; 61 min: 80% B; 64.9 min: 80% B;

65 min: 5% B. The instrument used was comprised of a Hewlett-Packard (Palo Alto, CA, USA) 1050 series pump, Agilent (Santa Clara, CA, USA) 1200 series G3113A auto-sampler, and G2226A nano-flow pump, coupled to a Thermo Finnigan (Waltham, MA, USA) LTQ mass spectrometer via nano-spray ESI. The separation column was self-packed (25 cm x 75 µm inner diameter) with 4 µm Jupiter (Phenomenex, Torrence, CA, USA) Proteo C_{12} beads, catalogue 00G-4396-E0).

2.7 Double-labelling Experiment

Doubly-labelled samples of BSA for the high-resolution data files were prepared as outlined in sections 2.2 through 2.6. Prior to LC-MS analysis, one 10 µg aliquot of BSA with the light isotopic label was combined with one 10 µg aliquot of BSA with the heavy isotopic label, resulting in a light:heavy ratio of 1:1.

2.8 Triple-labelling Experiment

The methods used to prepare a protein sample containing three isotopic labels are as in sections 2.2 through 2.6, with the following additions. During the addition of the labelling reagents (formaldehyde and sodium cyanoborohydride), one 30 µg aliquot of unlabelled BSA (“reference” sample) was labelled with normal formaldehyde and normal sodium cyanoborohydride, to produce an un-deuterated “light” label. A second 30 µg aliquot (representing the “test” sample) was labelled using deuterated formaldehyde and normal sodium cyanoborohydride to create an “intermediate” label which is 4 mass units heavier per label than the “light” variant. A third 30 µg aliquot (a duplicate of the “reference” sample) was labelled with both deuterated, ^{13}C formaldehyde and deuterated sodium cyanoborohydride to create the “heavy” label which was 8 mass units heavier per label than the light variant. These three labelled aliquots were combined prior to LC/MS analysis at a 1:1:1 ratio (based on the estimated protein concentrations) to produce a three-label sample.

2.9 Alternate Ratio Experiment

To prepare a sample containing a 1:1 “background” proteome and surrogate up and down-regulated proteins, four 10 µg aliquots of digested and labelled *E. coli* (two with light labels, two with heavy labels) were combined and brought up in 400 µL of nano-water with 0.1% TFA (LC/MS solvent A). Three 10 µg aliquots of digested and labelled BSA (two with heavy labels, and one with light labels) were combined and brought up in 300 µL of solvent A. Three 10 µg aliquots of digested and labelled lysozyme C (two with light labels, one with heavy labels) were combined and brought up in 300 µL of solvent A. The resulting lysozyme C and BSA mixtures were combined to produce a sample containing equal parts lysozyme C (as a surrogate “up-regulated” protein at a light:heavy ratio of 2:1), and BSA (as a surrogate “down-regulated” protein at a light:heavy ratio of 1:2). This mixture of proteins was then combined at three different ratios (1:10, 1:100 and 1:1000 in the lowest-concentration labelled protein in the mixture to *E. coli*) with the *E. coli* background. Each of these samples contained 100 µL of *E. coli* mixture, 60 µL of the protein mixture for the 1:10 sample, 6 µL of protein mixture for the 1:100 sample, and 6 µL of a 10-fold dilution of the protein mixture to yield a 1:1000 sample.

2.10 MS Acquisition and SEQUEST Search Parameters

The acquisition parameters for each of the experiments conducted above in sections 2.7,2.8, and 2.9 are provided in Appendix 1, Appendix 2, and Appendix 3, respectively. The SEQUEST search on the data files for the alternate ratio experiment (section 2.9) was performed by the undergraduates Gemma Regan and Samantha Rudolph, using Proteome Discoverer 1.3.0.339. Settings were left at defaults, excepting the following: Max. Precursor mass: 8000 Da, precursor mass tolerance: 2 Da. The protein database consisted of *E. coli* K12 and 12 standard proteins (including BSA and lysozyme C), with templates configured for proteins with 28 and 32 Da tags. Result filters were configured for peptides ranked with high confidence, a peptide maximum ΔCN score of 0.1, peptide rank of 4, false positive rate of 1% or less, 1 peptide per protein minimum, and the minimal scores for charge states of 1 thru 7, and >7 set to 1.5, 2.0, 3.25, 3.75, 2.75, 3.0, 3.2, and 3.4, respectively.

CHAPTER 3 WORKFLOW AND ALGORITHMS

3.1 OVERVIEW

This chapter describes the algorithms which were designed for the quantitation without tandem MS data. The order in which the algorithms are discussed is similar to the workflow that would be followed when performing the analysis of the data. A diagram of this workflow is shown below in Figure 7.

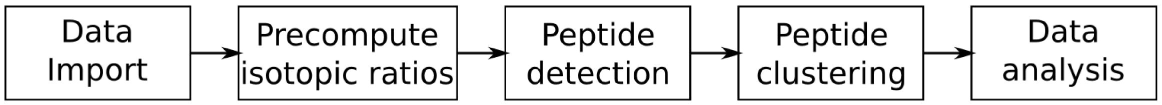


Figure 7: Overview of the workflow to analyze data with the proposed method

The process begins with the preparation and import of the data, such as converting the raw spectral data files from the instrument to a usable format. Subsequently, the isotopic ratios of the protein sample are pre-computed and modelled in preparation for use during the detection stage. Once the peptides have been detected in the mass chromatogram, the detections must be clustered, and appropriate data values recalculated based on the entire region over which the peptide signal was observed. Finally, the data is analyzed (and potentially cross-referenced with SEQUEST search results for additional validation).

3.2 DATA PREPARATION AND IMPORT

The files in which the data are stored must be converted from their native proprietary (.RAW) format produced by the XCALIBUR mass spectrometer software (Thermo Scientific, Waltham, MA, USA) to a different format to be imported into MATLAB (The

Mathworks Inc., Natick, MA, USA). As of this writing, no known direct conversion tools between these formats exist, so it was necessary to convert these files to an intermediate format prior to import into MATLAB. This issue of proprietary file formats has previously been addressed by Pedrioli *et al.*[42] through the design of a non-proprietary XML-based format (file extension .mzXML) which can then be parsed and stored as a MATLAB matrix. The conversion from RAW format to mzXML format can be done using a pre-written program named 'ReAdW' by the Seattle Proteome Center[43]. It is then possible to parse this mzXML file and import it into MATLAB by extracting the retention time, scan numbers, and Base64 encoded spectral data. The spectral data is stored as a series of paired values; the first value of a pair is the *m/z* value, the second value is the intensity value measured at the *m/z* value for the pair in question. The *m/z* range and interval of the MS scans is constant for the duration of the experiment.

To save space, these MS data sequences were reconstructed into a single matrix of size *m* by *n*, where *m* corresponds to the *m/z* dimension, and *n* represents the time dimension. Associated with this matrix are an *m*-by-1 vector of *m/z* values, and an *n*-by-1 vector of times, reducing an otherwise *m* by *n* vector of repeated *m/z* values to only a single vector of such values. The actual data import function may be seen in Appendix 4 and its key aspects are described in detail below. The conversion process of the Base64 encoded data to usable single precision values is a separate process and is discussed in a subsequent section.

3.2.1 mzXML Import Function

The function to import the mzXML files into MATLAB compatible data structures was originally written by Joseph Boutilier as part of his thesis[36]. The function included here is a completely re-written, optimized version which decreases processing time significantly from the original. Prior to optimization, the import of an mzXML file for a 90 minute MS run (approximately 400 megabytes in size) took approximately one hour to extract the MS data, or longer if the zoom scans and tandem MS data were also desired. After optimization, all three scan types for the same mzXML files can be imported in two to three minutes.

The mzXML format is an extensible markup language (XML) based layout, parts of which are human-readable and therefore easy to interpret by inspection. As a result of the highly-structured and defined nature of the mzXML markup language, it is also straightforward to parse and subsequently interpret using computer software.

The data contained in an XML file is delineated by a series of hierarchical tags which define instances of specific objects (e.g. a single MS scan in the case of mzXML) and a series of associated parameter-value pairs that describe the attributes of that specific instance. An example of such a tag contained in an mzXML file can be seen in Schema 1, below.

```
<parentFile fileName="Aon-1-2.RAW" fileType="RAWData"
fileSha1="ad72bffc9f55f3bbdefe6d55037f33bde5303a83" />
```

Schema 1: Example tag and its associated properties from an mzXML file.

The XML tag is denoted by the less-than symbol (<). The keyword immediately following it defines the type of the tag. The valid keywords depend upon the application for which the XML is designed – for this reason any XML usually has a publicly available format specification for reference. The specifications for various mzXML versions may be found (at the time of this writing) at http://sashimi.sourceforge.net/schema_revision/ as an XML schema document, and contains details of the types of valid tags and the corresponding children and key-value pairs which are considered legal within the XML.

This example tag is of the type “parentFile” and indicates (through the three property-value pairs) the original filename was “Aon-1-2.RAW”, the file type was “RAWData” and the SHA-1 checksum of the file in question (SHA-1 refers to the NIST-published method used to calculate the checksum data; these data allow one to validate that the contents of the source file have not been altered since the creation of this derivative file). The trailing slash and greater-than symbol (/>) indicate that this instance of the object has been fully described. As mentioned earlier, XML allows for a hierarchical structure, and in cases where a tag has child objects, the end of the parent object is denoted by a “closing” version of the same tag, which consists of the tag type, prefixed with a forward slash (/) and surrounded by greater-than and less-than symbols as before. An example tag may be seen in Schema 2.

```
  <msInstrument>
   <msManufacturer category="msManufacturer" value="Thermo
Scientific" />
   <msModel category="msModel" value="LTQ" />
   <msIonisation category="msIonisation" value="NSI" />
   <msMassAnalyzer category="msMassAnalyzer" value="ITMS" />
   <msDetector category="msDetector" value="unknown" />
   <software type="acquisition" name="Xcalibur" version="2.2" />
  </msInstrument>
```

Schema 2: An example of an mzXML tag with child objects, and its respective "closing" tag.

Here, the parent object (msInstrument) contains a series of child objects that describe the properties of the instrument on which the data file was created, such as the manufacturer, model, and ionization source, before being terminated by the closing msInstrument tag.

```
    <scan num="1" msLevel="1" peaksCount="15600" polarity="+" scanType="Full"
      filterLine="ITMS + p NSI Full ms [400.00-1700.00]"
retentionTime="PT0.5285S"  lowMz="400.083" highMz="1700"
basePeakMz="1358.42" basePeakIntensity="3009.96"
        totIonCurrent="1.05386e+007" >
              <peaks precision="32" byteOrder="network" contentType="m/z-int"
                            compressionType="none" compressedLen="0" >
                  %BASE64DATA%
             </peaks>
    </scan>
```

Schema 3: Sample mzXML code that is used to define a single (non-zoom) MS scan by the instrument. The Base64 data has been omitted for length.

The MS intensity data of interest is contained in a “scan” tag, an example of which can be seen in Schema 3. Note that the Base64 encoded data has been omitted, it has been replaced with the placeholder “%BASE64DATA%” for length reasons, as the encoded data can contain many thousands of characters.

The key attributes of interest (apart from the actual Base64-encoded intensity data) are the scan number (num), MS level (msLevel, with value 1 for standard MS, 2 for an MS/MS scan), the scan type (scanType, with value 'Full' for a scan of the full *m/z* range, or 'Z' for a zoom scan), and the time (retentionTime, with a value in seconds). Tandem MS scans will contain additional parameters of interest, namely the charge of the precursor ion and its mass. The additional parameters are not of particular interest for this research and are not imported.

Each of these properties are extracted and stored in MATLAB arrays such that given an index value into any of the property value arrays remains constant for all other property value arrays. For example, the fifth element of the time array corresponds to the fifth element of the scan number array, the fifth column (columns being the time dimension) of the MS intensity array, and so on. The tandem MS and zoom scan data are extracted, but tandem MS data are not currently used during data processing, and as a consequence of the file format, zoom scan data are required when the instrument is run in a higher-resolution mode.

As noted earlier, a similar function was written as a component of the previous work, but suffered from performance problems. This was largely attributed to the use of loops to iterate over objects such as file contents and groups of bits. Ultimately, it is faster to alter these iterations so they are performed in parallel, or at the very least remove the requirement for sequential processing to allow MATLAB to process the data as it deems optimal. In some cases additional performance could be gained through the use of built-in MATLAB functions, or modified versions of those functions which can make specific assumptions about this type of data that a more general implementation of the same function cannot.

3.2.2 Base64 Conversion

The mzXML format stores *m/z* and intensity pairs of an MS scan as a Base64 encoded string. These data must be decoded prior to use, and a MATLAB function has been written for this purpose. Base64 uses a subset of 64 ASCII characters to represent binary data. Thus, to convert from Base64 back to binary data, the sequence of characters must first be translated from their 8-bit ASCII representation to their 6-bit Base64 counterparts, which can be accomplished with a look-up table in which the value of the *i*th row contains the Base64 index of ASCII value *i*. The Base64 indices range from 0-63, and can therefore be represented by six place values in the binary number system (2^6=64). These binary representations are concatenated and repartitioned into groups of 8 bits, which restores the original eight-bit binary data. Conceptually, the first 8-bit sequence is composed of the entirety of the first six-bit number, and the first two bits of the next six-bit number. The following 8-bit sequence is composed of the last 4 bits of the second six-bit number, and the first 4 bits of the third 6-bit number. This pattern continues until 24 bits, after which it repeats (24 being the least common product of 6 and 8). As *m/z* values and intensities are not integers, they are stored as floating-point single-precision values (32 bits per value, or four sets of eight bits). Such values can be stored either with the most significant (highest value) bits first, or the least significant (lowest value) bits first. As the mzXML format and MATLAB differ in which bit ordering they use, it is necessary to reverse the order of the four eight-bit components that comprise a single-precision value. There do exist cases where the Base64 data are of insufficient length to produce an integer number of eight-bit values; in these cases the remaining space is padded with binary zeros prior to conversion. A visual overview of this process may be found in Table 1, and the Base64 decoding function itself in Appendix 5. Note that the byte significance reordering and conversion to a single-precision value are completed immediately following the Base64 conversion of the data, but these commands reside in the mzXML import function, so that the Base64 decoder is a generalized function which can be re-used elsewhere.

Table 1: An overview of the steps involved in converting (truncated) Base64 encoded data in an mzXML file to m/z and intensity pairs. The order of steps flows from the top of the table to the bottom.

Base64 data	Q	8	g	K	q	0	E	E
Character index	16	60	32	10	42	52	4	4
6-bit value	010000	111100	100000	001010	101010	110100	000100	000100
8-bit repack	01000011	11001000	00001010	10101011	01000001	00000100		
Decimal value	67	200	10	171	65	4		
Significance reorder	171	10	200	67	…			
32-bit value	400.0833							

Again, performance was a limiting factor, and optimization of this conversion function was similar to that of the mzXML import function; significant performance gains were achieved by re-writing the functions to replace an iterative sequence of commands with their matrix-based counterparts. Additionally, repeated calls to base two and base ten conversion functions were replaced with computationally efficient lookup tables, matrix operations, and boolean logic.

3.3 Computing Models of Peptide Pair Isotopic Peaks

Detecting the presence of peptide pairs in the MS data necessitates models to serve as a reference for comparison. This model must have the correct ratio of isotopic peaks for a particular *m/z* value to minimize the difference between the model and an actual isotopic pattern which is present in the spectral data. These models are computationally intensive to construct, and are therefore precomputed and stored for future reference.

It is worth noting that the version of this method discussed in the previous work[36] was limited to a total of four peaks (M and the M+1 to M+3 isotopic peaks). This was found to be insufficient at high mass values, and therefore the functions relating to isotopic peaks have been extended up to and including the M+10 peak.

3.3.1 Average Peptide Composition Formula

One issue that adds complexity to the modelling of isotopic peaks is the molecular formula of the peptides. Each peptide will be different, so a wide range of peptide formulas will result. Modelling the isotopic pattern for each of these individual peptides is a computationally expensive problem which becomes significantly more difficult when only the molecular weight (but not the formula) of the peptide is known. For this reason, an “average” peptide method from the previous work[36] is used. Briefly, this algorithm computes the distribution of amino acids in the protein sequence to be searched for, and determines the relative ratios of each of carbon, hydrogen, nitrogen, oxygen, and sulfur. This makes it possible to compute the elemental ratios of an “average” peptide's molecular formula, and subsequently the formula of an “average” peptide for any given mass. These average peptide formulae are then used in the following section to compute the isotopic ratios of the peaks at a given mass.

3.3.2 Isotopic Ratio Pre-Calculation

Computing the isotopic ratios for a given molecular formula can be approached in a number of ways. The simplest approach (conceptually) is to analyze the formula, and find the possible isotope counts that would give rise to an M+1 peak, then determine the counts for an M+2 peak, and so on. While this solution is acceptable for a single formula, it becomes a very tedious task when it is necessary to compute the isotopic ratios for a large number of formulae. This is because the entire calculation hinges on the counts for each isotope. The method described herein is computationally more efficient because it abstracts the counts to be independent of the lowest-mass isotopes for as long as possible – which means it is possible to pre-compute an abstracted table of isotope counts that give rise to each of the isotopic peaks. To determine the isotopic ratios, it is only necessary to select the appropriate counts (those that have an appropriate number of elemental atoms for the given formula) from the table, and sum their contributions to the pattern.

This precomputed table is calculated using the isotopcalc.m function in Appendix 7. As mentioned, the function is written so as to be independent of the number of lowest-mass isotopes for atoms in the molecule. It iterates over all possibilities of having zero to ten of a given isotope (^{13}C, ^{2}H, ^{15}N, ^{17}O, ^{18}O, ^{33}S, ^{34}S, ^{36}S). This gives rise to 10^8 possible combinations, though not all of them will fall within the first 10 isotopic peaks – for example, a molecule with one of every isotope mentioned would have a mass that is 11 higher than its lowest-mass counterpart (the consequences of limiting the algorithm to 10 isotopic peaks is discussed in a subsequent section). The mass difference of each of these possibilities is computed relative to the lowest-mass counterpart, and this mass difference is used to sort the possibilities into one of a series of *i* "M+*i*" bins, representing the M+1 to M+10 isotopic peaks. On completion of the precalculation, each bin will contain a list of the isotopic counts that contribute to that particular isotopic peak, and it is the contents of these bins that are stored for future use.

3.3.3 Computing an Isotopic Pattern for a Given Molecular Formula

Once the table from the previous section has been computed, it is possible to determine the isotopic patterns for a given elemental formula. The function written for this purpose may be found in Appendix 8. The first step in this process is establishing which of the precomputed possibilities are invalid, and eliminating them. Invalid possibilities are considered to be those table entries that have a higher total count for any particular element than defined by the input molecular formula, e.g. an entry with four

^{33}S atoms is not a valid contributing entry to the isotopic pattern for an input formula containing only two sulfur atoms. The valid possibilities are copied to a new table, and supplemented with the counts of the lowest-mass isotopes, which can be calculated by subtracting the sum of the isotopic counts for an element from the quantity of that element provided in the molecular formula. A multinomial distribution function (Appendix 9) is used to calculate the probability of each entry in this table. These probabilities represent the contribution of that particular combination of isotopes to the final isotopic pattern. All of these contributions are then summed and normalized to the lowest-mass (M) peak, to give the final isotopic ratios for that mass value.

Note that the multinomial distribution function makes use of repeated factorial calculations on large integers, which is computationally inefficient. As such, the standard factorial function has been replaced with a faster hybrid version (Appendix 10) that utilizes a look-up table of past factorial computations to eliminate these repeated calculations. However, this table also has limitations in the maximum value it can store before an overflow occurs. In MATLAB, this limit is reached when computing the factorial of 171 or higher; the double precision data type used cannot contain the resulting value and is set to infinity (Inf). Subsequent divisions necessary to calculate combinations result in invalid answers (either not-a-number, “NaN”, which can result from division by zero, or further “Inf” responses when dividing an infinite value by any other number) so it is necessary to use an alternate method of calculating the combinations by reducing the factorials as much as possible. This reduction is possible because the division of two factorials can be simplified to the product of integers greater than the lesser of the two numbers, up to and including the greater of the two, e.g. 5!/3! = 1x2x3x4x5/1x2x3 = 4x5.

3.3.4 A Note on Isotopic Ratio Limitations

The isotopic peak calculation method outlined previously is limited in two ways. First, it is constrained to a maximum of 10 isotopic peaks. Second, it only iterates over a maximum of 10 isotopes of each element. An inspection of the final contributions of each isotopic peak confirms that this is sufficient for the purposes of this research; even at the heaviest modelled mass of 5100 Da, (a triply charged species at 1700 *m/z*), the tenth isotopic peak (M+11) has a contribution which is 1.4% that of the most intense (M+4) peak. Intensities normalized to the base peak are given below in Table 2. This suggests that ten isotopic peaks are sufficient to establish a peptide pair model for which the unaccounted isotopic peak contribution is less than 1.4% of the maximum intensity.

Table 2: Intensities of isotopic peaks for heaviest possible mass, normalized to base peak.

	M	M+1	M+2	M+3	M+4	M+5	M+6	M+7	M+8	M+9	M+10
Intensity	1.00	2.86	4.34	4.62	3.85	2.66	1.59	0.83	0.39	0.17	0.07
% Max	21.6	61.9	94.0	100	83.3	57.7	34.3	18.0	8.52	3.66	1.44

The limitation of 10 isotopes of each element does not affect the isotopic patterns over the range calculated. Since each isotope has a minimum contribution of +1 to the mass value, a molecular formula containing more than 10 of any particular isotope (regardless of the isotope in question) will fall outside of the first 10 isotopic peaks, and will not be of consequence to the model.

Should the need arise, it is not a difficult procedure to increase the number of peaks being computed, though at a tradeoff of increased computational time during the model creation phase.

3.3.5 Modelling a Single Peptide Pair

Once the isotopic peak ratios have been computed, there is sufficient information to construct a vector which models the actual peaks as would be observed in the raw MS intensity data. The previous work of Boutilier established that using Gaussian peaks models the isotopic pattern acceptably. The width of the peaks is established depending on the resolution of the instrument and data; higher resolution data will have more sharply defined peaks, while the opposite is true for lower resolution data. The peak width value determined in this previous work (0.15) was found to provide acceptable matches with real spectral data, and remains unchanged for low-resolution data. Higher-resolution data necessitated the re-evaluation of this parameter, which was done by Samantha Rudolph as a summer research assistant, who found that the isotopic peaks were best modelled using widths of 0.08,0.07, and 0.06 for singly-, doubly-, and triply-charged species, respectively.

The relative locations of the peaks in the *m/z* domain (and their width) is inversely proportional to the charge (*z*) of the peptide being modelled; as the charge increases, the spacing between the isotopic peak pair decreases to 1/*z* *m/z* units; the width of the isotopic pattern decreases similarly. However, the spacing between the light and heavy isotopic patterns of a peptide is determined by the number of labels a peptide possesses. Recall that a single “label” is considered to be the dimethylation of a primary amine, and that in addition to the N-terminus of a peptide, any lysine residues present are also labelled. Each label contributes a separation of 4 mass units between the corresponding peaks of the light and heavy isotopic patterns. The resolution of the data initially used for this work limited the maximum charge value on a peptide to +3; higher charge values do not have sufficient separation between isotopic peaks (and the separation between peptides which are labelled with only a single tag) to make these signals clear enough to distinguish from non-peptide signals. The single mass unit separation also causes the two isotopic patterns of a labelled pair to overlap significantly.

These model vectors for *m/z* 400 may be seen in Figure 8. In this work, eight separate “classes” of pairs have been defined based on the charge and number of labels on the peptide being observed: singly charged peptides with one (sub-figure 1) and two labels (sub-figure 2), doubly charged peptides with one to three labels (sub-figures 3-5, respectively), and triply charged peptides with one to three labels (sub-figures 6-8, respectively). Each isotopic pattern is retained in a separate vector so that they may be modelled separately from one another. Some classes have a large separation between the light (solid) and heavy (dashed) isotopic patterns; these spaces may not be empty in the raw MS data. Therefore, these regions (and the regions before the first isotopic pattern and after the second pattern) are excised prior to fitting the model.

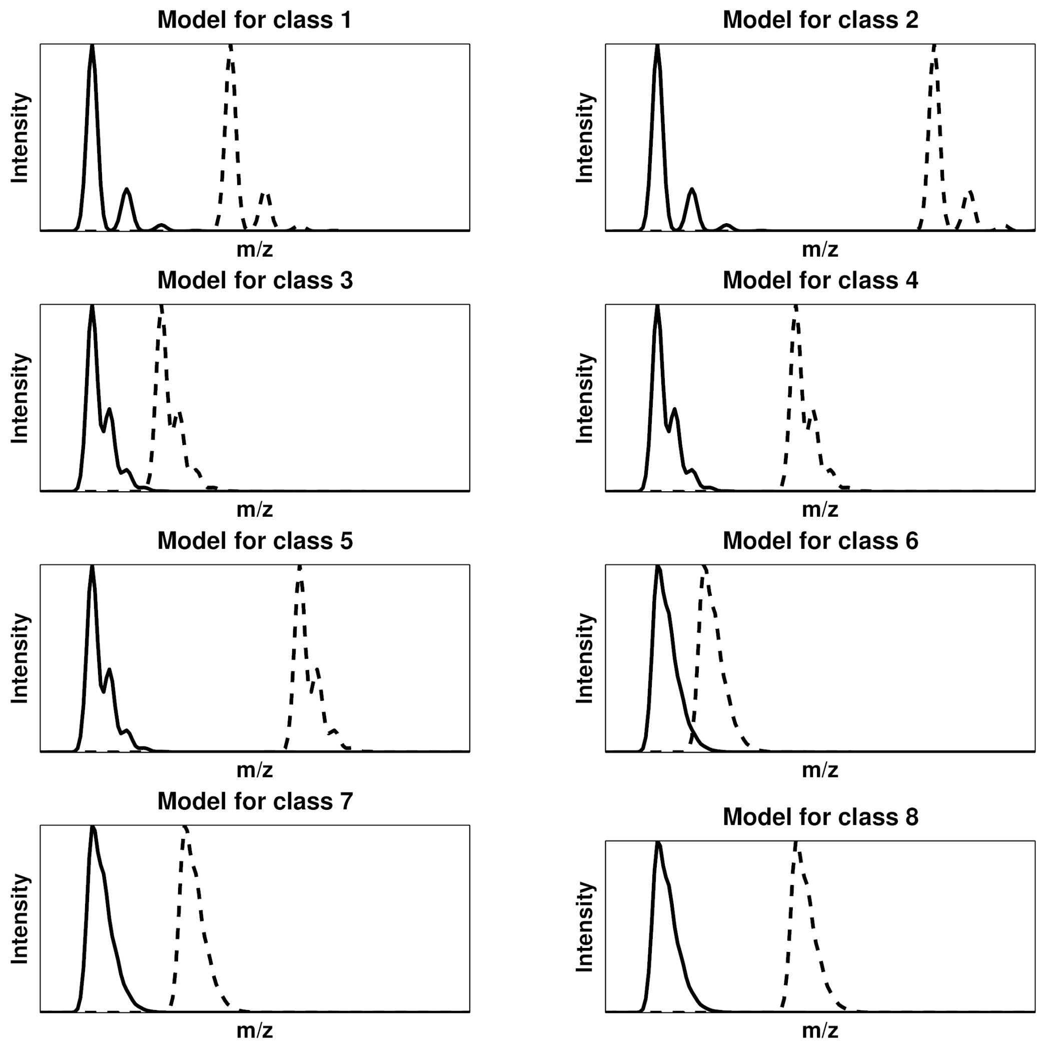


Figure 8: The isotopic peak models at m/z 400 for each of the eight classes of peptides: z=+1 with 1 and 2 labels (1,2), z=+2 with 1, 2, and 3 labels (3,4,5) and z=+3 with 1, 2, and 3 labels (6,7,8)

In contrast, the models at *m/z* 1700 are shown in Figure 9. Note the significant change in the intensities of the isotopic patterns. It should be noted that these patterns are computed at intervals of 10 *m/z* values - this is done to reduce computation time as the isotopic patterns do not change significantly enough within this interval to require recalculation of the model for every possible *m/z* value.

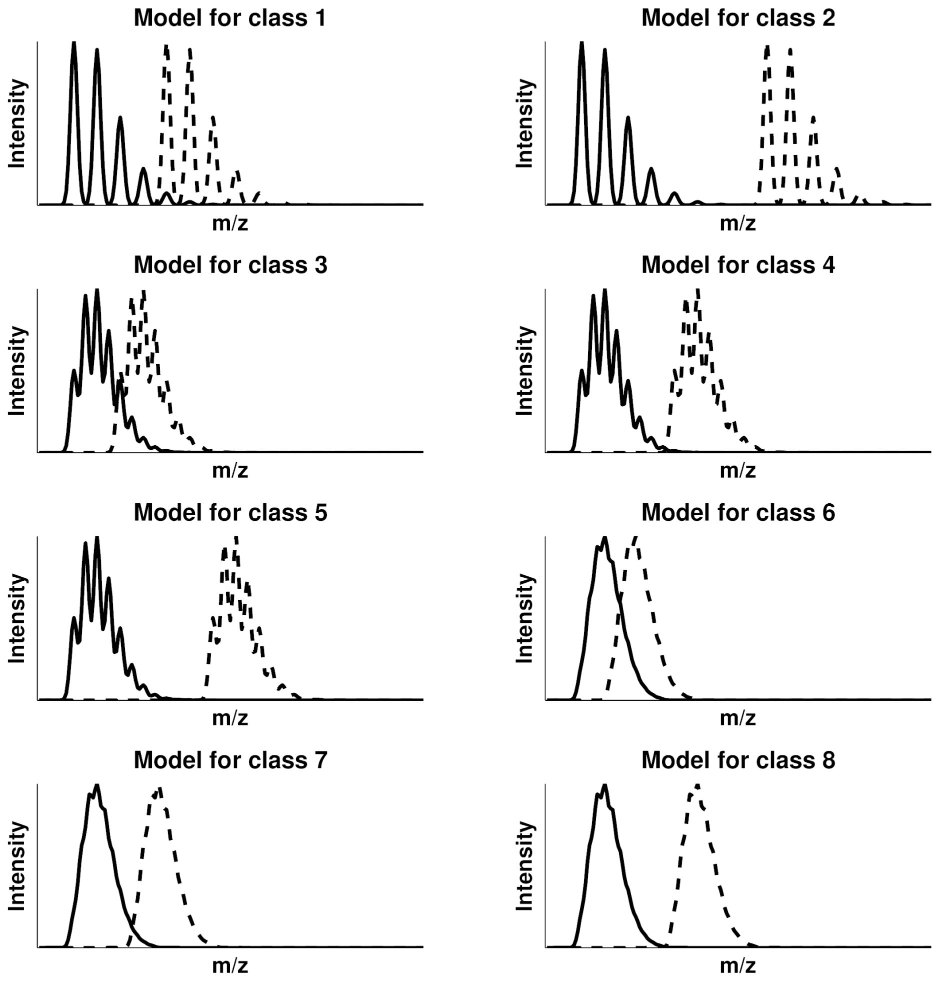


Figure 9: The isotopic peak models at m/z 1700 for each of the eight classes of peptides: z=+1 with 1 and 2 labels (1,2), z=+2 with 1, 2, and 3 labels (3,4,5) and z=+3 with 1, 2, and 3 labels (6,7,8)

3.4 Algorithm for Initial Detection of Peptide Pairs

The detection of peptide pairs in the raw data is a multistage process. An initial search of the data allows for the annotation of locations which contain potential peptide pairs. If we assign a single time point extracted from the raw MS data to a response vector, **r**, then there exists some coefficient matrix **c,** which, when multiplied with the model matrix **K**, reproduces an idealized version of the response vector, as in Equation 1.

$$\mathbf{r} = \mathbf{cK} \tag{1}$$

Here, **r** is a 1x*p* vector of mass spectral intensities, **c** is a 1x3 vector of model coefficients, and **k** is a 3x*p* matrix of profiles comprised of the light and heavy isotopic patterns, and a constant baseline. These isotopic profiles have been generated based on the presumed mass of the peptide fragment, and the baseline is a vector of ones. The *p* points of the mass spectral data correspond to a window which encompasses the model, but these points may not necessarily be contiguous. As the desired solution to this equation is the coefficient vector **c**, Equation 1 can be rearranged to isolate this term, yielding Equation 2, which estimates **c** as $\hat{\mathbf{c}}$.

$$\hat{\mathbf{c}} = \mathbf{rK}^{\mathbf{T}}\left(\mathbf{KK}^{\mathbf{T}}\right)^{-1} \tag{2}$$

Given that the model **K** is comprised of three components (two vectors modelling the light and heavy isotopic peak patterns, and a vector of ones representing the baseline), solving for $\hat{\mathbf{c}}$ yields three values, which will represent the intensity of each of those three components in the response vector **r**. Locations in the raw MS data which feature a pair of isotopic peaks which align with the model **K** should therefore exhibit high intensity values for the isotopic peak components of the model, relative to the intensity parameter of the baseline, and overall, a good fit to the measured data.

Since the models contain regions in which the peptide peaks are not present (such as between the last peak of the light pattern and the first peak of the heavy pattern), these areas are excised from the components **r** and **K** of the equation. Fitting these “blank” regions when a signal is present within them would decrease the quality of the fit, and erroneously indicate that the model was not valid.

As the computation of **c** is identical for any given *m/z* value throughout the entire time domain of the experiment, it is possible to process a matrix of response vectors in a single multiplication, greatly improving the efficiency of the calculation and reducing the required computation time for the entire matrix of MS intensity data. Equation 3 is similar to Equation 2, except that **r** and $\hat{\mathbf{c}}$ have been replaced with matrices **R** and $\hat{\mathbf{C}}$, producing a series of *m* intensity parameters, where *m* is the number of time points in the MS data.

$$\mathbf{R}\mathbf{K}^{\mathrm{T}}\left(\mathbf{K}\mathbf{K}^{\mathrm{T}}\right)^{-1}=\hat{\mathbf{C}} \tag{3}$$

It is necessary to have a measure of the quality of the fit at a particular point in the MS data; such a parameter makes it possible to set a cutoff below which the response vector is considered to lack a pair of peptide peaks, or contain peaks which are not quantifiable. The standard error of the estimate provides a measure of the deviation of the fitted model from the raw data; this error is computed as per Equation 4.

$$s_e(i)=\sqrt{\frac{(\boldsymbol{r}_i-\hat{\boldsymbol{r}}_i)(\boldsymbol{r}_i-\hat{\boldsymbol{r}}_i)^T}{p-3}} \tag{4}$$

It is then possible to compute a signal-to-noise ratio (taken as the minimum of the maximum fitted intensity for each isotopic pattern over the standard error of the fit) for each detection, which can be used to establish the presence of a peptide pair or not, as show in Equation 5. Note that the profile vectors are normalized to a maximum intensity of unity, therefore the coefficients c_1 and c_2 correspond to the maximum intensities of the fitted profiles.

A high signal-to-noise ratio will correspond to both a good fit to the two patterns and

$$(S/N)_i = \frac{min(c_{1i}, c_{2i})}{s_e(i)} \qquad (5)$$

low fit error; a poor signal-to-noise ratio indicates either a poor fit to the models, or a signal that is close to the same order of magnitude as the error in the fit. Figure 10 shows a surface plot of a representative singly-charged peak pair, with one label. This peak pair has several features which hinder a straightforward model fit and assignment of the peak class. Specifically, the interfering peaks between the two isotopic patterns, the intensity fluctuations over the chromatographic dimension, and the isotopic pattern spacing of 4 *m/z* units, which is a feature of three of the classes. These three classes will therefore show some degree of fit to the model, and the signal-to-noise ratio enables determination of the correct class. These reasons make the peak a good candidate for illustration of the modelling behaviour in commonly encountered situations. These features can be seen in Figure 11 and Figure 12.

The key features to note are the small interfering peak just before *m/z* value 406 in Figure 12 and the dip in intensity at 1500 seconds in Figure 11. The calculated signal-to-noise values in the same region are displayed subsequently in Figure 13. They are

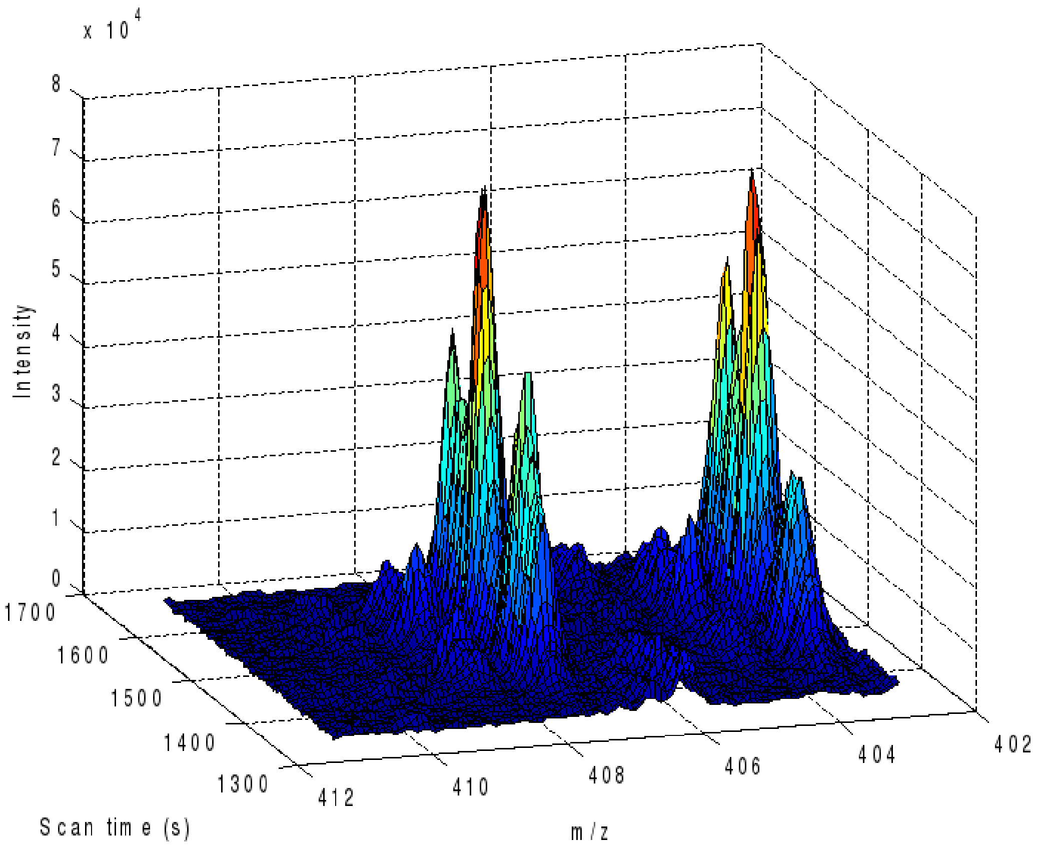


Figure 10: A representative singly-charged peak pair with one label. This peak pair is used to illustrate the signal-to-noise ratio behaviour and several aspects of fitting such as interference, multiple model fits, and chromatographic intensity variations.

represented as intensity maps, where a darker region corresponds to a larger signal-to-noise ratio. As expected, the signal-to-noise ratio for the correct class (singly charged with one label, sub-figure A) has the highest intensity values at the same coordinates as the base peak of the first isotopic pattern of the pair. This corresponds to when the model is perfectly aligned with the peak pair in the intensity data.

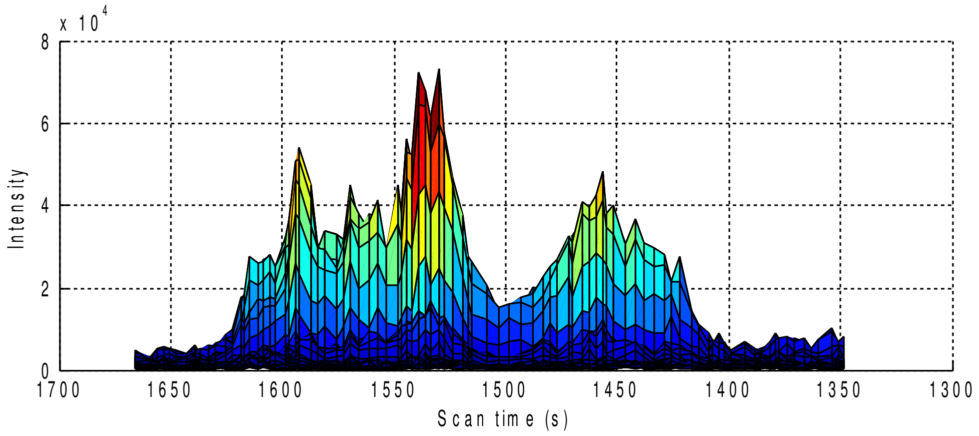


Figure 11: The chromatographic dimension of the peak pair in Figure 10. The feature of note is the dip in intensity at 1500 seconds, which can result in two separate groupings of detections if the signal-to-noise ratio drops below the cutoff value.

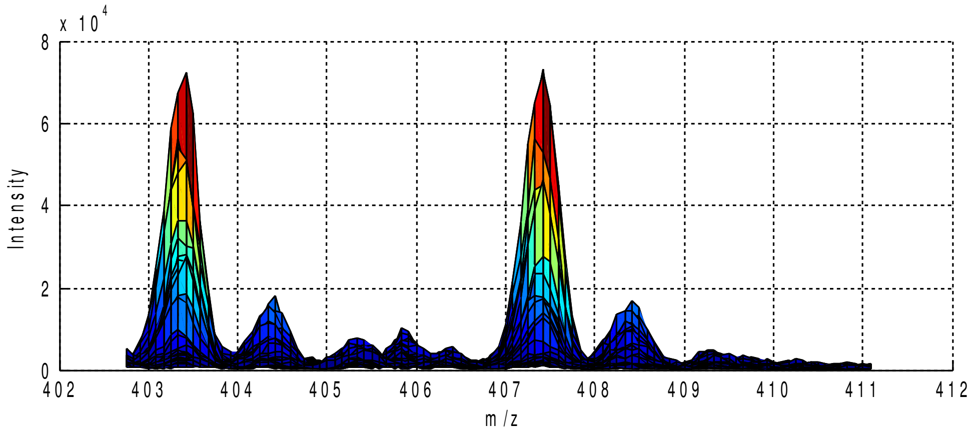


Figure 12: A view of the m/z dimension of the singly-charged, one-label peak pair shown in Figure 10. Note the interfering peak that appears just before m/z 406.

While it is difficult to see its presence in the three-dimensional surface plot, the interfering peak in Figure 10 appears at the lower retention times, corresponding to the left-hand side of the intensity maps (there are other peaks which appear at the higher end of the retention window shown, but these may actually be another peak pair of a differentclass, as evidenced by the dark region in sub-map C of Figure 13). The dip in intensity does not noticeably impact the signal-to-noise ratio values for this peak pair, but

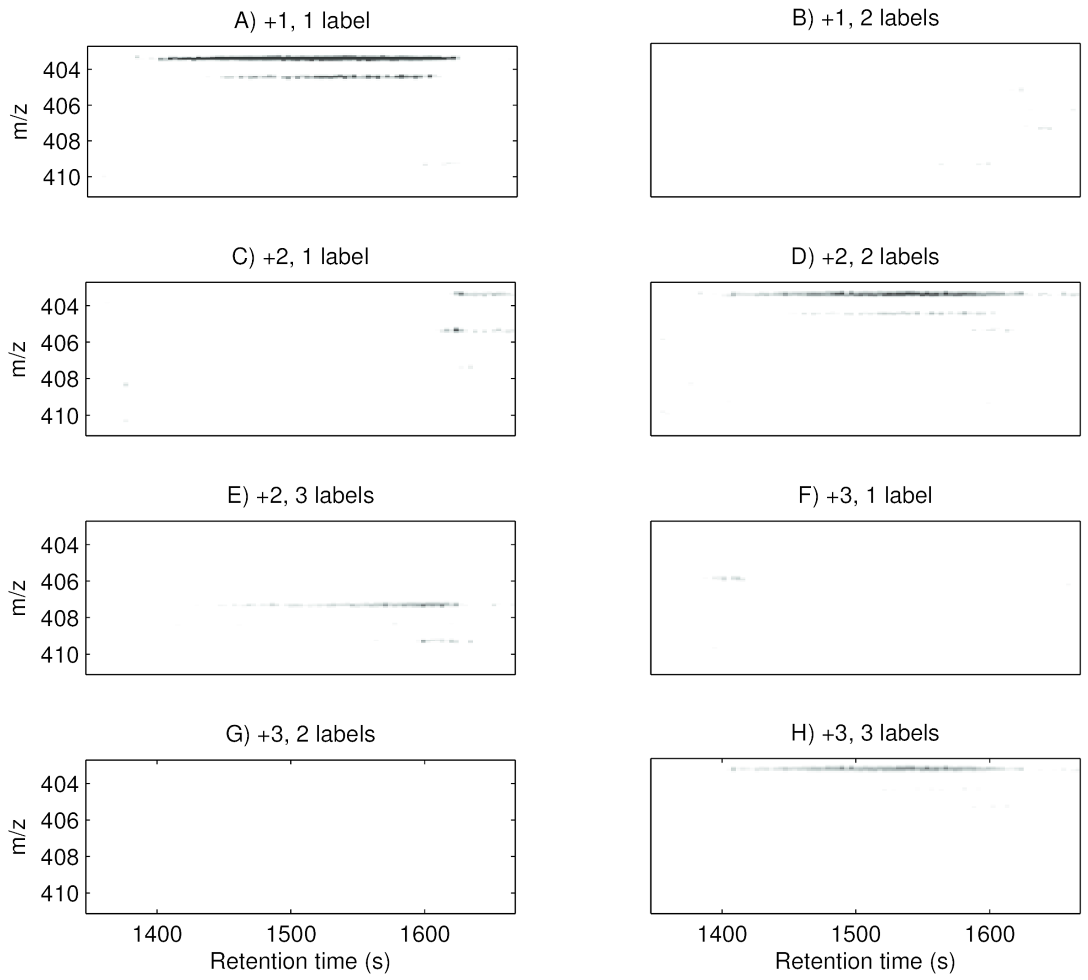


Figure 13: Intensity maps of the signal-to-noise ratio for each of the eight classes of peptide pairs, for the region corresponding to the peak shown in the previous figures. Darker regions represent larger signal-to-noise values, while values less than 3 are white.

in some cases it may drop below the minimum limit of 3. This value was based on visual inspection of the peaks, and is intentionally generous, since the peaks will be clustered and re-quantified later. Note that no classification is assigned for regions where the signal-to-noise ratio drops below 3 – and as such a dip in the intensity can appear as a gap in the peptide's classification band. This issue is addressed during the second stage of the detection when individual *m/z* and time points are clustered together. Provided the gap is narrower than the time threshold in the clustering algorithm, the gap would be "traversed" during the exploration phase, and both parts of the detection band would be assigned the same cluster ID.

Some of these maps exhibit several regions of high signal-to-noise values which appear at the base *m/z* value plus a multiple of the inter-peak width of an isotopic pattern. This arises from the regular spacing of the peaks in the isotopic pattern; the model fits well when shifted by an inter-peak width as the isotopic pattern has a degree of self-similarity – ignoring the base peak and aligning to the M+1 peak results in a reasonable match, but at lower intensity than the base peak (if the base peak is not the most intense, this self-similarity vanishes and aligning to the M+1 peak does not result in a match).

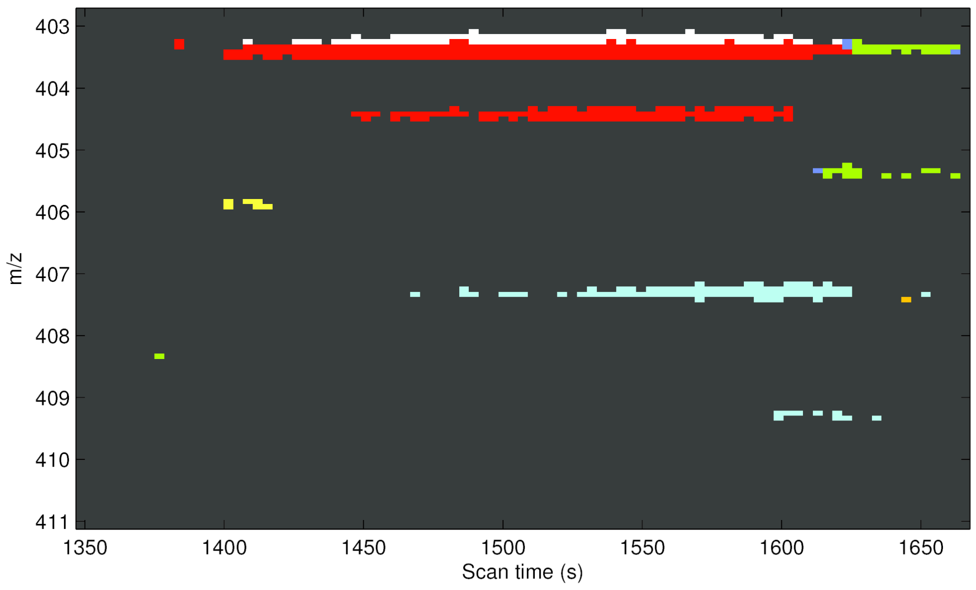


Figure 14: Assignment of the peptide class detected at each of the mass-to-charge values and time points in the raw data, after the initial detection is complete. The region shown here corresponds to that of Figure 10.

The final point of note is that there is a reasonably large (3 or greater) signal-to-noise value for both of the other classes that have a distance of 4 *m/z* units between the isotopic base peaks (sub-figures D and H). This is also an artifact of the model alignment to the raw data, and it manifests itself in the class assignment as a detection band which contains multiple adjacent classes, as in Figure 15. This is commonly seen for any charge and label combinations with the same spacing between the base peaks of the isotopic patterns. Colours represent peptide classes as follows: red: *z*=+1, 1 label; orange: *z*=+1, 2 labels; green: *z*=+2, 1 label; blue: *z*=+2, 2 labels; cyan: *z*=+2, 3 labels; yellow: *z*=+3, 1 label; pink: *z*=+3, 2 labels; white: *z*=+3, 3 labels.

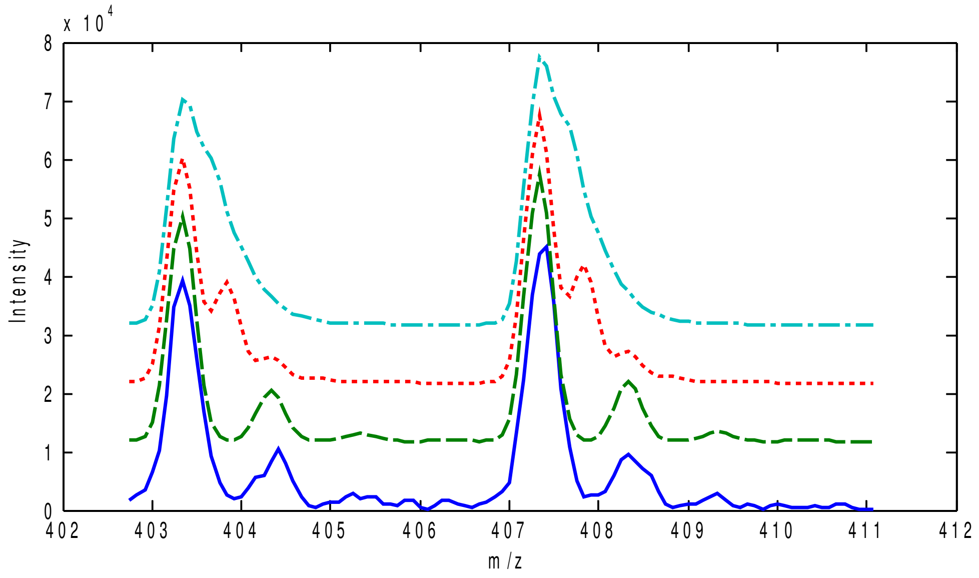


Figure 15: A peak pair extracted from the raw data (solid line) and three different models (dashed line: singly charged, 1 label; dotted line: doubly charged, two labels; dash-dot line: triply charged, three labels) with the same m/z spacing between isotopic patterns showing that all three models will have some degree of fit to the raw data. The correct model will have the highest signal-to-noise ratio.

Figure 15 shows the mass-to-charge region at a single retention time, upon which the three “matching” models in question have been superimposed. It is clear that the incorrect models have the same spacing between the light and heavy signals, and the proportional nature of the charge increase results in additional alignment of non-corresponding peaks.

As a result, these “incorrect” models still have relatively low fit errors, which necessitates reassessing the assigned model over the entire detection after the clustering phase is complete.

3.5 Algorithm for Clustering Signals

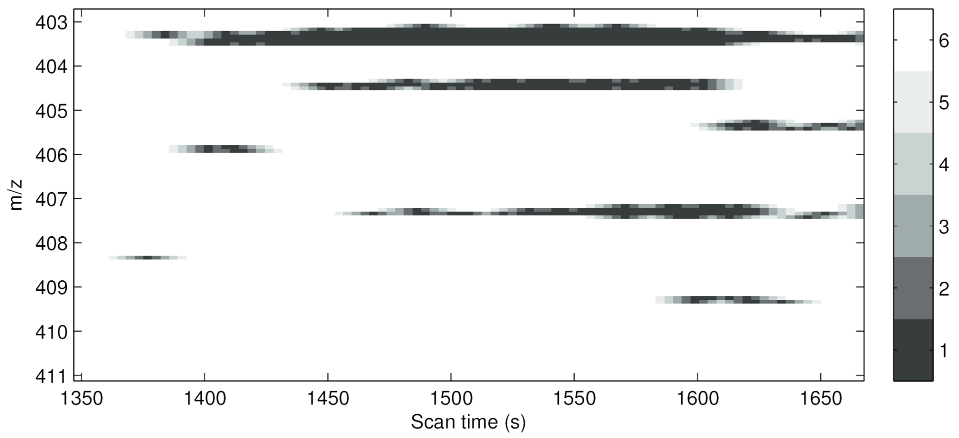
The initial detection algorithm assigns a classification for every *m/z* value and time point combination where a peptide peak pair was detected. As can be seen in Figure 15 (and discussed in the previous section), this does not produce a perfectly rectangular detection region of the same class, nor does it establish the chromatographic duration of the peak. Determining this contextual information for a signal requires an additional processing step in which individual detections are grouped together via a clustering algorithm into a single *m/z* value and time range over which a given signal is present. This process is divided into three distinct stages, which are the clustering of detections, calculation of the properties of a cluster, and overlap removal.

3.5.1 Clustering of Individual Detections

Grouping of detections at individual *m/z* and time values is a computationally challenging problem. This arises from the visual nature of the problem. While it is easier for an observer to view a pattern of points and group them based on separation in the *m/z* and time dimensions, it is computationally complex as it becomes necessary to select and define parameters that establish cut-off values beyond which a particular detection is no longer considered to belong to a group. The most straightforward parameters are the temporal distance, *m/z* distance, and the class index of the peptide pair. All three of these are used to determine a weighted Euclidean distance between two points and establish whether they are a part of the same cluster.

The clustering algorithm uses a last-in first-out data structure called a stack to visit each individual detection point in turn. This ensures that the area surrounding a detected point is explored, the nearest (and therefore most likely to be related) detections are processed prior to detections which are further away. “Processing” of a point refers to its removal from the stack and the evaluation of the criteria that is used to establish whether it is a part of the cluster that is currently being explored. If so, this point's immediate neighbours are placed at the top of the stack for subsequent exploration. Should the current point being processed be discarded, the next one is removed from the top of the stack and the process begins anew. Should the stack be empty (in other words, the immediate “area” surrounding the current candidate cluster has been explored and no new member points identified), the next detection point which has not been assigned a cluster ID is placed into the stack and processed as described until all points have been assigned a cluster ID.

Cluster IDs are tracked via a binary flag indicating if the algorithm is currently “visiting” a cluster. Visiting a valid point while this flag is true will retain the current cluster ID and assign the point to that cluster; if the algorithm visits a valid point while not visiting a known cluster (a “new” cluster has been discovered), then a new cluster ID is assigned to this point. Validity of a point is established using a minimum signal-to-noise ratio below which the point is not considered to match the model. Finally, the point's distance weighting (based on the three parameters mentioned earlier) is evaluated. If the weight of the point is both less than the specified cutoff, and less than any existing weight (points may be visited more than once with this algorithm, and thus it is necessary to ensure that only the “shortest” distance is used), the point's eight surrounding neighbours are added to the stack for exploration, as they may potentially be members of this cluster. Figure 16 shows the “clouds” that are formed as a result of the clustering algorithm on the sample region shown previously in Figure 15.



F**igure 16**: "Clouds" produced by the clustering function which establish the bounds of the region within which a detection is considered to be a part of the same cluster. Darker values indicate the spot is closer to a valid detection.

The darkest regions of this map correspond to a location where a peak pair was detected during the initial detection phase. The shading indicates a distance from the nearest valid detection, and the lightest shade (the background) indicates the weighted distance is greater than the maximum weighted distance a point may be from any neighbours while still remaining a part of that cluster. Therefore, these regions define the boundaries of a particular cluster and corresponding ID. Notice that even though some of the detections in Figure 15 are not contiguous regions they are still within the same dark region of this figure, and are therefore assigned the same cluster ID.

3.5.2 Cluster Contraction and Reprocessing

Now that the extent of each detection region has been established, the next phase of processing contracts these regions to a single *m/z* value corresponding to the base peak of the first isotopic pattern, and a time span over which this particular pattern is visible. In order to assign a single set of values to the intensity parameters, the chromatographic window over which the peak is present is averaged over the time dimension to produce a single vector in the *m/z* axis. This resulting vector will be of sufficient size that each of the eight models may be fitted to it at *p* different *m/z* points, where *p* is the maximum *m/z* range of the detection "band" for this particular cluster (that is, if a detection band spans more than a single *m/z* value, the models are fitted over the entire range of *p* *m/z* values). The overall maximum signal-to-noise ratio for each of these eight 1x*p* regions is identified, and the location of this maximum in the *m/z* dimension establishes the final *m/z* value at which the peak pair has been considered to be detected. Similarly, the class to which this maximum corresponds is taken to be the final class assignment of the peptide pair.

The final assignment of class types for the spectral region used in the previous figures can be seen in Figure 17. Only a single peptide detection is assigned, as the other detections either did not have enough individual detections in the chromatographic dimension, or the final signal-to-noise ratio computed was below the cutoff. Suitable values for these two parameters were determined through visual inspection of the data; specifying that a cluster should have a minimum of 5 detections in the chromatographic axis was found to eliminate single point detections arising from random alignment of noise with the model, while also avoiding very short chromatographic peaks which cannot be reliably quantified.

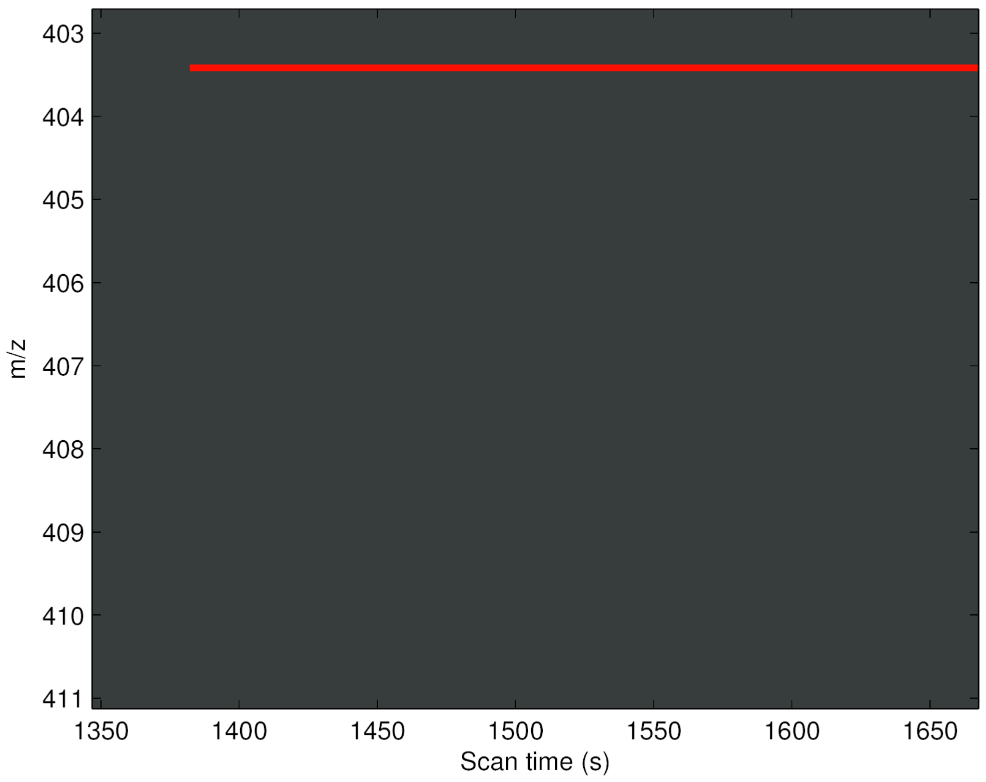


Figure 17: Final assignment of class and location of the peptide detections previously shown in Figure 15. Note that only a single final detection is indicated. The other detections were excluded either for not persisting long enough in the chromatographic axis (<5 points), or having a final signal-to-noise ratio which was too low (<8).

The signal-to-noise parameter was also determined by inspection. A minimum signal-to-noise value of 8 was selected because peaks below this value tended to be difficult to distinguish visually from the surrounding region. Additionally, lower values results in the inclusion of detection artifacts such as the second red band (between *m/z* 404 and *m/z* 405) in Figure 15. The value is somewhat restrictive in that it can potentially discard detections which are valid, but if the peak cannot be clearly distinguished from the background by inspection, the reliability of the final quantitation may be questionable regardless.

3.5.3 Cluster Overlap Removal

Once a definite location has been established for all clusters, it is necessary to establish regions in which the detections conflict by overlapping both in the temporal and *m/z* dimensions. This situation results in interference as the isotopic pattern of one detection can interleave and interfere with the isotopic pattern of another detection. For example, if two peptide pairs have isotopic pattern spacings of eight *m/z* units, and the second pair is offset by +4 *m/z* units from the first, the M+4 peak of the second pair's light pattern will overlap with the base peak of the first pair's heavy pattern, M+5 will overlap with M+1, and so on, respectively. In this situation, the overlaps will inflate the error when fitting the model to the raw data, and for this reason these regions cannot be reliably quantified with the current algorithm. Thus, affected time regions are excluded from the region over which the cluster is quantified.

The algorithm to detect these overlaps is straightforward and consists of locating the borders of each detected peak (both in the chromatographic and *m/z* dimensions) and then testing whether two or more of these boundaries overlap. If an overlap is detected, the extent of the overlap is determined. Should removing the overlap (on all involved detections) leave fewer time points than the minimum size specified earlier, the detection is marked with a negative class identifier (*e.g.* -1 instead of 1) to indicate it cannot be quantified, while allowing the location and class assignment to be retained. If sufficient points remain for quantitation, then the overlapping region is excised from the detection and the remainder is re-quantified. The extent of the region, and the area(s) over which it was quantified, are both retained. Finally, those detections which do not meet the minimum signal-to-noise ratio are discarded from the results list.

3.5.4 Parameter Selection

To make the designed algorithms as flexible as possible, they are configurable with several parameters (first pass signal-to-noise cutoff, clustering signal to noise cutoff, clustering weightings and distance cutoffs) to control their behaviour in excluding peaks and detections. Determining suitable values for these cutoff parameters was done based on manual inspection of the data and results. There is naturally some interplay between each of these parameters – for example, a higher first-pass signal-to-noise cutoff will also tend to increase the chromatographic size of a given peak, potentially resulting in more overlaps which must be processed.

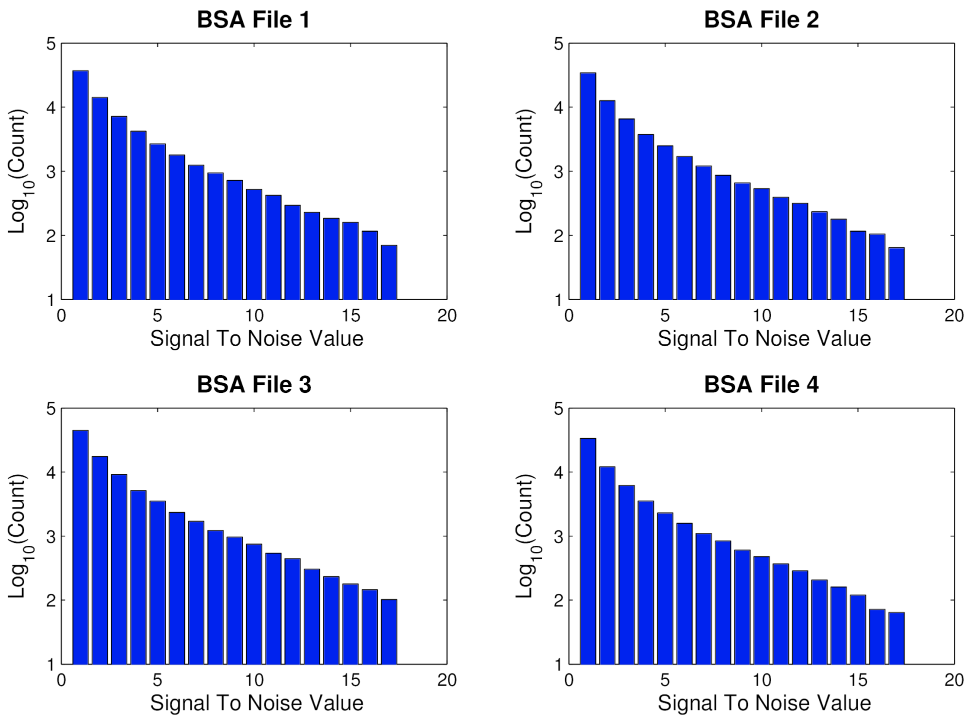


Figure 18: Histograms of the signal-to-noise ratios output by the first phase of detection. There are no immediately visible features such as plateaus or valleys that provide an intuitive cutoff during clustering. Counts are logarithmic to better represent the range of data in the figure.

As mentioned earlier, a base signal-to-noise ratio of 3 was selected for the initial detection stages. This was intended to be a loose control parameter that provides a reasonable filter that passes all but the very lowest quality detections, and behave similarly to the standard limit of detection (LOD) which is usually defined as three times the baseline. Examination of the initial signal-to-noise ratios computed for the peaks in each BSA data file did not reveal any distinguishing features to indicate an intuitive value to use as a cutoff. As seen in Figure 18, the distribution of signal-to-noise values

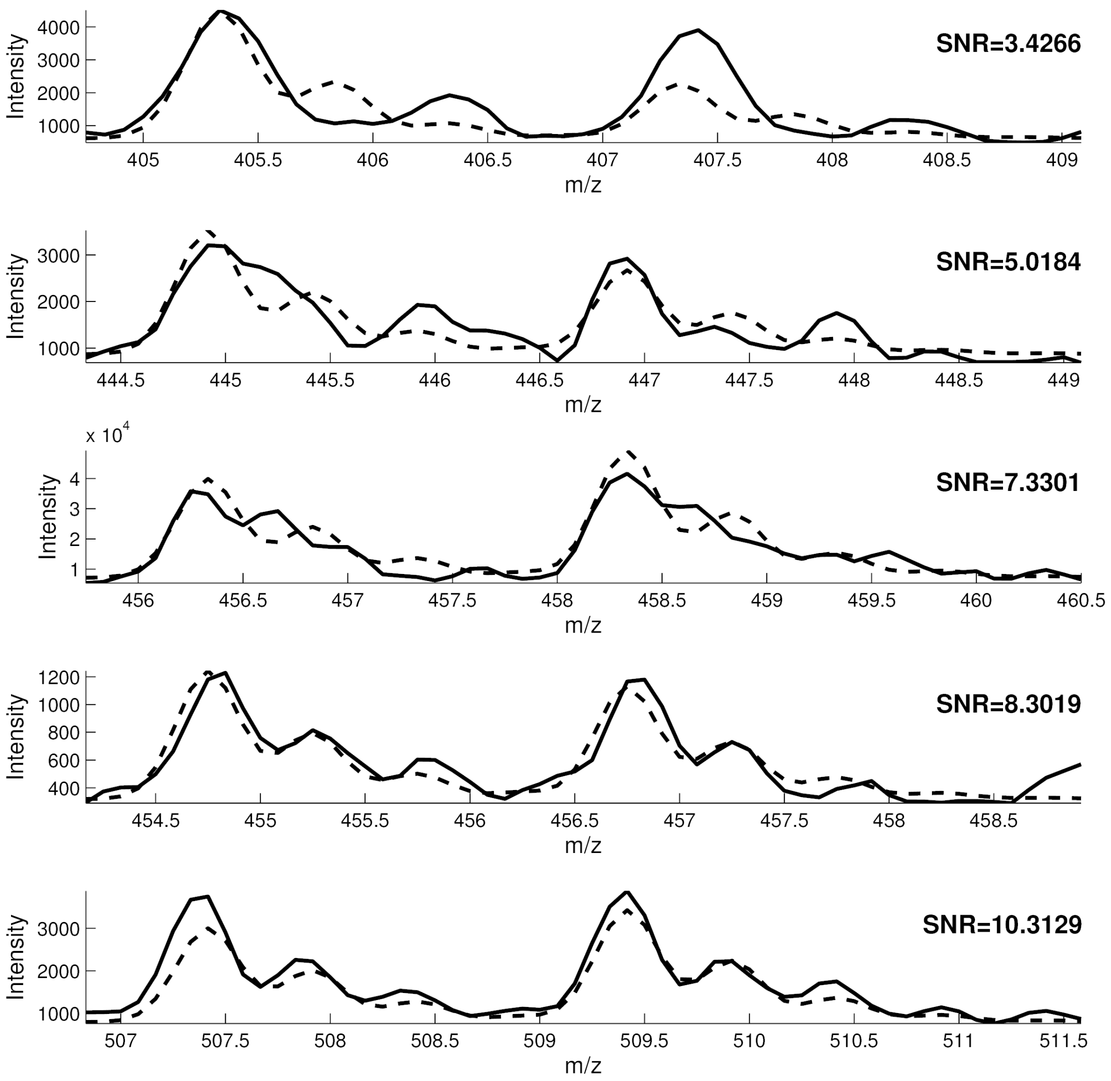


Figure 19: Representative peaks for doubly charged detections with one label at various signal-to-noise ratios. Note that the isotopic patterns and peak shapes become cleaner as the signal-to-noise ratio increases. The figure with the lowest signal-to-noise ratio appears to have clean peaks but note that the peaks are not spaced correctly for the class type. This is an inherent risk of selecting too low a signal-to-noise value when alignment artifacts are present. Solid lines are the raw signal while dashed lines represent the model.

appears to be a fairly continuous distribution with a long tail towards the higher signal-to-noise values. For this reason, determining a suitable cutoff value fell to manual inspection of the detection peaks in the raw data. Examples of these data are shown in Figure 19, which illustrates doubly-charged peaks with a single label which had final signal-to-noise ratios ranging from 3 up to 10. It is observed that the final isotopic patterns of the peak pair become clearer as the signal-to-noise ratio increases. The topmost plot (with signal-to-noise ratio just over 3) has “clean” peaks but the isotopic peak spacing corresponds to a singly-charged peptide. This was purposefully included to illustrate the erroneous detections that can result from a final cutoff that is too generous. That is, an acceptable signal-to-noise can be obtained for an incorrect model in this situation.

Initial exploration of detections with a signal-to-noise ratio of 10 or greater (based on parallels to the limit of quantitation, at 10 times the noise) revealed that these detections had signals which could be clearly seen in the raw data by manual inspection. This persisted when inspecting peaks with lower signal-to-noise ratios; only when inspecting detections with signal-to-noise ratios below 8 did the signals of these detections become less clearly defined. The overall shape of the peaks in the raw data was also examined and it was observed that at lower signal-to-noise values the peaks can become difficult to distinguish from the background (Figure 21). Based on these findings it was decided to set the final signal-to-noise cutoff at a value of 8. It should be noted that this parameter can change depending on the data in question, but the selected value should serve as a reasonable starting point for initial exploration when applying these methods to other data sets.

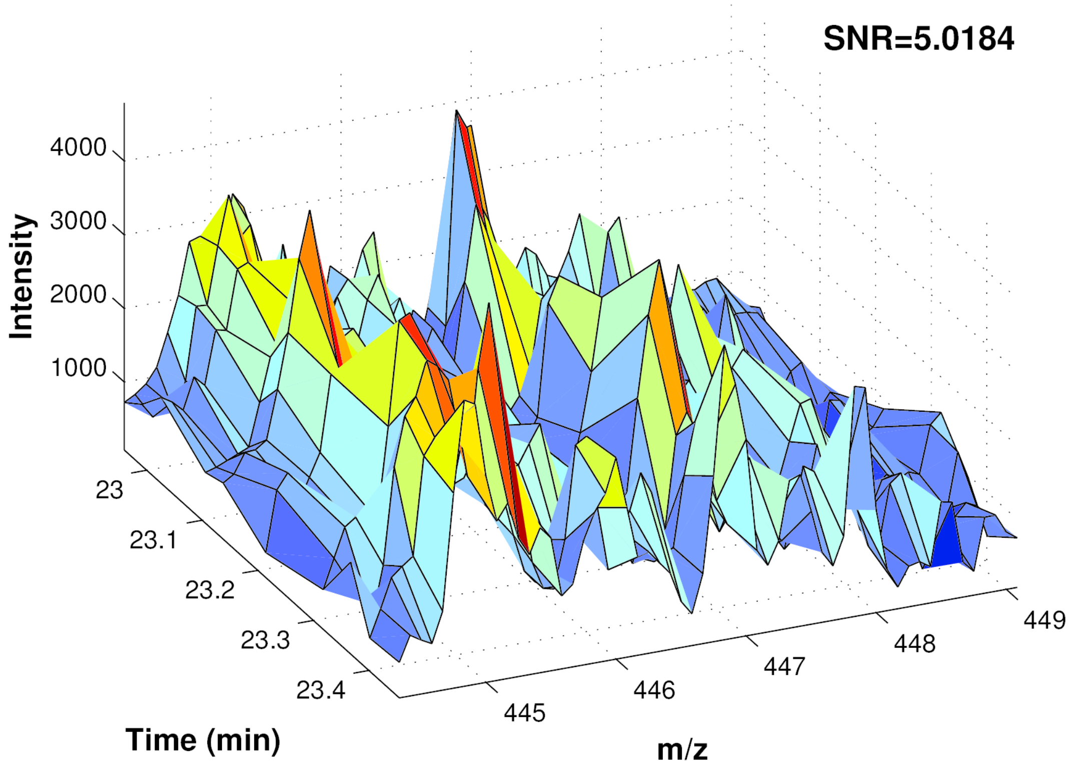


Figure 20: Surface plot of the raw spectral data for the peak from Figure 19 with a signal-to-noise ratio of 5.02, which is below the cutoff of 8. While the peak is visible, it is not well defined. Many of the peaks with lower signal-to-noise values have similar appearances.

The other clustering parameters were determined based on a best-guess estimate and visual inspection of the results; the minimum size of five time points appeared to eliminate the presence of scattered small detections which are either too small for reliable quantitation, or arise from random patterns of noise in the data that correlate well with the models. A corresponding value was also found to be suitable as a cutoff for the inclusion of points within the same cluster during the grouping phase of the algorithm, i.e. signals within five time points of each other and at the same *m/z* value were considered to belong to the same peptide. These values were set conservatively to prevent close, but distinct detections from being assigned a similar cluster.

This grouping phase also has parameters for the cluster “boundary” cutoff in the *m/z* direction and for the inclusion weighting in cases when the signal's best-fitting model (peptide class) changes within this normal *m/z* window of inclusion. The nature of the data is such that the *m/z* value of a peptide detection is more sharply defined than its temporal location, therefore these parameters were set such that only those signals which make direct “contact” with the current signal (in the *m/z* direction) are considered to be in the same cluster. This situation was only observed due to alignment artifacts, i.e. when a slight shift in the *m/z* direction of the model altered the classification of the peak cluster in a predictable patterns. This was observed when the separation of the isotopic patterns was ambiguous as to the class of the peptide signal. No cases were observed where this situation did not apply, therefore, the only time this cutoff value was utilized occurred in the presence of such alignment artifacts, no adverse behaviour of the algorithm was observed when the same cutoff was applied in all cases.

The final confirmation that the outlined algorithm functions as intended can be seen in the intensity ratios of one isotopic pattern to the other in a peak pair. Since the original samples for the BSA experiments were prepared at a 1:1 ratio of light:heavy labelled peptides, the observed ratios should reflect this information. This can be seen in the scatterplot (Figure 21). There does appear to be a slight offset from the expected value.

This may be attributed to sample preparation, as the light and heavy labelled samples originate from separate preparations and may not have been combined in an exact 1:1 ratio. The red + symbols are included to indicate detections which did not meet the minimum signal-to-noise cutoff of 8; the remaining detections (black points) form a distinct band around the theoretical ratio of 1:1.

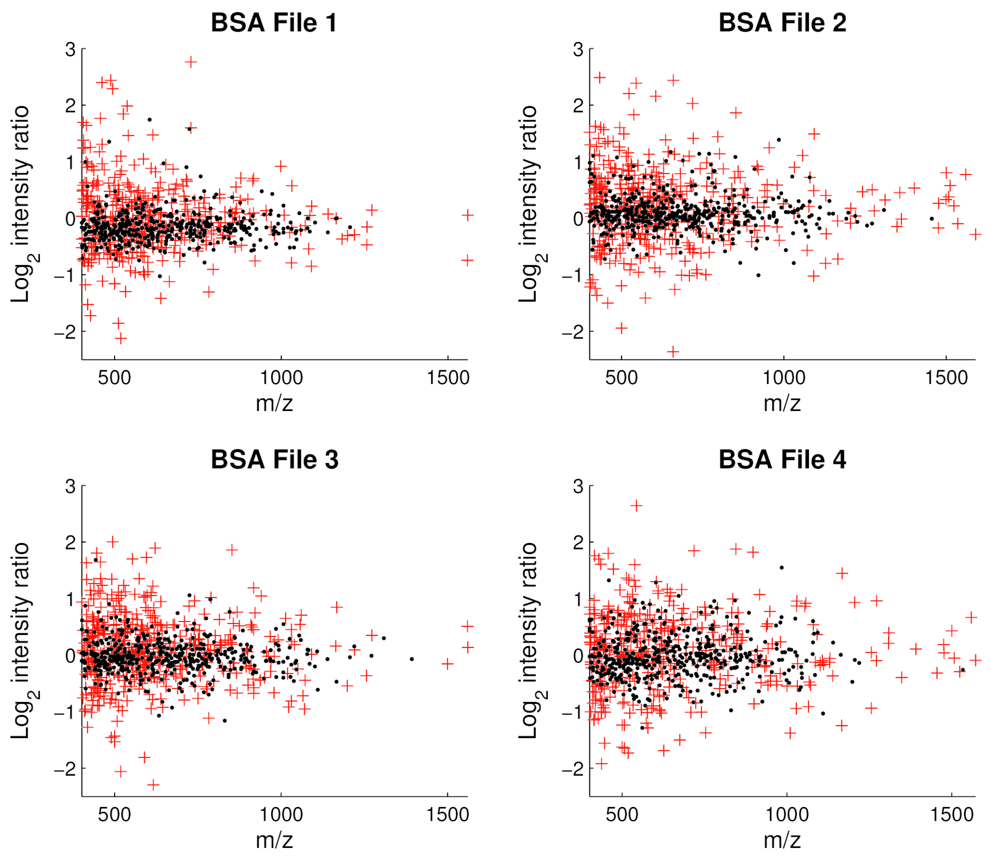


Figure 21: Plots of final computed ratios for each of the four BSA data files. Note the ratios are logarithmic for symmetric distribution about zero. Black points are those which have a signal-to-noise ratio of 8 or greater, red plus symbols represent detections that do not meet this criteria and are shown for reference.

3.6 High-Resolution and Triple Label Adaptability

A significant drawback of the reduced duty cycle imposed by the collection of tandem MS scans is a lower resolution in the standard MS scans, which can make it difficult to distinguish the individual peaks of an isotopic pattern at higher charge states. This limits the discriminating abilities of the model. If tandem MS scans are eliminated from the data collection process, it is possible to obtain higher-resolution MS scans by increasing the *m/*z interval of the full scans so that the time previously allocated to tandem scans is allocated to full scans instead.

Initial tests with a sample of these higher resolution data files (collected with an *m/z* interval of 1/50 as opposed to 1/12) exhibited promising results. These data files were of sufficient resolution to resolve the individual isotopic peaks of a triply-charged peptide pair, which was not possible when using the lower-resolution data set. However, the nature of these data files is such that the models used in fitting require both a higher density of points, and a better understanding of the relation between the data and the peak width parameter (σ) of the gaussian model used. Determining this peak width parameter for the higher-resolution data was the work of a summer student, Samantha Rudolph. After locating and analyzing a series of isotopic peaks in the higher-resolution data, it was found that the peak width parameter varied slightly with charge, resulting in calculated width parameters of 0.08, 0.07, and 0.06, for singly-, doubly-, and triply-charged isotopic patterns, as opposed to the value of 0.15 used previously in the lower-resolution data.

As these high-resolution data files have many more data points, the processing time was far greater than for the previous data files; initially the detection and clustering took approximately one day per file. The current algorithm has been revised for this scenario and functions acceptably with a reasonable balance between speed and memory usage, but even higher-resolution (and therefore larger) data files are achievable with other instruments. It is highly likely that the method of clustering and post-processing may need to be revisited in the future in order to perform well under these circumstances.

Another issue to be addressed in this strategy is the detection of peptide pairs when the corresponding protein is at a low level, or completely absent in either the test or reference samples. In these situations, the pair would not be detected because one of the isotopic patterns would be below the threshold of detection. These peptides (being either down- or up-regulated, respectively) are of great interest, and the corresponding loss of information would have important consequences.

The introduction of a third labelled sample (“triple-labelling”) can potentially solve this problem. This is achieved through the use of deuterated sodium cyanoborohydride and ^{13}C, deuterated formaldehyde, resulting in a mass increase of eight. In the proposed arrangement, the reference sample is split, with one aliquot labelled using the “light” (Δm=0) label and the other with the new “heavy” (Δm=8) label. The test sample is then labelled using the “intermediate” (Δm=4) label, and all three samples combined. In this configuration, the reference samples bracket the test sample and can be used to detect the peptide, after which the test sample's level of expression can be determined based on the intermediate label.

As mentioned, there exists a case where a protein may be down-regulated or absent from the reference. Fortunately, this can also be addressed in one of several ways. In time-course studies, where the actual ratio is not as important as the relative changes, the use of a “pooled” reference could be used (that is, it contains all of the proteins). In other studies, one of the labels could be applied to a mixture of the test and reference, or the roles of the test and reference samples could be reversed (with the 'test' sample's isotopic patterns bracketing that of the reference).

The triple labelling of peaks was also found to yield promising results. For both of these adaptations, the modifications to the original algorithm were minimal, and consisted only of altering the desired models to match the nature of the data – a testament to the flexibility of the approach outlined previously. Higher resolution merely requires models which have a matching peak shape and higher density of data points over the same *m/z* range. Triple labelling experiments required only the adaptation of the model's first vector to include a second isotopic pattern at the correct *m/z* distance. An example of the modified, singly-charged triple-label model at *m/z* 400 is shown below in Figure 22.

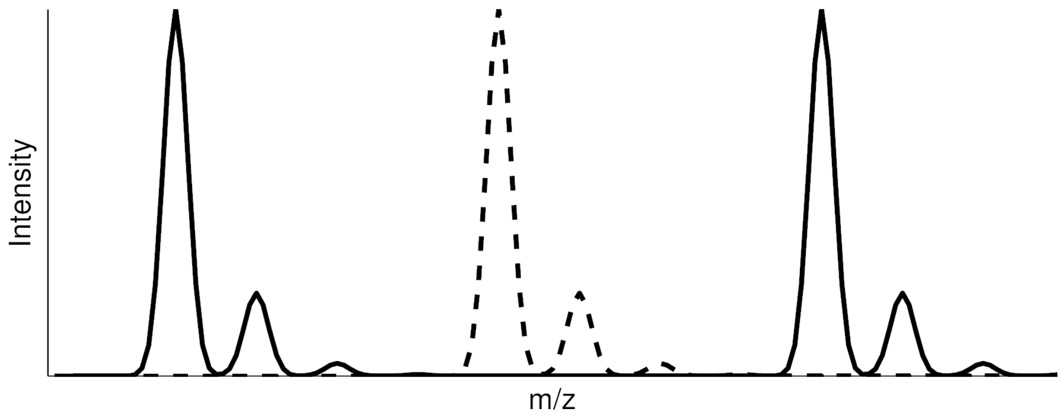


Figure 22: Example of the singly-charged, single-label model (class 1) adapted for triple-label detection. The solid line is the first vector, containing the two outer "detection" peaks, and the dashed line is the second vector corresponding to the second sample's signal.

3.7 Cross-Referencing with SEQUEST Results

The SEQUEST[21] computer software is one tool available to determine the peptide sequences and proteins detected in the mass chromatogram. This is achieved using a library of candidate proteins present in the sample. The sequences of candidate peptides are determined based on the protein sequence and digestion method. Finally, a theoretical fragmentation pattern is generated for these peptides, and compared to the observed tandem MS fragmentation spectrum to determine the best match (and thereby the peptide sequence and source protein).

3.7.1 SEQUEST Results Import

As the results of SEQUEST searches were available for the data files used for algorithm development, these can be used to verify the PITS detections by cross referencing them with SEQUEST detections. The first stage in this process is importing the SEQUEST search results into MATLAB. There are different software applications which can be used to produce these results, and for this work support for both BioWorks and Proteome Discoverer (both from Thermo Scientific, Waltham, MA, USA) was implemented, allowing the use of either format. The import process needed to begin from a common source file, and since both of these tools are capable of saving data in Microsoft Excel's .xls format (and MATLAB is capable of reading these files), the .xls file was selected as a starting point. It was then necessary to extract the relevant information from these files. The layout of these files differs significantly for BioWorks and Proteome Discoverer, and for this reason the .xls importing code was written so as to support importing both file types and create a list of represented peptides and their associated information.

The finished function may be seen in Appendix 14. As separate SEQUEST searches are performed for each of the light and heavy signals of peptide pairs, a single “import” operation takes both of these results files (in .xls format) and imports them together. This is necessary because it is possible for SEQUEST to detect only one of the two isotopically labelled peptides in a pair, and combining the results will give greater coverage than a single search alone. The SEQUEST results table is altered slightly from its original format to provide more relevant data than is included. First, it is necessary to determine the number of labels present in the detection. This can be done by determining the difference between the observed *m/z* value of the detection, and the calculated weight of the unlabelled sequence. Division by 28.03 (for a light label) and 32.06 (for a heavy label) returns a value that should be near an integer value (within round-off error) which corresponds to the number of labels. This is a separate function from the .xls import function and is provided in Appendix 15. The SEQUEST peptide sequence is retained unaltered, as are the charge, and mass of the MH+ peak. The scan numbers at which a peptide was identified are altered slightly, as SEQUEST may either provide a single scan number, or a range of scans separated by a hyphen. These scan numbers are translated into a series of scan "starts" and "ends" which correspond to the first and last scan number at which that SEQUEST entry was detected. For single scan detections, both the start and end value will be the same.

This import process is then repeated for the results table of the SEQUEST search for the peptides labelled with the heavy isotope. Note that the masses reported by SEQUEST in this table are for the MH+ peak of the peptide with the heavy isotopic label, and need to be translated into the corresponding MH+ of the peptide with the light label to be directly comparable to the results of the PITS algorithm. This is done by subtracting the weight difference of the isotopic label pair, which is shown in Equation 6.

$$MH^{+}_{Light}=MH^{+}_{Heavy} - (4*(\#\ Labels)*(Mass(D)-Mass(H))) \qquad (6)$$

Finally, the results tables are combined and appended with a flag indicating whether each SEQUEST result originated from the search for the peptide with either the light or heavy isotopic label.

Post-import, the SEQUEST results table can contain multiple entries for the same peptide, but at different, non-contiguous scan numbers. These entries are processed using the MATLAB combineScans.m function in Appendix 16 which condenses multiple entries for the same peptide into a single entry with a list of the scan numbers at which it was detected. This function also combines the detections of the same peptide but which are labelled with the light and heavy isotopic labels. The flag indicating the detection origin (light or heavy) is retained and stored for reference.

3.7.2 Cross-Referencing SEQUEST and PITS Results

Since the PITS algorithm does not determine the sequence of a peptide it detects, it is not possible to directly cross-reference the results sets. Instead, it is necessary to analyze the properties of the two sets to establish matches. These properties are the charges, number of labels, scan numbers (which correspond to elution time) and masses of the entries. This process can be automated and the MATLAB code is shown in Appendix 17. A SEQUEST entry is considered to match a PITS entry if the mass, detection range (time), charge and number of labels fall within acceptable limits. For the charge and number of labels (i.e. peptide class), the matches must be identical. Additionally, the SEQUEST detection time must overlap with the time range of the PITS entry to be considered a match (increasing this tolerance by any range provided only a small number of additional “matches” which cannot be guaranteed to be a valid match). Finally, the mass-to-charge value from PITS is converted to its corresponding MH+ value, and is considered a match if within four-twelfths of a mass unit. This wider range was necessary because there was some variability between the SEQUEST reported masses and the mass observed by the PITS algorithms. This difference is suspected to result from the alignment of the models to the lower-resolution MS data, in contrast with the higher-resolution (but limited *m/z* range) zoom scans, and the tandem scans used by SEQUEST.

CHAPTER 4 RESULTS AND DISCUSSION

4.1 Introduction

This chapter contains a discussion of the results obtained by processing a series of data files using the described algorithms. The four replicate BSA data files referred to herein are the same utilized in the previous work by Boutilier[36]. Similarly, the yeast (*Saccharomyces cerevisiae*) data file originated from the previous work. Both this data file and the four replicate BSA files were generated from samples which were dimethyl-labelled at a ratio of 1:1. The “high-resolution”, “triple-labelling” and “differential expression” data files analyzed in subsequent sections were prepared by a summer research student, Gemma Regan, and the methods used to obtain these files are described in Chapter 2.

4.2 PITS Detection Results for the Four Replicate BSA Data Files

The results of processing the four BSA data files with the described algorithms and parameters are given in Table 3. The first row corresponds to the total number of peaks which were detected using the PITS algorithm. The previous work by Boutilier[36] found a total of 535 peak pairs pooled across all samples, which is on par with these values, but a slight difference is to be expected, arising from the changes to the PITS algorithm since the previous incarnation. However, not all of these detections are quantifiable; a small selection of these detections are overlapped by other peaks and this interference prevents reliable quantitation in situations where there is an insufficient “clean” (non-overlapping)

Table 3: Summary of the peaks detected using the PITS algorithms outlined previously. Groups of counts separated by solid lines represent those that sum to the total number of detections in the results.

Data File	Aon_1-2.raw	Aon_2-3.raw	Aon_3-4.raw	Aon_4-5.raw
PITS Detections	493	552	540	525
Quantifiable	483	539	516	513
Interference	10	13	24	12
Charge (z)=1	136	179	141	166
Charge (z)=2	254	300	283	287
Charge (z)=3	103	73	116	72
# Labels=1	250	322	267	305
# Labels=2	195	194	218	185
# Labels=3	48	36	55	35

Table 4: Overview of the number of each "class" of peptide detected for the four BSA data files

Data File	Aon_1-2.raw	Aon_2-3.raw	Aon_3-4.raw	Aon_4-5.raw
1 Label, z=1	119	153	121	146
2 Labels, z=1	17	23	20	50
1 Label, z=2	113	150	130	147
2 Labels, z=2	127	136	133	129
3 Labels, z=2	14	14	20	11
1 Label, z=3	18	16	16	12
2 Labels, z=3	51	35	65	36
3 Labels, z=3	34	22	35	24

region to quantify (per the minimum temporal size parameter). In these cases, only 10-20 peaks were not quantifiable. The distribution of charges and labels appears to indicate that doubly charged peptides and single labels are more common. However, this does not provide insight into the combinations of charge and labels which are more commonly observed. These details are provided above in Table 4.

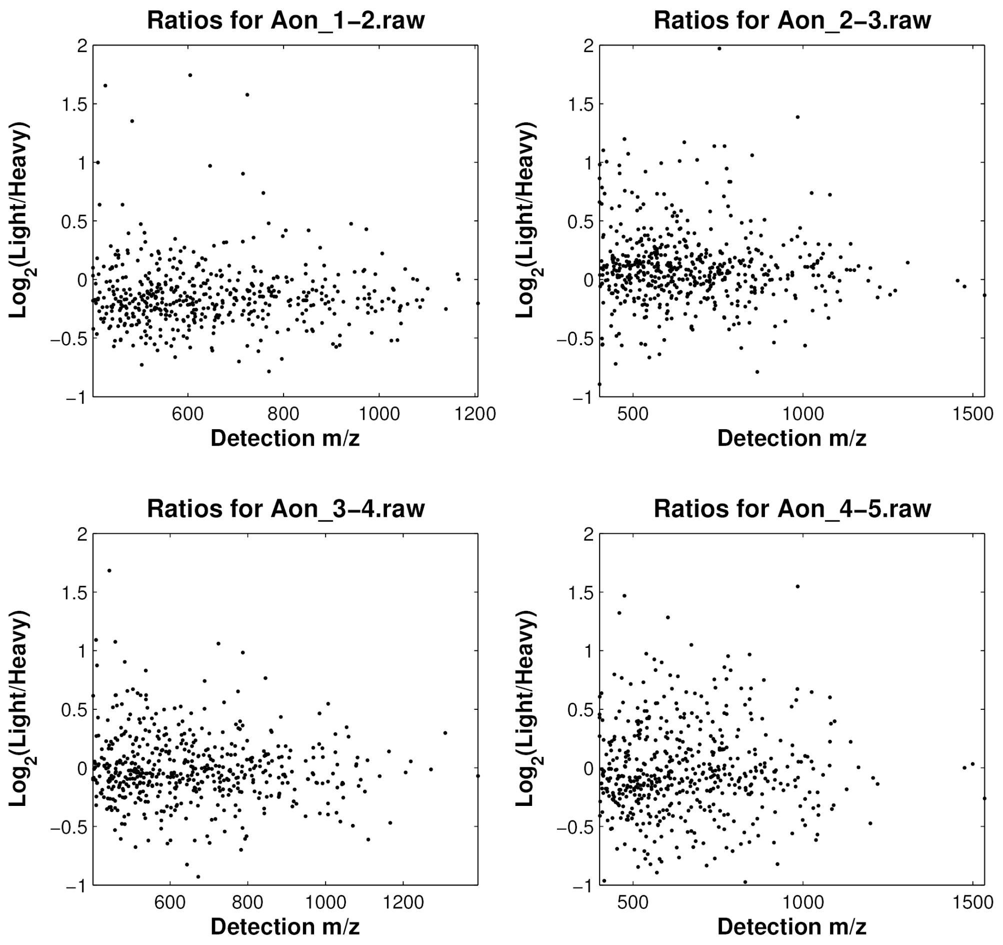


Figure 23: Computed ratios for each of the four replicate BSA data files.

From Table 4 it can be seen that singly charged peptides with one label, and doubly-labelled peptides with a charge of +1 and +2 are the most prevalent. This is expected when considering the amino acids present in peptides; single and doubly-labelled peptides contain either zero or one lysine residue, respectively. Given that the enzyme trypsin cleaves at either lysine or arginine residues this is to be expected. On the other hand, peptides with triple labels would have to contain at least two lysine residues (one n-terminal label and two other labelled primary amines). This situation is only likely to occur if there are multiple missed tryptic cleavages, which will result in peptide chains with additional primary amine groups.

The ratios of the light to heavy isotopic labels of these peaks are shown as a function of the *m/z* value in Figure 23. Overall, the distribution of these ratios appears as expected (spread about the designed ratio of 1:1). There is some slight variability in the spread and

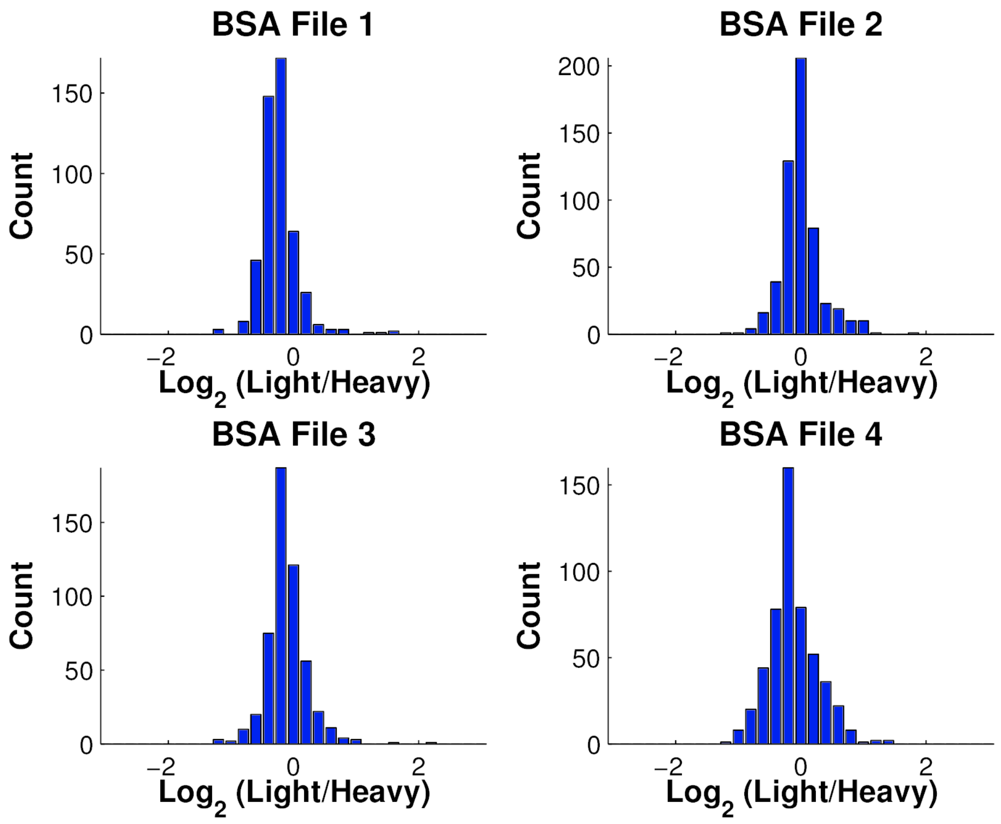


Figure 24: Histograms of the computed ratios for each of the four BSA replicates.

centering. This may be attributable to sample preparation, as the four samples were each digested and labelled separately prior to analysis, as opposed to replicate injections of the same sample. Histograms of these plots are shown in Figure 24, which illustrate this variability, but in general the three central bins do comprise the bulk of the detections, with a reasonably balanced distribution to either side of the center.

4.2.1 Supplementation of PITS Results with SEQUEST

The list of PITS detections was cross-referenced with the results of the previously performed SEQUEST searches for the four BSA replicates. Overall, there were found to be 160, 155, 162, and 153 PITS detections which had “matching” entries in the unfiltered SEQUEST results for replicates 1-4, respectively. This left 323, 384, 354, and 360 detections without matches. It was initially suspected that the large difference between the number of PITS detected peptides and the number which had matching SEQUEST entries could arise from a lack of zoom and tandem scans present on these detections. However, this was not the case, as a large number of the PITS entries without SEQUEST matches did have “candidate” zoom and tandem scans. “Candidate” scans are the zoom and tandem scans which were triggered within the region of the PITS detection – that is, the zoom scan's *m/*z range spans the value detected by PITS, occurred during the time range over which PITS observed the signal, and the tandem scan's parent ion must have a mass which falls within the PITS detection's mass range. While it is highly likely that these scans are indeed triggered on a peptide signal, there are two possibilities that make this difficult to confirm for every matched set of PITS detections and zoom or tandem scans without exhaustive manual investigation. Specifically, there may be more intense signals from non-peptide sources interleaved amongst the isotopic patterns, or the tandem scan was triggered by a peak other than the first peak of the light isotopic pattern. In the latter scenario, this makes it difficult to directly compare *m/z* values, as the tandem scans will report the *m/z* values on which they were triggered, whilst PITS identifies detections by the *m/z* value of the first peak of the light isotopic pattern.

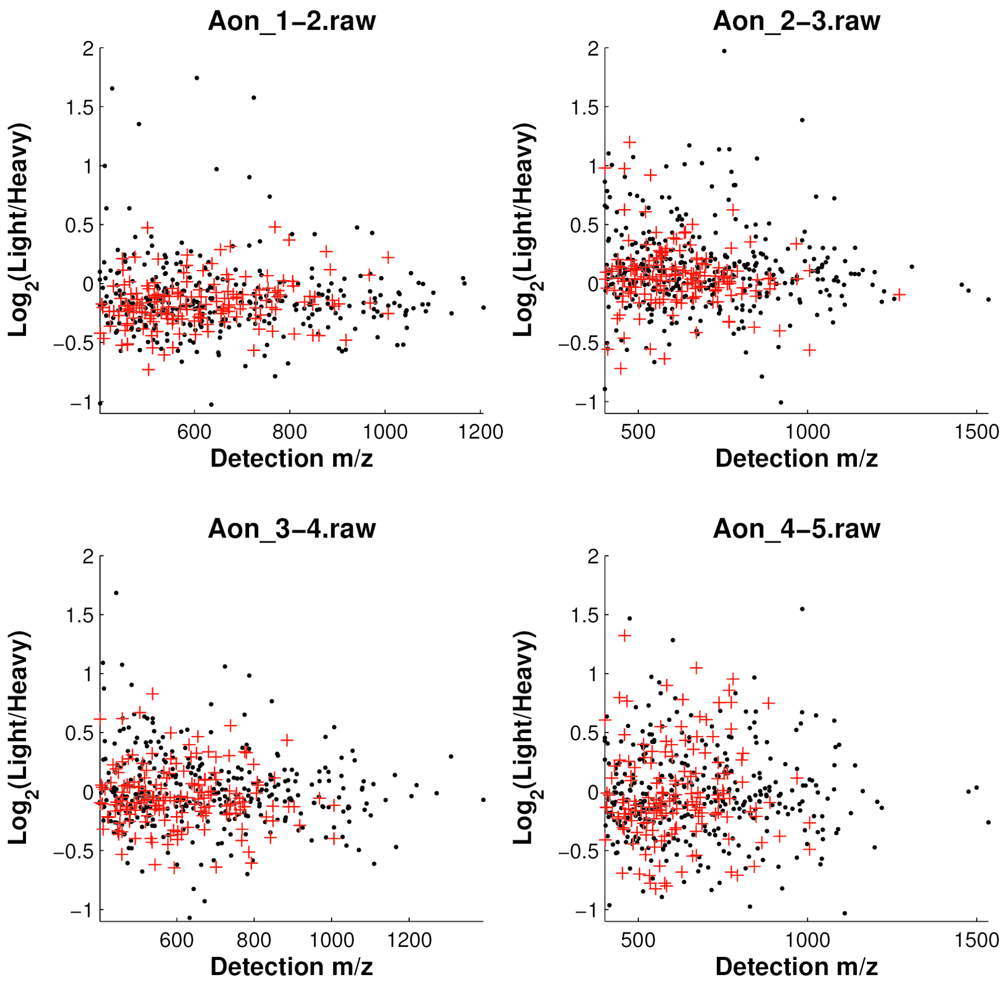


Figure 25: Plot of ratios as a function of m/z with SEQUEST matches indicated. Peptides with matches are plotted as red + symbols, while those without matches are plotted as black points.

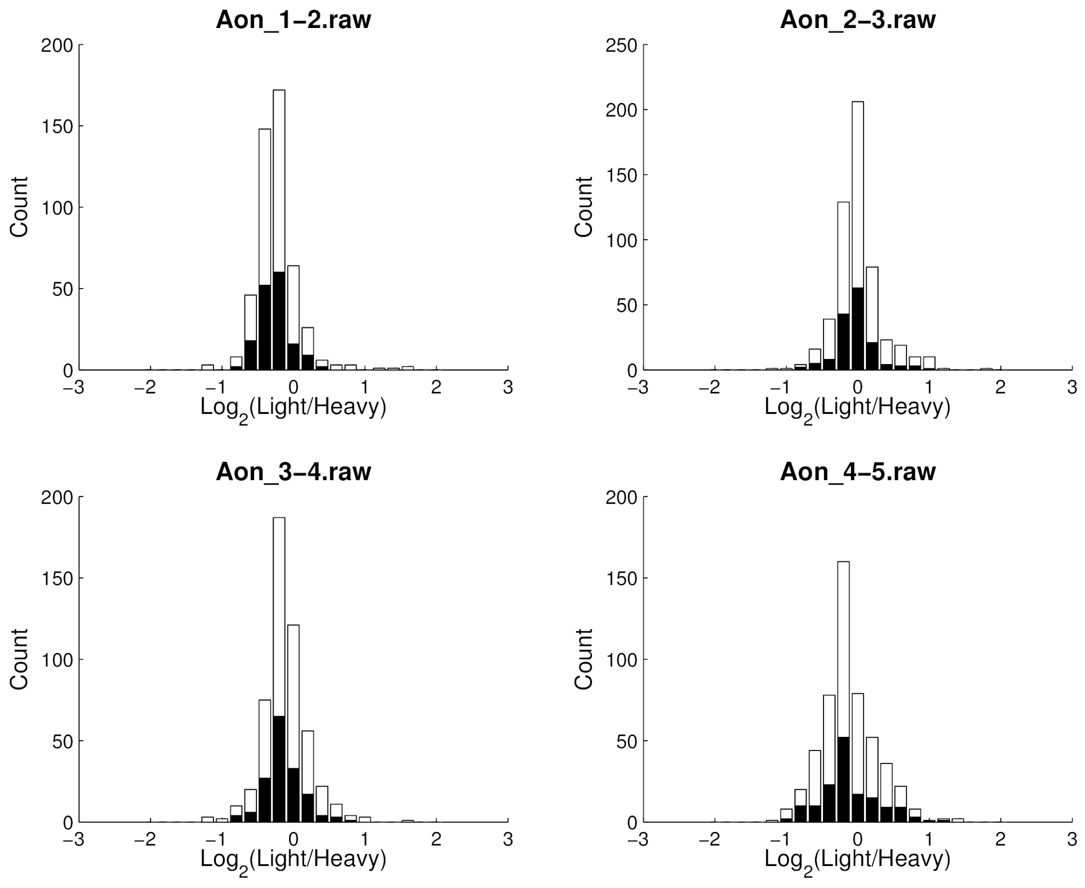


Figure 26: Histograms of the ratios separated by SEQUEST matches. Black bars correspond to detections with matches, white bars correspond to detections without SEQUEST matches.

These PITS detections which are “unaccounted for” exhibit all of the characteristics of their known valid counterparts – that is, the signals fit well to the isotopic peak model, and have fit ratios that are not distinguishable from the other peptides detected. Unfortunately it is not possible to conclude that they are peptides without a positive identification. These detections can be seen in Figure 25, a duplicate of Figure 23, except that detections which have SEQUEST matches are indicated with red '+' symbols as opposed to black dots. It is worth noting that there does not seem to be a clearly visible difference between these two groups with respect to their distribution about the axes, as further evidenced by the histograms in Figure 26, which shows that the distribution among the bins is highly similar for both groups.

The previous work of Boutilier[36] also included a list of 124 “known good” peptides which were established through manual inspection and filtering of the combined SEQUEST results of all four files. This list was also compared to the PITS and SEQUEST results. Four possible cases were considered and explored, consisting of whether the peptide was detected in a given file by PITS, SEQUEST, both methods, or neither method (that is, it was found in at least one data file but not the file being inspected). Peptides detected by PITS and not by SEQUEST will not appear in this analysis, as without identification, there is no possible way to establish their presence across all four files with certainty. Similarly, peptides which are detected by both or neither method are not of particular interest as they show the detection methods agree. The situations of particular interest occur when a peptide is not detected by PITS but is detected by SEQUEST. Only a small number (~10) such detections were found in each file. For these signals, the raw data was extracted and examined in detail. In all such cases, it was observed that the raw MS^1 data was of insufficient quality for PITS to detect the peptide. This was typically due to either signal interference, signals that were too brief in the temporal domain, or simply signals with a low signal-to-noise ratio. This is not unexpected, as the signal quality of zoom and tandem MS scans is generally superior to the raw MS scan. It is anticipated that this situation will improve with higher-resolution MS data, but this cannot be tested on the current instrumentation, since collection of higher-resolution data precludes the collection of tandem MS scans.

4.3 PITS Results for Yeast

Analysis of the yeast data file showed similar results as those observed for the BSA data files. This sample is more complex than the BSA data files as it contains an entire proteome as opposed to a single protein. In this case, a total of 1880 PITS detections were found. Of this total, 1739 were considered “quantifiable”, with 546, 1076, and 258 singly, doubly, and triply charged detections (respectively), 801 detections with 1 label, 878 with two labels, and 201 with 3 labels. Finally, the “class” distribution for these peptides is given in Table 5.

Table 5: Number of detections per peptide class for the *S. cerevisiae* data file.

# Labels	1	2	1	2	3	1	2	3
Charge (z)	1	1	2	2	2	3	3	3
# Detections	342	204	424	601	51	35	73	150

Unfortunately, this data file was prepared as part of the previous work by Boutilier,[36] and did not contain the requisite tandem MS scans for SEQUEST searching and identification, limiting further interpretation of these results, but a plot of isotopic peak ratios is well distributed about the expected 1:1 value and may be seen above in Figure 27, with the corresponding histogram in Figure 28. It is worth noting that the higher number of unquantifiable detections indicates that as the complexity of the sample increases, it will be necessary to adapt the quantification method of the PITS algorithm to handle scenarios where detections overlap.

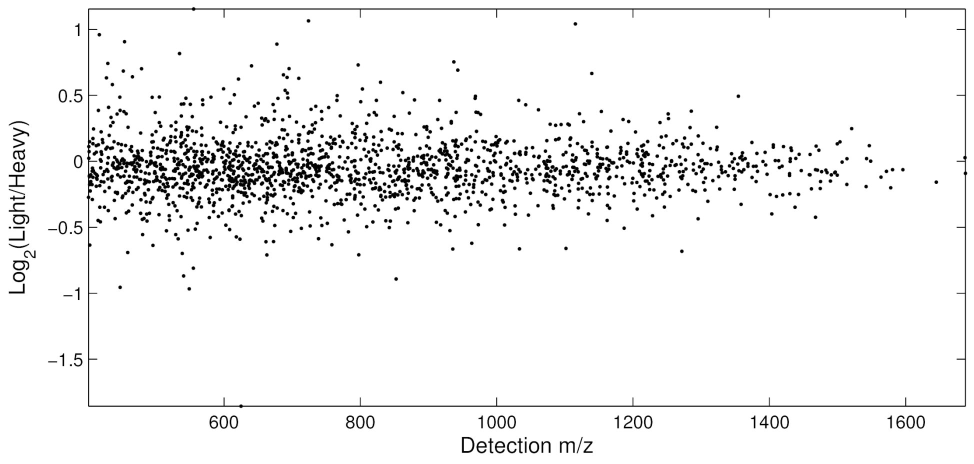


Figure 27: Plot of isotopic peak ratios as a function of *m/z* value for the *S. cerevisiae* data file. The distribution appears normal about the expected ratio of 1:1.

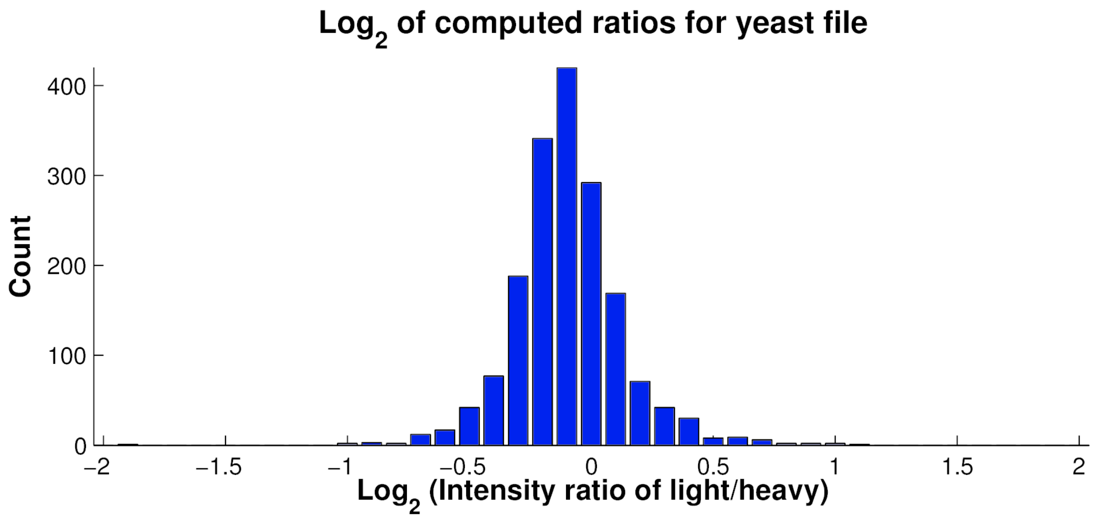


Figure 28: Histogram of the ratios computed for the *S. cerevisiae* data file.

4.4 PITS Results for High-Resolution Data

Six replicate injections of a sample of BSA labelled at a ratio of 1:1 were collected by undergraduate researcher Gemma Regan, with the instrument configured in a higher resolution mode (*m/z* increments of 1/50) and disabling the collection of zoom and tandem scans. The six data files collected at higher resolution were also analyzed with the same detection settings as the lower-resolution files (excepting the peak width parameter used in modelling). The counts for these six files are shown in Table 6.

Table 6: Counts of each type for the six higher-resolution BSA data files. Groups of entries separated by lines are those which sum to the total number of detections.

Data File	BSA_C1_ 01.raw	BSA_C2_ 02.raw	BSA_C1_ 03.raw	BSA_C2_ 04.raw	BSA_C1_ 05.raw	BSA_C2_ 06.raw
Total #	149	148	156	158	160	151
Quantifiable	149	148	156	155	158	151
z=1	57	52	58	60	59	56
z=2	84	84	88	85	91	83
z=3	8	12	10	13	10	12
1 label	74	64	78	69	81	70
2 labels	69	75	72	81	73	74
3 labels	6	9	6	8	6	7
1 Label, z=1	53	48	56	53	58	52
2 Labels, z=1	4	4	2	7	1	4
1 Label, z=2	20	15	21	15	22	17
2 Labels, z=2	61	64	64	66	66	63
3 Labels, z=2	3	5	3	4	3	3
1 Label, z=3	1	1	1	1	1	1
2 Labels, z=3	4	7	6	8	6	7
3 Labels, z=3	3	4	3	4	3	4

It is worth observing that these results appear to be more consistent across the six files for the same sample. This may be attributable to the higher resolution of the data resulting in a less ambiguous fit to the models for incompletely resolved isotopic patterns in relation to the alignment artifacts described in Chapter 3. The number of peptides

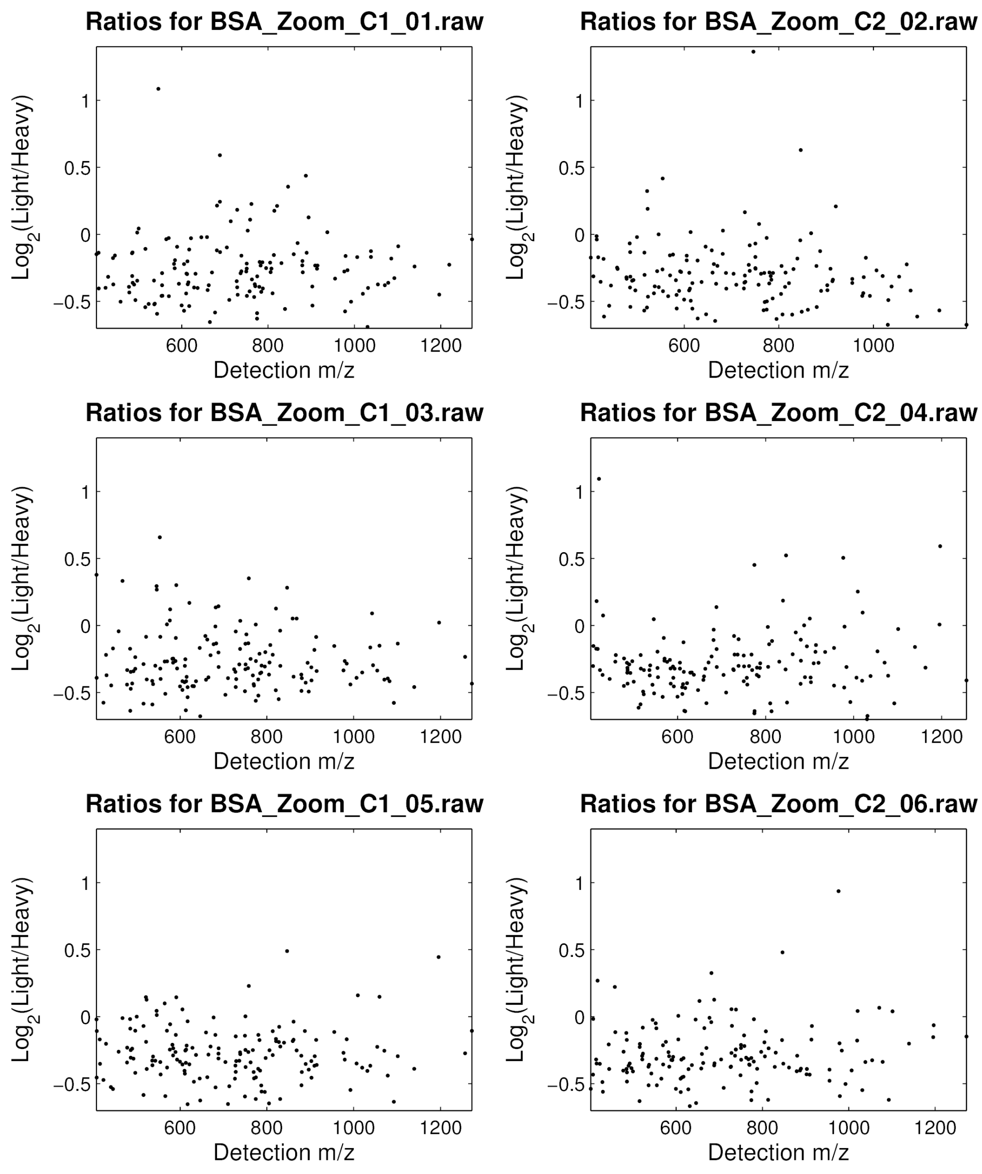


Figure 29: Plots of the intensity ratios for each detection as a function of the *m/z* value, for each of the six higher-resolution BSA data files.

observed here appears to be significantly lower than for the lower-resolution BSA files, and the reason for this discrepancy remains to be investigated during the continuation of this work. One possibility is that the higher resolution results in a greater sensitivity to any variations in the peak width, causing a lack of fit to the model and a lower than anticipated signal-to-noise ratio. This arises during the calculation of the residuals during fitting, as the residual error increases rapidly for small differences in peak width as the overall peak shape becomes narrower. The possibility of false detections in the lower-resolution data also cannot be excluded, especially given that these numbers are more consistent with the SEQUEST results. However, the consistency of the ratios in the lower-resolution data files contradicts this possibility. Finally, a variation in instrument sensitivity between the two sets of experiments is also a possible explanation.

Plots of the detection ratios for each of the six files may be seen in Figure 29. It is worth observing that there appears to be a downward shift in the ratios which is consistent across the files, which is indicative that the samples were likely not exactly at a 1:1 ratio when introduced into the MS instrument. Further analysis of the reproducibility of signal ratio calculations for replicate injections is planned once the alignment of points across multiple data files has been implemented.

4.5 PITS Results for Triple-Labelling Data

The triple-labelling of peptides is a method that should resolve the problem of detection when only a single isotopic pattern of the peak pair is present. As such, an exploratory sample of BSA was prepared at a ratio of 1:1 for the reference to test, per the methods described in Chapter 2. The data was collected at the standard resolution used for the four replicate BSA files (*m/z* increments of 1/12). The ratio plots may be seen below in Figure 30 and the summary results are shown subsequently in Table 7.

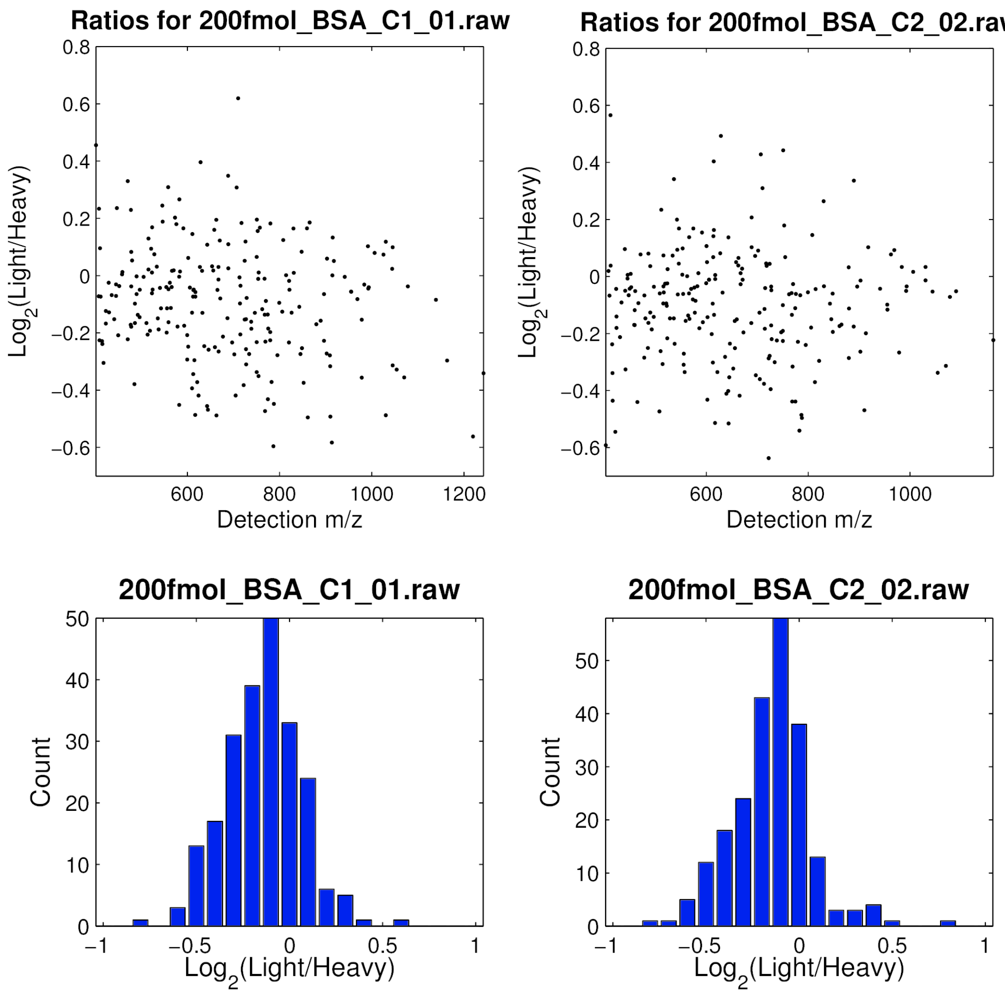


Figure 30: Histograms and plots of detected ratios as a function of m/z value for the two data files containing three isotopic labels.

The ratios shown above were determined based on the intensity of the two outer "reference" peaks over the intensity of the middle "test" peak, using the model modification described in the previous chapter. As can be seen from the plots, theoretical ratios are consistent with the designed value of 1:1. It is interesting to note that the total number of detections is higher than observed in the higher-resolution data files. It appears that this increase results from an increase in the number of peptides detected at higher charge states. It has been observed previously that at standard resolution, the

Table 7: Summary of peptide detection counts for the two BSA files with three isotopic labels.

Data File	200fmol_BSA_C1_01.raw	200fmol_BSA_C2_02.raw
Total #	226	231
Quantifiable	225	225
z=1	45	46
z=2	147	148
z=3	34	37
1 label	90	96
2 labels	111	106
3 labels	25	29
1 Label, z=1	37	37
1 Label, z=2	8	9
2 Labels, z=1	48	54
2 Labels, z=2	91	82
2 Labels, z=3	8	12
3 Labels, z=1	5	5
3 Labels, z=2	12	15
3 Labels, z=3	17	17

isotopic pattern of these signals is not as clearly distinguished as singly-charged peptides. Ergo, it is possible that the addition of the third label provides better detection in cases where the fit of only two isotopic patterns might otherwise be discarded for having a signal-to-noise ratio that was too low. As the sample was prepared at a 1:1 ratio, it does not offer any insight as to the detection abilities when a peptide may be down-regulated or up-regulated. While it is anticipated that this would provide results similar to the differential labelling experiment described in the next section, it should be noted that the addition of a third label does increase the complexity of a sample by a factor of 1.5, and this will require careful experimental design when a triple-label method is applied to a sample which is already complex, such as a complete proteome.

4.6 Detection of Differential Expression

The experiments reported up until this point have all used protein ratios of 1:1 in order to illustrate the ability of the algorithm to locate peptide pairs and report a reliable ratio. However, the algorithm must be able to detect differential protein expression to be useful. To demonstrate this ability, an experiment was designed in which up- and down-regulation of proteins was simulated amongst a complex proteome background.

Three mixtures were prepared as described earlier in Chapter 2,and two replicate runs were carried out in low-resolution mode (with tandem MS scans). These data files were then analyzed using PITS and SEQUEST in an attempt to confirm these differential expression ratios could be detected correctly. The initial results of these experiments were promising. Through cross-referencing detections with SEQUEST results, several points with expression ratios close to 3:1 were shown to have peptide sequences which originate from the lysozyme protein, as shown in Figure 31. While the target ratio of 2:1 is small and can be overlapped by a small number of ratios corresponding to the background which was prepared at 1:1, it was selected with the intent to establish whether such a small ratio resulted in a visually distinct grouping of points in the ratio plots, and to provide a visual reference for the range of concentration ratios that could be practically observed as deviating from 1:1 during a differential expression experiment.

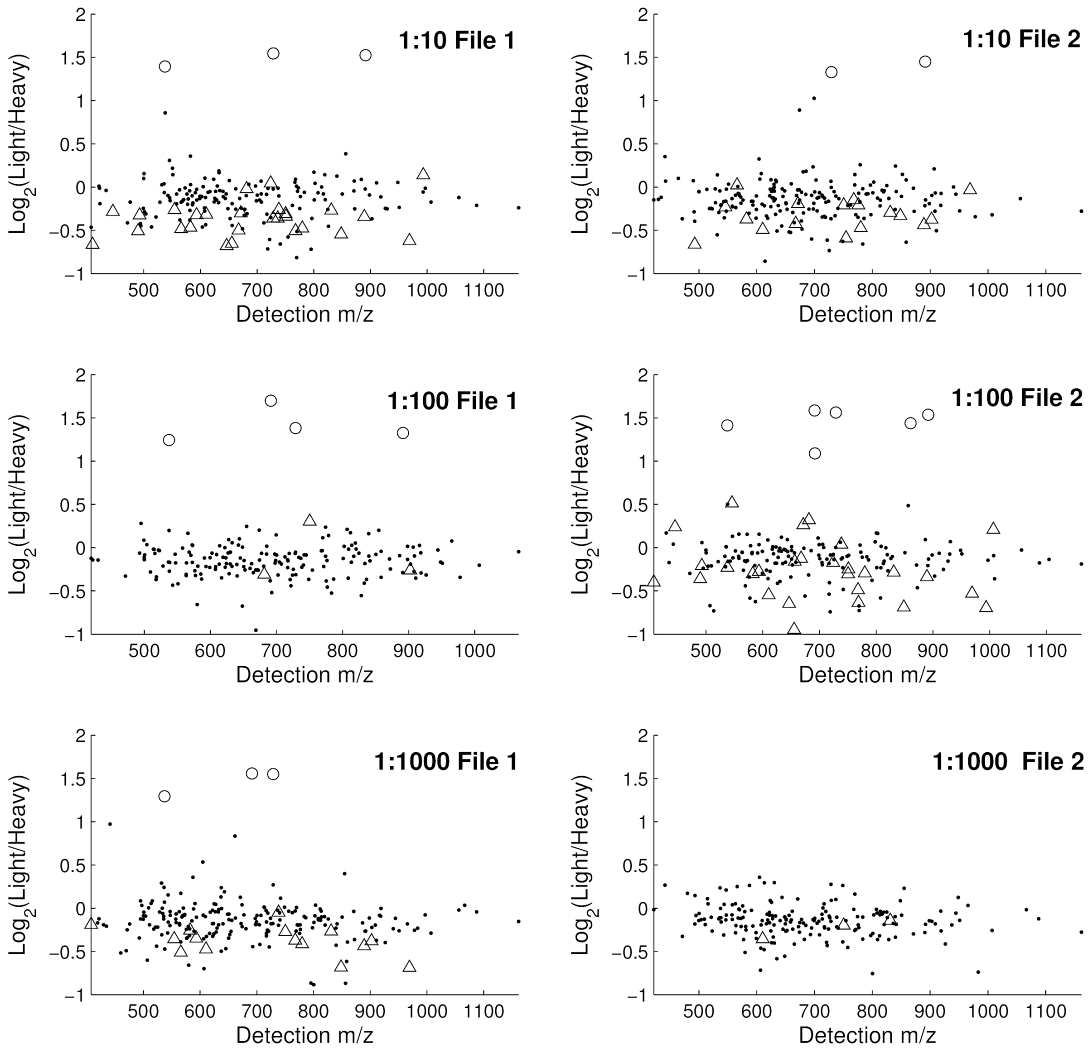


Figure 31: Plots of detected expression ratios as a function of *m/z*, grouped by organism. Lysozyme is represented with circles, BSA with triangles, and *E. coli* with solid points. Note the distinct grouping and separation of the lysozyme circles (where present).

Note that lysozyme is a relatively small protein yielding only a few peptides on digestion. Therefore, it was not expected that many matches would be observed, if at all. This is likely the reason that the experiments with the smallest amount of lysozyme failed to produce any matches in one of the replicates, as only 21 femtomoles of this protein is present, and divided 2:1 amongst the light and heavy labelled variants. Regardless, several matches which are consistently up-regulated have been found in the other runs. Unfortunately, the points identified as originating from BSA did not appear as a distinct group as expected. Instead, they are interspersed with the *E. coli* “background”. It is suspected that the origin of this error lies in the normalization of samples, as three pairs of light and heavy samples must be prepared separately prior to final dilution and combination for LC-MS/MS analysis.

The same experiment was repeated in an attempt to confirm the differential expression for both up-regulation and down-regulation. The results of this second set of experiments are shown in Figure 32.

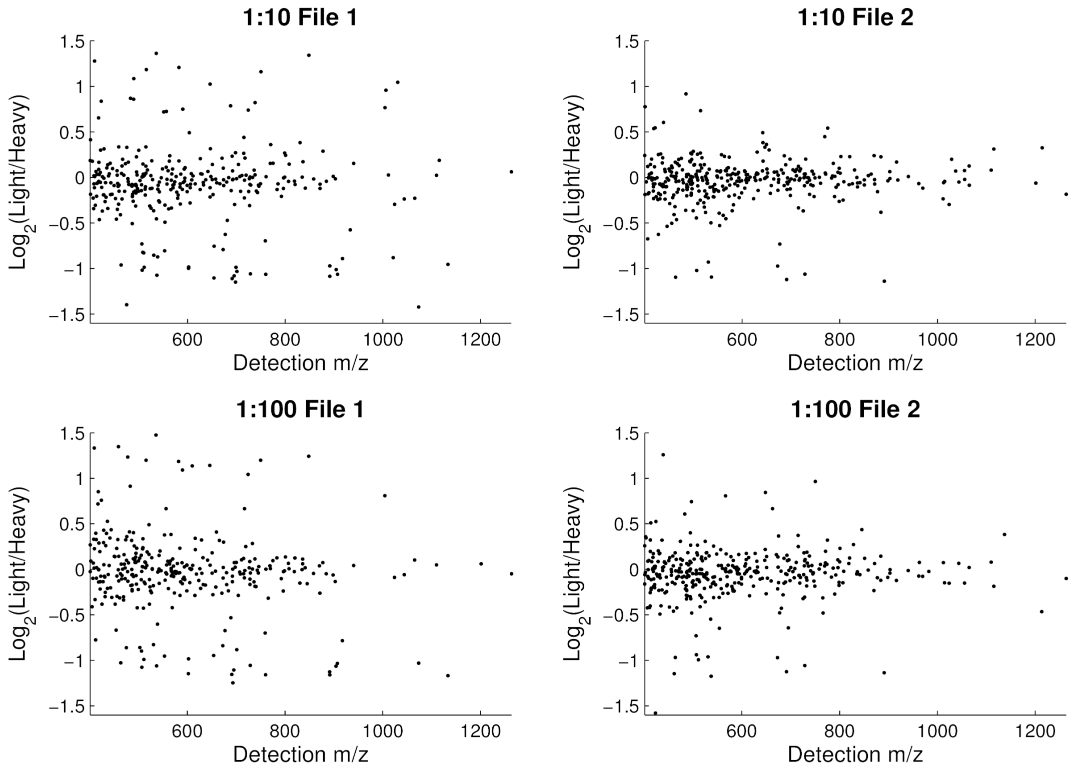


Figure 32: Calculated ratios for the second set of BSA, lysozyme, and *E. coli* samples.

In this experiment, the ratios of BSA and lysozyme were inverted (e.g. BSA was up-regulated at 2:1, and lysozyme was down-regulated at 1:2) to ensure that the problem did not originate specifically with BSA. The results clearly display groups of peptides around the appropriate ratios, but a definitive confirmation cannot be made, as the data collection was inadvertently run using the higher-resolution method which was not configured to collect MS^2 data. Ergo, no tandem MS data was collected and SEQUEST results could not be obtained to identify the differentially expressed proteins. It would appear highly likely that the patterns shown are consistent with the expected results, given that the masses of the up-regulated peptides identified as lysozyme (*m/z* 537.3, 728.8, 891.4 and 691.4) in the first experiment are also observed within one mass channel (1/12 *m/z*) in the second attempt, but with down-regulated ratios per the experimental design. Further experiments are planned to complete this validation.

CHAPTER 5 CONCLUSIONS AND FUTURE WORK

5.1 CONCLUSIONS

The methods outlined in this work have been shown to improve and extend the functionality of the PITS method of detecting peptides in MS data without the use of tandem MS data or the identification of the exact sequence of the peptide. Specifically, it has shown successful detection of peptides using customized algorithms to determine the anticipated isotopic ratios of a peptide based on the formula of an “average” amino acid, rapid initial detection of peptide pair signals in raw data from the MS instrument, and successful filtering and clustering of this initial detection to provide a final list of peptide detections which performs comparably to SEQUEST, while also detecting peptide peak pairs which are singly charged or not recognized from the database. Finally, it has shown success in quantifying the ratios of the intensities of these isotopically labelled peptide peak pairs in the raw data.

The software developed in this work has been compiled into a self-contained package to guide the user through each of the steps in the workflow. Information on prerequisites and how to use the package are provided in Appendix 18. The main interface of this package is depicted in Figure 33, and each “block” in this workflow can be clicked to open a dialog with additional information and instructions, or to run that particular step of the process.

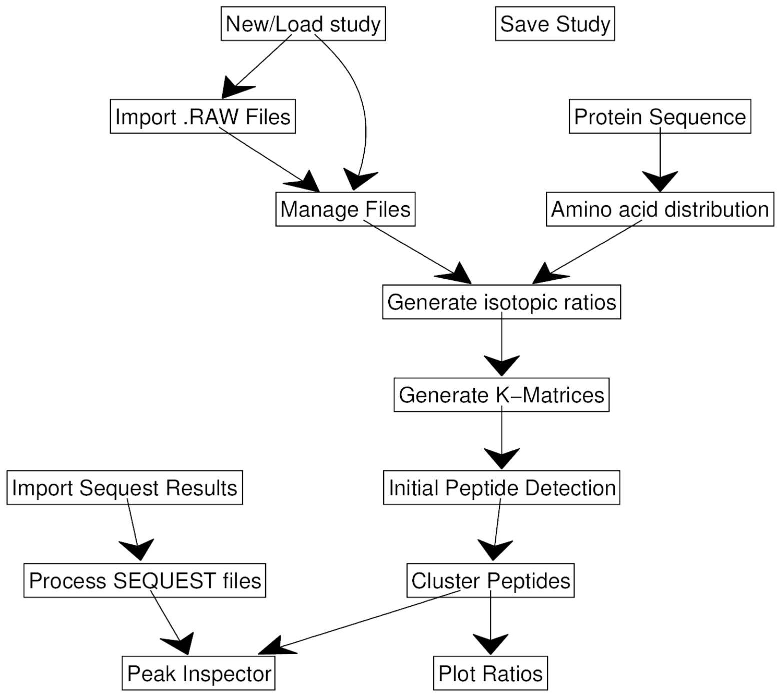


Figure 33: Main interface display of PITS software package

5.2 FUTURE WORK

The algorithms in this work have not yet reached their full potential, and much work remains to be completed before this can happen. There are many possible applications and future extensions for this work, and current research also focuses on expanding the capabilities of the workflow into several new regions. The preliminary experiments using non-uniform expression ratios, triple labels, and higher-resolution data have been shown to be successful, but further validation is necessary.

One aspect that requires further investigation is the metric used to determine the quality of the model fit. A relative fit error is currently used to express the signal-to-noise ratio, but this may not be the best approach. The noise in MS signals can be dependent on signal intensity, which translates to larger residuals for larger signals. A more appropriate evaluation might be based on weighted residuals. Accordingly, it will also be necessary to characterize the baseline and signal noise, and develop error models to estimate the reliability of the ratios calculated from the mass chromatographic data.

While the software speed is currently less of a bottleneck than the collection of the MS data, improvements are possible to increase the efficiency of the various stages, and this will become of significant importance as the size of data files increases, either in resolution or in the temporal direction.

It is anticipated that higher resolution, triple-labelling experiments such as those described in this work will provide the best results, but the analysis of such data has additional challenges in the model sensitivity to changes in peak width and position. Currently, these procedures will need to be investigated, as it is not clear whether they are optimal. Extending this concept even further, more complications are introduced when the data becomes of sufficient resolution that individual isotope peaks are resolved, or that one may observe the interleaving of two or more signals within the *m/z* range of a single isotopic pattern. This will necessitate the adaptation of the modelling methods to process these regions appropriately and prevent these signals from interfering with each other when quantifying the intensities.

Rather than limiting studies to single mass chromatograms, the goal is to apply these methods to larger-scale studies, In particular, these methods are targeted to longitudinal studies such as time-course analysis to discern large-scale changes in protein expression as opposed to biomarker studies. This will require the integration of multiple sets of data, so the time alignment of peptide detections is a desirable feature to be incorporated into this workflow, which will require evaluating existing methods for suitability, or implementing a new approach that integrates with the developed workflow. Early during the development of this work, one possible method was tested which relied on the use of PCA to correct time shifts from one sample to the next, but the complications arising from establishing relationships between data files were found to be more complex than anticipated. Therefore, the exploration of additional time alignment methods and quantitation of detections over time remains as a candidate for future research to further enhance the workflow.

Triple labelling, which was shown to be a viable approach when used together with the PITS algorithm, also requires further investigation. The best method for implementing the third label in the design of an experiment remains for future consideration, but the advantage of the PITS method in conjunction with this labelling method is that it can still be used to detect peptides when the isotope clusters are overlapped, and it does not preclude the use of more labels.

Finally, to validate this approach, it will be necessary to conduct an experiment on a larger experimental system of relevance. The goal of this experiment would be to demonstrate that meaningful information can be extracted through the multivariate data obtained in the experiment. Currently, such an experiment is underway to investigate the evolution of protein expression during yeast sporulation. Approximately twenty samples have been collected during this process by Gemma Regan, an undergraduate researcher, and these samples await preparation for analysis.

REFERENCES

1. Wilkins, M. R., Pasquali, C., Appel, R. D., Ou, K., Golaz, O., Sanchez, J., Yan, J. X., Gooley, A. A., Hughes, G., Humphery-Smith, I., Williams, K., Hochstrasser, D. F, From proteins to proteomes: Large scale protein identification by two-dimensional Electrophoresis and amino acid analysis, *Biotechnology*, **14**:**61** (1996)

2. Gygi, S., Rochon, Y., Franza, B., Aebersold, R., Correlation between protein and mRNA abundance in yeast, *Molecular and Cellular Biology*, **19 (3)**:**1720** (1999)

3. Anderson, N. L., Anderson, N. G., Proteome and proteomics: New technologies, new concepts, and new words, *Electrophoresis*, **19**:**1853** (1998)

4. Liu, Y., Huttengain, R., Collins, B., Aebersold, R., Mass spectrometric protein maps for biomarker discovery and clinical research, *Expert Review of Molecular Diagnostics*, **13 (8)**:**811** (2013)

5. Marshall, R., Simpson, J., Lukey, P., Strategies for biomarker discovery in fibrotic disease, *Biochimica et Biophysica Acta - Molecular Basis of Disease*, **1832 (7)**:**1079** (2013)

6. Arnaud, C., Biomarkers wanted, *Chemical and Engineering News*, **89 (30)**:**40** (2011)

7. Qin, X., Ling, B., Proteomics studies in breast cancer (Review), *Oncology Letters*, **3**:**735** (2012)

8. Shoemaker, L., Achrol, A., Sethu, P., Steinberg, G., Chang, S., Clinical neuroproteomics and biomarkers: From basic research to clinical decision making, *Neurosurgery*, **70 (3)**:**518** (2011)

9. Yates, J., Ruse, C., Nakorchevsky, A., Proteomics by mass spectrometry: approaches, advances, and applications, *Annual Review of Biomedical Engineering*, **11**:**49** (2009)

10. Han, X., Aslanian, A., Yates, J., Mass spectrometry for proteomics, *Current Opinion in Chemical Biology*, **12**:**483** (2008)

11. Malmström, J., Beck, M., Schmidt, A., Lange, V., Deutsch, E., Aebersold, R., Proteome-wide cellular protein concentrations of the human pathogen *Leptospira interrogans*, *Nature*, **460 (6)**:**762** (2009)

12. Gerber, S., Rush, J., Stemman, O., Kirschner, M., Gygi, S., Absolute quantification of proteins and phosphoproteins from cell lysates by tandem MS, *Proceedings of the National Academy of Sciences USA*, **100 (12)**:**6940** (2003)

13. Nesvizhskii, A., Vitek, O., Aebersold, R., Analysis and validation of proteomic data generated by tandem mass spectrometry, *Nature Methods*, **4 (10)**:**787** (2007)

14. Wall, M. J., Crowell, A. M. J., Simms, G. A., Liu, F., Doucette, A. A., Implications of partial tryptic digestion in organic-aqueous solvent systems for bottom-up proteome analysis, *Analytica Chimica Acta*, **703**:**194** (2011)

15. Gygi, S. P., Rist, B., Gerber, S.A., Turecek, F., Gelb, M. H., Aebersold, R., Quantitative analysis of complex protein mixtures using isotope-coded affinity tags, *Nature Biotechnology*, **17**:**994** (1999)

16. Ong, S., Blagoev, B., Kratchmarova, I., Kristensen, D. B., Steen, H., Pandey, A., Mann, M., Stable isotope labeling by amino acids in cell culture, SILAC, as a simple and accurate approach to expression proteomics, *Molecular and Cellular Proteomics*, **1**:**376** (2002)

17. Wang, Y., Ma, Z., Quinn, D., Fu, W., Inverse 18O labeling mass spectrometry for the rapid identification of marker/target proteins, *Analytical Chemistry*, **73**:**3742** (2001)

18. Yu, L., Bottari, P., Turecek, F., Aebersold, R., Gelb, M., Absolute quantitation of specific proteins in complex mixtures using visible isotope-coded affinity tags, *Analytical Chemistry*, **76**:**4104** (2004)

19. Hansen, K., Schmitt-Ulms, G., Chalkley, R., Hirsch, J., Baldwin, M., Burlingame, A., Mass spectrometric analysis of protein mixtures at low levels using cleavable 13C-isotope-coded affinity tag and multidimensional chromatography, *Molecular and Cellular Proteomics*, **2**:**299** (2003)

20. Thompson, A., Schafer, J., Kuhn, K., Kienle, S., Schwarz, J., Schmidt, G., Neumann, T., Hamon, C., Tandem mass tags: A novel quantification strategy for comparative analysis of complex protein mixtures by MS/MS, *Analytical Chemistry*, **75**:**1895** (2003)

21. Eng, J. K., McCormack, A. L., Yates, J. R. III, An Approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database, *Journal of the American Society for Mass Spectrometry*, **5 (11**):**976** (1994)

22. Perkins, D., Pappin, D., Creasy, D., Cottrell, J., Probability-based protein identification by searching sequence databases using mass spectrometry data, *Electrophoresis*, **20**:**3551** (1999)

23. Craig, R., Beavis, R., TANDEM: matching proteins with tandem mass spectra, *Bioinformatics*, **20 (9)**:**1466** (2004)

24. Ross, P. L., Huang, Y. N., Marchese, J. M., Williamson, B., Parker, K., Hattan, S., Khainovski, N., Pillai, S., Dey, S., Daniels, S., Purkayastha, S., Juhasz, P., Martin, S., Bartlet-Jones, M., He, F., Jacobson, A., Pappin, D. J., Multiplexed protein quantitation in *Saccharomyces cerevisiae* using amine-reactive isobaric tagging reagents, *Molecular and Cellular Proteomics*, **3**:**1154** (2004)

25. Lundgren, D., Hwang, S., Wu, L., Han, K.,, Role of spectral counting in quantitative proteomics, *Expert Reviews Proteomics*, **7 (1**):**39** (2010)

26. Zhu, W., Smith, J., Huang, C., Mass spectrometry based label-free quantitative proteomics, *Journal of Biomedicine and Biotechnology*, **2010** (2009)

27. Liu, H., Sadygov, R., Yates, J., A model for random sampling and estimation of relative protein abundance in shotgun proteomics, *Analytical Chemistry,* **76 (14):4193** 2004

28. Julka, S., Regnier, F., Quantification in proteomics through stable isotope coding: A review*, Journal of Proteome Research,* **3**:**350** (2004)

29. Yasui, Y., McLerran, D., Adam, B., Winget, M.,Thornquist, M., Feng, Z., An automated peak identification/calibration procedure for high-dimensional protein measures from mass spectrometers, *Journal of Biomedicine and Biotechnology*, **4**:**242** (2003)

30. Morris, J., Coombes, K., Koomen, J., Baggerly, K., Kobayashi, R., Feature extraction and quantification for mass spectrometry in biomedical applications using the mean spectrum, *Bioinformatics*, **21 (9)**:**1764** (2005)

31. Samuelsson, J., Dalevi, D., Levander, F., Rögnvaldsson, T., Modular, scriptable and automated analysis tools for high-throughput peptide mass fingerprinting, *Bioinformatics*, **20 (18**):**3628** (2004)

32. Noy, K., Fasulo, D., Improved model-based, platform-independent feature extraction for mass spectrometry, *Bioinformatics*, **23 (19**):**2528** (2007)

33. James, P., Quadroni, M., Carafoli, E., Gonnet, G., Protein identification by mass profile fingerprinting, *Biochemical and Biophysical Research Communications,* **195 (1)**:**58** (1993)

34. Henzel, J., Watanabe, C., Stults, J., Protein identification: The origins of peptide mass fingerprinting, *Journal of the American Society for Mass Spectrometry,* **14**:**931** (2003)

35. Yates, J., Speicher, S., Griffin, P., Hunkapiller, T., Peptide mass maps: A highly informative approach to protein identification, *Analytical Biochemistry,* **214**:**397** (1993)

36. Boutilier, Joseph M., Strategies to Impove Quantitative Proteomics: Implications of Dimethyl Labeling and Novel Peptide Detection (Masters Thesis), 2012 (Available via DalSpace (URI: http://hdl.handle.net/10222/14558))

37. Reisch, M., Tiny tools, *Chemical and Engineering News*, **91 (34):11** (2013)

38. Hsu, J., Huang, S., Chow, N., Chen, S., Stable-isotope dimethyl labeling for quantitative proteomics, *Analytical Chemistry*, **75 (24)**:**6843** (2003)

39. Melanson, J., Avery, S., Pinto, D. M., High-coverage quantitative proteomics using amine-specific isotopic labeling, *Proteomics*, **6**:**4466** (2006)

40. Boutilier, J.M., Warden, H., Doucette, A. A., Wentzell, P. D., Chromatographic behaviour of peptides following dimethylation with H2/D2-formaldehyde: Implications for comparative proteomics, *Journal of Chromatography B,* **908**:**59**

(2012)

41. The Quiagen guide to good microbial practice, 1999, http://www.pointbiolabs.com/uploads/5/0/9/8/5098737/___guide_to_good_microbiological_practices_iii.pdf (Last accessed December 16, 2013)

42. Pedrioli, P., Eng, J. Hubley, R., Vogelzang, M., Deutsch, E., Raught, B., Pratt, B., Nilsson, E., Angeletti, R., Apweiler, R., Cheung, K., Costello, C., Hermjakob, H., Huang, S., Julian, R., Kapp, E., McComb, M., Oliver, S., Omenn, G., Paton, N., Simpson, R., Smith, R., Taylor, C., Zhu, W., Aebersold, R., A common open representation of mass spectrometry data and its application to proteomics research, *Nature Biotechnology*, **22 (11):1459** (2004)

43. Tasman, N., Eng, J., Pratt, B., Chambers, M., ReAdW 4.3.1 (Computer software), http://tools.proteomecenter.org/wiki/index.php?title=Software:ReAdW (Last accessed December 17, 2013)

APPENDIX 1 HIGH-RESOLUTION MS ACQUISITION PARAMETERS

MS Run Time (min): 65.00
Sequence override of method parameters not enabled.
Divert Valve: not used during run
Contact Closure: not used during run
Syringe Pump: not used during run
MS Detector Settings:Real-time modifications to method disabled
Stepped collision energy not enabled
Additional Microscans:

MS2 0
MS3 0
MS4 0
MS5 0
MS6 0
MS7 0
MS8 0
MS9 0
MS10 0

Segment 1 Information
Duration (min): 65.00
Number of Scan Events: 1
Tune Method: June19_2013_NSITune
Scan Event Details:

1: ITMS + p norm !corona !pi oZ(400.0-1300.0)

Custom Data Dependent Settings: Not enabled

APPENDIX 2 TRIPLE-LABEL MS ACQUISITION PARAMETERS

MS Run Time (min): 65.00
Sequence override of method parameters not enabled.
Divert Valve: not used during run
Contact Closure: not used during run
Syringe Pump: not used during run
MS Detector Settings:Real-time modifications to method disabled
Stepped collision energy not enabled
Additional Microscans:
 MS2 0
MS3 0
MS4 0
MS5 0
MS6 0
MS7 0
MS8 0
MS9 0
MS10 0
Experiment Type: Nth Order Triple Play
Tune Method: May24_2013_NSITune
Scan Event Details:
    1: ITMS + p norm !corona !pi o(400.0-1300.0)
    2: ITMS + p norm !corona !pi Dep Zoom MS Most intense ion from (1)
Activation Type: CID
Min. Signal Required: 5000.0
Isolation Width: 2.00
Normalized Coll. Energy: 35.0
Default Charge State: 2
Activation Q: 0.250
Activation Time: 30.000
    3: ITMS + c norm !corona !pi Dep MS/MS Most intense ion from (2)
Activation Type: CID
Min. Signal Required: 5000.0
Isolation Width: 2.00
Normalized Coll. Energy: 35.0
Default Charge State: 2
Activation Q: 0.250
Activation Time: 30.000
Scan Events 2 and 3 repeated for top 3 peaks.
Data Dependent Settings: Use separate polarity settings disabled
Parent Mass List: (none)
Reject Mass List: (none)
Neutral Loss Mass List: (none)

Neutral loss in top: 3
Most intense if no parent masses found not enabled
Add/subtract mass not enabled
Charge state screening not enabled
Charge state rejection enabled
Unassigned charge states : rejected
Charge state 1 : rejected
Charge state 2 : not rejected
Charge state 3 : not rejected
Charge states 4+ : not rejected
Global Data Dependent Settings:
Use global parent and reject mass lists not enabled
Exclude parent mass from data dependent selection not enabled
Exclusion mass width by mass
Exclusion mass width low: 1.50
Exclusion mass width high: 2.50
Parent mass width by mass
Parent mass width low: 0.50
Parent mass width high: 0.50
Reject mass width by mass
Reject mass width low: 0.50
Reject mass width high: 0.50
Zoom/UltraZoom scan mass width by mass
Zoom/UltraZoom scan mass width low: 5.00
Zoom/UltraZoom scan mass width high: 5.00
Neutral Loss candidates processed by decreasing intensity
Neutral Loss mass width by mass
Neutral Loss mass width low: 0.50
Neutral Loss mass width high: 0.50
MS mass range: 0.00-1000000.00
MSn mass range by mass
MSn mass range: 0.00-1000000.00
Analog UV data dep. not enabled
Dynamic exclusion enabled
Repeat Count: 1
Repeat Duration: 30.00
Exclusion List Size: 50
Exclusion Duration: 120.00
Exclusion mass width by mass
Exclusion mass width low: 1.50
Exclusion mass width high: 2.50
Expiration: disabled
Isotopic data dependence not enabled
Custom Data Dependent Settings: Not enabled

APPENDIX 3 ALTERNATE RATIO MS ACQUISITION PARAMETERS

MS Run Time (min): 65.00
Sequence override of method parameters not enabled.
Divert Valve: not used during run
Contact Closure: not used during run
Syringe Pump: not used during run
MS Detector Settings:Real-time modifications to method disabled
Stepped collision energy not enabled
Additional Microscans:
MS2 0
MS3 0
MS4 0
MS5 0
MS6 0
MS7 0
MS8 0
MS9 0
MS10 0
Experiment Type: Nth Order Triple Play
Tune Method: June19_2013_NSITune
Scan Event Details:
1: ITMS + p norm !corona !pi o(400.0-1300.0)
2: ITMS + p norm !corona !pi Dep Zoom MS Most intense ion from (1)
Activation Type: CID
Min. Signal Required: 5000.0
Isolation Width: 2.00
Normalized Coll. Energy: 35.0
Default Charge State: 2
Activation Q: 0.250
Activation Time: 30.000
3: ITMS + c norm !corona !pi Dep MS/MS Most intense ion from (2)
Activation Type: CID
Min. Signal Required: 5000.0
Isolation Width: 2.00
Normalized Coll. Energy: 35.0
Default Charge State: 2
Activation Q: 0.250
Activation Time: 30.000
Scan Events 2 and 3 repeated for top 3 peaks.
Data Dependent Settings: Use separate polarity settings disabled
Parent Mass List: (none)
Reject Mass List: (none)
Neutral Loss Mass List: (none)

```
Neutral loss in top: 3
Most intense if no parent masses found not enabled
Add/subtract mass not enabled
Charge state screening not enabled
Charge state rejection enabled
Unassigned charge states : rejected
Charge state 1 : rejected
Charge state 2 : not rejected
Charge state 3 : not rejected
Charge states 4+ : not rejected
Global Data Dependent Settings:
Use global parent and reject mass lists not enabled
Exclude parent mass from data dependent selection not enabled
Exclusion mass width by mass
Exclusion mass width low: 1.50
Exclusion mass width high: 2.50
Parent mass width by mass
Parent mass width low: 0.50
Parent mass width high: 0.50
Reject mass width by mass
Reject mass width low: 0.50
Reject mass width high: 0.50
Zoom/UltraZoom scan mass width by mass
Zoom/UltraZoom scan mass width low: 5.00
Zoom/UltraZoom scan mass width high: 5.00
Neutral Loss candidates processed by decreasing intensity
Neutral Loss mass width by mass
Neutral Loss mass width low: 0.50
Neutral Loss mass width high: 0.50
MS mass range: 0.00-1000000.00
MSn mass range by mass
MSn mass range: 0.00-1000000.00
Analog UV data dep. not enabled
Dynamic exclusion enabled
Repeat Count: 1
Repeat Duration: 30.00
Exclusion List Size: 50
Exclusion Duration: 120.00
Exclusion mass width by mass
Exclusion mass width low: 1.50
Exclusion mass width high: 2.50
Expiration: disabled
Isotopic data dependence not enabled
Custom Data Dependent Settings:Not enabled
```

APPENDIX 4 MZXML IMPORT FUNCTION

```
%% Scan mzXML file into matlab for all MS,Zoom,MSMS scans
function [dat]=rawexportall_new(mzXML,savename)
% mzXML and savename needs to be string variables
% mzXML is the filename of the file being exported
% savename is the filename that you want to save the data as

tic;
fid = fopen(mzXML,'r+'); % read in mzXML data file
fContents=fread(fid,'*char');
fclose(fid);

main=fContents';
toCut=(main==' ' | main==char(10));
main=main(~toCut);

tokens = regexp(main,'<\?xmlversion="(.*?)</dataProcessing>','tokens');
header = tokens{1}{1,1};

 tokens = regexp(header,'fileName="(.*?)"','tokens');
dat.fileName=tokens{1}{1,1};
tokens = regexp(header,'scanCount="(.*?)"','tokens');
dat.scanCount=str2double(tokens{1}{1,1});

FulCount=1;
ZCount=1;
TCount=1;

match = regexp(main,'<scannum="(.*?)</peaks>','match');
nscan=length(match);
for iscan = 1:nscan;

    tokens = regexp(match{1,iscan},'msLevel="(.*?)"','tokens');
    msLevel=str2double(tokens{1}{1,1});

    tokens = regexp(match{1,iscan},'scanType="(.*?)"','tokens');
    scanType=tokens{1}{1,1};

    %Full Scans
    msLevelCheck=msLevel==1;
    scanTypeCheck=strncmp(scanType,'Full',4);
    if msLevelCheck && scanTypeCheck == true
    tokens = regexp(match{1,iscan},'compressedLen="0">(.*?)</peaks>',...
        'tokens');
    rawdata=tokens{1}{1,1};

    base64 = base64cvt(rawdata);
    DATA=swapbytes(typecast(base64,'single'));

    tokens = regexp(match{1,iscan},'scannum="(.*?)"','tokens');
    curscan=str2double(tokens{1}{1,1});
    dat.scanNum(1,FulCount)=curscan;

    tokens = regexp(match{1,iscan},'retentionTime="PT(.*?)S"','tokens');
    dat.rTime(1,FulCount)=str2double(tokens{1}{1,1});

    DATA_R=reshape(DATA,2,[])';
    dat.intensity(:,FulCount)=DATA_R(:,2);
    FulCount=FulCount+1;
    if FulCount == 2
        dat.massCharge(:,1)=DATA_R(:,1);
    end
end

%Zoom Scans
msLevelCheck=msLevel==1;
scanTypeCheck=strncmp(scanType,'Z',1);
if msLevelCheck && scanTypeCheck == true

    tokens = regexp(match{1,iscan},'compressedLen="0">(.*?)</peaks>',...
        'tokens');
    rawdata=tokens{1}{1,1};

    base64 = base64cvt(rawdata);
    [~,col]=size(base64);
    data_length=col/4;
    DATA=swapbytes(typecast(base64,'single'));

    tokens = regexp(match{1,iscan},'scannum="(.*?)"','tokens');
    curscan=str2double(tokens{1}{1,1});
    dat.scanNumZ(1,ZCount)=curscan;

    tokens = regexp(match{1,iscan},'retentionTime="PT(.*?)S"','tokens');
    dat.rTimeZ(1,ZCount)=str2double(tokens{1}{1,1});

    DATA_R=reshape(DATA,2,[])';
```

```
        dat.intensityZ(:,ZCount)=DATA_R(1:data_length/2,2);
        dat.massChargeZ(:,ZCount)=DATA_R(1:data_length/2,1);
        ZCount=ZCount+1;
    end

    %Tandem MS Scans
    msLevelCheck=msLevel==2;
    scanTypeCheck=strncmp(scanType,'Full',4);
    if msLevelCheck && scanTypeCheck == true

        tokens = regexp(match{1,iscan},'compressedLen="0">(.*?)</peaks>',...
            'tokens');
        rawdata=tokens{1}{1,1};

        base64 = base64cvt(rawdata);
        [~,col]=size(base64);
        data_length=col/4;
        DATA=swapbytes(typecast(base64,'single'));

        tokens = regexp(match{1,iscan},'scannum="(.*?)"','tokens');
        curscan=str2double(tokens{1}{1,1});
        dat.scanNumT(1,TCount)=curscan;

        tokens = regexp(match{1,iscan},'retentionTime="PT(.*?)S"','tokens');
        dat.rTimeT(1,TCount)=str2double(tokens{1}{1,1});

        tokens = regexp(match{1,iscan},'filterLine="ITMS(.*?)@cid','tokens');
        value=tokens{1}{1,1};
        value=value(14:end);
        %token should look like '+cNSIdFullms2NUMBER' and I need that NUMBER
        dat.MassIonT(1,TCount)=str2double(value);

        tokens=regexp(match{1,iscan},'precursorCharge="(.*?)"','tokens');
        dat.MassIonCharge(1,TCount)=str2double(tokens{1}{1,1});

        DATA_R=reshape(DATA,2,[])';
        dat.intensityT{TCount}=DATA_R(1:data_length/2,2)';
        dat.massChargeT{TCount}=DATA_R(1:data_length/2,1)';
        TCount=TCount+1;
    end
end

dat.runTime=toc/60;
save(savename,'dat')
```

APPENDIX 5 BASE64 DECODING FUNCTION

```
% Function to decompress base-64 xml data to binary (byte) format.
%
function byteout=base64cvt(charlst)
b64codes=ones(122,1)*-1; % starting off vector with the length of the number of ASCII
codes
b64set=[65:90 97:122 48:57 43 47]'; %ASCII codes for base 64 set
b64vals=[0:63]'; %base 64 values see (http://en.wikipedia.org/wiki/Base64)
b64codes(b64set)=b64vals; %a vector of base 64 codes where they are in ASCII


padding=(charlst=='=');
charCodes=uint8(b64codes(charlst(~padding)));
charCodes(padding)=0;
bitCodes=dec2bin(charCodes);
eightBitGroups=reshape(bitCodes',8,[])';
fastDecimal=eightBitGroups=='1'; %Do a faster conversion back to decimal than bin2dec
decodedB64=uint8(fastDecimal*[128 64 32 16 8 4 2 1]')';
if any(padding)
    last3Bytes=decodedB64(end-2:end);
    decodedB64(end-2:end)=0;
    nPadding=sum(padding);
    nShift=3-sum(padding);
    decodedB64(end-1:end-2+nShift)=last3Bytes(1:nShift);
    if nPadding==2
        decodedB64(end)=[];
    end
end

byteout=decodedB64;
```

APPENDIX 6 AVERAGE AMINO ACID FUNCTION

```
%PQ_MODULE
% ----- Begin Config. DO NOT EDIT! -----
%Config:NodeName='Amino acid distribution'
%Config:NodePrev='@NONE'
%Config.NodeNext='isotopeRatioGen.m'
%Config.NodeVarI='study'
%Config.NodeVarO='study'
%ENDCONFIG
% ----- End Config -----

function [study, success]=AADistribution(study)
success=false;
%% if txtfile is NOT in matlab already and file only has amino acids in txtfile (BSA)
%BSAseq.txt is the Amino Acid sequence for the Full length BSA precursor protein of
607 amino acids
%BSAseq2.txt is the Amino Acid sequence for the mature BSA protein of 583 amino
acids
%****Should use the BSAseq2.txt****

%old way
% function [AmNum,MW,ElementSummary,AAratio]=aminodis(txtFile,savename)

%Changed on July 21,2011 to use AvgAAMet.m which needs these variable to
%calulate the elemental composition of a peptide at a given MW

%type == 1 for BSA
%type == 2 for Yeast

proteinSequence=study.preliminary.proteinSequence;
load aminoData;
%% Calculate the number of each amino acid
proteinSequence=regexprep(proteinSequence,'\W','');
aminoCount=arrayfun(@(aaChar) sum(proteinSequence==aaChar),
char(aminoProperties(:,1)));

%% Calculate the relative and isotopic molecular weight
relMW=round((aminoCount'*cell2mat(aminoProperties(:,4)))+((2*1.00795)+15.9994));
isoMW=round((aminoCount'*cell2mat(aminoProperties(:,5)))+
((2*1.007825)+15.994915));
MW=[relMW isoMW];

%% Calculate the total amount of Carbon, Hydrogen, Nitrogen, Oxygen and Sulfur
```

```
elementCounts=cell2mat(aminoProperties(:,6:end));
Carbon=(aminoCount'*elementCounts(:,1));
Hydrogen=(aminoCount'*elementCounts(:,2));
Nitrogen=(aminoCount'*elementCounts(:,3));
Oxygen=(aminoCount'*elementCounts(:,4));
Sulfur=(aminoCount'*elementCounts(:,5));
Element=[Carbon Hydrogen Nitrogen Oxygen Sulfur];

%% Calculate the percentage of Carbon, Hydrogen, Nitrogen, Oxygen and Sulfur
CarbonPcnt=Carbon/sum(Element); HydrogenPcnt=Hydrogen/sum(Element);
NitrogenPcnt=Nitrogen/sum(Element); OxygenPcnt=Oxygen/sum(Element);
SulfurPcnt=Sulfur/sum(Element);
ElementPcnt=[CarbonPcnt HydrogenPcnt NitrogenPcnt OxygenPcnt SulfurPcnt];

%% Calculate the average number of Carbon, Hydrogen, Nitrogen, Oxygen and Sulfur
AvgCarbon=mean(aminoCount.*elementCounts(:,1));
AvgHydrogen=mean(aminoCount.*elementCounts(:,2));
AvgNitrogen=mean(aminoCount.*elementCounts(:,3));
AvgOxygen=mean(aminoCount.*elementCounts(:,4));
AvgSulfur=mean(aminoCount.*elementCounts(:,5));
AvgElement=[AvgCarbon AvgHydrogen AvgNitrogen AvgOxygen AvgSulfur];

ElementSummary=[Element; AvgElement; ElementPcnt;];

%% Calculate the ratios of each amino acid to the max and to the total
ratTot=aminoCount./sum(aminoCount);
ratMax=aminoCount./max(aminoCount);
AAratio=[ratTot'; ratMax';];

%% Calculate the discrete mean values
NumCarbon=Carbon/sum(aminoCount); NumHydrogen=Hydrogen/sum(aminoCount);
NumNitrogen=Nitrogen/sum(aminoCount);
NumOxygen=Oxygen/sum(aminoCount); NumSulfur=Sulfur/sum(aminoCount);
ElementNum=[NumCarbon NumHydrogen NumNitrogen NumOxygen NumSulfur];
ElementNorm=ElementNum/min(ElementNum);
MassElement=[12.00 1.007825 14.00307 15.994915 31.972071];
AvgAminoMass=ElementNum*MassElement';

%% Plot the Amino Acid Distribution
figure('Color','w','name','Amino Acid distribution')
axes('FontWeight','bold','FontSize',12)
bar(aminoCount)
set(gca,'XTick',1:1:22)
set(gca,'XTickLabel',{char(aminoProperties(:,1))})
```

```
xlabel('Amino Acid')
axis('tight')

study.preliminary.distribution.aminoCount=aminoCount;
study.preliminary.distribution.avgAminoMass=AvgAminoMass;
study.preliminary.distribution.elementNorm=ElementNorm;
study.preliminary.distribution.elementNum=ElementNum;
study.preliminary.distribution.elementMass=MassElement;
study.preliminary.distribution.aminoSymbols=char(aminoProperties(:,1));
%success is set to false, so that if a return is issued before the script
%completes, a failure is displayed.

%processing done. Set success to true so that a completion is indicated.
success=true;
```

APPENDIX 7 ISOTOPE POSSIBILITIES TABLE FUNCTION

```
% This function precomputes the distribution of possible isotopic
% combinations that comprises an M+i isotopic peak. We do this by iterating
% over all possiblities from 0 to 10  of each element, looking at the final
% mass difference with the base peak, and establishing the molecular weight
% difference from the base. This difference is used to determine which
% "bin" the isotope combination is a part of.
% No input
% Output: 1x10 cell array, with the ith cell containing all possible
% element isotope combinations that give rise to an M+i peak.

function XplusI=isotopcalc()
C=10; H=10; N=10; O=10; S=10;
[MWest]=isotopweight(C,0,H,0,N,0,O,0,0,S,0,0,0);
count=zeros(1,10);
percent=0;
for C13=0:C;
   for H2=0:H;
      for N15=0:N;
         display(['Done: ', num2str(percent),'%'])
         percent=percent+0.1;
         for O17=0:O;
            for O18=0:O;
               for S33=0:S;
                  for S34=0:S;
                     for S36=0:S;
                        [MolWeight]=isotopweight(C-C13,C13,H-H2, ...
                           H2,N-N15,N15,O-O17-O18,O17,O18, ...
                           S-S33-S34-S36,S33,S34,S36); %verison 3
                        diff=MolWeight-MWest;
                        if diff<=10 && diff>0
                           count(diff)=count(diff)+1;
                           XplusI{diff}(count(diff),:)=[C13,H2, ...
                              N15,O17,O18,S33,S34,S36];
                        end
                     end
                  end
               end
            end
         end
      end
   end
end
```

APPENDIX 8 FORMULA TO ISOTOPIC RATIOS FUNCTION

```
% ISOTOPEST2 is a script that will calculate isotopic ratios for a
% particular molcular weight using the analytical solution of the problem.

%Inputs: Array containing molecular formula for number of carbon, hydrogen,
% nitrogen, oxygen and sulfur in the atom, of the form [C H N O S]
% Output: 1x11 vector of the isotopic peak heights normalized to the base
% peak.

function [Mtot]=isotopest2(element)
C=element(1);
H=element(2);
N=element(3);
O=element(4);   %Separate the isotopes for easy conceptualization in code.
S=element(5);

% Attempt to load in a data file containing all of the possible
% combinations of isotopes that yield an M+I peak. If file is not there,
% generate it using isotopcalc().
persistent XplusI
if (isempty(XplusI))
   try
      load('isotopdata_10','XplusI');
   catch error
      XplusI=isotopcalc();
      save('isotopdata_10','XplusI');
   end
end

%Housekeeping - preallocating vars.
rowXI=cell(1,10);
XI=cell(1,10);

% For each of the isotopic peaks, establish which patterns are invalid -
% that is, they have more isotopes of an element than the input formula.
% Delete the offending rows, and store the size for later.
for i=1:10
    Car=XplusI{i}(:,1);
   Csum=sum(Car,2)>C;
    Hyd=XplusI{i}(:,2);
   Hsum=sum(Hyd,2)>H;
   Sul=XplusI{i}(:,6:8);
   Ssum=sum(Sul,2)>S;
```

```
    Oxy=XplusI{i}(:,4:5);
    Osum=sum(Oxy,2)>O;
    Nsum=XplusI{i}(:,3)>N;
    badRows=(Ssum) | (Osum) | (Nsum) | (Csum) | (Hsum);
    XI{i}=XplusI{i};
    XI{i}(badRows>0,:)=[];
    [rowXI{i},~]=size(XI{i});
end

% More houskeeping. Could use some lemon pledge.
XplusI2=cell(1,10);
rowplusI=cell(1,10);
hmplusI=cell(1,10);
mplusI=cell(1,10);
MplusI=cell(1,10);

% Now rebuild the table of isotope count possiblities to INCLUDE the
% number of the baseisotope present from the input formula.
for j=1:10
    for i = 1:rowXI{j}
        XplusI2{j}(i,:)=[C-XI{j}(i,1),XI{j}(i,1),H-XI{j}(i,2), ...
            XI{j}(i,2),N-XI{j}(i,3),XI{j}(i,3),O-XI{j}(i,4)-XI{j}(i,5),...
            XI{j}(i,4),XI{j}(i,5),S-XI{j}(i,6)-XI{j}(i,7)-XI{j}(i,8),...
            XI{j}(i,6),XI{j}(i,7),XI{j}(i,8)];
    end
    [rowplusI{j},~]=size(XplusI2{j});
end

hm=[C 0 H 0 N 0 O 0 0 S 0 0 0];
[M]=multnomdis(hm); %Calculate height of the M peak.

% Now iterate and compute the height of the M+jth peak, over each of the i
% possiblities for that peak.
for j=1:10
    for i = 1:rowplusI{j};
        hmplusI{j}=XplusI2{j}(i,:);
        [mplusI{j}]=multnomdis(hmplusI{j});
        MplusI{j}(i)=mplusI{j};
    end
end
% Sum and normalize to base peak over all 10, and return.
Mtot=[M/M, sum(MplusI{1})/M, sum(MplusI{2})/M, sum(MplusI{3})/M,  ...
    sum(MplusI{4})/M, sum(MplusI{5})/M, sum(MplusI{6})/M, ...
    sum(MplusI{7})/M, sum(MplusI{8})/M, sum(MplusI{9})/M, sum(MplusI{10})/M];
```

APPENDIX 9 MULTINOMIAL DISTRIBUTION FUNCTION

```
%% Multinomial Distribution calculation for isotopic distribution
%Input: Vector of isotopic counts of elements, in form:
% [C12 C13 H1 H2 N14 N15 O16 O17 O18 S32 S33 S34 S36]
% Output:
% Probability of that combination based on elemental probabilities.

function [M]=multnomdis(element)

C12=element(1); C13=element(2); Ctot=C12+C13;
H1=element(3); H2=element(4); Htot=H1+H2;
N14=element(5); N15=element(6); Ntot=N14+N15;
O16=element(7); O17=element(8); O18=element(9); Otot=O16+O17+O18;
S32=element(10); S33=element(11); S34=element(12); S36=element(13);
Stot=S32+S33+S34+S36;

PC12=0.9889; PC13=0.0111;
PH1=0.99985; PH2=0.00015;
PN14=0.9964; PN15=0.0036;
PO16=0.9976; PO17=0.0004; PO18=0.0020;
PS32=0.9500; PS33=0.0076; PS34=0.0422; PS36=0.0002;
 if Htot <= 170;
    MH=fastFactorial(Htot)/(fastFactorial(H1)*fastFactorial(H2));
 else
        dif=Htot-H1;
    if dif == 0
        MH=1;
    else
        Htop=[Htot-dif+1:1:Htot];
        Htop=cumprod(Htop);
        Mtop=Htop(end);
        Hbot=[1:1:dif];
        Hbot=cumprod(Hbot);
        Mbot=Hbot(end);
        MH=Mtop/Mbot;
    end
 end

if Ctot <= 170
    MC=fastFactorial(Ctot)/(fastFactorial(C12)*fastFactorial(C13));
else
    dif=Ctot-C12;
    if dif == 0
```

```
        MC=1;
    else
        Ctop=[Ctot-dif+1:1:Ctot];
        Ctop=cumprod(Ctop);
        Mtop=Ctop(end);
        Cbot=[1:1:dif];
        Cbot=cumprod(Cbot);
        Mbot=Cbot(end);
        MC=Mtop/Mbot;
    end
end

if Ntot <= 170
    MN=fastFactorial(Ntot)/(fastFactorial(N14)*fastFactorial(N15));
else
    dif=Ntot-N14;
    if dif == 0
        MN=1;
    else
        Ntop=[Ntot-dif+1:1:Ntot];
        Ntop=cumprod(Ntop);
        Mtop=Ntop(end);
        Nbot=[1:1:dif];
        Nbot=cumprod(Nbot);
        Mbot=Nbot(end);
        MN=Mtop/Mbot;
    end
end

MO=fastFactorial(Otot)/(fastFactorial(O16)*fastFactorial(O17)* ...
    fastFactorial(O18));
MS=fastFactorial(Stot)/(fastFactorial(S32)*fastFactorial(S33)* ...
    fastFactorial(S34)*fastFactorial(S36));
M1=MC*MH*MN*MO*MS;

M2=(PC12^C12)*(PC13^C13)* ...
   (PH1^H1)*(PH2^H2)*...
   (PN14^N14)*(PN15^N15)* ...
   (PO16^O16)*(PO17^O17)*(PO18^O18)* ...
   (PS32^S32)*(PS33^S33)*(PS34^S34)*(PS36^S36);

M=M1*M2;
```

APPENDIX 10 FAST FACTORIAL FUNCTION

```
function fx=fastFactorial(x)
persistent factorials; %generate persistent lookup factorial table
if (x==0)
    fx=1;
    return;
end
if (isempty(factorials)) % No table? Build it up to 100!
    factorials(1)=1;
    fastFactorial(100);
end
if (numel(factorials)<x)
    for i=numel(factorials)+1:x
        factorials(i)=factorials(i-1)*i;
    end
end
fx=factorials(x);
```

APPENDIX 11 GAUSSIAN MODEL FUNCTION

```
%PQ_MODULE
% ----- Begin Config. DO NOT EDIT! -----
%Config:NodeName='Generate K-Matrices'
%Config:NodePrev='isotopeRatioGen.m'
%Config.NodeNext='@NONE'
%Config.NodeVarI='study'
%Config.NodeVarO='study'
%ENDCONFIG
% ----- End Config -----
% The above lines MUST be present as the first thing in every module. The
% order of the variables is not critical. See GUIsample.m for details on
% the configuration variables above.

% This is a sample of a non-gui script module. The main function can use
% hard-coded variables, or varargin/varargout as appropriate, but the
% latter may be more difficult to interpret.

function [study,success]=Kmatrix_gen(study)
success=false;
ElementTot=study.preliminary.isotopeRatios.ElementTot;
Mtot=study.preliminary.isotopeRatios.Mtot;
Mass=study.preliminary.isotopeRatios.Mass;
MassIndx=study.preliminary.isotopeRatios.MassIndx;


mass_sigma=0.15;
mass_delta=[1 1 0.5 0.5 0.5 1/3 1/3 1/3];


CalcMass=(400:10:1700)';
Kmatrix1=cell(length(CalcMass),8);
Kmatrix2=cell(length(CalcMass),8);
Kmatdis1=cell(length(CalcMass),8);
Kmatdis2=cell(length(CalcMass),8);
for i = 1:length(CalcMass);
   SelMass=CalcMass(i);

   tempplus2=find(Mass==(SelMass*2));
   tempplus3=find(Mass==(SelMass*3));
   Ratplus1=Mtot(i,:);
   Ratplus2=Mtot(tempplus2,:);
   Ratplus3=Mtot(tempplus3,:);
```

```
ratio=[Ratplus1; Ratplus1; ...
   Ratplus2; Ratplus2; Ratplus2; ...
   Ratplus3; Ratplus3; Ratplus3;];

Norm1=ratio(1,:)./max(ratio(1,:)); Norm2=ratio(2,:)./max(ratio(2,:));
Norm3=ratio(3,:)./max(ratio(3,:)); Norm4=ratio(4,:)./max(ratio(4,:));
Norm5=ratio(5,:)./max(ratio(5,:)); Norm6=ratio(6,:)./max(ratio(6,:));
Norm7=ratio(7,:)./max(ratio(7,:)); Norm8=ratio(8,:)./max(ratio(8,:));

ratio_new=[Norm1; Norm2; Norm3; Norm4; Norm5; Norm6; Norm7; Norm8;];
tags=[1 2 1 2 3 1 2 3];
charge=[1 1 2 2 2 3 3 3];
offset=zeros(1,8);
yfitend1=zeros(1,8);
for k=1:8
   offset(k)=((4*tags(k))/charge(k));
   yfitend1(k)=(21+(2/12));
end

for k = 1:8
   pars=zeros(1,14);
   pars(2)=mass_delta(k);
   pars(3)=mass_sigma;
   pars(4)=ratio_new(k,1);
   pars(5)=ratio_new(k,2);
   pars(6)=ratio_new(k,3);
   pars(7)=ratio_new(k,4);
   pars(8)=ratio_new(k,5);
   pars(9)=ratio_new(k,6);
   pars(10)=ratio_new(k,7);
   pars(11)=ratio_new(k,8);
   pars(12)=ratio_new(k,9);
   pars(13)=ratio_new(k,10);
   pars(14)=ratio_new(k,11);
   mass_pos=18;
   pars(1)=(1/12)*mass_pos;

   name=['K',num2str(k)];

   yfit=[0:(1/12):yfitend1(k)];
   K1=pars(4)*exp(-(yfit-pars(1)).^2/(2*pars(3)^2))+...
      pars(5)*exp(-(yfit-pars(1)-pars(2)).^2/(2*pars(3)^2))+...
      pars(6)*exp(-(yfit-pars(1)-2*pars(2)).^2/(2*pars(3)^2))+...
      pars(7)*exp(-(yfit-pars(1)-3*pars(2)).^2/(2*pars(3)^2))+...
      pars(8)*exp(-(yfit-pars(1)-4*pars(2)).^2/(2*pars(3)^2))+...
      pars(9)*exp(-(yfit-pars(1)-5*pars(2)).^2/(2*pars(3)^2))+...
      pars(10)*exp(-(yfit-pars(1)-6*pars(2)).^2/(2*pars(3)^2))+...
      pars(11)*exp(-(yfit-pars(1)-7*pars(2)).^2/(2*pars(3)^2))+...
      pars(12)*exp(-(yfit-pars(1)-8*pars(2)).^2/(2*pars(3)^2))+...
      pars(13)*exp(-(yfit-pars(1)-9*pars(2)).^2/(2*pars(3)^2))+...
      pars(14)*exp(-(yfit-pars(1)-10*pars(2)).^2/(2*pars(3)^2));

        yfit2=[0:(1/12):yfitend1(k)];
         pars(1)=((1/12)*mass_pos)+offset(k);

    K2=pars(4)*exp(-(yfit2-pars(1)).^2/(2*pars(3)^2))+...
      pars(5)*exp(-(yfit2-pars(1)-pars(2)).^2/(2*pars(3)^2))+...
      pars(6)*exp(-(yfit2-pars(1)-2*pars(2)).^2/(2*pars(3)^2))+...
      pars(7)*exp(-(yfit2-pars(1)-3*pars(2)).^2/(2*pars(3)^2))+...
      pars(8)*exp(-(yfit2-pars(1)-4*pars(2)).^2/(2*pars(3)^2))+...
      pars(9)*exp(-(yfit2-pars(1)-5*pars(2)).^2/(2*pars(3)^2))+...
      pars(10)*exp(-(yfit-pars(1)-6*pars(2)).^2/(2*pars(3)^2))+...
      pars(11)*exp(-(yfit-pars(1)-7*pars(2)).^2/(2*pars(3)^2))+...
      pars(12)*exp(-(yfit-pars(1)-8*pars(2)).^2/(2*pars(3)^2))+...
      pars(13)*exp(-(yfit-pars(1)-9*pars(2)).^2/(2*pars(3)^2))+...
      pars(14)*exp(-(yfit-pars(1)-10*pars(2)).^2/(2*pars(3)^2));
    Kmatdis1{i,k}=yfit;
    Out{i,k}(1,:)=K1/max(K1);
    Out{i,k}(2,:)=K2/max(K2);
    clear K1 K2
  end
end
study.preliminary.kMatrices=Out;
success=true;
```

APPENDIX 12 PEPTIDE FINDER FUNCTION

```
function peptideFinder(dataFile, models, modelMass)
%%
% Configuration stuff.
start=1;
config.dataFile=dataFile;
config.modelFile='INTERNAL_FROM_WORKFLOW';
config.ratioFile='INTERNAL_FROM_WORKFLOW';
config.outputSuffix='pF';
config.minSignal=100;
config.minSNR=3;
% End configuration

% Begin load-in
inData.models=models;
inData.modelMass=modelMass;
load(config.dataFile);
inData.raw=dat.intensity;
inData.mz=dat.massCharge;
inData.scanTime=dat.rTime;
clear('dat');
[inData.numMzPts,inData.numTimePts]=size(inData.raw);
% End load-in.

% Begin model parameter calculation
model.basePeakIndex=19;
model.isotopePeakWidth=12;
model.numIsotopePeaks=10;
model.lightHeavyDeltaMass=4;
model.numTypes=8;
model.charge=[1 1 2 2 2 3 3 3];
model.numLabels=[1 2 1 2 3 1 2 3];
% ------------------------
model.width=(model.numIsotopePeaks+2)*model.isotopePeakWidth;
% Add 2 peak widths for pre/post falloff.
model.firstPeakStart=ones(1,model.numTypes)*(model.basePeakIndex-7);
%compute model start. Subtract 7 to get to base of M peak.
model.secondPeakStart=model.firstPeakStart+((model.isotopePeakWidth* ...
   model.lightHeavyDeltaMass.*model.numLabels)./model.charge);
%model.peakEndOffset=(1./model.charge)*model.width;
model.peak1NonZeroIndex=cellfun(@(model) find(model(1,:)>0.01),...
   inData.models,'uniformOutput',false);
model.firstPeakEnd=cellfun(@(indices) max(indices)+1,model.peak1NonZeroIndex);
model.peak2NonZeroIndex=cellfun(@(model) find(model(2,:)>0.01),...
   inData.models,'uniformOutput',false);
model.secondPeakEnd=cellfun(@(indices) max(indices)+1,model.peak2NonZeroIndex);
% end model parameter calculation.

% Begin prepare output arrays.

outData.bestModel=sparse(zeros(inData.numMzPts,inData.numTimePts));
outData.modelSNR=sparse(zeros(inData.numMzPts,inData.numTimePts));
outData.modelParam1=sparse(zeros(inData.numMzPts,inData.numTimePts));
outData.modelParam2=sparse(zeros(inData.numMzPts,inData.numTimePts));
outData.modelParam3=sparse(zeros(inData.numMzPts,inData.numTimePts));
% End prepare output arrays.

% Begin prepare temporary arrays
tempData.thisMassModel=[];
tempData.excisedModel=[];
tempData.idealRawData=[];
tempData.modelFit=[];
tempData.SSR=[];
tempData.stdErr=[];
tempData.excisedData=[];
% End prepare temporary arrays
%%
progress=waitbar(0,['Processing ...'],'Name','Processing points...');
for massPtr=start:inData.numMzPts-7;
   if (massPtr==1||mod(massPtr,120)==0)
      tempData.thisModelIndex=ceil(massPtr/120);
      tempData.thisMassModel=inData.models(tempData.thisModelIndex,:);
      tempData.firstPeakEnd=model.firstPeakEnd(tempData.thisModelIndex,:);
      tempData.secondPeakEnd=model.secondPeakEnd(tempData.thisModelIndex,:);
      tempData.totalLength=tempData.secondPeakEnd-model.firstPeakStart;
      tempData.excisedModel=arrayfun(@(model,mStart,mEnd) model{1}(:, ...
         mStart:mEnd),tempData.thisMassModel,model.firstPeakStart,...
         tempData.secondPeakEnd,'UniformOutput',false);
      tempData.excisedModel=cellfun(@(model) model',tempData.excisedModel,...
         'uniformOutput',false); % transpose for function.
      tempData.excisedModel=arrayfun(@trimInterpeakSpace, ...
         model.firstPeakStart, ...
         tempData.firstPeakEnd, ...
         model.secondPeakStart, ...
         tempData.secondPeakEnd, ...
         tempData.excisedModel, 'uniformOutput',false);
      tempData.excisedModel=cellfun(@(model) model',tempData.excisedModel,...
        'uniformOutput',false); % transpose for function.
    tempData.excisedModel=cellfun(@(model) [model; ...
        ones(1,numel(model(1,:)))], tempData.excisedModel,...
        'uniformOutput',false);
    tempData.inverseModel=cellfun(@(model) model'/(model*model'),...
        tempData.excisedModel,'uniformOutput',false);
end
tempData.excisedData=arrayfun(@(mSize) inData.raw(massPtr:min(...
    massPtr+mSize,inData.numMzPts),:),tempData.totalLength,...
    'UniformOutput',false);
tempData.excisedData=arrayfun(@trimInterpeakSpace, ...
    model.firstPeakStart, ...
    tempData.firstPeakEnd, ...
    model.secondPeakStart, ...
    tempData.secondPeakEnd, ...
    tempData.excisedData, 'uniformOutput',false);
tempData.modelFit=cellfun(@(rawData,inverseModel)(rawData'*inverseModel)',...
    tempData.excisedData,tempData.inverseModel,'uniformOutput',false);
tempData.idealRawData=cellfun(@(model,fits) fits'*model, ...
    tempData.excisedModel, tempData.modelFit,'uniformOutput',false);
tempData.SSR=cellfun(@(rawData,idealRawData) ...
    sum((rawData-idealRawData').^2), tempData.excisedData, ...
    tempData.idealRawData,'uniformOutput',false);
tempData.stdErr=cellfun(@(SSR,model) sqrt(SSR/(numel(model(1,:))-3)),...
    tempData.SSR, tempData.excisedModel,'uniformOutput',false);
tempData.SNR=cellfun(@(stdErrs,fits) min([fits(1,:)./stdErrs; fits(2,:)...
    ./stdErrs]),tempData.stdErr,tempData.modelFit,'uniformOutput',false);
tempData.validLocations=cellfun(@(SNRs,fits) ((fits(1,:)>config.minSignal)...
.*(fits(2,:)>config.minSignal).*(SNRs>config.minSNR)),tempData.SNR,...
tempData.modelFit,'uniformOutput',false);
tempData.filteredFit=cellfun(@(fits,valid) fits.*repmat(valid,3,1),...
    tempData.modelFit,tempData.validLocations,'uniformOutput',false);
tempData.filteredSNR=cellfun(@(SNRs,valid) SNRs.*valid, tempData.SNR,...
    tempData.validLocations,'uniformOutput',false);
tempData.filteredSNR=cell2mat(tempData.filteredSNR');
tempData.param1=cell2mat(cellfun(@(fit) fit(1,:),tempData.filteredFit,...
    'uniformOutput',false)');
tempData.param2=cell2mat(cellfun(@(fit) fit(2,:),tempData.filteredFit,...
    'uniformOutput',false)');
tempData.param3=cell2mat(cellfun(@(fit) fit(3,:),tempData.filteredFit,...
    'uniformOutput',false)');
[tempData.classMaxVal,tempData.classType]=max(tempData.filteredSNR);
tempData.selectedIndex=sub2ind(size(tempData.param1),tempData.classType,...
    1:numel(tempData.classType));
    outData.modelParam1(massPtr+7,:)=tempData.param1(tempData.selectedIndex);
    outData.modelParam2(massPtr+7,:)=tempData.param2(tempData.selectedIndex);
    outData.modelParam3(massPtr+7,:)=tempData.param3(tempData.selectedIndex);
    tempData.classType(tempData.classMaxVal==0)=0;
    outData.bestModel(massPtr+7,:)=tempData.classType;
    outData.modelSNR(massPtr+7,:)=tempData.classMaxVal;
    waitbar(massPtr/inData.numMzPts,progress,['Processing m/z ' ...
        num2str(massPtr) ' of ' num2str(inData.numMzPts) '.' ]);
    if (any(outData.bestModel(massPtr+7,:)==1 & ...
            outData.modelSNR(massPtr+7,:)>15) )
        keyboard;
    end
end
close(progress);
save([config.dataFile '-' config.outputSuffix],'outData','inData','model','config');

% Begin helper functions

function outData=trimInterpeakSpace(peak1Start,peak1End,peak2Start,peak2End,data)
        % Pad end of data with zeros if it is smaller than the range we
        % want to excise. Fit over this area will be zero, and discarded by
        % the check that the fit parameter is less than config.minSignal.
        if numel(data{1}(:,1)<peak2End-peak1Start)
            data{1}(end:(peak2End-peak1Start)+1,:)=0;
        end
        if peak1End<peak2Start
            outData=[data{1}(1:peak1End-peak1Start,:); ...
                data{1}(peak2Start-peak1Start:peak2End-peak1Start,:)];
        else
            outData=data{1}(1:peak2End-peak1Start+1,:);
        end
```

APPENDIX 13 CLUSTERING FUNCTION

```
function buildCluster2(dataFile)

% Begin load-in

load(dataFile,'-mat');

% End load-in.

% Begin Configuration

config.weightCutoff=6;
config.typeCutoff=5;
config.timeUnitWeight=1;
config.mzUnitWeight=5;
config.minTimeSize=5;
config.minSNRofMinSize=3;
config.samePeptideTimeDistance=round(0.1*numel(inData.raw(1,:)));
config.minOverallSNR=8; % 8 for _REAL_ data, 3 for testing.

% End configuration.

% Begin prep for locating clusters.

[tempData.clusterMzs,tempData.clusterTimes]=find(outData.bestModel>0);
tempData.clusterIDs=sparse(zeros(size(outData.bestModel)));
tempData.clusterWts=ones(size(outData.bestModel))*(config.weightCutoff+1);
tempData.visitedPts=zeros(size(outData.bestModel));
tempData.toLookAt=java.util.Stack();
tempData.diagWeight=sqrt(config.timeUnitWeight^2+config.mzUnitWeight^2);
tempData.lastClusterID=0;

% End variable setup for locating.

%Begin determining extent of clusters
progress=waitbar(0,['Processing ...'],'Name','Processing points...');
for clusterPt=1:numel(tempData.clusterMzs)
    startMz=tempData.clusterMzs(clusterPt);
    startTime=tempData.clusterTimes(clusterPt);
    if tempData.visitedPts(startMz,startTime)==0 &&
outData.bestModel(startMz,startTime)~=0

tempData.toLookAt.push([startMz,startTime,1,outData.bestModel(startMz,startTime)]);
```

```
    tempData.inCluster=false;
    while (~tempData.toLookAt.empty())
      thisPoint=tempData.toLookAt.pop();
      thisMz=thisPoint(1); thisTime=thisPoint(2); thisWeight=thisPoint(3);
lastType=thisPoint(4);
      tempData.stackEmpty=false;
      while thisMz<1 || thisMz>inData.numMzPts || thisTime<1 ||
thisTime>inData.numTimePts
        if tempData.toLookAt.empty()
          tempData.stackEmpty=true;
          break;
        end
        thisPoint=tempData.toLookAt.pop();
        thisMz=thisPoint(1); thisTime=thisPoint(2); thisWeight=thisPoint(3);
lastType=thisPoint(4);
      end
      if tempData.stackEmpty
        break;
      end
      thisType=outData.bestModel(thisMz,thisTime);
      tempData.deltaWeight=thisWeight;
      if ~tempData.inCluster
        tempData.lastClusterID=tempData.lastClusterID+1;
      end
      if thisType~=0 && outData.modelSNR(thisMz,thisTime) >
config.minSNRofMinSize
        % we have a valid point. Determine what cluster it matches.
        if thisType~=lastType && tempData.inCluster && lastType~=0
          if numel(config.typeCutoff)==1
            tempData.deltaWeight=config.typeCutoff;
          else

tempData.deltaWeight=config.typeCutoff(min(thisType,lastType),max(thisType,lastType)
);
          end
        else
          tempData.deltaWeight=1;
        end
      end
      if ~tempData.inCluster
        tempData.inCluster=true;
      end
      if tempData.deltaWeight<config.weightCutoff &&
tempData.deltaWeight<tempData.clusterWts(thisMz,thisTime)
```

```
                tempData.clusterIDs(thisMz,thisTime)=tempData.lastClusterID;
                tempData.clusterWts(thisMz,thisTime)=tempData.deltaWeight;
                % This is a valid cluster point. Explore the neighbours.
                tempData.toLookAt.push([thisMz-
1,thisTime,tempData.deltaWeight+config.mzUnitWeight,thisType]);

tempData.toLookAt.push([thisMz+1,thisTime,tempData.deltaWeight+config.mzUnitWei
ght,thisType]);
                tempData.toLookAt.push([thisMz,thisTime-
1,tempData.deltaWeight+config.timeUnitWeight,thisType]);

tempData.toLookAt.push([thisMz,thisTime+1,tempData.deltaWeight+config.timeUnitWe
ight,thisType]);
                tempData.toLookAt.push([thisMz-1,thisTime-
1,tempData.deltaWeight+tempData.diagWeight,thisType]);
                tempData.toLookAt.push([thisMz-
1,thisTime+1,tempData.deltaWeight+tempData.diagWeight,thisType]);
                tempData.toLookAt.push([thisMz+1,thisTime-
1,tempData.deltaWeight+tempData.diagWeight,thisType]);

tempData.toLookAt.push([thisMz+1,thisTime+1,tempData.deltaWeight+tempData.diagW
eight,thisType]);
            end
            tempData.visited(thisMz,thisTime)=1;
        end
    end
    if (mod(clusterPt,50)==0)
        waitbar(clusterPt/numel(tempData.clusterMzs),progress,['Phase 1 of 3 - Processing '
num2str(clusterPt) ' of ' num2str(numel(tempData.clusterMzs)) ' points.' ]);
%         disp(['Processed ' num2str(clusterPt) ' of '
num2str(numel(tempData.clusterMzs))]);
    end
end

% End locating clusters.

% Begin Postprocessing each cluster
outData.clusterTypes=sparse(zeros(size(outData.bestModel)));
tempData.clusterWts(tempData.clusterWts==config.weightCutoff+1)=0;
tempData.clusterWts=sparse(tempData.clusterWts);
tempData.clusterInfo=zeros(tempData.lastClusterID,7);
tempData.modelOffset=((model.isotopePeakWidth/2)+1);
tempData.knownLocations=tempData.clusterWts==1;
% Filter ID "clouds" down to regions where peptideFinder identified
% something.
tempData.clusterIDs(~tempData.knownLocations)=0;
outData.peakStats=zeros(tempData.lastClusterID,9);
for thisCluster=1:tempData.lastClusterID
%     if thisCluster==25225
%         a=1;
%     end
    if mod(thisCluster, 25)==0
        waitbar(thisCluster/tempData.lastClusterID,progress,['Phase 2 of 3 - Processing '
num2str(thisCluster) ' of ' num2str(tempData.lastClusterID) ' clusters' ]);
%         disp(['Processed ' num2str(thisCluster) ' of ' num2str(tempData.lastClusterID)]);
    end
    tempData.boolLocation=tempData.clusterIDs==thisCluster;
    [tempData.clusterMzs,tempData.clusterTimes]=find(tempData.boolLocation);
    if isempty(tempData.clusterMzs)
        continue;
    end
    if max(max(tempData.clusterTimes))-
min(min(tempData.clusterTimes))<config.minTimeSize || ...
            numel(tempData.clusterMzs)<config.minTimeSize
        continue; % Check that we span the required time length and have the required
minimum number of points.
    end
    % Get the range over which our cluster exists:
    tempData.topEdge=min(tempData.clusterMzs);
    tempData.bottomEdge=max(tempData.clusterMzs);
    tempData.leftEdge=min(tempData.clusterTimes);
    tempData.rightEdge=max(tempData.clusterTimes);
    % Grab the correct models for our fit
    tempData.thisModelIndex=ceil(tempData.topEdge/120);
    tempData.thisMassModel=inData.models(tempData.thisModelIndex,:);
    tempData.firstPeakEnd=model.firstPeakEnd(tempData.thisModelIndex,:);
    tempData.secondPeakEnd=model.secondPeakEnd(tempData.thisModelIndex,:);
    tempData.totalLength=tempData.secondPeakEnd-model.firstPeakStart+1;
    tempData.excisedModel=arrayfun(@(model,mStart,mEnd) model{1}
(:,mStart:mEnd),tempData.thisMassModel,model.firstPeakStart,tempData.secondPeakEn
d,'UniformOutput',false);
    tempData.excisedModel=cellfun(@(model)
model',tempData.excisedModel,'uniformOutput',false); % transpose for function.
    tempData.excisedModel=arrayfun(@trimInterpeakSpaceNoPadding, ...
        model.firstPeakStart, ...
        tempData.firstPeakEnd, ...
        model.secondPeakStart, ...
        tempData.secondPeakEnd, ...
        tempData.excisedModel, 'uniformOutput',false);
    tempData.excisedModel=cellfun(@(model)
model',tempData.excisedModel,'uniformOutput',false); % transpose for function.
    tempData.excisedModel=cellfun(@(model) [model; ones(1,numel(model(1,:)))],
tempData.excisedModel , 'uniformOutput',false);
    tempData.inverseModel=cellfun(@(model) model'/(model*model'),
tempData.excisedModel,'uniformOutput',false);
    %Now excise the raw data so we can determine type/fit.
    tempData.excisedData=arrayfun(@(mSize) inData.raw(tempData.topEdge-
tempData.modelOffset:min(tempData.bottomEdge-
tempData.modelOffset+mSize+1,inData.numMzPts),tempData.leftEdge:tempData.rightE
dge),tempData.totalLength,'UniformOutput',false);
    tempData.fitData=cell(1,(tempData.bottomEdge-tempData.topEdge)+1);
%     if max(max(tempData.excisedData{1}))>10000
%         keyboard;
%     end
    for thisOffset=1:(tempData.bottomEdge-tempData.topEdge)+1
        tempData.subWindow=arrayfun(@(data,length)
cutSubWindow(data,length,thisOffset),tempData.excisedData,tempData.totalLength,'unif
ormOutput',false);
        tempData.subWindow=arrayfun(@trimInterpeakSpaceNoPadding, ...
            model.firstPeakStart, ...
            tempData.firstPeakEnd, ...
            model.secondPeakStart, ...
            tempData.secondPeakEnd, ...
            tempData.subWindow, 'uniformOutput',false);

tempData.fitData{thisOffset}=cellfun(@fitModeltoSVD,tempData.inverseModel,tempDa
ta.excisedModel,tempData.subWindow,'uniformOutput',false);
    end
    tempData.SNRs=cellfun(@(fits)
cell2mat(fits),tempData.fitData,'uniformOutput',false);
    tempData.SNRs=cellfun(@(fits) fits(4,:),tempData.SNRs,'uniformOutput',false);
    tempData.SNRs=cell2mat(tempData.SNRs');
     if (max(max(tempData.SNRs))>config.minOverallSNR)
        [bestRow,bestClass]=find(tempData.SNRs==max(max(tempData.SNRs)));
        outData.clusterTypes(tempData.topEdge+(bestRow-
1),tempData.leftEdge:tempData.rightEdge)=bestClass;
        outData.peakStats(thisCluster,1)=tempData.topEdge+(bestRow-1);
        outData.peakStats(thisCluster,2)=tempData.secondPeakEnd(bestClass)-12;
        outData.peakStats(thisCluster,3)=tempData.leftEdge;
        outData.peakStats(thisCluster,4)=tempData.rightEdge;
        outData.peakStats(thisCluster,5)=bestClass;
        outData.peakStats(thisCluster,6:9)=tempData.fitData{bestRow}{bestClass};
    end
end
emptyData=sum(outData.peakStats,2)==0;
outData.peakStats(emptyData,:)=[];
clear tempData; % clean up, prepare for overlap check.
waitbar(0.9,progress,['Phase 3 of 3 - Recomputing Statistics...' ]);
tempData.numClusters=numel(outData.peakStats(:,1));
tempData.boundingBoxes(:,1)=outData.peakStats(:,1)-7;
tempData.boundingBoxes(:,2)=outData.peakStats(:,1)+outData.peakStats(:,2)-7;
tempData.boundingBoxes(:,3)=outData.peakStats(:,3);
tempData.boundingBoxes(:,4)=outData.peakStats(:,4);

tempData.bbStats=outData.peakStats;
tempData.bbStats(:,1:4)=tempData.boundingBoxes;
tempData.bbCell=num2cell(tempData.bbStats,2);

% Prior to overlap, check whether some of the detections are possibly the
% same peptide but it has dropped out for some reason in between
% (interference or too low quality)
tempData.relatedDetections=cellfun(@(thisSignal)find( ...
   tempData.bbStats(:,1)>=thisSignal(1)-1 & ...
   tempData.bbStats(:,1)<=thisSignal(1)+1 & ...
   tempData.bbStats(:,5)==thisSignal(5) & ...
   thisSignal(3)<=tempData.bbStats(:,4)+config.samePeptideTimeDistance & ...
   thisSignal(4)>=tempData.bbStats(:,3)-config.samePeptideTimeDistance ...
   ),tempData.bbCell,'uniformoutput',false);
for i=1:numel(tempData.relatedDetections)
   if min(tempData.relatedDetections{i})~=i
      tempData.relatedDetections{i}=-1;
   end
end

%now that we know which detections are related, rebuild the data table to
%account for this.
tempData.mergedDetections=zeros(i,9);
tempData.quantifyMask=cell(i,1);
tempData.detectionMask=false(size(inData.raw));
for i=1:numel(tempData.relatedDetections)
   if tempData.relatedDetections{i}~=-1
      detectionTable=tempData.bbStats(tempData.relatedDetections{i},:);
      tempData.mergedDetections(i,[1,2,5])=tempData.bbStats(i,[1,2,5]);
      tempData.mergedDetections(i,3)=min(detectionTable(:,3));
      tempData.mergedDetections(i,4)=max(detectionTable(:,4));
      tempData.quantifyMask{i}=false(1,1+tempData.mergedDetections(i,4)-
tempData.mergedDetections(i,3));
        for j=1:numel(detectionTable(:,1))
            tempData.quantifyMask{i}(1+detectionTable(j,3)-
tempData.mergedDetections(i,3):1+detectionTable(j,4)-
tempData.mergedDetections(i,3))=true;
        end

tempData.detectionMask(tempData.mergedDetections(i,1),tempData.mergedDetections(i,
3):tempData.mergedDetections(i,4))=tempData.quantifyMask{i};
    end
end
keepDetections=~cellfun('isempty',tempData.quantifyMask);
tempData.quantifyMask=tempData.quantifyMask(keepDetections,1);
tempData.mergedDetections=tempData.mergedDetections(keepDetections,:);
tempData.finalDetectionMask=false(size(inData.raw));
%Finally, handle overlapping signals and mark as appropriate.
for i=1:numel(tempData.mergedDetections(:,1))
    overlapTest=tempData.detectionMask(...
        tempData.mergedDetections(i,1):tempData.mergedDetections(i,2),...
        tempData.mergedDetections(i,3):tempData.mergedDetections(i,4));
    overlapColumns=sum(overlapTest)>1;
    overlapTest(:,overlapColumns)=false;
    tempData.finalDetectionMask(...
        tempData.mergedDetections(i,1):tempData.mergedDetections(i,2),...
        tempData.mergedDetections(i,3):tempData.mergedDetections(i,4))=overlapTest;
end
tempData.quantifyMask=cellfun(@(detection)tempData.finalDetectionMask(detection(1)
,detection(3):detection(4)),num2cell(tempData.mergedDetections,2),'uniformoutput',false
);


tempData.recalcStats=cellfun(@(dim,mask)requantify(dim,mask),num2cell(tempData.me
rgedDetections,2),tempData.quantifyMask,'uniformoutput',false);
%tempData.quantRanges=cellfun(@(stats) stats{2},
tempData.recalcStats,'uniformOutput',false);
%tempData.recalcStats=cellfun(@(stats) stats{1},
tempData.recalcStats,'uniformOutput',false);
tempData.recalcStats=cell2mat(tempData.recalcStats);
tempData.goodSNR=tempData.recalcStats(:,9)>=config.minOverallSNR |
tempData.recalcStats(:,9)==0;

tempData.recalcStats=tempData.recalcStats(tempData.goodSNR,:);
%outData.quantRanges=tempData.quantRanges(tempData.goodSNR,:);
```

```
outData.finalStats=tempData.recalcStats;
%restore original format.
outData.quantifyMask=tempData.quantifyMask;
outData.finalStats(:,2)=outData.finalStats(:,2)-outData.finalStats(:,1);
outData.finalStats(:,1)=outData.finalStats(:,1)+7;
outData.finalStatsColumns={'First Peak m/z';'m/z duration';'Start time';'End
time';'Class';'Intensity 1';'Intensity 2';'Baseline Intensity';'Signal To Error'}
save(dataFile,'outData','config','-append');
close(progress);



function outData=fitModeltoSVD(invModel,model,data)
averagePeak=mean(data');
% [p1,p2,p3]=svd(shiftedData,'econ');
% component=p1*p2;
% component=component(:,1)';
fit=averagePeak*invModel;
idealData=fit*model;
SSR=sum((idealData-averagePeak).^2);
stdErr=sqrt(SSR/(numel(invModel(:,1))-3));
SNR=min(fit(1)/stdErr,fit(2)/stdErr);
if (SNR>10)
 %   keyboard;
end
outData=double([fit'; SNR]);


function outData=trimInterpeakSpace(peak1Start,peak1End,peak2Start,peak2End,data)
% Check that the data is oriented correctly for the function. If
% not, rotate it, and rotate back when done.
% Pad end of data with zeros if it is smaller than the range we
% want to excise. Fit over this area will be zero, and discarded by
% the check that the fit parameter is less than config.minSignal.
if numel(data{1}(:,1)<peak2End-peak1Start)
    data{1}(end:(peak2End-peak1Start)+1,:)=0;
end
if peak1End<peak2Start
    outData=[data{1}(1:peak1End-peak1Start,:); data{1}(peak2Start-
peak1Start:peak2End-peak1Start,:)];
else
    outData=data{1}(1:peak2End-peak1Start+1,:);
end
```

```
function
outData=trimInterpeakSpaceNoPadding(peak1Start,peak1End,peak2Start,peak2End,data)
% Check that the data is oriented correctly for the function. If
% not, rotate it, and rotate back when done.
% Pad end of data with zeros if it is smaller than the range we
% want to excise. Fit over this area will be zero, and discarded by
% the check that the fit parameter is less than config.minSignal.
if peak1End<peak2Start
   outData=[data{1}(1:peak1End-peak1Start,:); data{1}(peak2Start-
peak1Start:peak2End-peak1Start,:)];
else
   if (peak2End-peak1Start+1)>numel(data{1}(:,1))
      keyboard;
   end
   outData=data{1}(1:peak2End-peak1Start+1,:);
end

function outData=requantify(thisPeak,detectionMask)
% This function detects the overlap between two peaks and adjusts the
% window sizes appropriately, before re-computing the fit using the
% corrected window.
config=evalin('caller','config');
fullStats=thisPeak;
canRequantify=true;

if sum(detectionMask)<config.minTimeSize
   canRequantify=false; % cant quantify, too small after removing overlap.
   fullStats(5)=fullStats(5)*-1; % Mark as unquantifiable.
else
   newRange=thisPeak(3):thisPeak(4);
   newRange=newRange(detectionMask);
end

if (canRequantify)
   rawData=evalin('caller','inData.raw');
   rawData=rawData(thisPeak(1):thisPeak(2),newRange);
   reqModel=ceil(thisPeak(1)/120);
   thisMassModel=evalin('caller',['inData.models(' num2str(reqModel) ',:)']);
   firstPeakStart=evalin('caller','model.firstPeakStart');
   firstPeakEnd=evalin('caller',['model.firstPeakEnd(' num2str(reqModel) ',:)']);
   secondPeakStart=evalin('caller','model.secondPeakStart');
   secondPeakEnd=evalin('caller',['model.secondPeakEnd(' num2str(reqModel) ',:)']);
   trimmedModel=arrayfun(@(model,mStart,mEnd) model{1}
(:,mStart:mEnd),thisMassModel,firstPeakStart,secondPeakEnd,'UniformOutput',false);
   trimmedModel=cellfun(@(model) model',trimmedModel,'uniformOutput',false);
   trimmedModel=arrayfun(@trimInterpeakSpaceNoPadding, ...
      firstPeakStart, ...
      firstPeakEnd, ...
      secondPeakStart, ...
      secondPeakEnd, ...
      trimmedModel, 'uniformOutput',false);
    trimmedData=trimInterpeakSpace( ...
      firstPeakStart(fullStats(5)), ...
      firstPeakEnd(fullStats(5)), ...
      secondPeakStart(fullStats(5)), ...
      secondPeakEnd(fullStats(5)), ...
     {rawData});
   trimmedModel=trimmedModel{fullStats(5)}; % Get data only for correct class.
   trimmedModel(:,3)=ones(numel(trimmedModel(:,1)),1);
   trimmedModel=trimmedModel';
   invTrimmedModel=trimmedModel'/(trimmedModel*trimmedModel');
   stats=fitModeltoSVD(invTrimmedModel,trimmedModel,trimmedData);
   fullStats(6:9)=stats';
end
outData=fullStats;

function out=cutSubWindow(data,length,thisOffset)
try
   out=data{1}(thisOffset:thisOffset+length,:);
catch errorWeDontCareAboutBecauseWeAreLazy
   out=zeros(length,numel(data{1}(1,:)));
end
```

APPENDIX 14 SEQUEST XLS IMPORT

```
%fileName is the xls file to read that has been exported from Bioworks.
% It should contain only ONE protein for now.
% Masses reported are for the LIGHT location of a particular sequence,
% regardless if whether it is actually light or heavy.
%Output needs to be fed through CombineScans to prepare for use with
%SequestCompare.
function dataOut=xlsExtract(filePath,lightFileName,heavyFileName,xlsSource)

% desiredColumns=[1:2 3 5 4 6]; % was 2:4,6 for old sequest
% [~,~,fileContent]=xlsread([filePath lightFileName]);
% data=fileContent(1:end,desiredColumns);
% Above is no longer needed with new load<source> methods.

switch xlsSource
    case 'BW'
        fileLoadFunction=@loadBioworks;
    case 'PD'
        fileLoadFunction=@loadProteomeDiscoverer;
end


data=fileLoadFunction([filePath lightFileName]);
%
data(:,5)=cellfun(@(seq,wt)sequenceWeight(upper(seq),wt,'L'),data(:,2),data(:,3),'unifor
moutput',false);
% % Compute # labels by mass difference
data(:,end+1)=cellfun(@(seq)sum(seq=='k'|seq=='K')+1,data(:,2),'uniformoutput',false);
% Compute # labels based on K
dataOut.sequence=data(:,2);
dataOut.scanType=repmat('L',numel(dataOut.sequence),1);
dataOut.MHMass=cell2mat(data(:,3));
dataOut.z=cell2mat(data(:,4));
dataOut.numLabels=cell2mat(data(:,end));
dataOut.organism=data(:,5);
dataOut.protein=data(:,6);
[dataOut.scanStart,dataOut.scanEnd]=cellfun(@splitScans,data(:,1),'uniformoutput',false)
;
dataOut.scanStart=cell2mat(dataOut.scanStart);
dataOut.scanEnd=cell2mat(dataOut.scanEnd);

data=fileLoadFunction([filePath heavyFileName]);
%
```

```
data(:,5)=cellfun(@(seq,wt)sequenceWeight(upper(seq),wt,'H'),data(:,2),data(:,3),'unifor
moutput',false);
data(:,end+1)=cellfun(@(seq)sum(seq=='k'|seq=='K')+1,data(:,2),'uniformoutput',false);
% Compute # labels based on K
dataOut.sequence=[dataOut.sequence;data(:,2)];
dataOut.scanType=[dataOut.scanType;repmat('H',numel(data(:,2)),1)];
dataOut.MHMass=[dataOut.MHMass; (cell2mat(data(:,3))-
(4.025108*cell2mat(data(:,end))))]; %Convert heavy weights to light.
dataOut.z=[dataOut.z;cell2mat(data(:,4))];
dataOut.numLabels=[dataOut.numLabels;cell2mat(data(:,end))];
dataOut.organism=[dataOut.organism;data(:,5)];
dataOut.protein=[dataOut.protein;data(:,6)];
[tScanStart,tScanEnd]=cellfun(@splitScans,data(:,1),'uniformoutput',false);
dataOut.scanStart=[dataOut.scanStart;cell2mat(tScanStart)];
dataOut.scanEnd=[dataOut.scanEnd;cell2mat(tScanEnd)];
dataOut.lightFile=lightFileName;
dataOut.heavyFile=heavyFileName;
dataOut.sourcePath=filePath;

function [left,right]=splitScans(scanRange)
splitLoc=strfind(scanRange,'-');
if isempty(splitLoc)
   if isstr(scanRange)
      left=str2num(scanRange);
   else
      left=scanRange;
   end
   right=left;
else
   left=str2num(scanRange(1:splitLoc-1));
   right=str2num(scanRange(splitLoc+1:end));
end

function data=loadBioworks(file)
[~,~,raw]=xlsread(file);
data={};
temp='BSA';
temp2=raw{3,2};
% temp4=',';

%REFORMAT
rawTrunc=raw(4:end,:);            %remove top 3 lines

organism=cellstr(repmat(temp,length(rawTrunc),1));
```

```
protein=cellstr(repmat(temp2,length(rawTrunc),1));
rawTrunc(:,3)=cellfun(@(str)str(3:end-2),rawTrunc(:,3),'uniformOutput',false);


% for i=1:length(rawTrunc)    %remove ,file name from scan column Not
% needed?
%     scan=rawTrunc{i,2};
%     for j=1:length(rawTrunc{i,2})
%         if(scan(j)==temp4)
%             temp5=j;
%             rawTrunc{i,2}=scan(1:temp5-1);
%         end
%     end
% end
data(:,1)=rawTrunc(:,2); %scans
data(:,2)=rawTrunc(:,3); %sequences
data(:,3)=rawTrunc(:,4); %MH+
data(:,5)=organism;      %organism
data(:,4)=rawTrunc(:,6); %charge
data(:,6)=protein;       %protein

function data=loadProteomeDiscoverer(file)
[num,txt,raw]=xlsread(file);

%INITIALIZATION
pepcountstore=zeros(length(raw),1);
prostore=cell(length(raw),1);

BSAcount=0;
lysocount=0;
ecolicount=0;
keratincount=0;

p1='bovine';
p2='Lyso';
p3='coli';
p4='keratin';

type1='BSA';
type2='Lysozyme';
type3='E.Coli';
type4='Keratin';

start1=1;
```

```
fin1=length(p1);
start2=1;
fin2=length(p2);
start3=1;
fin3=length(p3);
start4=1;
fin4=length(p4);

pepstore=cell(length(raw),1);
chargestore=zeros(length(raw),1);
mzstore=chargestore;
fstscanstore=chargestore;
lstscanstore=chargestore;
mhplusstore=chargestore;

prolist=cell(length(find(pepcountstore>0)),1);

type=zeros(length(prolist),1);

peplist=cell((sum(pepcountstore)),1);

pepcountlist=zeros(sum(pepcountstore>0),1);
chargelist=zeros(sum(chargestore>0),1);
mzlist=zeros(sum(mzstore>0),1);
fstscanlist=zeros(sum(fstscanstore>0),1);
lstscanlist=zeros(sum(lstscanstore>0),1);
mhpluslist=zeros(sum(mhplusstore>0),1);

ones=zeros(length(type),1);
twos=zeros(length(type),1);
threes=zeros(length(type),1);
fours=zeros(length(type),1);

startscan=[];
endscan=[];
charge=[];
mz=[];
peptide=[];
proteintype=[];
protein=[];
mhplus=[];

var=NaN;
```

```
%COLLECTING INFORMATION
for i=2:length(raw)
    if isa(raw{i,1},'char')
        pepcountstore(i,:)=raw{i,8};
        prostore{i,:}=raw{i,2};
    end
end

 for i=2:length(pepcountstore)
    if pepcountstore(i)>0
        k=i+2;
    if k>length(pepcountstore)
        break
    end
        for j=1:pepcountstore(i)
            pepstore{k,:}=raw{k,3};
            chargestore(k,:)=raw{k,20};
            mzstore(k,:)=raw{k,21};
            mhplusstore(k,:)=raw{k,22};
            fstscanstore(k,:)=raw{k,25};
            lstscanstore(k,:)=raw{k,26};
            if length(pepcountstore)==1
                break
            end

            k=k+3;
            if k>length(pepcountstore)
                break
            end
            if isequaln(var,raw{k,20});
                k=k+1;
                if k>length(pepcountstore)
                    break
                end
            end
        end
    end
 end

  j=1;%condense protein list
 for i=1:length(prostore)
     if isa(prostore{i,:},'char')
         prolist{j,:}=prostore{i};
         j=j+1;
     end
 end

 j=1;%condense peptide list
for i=1:length(raw)
    if isa(pepstore{i,:},'char')
        peplist{j,:}=pepstore{i};
        j=j+1;
    end
end

j=1;%condense peptide count list
for i=1:length(raw)
    if pepcountstore(i)>0
        pepcountlist(j,:)=pepcountstore(i);
        j=j+1;
    end
end

j=1;%condense charge list
for i=1:length(raw)
    if chargestore(i)>0
        chargelist(j,:)=chargestore(i);
        mzlist(j,:)=mzstore(i);
        fstscanlist(j,:)=fstscanstore(i);
        lstscanlist(j,:)=lstscanstore(i);
        mhpluslist(j,:)=mhplusstore(i);
        j=j+1;
    end
end

pepcountlist2=zeros(length(pepcountlist),1);
for i=2:(length(pepcountlist))                          %cumulative peptide count list
    pepcountlist2(1,:)=pepcountlist(1);
    pepcountlist2(i:end,:)=pepcountlist(i:end,:)+sum(pepcountlist(1:i-1,:));
end

for i=1:length(prolist)                %get separate protein and type counts
    pro=prolist{i,1};

    start1=1;
    fin1=length(p1);

    while fin1<=length(pro)
        portion=pro(1,start1:fin1);
        evaluate=portion==p1;
        add=sum(evaluate);

        if add==length(p1)
            BSAcount=BSAcount+1;
            type(i,:)=1;
            start1=start1+1;
            fin1=fin1+1;
        else
            start1=start1+1;
            fin1=fin1+1;
        end
    end

    start2=1;
    fin2=length(p2);

    while fin2<=length(pro)
        portion=pro(1,start2:fin2);
        evaluate=portion==p2;
        add=sum(evaluate);
        if add==length(p2)
            lysocount=lysocount+1;
            type(i,:)=2;
            start2=start2+1;
            fin2=fin2+1;
        else
            start2=start2+1;
            fin2=fin2+1;
        end
    end

    start3=1;
    fin3=length(p3);

    while fin3<=length(pro)
        portion=pro(1,start3:fin3);
        evaluate=portion==p3;
        add=sum(evaluate);
        if add==length(p3)
            ecolicount=ecolicount+1;
            type(i,:)=3;
            start3=start3+1;
```

```
            fin3=fin3+1;
        else
            start3=start3+1;
            fin3=fin3+1;
        end
    end

    start4=1;
    fin4=length(p4);

    while fin4<=length(pro)
        portion=pro(1,start4:fin4);
        evaluate=portion==p4;
        add=sum(evaluate);
        if add==length(p4)
            keratincount=keratincount+1;
            type(i,:)=4;
            start4=start4+1;
            fin4=fin4+1;
        else
            start4=start4+1;
            fin4=fin4+1;
        end
    end

end

for i=1:length(type)
    if (type(i)==1)
        ones(i,:)=(i);
    elseif (type(i)==2)
        twos(i,:)=(i);
    elseif (type(i)==3)
        threes(i,:)=(i);
    else
        fours(i,:)=(i);
    end
end

finalStatL=cell(length(prolist),8);

for k=1:length(prolist)
     for i=1:length(type)
         if type(i)==1
```

```
   finalStatL{i,1}=type1;
   finalStatL{i,2}=prolist{i,:};
      if ones(i)==1
         ran=1:pepcountlist2(i);
         finalStatL{i,3}=(peplist(ran,:));
         finalStatL{i,4}=(chargelist(ran,:));
         finalStatL{i,5}=(mzlist(ran,:));
         finalStatL{i,6}=(mhpluslist(ran,:));
         finalStatL{i,7}=(fstscanlist(ran,:));
         finalStatL{i,8}=(lstscanlist(ran,:));
      else
         ran=(pepcountlist2(i-1)+1):pepcountlist2(i);
         finalStatL{i,3}=(peplist(ran,:));
         finalStatL{i,4}=(chargelist(ran,:));
         finalStatL{i,5}=(mzlist(ran,:));
         finalStatL{i,6}=(mhpluslist(ran,:));
         finalStatL{i,7}=(fstscanlist(ran,:));
         finalStatL{i,8}=(lstscanlist(ran,:));
      end
elseif type(i)==2
   finalStatL{i,1}=type2;
   finalStatL{i,2}=prolist{i,:};
      if twos(i)==1
         ran=1:pepcountlist2(i);
         finalStatL{i,3}=(peplist(ran,:));
         finalStatL{i,4}=(chargelist(ran,:));
         finalStatL{i,5}=(mzlist(ran,:));
         finalStatL{i,6}=(mhpluslist(ran,:));
         finalStatL{i,7}=(fstscanlist(ran,:));
         finalStatL{i,8}=(lstscanlist(ran,:));
      else
         ran=(pepcountlist2(i-1)+1):pepcountlist2(i);
         finalStatL{i,3}=(peplist(ran,:));
         finalStatL{i,4}=(chargelist(ran,:));
         finalStatL{i,5}=(mzlist(ran,:));
         finalStatL{i,6}=(mhpluslist(ran,:));
         finalStatL{i,7}=(fstscanlist(ran,:));
         finalStatL{i,8}=(lstscanlist(ran,:));
      end
elseif type(i)==3
   finalStatL{i,1}=type3;
   finalStatL{i,2}=prolist{i,:};
      if threes(i)==1;
         ran=1:pepcountlist2(i);
```

```
                finalStatL{i,3}=(peplist(ran,:));
                finalStatL{i,4}=(chargelist(ran,:));
                finalStatL{i,5}=(mzlist(ran,:));
                finalStatL{i,6}=(mhpluslist(ran,:));
                finalStatL{i,7}=(fstscanlist(ran,:));
                finalStatL{i,8}=(lstscanlist(ran,:));
            else
                ran=(pepcountlist2(i-1)+1):pepcountlist2(i);
                finalStatL{i,3}=(peplist(ran,:));
                finalStatL{i,4}=(chargelist(ran,:));
                finalStatL{i,5}=(mzlist(ran,:));
                finalStatL{i,6}=(mhpluslist(ran,:));
                finalStatL{i,7}=(fstscanlist(ran,:));
                finalStatL{i,8}=(lstscanlist(ran,:));
            end
        else
          finalStatL{i,1}=type4;
          finalStatL{i,2}=prolist{i,:};
            if fours(i)==1;
                ran=1:pepcountlist2(i);
                finalStatL{i,3}=(peplist(ran,:));
                finalStatL{i,4}=(chargelist(ran,:));
                finalStatL{i,5}=(mzlist(ran,:));
                finalStatL{i,6}=(mhpluslist(ran,:));
                finalStatL{i,7}=(fstscanlist(ran,:));
                finalStatL{i,8}=(lstscanlist(ran,:));
            else
                ran=(pepcountlist2(i-1)+1):pepcountlist2(i);
                finalStatL{i,3}=(peplist(ran,:));
                finalStatL{i,4}=(chargelist(ran,:));
                finalStatL{i,5}=(mzlist(ran,:));
                finalStatL{i,6}=(mhpluslist(ran,:));
                finalStatL{i,7}=(fstscanlist(ran,:));
                finalStatL{i,8}=(lstscanlist(ran,:));
            end
        end
    end
 end

%REFORMAT (expand all cells)
 for i=1:length(finalStatL)
    startscan=[startscan; finalStatL{i,7}];
    endscan=[endscan; finalStatL{i,8}];
    charge=[charge; finalStatL{i,4}];
    mz=[mz; finalStatL{i,5}];
    peptide=[peptide; finalStatL{i,3}];
    mhplus=[mhplus; finalStatL{i,6}];

    for j=1:length(finalStatL{i,3})
        proteintype=[proteintype; finalStatL(i,1)];
        protein=[protein; finalStatL(i,2)];
    end
 end

[scanorder,idx]=sort(startscan);        %organizes by sorting scan
startscan=startscan(idx);
endscan=endscan(idx);
charge=charge(idx);
mz=mz(idx);
peptide=peptide(idx);
proteintype=proteintype(idx);
protein=protein(idx);
mhplus=mhplus(idx);

%FINAL OUTPUT (excel)
scanmatrix=cellstr([num2str(startscan) repmat(' - ',numel(startscan),1)
num2str(endscan)]);
% finalStatsL=[scanmatrix peptide num2cell(mz) proteintype num2cell(charge)
num2cell(mhplus) protein];
data(:,1)=scanmatrix; %scans
data(:,2)=peptide; %sequences
data(:,3)=num2cell(mhplus); %MH+
data(:,5)=proteintype;      %organism
data(:,4)=num2cell(charge); %charge
data(:,6)=protein;      %protein
    left=str2num(scanRange);
    right=left;
else
    left=str2num(scanRange(1:splitLoc-1));
    right=str2num(scanRange(splitLoc+1:end));
end
```

APPENDIX 15 LABEL COUNTING FUNCTION

```
% This function takes the sequence weight, observed weight (by sequest) and
% the label type (L/H), and uses these to determine the number of labels present
% by the difference of these weights.
function dOut=sequenceWeight(sequence, obsMW, labelType)
sequence=sequence(3:end-2);
mwTable={'G', 57.02146'; ...
    'A', 71.03711;
    'S', 87.03203;
    'P', 97.05276;
    'V', 99.06841;
    'T', 101.04768;
    'C', 160.03068 %103.00918+57.03404; % IAA reaction adds weight to C
    'L', 113.08406;
    'I', 113.08406;
    'N', 114.04293;
    'D', 115.02694;
    'Q', 128.05858;
    'K', 128.09496;
    'E', 129.04259;
    'M', 131.04048;
    'H', 137.05891;
    'F', 147.06841;
    'R', 156.10111;
    'Y', 163.06333;
    'W', 186.07931;
    '#', 79.96633;
    '*', 15.99492};

MW=0;
for i=1:numel(mwTable(:,1));
    MW=MW+(sum(sequence==mwTable{i,1})*mwTable{i,2});
end
MW=MW+18.0106;
labelWeight=(obsMW-MW)-1;
light=labelWeight/28.0313;
heavy=labelWeight/32.056408;
if labelType=='L'
    nLabels=light;
else
    nLabels=heavy;
end
dOut=round(nLabels);
```

APPENDIX 16 SEQUEST SCAN COMBINATIONS

```
%Prepares output of xlsExtract for use with SequestComare by combining
%duplicate sequences' scans into a single entry.
function outData=combineScans(inData,scanIndices)
outData.sequence={};
outData.z=[];
outData.MHMass=[];
outData.fullMsScans={};
outData.scanSource={};
chargeAndSequence=cellstr([num2str(inData.z),repmat(':',numel(inData.z),1),...
    char(inData.sequence)]);
peptideChargeList=unique(chargeAndSequence);
peptideList=cellfun(@(name)name(3:end),peptideChargeList,'uniformOutput',false);
chargeList=cellfun(@(name)str2num(name(1)),peptideChargeList);
for i=1:numel(peptideChargeList)
    thisSequence=peptideChargeList{i}(3:end);
    thisCharge=str2num(peptideChargeList{i}(1));
%     if all(~(thisSequence=='#')) % Not a PTM
        thisPeptideLocations=find(strcmp(inData.sequence,thisSequence) & ...
            inData.z==thisCharge);
        outData.sequence=[outData.sequence;{thisSequence}];
        outData.z=[outData.z;thisCharge];
        outData.MHMass=[outData.MHMass;inData.MHMass(thisPeptideLocations(1))];
        scanOutput=[];
        scanSource='';
        for j=1:numel(thisPeptideLocations)
            thisScanIdxStart=find(scanIndices<=...
                inData.scanStart(thisPeptideLocations(j)),1,'last');
            thisScanIdxEnd=find(scanIndices<=...
                inData.scanEnd(thisPeptideLocations(j)),1,'last');
            scanOutput=[scanOutput, scanIndices(thisScanIdxStart:thisScanIdxEnd)];
            scanSource=[scanSource,inData.scanType(thisPeptideLocations(j))];
        end
        outData.scanSource=[outData.scanSource;{scanSource}];
        outData.fullMsScans=[outData.fullMsScans;{scanOutput}];
        outData.scans=outData.fullMsScans;
%     end
end
```

APPENDIX 17 SEQUEST CROSS REFERENCER

```
function out=sequestCompare(pfResults,sequest,inData)
mzTolerance=4/12;
timeTolerance=1;

% Start by getting common terms together. The imported XLS file already has
% most of the stuff we need, but our data could use some changes.
class2Charge=[1 1 2 2 2 3 3 3];
class2Labels=[1 2 1 2 3 1 2 3];
index2massCharge=inData.mz;
index2scanNum=inData.scanIdx;

pfResults(:,5)=abs(pfResults(:,5));

myResults.MZ=index2massCharge(pfResults(:,1));
myResults.z=class2Charge(pfResults(:,5));
myResults.numLabels=class2Labels(pfResults(:,5));
myResults.unfixedMass=((myResults.MZ-1.007825).*myResults.z')+1.007825;
myResults.mass=myResults.unfixedMass;
myResults.scanStart=index2scanNum(pfResults(:,3))';
myResults.scanEnd=index2scanNum(pfResults(:,4))';
myResults.midScan=((myResults.scanStart+myResults.scanEnd)./2);


zMatch=arrayfun(@(sZ) find(myResults.z==sZ),sequest.z,'uniformOutput',false);


labelMatch=arrayfun(@(seq) find(myResults.numLabels==countLabels(seq{1})),...
   sequest.sequence,'uniformOutput',false);
expandedWindow=[myResults.scanStart,myResults.scanEnd];

timeMatch=arrayfun(@(sScans) find(arrayfun(@(mScansS,mScansE) ...
   any(ismember(sScans{1},mScansS:mScansE)),expandedWindow(:,1),...
   expandedWindow(:,2))),sequest.scans,'uniformoutput',false);

 massMatch=arrayfun(@(sMass,sCharge) find(myResults.mass<=sMass+...
    (mzTolerance*sCharge) & myResults.mass>=sMass-(mzTolerance*sCharge)),...
 sequest.MHMass,sequest.z,'uniformOutput',false);
 finalMatch=cellfun(@(z,label,time,mass) intersect(intersect(z,label),...
    intersect(mass,time)), ...
   zMatch,labelMatch,massMatch,timeMatch,'uniformOutput',false);
   sum(~cellfun('isempty',finalMatch))
```

```
    sequestIndex=~cellfun('isempty',finalMatch);
    myIndex=cellfun(@(myIdx,seqScans) myIdx(nearest(...
        myResults.midScan(myIdx),seqScans)),finalMatch(sequestIndex),...
        sequest.scans(sequestIndex),'uniformOutput',false);
    mySingleIndex=cellfun(@(myIdx,seqScans) myIdx(nearestSingle(...
        myResults.midScan(myIdx),seqScans)),finalMatch(sequestIndex),...
        sequest.scans(sequestIndex));
      out.unfixedPITSMHMass=myResults.unfixedMass(mySingleIndex);
      out.fixedPITSMHMass=myResults.mass(mySingleIndex);
      out.z=myResults.z(mySingleIndex)';
      out.sequestMHMass=sequest.MHMass(sequestIndex);
      out.sequence=sequest.sequence(sequestIndex);
       out.PITSTimes=cellfun(@(idx) [myResults.scanStart(idx),...
          myResults.scanEnd(idx)],myIndex,'uniformoutput',false);
      out.sequestTimes=sequest.scans(sequestIndex);
      out.sequestIndex=find(sequestIndex);
      out.PITSIndex=mySingleIndex;
      out.allPitsMatches=myIndex;
end

function nLabels=countLabels(sequence)
    sequence=sequence(3:end-2);
    nLabels=sum(sequence=='K')+1;
end

function nearestIndex=nearest(myScans,seqScans)
seqRange=min(seqScans):max(seqScans);
minDists=arrayfun(@(myScan) min(abs(myScan-seqRange)),myScans);
nearestIndex=find(minDists==min(minDists));
end

function nearestIndex=nearestSingle(myScans,seqScans)
seqRange=min(seqScans):max(seqScans);
minDists=arrayfun(@(myScan) min(abs(myScan-seqRange)),myScans);
nearestIndex=find(minDists==min(minDists));
nearestIndex=nearestIndex(1);
end

function clickFcn(object,~,~)
persistent waiting lastPt lastLine;
    rawData=evalin('sequestCompare','rawData');
    index2scanNum=evalin('sequestCompare','index2scanNum');
```

```
thisPt=get(object,'currentPoint');
if (waiting == 1)
    hold on;
        try
            delete(lastLine);
        end
    lastLine=line([lastPt(1),lastPt(1),thisPt(1),thisPt(1),lastPt(1)],...
        [thisPt(3),lastPt(3),lastPt(3),thisPt(3),thisPt(3)],...
        'hittest','off','color','white','linewidth',2);
    hold off;
    timeCoords=find(abs(index2scanNum-min(lastPt(1),thisPt(1)))==...
        min(abs(index2scanNum-min(lastPt(1),thisPt(1)))));
    timeCoords(2)=find(abs(index2scanNum-max(lastPt(1),thisPt(1)))==...
        min(abs(index2scanNum-max(lastPt(1),thisPt(1)))));
    massCoords=((min(lastPt(3),thisPt(3))-400)*12):((max(lastPt(3),...
        thisPt(3))-400)*12);
    rawData=rawData(uint16(massCoords),timeCoords(1):timeCoords(2));
    figure(2);
    surf(double(rawData));
    figure(3);
    clf;
    copyobj(object,figure(3));
    set(gca,'xLim',[min(lastPt(1),thisPt(1)) max(lastPt(1),thisPt(1))],...
        'yLim',[min(lastPt(3),thisPt(3)),max(lastPt(3),thisPt(3))]);
    hold on;
    axes();
    [xGrid,yGrid]=meshgrid(index2scanNum(timeCoords(1):timeCoords(2)),...
        (massCoords/12)+400);
     contour(xGrid,yGrid,double(rawData));
     set(gca,'color','none');
    hold off;
    waiting=0;
else
    lastPt=thisPt;
    waiting=1;
end
end
```

APPENDIX 18 SOFTWARE PREREQUISITES AND USE

The toolchain described in this work is written as a contained workflow designed to be as easy to use as possible. While the algorithms are described previously, the complete suite is currently only available by request whilst it is still under development. The only prerequisite (if automated import of .RAW files is desired) is the installation of the ReAdW component of the trans-proteome pipeline toolsuite by the Seattle Proteome Center (this package additionally requires the XCALIBUR software to interface with .RAW files).

The software can be launched by running the 'modularizer.m' file contained within; this will display a flowchart-like interface illustrating all of the stages of the data analysis process developed; each of these steps may be clicked in sequence to open an additional dialog with information and details to complete that step. In this fashion, one may transgress through the workflow from start (.RAW file) to finish (saved list of detections and their properties) without needing a detailed list of MATLAB commands and scripts to run in a particular order. Additionally, the modular nature of the workflow allows those interested in expanding or customizing certain steps to easily create their own data manipulation “modules” at any point in the process.