Protein Level Probabilities for Shotgun Proteomics Experiments

J O S É F E R N A N D E Z N A V A R R O

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2013

Protein Level Probabilities for Shotgun Proteomics Experiments

J O S É F E R N A N D E Z N A V A R R O

DD221X, Master's Thesis in Computer Science (30 ECTS credits)
Master Programme in Computational and Systems Biology 120 credits
Royal Institute of Technology year 2013
Supervisor at CSC was Lukas Käll
Examiner was Jens Lagergren

TRITA-CSC-E 2013:018
ISRN-KTH/CSC/E--13/018--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.kth.se/csc

Abstract

There exist many techniques to perform the analysis of proteins. Current Shotgun Proteomics based methods assign peptide level scores to peptide-spectrum matches obtained by matching observed spectra against a database of theoretically generated spectra from a set of known proteins. This set of peptide-spectrum matches is ranked according to a score based on the quality of the match. Subsequently, the candidate present proteins can be inferred from the confidently identified peptides. This process seems straightforward and out of the box, however, it has some notable weaknesses. The imperfections of the set of scores given by the database search engine tool lead to the need to apply a post-processing tool that can give more accurate scores yielding more precise information of the number of peptides believed to be present in the sample. However, the set of proteins believed to be present in the analyzed sample should not be inferred directly from the set of high scored peptides. For example, in cases where there are peptides that form part of more than one protein, or when there is a protein that contain both high and low scored peptides, or when there is a high scored peptide that indicates that the protein is present but it is a false positive, etc. Therefore, there is a need for a tool to infer the list of confident proteins from the set of scored peptides and assign confidence scores to the inferred proteins accounting for the problems described. We present in this thesis work a software tool for the analysis of proteins. This tool is based on the integration of the protein inference tool Fido with the post-processor Percolator as a combined PSM post-processing and protein inference package that efficiently estimates protein level probabilities and confidence scores. We show that our integration of Fido and Percolator surpasses the state-of-the-art protein inference tool Protein prophet in terms of calibration, sensitivity, specificity and number of proteins correctly identified.

Referat

Integration och implementering av proteinsannolikhetsberäkningar i Percolator för proteomikstudier

Proteomik har på senare tid blivit ett mycket viktigt område inom molekylär cellbiologi. Det kan beskrivas som läran om proteiner, hur de modifieras, när och var de uttrycks, hur de är involverade i de metaboliska vägarna samt hur de interagerar med varandra. Det finns många tekniker för analys av proteiner, men ″Shotgun Proteomics ″ är den mest etablerade och anses vara det primära verktyget att studera protiners primärstruktur i stor skala. Nuvarande ″Tandem Mass Spectrometry ″ baserade metoder tilldelar peptidnivåpoäng till peptidspektrumträffar vilka erhålls genom sökning av spektra mot en databas av teoretiskt genererade spectra från kända proteiner. Detta set av peptidspektrumträffar kommer att rangordnas baserat på poäng vilka baseras på kvaliteten på matchningen. Proteinkandidater kan härledas ur ett subset av proteiner identifierade med hög konfidensgrad. Denna process verkar vara simpel, men har dock påvisat uppenbara svagheter vilka vi kommer att beskriva i kommande stycken. Bristerna i poänguppsättningen givna av databasens sökmotor leder till behovet av att applicera ett verktyg för efterbehandling vilket kan ge en mer korrekt poänguppsättning, vilket i sin tur ger mer korrekt information om antalet peptider vilka förväntas finnas i provet. Huvudtanken bakom efterbehandlingsverktyg är att kombinera olika poäng och funktioner givna av sökmotorn för att kunna uppskatta bättre poängrankning av peptider efter deras sannolikhet att befinna sig i provet. Uppsättningen av proteiner vilka tros existera i provet ska dock inte härledas direkt ut ett sett med proteiner med höga poäng. Till exempel, i fall där flera pepetider bildar en del av fler än ett protein, eller när ett protein innehåller peptider med både höga och låga poäng, eller när det finns en peptid med högt poängtal vilket indikerar att proteinet är närvarande, men är egentligen ett falskt positiv, osv. Ett efterbehandlingsverktyg på proteinnivå behövs därför för att tilldela konfidenspoäng till en lista av proteiner vilka tros existera i provet och står för problemet beskrivet ovan.

Acknowledgments

I would like to thank my supervisor Lukas Käll, without whose patience and dedication this thesis work would not have been possible. I would also like to thank my colleagues Viktor Granholm and Luminita Moruz whose help and advices have been very important for this thesis work. I would also like to thank Matrix Science for supporting this thesis work. Finally, I would like to thank my dear friend Vicky Brady and my colleague Magnus Rosenlund for proof reading and correcting this thesis work.

Contents

Chapter 1

Introduction

Proteomics is an important area in the field of molecular cell biology. It can be defined as the study of proteins, how they are modified, when and where they are expressed, how they are involved in metabolic pathways and how they interact with one another. Shotgun Proteomics is widely established and regarded as the primary tool to study the protein content of biological mixtures with a high throughput. One frequent goal of a shotgun proteomics experiment is to obtain a list of high confident proteins, where a high confident protein means a protein that is likely to be present in the analyzed sample. In order to do this, proteins have to be statistically ranked according to a confidence score. Tandem mass spectrometry based proteomics tools[23] assign peptide level statistic scores to peptide-spectrum matches (PSMs) obtained by searching a set of observed spectra against a database of known protein sequences. This set of scored peptide-spectrum matches is then re-ranked by using a so called post-processing tool which will estimate more calibrated and accurate peptide level scores by combining different scores and information obtained from the database search engine tool results. The main idea behind post-processing tools is to combine different scores and features given by the search engine tool in order to estimate a better score. This score will rank the peptides according to how likely they are to be in the sample. State-of-the-art post-processing tools such as Percolator[9] or Peptide prophet[11] show a considerable improvement in the number and the accuracy of the PSMs correctly identified[1], they also give a better distinction among correct and incorrect PSMs by estimating useful statistically relevant information such as p values, q values[25], posterior probabilities[8] and false discovery rates[24]. These sets of statistically relevant measurements allow us to classify and rank the peptides according to the probability of being present in the analyzed sample. They can also be used to deduce thresholds where the expected number of false positives is known or to give relevant information about the confidence of a single PSM. They can also be used to measure the performance, calibration and quality of the results given by the post-processor. However, what we really need is a list of proteins and their respective probabilities, in order to be

[1] A peptide with a very high confidence to be present in the sample.

able to distinguish the high confident proteins from the other. This set of candidate high confident proteins should not be inferred directly from the set of high scored peptides for reasons such as the peptide degeneracy problem as well as many others reasons that we will describe in further sections. Therefore, it is recommendable to apply a protein inference tool to compute protein level probabilities from a set of scored PSMs. The protein inference tool will give us a list of scored proteins ranked according to their probability of being present in the sample, it will also give us statistically relevant information about the quality and sensitivity of the results and it will address efficiently the problems associated with the protein inference. State-of-the-art protein inference tools such as Protein prophet[14] group PSMs into proteins and then calculate a confidence score for each identification based on a combination of the unique[2] peptide level probabilities for a certain protein. Such methods tend to penalize the probabilities of proteins corresponding to single hits[3] and reward those corresponding to multiple hits. These assumptions might lead to cases where proteins containing several peptides with low scores have a better score than proteins containing fewer peptides with high scores. This problem may get worse when using large datasets or in the cases of there being many shared peptides. An alternative to Protein prophet is the recently published software called Fido[19]. Fido estimates protein level probabilities accurately by using a novel Bayesian graph-theoretic approach which addresses the problems described before as well as the problem with the shared peptides[4]. The authors of Fido show that the number of correctly identified proteins given by Fido equal and surpass in some cases Protein prophet. Furthermore, the problems described display a knowledge of the importance of having a tool to infer a set of confident proteins from a set of scored PSMs and the problems associated with it. Given this, and the good performance shown by Fido, are among the main reasons that have motivated this thesis work.

[2]Many PSMs might match the same peptide, usually in order to estimate protein probabilities the PSM with the highest probability will be kept.

[3] A protein that has been matched by one single unique PSM.

[4] Same PSM is part of more than one protein.

Part I

Shotgun Proteomics

Chapter 2

Theory and Background

2.1 Tandem Mass Spectrometry

Mass spectrometry based proteomics methods were developed during the early 90s[3]. They quickly became popular and displaced 2d PAGE and Edman degradation based methods which were the most commonly used methods for protein identification previously. Mass spectrometry based methods provided a considerable increase in sensitivity and decrease in peptide fragmentation time. In the last decade, some important technological advances have established mass spectrometry or more precisely tandem mass spectrometry based methods as the primary tool for the study of the protein content of biological samples.

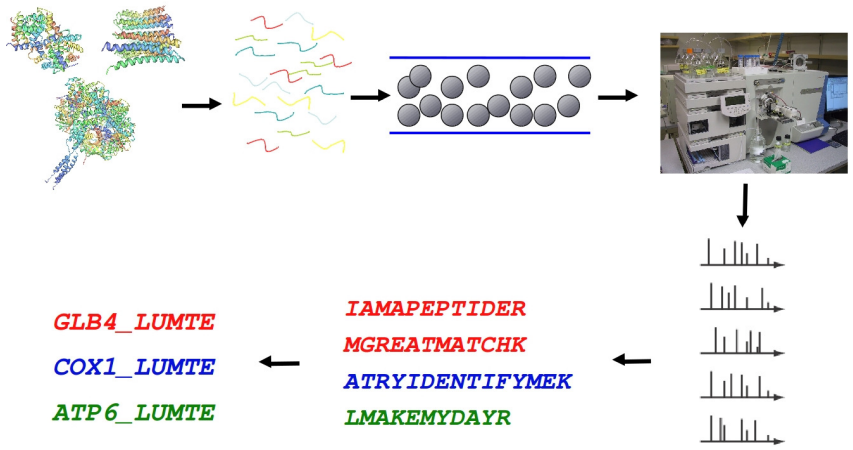

Figure 2.1. Shotgun proteomics workflow. This figure illustrates the main steps of a shotgun proteomics experiment. From the original sample being analyzed to the candidate proteins believed to be present in the sample. Figure from [13].

The typical workflow of a tandem mass spectrometry based experiment as it can be seen in figure 2.1 can be described in a few steps. First of all, we need to extract and isolate the proteins that we want to study in the analyzed sample. This process

is called **sample preparation** and it basically consists of the purification of the biological sample subject to be studied. Once the source and its proteins have been isolated and purified, the proteins have to be digested into peptides. A peptide is a chain of amino acids compounds linked by peptide bonds whose length (number of amino acids) can be very different. The digestion is typically done by using a sequence specific enzyme that will cleave off the protein sequence into peptides at certain amino acids. This process is called **protein digestion** as we can see in the second step in figure 2.2.

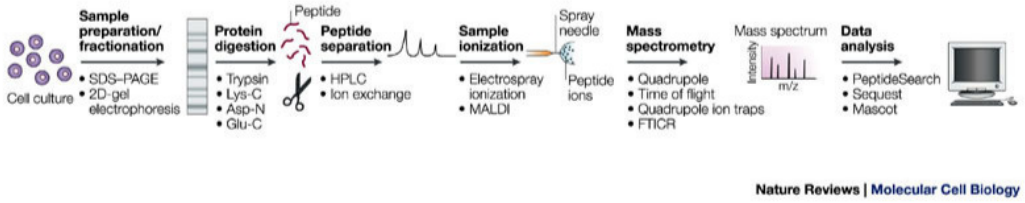

Figure 2.2. The mass-spectrometry/proteomics experiment. This figure gives a more detailed explanation of the steps of a shotgun proteomics experiment. Starting off by the sample preparation and finishing with the data analysis. Figure from [23].

Usually the most common enzyme used for the protein digestion is trypsin which cleaves proteins on the carboxyl-terminal side of the arginine and lysine residues. The main reason to use trypsin is its high specificity and stability. Peptides obtained by digestion using trypsin are highly rich in information and easily interpretable by the mass spectrometer due to this specificity. Once a list of peptides by protein digestion have been obtained they need to be separated in a process called **liquid chromatography**, or peptide separation as it is referred to in some literature. This separation is done according to the propensity of the peptides to sticle to the stationary phase of the chromatographic column. This is done in order to introduce them separately in the mass spectrometer, generating a time separation window called **retention time**. Once the peptides have been eluted they need to be ionized in order for them to be trapped in the mass spectrometer analyzer, this process is called **peptide ionization**. The most common technique to ionize the peptides is called *electro-spray ionization* which overcomes the propensity of the molecules to fragment when ionized. Once the peptides have been ionized, the mass spectrometer traps the ions by electric or magnetic fields. Now that the eluted peptides have been ionized and trapped, the peptide ions are thus ready to be analyzed in the mass spectrometer in order to determine the mass-charge ratio of the peptides. Once the mass-charge ratios and the intensity peaks have been estimated, more stages of separation are added to the process and this produces what is called ″tandem mass spectrometry″. In a tandem mass spectrometry based experiment, selected peptide ions observed in the previous step (MS) are fragmented into smaller pieces obtaining a new list of mass-charge values and intensities for all the new fragmented ions as we can see in the figure 2.3. The fragmentation pattern

encoded by the MS2 (where 2 stands for two stages of separation) or tandem mass spectrometer, allows the identification of the amino acid sequence of the peptide that produced it originally. Having more stages of separation gives more accuracy and precision. Once the peptide ions have been fragmented and detected in the mass spectrometer, we obtain a so called spectra. The sequence of the peptides can then be estimated from the observed spectra, this process is called **peptide identification**.

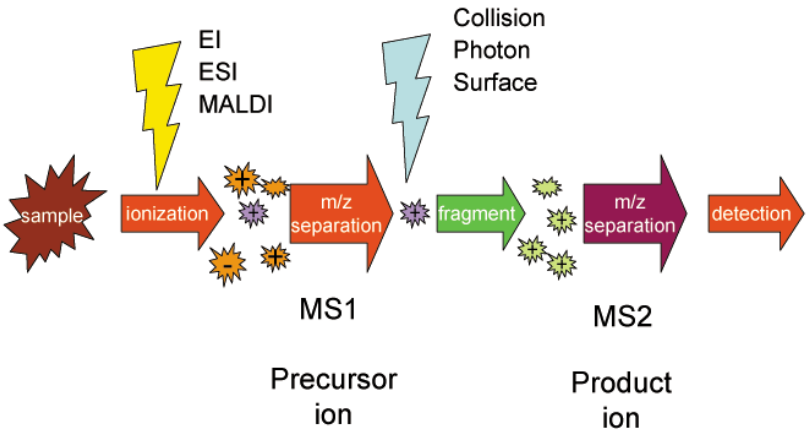

Figure 2.3. Schematics of a tandem mass spectrometry experiment. Figure from Wikipedia.

There are two main approaches to perform the peptide identification, starting with the fragmentation of the spectra. We can match the spectra against a database of theoretically generated spectra as described in figure 2.4 or we can generate the peptides from the original spectra in a process called *de novo* sequencing. The first approach is the most commonly used and it is the more accurate. There exist many tools that allow us to perform the database search, Sequest, Mascot, Crux, X!Tandem, OMSSA. However, even though the scoring system and the algorithms they use are different, they all share a similar functionality. They take as input the spectra obtained by a tandem mass spectrometry based experiment and compare them with theoretically generated spectra obtained in silico from a database of known proteins. For this comparison, several different criteria are applied to finally output a list of candidate peptides for each spectrum, ranked according to a particular score which varies from one search engine to another, but generally accounts for the similarity between the experimental and the theoretically generated spectra.

In our experiments, we ran the searches using Crux which is an open source implementation of Sequest available at (http://noble.gs.washington.edu/proj/crux/). Basically, what Sequest does is for each spectrum the top 500 candidate peptides are selected through a fast calculated preliminary score Sp. Then a theoretical spectrum is calculated for each of these top candidates and a normalized cross correlation score, Xcorr, is calculated. Another score deltaCn, is also provided which

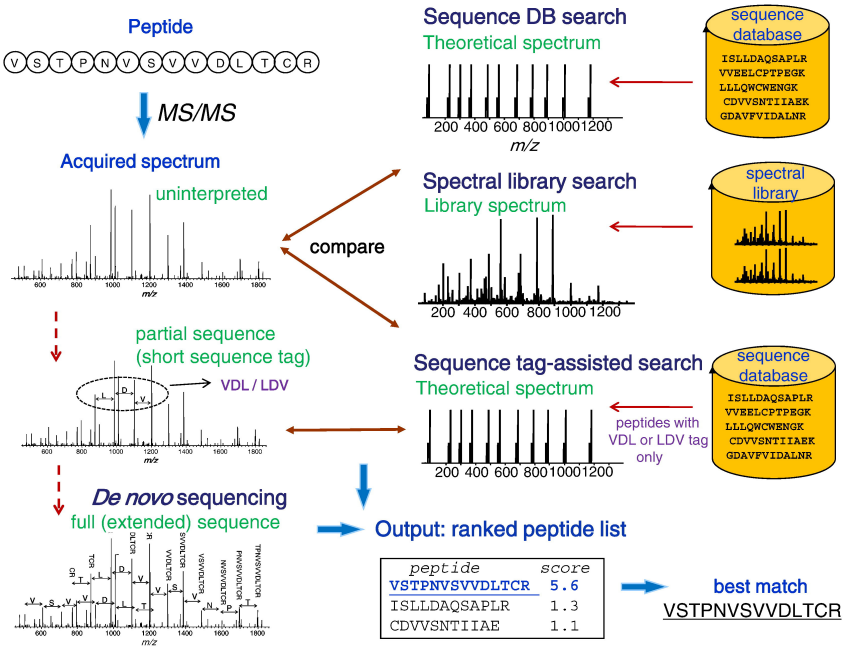

Figure 2.4. In silico database search workflow. The theoretically generated spectrum is compared to the acquired spectrum. If they match, a quality score is calculated and the spectrum with the highest scores is kept. Figure from Wikipedia.

gives the relative difference between the first and second best Xcorr.
The problem with the database search tools is that even though they find a match for every spectrum, what we call peptide-spectrum match, in reality only a small amount of those peptide-spectrum-matches (PSMs) contain peptides that are actually present in the original sample[9]. Also, they provide different scores but only the correlation score is used to rank the PSMs. Additionally, the current scoring methods of the database search tools have many weak points as described in [9], [11] and [15] such as the overlapping between scores of incorrect and correct peptide identifications, unknown error rates (false positives and false negatives), unknown sensitivity as well as a high rate of false negatives. This creates a need to generate a new set of statistical confidence scores and error rates for the PSMs in order to get a better classification of the correct and incorrect PSMs and this is where the post-processing tools come into the equation.

2.2 Post-processing tools

In this section we will describe the main idea behind the functionality of the post-processing tools. We will mainly focus on the two state-of-the-art and most commonly used tools for such purpose, which are Percolator and the so called ″Trans-Proteomic Pipeline″ which includes Peptide prophet and iProphet for estimating peptide level probabilities and Protein prophet for estimating protein level probabilities. Finally, we will describe the functionality and characteristics of Fido[19] as the protein level post processing tool that has been integrated into Percolator in this thesis work.
Using the scores received from the database search engine tools, in a tandem mass spectrometry based experiment, is not the best method by which to rank the peptides from incorrect to correct based on their level of confidence of being present in the analyzed sample, as described in the previous chapter. For this reason, recent years have witnessed a proliferation of the post-processing tools. In general, the idea is to jointly model a new confidence score using different sources of information and scores obtained from the database search engine tool in order to generate a new set of scores with a much higher significance. This new set of scores will give more relevant information of whether the peptide is believed to be present in the sample or not giving a better classification among correct and incorrect PSMs. Therefore, the information given by this new score will be better and more accurate than the original confidence score given by the database search engine tool. The post-processing tool will also give statistically relevant information as described in the introduction. To perform this estimation, the post-processing tools make use of statistical modeling and machine learning techniques which might differ among different post-processing tools.

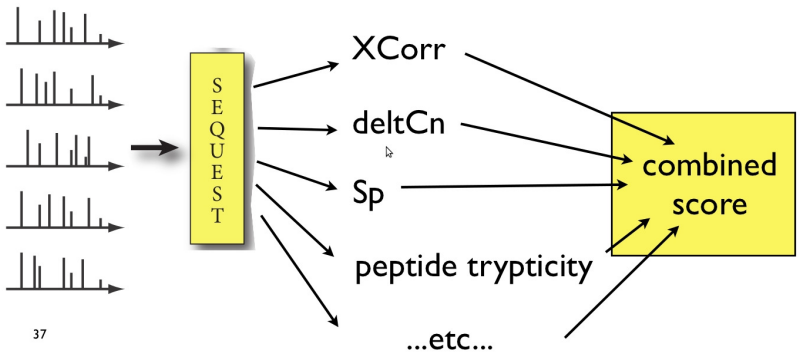

Figure 2.5. The main idea behind the post-processing tool Percolator, to combine different scores from the search engine tool to obtain a better score.

Current tools to estimate peptide level probabilities are well calibrated and tested so their level of accuracy and sensitivity are very high. However, as mentioned in the previous chapter, the set of candidate proteins should not be inferred directly from the set of scored peptides. Estimating protein level probabilities is not

as simple as estimating peptide level probabilities. We have to take into account that the errors that we carry on from the PSMs will increase at protein level, the peptide degeneracy problem[1], and last but not least the problems associated with the estimation of false discovery rates at protein level.

[1] Peptide degeneracy occurs when a peptide is shared by more than one protein.

2.2.1 Percolator

Percolator[9] uses a semi supervised machine learning algorithm to re-rank the PSMs by computing posterior error probabilities and false discovery rates as well as a new confidence score for each PSM. Percolator distinguishes between high confident and low confident PSMs by using a set of 20 features obtained from the database search engine tool. Those features will define the set of weights that Percolator uses to train the classifier. Those weights are learned by using a so called self training classification algorithm based on a support vector machine described in [1]. The trained SVM[1] allows Percolator to discriminate and rank PSMs efficiently. This concept of adaptative dynamic training allows Percolator to eliminate the need to construct a manually generated global set of weights by using the data set being post-processed instead. It also avoids overfitting to a particular type of spectra by applying a 3-fold cross validation[12] algorithm that allows Percolator to train and validate in different portions of the dataset composed of PSMs as shown in figure 2.6.

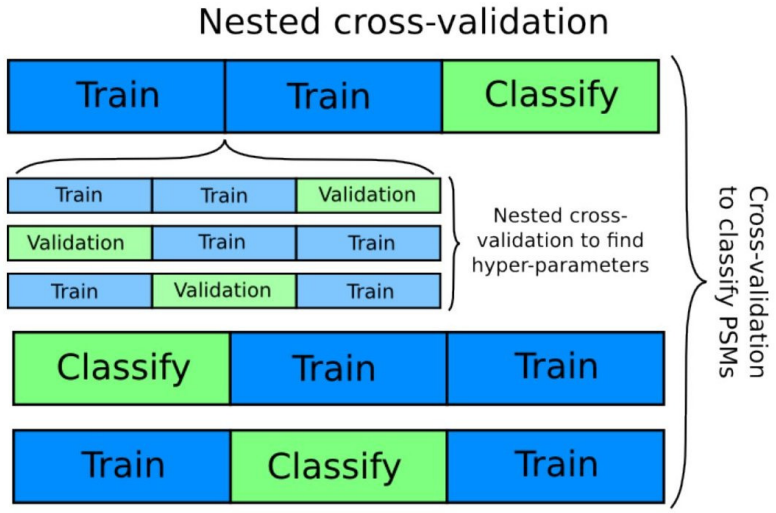

Figure 2.6. The cross validation of Percolator. The original set of PSMs is divided into three subsets, for each iteration a different subset will be used to classify while the other two will be used to train the model. There is also an internal cross validation done in every iteration to estimate the most optimal values for the hyper parameters of the model.

In order to obtain a negative control set to perform the classification, what is known as the null model, Percolator uses a target-decoy approach described in [7]. The target-decoy approach uses the original database of known proteins that are used to match the spectra. The difference is that the sequence of every protein in

[1] Support vector machine.

the database will be transformed[2] in order to obtain a database of incorrect proteins or more precisely a database of proteins that are guaranteed not to be present in the sample that is being analyzed. Once we have generated our decoy database of protein sequences, we need to run our database search engine tool against the target[3] database and the decoy database obtaining two sets of PSMs, target PSMs and decoy PSMs, as we can see in the figure 2.7. Percolator uses the target PSMs as positive examples for the classifier and the decoy PSMs as negative examples for the classifier. It also uses the set of decoy PSMs to estimate the statistically relevant information such as the FDR[4], the p values, the posterior error probabilities and the q values. This estimation is done assuming that the decoy PSMs model the null hypothesis. In the next sub sections we will describe the main significance scores given by Percolator which are all based on the target-decoy model.

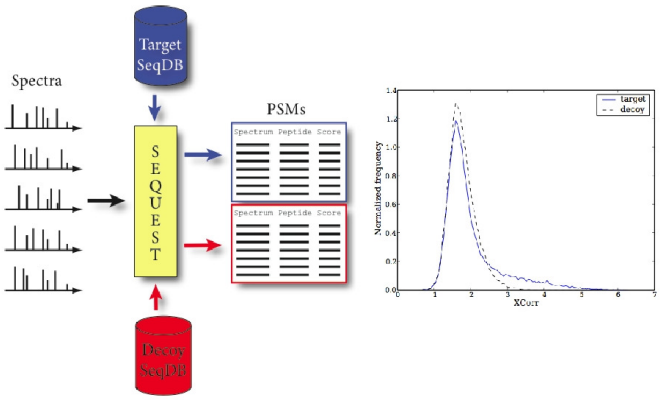

Figure 2.7. Work flow of the separate target decoy model. In this figure we can see how we obtain a different set of target PSMs and decoy PSMs by running the search engine tool against a target database and a decoy database. The graph shows the distributions of target and decoy PSMs scores.

[2] In most of the cases the sequence will be reversed, or shuffled, or randomly redistributed using a Markov chains model.

[3] The original not modified database.

[4] False discovery rate.

p value

The p value is probably the most commonly used significance measure in statistics. The p value can be roughly defined as the probability of obtaining a test statistic at least as extreme as the one that was actually observed, assuming that the null hypothesis is true. In other words the probability that an incorrect identification gets a score higher than or as high as the observed score. Therefore, a low p value means that the probability that a particular observation would occur by chance is small. A better description of how the p value is used for quality assessments is given in [5].

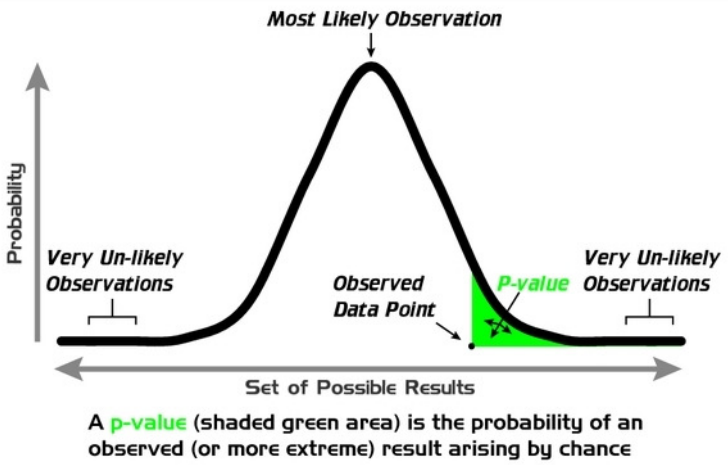

Figure 2.8. Graphical definition of the p value for a normal distribution. Figure from Wikipedia

Percolator estimates an empirical p value for each PSM. For this, it assumes that the decoy PSMs model the null hypothesis. Therefore, when the set of target PSMs and decoys PSMs are both ranked by a significance score, the estimation of the p values is straight-forward :

$$\widehat{pvalue}(x) = \frac{\mid \{y \mid y \geq x, y \in D\} \mid +1}{\mid \{y \mid y \in D\} \mid +1} \tag{2.1}$$

D stands for decoy PSMs. For a given target PSM with a score x, its p value would be the fraction of decoy PSMs that receive a score of x or higher.

False Discovery Rate

The false discovery rate (FDR) is a statistical measurement used in multiple hypothesis testing to correct for multiple comparisons. Given a list of rejected hypotheses, the FDR measures the expected proportion of incorrectly rejected null hypotheses. In practical terms, the FDR is the expected proportion of false positives among all significant hypotheses; for example, if 1000 observations were experimentally predicted to be different, and a maximum FDR for these observations was 0.10, then 100 of these observations would be expected to be false positives. In our case, the FDR associated with a particular score threshold is defined as the expected fraction of PSMs above that threshold that are believed to be incorrect. In short, the fraction of decoy PSMs from all the PSMs that are relevant (above a certain threshold). The FDR is a good statistical significance score because it gives an estimation of the expected number of incorrect PSMs at a certain score.
A simple way of estimating the FDR for a given score threshold (x) :

$$\widehat{FDR}(x) = \frac{\mid \{y \mid y \geq x, y \in D\} \mid}{\mid \{y \mid y \geq x, y \in T\} \mid} \tag{2.2}$$

Where D stands for the set of decoy PSMs and T stands for the set of target PSMs. If for example, at a score threshold of 3.0 we observe 3849 target PSMs and 219 decoy PSMs, the FDR would thus be 5.7%. We can see in the figure 2.9 the area that corresponds to the FDR and that the computation of the FDR is straightforward. However, we have to take into consideration that to estimate the FDR we assume that false positive PSMs are equally distributed among the dataset, also we assume that a false positive PSM is equally likely to have been matched against a decoy or a target protein entry in the database. Furthermore, in order to follow these assumptions we have to account for the fact that whereas all decoy PSMs are incorrect, not all the target PSMs are correct. Therefore, we should adjust the number of target PSMs taking into consideration the expected number of incorrect PSMs included in the target PSMs. We call this ratio π_o. Therefore, the formula to estimate the FDR taking into consideration the ratio of incorrect target PSMs is :

$$\widehat{FDR}_{\pi_o}(x) = \pi_o \times \frac{\mid \{y \mid y \geq x, y \in D\} \mid}{\mid \{y \mid y \geq x, y \in T\} \mid} \tag{2.3}$$

Where we can estimate π_o as (assuming that x is a very low score close to zero):

$$\widehat{\pi_o} = \frac{|\{y \mid y \leq x, y \in T\}|}{|\{y \mid y \leq x, y \in D\}|} \tag{2.4}$$

The FDR provides useful statistical information, although as shown in [7] the FDR has the property that it is not a function of the underlying score. Therefore, two different scores can lead to the same FDR, making it difficult to apply a FDR threshold to a dataset. Also, it is not linear, for instance its relevance applies only to the whole set. Besides, in practice we want to have a particular statistical measure for each PSM or what we would call a local FDR or q value as described in the next section.

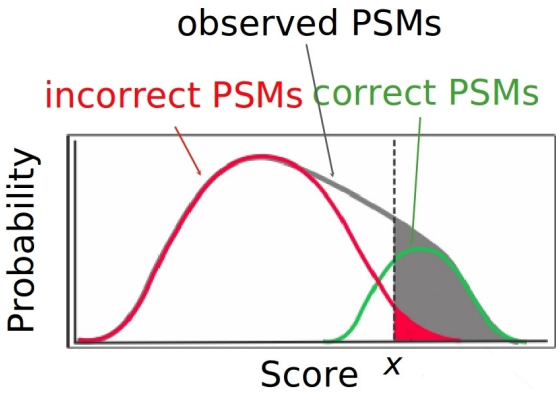

Figure 2.9. False Discovery Rate. As we can see in this figure, at a score threshold x the FDR would be the amount of incorrect PSMs (red) divided by the respective amount of target PSMs above the x threshold (gray) at the same threshold.

q value

The q value assigns error estimations to individual identifications rather than to sets as the FDR does. It is a very useful measurement to account for false positives and it is locally estimated, therefore each PSM has its own q value. What we want with the q values is to improve and adjust the information given by the p values by using an FDR approach. The q value can be defined as the minimum FDR threshold where PSMs (at a certain threshold) are believed to be correct, therefore present in the analyzed sample. For example, in a set of 1000 PSMs ranked by a score, we observe that the PSM ranked to position 100 has a p value of 0.01 and a q value of 0.0141. Therefore, with a p value of 0.01, we would expect around 10 false positive PSMs ($1000 \times 0.01 = 10$). On the other hand, the q value is a bit bigger (0.0141), which means that we would expect 1.41% of all the PSMs with a q value less than this to be false positives. This situation is much better. We know that 100 PSMs have a q value less than 0.0141, thus we would expect ($100 \times 0.0141 = 1.41$) false positive PSMs. In short, false positives according to p values take all 1000 PSMs into account when estimating how many false positives we should expect to see at a certain score threshold, whereas q values take into account only the PSMs with q values less than the threshold we choose. Of course, it is not always the case that q values will result in less false positives, but what we can say is that the q values give a more accurate account of the expected number of false positives for a given score threshold. Therefore, the q values stand as the best way to account for false positives and establish a confident score threshold for our PSMs. In our particular case the q value is defined as the minimal FDR threshold at which a given PSM is accepted.

$$\widehat{qvalue}(x') = \min_{x \leq x'}(\widehat{FDR}(x)) \tag{2.5}$$

Here x' stands for the score threshold and x stands for the set of scored PSMs. This formula estimates what we call ″empirical q values″ since they are estimated using the target-decoy model. In equation 2.5, we estimate the empirical q values using the standard FDR method without including πo. Percolator estimates the empirical q values at PSM and peptide level including π_o (FDR_{π_o}). This procedure works well at PSM and peptide level due to the equal distributions of false positive targets and decoys PSMs. However, these assumptions do not hold at protein level, where decoy and target false positive proteins are not equally distributed. This is because false positive target PSMs will match both correct and incorrect proteins whereas false positive decoy PSMs will match only the subset of incorrect proteins. This also implies that the distributions of false positive target and decoy proteins will be different. The problems mentioned give rise to the necessity for a different

approach to estimate the empirical q values at protein level rather than using the standard target-decoy approach.
There is another procedure to estimate q values using the posterior error probabilities. We call these q values ″estimated q values″. To compute the estimated q value as described in [8], for a given threshold x^t we simply need to sum up the posterior error probabilities of the PSMs with score above the score threshold x^t, and then divide the resulting sum by the total number of PSMs up to the threshold. This is described in the following formula :

$$\widehat{qvalue}_{PEP}(x^t) = \min_{x' \geq x^t} \frac{\sum_{x \in \{y | y \geq x', y \in T\}} P(H_o \mid X = x)}{\mid \{y \mid y \geq x^t, y \in T\} \mid} \tag{2.6}$$

Where x^t is the given score threshold, P(H_o|X=x) is the PEP and T is the set of target PSMs.
Theoretically, the empirical and estimated q-values should follow a linear correlation with similar values. The comparison between these two different methods of estimation of the q values is a good way of measuring the calibration and accuracy of the results given by our model.

Posterior Error Probabilities

The posterior error probability is roughly defined as the probability that an identification with score x is incorrect[1]. The statistical measurements described previously are very useful to rank the PSMs and set thresholds to differentiate significant from not significant PSMs and also to have an approximate estimation of the false positives among the significant. However, what we want is an individual confidence score that can tell us how likely it is for a PSM to be present or not in the sample. This probability can be very useful when we want to look at a single PSM, for example if we are interested in a PSM that has a low q value, we would like to know the probability of that PSM being incorrect, because in some cases low ranked PSMs might have a high PEP, thus we cannot assume that all PSMs with a low q value are high confident. Therefore, the PEP is a very useful and important statistical measure. The estimation of the PEPs is not trivial. Whereas the estimated q value is simply the sum of the PEPs of significant PSMs divided by the number of significant PSMs, the reverse is not true. The PEP is basically the derivative of the estimated number of false identification. Peptide prophet[11] was one of the pioneers in assigning posterior probabilities to PSMs using an unsupervised[2] scheme to fit parametric[3] gaussian distributions to the observed PSM score distribution; one distribution for the incorrect PSMs and another distribution for the correct PSMs. Consequently the posterior probability can be computed by Bayesian inference. Percolator uses a non-parametric[4] logistic regression method [8] to efficiently estimate posterior error probabilities. Basically, for a target PSM with score x what we need to estimate is the probability of the null hypothesis being true P(Ho|X=x), which is equivalent to the PEP or the probability of being incorrect. Thus we can use Bayes rule to estimate it as described in the following formula :

$$P(H_o|X = x) = \frac{P(H_o)P(X = x|H_o)}{P(X = x)} \tag{2.7}$$

Furthermore, assuming that P(H_o) can be interpreted as the fraction of incorrect target PSMs (remember π_o described in the previous section) and denoting

[1] Other post processing tools give the posterior probability, i.e. the probability of the identification being correct, which by definition is 1 - $P(H_o \mid X)$.

[2] Refers to the problem of trying to find hidden structure in unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution.

[3] Newer versions of Peptide prophet allow the inclusion of decoy PSMs changing the model from parametric to semi-parametric.

[4] A model is non-parametric when it does not make initial assumptions about the parameters of the distributions. They are actually defined by the data.

P(X=x|H_o) = f_0(x) and P(X=x) = f(x).
What we need to compute is :

$$P(H_o|X = x) = \pi_o \frac{f_o(x)}{f(x)} \tag{2.8}$$

However, there is no clear way to estimate $\frac{f_o(x)}{f(x)}$ since we do not know the incorrect PSMs among the target PSMs. Therefore, Percolator reformulates $\frac{f_o(x)}{f(x)}$ as : $\frac{p(x)}{1-p(x)}$, where p(x) is the probability of selecting a decoy PSM with score x from the set of target and decoy PSMs. To solve this Percolator uses a non-parametric logistic regression algorithm described in [17] which models the distribution of the ratio of decoys using cubic splines. It also uses a smoothing parameter estimated by cross validation. Once the ratio of decoys have been fitted and smoothed, the PEPs of the PSMs can be computed with the formula described above. A detailed explanation of the algorithm is given in [11].

2.2.2 Trans-Proteomic Pipeline

The Trans-Proteomic Pipeline (TPP) is a widely-used freely available open source proteomics data analysis pipeline developed at the Institute for Systems Biology (ISB) in Seattle. The TPP includes Peptide prophet, iProphet and Protein prophet as well as many other useful tools for protein analysis. Their post-processing tools are probably the most widely used in the field. The tools included in the TPP for post-processing and protein inference perform well at both peptide and protein level, establishing them as state-of-the-art tools for post-processing protein analysis for tandem mass spectrometry based experiments. Therefore, we have compared our integration of Fido and Percolator for the estimation of protein level probabilities to the results generated by TPP or more precisely Protein prophet on the same datasets.
We now proceed to describe the main post-processing tools of the TPP that are used in tandem mass spectrometry based experiments.

Peptide Prophet

Peptide prophet is a shotgun proteomics based post-processing tool that like Percolator uses machine learning techniques to classify the PSMs using different scores and attributes extracted from the database search engine tool. The main idea behind Peptide prophet is to distinguish between correct and incorrect PSMs by employing an Expectation Maximization algorithm[2] to fit two parametric distributions among all the PSMs, one distribution for the correct PSMs and another one for the incorrect PSMs. In order to fit the distributions Peptide prophet generates a discriminant score that is obtained combining different scores from the data base search engine tool as well as some additional features such as the number of tryptic termini[1]. Peptide prophet uses Bayesian classification to compute the posterior probability of the PSMs being correct. Peptide prophet uses Sequest as the preferred data base search engine tool, although it is able to process spectra obtained from most of the different database search engine tools that are available. Peptide prophet uses a parametric (unsupervised) model when it is executed with only target PSMs and a semi-parametric (semi-supervised) model when it is executed with target and decoy PSMs. The results we obtained using the parametric model were good in terms of calibration and sensitivity, although Percolator performed better. However, the results we obtained using the semi-parametric model showed a clear tendency to overfit and a poor classification expressed in overestimated probabilities with a high content of false positives. A possible explanation for this is that Peptide prophet uses the whole data set for training and validation, when it runs the expectation maximization algorithm to fit the gaussian distributions to find the set of parameters that gives the maximum likelihood, this procedure increases the risk of overfitting[2] to a particular type of spectra, especially with the models Peptide prophet uses in semi-supervised mode. Also, the fewer number of features that Peptide prophet uses to classify the PSMs comparing to Percolator, as well as the sensitivity of these attributes, increases the risk of overfitting or overestimating the probabilities. However, the performance, robustness and quality of the PSM level probabilities[3] given by Peptide prophet in parametric mode have established it as a state of the art post-processing tool to infer peptide level probabilities.

[1] As we analyze peptides in our tandem mass spectrometry experiments we have to digest the proteins using a protease. This is normally done using trypsin. Trypsin cleaves after arginine and lysine but exhibits also some unspecific cleavage. Two tryptic ends means that both ends were correctly cut by trypsin (wikipedia).

[2] Overfitting generally occurs when a model is excessively complex, such as having too many parameters relative to the number of observations. A model which has been overfit will generally have poor predictive performance, as it can exaggerate minor fluctuations in the data (wikipedia).

[3] Note that the probabilities given by Peptide prophet are related to the PEPs given by Percolator, Peptide prophet gives the probability of the PSM being correct whereas Percolator gives the probability of the PSM being incorrect.

iProphet

iProphet [21] is a tool made to refine or readjust the probabilities obtained with Peptide prophet in order to improve peptide and protein identification rates and error estimates. It is designed to be used as an intermediate step between Peptide prophet and Protein prophet. It combines the evidence from multiple identifications of the same peptide sequence across different spectra, experiments, precursor ion charge states and modified states to obtain new readjusted PSM level probabilities. Their authors claim that iProphet increases the number of correctly identified PSMs at a constant q value. The main advantage of iProphet is that it combines results from different database search engine tools run on the same spectra. That has proven to give more accurate posterior probabilities and q value estimations at PSM level, which is obviously important to obtain better protein level posterior probabilities and q value estimations. What iProphet does basically is to iteratively, by using the expectation maximization algorithm, compute six models[1] using different levels of information extracted from the spectra to finally re-adjust the PSM probability obtained from Peptide prophet. iProphet combines the information from all the models[2] as well as the posterior probability given by Peptide prophet to estimate a new posterior probability for each peptide using Bayesian inference.
The authors of iProphet showed in [21] an improvement of the PSM level probabilities, although the improvement that iProphet gives using only one single database search engine tool is much lower and it does not differ much from the probabilities given by Peptide prophet as we noticed in our experiments (data not shown).

[1] Each model will have a negative and a positive distribution.

[2] NSS (number of sibling searches) + NSR (number of replicate spectra) + NSE (number of sibling experiments) + NSI (number of sibling ions) + NSM (number of sibling modifications).

Protein Prophet

Protein prophet is the tool of the TPP designed to infer proteins and protein probabilities. Protein prophet groups peptides[1] by protein and readjusts the peptide probabilities according to the siblings each peptide has. This gives more relevance to proteins that contain multiple PSMs rather than proteins containing few PSMs. This readjustment of the peptide probabilities is done by computing the negative and positive distribution of the number of sibling peptides (NSP) using the expectation maximization algorithm. The NSP distributions are strongly related to the dataset, more precisely to the sample coverage of the dataset which is just the ratio between the number of peptides and the number of proteins. Readjusting peptide probabilities according to the number of sibling peptides will solve the problem of having higher FDR at protein level by rewarding proteins with more than one peptide. However, there still is the problem with peptide degeneracy which Protein prophet addresses by assigning weights to the peptides. Originally, in a protein all its peptide weights will sum to one. Protein prophet will compute the distributions of weights with the expectation maximization algorithm that aims to derive the simplest list of proteins that includes all the peptides using what is called a parsimony approach (Occam's razor method).
Protein prophet addresses the peptide degeneracy by rewarding proteins with a higher number of peptides but this might not be the best solution to this problem as a protein might be matched by many peptides with a low score. This will inflate the protein probability that in some cases might be even higher than proteins containing only high scored peptides. Also, Protein prophet might assign a very high probability to a protein that is being matched only by one peptide with a high score; and this might not be correct especially in cases where the peptide is a false positive. These issues will grow with the size of the dataset and it is due to the way in which Protein prophet addresses peptide degeneracy. The peptide degeneracy is probably the main issue obstructing an efficient optimization of the protein probabilities since the probability of a protein with degenerated peptides depends on the proteins that its degenerated peptides are matching to.

[1] Note that several spectra can match the same peptide, therefore in those cases Protein prophet will take the PSM with the highest score.

2.3 Fido

Fido [19] implements a Bayesian graph-based method to compute protein posterior probabilities. Regarding the peptide degeneracy problem, Fido rewards protein sets that contain independent evidence in addition to degenerate peptides. As a result of this, a protein with strong independent supporting evidence but with degenerate peptides will have a probability that will not depend totally on the proteins that its degenerate peptides are matched to. Fido uses an optimized marginalization process that theoretically computes every possible combination of proteins and their relative score and it does it in a relatively quick computational time even in large data sets. Fido makes seven assumptions to generate its probability model :

1.- Conditional independence of peptides given proteins.
2.- Conditional independence of spectra given peptides.
3.- Emission of a peptide associated with a present protein.
4.- Creation of a peptide from noise.
5.- Prior belief a protein is present in the sample.
6.- Independence of prior belief between proteins.
7.- Dependence of a spectrum only on the best-matching-peptide.

The first and second assumptions are self-explanatory. The third assumption refers to the probability by which a peptide is generated by a protein containing it. This is therefore a constant parameter and it is called α; this parameter will adjust the probability of some peptides, thus its value will influence the quality of the protein probabilities as we will see in further sections. The fourth assumption refers to the probability of having a truly incorrect peptide erroneously present in a protein, this is also a constant parameter with the name β and it influences the proteins probabilities as well. The fifth assumption refers to the prior belief that any protein is present in the sample, it is also a constant parameter with the name γ and as was the case with the two parameters described above, it affects the protein probabilities. The sixth assumption states that the prior probabilities γ of all the proteins are independent, and finally the seventh assumption accounts for the fact that only the highest scored PSM is kept for each spectrum.
This probability model allows Fido to efficiently estimate the likelihood $P(D|R)$[1] of a set of proteins given a set of peptides. The clever part of the algorithm is that from a graph theory perspective once a set of proteins has been specified, the set of PSMs from each protein is independent of each other, thus the likelihood can be estimated by a product over those graphs which will considerably decrease the computational time. Fido applies Bayes rule to the mentioned likelihood and marginalizes over the set of proteins to estimate the probabilities of each protein in the set based on the simple assumptions of peptide independence between different

[1] D represents the set of PSMs and R represents the set of proteins.

proteins discussed above. However, for larger data sets the computational time can still be prohibitive. Therefore, Fido introduces three pre processing steps that can considerably speed up the computation of protein probabilities :

1.- **Partitioning** : where the graph of protein-peptides is divided into subgraphs that contain only connected proteins, so the probability can be computed individually for each sub-graph.

2.- **Clustering** : where proteins with identical connectivity are grouped together as one protein, as we can see in the figure 2.10 proteins 1 and 2 are grouped together.

3.- **Pruning** : where graphs that contain more than one protein shared by a peptide with a very low confidence score are separated by creating a copy of each peptide for each new graph created. This will create graphs with fewer proteins.

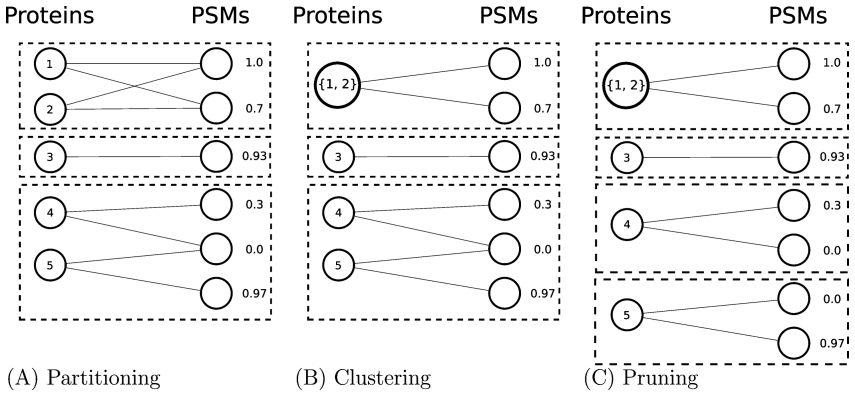

Figure 2.10. Figure showing the effects of the three speeding up procedures. Figure from [19]

Fido performs at least as well as state-of-the-art protein inference tool Protein prophet by comparing the ROC[2] curves and the divergence between empirical and estimated q values as well as the number of proteins correctly identified [19]. The divergence between the empirical and estimated q values will give information about how well calibrated the model is and the ROC curve will give information about the sensitivity and specificity of the model. Ideally, the estimated and empirical q values should follow a similar distribution for each probability and the ROC curve should be displaced towards the y axis. This would imply a low trade-off in terms of the number of false positives in the results.
The problem associated with the estimation of the divergence between empirical and estimated q values is that the empirical q values are estimated according to the

[2] Number of target proteins as a function of the number of decoy proteins.

naive target-decoy approach. However, the assumptions for the target-decoy approach, do not hold at protein level due to the asymmetry of the target-decoy false positive distribution. This asymmetry implies that decoy PSMs will only match incorrect proteins whereas target PSMs will match both correct and incorrect proteins. Therefore the number of false positives is higher at protein level which implies that the FDR is also higher at protein level. Ideally, the empirical q values of the proteins have to be readjusted to account for the problem described above. In order to solve this problem in our integration of Fido with Percolator, we implemented a new method to estimate the ratio of incorrect target proteins to adjust the empirical q values at protein level.

2.3.1 Protein Level False Discovery Rate - MAYU

As described in the previous section, it is a well known problem to efficiently estimate protein level FDR taking into account the asymmetry of the target-decoy entities at protein level. However, a recently published article [18] describes a new approach to estimate protein level FDR efficiently. As described previously, an estimation of the protein level FDR has to account for false and true positive PSMs distributed differently across the protein database because false positive PSMs are distributed over all entries in the database whereas true positive PSMs are only distributed over the small subset of present proteins.
What the authors of Mayu propose is an extension of the well established target-decoy competition model to estimate FDR at protein level under the assumption that a protein is considered to be a true positive if it contains at least one true positive PSM and that a protein is considered to be a false positive if all its PSMs are false positives. With this assumption a protein can be a true positive even if it has false positive PSMs. Another important assumption is that for proteins with the same length[1] the number of false positives are uniformly distributed over the target proteins. Given this, and assuming that the number of false positives follows a hyper-geometric distribution, the expected number of false positive proteins can be estimated as the expectation value[2] of the distribution. Having the expected number of false positive proteins allows us to straightforwardly estimate the protein level FDR as the ratio of the total number of expected false positive proteins and the total number of target proteins. The estimated FDR also allows us to compute empirical q values for a given threshold by simply multiplying the protein level FDR by the ratio of decoy proteins and target proteins at the given threshold. In figure 2.11 the procedure to estimate the protein FDR is described.
Basically what Mayu does can be described in a few steps.

1.- It bins up all the proteins of the database according to the protein length

2.- It estimates the expected number of false positives of each bin using a hyper-geometric distribution of the total number of proteins in the bin, the number of target proteins in the bin that contain PSMs with an empirical q value below 1% and the number of decoy proteins in the bin that contain PSMs with an empirical q value below 1%

3.- It estimates the FDR of the bin by dividing the expected number of false positives by the number of target proteins in the bin.

4.- It sums up the FDRs of each bin and divides the sum by the total number of target proteins of the database that have PSMs with an empirical q value below 1% to obtain the protein level FDR.

[1] The number of peptides obtained by tryptic digestion.
[2] The probabilistically weighted average of the distribution.

The authors of Mayu showed in [18] that their estimation of the protein level FDR is more accurate and robust than the FDR estimated by the naive target-decoy competition approach, especially when the size of the database is relatively big. Furthermore, we decided to include in Percolator a protein level FDR estimator based on Mayu. We use the estimated protein level FDR to adjust the empirical q values at protein level as we will see in the implementation section.

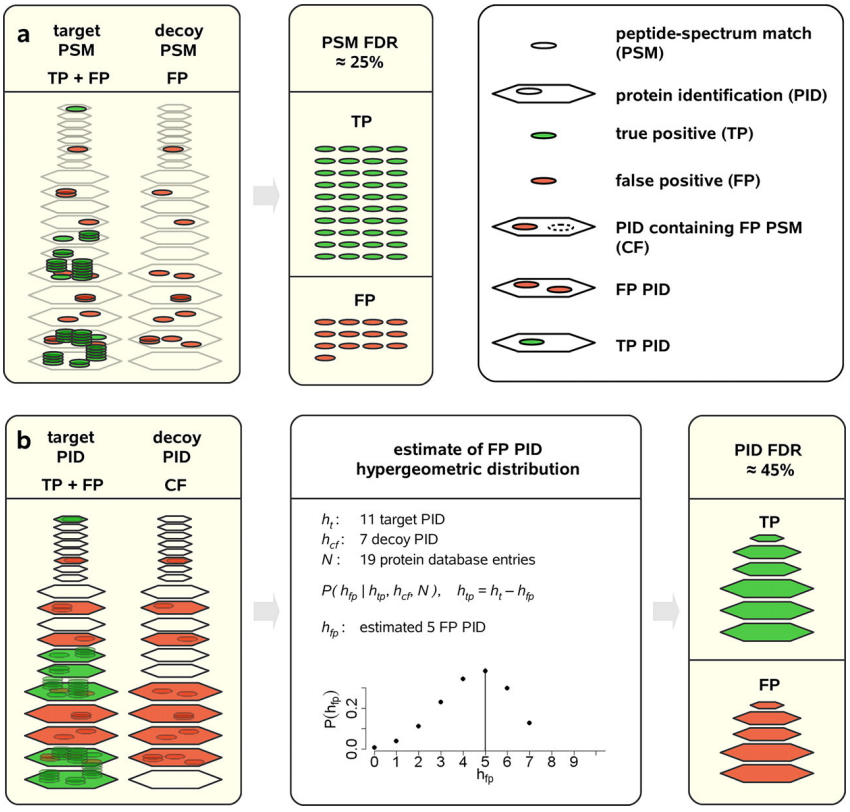

Figure 2.11. Mayu protein identification false discovery rate estimation. In a we can see the estimation of the FDR at PSM level using the target-decoy competition. In b we can see the protein identification done by Mayu. Figure from [18]

Part II

Experimental part

Chapter 3

Methods and Implementation

3.1 Integration of Fido into Percolator

We have described in the previous sections all the theory and concepts that had to be understood in order to perform the integration of Fido into Percolator. We began by discussing what a tandem mass spectrometry based experiment consists of, the main characteristics of the state-of-the-art peptide level post-processing tools and also the reasons why they play a relevant and important role in the workflow of a tandem mass spectrometry based experiment. We have given a description of the main statistical measurements used to rank the PSMs, peptides and proteins. Finally, we introduced the state-of-the-art protein inference tools, Protein prophet and Fido and explained the important role they play in a shotgun proteomics experiment. The main problems of estimating protein level probabilities efficiently, were mentioned along with the way in which they are addressed by Fido and Protein prophet. We also introduced the problems associated with the estimation of the FDR at protein level and the solution proposed by the authors of Mayu.
In this section we will describe the procedure followed to integrate Fido into Percolator to allow Percolator to infer proteins and compute protein probabilities efficiently. We will describe the main parameters that we have included in Percolator for the estimation of protein probabilities, we will also describe the procedure followed for the calibration of these parameters and how they can be modified to gain better performance. These parameters play a role in the performance of Fido and the way the statistics are estimated. They have been included to allow the user to gain better performance for specific scenarios. All the parameters have been initialized to the optimal values for the most common scenarios. We will also describe our implementation and integration of a protein level FDR estimator based on the description given by the authors of Mayu and we will describe how we use the estimated protein level FDR to adjust and improve the empirical q values used in our model, we can then use the adjusted empirical q values to calibrate the parameters and meassure the performance of the model. We will describe how we estimate the statistically relevant measurements (q values and p values) and finally we will

explain the procedure followed to benchmark and compare the results given by our integration against the state-of-the-art protein inference tool Protein prophet.
Fido's source code has been written in C++ which facilitated the integration into Percolator which has also been written in C++. The original Fido implementation takes as input a list of PSMs, each PSM includes its probability of being present[1] and the list of proteins that the PSM is matching to in the database. Fido also takes as input the values of the parameters α , β and γ. These parameters affects the model that Fido uses to estimate the probabilities as described in section 2.3. Percolator outputs a list of unique peptides with their respective posterior error probabilities, that can easily be converted to posterior probabilities using this simple formula for a given peptide $X : P(H_1 \mid X) = 1 - P(H_0 \mid X)$. Fido choses the highest scored PSM for each peptide prior the computation of the protein probabilities, however, percolator re-scores the unique peptides which for instance gives better calibration at peptide level than the method used by Fido. The integration of Fido into Percolator was done by creating an interface between Fido and Percolator. This interface takes the set of unique scored peptides given by Percolator and submits them to Fido prior to previous adaptation. Fido will then estimate the protein probabilities according to the given set of peptides and their respective posterior probabilities as well as the set of parameters that Fido needs for its model. Once the protein probabilities have been estimated, Percolator will estimate the respective statistics such as q values and output the final results.
As mentioned previously, Fido needs a value for the parameters α , β and γ. The value of these parameters affects the model that Fido uses to estimate the probabilities, thus they need to be initialized according to the dataset to give the most calibrated and robust results possible. To address this problem, we implemented a grid search over the three parameters. The grid search iterates over three possible ranges of values that are already defined for α , β and γ. After that, it obtains the protein probabilities for each combination of values for the parameters and calculates a score using an objective function that measures the quality of the protein probabilities to keep the α , β and γ with the best score. We selected the following objective function :

$$F(R) = (\lambda ROCN(R)) - ((1 - \lambda)MSEFDR(R)) \tag{3.1}$$

Where R represents the set of proteins and their respective posterior probabilities. The parameter λ will balance the equation. For a λ close to 1 the model will be more accurate, giving more relevance to the score given by the ROCN curve

[1] Posterior probability given by any post-processing tool.

function. For a λ value close to 0 the model will be calibrated to give more relevance to the score given by the MSE FDR divergence function. In our experiments we tested how the system performs for different values of λ, concluding that a λ with a low value (0.10 - 0.20) consistently performs better in terms of the number of correctly identified proteins, as well as regarding sensitivity and calibration. We have established a default value for λ of 0.15 which has given the best results in our experiments, although we have included a parameter in Percolator to enable the user to modify the λ value.
In the objective function, what we are mainly aiming for is the best relationship between empirical q values and estimated q values, this is estimated by the MSE FDR divergence function. The MSE FDR divergence function estimates the squared error of the area of the difference between the empirical and the estimated q values for a given set of proteins and their respective posterior error probabilities. The MSE FDR divergence function gives a score to the level of separation between the empirical and the estimated q values. The level of separation defines how well calibrated the protein probabilities given by Fido are. The ROCN function assigns a score to the average ROC sensitivity curve allowing between 0 and N decoy proteins. The ROCN function will compute the area of the ROC curve rewarding a high sensitivity and a high specificity. The objective function 3.1 tries to minimize the MSE FDR divergence and maximize the ROCN functions values.
In order to run the grid search to estimate the optimal values for the parameters α , β and γ, we needed to define the range of values for each parameter. We noticed in our experiments that the best values for each parameter are always oscillating in a small specific interval, and in the particular case of γ its optimal value is 0.5 in most of the cases. The size of the range of values will affect the computational speed of the procedure of estimating the protein probabilities. Therefore, we decided to include a new parameter in Percolator to indicate the depth[1] of the grid search. The different range of values for every depth level are included in appendix A.
While computing the MSE FDR divergence, we only compare the empirical and estimated q values that are below a certain threshold. The reason for doing this is to consider only the proteins inside a high confidence interval. In practice, we are only interested in the proteins with a q value lower than 10% (0.1) and in the majority of the cases 5% (0.05) or even 1% (0.01). Therefore, we are interested in a low divergence between estimated and empirical q values for proteins inside a high confidence interval or in other words a low q value.
We have run several experiments using different values for the mentioned threshold and the results showed that at a very low threshold of 1%, the system performs in a more conservative way giving a lower number of confident proteins but with a very high specificity in terms of the number of false positives. This is clearly the result of using a small threshold whereby the values of the divergence between empirical and estimated q values are smaller and so the objective function will give more relevance to the trade-off between sensitivity and specificity given by the ROCN function. Al-

[1] Where depth is the size of the range of values for each parameter in the grid search.

though, this implies that the divergence between estimated and empirical q values might be bigger, and thus that the protein probabilities might have a lower calibration. The default value for the threshold that we have chosen is 5% (0.05) which gave the best performance in our experiments, although we have included a parameter in Percolator which enables the user to modify the threshold of the MSE FDR divergence function. A value of 0.05 for the MSE FDR divergence threshold parameter means that only proteins with an estimated q value below 0.05 will be taken into account for the estimation of the divergence between estimated and empirical q values in the MSE FDR divergence function. A more clear description of the scoring function is shown in **??**.

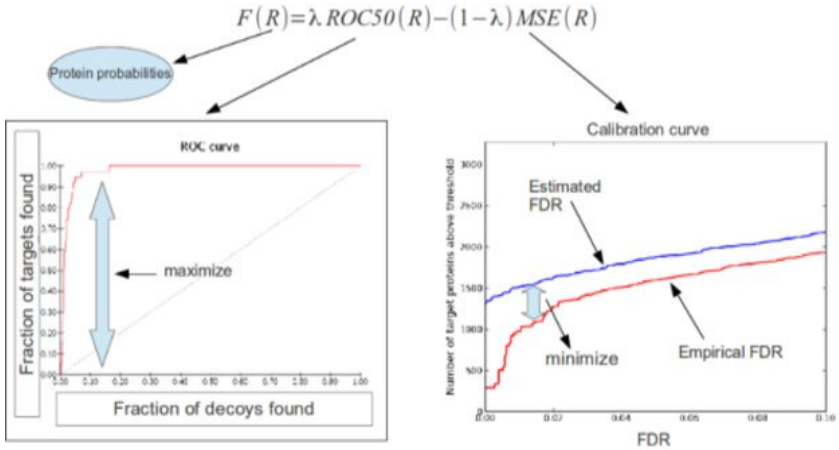

Figure 3.1. Figure that shows the way the ROCN and MSEFDR compute a score for the quality of the computed protein probabilities.

Different values for the referred threshold might give a better performance for different scenarios, for example in cases where the specificity of the model is more important. Also in cases where we use a small dataset where a higher threshold may be needed in order to compensate for the small number of q values taken into account when estimating the MSE FDR divergence score.
Another factor to be considered is the maximum number of decoy proteins allowed at a certain threshold for calculating the ROCN curve. What the ROC curve gives is the trade-off between specificity and sensitivity for a given set of protein probabilities. Sensitivity would be given by the ratio of true positive proteins and specificity would be given by the ratio of false positive proteins. In short, the ROC curve is estimated by comparing the ratio of true positives against the ratio of false positives in the range of probabilities 0-1, in order to see the trade-off between the number of true positives and false positives of the results given by the model. Ideally, what we want is a curve shifted to the top left which means a set of protein probabilities with a high sensitivity and a high specificity. In other words, a high number of true positives and a low number of false positives. We assign a score to the curve by estimating its area, that is why our objective function tries to maximize the ROCN

value. Usually, for most of the standard experiments the ROC curve is estimated allowing 50 false positives, that is what is called ROC50 curve. We performed different experiments modifying the N value and there was no apparent improvement for most of the datasets. However, in some datasets the performance was better when the N value of the ROCN curve function was close to the number of decoys proteins with an estimated q value of approximately 0.05. Therefore, we decided to compute the N value of the ROCN curve function according to the number of decoy proteins present at an estimated q value of 5% in the set of protein probabilities being scored for the grid search.
During the implementation of the interface between Fido and Percolator, one of the issues that we wanted to improve was the computational speed. We optimized the code of Fido and our integration to try to decrease the computational time to minimum levels. At the same time, we attempted to maintain the performance and quality of the results. For this purpose, we added new parameters to Percolator to configure and control the computational speed and quality of the results. We also ran experiments in order to define the set of parameters and the data structures that give a good performance while keeping a low computational execution time. In order to gain computational speed, we introduced three new parameters that can speed up the procedure if used accordingly :

Grouping = it activates or deactivates the grouping of protein with similar connectivity, for example if proteins P1 and P2 have the same peptides matching both of them, P1 and P2 can be grouped as 1 single protein P3. By default the grouping is deactivated which decreases the speed but increases the efficiency of the results.

Pruning = it activates or deactivates the pruning of peptides with a very low score when activated (approx. 0.0). This means that if a peptide with a very low score is matching two proteins P1 and P2 forming one group, we can prune the peptide which will duplicate it in order to generate two new groups, a group for P1 and another group for P2. By default, the pruning is activated which increases the speed.

Separation = it activates or deactivates the separation of the protein-peptide graphs into sub-graphs. By default, protein probabilities will be estimated individually for each sub-graph which increases the speed.

Reduce tree = it activates or deactivates the reduction of the low scored peptides in order to obtain a smaller number of proteins. This will speed up the grid search process.

All the parameters included in Percolator for protein level probabilities estimation, as well as their default and recommended values, are described in the parameters appendix section A.

3.1.1 Protein level FDR

As described in section 2.3.1 the premises of the standard target-decoy competition method to estimate the FDR do not hold totally at protein level due to the asymmetry of the target and decoy false positives distribution. A good solution for this problem is proposed by the authors of Mayu [18]. Given the good results shown by the authors of Mayu, we decided to implement a protein level FDR estimator based on this solution .
The implementation was done following the description of the method given in [18], although there was a complication when it comes to estimating the protein length[1] and the total number of proteins in the database including the proteins that were not matched against any spectra. Percolator did not provide that information, thus we needed to add a new parameter to Percolator to pass a link to the original FASTA[2] database containing the target and decoy proteins that were used to match the spectra. Besides that, the implementation went smoothly without any particular complication. Once again, we tried to ensure that the procedure would be as quick and as robust as possible. In order to test that the protein level FDR estimated by our implementation was similar to the protein level FDR estimated by Mayu, we performed experiments with three different datasets. The results are shown in table 3.1

Table 3.1. Comparison of the protein level FDR estimated by Percolator and Mayu. The second and fourth columns represent the estimated protein level FDR of the whole dataset. The third and fifth columns represent the estimated total number of expected false positive proteins in the whole dataset.

Dataset	Percolator/Fido FDR	Percolator/Fido exp. FP	Mayus FDR	Mayus exp. FP
Yeast	34%	698	31%	694
Yeast Klammer	27%	397	25%	380
Human Leukate	33%	126	29%	118

As we can see, the estimated protein level FDR and the expected number of false positive proteins are basically the same for Mayu and our protein level FDR estimator when they are both run on the same dataset. There is, however, a small difference. This is due to implementation details. Mayu groups the proteins according to their genes in order to estimate their length; proteins of the same gene group that have similar tryptic digested peptides will only be counted once. For example, two proteins P1 and P2 that form part of the same gene group and have a particular tryptic digested peptide in common. This tryptic digested peptide will only be counted for P1 when estimating the length of P1 and P2. Since Mayu defines the protein length as the number of tryptic digested peptides in the protein

[1] Where length here stands for the number of peptides obtained by tryptic digestion.
[2] FASTA format is a text-based format for representing either nucleotide sequences or peptide sequences, in which nucleotides or amino acids are represented using single-letter codes.

sequence, it only counts different tryptic digested peptides for proteins belonging to the same gene group while estimating the protein length. We skipped this part of the implementation to gain computational speed. Also, when Mayu estimates the empirical q values for the set of PSMs, then only the subset of proteins that contain PSMs with an empirical q value below a certain threshold will be used for the estimation of the protein level FDR. The way Mayu estimates the empirical q values is slightly different from the way Percolator estimates the empirical q values at PSM level, Percolator adjusts them by including the ratio π_o. Therefore, the number of proteins containing PSMs with an empirical q value below a threshold x will be slightly different between Percolator and Mayu and that will also affect the results, although not considerably as can be seen in table 3.1.
The main reason to include a protein level FDR estimator is to adjust the empirical q values accounting for the number of false positive proteins. This new set of *adjusted* empirical q values are also used in the estimation of the MSE FDR divergence curve while doing the grid search to estimate α , β and γ. We performed experiments with and without use of the protein level FDR to adjust the empirical q values. We could observe that the results obtained were more calibrated when comparing the adjusted empirical q values with the estimated q values. In some experiments the number of correctly identified proteins were even higher due to the use of this new adjusted empirical q value in the grid search for the estimation of α , β and γ preceding computation of the protein probabilities.
As mentioned previously, Percolator estimates empirical and estimated q values. They are both used in the calibration of the parameters of the model that Fido uses to estimate the protein probabilities. The estimated q values are calculated using formula 2.6. The empirical q values can be estimated in three different ways. By the standard target-decoy approach described in formula (2.2), by the standard target-decoy approach and adjusting with π_o described in formula (2.3), and finally by the standard target-decoy approach but using the protein level FDR to adjust for the false positive proteins by this formula :

$$\widehat{qvalue}_{gfdr}(x) = gfdr(T, D, PSMs) \times \frac{\mid \{y \mid y \geq x, y \in D\} \mid}{\mid \{y \mid y \geq x, y \in T\} \mid} \tag{3.2}$$

Here *gfdr* stands for our method that estimates the protein level FDR based on the implementation of Mayu. The function gfdr uses the set of target proteins, the set of decoy proteins and the set of scored PSMs. D is the set of decoy proteins, T is the set of target proteins and x is the probability score threshold.
In general, what we do by including the protein level FDR into the equation is account for the ratio of the expected number of false positive proteins. A similar idea is used behind the ratio π_o that adjust the empirical q values accounting for the

number of incorrect target entities. That is why we have included both options in our implementation. The user can decide to adjust the empirical q values by either π_o or the protein level FDR, or simply decide not do any adjustment. In practical terms, it is better to adjust the empirical q values with the protein level FDR since the assumptions we make to estimate π_o are not completely valid at protein level. Also, if we decide to adjust the empirical q values with π_o, it will only affect the final empirical q values that are going to be included in the output, and not the empirical q values that we use in the MSE FDR divergence function to calibrate the parameters of the model. The reason for this is that we need the set of proteins to be scored prior to the estimation of the π_o. This will imply that if we want to adjust the empirical q values by π_o prior to the estimation of the MSE FDR divergence curve, we would need to estimate a π_o for each set of protein probabilities that are scored in the grid search. This will be computationally prohibitive. Therefore, it is more logical to adjust the empirical q values with the protein level FDR.
The figure 3.2 shows a considerable improvement in the number of target proteins as a function of the adjusted empirical q values (red) and the standard(unadjusted) empirical q values (blue). In figure 3.3 we can see the performance in terms of calibration. The correlation between the empirical and estimated q values for the adjusted q values (red) is bigger than the unadjusted q values (blue). In the figures 3.4 and 3.5 we can observe the same effect but with a different dataset.

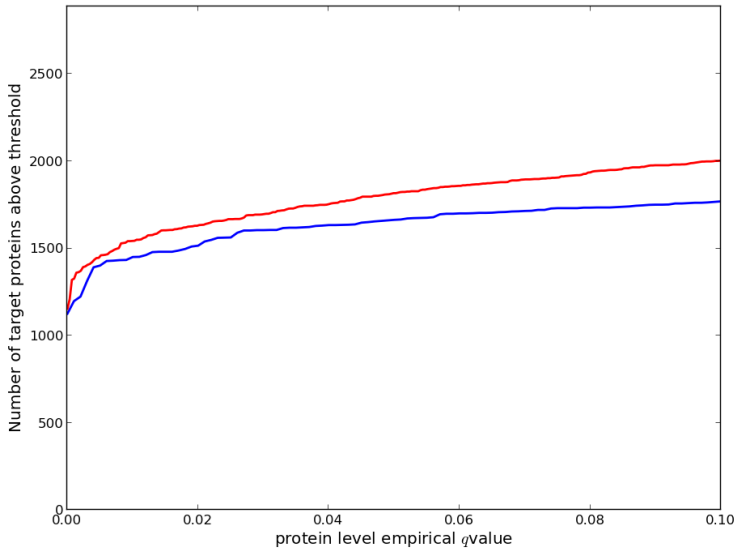

Figure 3.2. Number of target proteins as a function of the empirical q values. Blue : Percolator/Fido without using the protein level FDR adjustment. Red : Percolator/Fido using protein level FDR adjustment. Yeast dataset.

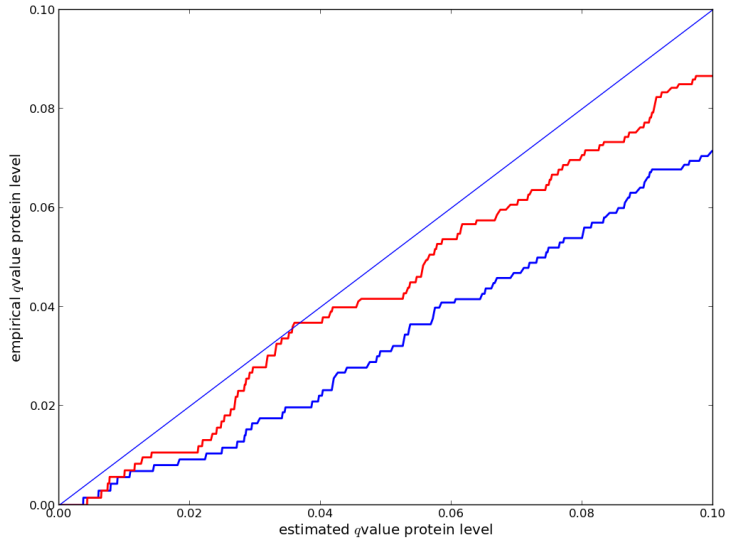

Figure 3.3. Estimated q values as a function of the empirical q values. Blue : Percolator/Fido without using protein level FDR adjustment. Red : Percolator/Fido using protein level FDR adjustment. Yeast dataset.

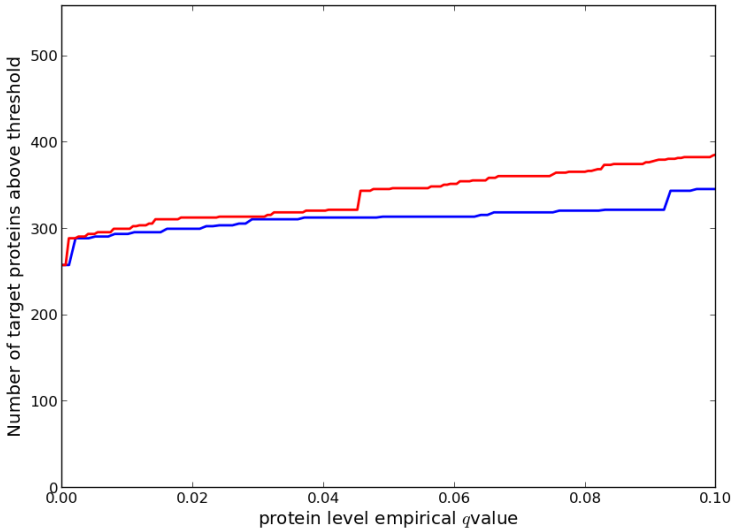

Figure 3.4. Number of target proteins as a function of the empirical q values. Blue : Percolator/Fido without using protein level FDR adjustment. Red : Percolator/Fido using protein level FDR adjustment. Streptococcus dataset.

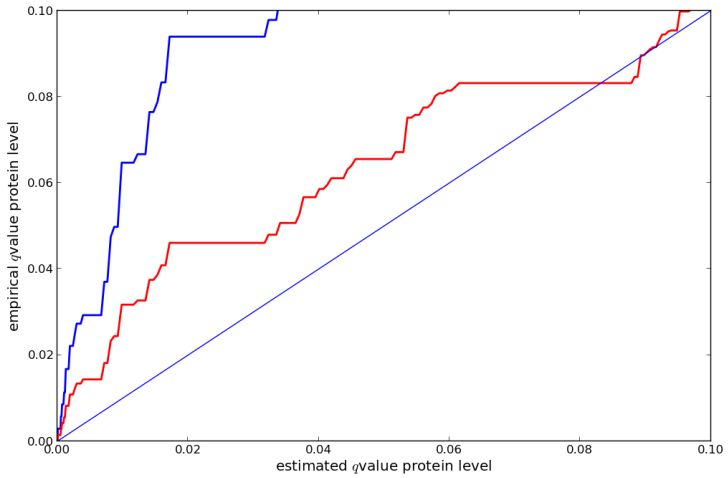

Figure 3.5. Estimated q values as a function of the empirical q values. Blue : Percolator/Fido without using protein level FDR adjustment. Red : Percolator/Fido using protein level FDR adjustment. Streptococcus dataset.

3.1.2 Datasets and experimental information

For all the experiments and tests done in this thesis work we used three different publicly available datasets. A yeast dataset, a human leukate dataset and a streptococcus dataset.

Table 3.2. Different search parameters used with Crux to match the spectra against the databases.

Dataset	Enzyme digestion	Mass tolerance	MS1 mass type	MS2 mass type	Missed cleavages
Yeast	trypsin/partial-digest	+/-10 ppm	mono	average	Yes
Human Leukate	trypsin/partial-digest	+/-3 daltons	average	average	Yes
Streptococcus	trypsin/partial-digest	+/-3 daltons	mono	average	Yes

The fragmentation spectra files were searched with Crux v1.37, using the sequest-search command. The parameters that we used to perform the searches are described in the Table 3.2.
The yeast dataset is composed of two runs of raw MS2 spectra files that were matched separately. The results shown in this thesis work correspond to the first run. In the sample preparation the proteins were digested to peptides by incubation with trypsin for 16 hours at room temperature. Mass spectra were acquired on a LTQ Velos Orbitrap (Thermo Fisher Scientific) mass spectrometer operated on an 11-scan cycle consisting of a single high-resolution precursor scan event at 60,000 resolution (at 400 m/z) followed by ten data-dependent MS/MS scan events using collision induced dissociation (CID). The spectra files were matched against the ENSEMBL 3.64 protein database.
The Streptococcus dataset is composed of 12 runs raw mzXML spectra files obtained from the PeptideAtlas raw data repositories (http://www.peptideatlas.org/repository/) as accession PAe000283. All the spectra files were matched separately and combined afterwards. The spectra files were matched against the Streptococcus database RefSeq (version NC002737) appended with the human IPI database version 3.54.
The Human Leukate dataset is composed of 18 runs RAW mzXML files obtained from the global repository (https://proteomecommons.org) under the name of *Global survey of Human T leukemic cells*. The mxXML files used in the experiment were the whole cell lysate number 1. All the spectra files were matched separately and combined afterwards. The spectra files were matched against human RefSeq protein sequence database.
For every database we obtained a decoy version by reversing the sequences, and we matched all the spectra files against the reversed decoy databases. For every dataset, the results obtained by matching the spectra against the target and decoy databases were processed by Percolator/Fido version v.2.4.0 and the TPP pipeline version v.4.5.0. For Percolator/Fido, the results obtained from the matching of the

spectra against the databases were processed and combined with *sqt2pin* in order to generate a single input file containing both target and decoy PSMs for each dataset. The files containing the PSMs were processed with Percolator/Fido (with the flag to estimate protein probabilities activated and the default configuration for the rest of the parameters).
For the TPP pipeline, the results obtained from the matching of the spectra against the databases were processed and combined with *Interact* in order to generate a single input file containing both target and decoy PSMs for each dataset. The files containing the PSMs were processed with Peptide prophet in both the parametric (unsupervised) and non-parametric (semi-supervised) mode. For the parametric mode we used only the files containing target PSMs. For every dataset the default models of Peptide prophet were used, except for the streptococcus and the yeast dataset where the accurate mass model was activated. The results obtained from Peptide prophet were processed with iProphet (note that we are only using one search engine). The results obtained from iProphet were then processed with Protein prophet using its default configuration.
In summary, all the datasets were processed with our implementation of Percolator and Fido with the default configuration of parameters and with Peptide prophet/iProphet/Protein prophet in both parametric and semi-parametric mode.

Part III

Results and Conclusions

Chapter 4

Results and Conclusions

4.1 Results

Initially, we ran the first experiments using a simple target-decoy procedure. We used the original database as the target database and its reversed sequences as the decoy database. We matched the spectra files against both the target and decoy databases as described in the previous section. We used the decoy PSMs as negative examples to train the model and the target PSMs as positive examples. Protein probabilities were estimated from the unique peptides (previous adjustment with iProphet in the case of Protein prophet). We measured the performance of the results by using the empirical and estimated q values computed as described in the section 2.2.1. The ideal shotgun proteomics protein inference tool performs an efficient classification of present and non-present proteins and rank the proteins according to their probability of being present in the analyzed sample. We want the protein inference tool to be robust, with a high calibration[1], a high sensitivity[2] and a high specificity[3]. Using the set of scored decoy proteins is a good way to measure how robust and sensitive our protein inference tool is. If there is a large number of high confident proteins in the results but also a high number of false positive proteins, this means that the protein inference tool performs a poor classification. This situation is not desirable.
In order to compare the performance and quality of the results given by our integration of Percolator and Fido and the results given by Protein prophet, we performed two different experiments. First, we used a standard target-decoy based experiment and estimated the empirical and estimated q values from the set of proteins and their respective posterior error probabilities. We then compared the number of correctly identified proteins as a function of the estimated q values and as a function of the empirical q values, always in a high confidence interval (1% - 10% q value).

[1] A highly calibrated protein inference tool is one that has a good correlation between its estimated and empirical q values.
[2] Sensitivity is the ability of the protein inference tool to detect correct proteins.
[3] Specificity is the ability of the protein inference tool to detect incorrect proteins.

We then compared the correlation between empirical and estimated q values in order to measure the calibration of the results. We observed that the performance of percolator/fido was good in terms of calibration and sensitivity, we found in the results a high number of correct proteins with a low rate of false positives and a good correlation between empirical and estimated q values. On the other hand, the performance of Protein prophet when using peptide level probabilities estimated with Peptide prophet in semi-supervised mode and readjusted with iProphet, was not good. The number of high confident proteins given by Protein Prophet was higher than the number given by Percolator and Fido, but with a big trade-off in calibration, specificity and accuracy, including a very high number of false positives in the high confidence interval of the results. We suspect that a possible explanation for this poor performance is the overestimation of the peptide probabilities given by Peptide prophet in semi-supervised mode as we described in section 2.2.2. Therefore, if the set of peptides have overestimated probabilities and a high number of false positives, Protein prophet will have more chances to infer overestimated protein probabilities with a high number of false positives. Also, the problems associated with Protein prophet described in section 2.3 could be possible explanations for the high sensitivity, low specificity and low calibration given in the results of Protein prophet in this particular scenario of standard target-decoy competition. Therefore, in order to make the comparison more fair and accurate, we performed another experiment.

For every decoy database we generated two new sets of decoy databases using the tool *Mimic*, a decoy database scrambler that conserves homology. One of the sets was appended to the target database and the other one was appended to the decoy database in order to have target and decoy databases of the same size. Therefore, we created a new target and a new decoy database. The new target database contained the original protein sequences and a set of decoy proteins that from now on we will be called *hidden decoys*. The new decoy database contained only decoy protein sequences and it was of the same size as the target database. The procedure that we followed to create the databases is illustrated in figure 4.1. What we aim to by using these hidden decoys is to have separate sets of decoys for training the model and for generating the statistics, since we use the statistics to measure the performance. Therefore, we use the normal decoys to train the model and the hidden decoys to compute the statistics (q values, p values and so on) and benchmark. The hidden decoys experiment allowed us to measure the performance, calibration and sensitivity of Percolator/Fido and Protein prophet more accurately. It also allowed us to compare our integration against the results given by Protein prophet using peptide level probabilities given by Peptide prophet in unsupervised mode and readjusted by iProphet. The computation of the estimated q values as described in the equation 2.6 will not be affected by the hidden decoys. However, the computation of the empirical q values is slightly different using this procedure since we discard the set of decoy proteins when estimating the statistics and use the set of hidden decoys instead. The new formula to estimate empirical q values using the hidden decoys is :

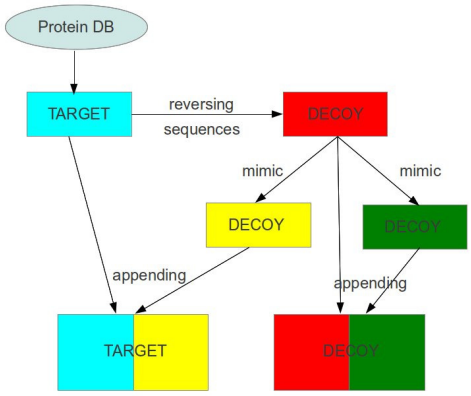

Figure 4.1. Figure showing the procedure for creating the target database with hidden decoys and the respective decoy database, both of the same size.

$$\widehat{qvalue}(x) = r \times \frac{|\{y \mid y \geq x, y \in H\}|}{|\{y \mid y \geq x, y \in T\}|} \tag{4.1}$$

Where r stands for the ratio of hidden decoys in the target database which in our case is 1/2, assuming that a hidden decoy is twice as likely to be matched to a target in the database search procedure. H stands for the set of hidden decoys and T stands for the set of targets (including the hidden decoys). From now on, every time we refer to empirical q values in the results section, we estimate them with equation 4.1.
All the figures and results shown in this section were generated using the hidden decoys procedure. Also, in order to make a fair comparison with Protein prophet, the empirical and estimated q values that we used in the experiments were both estimated in the same way for Percolator/Fido and Protein prophet. The estimated q values were estimated using equation 2.6 and the empirical q values were estimated using equation 4.1. We will proceed now to show the main results that we obtained from our comparative experiments.
First of all, in table 4.1 we indicate the number of correct proteins at an estimated q value of 10%, the number of hidden decoys at an estimated q value of 10% and the number of decoys at an estimated q value of 10% obtained in our experiments. As we can see in table 4.1, our integration outperformed Protein Prophet for every dataset. The number of target proteins obtained with Protein Prophet in semi-supervised mode[1] was higher than the number of target proteins obtained with Percolator/Fido. However, this came at a high cost in terms of sensitivity, specificity and calibration of the results. As we can see in table 4.1, the number of false

[1] Referring to the results given by Protein Prophet where the peptide probabilities were obtained with Peptide prophet in semi-supervised mode and readjusted with iProphet afterwards.

positives (hidden decoys) found at a high confidence interval (up to 10% q value) in the results given by Protein Prophet in semi-supervised mode, was considerably higher. The overestimation of the protein probabilities given by Protein prophet in semi-supervised mode is clear, it does not perform a good classification. However, the performance of Protein prophet in unsupervised mode[2] was good, even though it was worse than the performance of Percolator/Fido, it presented good levels of sensitivity and specificity. However, the number of proteins correctly identified was considerably lower than the number obtained with Percolator/Fido.

Table 4.1. Performance of Percolator/Fido compared to Protein prophet in both semi-supervised and unsupervised mode.

Dataset	protein inference tool	target proteins	hidden decoy proteins	decoy proteins
Yeast	Percolator/Fido	2024	253	707
Yeast	Protein prophet semi-supervised	2578	1943	4554
Yeast	Protein prophet unsupervised	1618	63	-
Streptococcus	Percolator/Fido	416	76	167
Streptococcus	Protein prophet semi-supervised	1328	1090	3016
Streptococcus	Protein prophet unsupervised	335	27	-
Human Leukate	Percolator/Fido	227	78	161
Human Leukate	Protein prophet semi-supervised	247	209	498
Human Leukate	Protein prophet unsupervised	124	5	-

We present now a series of figures that will describe the results obtained in our experiments in a more clear way. We will present and discuss the results obtained with the yeast dataset at protein level, although the results obtained with all the datasets at PSM, peptide and protein level are included in appendix B.
In figure 4.2 we can see the number of target proteins as a function of the number of hidden decoy proteins. Ideally, for this type of graph, a good performance would be shown with a curve shifted to the top left side of the graph. This would mean that the results given by the protein inference tool have a high sensitivity and a high specificity. As we can see in red, the low specificity of Protein prophet in semi-supervised mode is obvious, this is another indicator of the overestimation of the protein probabilities given by Protein prophet in semi-supervised mode. We have actually observed this tendency for all the datasets at PSM, peptide and protein level. From figure 4.2 we can also infer that Percolator/Fido outperforms Protein prophet in unsupervised mode. In this case, our results presented a bit lower specificity expressed in a slightly higher number of false positives, but still at acceptable values as we can see in table 4.1.
In figure 4.3 we present the results obtained in terms of sensitivity. We show the number of target proteins as a function of the estimated q values in a high confidence interval (0% - 10% q value). Percolator/Fido clearly outperformed Protein prophet in unsupervised mode. However, Protein prophet in semi-supervised mode

[2] Referring to the results given by Protein prophet where the peptide probabilities were obtained with Peptide prophet in unsupervised mode and readjusted with iProphet afterwards.

gave a higher number of target proteins in the high confident interval but with a big trade-off in specificity and calibration for the reasons that we described above.

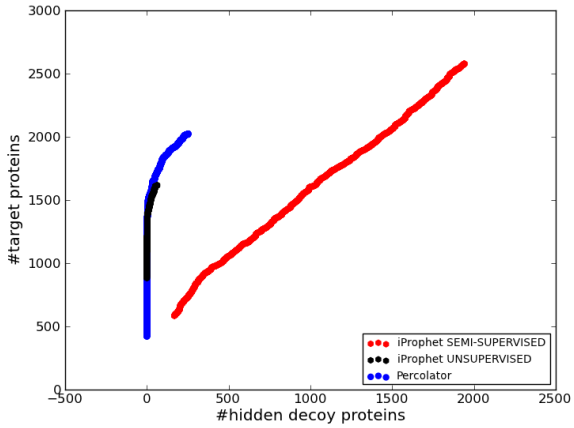

Figure 4.2. Yeast dataset. Figure showing the target proteins as a function of the hidden decoy proteins.

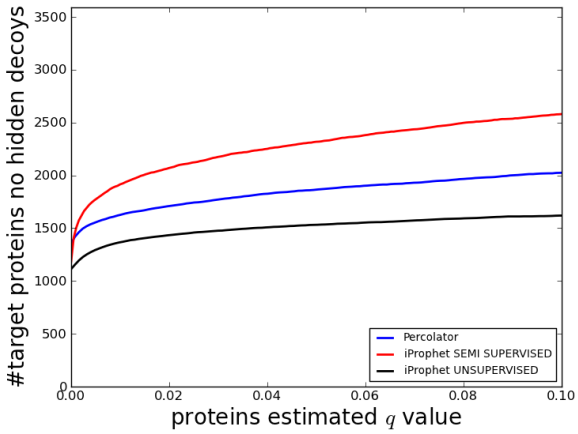

Figure 4.3. Yeast dataset. Figure showing the number of target proteins (not including hidden decoys) as a function of the estimated q values.

If we were only looking at figure 4.3 to measure the performance of our protein inference tool, we could think that Protein prophet in semi-supervised mode performs better due to the higher number of high confident target proteins identified. However, the performance of a protein inference tool cannot be measured only in terms of the number of high confident target proteins. We also need to consider the quality, calibration and sensitivity levels of the results. Protein prophet in semi-

supervised mode showed a clear tendency to overestimate the results making them more unreliable than the results given by Percolator/Fido or Protein prophet in unsupervised mode.
We can see in figures 4.4 and 4.5 the results in terms of calibration, sensitivity and specificity. Figure 4.5 shows the number of target proteins as a function of the empirical q values estimated with the formula 4.1. As we can see, Percolator/Fido outperforms Protein prophet in unsupervised mode. We can clearly notice the low specificity and calibration of the results obtained with Protein prophet in semi-supervised mode. This is represented as a low correlation between empirical and estimated q values. We can also notice that the number of target proteins as a function of the empirical q values is very low, this is due to the very high number of false positive proteins given by Protein prophet in semi-supervised mode. Percolator/Fido and Protein prophet in unsupervised mode showed good levels of calibration as the number of target proteins obtained at both estimated and empirical q value was very similar, represented by similar curves in figures 4.3 and 4.4 . Also, figure 4.5 shows the level of calibration in a more clear way by comparing the estimated and empirical q values. Percolator/Fido and Protein prophet in unsupervised mode showed good levels of calibrations, whereas Protein prophet in semi-supervised mode showed a very bad calibration in the results where the correlation is hardly appreciable.
We obtained the same results for every dataset that we used in our experiments at all levels (PSM, peptide and protein). Percolator/Fido and Protein prophet in unsupervised mode gave both good levels of calibration and specificity. Although, Percolator/Fido outperformed Protein prophet in unsupervised mode in terms of sensitivity and number of entities correctly identified. For Protein prophet the results obtained were always similar; overestimated probabilities with a very poor classification in terms of calibration and specificity.

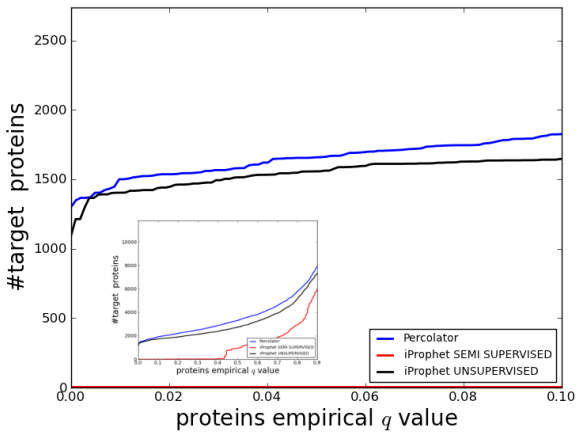

Figure 4.4. Yeast dataset. Figuring showing the number of target proteins (not including hidden decoys) as a function of the empirical q value.

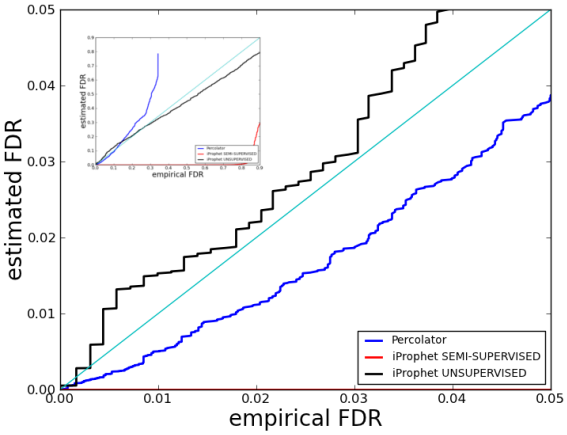

Figure 4.5. Yeast dataset. Figure showing the estimated q value as a function of the empirical q value. As we can see, the red curve can only be seen in the small sub-graph on the top left due to the very low correlation given by Protein prophet in semi-supervised mode. The blue diagonal represents a perfect correlation.

4.2 Conclusions and future work

Shotgun proteomics is recognized as one of the main techniques for protein identification and quantification. However, there is still a long way to go when it comes to the data analysis. Current state-of-the-art tools for shotgun proteomics data analysis and post-processing, are mainly focused on and optimized for the peptide level. However, what we are looking for is the list of proteins believed to be present in the analyzed sample. We described in the introduction the reasons why the set of confident proteins should not be inferred directly from the set of scored peptides. In this thesis work we present a tool to infer the list of candidate proteins believed to be present in the analyzed sample which estimates protein level probabilities based on the integration of Fido and Percolator. We showed that Percolator/Fido efficiently addresses the main problems associated with the estimation of protein probabilities. We have compared and benchmarked our integration (Percolator/Fido) against the state-of-the-art protein inference tool Protein prophet, showing that Percolator/Fido outperforms Protein prophet in terms of sensitivity, specificity and calibration of the results. We also noticed that Percolator/Fido is more robust and less sensible to specific features of the spectra than iProphet/Protein Prophet. Percolator gives a good performance regardless of the scenario, whereas iProphet and Protein prophet are very sensible to certain attributes, and therefore less robust. We noticed that the configuration of the search parameters can affect drastically the quality of the results of iProphet. Attributes such as the size of the dataset, the resolution of the mass tolerance, the way the decoys are generated, the target-decoy ratio as well as some other can affect considerably the performance of iProphet and Protein prophet. We observed that the characteristic and the number of features that iProphet uses to perform the classification are very sensible to certain scenarios and/or configurations.
We believe Percolator/Fido represents a step forward in protein identification for shotgun proteomics experiments.
With regard to the main drawbacks, we have noticed that the computational time is an issue in Percolator/Fido. Percolator/Fido does not perform as well in terms of computational speed when compared to other protein inference tools, especially for big datasets. We have tried to decrease the computational time to the minimum levels but it can still be high for big datasets. For that purpose, we included a set of parameters in Percolator that can decrease the computational time when using appropriately. For example, we do not need to run the grid search twice for the same dataset, if we already know the values of the parameters of the model of Fido, we can pass them to Percolator, and thus the grid search will be skipped saving a considerable amount of computational time. Nevertheless, we are considering some possible solutions to decrease the computation time. The most feasible solution for this problem could be the parallelization of the grid search of the parameters of the model. This would theoretically save computational time.
Another drawback is the number of parameters needed for the model of Fido, we implemented a grid search to estimate them based on the quality of the probabil-

ities. This is obtained by scoring the divergence between empirical and estimated q values and the ROCN curve. This, perhaps, is not the most efficient way to solve this problem. The ROC curve assumes that all the target proteins are true positives which is not necessarily true; some of them are false positive proteins. Additionally, it is good to keep a high correlation between empirical and estimated q values but at protein level the most calibrated results might not be the best results. This might really depend on the dataset and the ratio of decoy proteins with respect to the target proteins. We have considered some nonparametric alternatives to solve this problem, although we have not yet implemented any of them.
We also think that a possible improvement for Percolator/Fido would be to include more information when estimating protein probabilities, such as predictability levels or retention time based predicted proteins. This work is also ongoing in our lab.
To conclude with, we would like to mention that Protein prophet as well as many others protein inference tools, base their models to estimate protein probabilities employing parsimony and protein grouping. The parsimony approach assumes that the solution containing the simplest and smallest set of proteins covering the set of scored peptides, is the best solution. Recent studies have shown that the parsimony approach is less reproducible. Therefore, we concluded that the reproducibility and sensitivity of the results, the addition of extra layers of information and the computational speed, are the most important issues to study in order to improve the estimation of protein level probabilities.

Part IV

Bibliography

Bibliography

[1] Bernhard E. Boser and et al. A training algorithm for optimal margin classifiers. pages 144–152, 1992.

[2] Frank Dellaert College and Frank Dellaert. The expectation maximization algorithm. 2002.

[3] J.K. Eng, A.L. McCormack, and J.R. Yates. An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. *Journal of the American Society for Mass Spectrometry*, 5(11):976–989, 1994.

[4] Seattle Proteome Center Institute for Systems Biology. Proteomics informatic course syllabus, 2010.

[5] Viktor Granholm and Lukas Käll. Quality assessments of peptide-spectrum matches in shotgun proteomics. *Proteomics*, February 2011.

[6] Viktor Granholm, William S. Noble, and Lukas Kall. On Using Samples of Known Protein Content to Assess the Statistical Calibration of Scores Assigned to Peptide-Spectrum Matches in Shotgun Proteomics. *Journal of Proteome Research*, 0(0), March 0000.

[7] L. Kaell, J. D. Storey, M. J. MacCoss, and W. S. Noble. Assigning significance to peptides identified by tandem mass spectrometry using decoy databases. *Journal of Proteome Research*, 7:29+, 2008.

[8] L. Käll, J. D. Storey, and W. S. Noble. Non-parametric estimation of posterior error probabilities associated with peptides identified by tandem mass spectrometry. *Bioinformatics (Oxford, England)*, 24(16), August 2008.

[9] Lukas Kall, Jesse Canterbury, Jason Weston, William S. Noble, and Michael J. MacCoss. Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nature Methods*, 2007.

[10] Lukas Kall, John D Storey, Michael J MacCoss, and William Stafford Noble. Posterior error probabilities and false discovery rates: two sides of the same coin. *J Proteome Res*, 7(1):40–44, 2008.

[11] Andrew Keller, Alexey I. Nesvizhskii, Eugene Kolker, and Ruedi Aebersold. Empirical statistical model to estimate the accuracy of peptide identifications made by MS/MS and database search. *Analytical chemistry*, 74(20):5383–5392, October 2002.

[12] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. pages 1137–1143, 1995.

[13] Olga Vitek. Lukas Kall. An insight into computational and statistical mass spectrometry-based proteomics, 2010.

[14] A.I. Nesvizhskii, A. Keller, E. Kolker, and R. Aebersold. A statistical model for identifying proteins by tandem mass spectrometry. *Anal Chem*, 75(17):4646–58, 2003.

[15] Alexey I. Nesvizhskii. A survey of computational methods and error rate estimation procedures for peptide and protein identification in shotgun proteomics. *Journal of proteomics*, August 2010.

[16] Alexey I. Nesvizhskii and Ruedi Aebersold. Interpretation of shotgun proteomic data: the protein inference problem. *Molecular & cellular proteomics : MCP*, 4(10):1419–1440, October 2005.

[17] Bernard. W. Silverman P.J. Green. *Non-parametric Regression and Generalized Linear Models: A roughness penalty approach.* Chapman and Hall, 1994.

[18] Lukas Reiter, Manfred Claassen, Sabine P. Schrimpf, Marko Jovanovic, Alexander Schmidt, Joachim M. Buhmann, Michael O. Hengartner, and Ruedi Aebersold. Protein Identification False Discovery Rates for Very Large Proteomics Data Sets Generated by Tandem Mass Spectrometry. *Mol Cell Proteomics*, 8(11):2405–2417, November 2009.

[19] O. Serang, M.J. MacCoss, and W.S. Noble. Efficient marginalization to compute protein posterior probabilities from shotgun mass spectrometry data. *J Proteome Res*, 9(10):5346–57, 2010.

[20] Oliver Serang. *A practically efficient graph-theoretic approach to protein identification in mass spectrometry.* Dissertation, University of Washington, 2011.

[21] D. Shteynberg, E.W. Deutsch, H. Lam, J.K. Eng, Z. Sun, N. Tasman, L. Mendoza, R.L. Moritz, R. Aebersold, and A.I. Nesvizhskii. iprophet: Multi-level integrative analysis of shotgun proteomic data improves peptide and protein identification rates and error estimates. *Mol Cell Proteomics*, 2011.

[22] Marina Spivak, Jason Weston, Léon Bottou, Lukas Käll, and William Stafford S. Noble. Improvements to the percolator algorithm for peptide identification from shotgun proteomics data sets. *Journal of proteome research*, 8(7):3737–3745, July 2009.

[23] Hanno Steen and Matthias Mann. The abc's (and xyz's) of peptide sequencing. *Nature Reviews Molecular Cell Biology*, 5(9):699–711, September 2004.

[24] John D. Storey. A direct approach to false discovery rates. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(3):479–498, August 2002.

[25] John D. Storey. The positive false discovery rate: a bayesian interpretation and the q -value. *The Annals of Statistics*, 31(6):2013–2035, December 2003.

[26] Larry Wasserman. *All of statistics : a concise course in statistical inference.* Springer, 2006.

Part V

Appendices

Appendix A

Parameters for protein probability estimation in Percolator

This is a list of the parameters that we have included in our integration of Fido with Percolator.

Flag	Function
-A –protein	activates the estimation of protein probabilities
-a –alpha	fixes the α value of the model of Fido. If this parameter is given and valid, the grid search of this parameter will not be done, thus we will save computational time.
-b –beta	fixes the β value of the model of Fido. If this parameter is given and valid, the grid search of this parameter will not be done, thus we will save computational time.
-G –gamma	fixes the γ value of the model of Fido. If this parameter is given and valid, the grid search of this parameter will not be done, thus we will save computational time.
-g –allow-protein-group	when activated, the calculation of the empirical and estimated q values will be done. Proteins of the same group will be counted as one, therefore the q value will be the same for all the proteins of the group. Proteins are grouped by their PEP prior to the estimation of the q values. If this option is activated the q value for all the proteins belonging to a group will be the same and will be calculated according to the number of target proteins and their respective PEPs.
-I –protein-level-pi0	when activated, the empirical q values will be adjusted by the ratio π_o. The adjustment will only be valid for the empirical q values of the final set of proteins that will be outputted. It will not be used to adjust the empirical q values used to estimate the parameters of the model. This option cannot be used in combination with the option -Q.

-q –empirical-protein-q	when activated, includes the empirical q values in the xml formatted output file of Percolator for every protein.
-N –group-proteins	when activated, it allows the grouping of proteins with similar connectivity, for example if proteins P1 and P2 have the same peptides matching both of them, P1 and P2 can be grouped as one single protein P3. This may increase the speed of the estimation of the probabilities.
-E –no-separate-proteins	when activated, it deactivates the partitioning option of Fido, where the protein-peptide graph is partitioned in subgraphs that contain only connected proteins, so the probability can be computed individually for each subgraph saving computational time.
-C –no-prune-proteins	when activated, Fido will not prune peptides with a very low score which means that if a peptide with a very low score is matching two proteins, when we prune the peptide, it will be duplicated to generate two new protein groups
-d –depth	indicate the depth level (range of values) for the grid search of the parameters α, β and γ from 0 highest depth to 3 lowest depth.
-T –reduce-tree	Reduce the tree of proteins in order to estimate α, β and γ faster.
-Q –protein-fdr	when activated, the empirical q values will be adjusted by the protein level FDR estimated with Percolator based on Mayu. This option cannot be used in combination with option -I. When activated, a link to a combined database containing the target and decoy proteins that we used to match the spectra against, has to be submitted with the parameter TD. The readjustment of the empirical q values will affect the empirical q values used in the estimation of the parameters of the model.
-TD –database	indicate where a combined database containing target and decoy proteins that we used to match the spectra against is located. The database has to be in FASTA format and it must contain target and decoy proteins. This option has to be used in combination with the option -Q.
-P –pattern	indicate the text pattern that will identify the decoy proteins in the database. By default the pattern is *random*. This option is important if we use the option -Q and -TD and the decoy proteins present in the combined database are not labeled with the pattern *random*.

Appendix B

Protein, peptide and PSM level plots

In this appendix section we include all the figures that we generated for different datasets at psm, peptide and protein level. We also include tables showing the performance of iProphet compared to Percolator at both psm and peptide level for all the datasets.

Table B.1. Performance of Percolator at peptide level compared to iProphet in both semi-supervised and unsupervised mode at an estimated q value of %10.

Dataset	**post-processor tool**	**target peptides**	**hidden decoy peptides**	**decoy peptides**
Yeast	Percolator	7824	333	1325
Yeast	iProphet semi-supervised	6964	4844	16970
Yeast	iProphet unsupervised	7781	255	-
Streptococcus	Percolator	1050	65	267
Streptococcus	iProphet semi-supervised	1197	1293	4950
Streptococcus	iProphet unsupervised	1061	72	-
Human Leukate	Percolator	1072	42	1877
Human Leukate	iProphet semi-supervised	789	537	498
Human Leukate	iProphet unsupervised	1040	38	-

Table B.2. Performance of Percolator at psm level compared to iProphet in both semi-supervised and unsupervised mode at an estimated q value of %10.

Dataset	**post-processor**	**target psms**	**hidden decoy psms**	**decoy psms**
Yeast	Percolator	13047	534	2297
Yeast	iProphet semi-supervised	7362	4876	17113
Yeast	iProphet unsupervised	12525	364	-
Streptococcus	Percolator	4746	231	963
Streptococcus	iProphet semi-supervised	3118	1482	6085
Streptococcus	iProphet unsupervised	4776	257	-
Human Leukate	Percolator	1388	56	220
Human Leukate	iProphet semi-supervised	809	535	1875
Human Leukate	iProphet unsupervised	1384	54	-

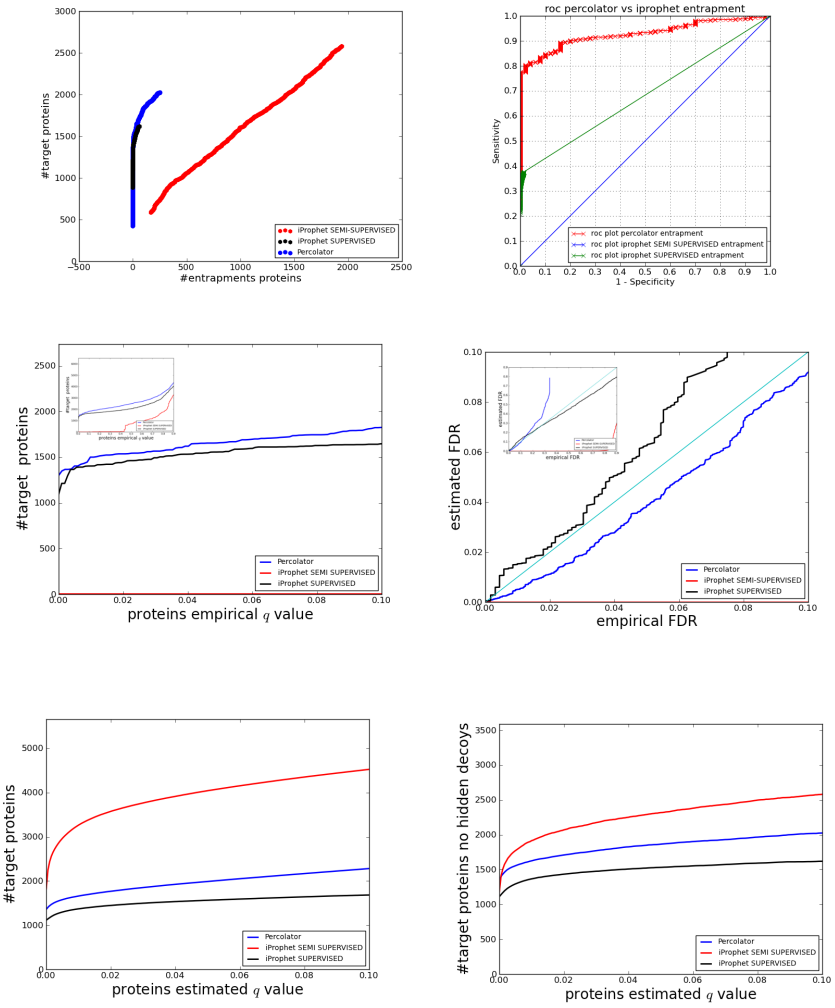

Figure B.1. Yeast dataset (proteins). a) number of target proteins as a function of the number of hidden decoy proteins both obtained at an estimated q value of 10%. b) ROC50 curve comparing the ratio of true positives against the ratio of false positives. c) target proteins as a function of the empirical q values. d) empirical q values vs estimated q values. e) number of target proteins as a function of the estimated q values. f) number of target proteins, excluding hidden decoys as a function of the estimated q values.

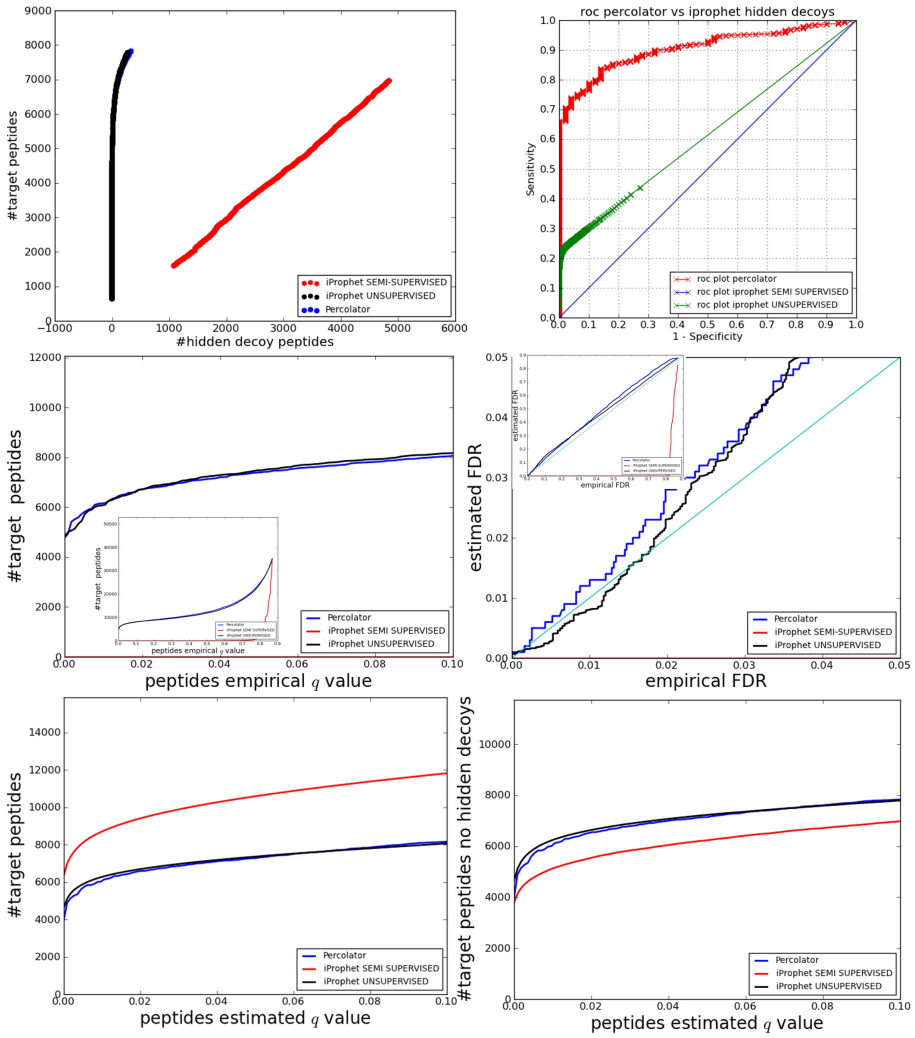

Figure B.2. Yeast dataset (peptides). a) number of target peptides as a function of the number of hidden decoy peptides both obtained at an estimated q value of 10%. b) ROC50 curve comparing the ratio of true positives against the ratio of false positives. c) target peptides as a function of the empirical q values. d) empirical q values vs estimated q values. e) number of target peptides as a function of the estimated q values. f) number of target peptides, excluding hidden decoys as a function of the estimated q values.

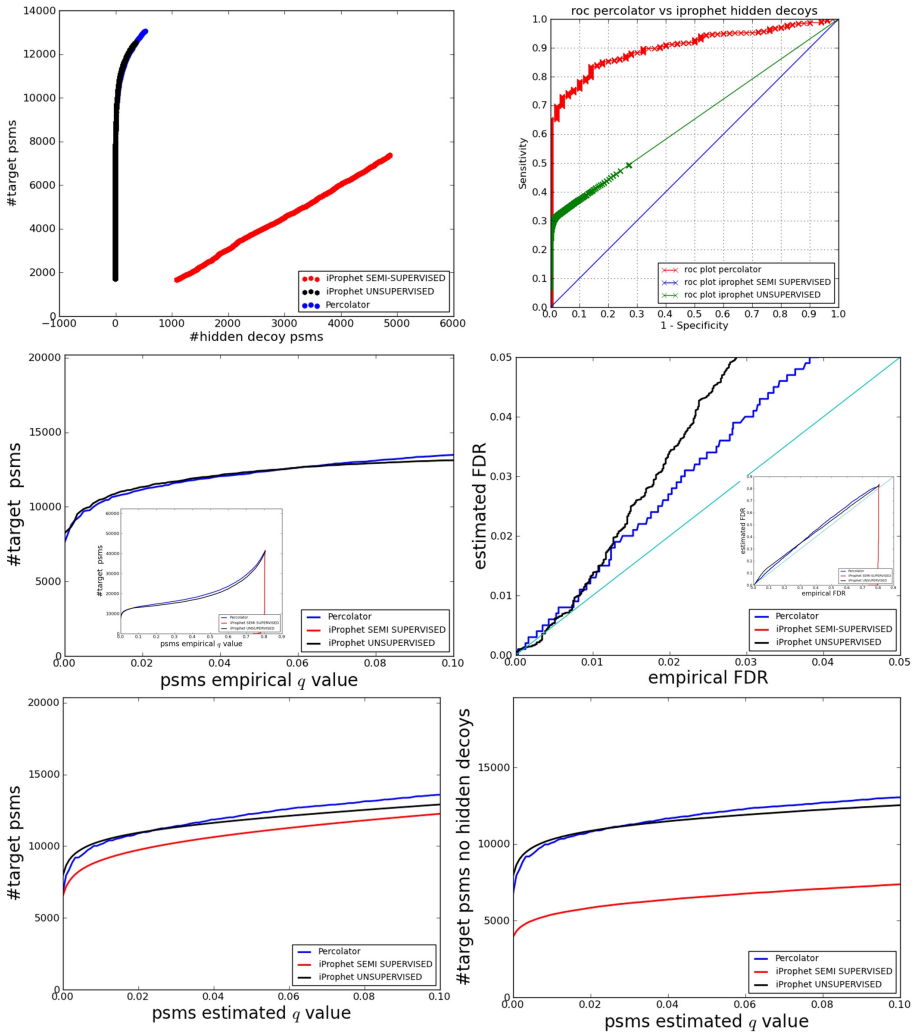

Figure B.3. Yeast dataset (psms). a) number of target psms as a function of the number hidden of decoy PSMs both obtained at an estimated q value of 10%. b) ROC50 curve comparing the ratio of true positives against the ratio of false positives. c) target psms as a function of the empirical q values. d) empirical q values vs estimated q values. e) number of target psms as a function of the estimated q values. f) number of target psms, excluding hidden decoys as a function of the estimated q values.

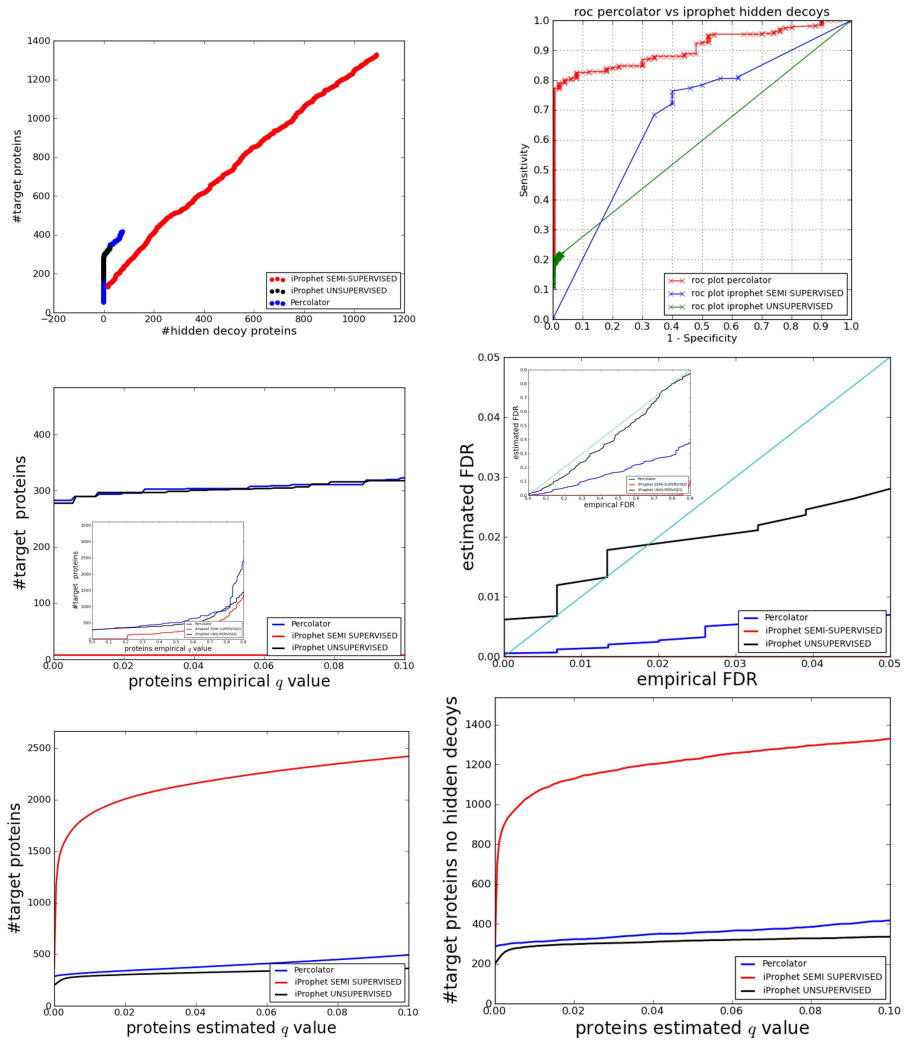

Figure B.4. Streptococcus dataset (proteins). a) number of target proteins as a function of the number of hidden decoy proteins both obtained at an estimated q value of 10%. b) ROC50 curve comparing the ratio of true positives against the ratio of false positives. c) target proteins as a function of the empirical q values. d) empirical q values vs estimated q values. e) number of target proteins as a function of the estimated q values. f) number of target proteins, excluding hidden decoys as a function of the estimated q values.

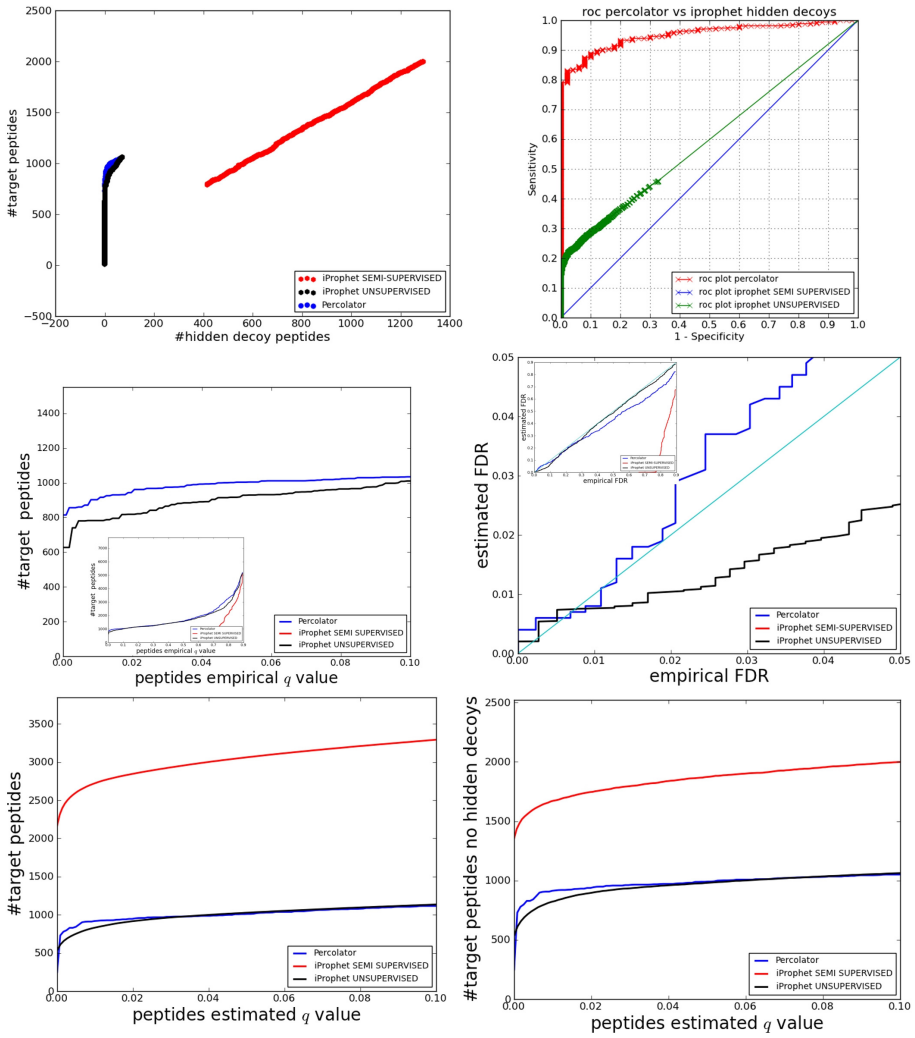

Figure B.5. Streptococcus dataset (peptides). a) number of target peptides as a function of the number of hidden decoy peptides both obtained at an estimated q value of 10%. b) ROC50 curve comparing the ratio of true positives against the ratio of false positives. c) target peptides as a function of the empirical q values. d) empirical q values vs estimated q values. e) number of target peptides as a function of the estimated q values. f) number of target peptides, excluding hidden decoys as a function of the estimated q values.

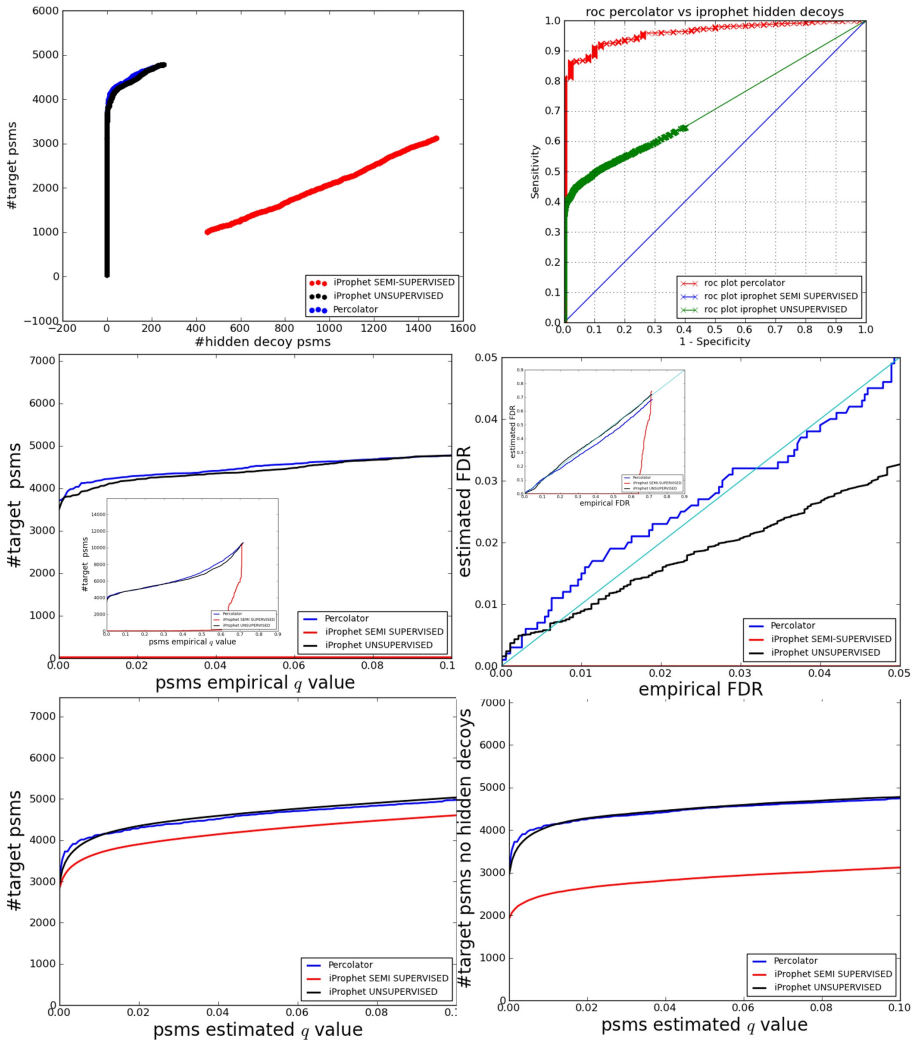

Figure B.6. Streptococcus dataset (psms). a) number of target psms as a function of the number of hidden decoys PSMs both obtained at an estimated q value of 10%. b) ROC50 curve comparing the ratio of true positives against the ratio of false positives. c) target psms as a function of the empirical q values. d) empirical q values vs estimated q values. e) number of target psms as a function of the estimated q values. f) number of target psms, excluding hidden decoys as a function of the estimated q values.

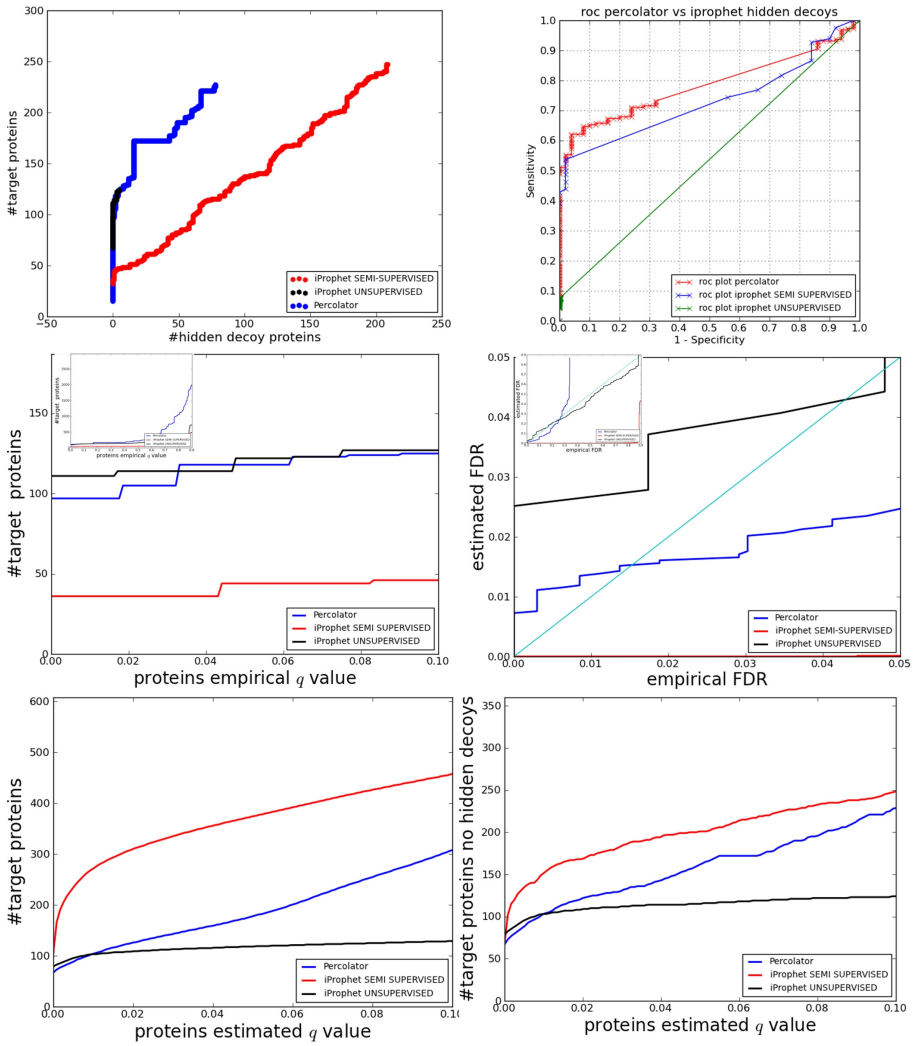

Figure B.7. Human Leukate dataset (proteins). a) number of target proteins as a function of the number of hidden decoy proteins both obtained at estimated q value of 10%. b) ROC50 curve comparing the ratio of true positives against the ratio of false positives. c) target proteins as a function of the empirical q values. d) empirical q values vs estimated q values. e) number of target proteins as a function of the estimated q values. f) number of target proteins, excluding hidden decoys as a function of the estimated q values.

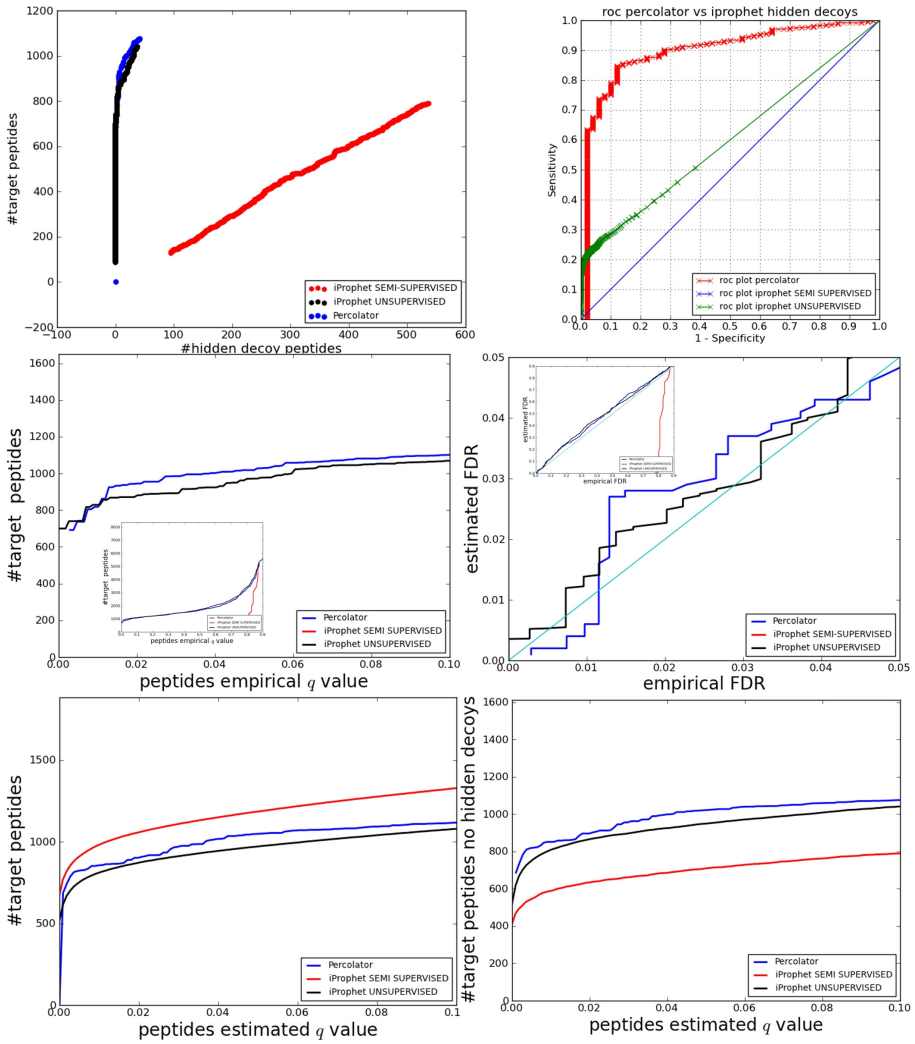

Figure B.8. Human Leukate dataset (peptides). a) number of target peptides as a function of the number of hidden decoy peptides both obtained at an estimated q value of 10%. b) ROC50 curve comparing the ratio of true positives against the ratio of false positives. c) target peptides as a function of the empirical q values. d) empirical q values vs estimated q values. e) number of target peptides as a function of the estimated q values. f) number of target peptides, excluding hidden decoys as a function of the estimated q values.

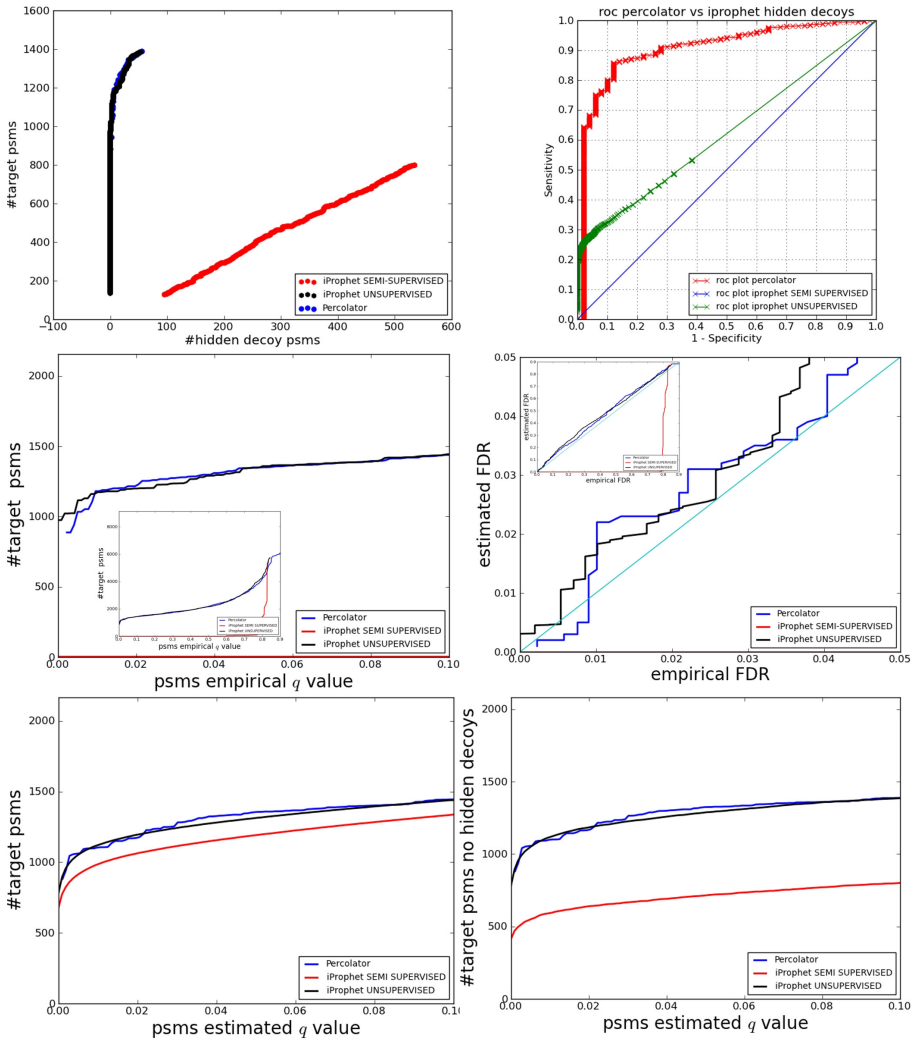

Figure B.9. Human Leukate (psms). a) number of target psms as a function of the number of hidden decoy PSMs both obtained at an estimated q value of 10%. b) ROC50 curve comparing the ratio of true positives against the ratio of false positives. c) target psms as a function of the empirical q values. d) empirical q values vs estimated q values. e) number of target psms as a function of the estimated q values. f) number of target psms, excluding hidden decoys as a function of the estimated q values.

Appendix C

List of figures and Tables

List of Figures

List of Tables

TRITA-CSC-E 2013:018
ISRN-KTH/CSC/E--13/018-SE
ISSN-1653-5715

www.kth.se