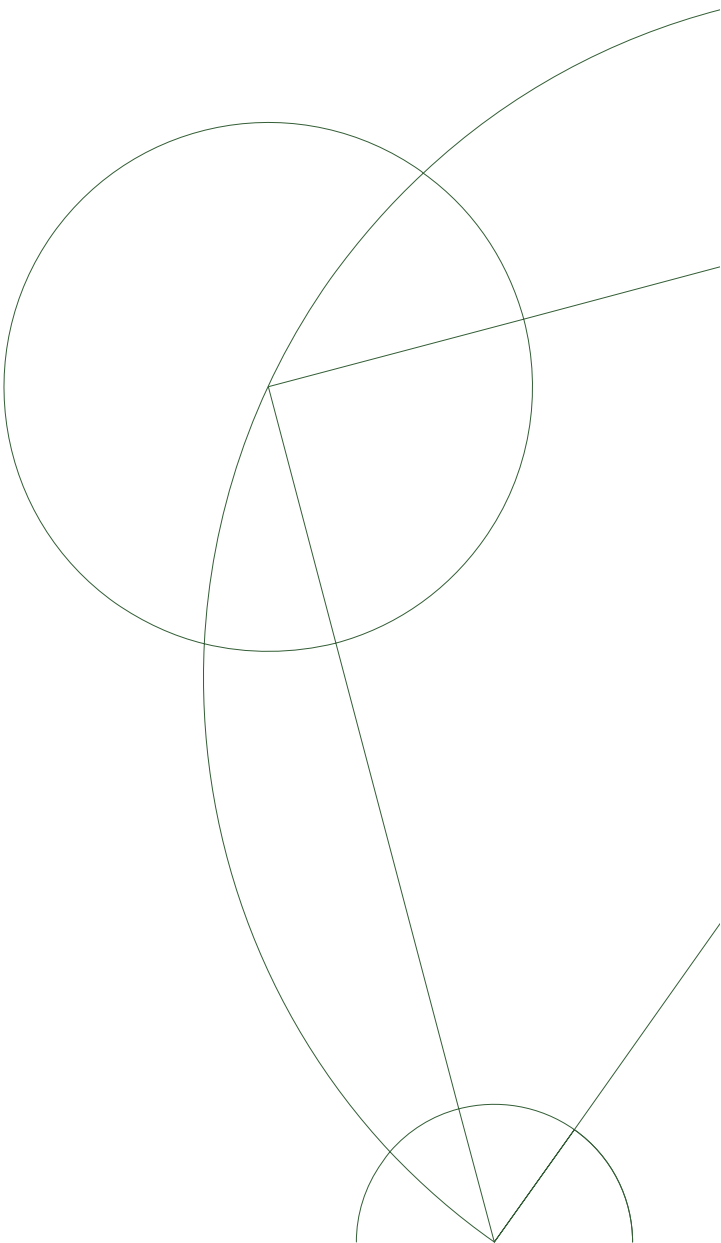
FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Bioinformatics Master Thesis

Development of label-free quantification methods in proteomics

Antonio Ortega Jiménez <rnq313@alumni.ku.dk>



Supervisors

Thomas Hamelryck <thamelry@gmail.com>

Mathias F. Gruber <mafg@novozymes.com>

Contents

Abbreviations

ELBO Evidence Lower Bound

ESI Electrospray Ionization

FDR False Discovery Rate

FT-ICR Fourier Transform Ion Cyclotron Resonance

HPLC High Pressure Liquid Chromatography

IID Independent Identically Distributed

IT Ion Trap

KL Kullback-Leibler (divergence)

log2FC $\log_2$(Fold Change)

m/z mass charge ratio

MALDI Matrix-Assisted Laser-Desorption Ionization

MCMC Markov Chain Monte Carlo

MS Mass Spectrometry

MS/MS Tandem mass spectrometry

MS1 first tandem MS analyzer

MS2 second tandem MS analyzer

NHST Null-Hypothesis Significance Testing

NUTS No-U-Turn-Sampler

NZ Novozymes A/S

PTM Post-Translational Modification

Q Quadrupole

RT Retention time

SC Spectral Counting

TOF Time of Flight

TPP Trans Proteomic Pipeline

VI Variational Inference

XIC Extracted Ion Current

Preface

TODO: get nomenclature straight: what is a sample and what is a fraction

Introduction

0.1 Aminoacids and proteins

Proteins represent the last link in the central dogma of biology, where information encoded in DNA, is transcribed to RNA for posterior translation into proteins at the ribosome.

Proteins are made up of 20 basic units, called aminoacids. All aminoacids share a common chemical structure, where a carbon atom (C_α) is covalently bonded to a hydrogen atom, a carboxyl group, an amino group, and last but not least, a radical, also called side chain of the aminoacid. The side chain differs between aminaocids and generates them from each other. A slight deviation from this pattern exists in proline, where the radical is bound to the nitrogen atom, making it an iminoacid. Even though the side chains are all different, they can be classified into four different groups: aliphatic, polar, positively charged and negatively charged (see figure 1).

Two aminoacids are joined together through the formation of a peptidic (covalent) bond between them. Such a linkage is formed by removal of the elements of water (dehydration) from the α-carboxyl group of one amino acid and the α-amino group of another [**?**]. The remaining α-amino and α-carboxyl groups are available for linkage to other aminoacids, and in this way peptidic chains or peptides can be created.

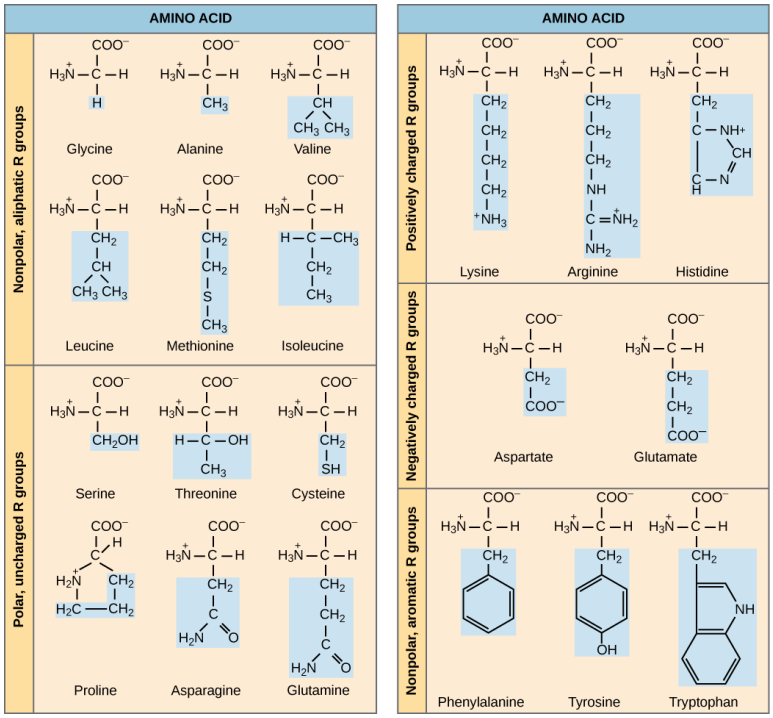


Figure 1 CAPTION AND REFERENCE

While there are 20 basic units that constitute the majority of naturally observable proteins, their side chains can be modified both by physiological processes and by experimental procedures cite. One frequent instance of such modifications is the oxidation of methionine.

0.2 The protein-focused biotechnology industry

Proteins carry out most of the cell's molecular functions, they work as molecular agents that can perform an extremely wide range of tasks. The advent of biotechnology has sought to take advantage of this power, either by using proteins as present in natural conditions (wild type) or engineered by humans. This potential economic activity is carried out by several biotech companies, including Novozymes A/S (NZ).

NZ is a company whose line of business consists of the development of enzymatic products performing chemical transformations in different industrial processes. The application of these products, instead of conventional chemical-based solutions, has the advantage that they require less chemical substances, potentially simplifying industrial processes, reducing their costs and their environmental impact. Notorious examples of such applications include waste-water treatment, household care and the baking industry.

The advancement of the way NZ does protein research is thus key to place the organization ahead of its competitors. The refinement of the currently used tools and the development of new ones could be of great significance for the company.

Protein research can be approached from different angles. This thesis exploited the combination of mass spectrometry (MS) and proteomics workflows (see chapter 1) for the qualitative and quantitative characterization of protein samples.

0.3 Objectives of the Thesis

In line with the goal of making NZ more competitive, this project aimed at the following objectives:

1. Develop an open-source, Linux based and easily deployable pipeline for the analysis of MS data, starting at the raw high-throughput data files and ending in the biological interpretation of the results.

2. Evaluate this pipeline with a benchmark dataset to assess if the pipeline is able to reflect the biological phenomena captured in the data.

3. Establish a label-free quantification probabilistic model that provides relative abundance estimates and a measurement of their uncertainty based on the available data.

0.4 Structure of the Thesis

An overview over the MS and following computational data analysis steps is presented in 1. The pipeline development and its benchmark are explained in chapters 2 and 4, while the modelling problem is introduced in chapter 3. Finally, a conclusion of the work is given in chapter .

Chapter 1

Review on mass spectrometry and shotgun proteomics

Life systems consist of complex systems, meaning their behaviour cannot be easily explained by analyzing the individual elements alone. Moreover, they present multiple layers of complexity, given by the nature of the elements that make it up. The layer provided by proteins is one of them, and its study is called proteomics. It is a complex layer because thousands of different proteins can be present in a single cell at any time, and their exact composition and quantities constantly change, responding to the stimuli of the surrounding environment. The study of the protein-specific complexity is called proteomics. With proteomics, one endeavors to infer the protein composition of a sample, and eventually quantify its protein amounts.

The existing approaches are divided into two types of paradigms: top-down and bottom-up. In the top-down paradigm, intact proteins are directly used for the analysis. In the bottom-up paradigm (see figure 1.1), the proteins are first cleaved into smaller parts, and these parts are then used

for identification, characterization, and quantification. These smaller parts are called peptides. [**?**] Such peptides acquire physicochemical properties fitting the requirements of the downstream analytical methods, mainly the mass spectrometer (MS), which performs the data acquisition. The bottom-up paradigm is most often used because peptides are much more suitable to analysis by mass spectrometry, as explained in 1.2.3. The top-down paradigm will be ignored in the rest of the manuscript.

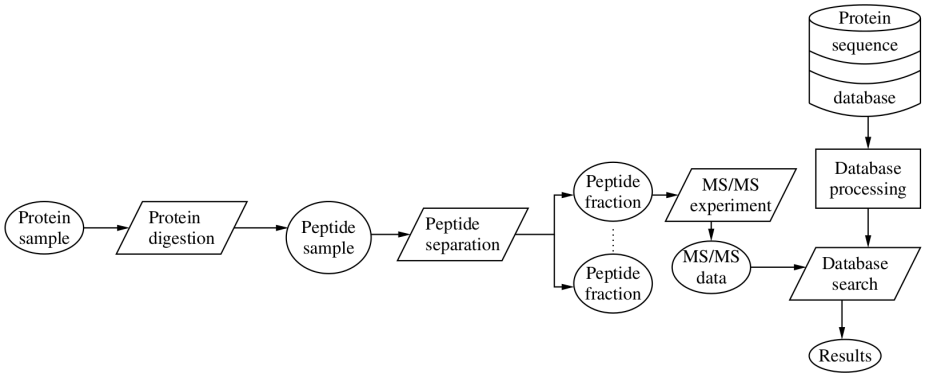


Figure 1.1 Overview of a standard bottom-up proteomics analysis. Figure 1.3 from [**?**].

MS is performed by means of a mass spectrometer, an ensemble of pieces of equipment that can acquire mass measurements for plenty of sample components. A detailed explanation of the sample processing required prior to MS is given in section , while an overview on mass spectrometers is given in section . The result of the MS analysis is a dataset that, with adequate computational analysis tools, is enough to perform the inference steps required to gather knowledge about the original protein sample. These inference steps can be condensed to the peptide and protein inference problems, explained in section . A third computational problem needs to be solved if quantitative, and not just qualitative information, is to be gained from the experiment. This is the quantification problem, explained in section 1.7.

A summary of the bottom-up approach MS analytical pipeline is provided in the rest of the chapter. It can be divided into two main steps:

1. MS analysis and data generation. Sections 1.1 to 1.3.

2. Computational analysis of data. Sections 1.4 to 1.7.

MS analysis and data generation

1.1 Sample processing

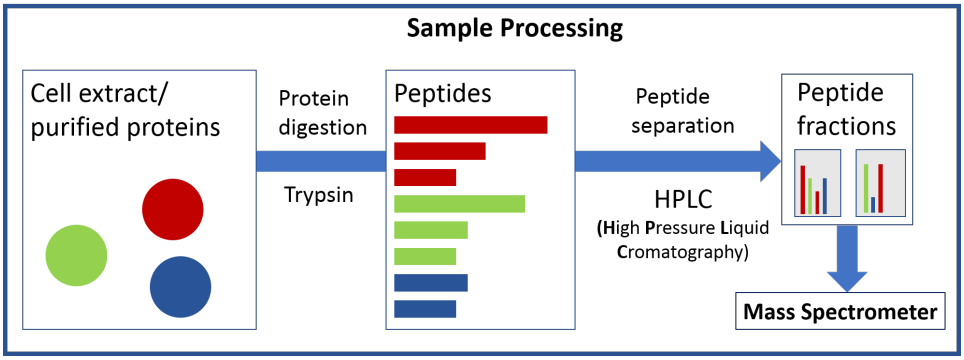


Figure 1.2 Overview of the sample processing step prior to mass spectrometry analysis. First, proteins are denatured and digested with a specific protease like Trypsin. This yields a peptide mix that is separated into peptide fractions that can be introduced in the mass spectrometer.

1.1.1 Protein digestion

An MS experiment starts with the generation of a protein sample from the biological system of interest. Proteins are then denaturalized so as to remove bias due to the divergent properties acquired by folded proteins. Then, proteins are subjected to digestion with specific enzymes, which cut the aminoacidic chains following a predictable pattern. Trypsin is the most frequently used for this task. It cuts peptidic bonds whenever a positively

charged residue, either Lysine (K) or Arginine (R), lies on the carboxyl side of the peptidic bond. Since roughly 1/10 residues are either R or K, the average peptide length is 10 residues. As explained in 1.2.3, this length distribution is fitted to the resolution of the MS analyzer. Moreover, since R and K are positively charged aminoacids, the resulting peptide is guaranteed in most cases to be able to capture at least one charge, thus making it visible in the spectrometer. All of these properties combined, together with its low prize, makes Trypsin the enzyme of choice in this step for most cases.

Even though enzymes are very specific, the cleaving process is far from perfect, as there could be: [**?**]

1. Missed cleavages.
2. Unsuspected cleavages during the maturation/life cycle of the protein.
3. Unexpected cleavages occurring either in the wet-lab procedure of the proteolytic treatment.
4. Naturally occurring, intentionally or unintentionally induced chemical modifications.

Missed cleavages can happen due to steric impediments or the presence of specific aminoacids that can weaken the enzyme's function. This is the case of Trypsin whenever the residue on the other side of the peptidic bond is Proline. Altogether, a variability is created in the cleavage process that, though limited, needs to be taken care of in downstream analysis, as it could introduce biases in peptide observability.

The result of this process is a complex mix of peptides, made up by hundreds or thousands of different molecules, following a length distribution given by the cleavage sites frequency and each protein's aminoacidic composition.

A peptide separation step is required before introducing the sample in the spectrometer.

1.1.2 Peptide separation

If presented with the problem of analyzing a mixture of peptides, the capacities of mass spectrometers are easily overwhelmed by a too complex mixture, resulting in the analysis of only a minor part of the total protein of the sample. This can be surmounted by independently analyzing different fractions of the sample over time. The required sample separation is achieved by High-Performance Liquid Cromatography (HPLC) methods, like reverse phase chromatography (separating on hydrophobicity) and strong cation exchange chromatography (separating on isoelectric point) [**?**].

During HPLC, the peptide mix loaded into a column containing a stationary and a solid phase. These phases create an environment where peptides interact differently based on their physico-chemical properties, set by the nature of the phases. The output of the column, called elute, will consist of subsets or fractions of peptides leaving the column at different retention times (RT) i.e the amount of time passed before the peptide is observed in the mass spectrometer. Therefore, the input to the machine will consist of a simplified mix of peptides at a time.

1.2 The mass spectrometer

The mass spectrometer is the ensemble of pieces of equipment analyzing a peptide mix such as those generated following the workflow enunciated in sections 1.1 and 1.1.2. It consists of three main parts: an ion source, a mass

analyzer, and a detector (see figure 1.3).

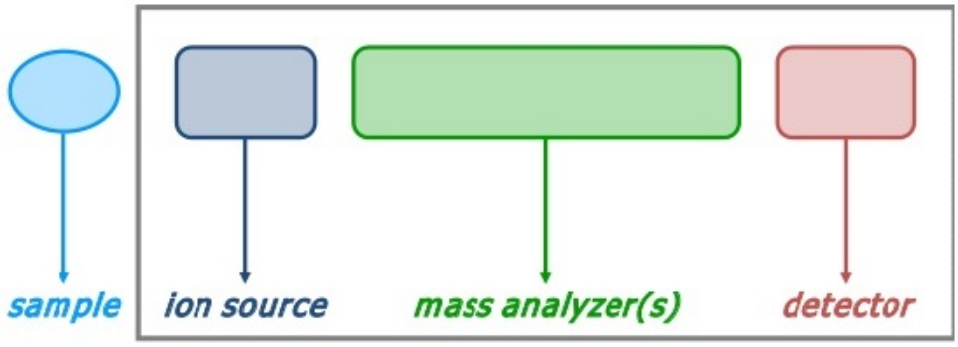


Figure 1.3 Schematic view of a mass spectrometer. Taken from [1]

1.2.1 The ion source

All mass spectrometers exploit the physical properties of mass and electric charge exhibited by the analyzed components. Ionization of the analytes is absolutely essential prior to any measurement, as analytes left uncharged will be unobservable to the equipment.

This step is performed in the ion source [**?**]. The most frequent ionization methods in proteomics are Matrix-Assisted Laser Desorption-Ionization (MALDI) and Electro Spray Ionization (ESI) [**?**]. Most peptides ionized by MALDI will acquire a single charge, whereas ESI can provide multiple charges (+2, +3, etc) too. Thus, the charge exhibited by an ion is not obvious when produced via ESI. Moreover, ESI can be run online with the right robotic equipment, while MALDI demands waiting time for vacuum generation. Finally, due to the chemical nature of the matrix components, MALDI ionizes more easily peptides containing aminoacids featuring aromatic rings (PYW), thus introducing a bias. Bias in ESI is less predictable.

[1] https://www.slideshare.net/joachimjacob/bits-introduction-to-mass-spec-data-generation

This is known as the competitive ionization problem [**?**].

The acquired charge yields a mass/charge (m/z) ratio, a property that is applied in the downstream component separation and measurement steps.

1.2.2 The mass analyzer

The plethora of ion separation methods is reflected upon the range of different analyzers available, mainly time of flight (TOF), Ion trap (IT) and quadrupole (Q). These apply different principles to perform the same task: separation (analysis) of the ion mix by the m/z ratio.

Moreover, two other analyzers exist which combine mass analysis with intensity measurement. These are Fourier Transform Ion Cyclotron Resonance (FT-ICR) and Orbitrap. They both register cylotron resonance frequencies that are Fourier transformed into the spectrum space. Remarkably, FT-ICR exhibits great resolving power, at the cost of high maintenance costs and difficult operability [**?**].

1.2.3 The detector

Detectors measure the intensity of an incoming ion signal. The ion's m/z ratio is known thanks to the previous mass analysis step. Performed for enough m/z ratios, the detector can produce a MS spectrum, which shows the intensity of ion current over an m/z range. Some topics in signal detection in MS need to be discussed.

On the one hand, the precision of the signal measurement is given by its mass resolution It is conventionally defined as the closest distinguishable separation between two peaks of equal height and width [**?**]. The resolution

decreases as the m/z ratio increases because small increments in the m/z ratio become negligible at high m/z ratios. This is one of the reasons why proteins are better fit for analysis when digested into peptides, as m/z are reduced, thus increasing the mass resolution.

On the other hand, due to the natural occurrence of isotopes, particularly ^{13}C, the same peptide will induce the measurement of several signals with very close m/z values. They constitute the isotopic envelope of the ion (see figure 1.4), and represent the signal created by peptides containing an increasing number of ^{13}C atoms. Every time a ^{12}C is replaced by ^{13}C, the mass increases by 1 Da. Even though the natural abundance of ^{13}C is 1.1 %, the sheer number of carbon atoms in a peptide makes it likely that at least one or even more carbon atoms will be ^{13}C, eventually driving the pure ^{12}C signal to comparatively small intensity values, and down to intensities below the background noise. Such event can be problematic if it entails that the ^{13}C peak is confused for the ^{12}C peak.

The resolution achieved by modern equipment allows for the distinction of each individual signal in most isotopic envelopes. Remarkably, the separation across peaks in the envelope can be used to infer the charge of the peptide, as increases of 1 Da at charge 1 will induce a separation of 1 m/z, while at charge 2 it will be $1/2 = 0.5$ m/z, at 3 $1/3 = 0.33$ m/z, and so on (see figure 1.4).

It is up to the MS technician to decide on the best pieces of equipment according to their availability and particularities of the dataset.

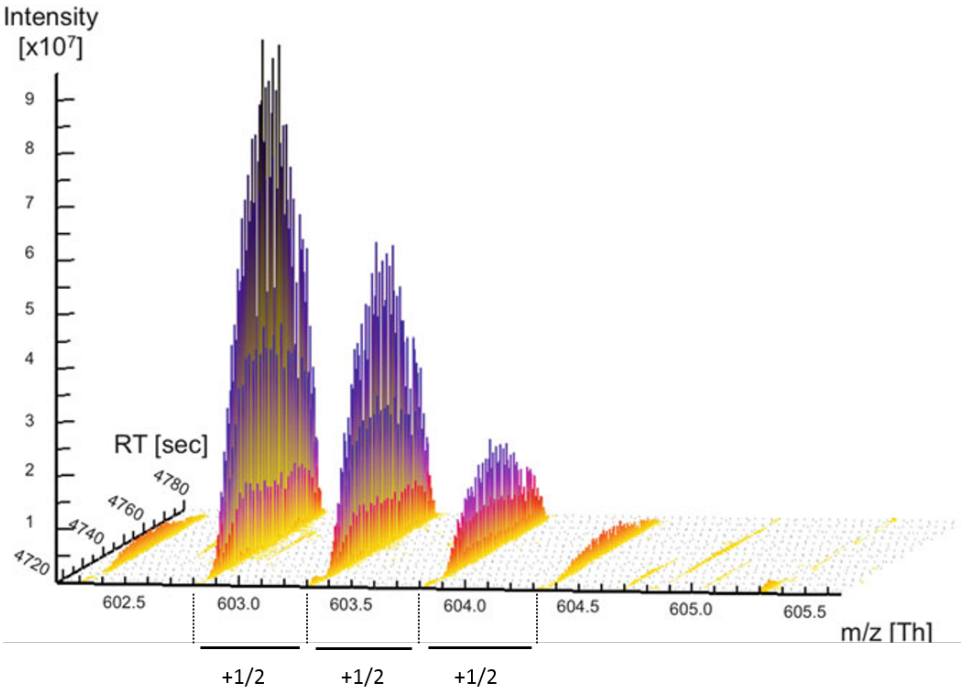


Figure 1.4 A doubly charged isotopic envelope with its monoisotopic ion measured at 602.8 Th. Each peak in the envelope is separated by 0.5 Th. This is explained by the peptide having 2 positive charges that make every extra Da in the ion mass account for 1/2 extra Th. Adapted from [**?**].

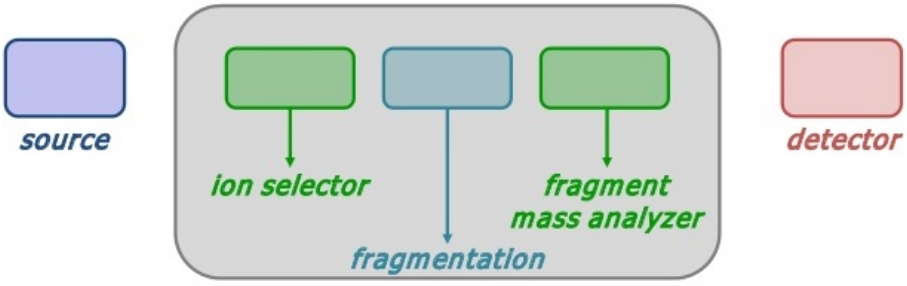


Figure 1.5 Illustration of the tandem MS workflow. The first spectrometer acts as an ion selector, that lets through ions with a given m/z ratio. The second spectrometer does not perform mass analysis, but instead provides the medium where peptide fragmentation occurs. Finally the third spectrometer records fragment mass spectra. Taken from [3].

1.3 Tandem MS workflow

[3] https://www.slideshare.net/joachimjacob/bits-introduction-to-mass-spec-data-generation

Shotgun proteomics analyses make use of two or more mass spectrometers connected in series, giving rise to the so-called Tandem MS (MS/MS) workflow. In this setting, each mass spectrometer collects a different type of spectra and thus different information (see figure 1.6. An extra spectrometer, usually a quadrupole, is introduced in between.

- The first spectrometer records the intensity versus m/z ratio of the peptides eluting from the column at a given time and is used to filter ions exhibiting a selected m/z ratio.
- The ions filtered in the first spectrometer undergo fragmentation in the second spectrometer.
- The last spectrometer records the intensity versus m/z ratio of the fragments produced in the previous step. The produced spectrum can be use to read out the peptide sequence.

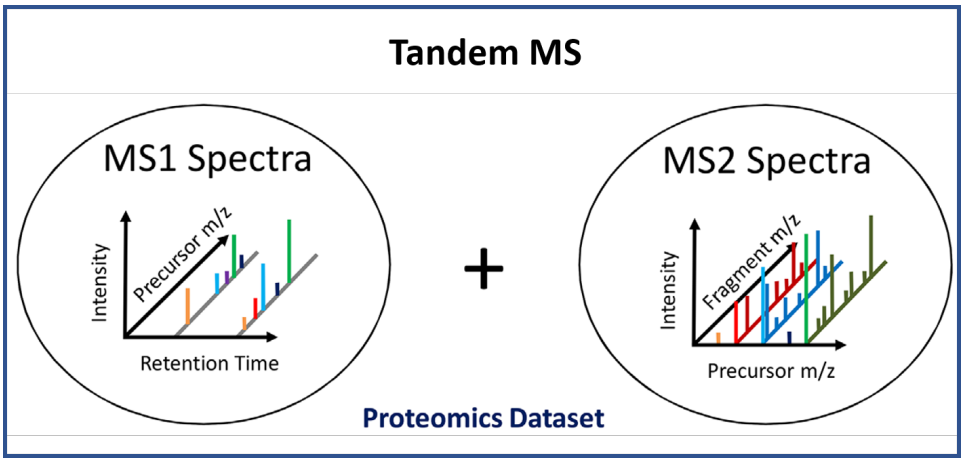


Figure 1.6 Illustration of the different spectra collected in tandem MS. MS1 spectra record precursor intensity vs m/z ratio at different times. MS2 spectra record the same magnitudes but the signal is generated by the fragments produced during fragmentation by the ion filtered in the first spectrometer. Altogether, they enable peptide identifications. Figure adapted from [**?**].

1.3.1 Fragmentation

The information that can be extracted from MS1 scans consists of the m/z ratio and retention time of the peptide ion, but not its sequence. Having the latter is paramount if the spectrum is to be matched to a theoretical peptide.

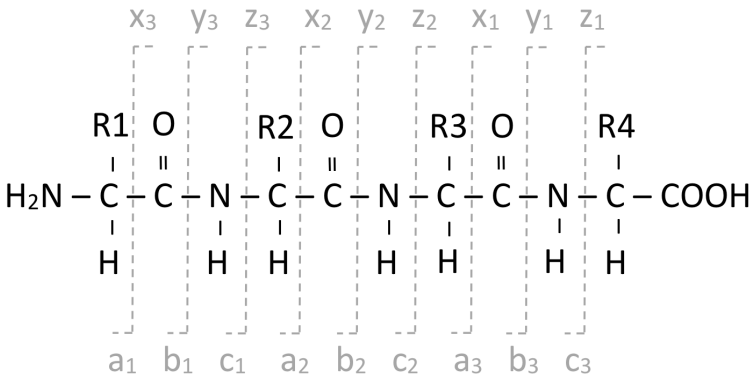


Figure 1.7 The common fragments and their relation to the peptide sequence can be organized into 2 groups of 3 series each. The abc fragments keep the N-terminal residue, while xyz keep the C-terminal one. Specific fragmentation techniques make fragments belonging to one series more likely than others. Other fragments are possible but much less likely. The fragment nomenclature was introduced in [**?**].

Fortunately, the peptide sequence of the ion can be inferred if fragmentation is performed. During peptide fragmentation, bonds along the peptidic chain are broken, turning the peptide into smaller fragments. These fragments will consist of truncated versions of the original peptide at different positions, thus making it possible to read an m/z ratio difference between any pair of fragment ions. The difference can be exploited to infer which aminoacid is present in the longer fragment. If this process is repeated for enough pairs of contiguous fragments, with the right software, a sequence can be read from the MS2 spectrum. explained in 1.4.

Computational analysis

1.4 Spectra processing: search engines

Computational analysis of MS data starts with the matching of the spectra to a referene proteome (see figure 1.8). MS search engines are capable of performing the crucial step of peptide to spectrum matching (PSM). During this step, search engines perform *in silico* prediction of the peptides produced in the protein digestion step and their expected spectrum.

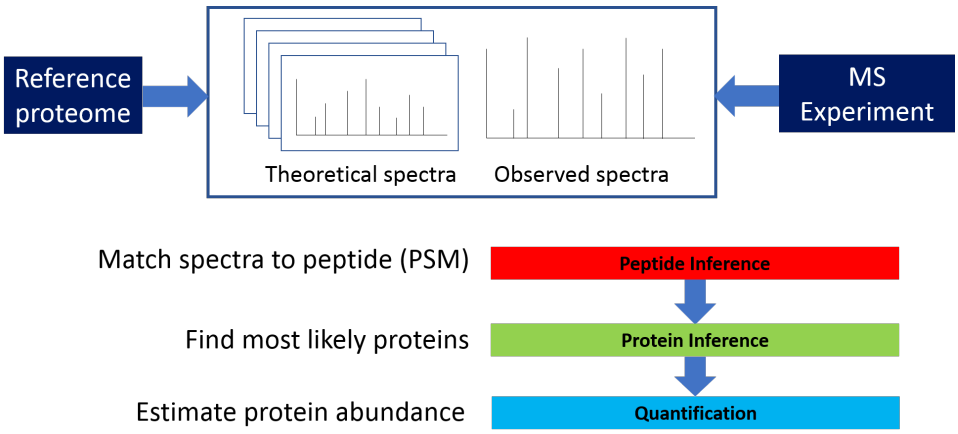


Figure 1.8 Diagram of the MS computational analysis. The PSM process maps the observed spectra to a list of peptides in the reference proteome that could have generated them. In other words, the software infers what peptides were introduced in the spectrometer. The analysis continues during protein inference and quantification.

Given the stochastic nature of the protein cleavage and spectra recording processes, the resulting spectra exhibit variability manifested in missing peaks or spurious ones. Furthermore, random (wrong) matches can be returned by the PSM process when running against a sufficiently big database. This translates to the generation of multiple matches, of which one, if any, will be correct. Therefore, the lists of matches need to be somehow ranked

by goodness-of-fit. The issue is addressed by search engines through the deployment of statistical models that provide scoring systems. Assuming the correct protein is present in the database, a good scoring system should give the best score to the right peptide. Under this circumstances, if repeated for several peptides, enough evidence for the presence of individual proteins can be collected.

Multiple search engines exist that implement different matching and scoring algorithms. The most modern ones include MS-GF+, MS-Amanda, Comet, X!Tandem or Andromeda. Notably, the results of each individual search engine can be combined to gather their strengths, at the expense of an increased computational cost and time [**?**].

1.5 Validation and quality control

The scoring systems implemented in search engines provide the best matches, but they are bound to contain false identifications. Nevertheless, these scores can be used to apply a filter that aims at minimizing the amount of errors.

A common filter is the false discovery rate (FDR), usually set to 1%, indicating that after the its application, only one out of a hundred filtered matches are expected to be false positives (wrong matches).

The most commonly used method to compute the FDR of a list of matches is the target-decoy search. Using this method, the search engine replicates the matching process, using the same spectra, but instead against a decoy database. The decoy database is generated by reversing or more generally applying a randomization technique upon the sequences present in the orig-

inal database (target).

All matches to the decoy are by definition wrong. Since the basic properties of the decoy (size, composition, etc) remain identical to those of the target the amount of matches to the decoy exhibiting more than a given score s can be regarded as an estimate of the number of false identifications ($\hat{n}_{fp}$) in the list of target results exhibiting at least the same score. This is because the existence of shared properties entails that random matches are equally likely to happen in both databases [**?**]. Together with the number of PSMs passing a given score in the target ($n_{tp} + n_{tp}$), the FDR can be computed using equation 1.1.

$$FDR = \frac{\hat{n}_{fp}}{n_{fp} + n_{tp}} \tag{1.1}$$

The equation tells us that we can compute the FDR at any score by counting how many decoy hits have a greater score ($\hat{n}_{fp}$), and dividing by the length of our target hits list. Thus, the score that makes the FDR equal to a predefined value, frequently 0.01 or 1 %, can be computed and used as threshold for the target hits.

The minimal FDR at which a given PSM is considered a valid match constitutes the PSM's q-value, i.e it is the smallest FDR we can allow while still keeping the PSM. Related to the q-value, the Posterior Error Probability (PEP) is an estimate of the probability of a given PSM of being an incorrect assignment (see figure 1.9). The PSM's confidence is just defined as $1 - PEP$ [**?**]. PEP can be computed from the decoy search results and provides another useful measurement of the uncertainty in the target results.

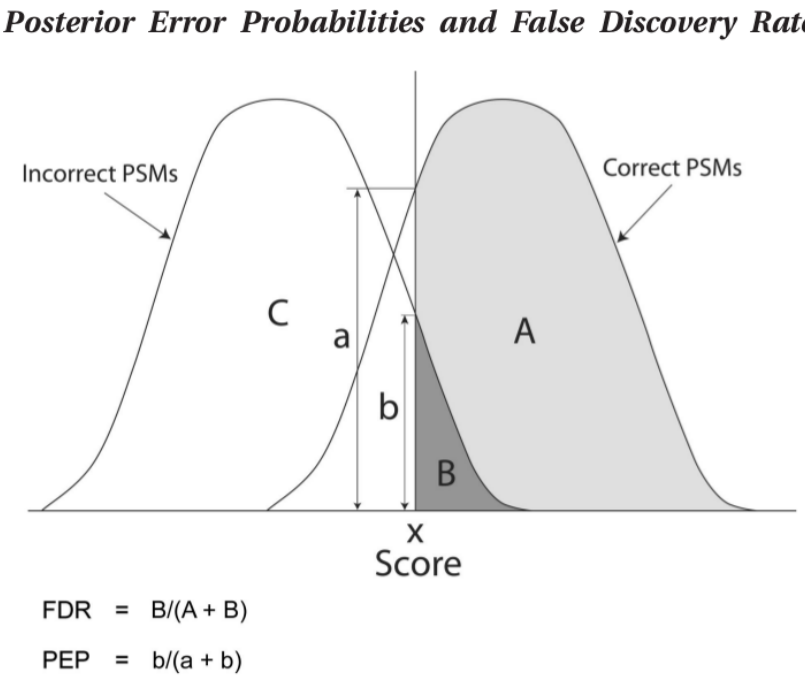


Figure 1.9 Visualizing FDR and PEP. The FDR is defined as the ratio between False Positives and the sum of True and False Positives found in the list of PSMs of score greater than s. The PEP corresponds to the same ratio, but only a specific score, hence its alternative name of local-FDR. Figure from [**?**].

1.6 Peptide and protein inference

Two steps in protein identification can be distinguished:

1. **Peptide inference**: infer the peptides present in the sample.
2. **Protein inferece *proper***: based on the inferred peptides, infer what proteins generated them. This is not trivial as peptides are degenerate and frequently map to more than one protein.

The result of the PSM step returns the inferred list of peptides. The ensemble of proteins most likely to have generated the list of peptides stemming from the filtered PSMs can be inferred using different algorithms. The de-

generate nature of peptides is dealt with the Occam's razor principle, which states that the most likely solution is the simplest one. Thus, protein inference algorithms aim at explaining the maximum amount of peptides using the least amount of proteins.

1.7 Protein quantification

The combination of all the aforementioned computational analyses yields a list of protein identities that reports the protein composition i.e qualitative information of the original sample. However, in most proteomics applications, quantitative data can be of great interest, as many biological phenomena are manifested mainly through changes in the protein abundances, rather than protein presence alone. For instance, cancer cells in response to a drug could modulate the abundance of several proteins without removing them from the cytosol or introducing new ones.

Protein quantification pipelines can be classified based on whether isobaric labelling was used (label-based) or not (label-free). These are explained in subsection 1.7.1. If the label-free approach is employed, more distinctions can be made based on:

- The proxy used for quantification: spectral counting (SC) or extracted ion currents (XIC). These are explained in 1.7.2
- The way the data are brought to the protein level from the peptide level: summarization-based vs. peptide-based, explained in 1.7.3.

1.7.1 Label-based and label-free approaches

Two paradigms exist in protein quantification: label-based and label-free. In label-based quantification, originally identical peptides from a number of different samples are made distinguishable by their masses via the incorporation of a label. All label-based methods simultaneously analyze several samples in each experiment, removing the difficulties associated with between-run variability [**?**]. The finite number of "plexes" available for a given label sets the limit to how many samples can be differentially quantified [**?**]. Different techniques, like Stable Isotope Labeling by Amino acids in Cell culture (SILAC) or Isotope-Coded Affinity Tags (ICAT), differ in the nature of the label and the way it is introduced. Remarkably, peptide labeling costs can be quite high. This, together with the limited amount of samples that can be compared makes a case for label-free quantification.

In the label-free quantification approach, peptides from different samples are not labelled differently and are thus distinguished by their presence in different, independent MS runs. In order to account for the introduced inter-run variability in peptide identifications and RT, a match-between-runs (MBR) processing step can be carried out (see figure 1.10).

This way, extra identifications are attained through a reanalysis of the spectra, factoring in the information collected in replicate runs. The RT and precursor mass of unidentified spectra in one replicate is matched to that of identified spectra in the remaining replicates. As a consequence, more identifications with a lesser fraction of missing datapoints are achieved.

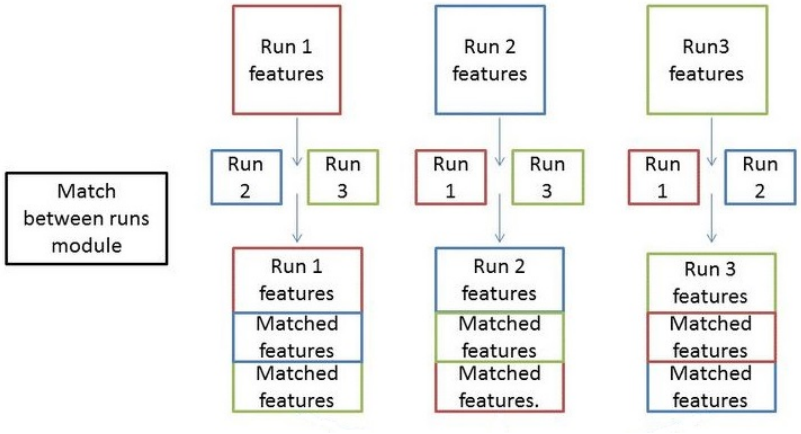


Figure 1.10 Illustration of the MBR step. **A** the information gathered from matches in replicate runs is put to use during a reanalysis of the spectra. Modified from supplementary information in [**?**].

1.7.2 SC and XIC based quantification

Quantification can be SC or XIC based.

On the one hand, spectrum counting based quantification is the simplest method in proteomics. The number of peptides a peptide is detected in the spectrometer is detected as proxy for its abundance. It relies on the rationale that highly abundant peptides will have a higher intensity and are thus more likely to trigger the acquisition of MS/MS spectra. These methods have the advantage that they are very simple to implement and don't require any further data processing.

On the other hand, XIC based methods rely on intensity measurements at any level of the MS workflow as proxies for protein abundance. A wide range of algorithms are available to process these data and output estimates of protein abundance. All of them require an intensive preprocessing step, usually including (I) taking the log_2 intensity to make the distributions symmetrical and thus fit for diverse parametric tests, and (II) quantile normalization to

address between-runs variability in the intensity measurements. They can be classified in the bases of which MS level is used as proxy for the protein abundances and on whether or not a summarization step is performed to aggregate peptide-level data into protein-level data, or not.

As stated in [**?**], *although the abundance of proteins and the probability of their peptides being selected for MS/MS sequencing are correlated to some extent, XIC-based methods should clearly be superior to SC given sufficient resolution and optimal algorithms. These advantages are most prominent for low-intensity protein/peptide species, for which a continuous intensity readout is more information-rich than discrete counts of spectra.* For this reason, only the XIC approach will be regarded in the rest of the manuscript.

For a more robust XIC based quantification, a feature extraction step is usually executed to extract the apex intensity of the identified peak clusters, as shown in figure 1.11.

1.7.3 XIC-peptide-based models for label-free quantification

The data collected in the mass spectrometer refers to peptides originating from a latent protein composition, given by the original sample. However, the data interpretation requires the transfer of these peptide-level data into the protein level. This can be done by either (I) performing an aggregation of the peptide-level data, where a summary value of the peptide-level data is taken as representative for the protein-level data, or (II) performing the protein quantification directly at the peptide-level by means of linear regression models.

As stated in [**?**] *peptides originating from the same protein can indeed be considered technical replicates and theoretically should lead to similar abun-*

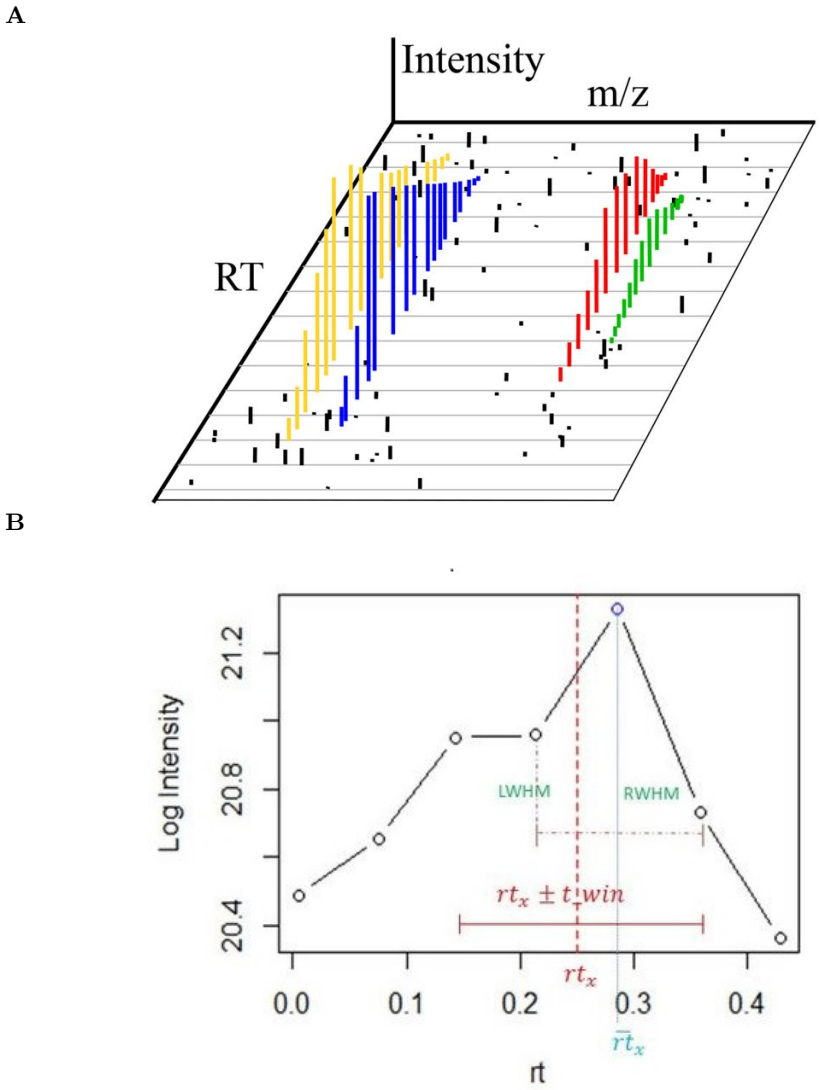


Figure 1.11 Illustration of the apex intensity extraction step.**A** Visualization of a 3D peak cluster on the RT / m/z plane. An ion current at the same m/z ratio, thus corresponding to the same precursor ion, is detected with varying degrees of intensity over a more or less narrow time window, spreading the signal over time. Taken from [**?**]. **B** A refinement analysis of these data attempts to fit a mathematical model of the signal over time and extract a representative measurement, such as the highest (apex) MS1 intensity. Modified from supplementary information in [**?**].

dance estimates. However, the summarization of the peptide intensities into protein expression values is cumbersome, and most summarization-based

methods do not correct for differences in peptide characteristics or for the between-sample differences in the number of peptides that are identified per protein. This might introduce bias and differences in uncertainty between the aggregated protein expression values, which are typically ignored in downstream data analysis steps.

It is for this reason that peptide-based models offer the statistical framework required to learn as much from the data as possible. This translates into improved results when compared to the other aforementioned methods [**?**]. This hypothesis is the motivation for the method explained in chapter 3.

Chapter 2

A label-free quantification proteomics pipeline

Summary

A pipeline making use of the set of tools published by the Compomics and StatOmics groups SearchGUI [**?**], PeptideShaker [**?**], moFF [**?**] and MSqRob [**?**] , was developed to support complete label-free protein quantification analyses using the most recent advances in the field with open-source software. The pipeline can be run on Linux computer clusters to perform (I) peptide to spectrum matching against a reference database, (II) quality control and filtering, (III) MBR and feature extraction, (IV) protein inference and (V) relative quantification. Its output can be passed to follow-up analyses in R or Python to get a biological interpretation of the results. A benchmark of its performance was accomplished using the proteome benchmark dataset published in [**?**]. The results exhibited an accuracy similar to those achieved by the MaxQuant [**?**] software, excepting a bias produced by the sample fractionation of this dataset.

2.1 Introduction

2.1.1 Background

Several proteomics pipelines are available on the internet under different licensing conditions. Many are released as closed-source commercial software, where information on how the program works is kept from the user. This is a serious drawback as it hinders the study of the implemented models and its customisation. Open-source, free alternatives, like the Trans Proteomic Pipeline (TPP) [**?**] or openMS [**?**] are nevertheless available for the community, but they either lack good documentation, good broad base or format exchangeability. MaxQuant, a freeware but closed-source, monolithic proteomics analysis suite [**?**], is developed for Windows and has only very recently been ported to Linux [**?**]. Nevertheless has been extremely successfully adopted by the scientific community due to its ease of use and comprehensive pipeline.

2.1.2 Goals

The development of a pipeline attempting to achieve the following goals will be described in this chapter.

1. Fully command line, documented and Linux-supported interface for easy automation and scalability.
2. Open-source for easy customisation and innovation of analyses.
3. Free-licensed and cost-free, so anybody with the knowledge can run it, democratizing the analyses.

2.2 Materials and Methods

2.2.1 Data generation and loading

The proteome benchmark dataset from [?] was reanalysed starting at the output RAW files available at the PRIDE repository [1]. Briefly, the *Homo sapiens* and *E. coli (strain K12)* proteomes were mixed in 1:1 (condition L) and 1:3 (condition H) proportions, with 3 replicates for each combination. Moreover, each of the 3 replicates of the 2 conditions was analysed over 24 fractions. This experimental setting thus generated a total of $2 \times 3 \times 24 = 144$ RAW files. One file was missing in the repository. The ThermoRawFileParser [2] program was used to convert RAW files to the MGF open format.

The MQ+LFQ pipeline results were obtained starting at the Supplemental table 1 from the MaxLFQ paper supplemental data [3] [?], which is available as an excel file. The results of the MQ+Rob pipeline employed the Levenberg-Marquandt minimised peptide intensities stored in the peptides.txt file contained in the spectraHeLaEColi.zip in the supplemental data.

2.2.2 Decoy database preparation and search

The spectra saved in the MGF files obtained in the previous step were passed to the MS-GF+ search engine [?] by means of the SearchGUI `SearchCLI` tool [?] utility. The search parameters were set using the `IdentificationParameterCLI`. In order to account for potential post-translational modifications, the search was conducted allowing for the following variable modifications:

[1] https://www.ebi.ac.uk/pride/archive/projects/PXD000279
[2] https://github.com/compomics/ThermoRawFileParser
[3] http://www.mcponline.org/content/13/9/2513/suppl/DC1

oxidation of M and deamidation of N and Q. Moreover, C carbamidomethylation was set as fixed modification. The enzyme was set to semispecific Trypsin, allowing for a non-tryptic cleavage on any side of the peptide. Up to two missed cleavages were allowed. The precursor tolerance was 10 ppm and the fragment tolerance 0.5 Da.

The target database was created by combining the Uniprot proteomes for *E. coli (strain K12)* (UP000000625) and *Homo sapiens* (UP000005640), downloaded in June 2018. The decoy database was created using the `FastaCLI` utility in SearchGUI by reversing all sequences in the target.

2.2.3 Quality control and validation

The SearchGUI results were filtered using the default built-in checks available in the PeptideShaker utility `PeptideShakerCLI` [**?**] [4]. By default, the FDR was set to 1%. PEP and confidence statistics were computed using the PeptideShaker built-in algorithms. Output was extracted via the Default PSM report txt file, available in the `ReportCLI` utility.

2.2.4 Data refinement

The moFF command line utility [**?**] was applied to perform (I) match-between-runs and (II) extract MS1 apex intensity of each peak cluster. This required passing the original RAW files, together with the Default PSM report from PeptideShaker. Output was exported to a peptide summary file, containing one row per peptide and for every peptide, the detected apex intensity in each sample.

[4] https://github.com/compomics/peptide-shaker/issues/300

2.2.5 Quantification

Relative quantification was performed using the MSqRob utility by passing the peptide summary file from moFF. Prior to quantification, the data was preprocessed using the `preprocess_MSnSet()` function. In a nutshell, (I) MS1 apex intensities were log_2 transformed, (II) quantile normalized, (III) peptides belonging to protein groups that contained one or more proteins that were also present in a smaller protein group were discarded [**?**], and (IV) protein groups with only 1 peptide were dropped.

Once preprocessing is done, a ridge regression model with Huber weights and empirical Bayes estimation of protein variance, implemented in the MSqRob package [**?**], was fit to every protein individually. The peptide and fraction effects were considered random, while the condition was treated as a fixed effect. The significance of the treatment effect differences was assessed through a Student's T test, implemented in the `test.contrast()` function. Only protein groups that could be unambiguously mapped to one of the organisms were considered for the analysis.

Quantification in the MQ+LFQ pipeline was executed using the LFQ intensity columns stored in the supplemental data file. Intensities were log2 transformed and averaged. The difference between conditions H and L was taken as estimate of the log2FC. Significance was evaluated by means of the two-tailed Welch Two Sample t-test implemented in the `t.test()` function in R. P-values were corrected using the FDR method implemented in the `p.adjust()` function in R.

2.2.6 Code implementation

2.3 Results

2.3.1 Thousands of spectra were identified and validated

The preprocessing of the RAW files produced by the mass spectrometer into the MGF open format enabled searchGUI to dispose of the registered spectra. PeptideShaker quality control and filtering capabilities (see figure 2.2) carried out the required search results validation. As expected, matches to the target and decoy exhibited similar score distributions at low score values, while a divergence is observed at higher score values. Likewise, the m/z error was found to be closer to 0 on validated PSMs than on those which did not pass the 1 % FDR filter. The application of this filter implied that the FNR (false negative rate) was set to 5 %, i.e 5 out of every 100 discarded matches were estimated to be true positives.

The 1% FDR cutoff selected PSMs with a score higher than 78, which translated to a confidence of at least 65%.

The combination of both programs enabled the identification and validation (matching) of thousands of spectra with high confidence in all samples. However, when compared to the total amount of spectra available, the percentage of matched spectra was on average 37.4%, with a marked decrease starting at fraction 14 (see figure 2.3).

2.3.2 Protein inference

Total peptides	Unique peptides	Non-unique	Proteins	Protein groups
46595	27384	19211	25056	10039

Table 2.1 Protein inference results. The counts of the different molecular entities detected by peptideShaker are displayed.

A total of 46595 peptides were inferred to be present in at least one of the fractions (see table 2.1). Of them more than 27 thousand mapped to a unique protein. However, a significant amount mapped to more than one protein. Such peptides are called non-unique peptides. The 25056 detected proteins were grouped into 10039 protein groups. Protein groups are peptide generating entities for which enough data to confirm the presence of at least one of them is available, but not to exactly assess which of them. Thus, protein inference algorithms create them when a does not uniquely map to a protein because it is not possible to distinguish which protein it truly comes from.

Protein groups are very frequent due to several factors. For example, protein isoforms, consisting of protein sequences differing in potentially only one aminoacid, are very difficult to resolve. PeptideShaker executes a smart protein grouping by harnessing the annotation of proteins and classifying protein groups based on how well the annotation backs the protein group.

However, in the present analysis the quality of the protein group was not taken into account, as explained in the Materials and Methods.

2.3.3 MBR and apex intensity extraction evaluation

The match between runs step allowed for increased identifications by transferring successful matches between replicate runs. The results of this process for the 13th fraction of the L condition is shown in figure 2.4.

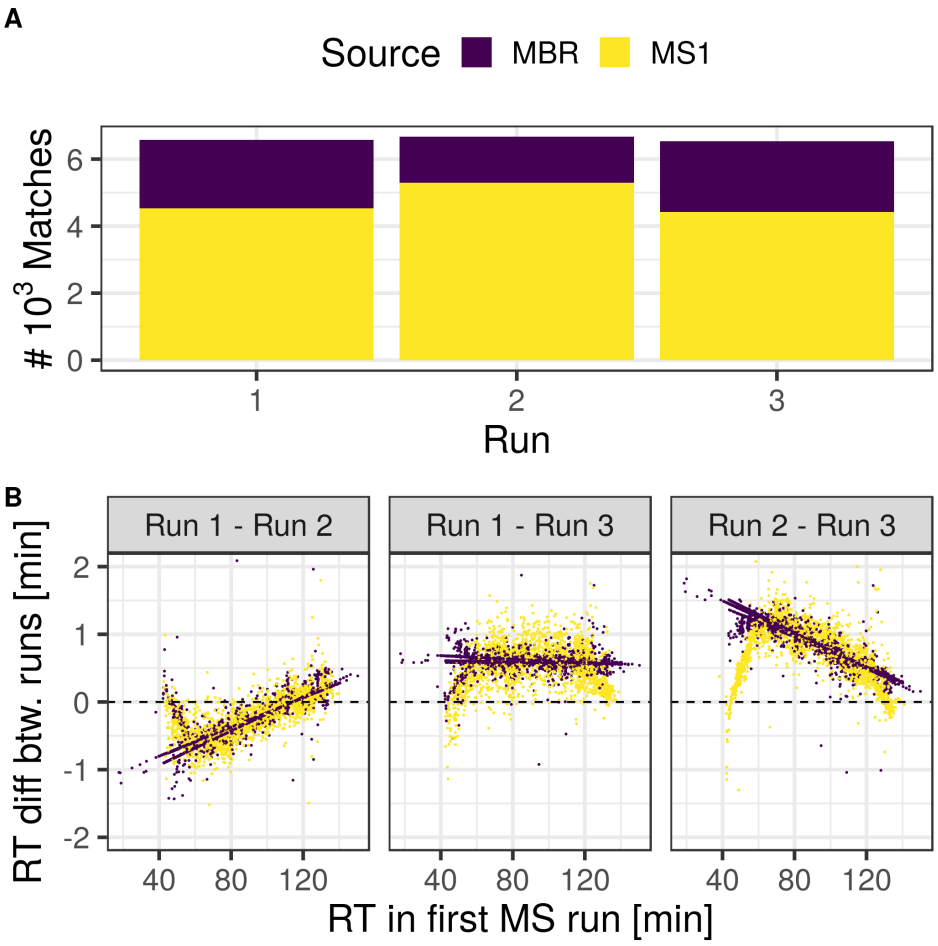


Figure 2.4 Match Between Runs with moFF. **A** Count of identifications on each run segreagated by source. More than 4k spectra were identified and validated by SearchGUI+peptideShaker. In this particular case, hundreds of new identifications were accomplished. **B** Match visualization. Every dot represents a peptide shared across 2 runs. The coordinate system illustrates its retention time on the first run on the x axis and the difference with the second run on the y axis. The color depicts whether the identification was carried out during the PSM process (MS1), or thanks to a cross-identification achieved by the MBR module (MBR).

Once as many identifications as possible were gathered, a refinement of the measured MS1 intensity can be implemented to select the apex of every peak cluster, which yields robust intensity measurements for each sample (see figure 2.5).

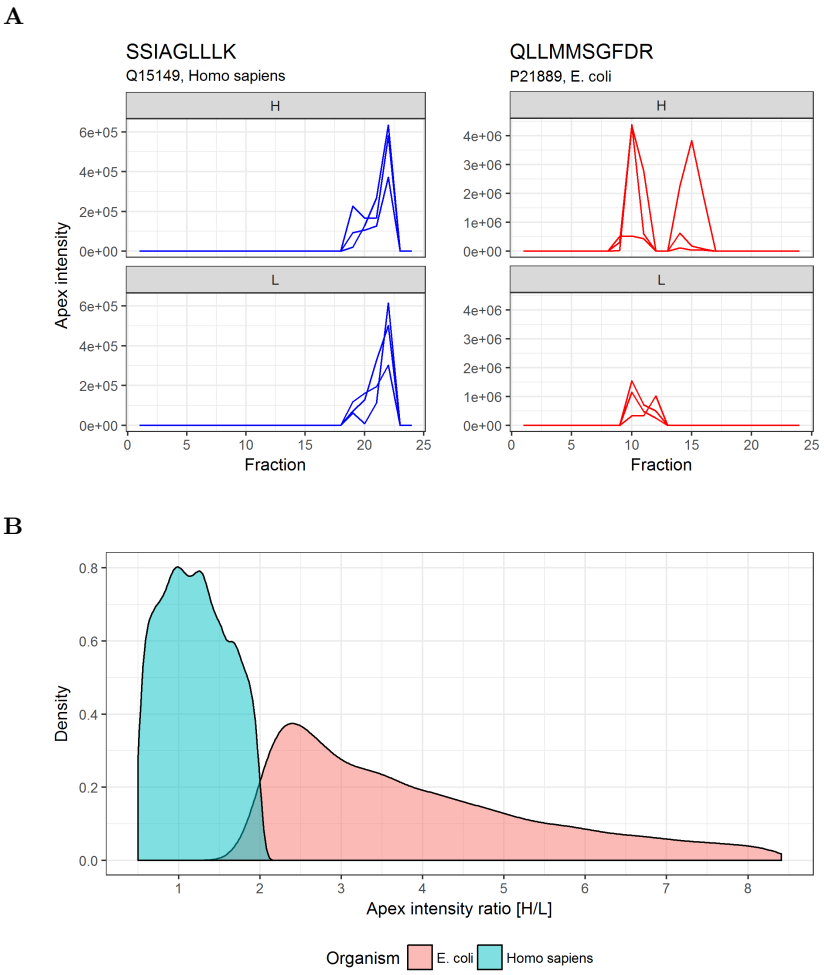


Figure 2.5 A The apex intensity profile across fractions for 2 different peptides, one from *Homo sapiens*, and one from *E. coli*. The figure illustrates the intrinsic intensity variability between technical replicates, particularly in the case of the *E. coli* QLLMMSGFDR peptide, as it was almost non-existent in one of the runs. **B** The expected pattern of overall similar intensities for the *Homo sapiens* data and 3-fold higher intensities for the *E. coli* data in condition H was observed, confirming an acceptable performance of the protocol.

2.3.4 Quantification

The extracted apex MS1 intensities were used as proxy for peptide abundance to estimate protein $\log_2$(ratios), or *log2-fold-changes* (log2FC) across condition H and condition L. During quantification, it is important to take into account as many of the effects influencing protein quantification. In this case, besides the different H and L conditions, peptide, run and fraction effects could be distinguished. The MSqRob quantification engine treats effects differently depending on whether or not they are random, or fixed. Fixed effects should have a consistent impact in the ion currents measured in the spectrometer, whereas random effects are truly random and thus unpredictable.

The result of the process was the successful quantification of 5307 protein groups out of 10039 (see table 2.2).

	log2FC	qval	Protein	*Organism*
1	-4.76E-01	1.42E-24	P78527	*Homo sapiens*
2	6.80E-01	6.75E-24	P0A8V2	*E. coli*
3	8.50E-01	4.39E-22	P25516	*E. coli*
4	7.07E-01	2.05E-18	P63284	*E. coli*
5	1.04E+00	1.21E-17	P37095	*E. coli*
6	7.78E-01	1.51E-16	P13029	*E. coli*
7	8.89E-01	5.62E-16	P77804	*E. coli*
8	8.53E-01	1.72E-15	P23721	*E. coli*
9	-6.97E-01	1.41E-14	P09874	*Homo sapiens*
10	1.37E+00	1.41E-14	P37666	*E. coli*

Table 2.2 Results of the quantification pipeline. The 10 protein groups with the lowest *q-val* as estimated by MSqRob are shown. Notably, eight correspond to proteins belonging to *E.coli*. The two human proteins exhibited in both cases an absolute value of the log2FC below 1.

The performance of the quantification can be evaluated thanks to the experimental design of the dataset. Since the *E. coli* proteome was mixed on

a 3:1 ratio in condition H, a log2FC of $log2(3) = 1.58$ is expected for proteins coming from this organism. Likewise, human proteins should have a log2FC of 0. As such, the evaluation can be reformulated as a classification task were the quantification algorithm tries to distinguish between proteins coming from one or the other organism.

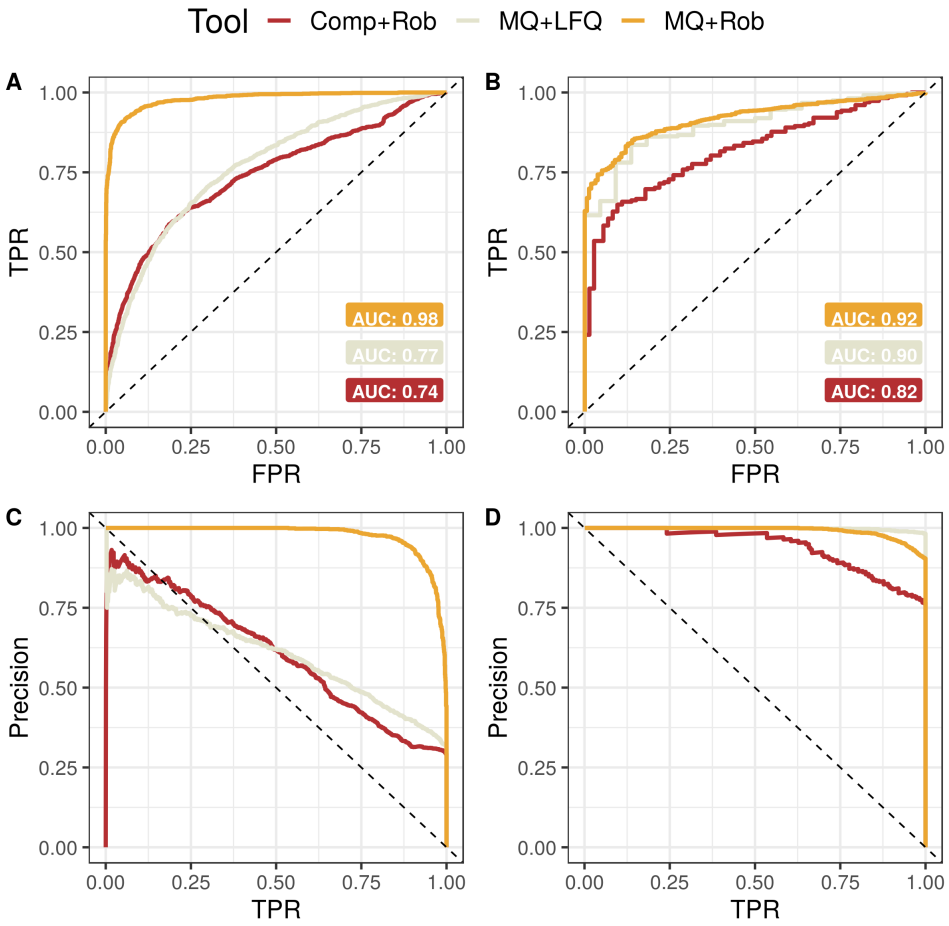


Figure 2.6 ROC and PR curves displaying the behaviour of 3 different label-free quantification pipelines publicly available. **Comp+Rob** is the pipeline presented in the previous sections. **MQ+LFQ** consists a standard MaxQuant upstream analysis combined with its LFQ quantification engine. Finally, MQ+Rob blends MaxQuant upstream analysis with the MSqRob quantification engine. **A, B** ROC curve produced by the three algorithms with no log2FC filter, and keeping proteins with log2FC greater than 1, respectively. **C, D** PR curve produced by the three algorithms with no log2FC filter, and keeping proteins with log2FC greater than 1, respectively.

Some of the most frequent evaluations of the performance of a binary classifier are Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves [**?**], which demonstrate the performance of the algorithm at different cutoff values of a predictor variable. Together, they show the False Positive Rate (FPR), the True Positive Rate (TPR) or recall, and the precision exhibited by the algorithm.

In the present problem, they can be used to show how good the *q-val* associated to a protein group is at separating the two proteomes. Thus, *E. coli* proteins are treated as positives, and those from *Homo sapiens* are treated as negatives. Moreover, a filter based on the log2FC can be applied to enrich the data in positives and facilitate the classification task. In order to compare the results of the here presented pipeline (Comp+Rob) with those achieved by other available tools, the analysis was repeated using the data processing implemented in MaxQuant [**?**] at different steps. The pipeline combining MaxQuant and MaxLFQ (MQ+LFQ) was fully carried out using the MaxQuant suite, whereas the combination of MaxQuant and MSqRob (MQ+Rob) used MaxQuant up to the fraction normalization step.

The result of this analysis is shown in figure 2.6. Remarkably, Comp+Rob achieved similar results to those resulting from MQ+LFQ. Nonetheless, the latter performed clearly better when the log2FC filter was applied, indicating that a combined fold change and q-value criteria is best for declaring proteins as differentially abundant. In all cases MQ+Rob was as good as MQ+LFQ, if not much better.

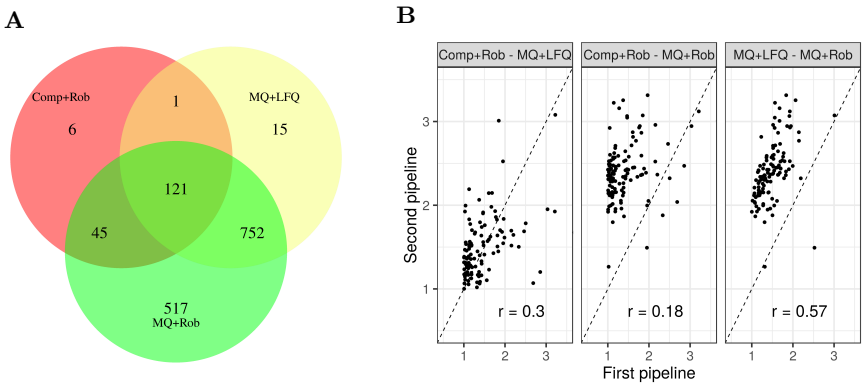


Figure 2.7 Agreement between the different pipelines. **A** The Venn diagram displays the number of *E. coli* proteins found to be differentially abundant by each pipeline. **B** The log2FC estimated in all three pipelines for the 121 shared proteins is plotted by pairs to evaluate their correlation.

In order to visualize the level of agreement of the pipelines for the assignment of differentially abundant proteins, a Venn diagram was constructed (see figure 2.7 A). Venn diagrams provide a straightforward visualization of the overlap of several ensembles of proteins. The scrutiny of the counts reveals a high degree of agreement, with most proteins detected by Comp+Rob or MQ+LFQ being detectd by MQ+Rob too. Remarkably, 517 proteins were detected by MQ+Rob alone, which further confirmed its resolving power. Analyzing the log2FC pairwise-correlation (see figure 2.7 B) reveals that the highest correlation was found between pipelines using MaxQuant as upstream spectra processor, as expected. Even though the observed correlation was low, the estimated log2FC was always greater than 1. The MQ+Rob pipeline systematically estimated log2FC values greater than the other 2.

Additionally, the accuracy with which each program assigns a log2FC estimate to each protein group, and how significantly different from the null value of 0 it was found to be, provides further insights on the pros and cons of each pipeline, as shown in figure 2.8. This analysis confirmed that the

pipeline providing the best separation between organisms was MQ+Rob.

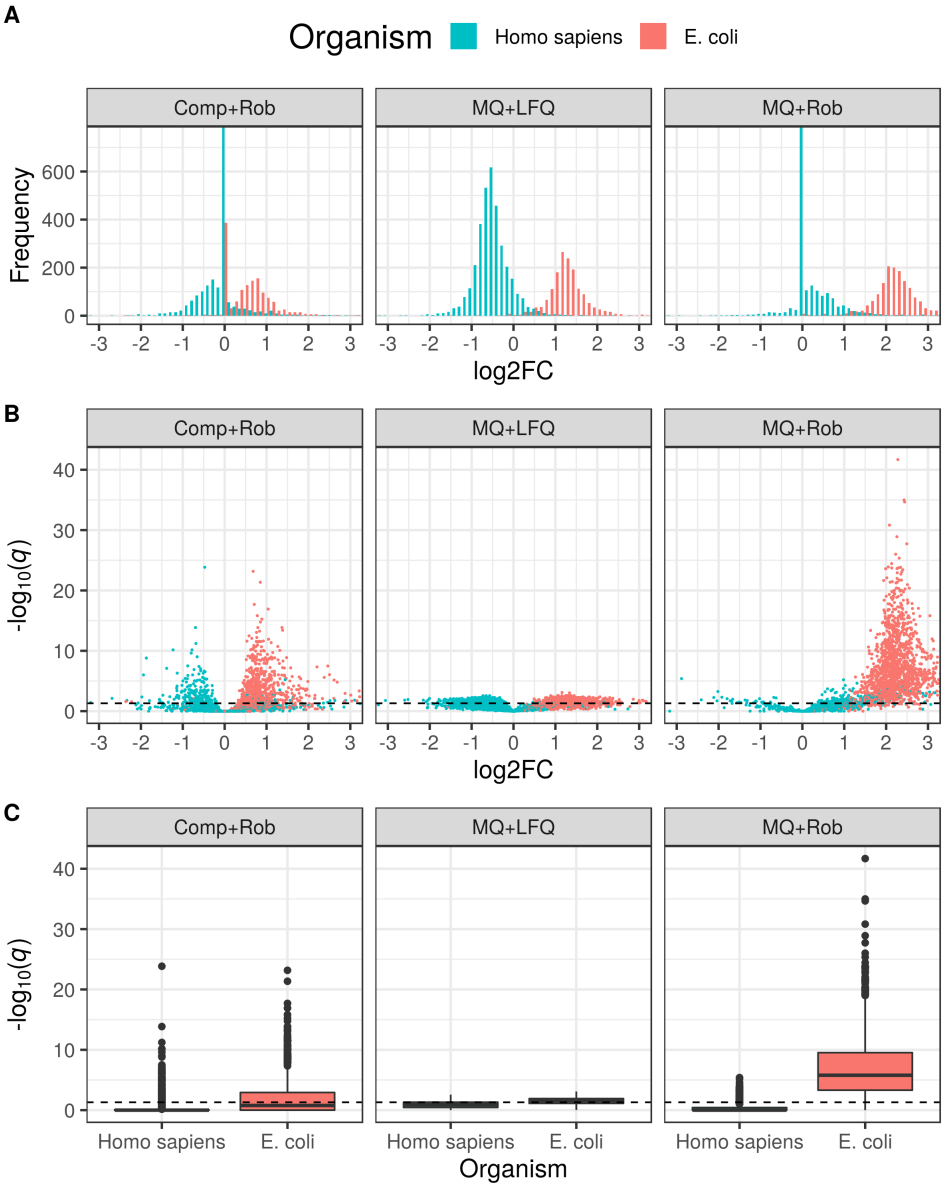


Figure 2.8 Overview over the different pipelines'performance. Quantification results are segregated by the source organism. **A** Histogram of the log2FC estimates. **B** Volcano plot, where every protein group is represented by a dot and the coordinates map to its log2FC on x and the minus logarithm of the q-value on y. **C** Boxplots of the minus logarithm of the q-value distribution.

2.4 Discussion

2.4.1 Improvement of the PSM step

The low attained matching rate manifests existing room for improvement in the currently available tools. As explained in section 1.4, it has been shown that the combined usage of multiple search engines can increase identifications, since the different statistical frameworks implemented in each of them compensate each other's caveats. Only MS-GF+ was used in thi work for simplicity. Furthermore, *de novo* search engines are available and supported by SearchGUI, and the latest developments in the field, like the IdentiPy engine [**?**] are progressively incorporated to the tool [5]. Their usage is predicted to improve identification rates and quality of the matches.

The most important reason why many spectra remain unidentified is the presence of post-translational modifications (PTMs), which exponentially increase the search space, forcing most workflows to discard many of the peptides featuring a PTM. New approaches to the problem are emerging, mainly machine learning methods for the handling of unexpected modifications [**?**]. Moreover, the prediction of MS2 peak intensities patterns from peptide sequences promises to increase the amount of evidence available during the PSM process, thus boosting correct identifications [**?**] [**?**].

Finally, the extremely low matching rates in the latter fractions translated to decreased contributions to the number of identifications. This indicated that decreasing the number of fractions would not have had a major impact in the experiment's depth.

[5] https://github.com/compomics/peptide-shaker/issues/309

2.4.2 Improvement of the feature extraction step

While the apex intensity is a good estimate of the peptide abundance, other methods are available that could potentially provide more robust data, such as the area of the peak. This is the approach taken by the recently published tool RawQuant [**?**], or the feature detection tool in the OpenMS pipeline [**?**]. Moreover, in order to fine tune the performance of the MBR and Apex intensity extraction steps, several input arguments can be adapted to each dataset. These are the spectrometer's m/z resolution, the time window used during the apex search and whether or not to activate outlier filtering and what weights to use.

2.4.3 Improvement of the quantification step

The deployed relative quantification approach successfully manages to estimate the log2FC of several of proteins in the dataset with high confidence. However, it was not found to be the best performing of the assayed pipelines in the benchmark dataset. Several axes of improvement are clear from the results.

Management of sample fractionation

Sample fractionation allows for a deeper analysis of the peptide mix, which in turn leads to increased identifications and an increased amount of data to be analyzed. However, it psimultaneously provides an extra random effect that difficults quantification if not handled properly.

The way the MaxLFQ engine solves the problem is by aggregating the data from the same sample across fractions using a ponderated average. One nor-

malizing weight is determined for every sample via the *Levenberg-Marquant minimisation of the overall proteome variation* $H(N)$ as defined in [**?**]. A custom implementation was written in Python using the scipy module, but the results were not satisfactory.

MSqRob treats the fractions as a random effect, as explained in section 2.3.4. As a consequence, the statistical model in MSqRob can be overwhelmed by the inconsistency resulting from a 24-plex sample fractionation, leading to null log2FC estimates for many *2*E. coli proteins. This is clearly manifested in figure 2.8 as the bins at log2FC of 0, which represent protein groups for which MSqRob did not find enough evidence to discard the null hypothesis. While no *E. coli* proteins fell in this category in the MQ-MSqRob pipeline, many did in the Comp+Rob, proving room for improvement in this aspect.

In any case, the way sample fractionation is handled should depend upon the fractionation method. For example, membrane fractionation can be exploited when working with membrane proteins [**?**].

To sum up, the presently implemented fraction handling is correct and provides good results, but fine-tuning of the pipeline at this step is expected to improve the results of the quantification. Thanks to the open-source nature of the software, this is indeed possible.

The pitfalls of mass spectrometry

The application of mass spectrometry to proteomics is a very hot topic in research, driving innovations in the way the different components of the spectrometer work. One recent example is the development of the timsTOF™ Pro (Trapped Ion Mobility Spectrometry-Time Of Flight) system by Bruker, which supports PASEF (Parallel Accumulation and SErial Fragmentation)

spectrometry [6]. This technology adds an extra dimension peptide separation process, leading to cleaner spectra. Notwithstanding, the developments also reveal the immature state of the technology. Slight inconsistencies in the protein extraction and digestion, bias in the peptide ionization, presence of unexpected PTMs, and detector saturation and insensitivity, remain as problems contributing to the eventual measurement variability characteristic of mass spectrometry [**?**]. Thus, adequate cleaning of the peptide dataset, and care to ignore outliers is preeminent. Indeed, the input file for the MQ+LFQ consisted of a preprocessed file which removed the most problematic proteins [**?**], which explains why more than 500 extra proteins could be quantified when using MSqRob (see figure 2.7 A).

Support for absolute quantification

While relative quantification like that performed by MSqRob is enough to infer differential abundance of proteins across conditions, absolute quantification would provide an estimate of protein quantities that would support many other analyses, and facilitate comparison between datasets, in a way similar to what the normalised read counts in FPKM (Fragments Per Kilobase per Million Reads) does in transcriptomics. On the other hand, MaxLFQ provides absolute estimates, albeit less robust.

Quantification of uncertainty

All the quantification approaches enunciated until now make use of different frequentist approaches to the problem, that evaluate how significantly dif-

[6] https://www.prnewswire.com/news-releases/bruker-launches-the-timstof-pro-mass-spectrometer-to-enable-the-revolutionary-pasef-method-for-next-generation-proteomics-300520791.html

ferent the data are from what would be expected under a null hypothesis of a null ratio across the pair of conditions. Yet, it does not provide probabilistic interpretations of the uncertainty behind the estimate. The development of a Bayesian framework which endeavors at solving this issue is presented in chapter 3.

2.5 Conclusion

An open-source, free, platform-agnostic and customisable label-free relative quantification pipeline assembling publicly available tools was presented in this work. The pipeline is capable of providing qualitative information on the protein composition of a sample using the latest proteomics search engines and validation software. Moreover, it can be extended to support more complex tasks, like *De novo* search, or the study of PTMs, thanks to its non-monolithic nature and usage of open-formats. Likewise, the openness of the data in all steps allows for quality control checks along the way. However, the comparison against other workflows shows the room for improvement in the quantification step when dealing with fractionated data, albeit deficiencies in this aspect should have no impact in non-fractionated datasets.

Either the pipeline developed in the presented chapter or those based on MaxQuant are ready to be deployed in a real environment and provide NZ with useful insights on its proteomics data for free.

Chapter 1

A label-free quantification proteomics pipeline

Summary

A pipeline making use of the set of tools published by the Compomics and StatOmics groups SearchGUI [**?**], PeptideShaker [**?**], moFF [**?**] and MSqRob [**?**] , was developed to support complete label-free protein quantification analyses using the most recent advances in the field with open-source software. The pipeline can be run on Linux computer clusters to perform (I) peptide to spectrum matching against a reference database, (II) quality control and filtering, (III) MBR and feature extraction, (IV) protein inference and (V) relative quantification. Its output can be passed to follow-up analyses in R or Python to get a biological interpretation of the results. A benchmark of its performance was accomplished using the proteome benchmark dataset published in [**?**]. The results exhibited an accuracy similar to those achieved by the MaxQuant [**?**] software, excepting a bias produced by the sample fractionation of this dataset.

Figure 2.1 Schema of the presented pipeline. First, .RAW files produced by the spectrometer are converted to an open format like the Mascot Generic Format (MGF). After that, spectra are searched by means of a search engine to perform the PSM step. Thereafter, PSMs are validated and the most likely set proteins is inferred. If quantative information is to be extracted, a quantification step attempting to estimate protein quantities is executed. The biological interpretation of the pipeline results can be achieved by interacting with public databases using R/Bioconductor or Python thanks to their open format.

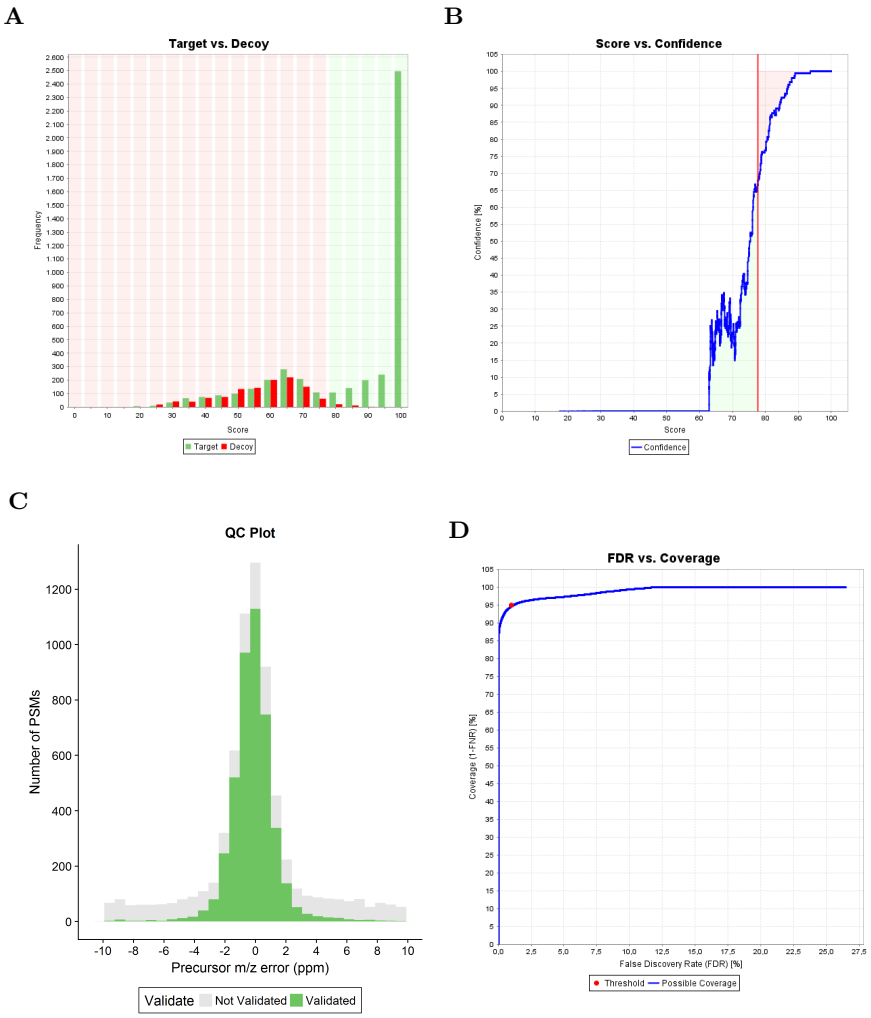


Figure 2.2 Quality control and validation of the 13th fraction of the third replicate in condition L. **A** Score distribution for matches to the decoy and the target databases. **B** Evolution of the PSM score with confidence. The implemented cutoff at FDR of 1 % is displayed with a red vertical line. **C** Distribution of the difference between the predicted and measured m/z values, segregated by validation status. **D** ROC curve built upon the number of false positives and negatives estimated from the decoy search. The cutoff is displayed as a red dot.

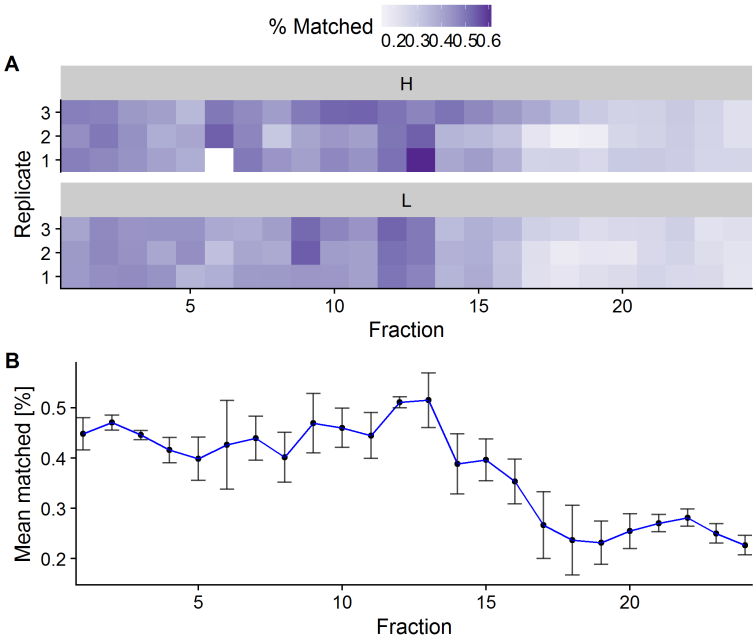


Figure 2.3 Percentage of matched spectra in all samples. The total number of spectra per sample ranged between 5894 and 20249. **A** Percentages are encoded with a blue palette, the darker, the higher, and viceversa. **B** The mean for each analysed fraction across conditions and replicates is displayed together with error bars to represent the standard deviation. The sixth fraction of the first replicate in condition H was missing in the PRIDE data repository.

Chapter 3

BayesQuant: Probabilistic estimation of protein ratios

Summary

The current label-free quantification methods, reviewed in section 1.7 all rely on frequentist statistics, which return point estimates of model parameters such as the estimate of the abundance ratio (fold change) across conditions. However, a Bayesian based approach to this problem, that we know of, is lacking in the literature. As a response to this shortcoming, a statistical model, named BayesQuant, was written in the probabilistic programming framework PyMC3 and tested on the same benchmark dataset from chapter 2). The execution of the three steps required when doing Bayesian modelling, mainly (I) model implementation, (II) computation of posterior probabilities and (III) model checking, will be described for this particular problem in the present chapter, together with a discussion on its usability and its strengths.

3.1 Introduction

3.1.1 Frequentist and Bayesian statistics

Both the LFQ and MSqRob quantitication engines presented in chapter 2 took a frequentist approach to the problem of protein quantification. Such approaches, establish a null hypothesis H_0, complementary to the actual hypothesis being tested (alternative hypothesis H_1). The information about the phenomena being modelled is put to use to measure how likely this null hypothesis is via Null-Hypothesis Significance Testing (NHST).

$$H_0 : \log_2(FC) = 0$$
$$H_1 : \log_2(FC) \neq 0$$

The results of this analysis will depend not only of the observed data, but also the data generation process [**?**]. This way, both tools returned point estimates of the log2FC between a pair of conditions and a p-value, which provides a measurement of the probability of the null hypothesis. The null hypothesis usually states that the true value of the parameter is 0. Thus, p-values do not say anything about the probability of the alternative (our) hypothesis.

The Bayesian statistical framework provides an alternative point of view by revolving the role played by the model parameters and the observed data. While frequentist statistics treats the data as random and the parameters as fixed, a Bayesian framework swaps their roles and yields a probability distribution for any model parameter given the provided data. Bayesian analyses

rely on the observed data and prior knowledge about the phenomenon only, and comes equipped with an equation that mathematically formalizes how to introduce these two dependencies in a valid way. This is the so called Bayes's theorem.

$$P(\theta|y) = \frac{P(y|\theta)P(\theta)}{P(y)} \tag{3.1}$$

The Bayes theorem is an extremely versatile tool taking a role analogous to that of statistics like T, F, or χ^2. Unlike them, which are tailored to specific scenarios, it can be used compute the probability distribution of almost any parameter in any model. 4 terms can be distinguished in its expression

1. $P(\theta)$: the **prior** probability distribution of the model parameter, introduces previous knowledge about the phenomenon being modelled.

2. $P(y|\theta)$: the probability distribution of the observed data (y) over the possible parameter space. It is also known as **likelihood** of the model, and its duty is updating our beliefs about the phenomenon by capturing the information in the data.

3. $P(y)$: the probability of the data, defined as the marginal probability of the data given a parameter value, for all possible values. It acts as a normalizing constant that makes the resulting distribution a true probability distribution adding up to 1. It is also known as the data **evidence**.

4. $P(\theta|y)$: the updated probability distribution of the model parameter, with the information extracted from the observed data. Since it reflects the beliefs about the phenomena after observing data, it is called **posterior** probability distribution,as opposite to the prior.

The Bayes's theorem formulated above is thus read as *the posterir probability of the parameter θ is equal to the **prior** probability distribution times the likelihood divided by a constant.*

It can be applied to the quantification problem, where θ turns into the log2FC parameter we try to estimate. All is needed is the computation of the posterior probability distribution. Unfortunately, this is tractable analytically for simple models only. However, Markov Chain-Monte Carlo (MCMC) and Variational Inference (VI) methods can be used to approximate this posterior. Both of which are now possible to run thanks to the advent of modern computing.

3.1.2 Inference methods: MCMC and VI

On the one hand, MCMC methods simulate sampling from the posterior distribution by constructing an ergodic Markov chain on θ whose stationary distribution is the posterior $p(\theta \mid y)$ [**?**]. Monte Carlo sampling is performed on the Markov Chain, (hence the name of the technique) to randomly explore the parameter space with some heuristics. This heuristic guarantees convergence of the empirical estimate with the true posterior, provided a big enough sample size. Sampling convergence is defined as the status reached by the sampler when it estimates a distribution of probability that does not change anymore, regardless of how much longer the sampler runs [**?**] [1]. The first sampling algorithms, like Metropolis-Hastings [**?**] [2] [3] have given way to the much more efficient sampler NUTS [**?**].

[1] http://www.cs.jhu.edu/ jason/tutorials/variational.html

[2] A very nice tutorial on how Metropolis-Hastings works http://twiecki.github.io/blog/2015/11/10/mcmc-sampling/

[3] Likewise, this resource provides very illustrative explanations of Hamiltonian-MC and NUTS http://elevanth.org/blog/2017/11/28/build-a-better-markov-chain/

On the other hand, VI methods, very recently developed and introduced in PyMC3, provide a fast alternative to MCMC methods, yet they are not guaranteed to asymptotically approximate the true posterior. In VI, optimization, instead of sampling, is employed as way to approximate the posterior [**?**]. More concretely, a family of densities Q over the parameters θ is defined. The goal of VI is to find the single density q within the family Q that best approximates $p(\theta|y)$. This approximate density should be complex enough to reach a good approximation, and at the same time simple enough to be computationally easy to work with. The best candidate is defined as the one minimising the Kullback-Leibler (KL) divergence (see equation 3.2).

$$q^*(\theta) = \operatorname{argmin} KL(q(\theta) \parallel p(\theta|x)) \tag{3.2}$$

where $q(\theta)$ stands for the whole family of densities, $p(\theta|x)$ stands for the true posterior and KL stands for the KL divergence. $q^*(\theta)$ is the best candidate density.

From [4] *the term variational is used because the best q in Q is picked. The term derives from the "calculus of variations," which deals with optimization problems that pick* [...] *a particular q in Q, specified by setting some variational parameters i.e the knobs on Q that need to be turned to get a good approximation.*

One of the terms yielded by the decomposition of the conditional density embodied by the posterior is the evidence of the data $p(x)$. This term is computationally intractable, thus making the computation of the exact

[4] http://www.cs.jhu.edu/ jason/tutorials/variational.html

KL divergence impossible. A lower bound up to an additive constant can however be computed thanks to Jensen's inequality [**?**] This approximate amount is the Variational or Evidence Lower BOund (ELBO). *The ELBO is the negative KL divergence of plus logp(x), which is a constant with respect to q(z). Thus maximizing the ELBO is equivalent to minimizing the KL divergence* [**?**].

Finally, several density approximation families are available. The mean-field family of densities is one of the most simple, as it ignores all dependencies between latent variables, and treats them as independent. From http://www.cs.jhu.edu/ jason/tutorials/variational.html the mean-field approximations works by *pretending that the variables are just behaving that way "on their own." The mean-field method throws away all of the interactions.* A generic mean-field family member is shown in equation 3.3.

$$q(\theta_1, ..., \theta_m) = \prod_{j=1}^{m} q(\theta_j) \tag{3.3}$$

The demanding computational cost of the MCMC and VI schemes has only been recently met by the power of modern computers, thus making the approach feasible. Several Bayesian statistics frameworks are available on popular programming languages like R (JAGS [**?**]) and Python (PyMC3 [**?**]), all making use of a programming paradigm called **probabilistic programming**.

3.1.3 Probabilistic programming

The management of uncertainty in statistics is done by means of probability distributions, which account for the possible values that a parameter could

take, together with how likely they are. Probabilistic programming offers the framework to build complex statistical models by storing probability distributions as variables of the program. Moreover, probabilistic programming packages supply the tools needed to perform inference with these models by taking experimental evidence (data) and fitting the distributions to the data using Bayes theorem. The fit of the model to the data entails the computation of the posterior probability distribution, which is tallied using MCMC algorithms provided with the probabilistic programming tool.

PyMC3 is a mature Python module dedicated to support probabilistic programming. It features intuitive model specification syntax, modern and powerful MCMC sampling algorithms like the No-U-Turn-Sampler (NUTS) as well as Automatic Differentiation Variational Inference (VI) [5]. Therefore, it provides the tools required to build a Bayesian model for the probabilisitc estimation of protein log2FC in proteomics datasets, that is, together with an estimate of its uncertainty.

3.1.4 Goals

1. Develop a Bayesian model to estimate log2FC in proteomics datasets together with the uncertainty of the estimate.

2. Make it scalable and fast for usability in ordinary modern computers.

[5] https://github.com/pymc-devs/pymc3

protein	Organism	H1	H2	H3	L1	L2	L3
P0A8I8	E. coli	25.13	25.24	24.39	21.71	22.67	22.40
P0A8I8	E. coli	21.49	23.10	23.38	21.34	22.65	21.25
P0A8I8	E. coli	24.10	24.54	23.81	19.88	20.87	20.30
A6NDG6	Homo sapiens	25.08	24.98	23.83	22.54	22.52	24.96
A6NDG6	Homo sapiens	22.70	24.32	23.00	22.29	23.27	23.91
A6NDG6	Homo sapiens	21.18	22.51	23.25	22.30	21.75	23.61

Table 3.1 Sample data input for the Bayesian model. Every row represents a unique peptide. The first column refers to the protein it was found to map to in the protein inference step. The second column is an annotation field, in this case indicating the protein's organism. The remaining columns indicate the $log_2(Intensity)$ registered for each peptide in the corresponding run. In this case, three peptides were observed for the proteins with ids *P0A8I8* and *A6NDG6*.

3.2 Materials and Methods

3.2.1 Data input

The peptides.txt and proteinGroups.txt files produced by MaxQuant [**?**] after the analysis of the dataset published in [**?**] were used as input for the modelling algorithm when running without sequence modelling, and processed using the `preprocess_MaxQuant` and `MSnSet2protdata` functions in the MSqRob [**?**] package, similar to what was done in section 2.2.2.

The final state of the data is shown in table 3.1 for a single protein.

3.2.2 Sequence feature extraction

When running with active sequence modelling, the peptide sequences and their neighborhood in the protein was extracted from *E. coli* and *Homo sapiens* proteome databases (2.2.2). The protein neighborhood was defined by a window spanning 15 aminoacids on both sides of the peptide. It was extracted using the seqinr [**?**] and GenomicRanges [**?**] packages implemented

in R and Bioconductor.

Features to model the peptide effect based on sequence were extracted using BLABLA

3.2.3 Hierarchical modelling

The MS1 intensity measurements are affected by two main sources of variation:

1. The fixed effect that the researcher aims at unraveling with proteomics (treatment effect)
2. Random noise produced by experimental procedures, sequence bias, etc, which lead to a loss of data quality and resolving power.

A mass spectrometry-proteomics workflow aims at providing estimates of the treatment effect, upon which the biological interpretation of the results is done, while minimising the remaining unwanted effects. The mathematical framework implemented to model these effects was similar to the one used by MSqRob:

$$y_{ijkl} = \beta^0_{ijkl} + \beta^{treatment}_{ij} + \beta^{peptide}_{ik} + \beta^{run}_{il} + \epsilon_i \tag{3.4}$$

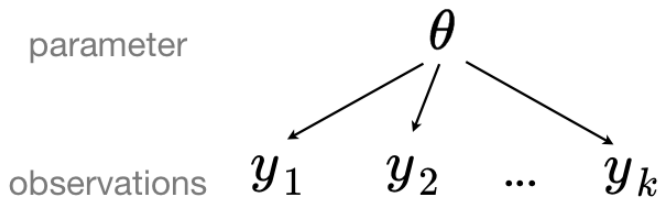
where y_{ijkl} stands for the $\log_2$-transformed intensity registered in treatment j, peptide k and run l. This way, separate treatment, peptide and run effects are distinguished. i stands for the protein index, which is evidently the same for all measurements belonging to peptides mapped to the same protein. ϵ models the noise that cannot be explained with the three effects aforementioned, and is the same for all measurements associated to the same

protein.

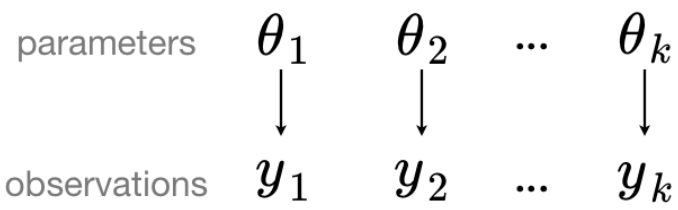
For example, the measurement $y_{P,A,p,1}$ corresponds to a measurement mapped to protein P. The measurement is impacted by a fixed effect, produced by treatment A, but also the peptide effect in peptide p and the run effect in run 1. An intercept term was also added to account for the starting intensity value.

The data is thus affected by different effects acting independently, and in order to model them properly, multilevels need to be defined. Multilevel effects can be modelled with three approaches: pooled, unpooled, and hierarchically (see figure 3.1). Hierarchical modeling is used when the sampling variance is not the only source of variation among the parameters, but they still share a dependency. The intercept and the three effects were modeled as independent hierarchies.

Pooled



Unpooled



Hierarchical

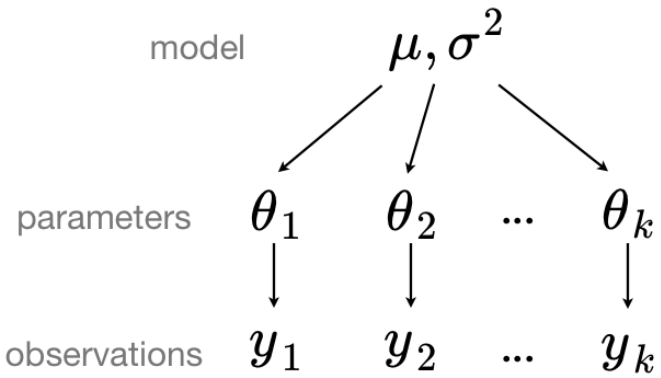


Figure 3.1 Alternative multilevel modelling approaches. A pooled model assumes that the parameter distribution is the same for all the datapoints. The unpooled model represents the opposite case, there every datapoint is assumed to be modelled by a different probability distribution of the parameter for each. Finally, a hierarchical model settles for a middle ground where the parameters will not be exactly the same but not completely different either. This is achieved by generating a global distribution from which the parameter for each datapoint is sampled [6].

This way, independent and identically distributed (IID) priors were set for

[6] https://docs.pymc.io/notebooks/multilevel_modeling.html

the intercept, treatment, peptide and run effects. The particular effect observed on each peptide was then modelled as a value sampled from the corresponding IID prior. A diagram of the resulting model is displayed in figure 3.2).

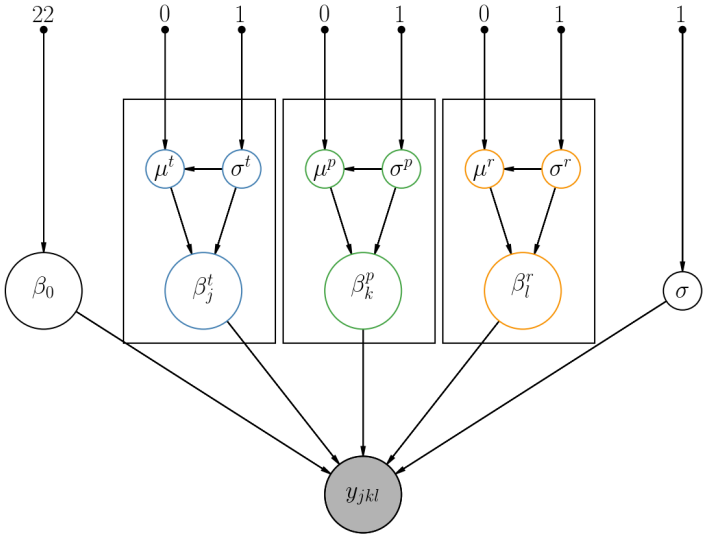


Figure 3.2 Diagram of the Bayesian model developed in the present work, without sequence modelling of the peptide effect. It is represented as a Directed Acyclic Graph (DAG) where nodes represent hyperparameters or random variables (in a circle) and directed edges represent the dependencies between them. The nodes on the top depict the hyperparameters of the model, governing the prior probability distributions represented by the nodes to which their edges lead to. The remaining nodes articulate the probabilistic model and all lead to a final node (y_{ijkl}) which represents the modelling of the observed data with the instantiated model. The model hierarchies are organized in boxes. The one corresponding to the treatment effect can be read as *the treatment effect β_{ij}^t observed in the data is modelled as a probability distribution conditional on the value of μ^t and σ^t, which are in turn probability distributions governed by the hyperparameters 0 and 1.*

3.2.4 Prior probability distribution specification

Both because of the little knowledge on the value of the parameters governing the data generation process and to provide a prior that everyone can

agree upon, non informative priors were provided to the model. This was condensed in the specification of Normal and Half Normal distributions for all the random stochastic variables in the model. The hyperparameters were selected to be as non informative as possible.

3.2.5 Posterior probability distribution computation

Both the MCMC NUTS sampler and VI mean-field strategies were applied to approximate the posterior probability distribution of the log2FC.

3.2.6 Model checking

Posterior predictive checks of the posterior distribution were carried out using the `sample_ppc()` function in PyMC3. A visual evaluation of the differences between simulated and observed data was used to assert the correctness of the models.

3.2.7 PyMC3 implementation

In order to get started with PyMC3, we first need to import it.

```
1 import pymc3 as pm
```

The model is initialized as a Python context manager. Within the context manager, the model is implemented by defining prior distributions for the model parameters and establishing the dependency relationships between them.

The code below formalizes in PyMC3 code the bias in MS1 intensity measurements due to a random effect.

```
with pm.Model() as model:
        sigma = pm.HalfNormal('sigma', 1)
        mu = pm.Normal('mu', mu=0, sd=sigma)
```

Which is equivalent to the following statistical notation:

$$\sigma \sim \mathcal{HN}(1)\,. \tag{3.5}$$

$$\mu \sim \mathcal{N}(0,\, \sigma)\,. \tag{3.6}$$

and can be read as *the prior probability distribution of the random effect follows a normal distribution with mean 0 and standard deviation σ. In turn, σ's prior probability distribution follows a half normal distribution with standard deviation 1.* 1 and 0 act as hyperparameters of the model, and introduce the state of beliefs or knowledge on the system prior to seeing the data.

The hierarchical structure of the model is set by the definition of a parameter distribution from which the value for each element (peptide) being modelled is sampled. This can be done by defining a new normal distribution where its μ and σ are set to the random variables defined above. Im PyMC3 code,

```
        betae = pm.Normal("betae", mu, sigma, n_elements)
```

which is equivalent to:

$$\beta_i^e \sim \mathcal{N}(\mu, \sigma)\,. \tag{3.7}$$

and can be read as *the probability distribution of the bias observed in the i^{th} element due to the effect here modelled is said to follow a normal distribution with mean μ and standard deviation σ both defined as random probability distributions above.*

Finally, the equation 3.4 defined above closes the model and binds the data to the model parameters. Its PyMC3 implementation is the following:

```
6         epsilon = pm.HalfNormal('epsilon', 1)
7         m = pm.Deterministic("m", beta)
8         obs = pm.Normal("obs", m, epsilon, observed=y)
```

which is equivalent to

$$\begin{aligned} \epsilon &\sim \mathcal{HN}(1)\,. \\ m &= \sum_{i=1} \beta_i\,. \\ y &\sim \mathcal{N}(m, \epsilon) \end{aligned} \tag{3.8}$$

and is read as *the observed data is modelled by a normal distribution with mean m and standard deviation ϵ. m is a random deterministic variable resulting from the sum the effects defined above. ϵ follows a new half normal distribution with standard deviation 1.* A deterministic variable is a random variable that acquires a fixed value if all random variables it has a

dependency on take a fixed value too, i.e its stochasticity disappears if its parameters are fixed.

3.3 Results

3.3.1 Running BayesQuant

BayesQuant takes a peptide_summary_intensity_moFF_run.tab (moFF) or a peptides.txt file (MaxQuant) as input, containing a peptide per row. Preprocessing using the MSqRob `preprocess_MSnSet()` function, as well as other data munging functions, is required.

```
Rscript prepare_BayesQuant.R --pepf peptides.txt \
    --filetype MaxQuant --exp exp_annotation.tsv \
    --output .
```

The R script outputs the input file for BayesQuant. Its data is introduced in BayesQuant via the following code:

```
# Load the module
from BayesQuant import BayesQuant
bayesquant = BayesQuant()
# Read the dataset
bayesquant.read_data(data="ms1_intensities.tsv")
# Compile a model for proteins with 3 peptides
model = bayesquant.compile_model(n_peptides=3)
# Load a specific protein
```

```
9 bayesquant.load_data("A6NDG61")
```

The snippet above loads the program, reads in the data, selects the data for a specific protein, and builds a model that matches the number of peptides observed. The posterior distribution can be computed using pure MCMC or VI methods, as stated in section 3.2.5. The VI procedure will now be explained, due to its significantly better performance.

```
10 # VI approximation: much faster and pretty accurate
11 trace_advi = bayesquant.fit(model_name=p, n_draws=40000)
```

The `bayesquant.fit()` function carries out two tasks:

- Finds the best mean-field approximation $q(\theta)$ to the true posterior $p(\theta|x)$. This approximate distribution is easier to work with.
- Sample from the approximate posterior, returning a collection of samples called trace.

The values stored in the trace obtained via VI will follow an approximation to the true posterior, and contain the results of the modelling process. Once it'obtained, the inference is complete and model checking ought to be performed to validate the results as explained in section 3.3.5. It is run with

```
12 # VI approximation: much faster and pretty accurate
13 bayesquant.pcc()
```

3.3.2 Variational inference optimisation evaluation

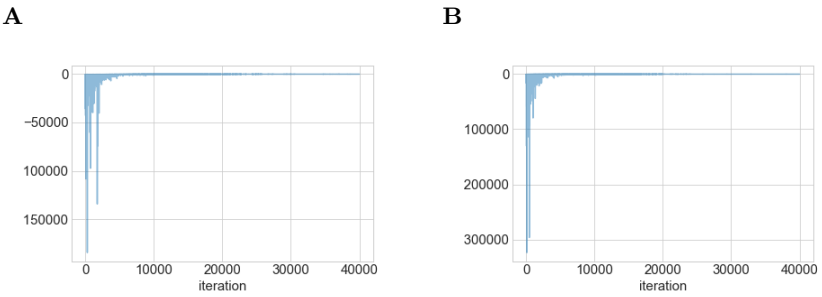


Figure 3.3 Progression of the ELBO of the approximation for proteins P0A818 (*E. coli*, **A**) and A6NDG6 (*Homo sapiens*, **B**) over 40k iterations.

3.3.3 Basic model

The results of the model fitting step are shown in figures **??**, and **??** for one protein from *E. coli* and *Homo sapiens*, respectively.

3.3.4 Extended model: peptide effect with sequence features

3.3.5 Model checking

However the posterior probability is computed, a proper Bayesian analysis is not finished without a model checking step where

- In MCMC methods, evidence for non-convergence must be collected. Albeit lack of evidence of non-convergence does not guarantee convergence, the existence of evidence definitely proofs it [**?**].
- Posterior Predictive Checks are carried out to measure how well the model captures the data.

MCMC asymptotically approaches the posterior after convergence, however, this condition is sometimes fulfilled by running long chains. Too short

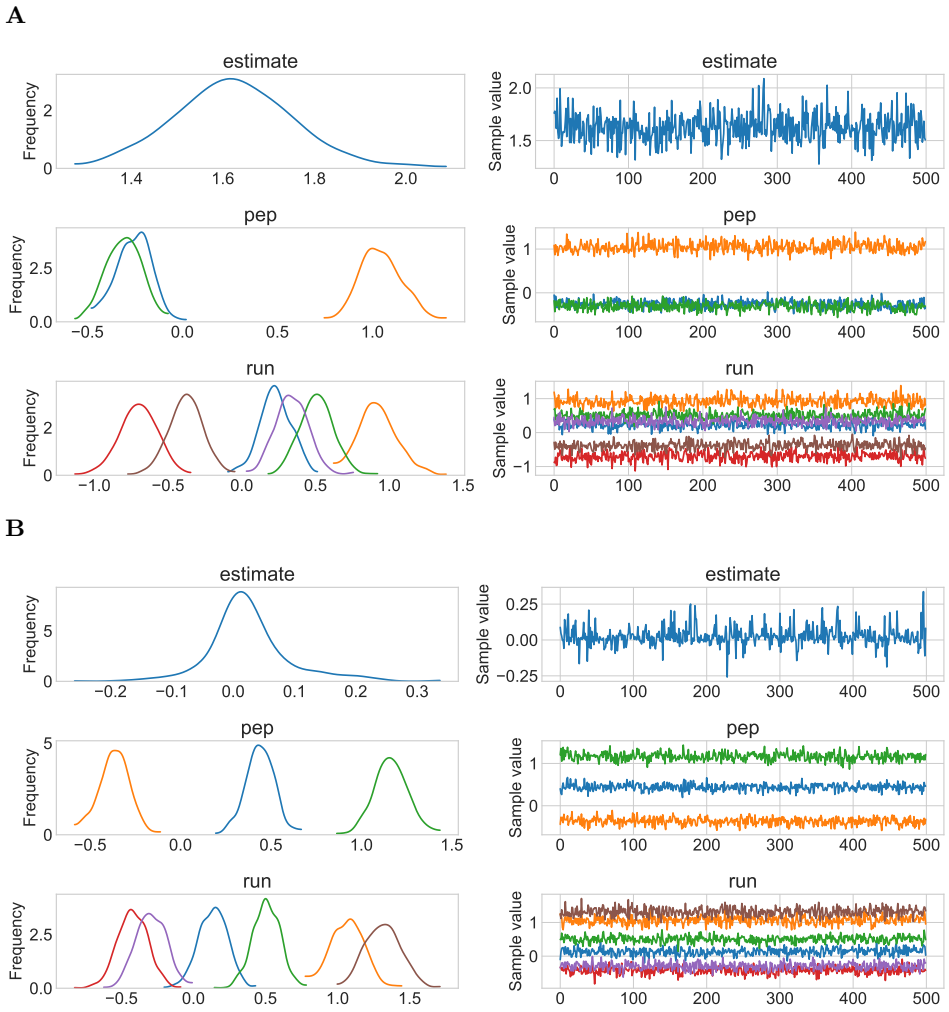
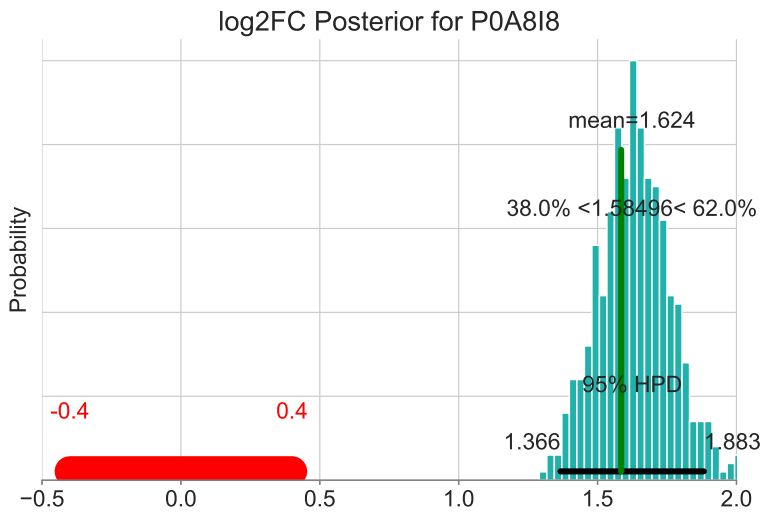


Figure 3.4 Traceplot of the model fit for proteins P0A818 (*E. coli*, **A**) and A6NDG6 (*Homo sapiens*, **B**). The left panel shows the frequency density over the parameter space for several model parameters: (I) the log2FC estimate (difference of treatment effects), (II) the effects in the three observed peptides, and (III) the effects in the six available runs. The right panel displays the sampled values from the VI approximation stored in the trace.

a chain, reflected by the lack of convergence, will fail to approximate the posterior and return wrong results. Most importantly, neither reaching convergence nor the best VI approximation guarantee that the fitted model actually captures the data generation process. Therefore, the true posterior could be a bad reflection of the natural process being model and the best

A



B

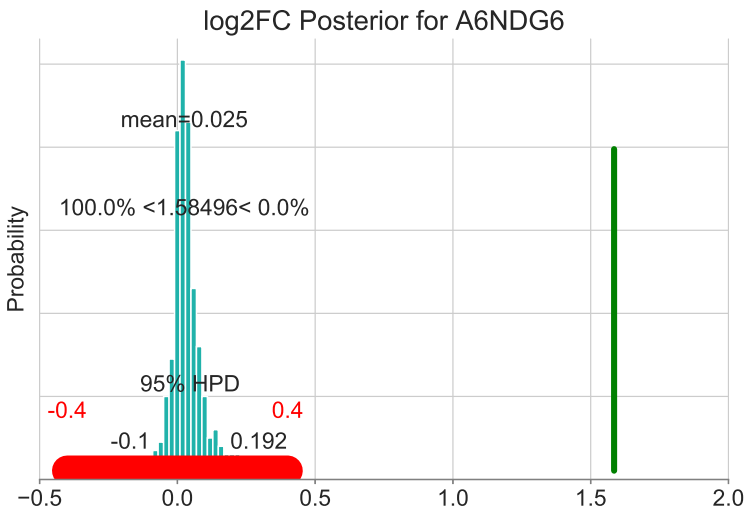


Figure 3.5 Annotated posterior probability distribution for the log2FC estimate for proteins P0A818 (*E. coli*, **A**) and A6NDG6 (*Homo sapiens*, **B**). The bar height is mapped to the probability mass in the corresponding interval. The 95% High Probability Density Interval (HPDI) is marked with a black line. The Region Of Practical Equivalence (ROPE), shown in red, is a small range of parameter values that are considered to be practically equivalent to the null value in the present application [**?**], in this case a 0.4 window around 0. A green vertical line marks the expected log2FC estimate for *E. coli* proteins ($\log_2(3)$).

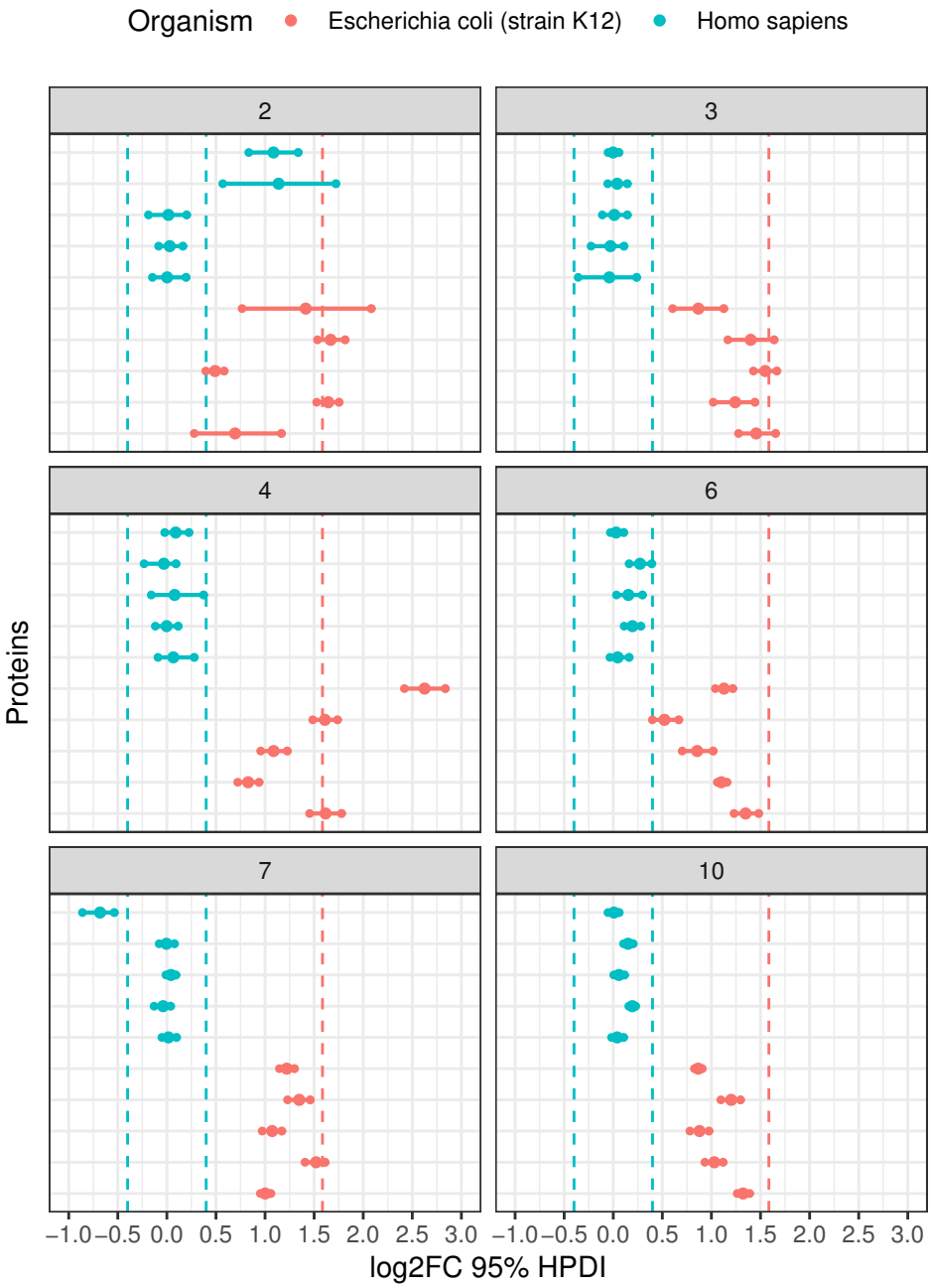


Figure 3.6 Visualization of the 95% HPDI inferred from 5 bacterial and human proteins with 2,3,4,7,6 and 10 peptides. The intervals are represented by horizontal lines. The dot represents the mean of the whole distribution. Vertical blue lines represent the ROPE defined as in figure ??, whereas the red line represents the expected estimate for bacterial proteins.

approximation will not solve it.

3.4 Discussion

3.5 Conclusion

Chapter 4

Pipeline benchmarking on NZ data

Summary

A benchmark dataset generated by NZ technicians was run through the pipeline presented in chapter 2 to showcase its performance. The experiment consisted of the application of two different treatments to THP-1 cell cultures. One group was subjected to an experimental procedure triggering the immunological response, while the other group was subjected to a negative control. The analysis of the resulting dataset through the aforementioned pipeline should thus reflect a change in the protein profile corresponding to an activation of the immune system in the first group when compared to the second. The results confirmed that the computational analysis successfully captured this response. It was concluded that the software can be used in future experiments where the expected biological phenomena is not known.

Conclusion

Appendix

```python
import pymc3 as pm
import numpy as np
import matplotlib.pyplot as plt
from theano import shared
import shutil
import os

def protein_model(observed_sh, feats_sh, x_treat_sh, x_pep_sh, x_run_sh, x_estimate_sh,
    n_peptides, model_name, n_draws=1000, n_chains=3,
    hierarchical_center=False, remove_backend=True, sequence=False):

    # Check working environment
    if not os.path.isdir("traces") or not os.path.isdir("plots/traceplots"):
        msg = "Please create a traces dir and a plots/traceplots dir before running this code"
        raise Exception(msg)

    if remove_backend and os.path.isdir(model_name):
        shutil.rmtree(model_name)

    # The number of proteins in this model is always one
    # i.e this model is fitted protein-wise
    n_prots = 1
    # The number of features is set to 9 for now
    # All peptides have 9 features, stored in feats_shared
    n_features = 9


    with pm.Model() as model:

        # Build a hyerarchical linear model of
        # log2(MS1 intensities) by accounting for:

            # Peptide effect
            # Run (batch) effect
            # Treatment effect
            # Remaining random effects

        # The difference in treatment effects is an estimate of the log2FC


        # Set a prior on the intercept
        intercept = pm.Normal("intercept", 22, 1)

        # Set a prior on the remaining random effects
        sigma = pm.HalfNormal('sigma', 1)

        ## Set priors on the peptide effect
        ##################################################
        sigma_pep = pm.HalfNormal('sigma_pep', 1)
```

```
50
51          # Not using the sequence
52          if not sequence:
53              mu_pep = pm.Normal('mu_pep', mu=0, sd=sigma_pep, shape=(n_peptides, 1))
54
55          # Using the peptide sequence
56          else:
57              # sequence based modelling
58              mu_theta = pm.Normal('theta_generic', 0, sigma_pep, shape = 1)
59              theta = pm.Normal('theta', mu_theta, sigma_pep, shape = (n_features, 1))    # 9x1
60              theta_inter = pm.Normal('theta_inter', mu_theta, sigma_pep, shape = 1)
61              mu_pep = pm.Deterministic("mu_pep", theta_inter + feats_sh.dot(theta)) # n_peptidesx1
62
63
64          ## Set priors on the treatment and run effects
65          ################################################
66          sigma_treat = pm.HalfNormal('sigma_treat', 1)
67          mu_treat = pm.Normal('mu_treat', 0, sigma_treat)
68          sigma_run = pm.HalfNormal('sigma_run', 1)
69          mu_run = pm.Normal('mu_run', 0, sigma_run)
70
71          # Standard implementation of the hyerarchies
72          if hierarchical_center:
73              pep = pm.Normal("pep", mu_pep, sigma_pep) # n_peptidesx1
74              treat = pm.Normal('treat', mu_treat, sigma_treat, shape = (n_prots*2, 1))
75              run = pm.Normal('run', mu_run, sigma_run, shape = (n_prots*6, 1))
76
77          # Reparametrization to escape funnel of hell as noted in
78          # http://twiecki.github.io/blog/2017/02/08/bayesian-hierchical-non-centered/
79          else:
80              pep_offset = pm.Normal("pep_offset", mu=0, sd=1, shape = (n_peptides, 1))
81              pep = pm.Deterministic("pep", mu_pep + pep_offset * sigma_pep)
82              treat_offset = pm.Normal("treat_offset", mu=0, sd=1, shape=(n_prots*2, 1))
83              treat = pm.Deterministic("treat", mu_treat + treat_offset*sigma_treat)
84              run_offset = pm.Normal("run_offset", mu=0, sd=1, shape=(n_prots*6, 1))
85              run = pm.Deterministic("run", mu_run + run_offset*sigma_run)
86
87
88          # Model the effect for all peptides
89          # The sh variables consist of -1,0,1 matrices telling pymc3
90          # which parameters shall be used with each peptide
91          # In practice, the "clone" each parameter to fit the shape of observed_sh
92          # observed_sh is a n_peptides*6x1 tensor
93          # The first 6 numbers store the MS1 intensities of the first peptide in the 6 runs
94          # The next 6 those of the second peptide, and so on
95
96          estimate = pm.Deterministic('estimate', pm.math.sum(x_estimate_sh.dot(treat), axis=1))
97          treatment_effect = pm.Deterministic("treatment_effect", pm.math.sum(x_treat_sh.dot(treat), axis=
98          peptide_effect = pm.Deterministic("peptide_effect", pm.math.sum(x_pep_sh.dot(pep), axis=1))
99          run_effect = pm.Deterministic("run_effect", pm.math.sum(x_run_sh.dot(run), axis=1))
```

```
100
101         # BIND MODEL TO DATA
102         mu = pm.Deterministic("mu",
103             intercept + treatment_effect + peptide_effect + run_effect) #n_peptides*6x1
104         if hierarchical_center:
105             obs = pm.Normal("obs", mu, sigma, observed=observed_sh)
106         else:
107             obs_offset = pm.Normal("obs_offset", mu=0, sd=1, shape=(n_peptides*6,1))
108             obs = pm.Normal("obs", mu+obs_offset*sigma, sigma, observed=observed_sh)
109
110
111     print("Success: Model {} compiled".format(model_name))
112
113     with model:
114         # Parameters of the simulation:
115         # Number of iterations and independent chains.
116         n_sim = n_draws*n_chains
117
118         # Save traces to the Text backend i.e a folder called
119         # model_name containing csv files for each chain
120         trace_name = 'traces/{}'.format(model_name)
121         db = pm.backends.Text(trace_name)
122         trace = pm.sample(draws=n_draws, njobs=n_chains, trace=db,
123                           tune=2000, nuts_kwargs=dict(target_accept=.95))
124
125     # Save a traceplot
126     pm.traceplot(trace, varnames=["estimate"])
127     traceplot = "plots/traceplots/{}.png".format(model_name)
128     plt.savefig(traceplot)
129     plt.close()
130
131     return model
```