Decoy-free protein-level false discovery rate estimation
Ben Teng, Ting Huang, Zengyou He*
School of Software, Dalian University of Technology, Dalian, China.

Associate Editor: Prof. Martin Bishop

ABSTRACT
Motivation: Statistical validation of protein identifications is an important issue in shotgun proteomics. The false discovery rate (FDR) is a powerful statistical tool in evaluating the protein identification result. Several research efforts have been made for FDR estimation at the peptide-level. However, there are still certain drawbacks in the existing FDR estimation methods based on the target-decoy strategy.

Results: In this article, we propose a decoy-free protein-level FDR estimation method. Under the null hypothesis that each candidate protein matches an identified peptide totally at random, we assign statistical significance to protein identifications in terms of the permutation p-value and use these p-values to calculate the FDR. Our method consists of three key steps: (1) generating random bipartite graphs with the same structure; (2) calculating the protein scores on these random graphs and (3) calculating the permutation p-value and final FDR. Since it is very time-consuming or prohibitive to execute the protein inference algorithms for thousands of times in step 2, we first train a linear regression model using the original bipartite graph and identification scores provided by the target inference algorithm. Then we use the learned regression model as a substitute of original protein inference method to predict protein scores on shuffled graphs. We test our method on six public available data sets. The results show that our method is comparable to those state-of-the-art algorithms in terms of estimation accuracy.

Availability: The source code of our algorithm is available at: https://sourceforge.net/projects/plfdr/
Contact: zyhe@dlut.edu.cn

1 INTRODUCTION
Shotgun proteomics is a strategy that is capable of identifying complex protein mixtures by combining high-performance liquid chromatography (LC) and mass spectrometry (MS). In shotgun proteomics, the protein identification procedure has two main steps: peptide identification and protein inference (Nesvizhskii et al., 2007). In peptide identification, we search the experimental MS/MS spectra against a protein sequence database to obtain a set of peptide-spectrum matches (PSMs). In protein inference, we report a set of proteins by assembling peptide identification results (Huang et al., 2012).

Basically, there are two major computational issues in protein identification that have to be solved. On one hand, we need to develop effective and fast identification/inference algorithms at both the peptide-level and the protein-level. On the other hand, controlling the quality of identified peptides and inferred proteins is as important as developing identification algorithms.

In fact, inferred proteins are more biologically relevant than identified peptides in a proteomics experiment. Therefore, it is vital to control the quality of identification results at the protein-level. However, the accurate assessment of statistical significance of protein identifications remains an open question (Spirin et al., 2011).

To date, several research efforts have been made to estimate the protein-level error rate in terms of false discovery rate (FDR). The mainstream approach for FDR estimation is the target-decoy strategy, which searches a target-decoy concatenated database so that the number of false positive protein identifications can be estimated from the number of decoy proteins. For instance, the MAYU method (Reiter et al., 2009) is a typical representative in this category. By adapting the target-decoy strategy to the protein inference task, the MAYU method first assumes that protein identifications containing false positive PSMs are uniformly distributed over the target database. Then, the number of false positive protein identifications is hypergeometrically distributed. As a result, the expected number of false positive protein identifications can be calculated as the probability weighted average. Finally, the protein identification FDR is computed as the ratio of the expected number of false positive protein identifications and the total amount of protein identifications mapping to the target database.

However, this valuable approach has certain drawbacks (Kim et al., 2008). Firstly, searching both the target and the decoy database will certainly double the running time in the protein identification process. Secondly, the FDR estimation result may be unstable since we usually use only one decoy database with the same size of target database. Finally, the protein FDR value is calculated according to the distribution of decoy peptides across different proteins, making it possible to propagate errors at the peptide-level to the protein-level in a non-trivial manner.

In this paper, we propose a new method for estimating the FDR at the protein-level without searching the decoy database. Our method uses random permutation to assess the statistical significance of each protein in terms of p-value and then calculates the final FDR. First of all, the input for the protein inference problem can be modeled as a bipartite graph: the left is a set of identified peptides and the right is the set of candidate proteins with at least one matched peptide (Huang and He, 2012). From this bipartite graph, a protein inference algorithm calculates the probability or score for each protein. The null hypothesis in our method is that each candidate protein matches an identified peptide totally at random. Under this null hypothesis,
we first create multiple random bipartite graphs with the same set of peptides and proteins. Each random bipartite graph has the same structure as the original one meanwhile, i.e. each protein (peptide) is connected to the same number of peptides (proteins). Then, we run the same protein inference algorithm on these random bipartite graphs, and check if the score of each protein is significantly different on the real graph than on the randomized graphs. That is, we calculate the permutation $p$-value of one protein as the percent of random graphs that produce a larger score than its original score. Finally, we sort the proteins according to their $p$-values and calculate the FDR at different thresholds with permutation $p$-values as input using the method in Storey and Tibshirani (2003).

Our method has three key steps: (1) generating random bipartite graphs with the same structure; (2) calculating the protein scores on these random graphs and (3) calculating the permutation $p$-value and FDR. Among these steps, it is relatively easy to perform the first step and the third step. However, it is very time-consuming or prohibitive to execute some protein inference algorithms for thousands of times to fulfill step 2. To address this issue, we first train a linear regression model using the original bipartite graph and identification scores given by the target inference algorithm. Then, we use the learned regression model as a substitute of original protein inference method to predict protein scores on shuffled graphs.

Experimental results on several real proteomics data sets show that our method is effective in FDR calculation. Overall, the salient features of the work described in this article can be summarized as follows:

- It can calculate the FDR without using a decoy database.
- It can be applied to evaluate the protein identification results from any algorithm that outputs protein probabilities/scores.

The rest of this article is organized as follows. In Section 2, we describe our method in detail. Section 3 presents the experimental results and Section 4 concludes the article.

## 2 METHODS

In our method, we use permutation test to calculate the $p$-value of each protein and then use these $p$-values to calculate the final FDR. Therefore, our method mainly consists of the following steps:

1. Generating multiple random bipartite graphs with the same structure;
2. Building a linear regression model with the original bipartite graph and protein identification scores reported by the target protein inference algorithm;
3. Predicting the protein scores on random graphs using the trained linear regression model;
4. Calculating the permutation $p$-value and FDR.

Figure 1 gives an overview of this method. In the following, we will explain each step in detail. Since step 2 is tightly correlated with step 3, we will elaborate them in the same sub-section.

### 2.1 Generating random bipartite graphs

If we wish to test the null hypothesis that each candidate protein hits an identified peptide at random, it is important to construct random graphs with the same structure as the original one. It ensures that the shuffled graphs are comparable to the original graph, whereby each protein (peptide) is connected to the same number of peptides (proteins).

Given $m$ candidate proteins and $n$ identified peptides, the bipartite graph $G(L, R, E)$ models the input for the protein inference problem. The vertex $j \in L$ corresponds to the $j$th peptide and $|L| = n$. Similarly, the vertex $i \in R$ represents the $i$th candidate protein and $|R| = m$. Then, there exists an edge $(j, i) \in E$ if the $i$th candidate protein contains the $j$th peptide. We generate a random bipartite graph by the following steps:

1. Record the degree of each protein/peptide (the number of peptides/proteins that each protein/peptide is connected with) in the original bipartite graph.
2. Randomly select one protein $i$ and one peptide $j$ whose degrees are both more than 0, and ensure that there is no edge between them in the current random graph.
3. Connect protein $i$ and peptide $j$ and decrease the degrees of protein $i$ and peptide $j$ by 1.
4. If there are no such protein and peptide, and the number of edges in the current random graph is less than that in the original graph, select one edge $(j_k, i_k)$ in the random graph such that there is no edge between $j_k$ and $i$ in the random graph. Remove edge $(j_k, i_k)$ and add an edge $(j_k, i)$. Increase the degree of protein $i$, by 1 and decrease the degree of protein $i$ by 1.
5. Do step 2 to 4 until the generated random graph has the same number of edges with the original graph.
Algorithm 1 shows a straightforward implementation of this generation method.

**Algorithm 1 Random-Graph-Generation**

**Input:** Original graph \( G(L, R, E) \)

**Output:** Random graph \( G'(L, R, E') \) with the same structure as \( G \)

1: \( E' \leftarrow \text{null} \)
2: Initialize the degree of each protein and each peptide
3: while \(|E'| < |E|\) do
4: randomly select one protein \( i \) with \( \text{Degree}(i) \geq 1 \)
5: if all \( j \) with \( \text{Degree}(j) \geq 1 \) make \( (j, i) \in E' \) then
6: select \((j_e, i_e) \in E'\) such that \((j_e, i_e) \notin E'\)
7: remove \((j_e, i_e)\) from \(E'\) and add \((j_e, i)\) to \(E'\)
8: \( \text{Degree}(i) \leftarrow \text{Degree}(i) + 1 \)
9: continue
10: end if
11: randomly select one peptide \( j \) with \( \text{Degree}(j) \geq 1 \)
12: while \((j, i) \in E'\) do
13: re-select one peptide \( j \) with \( \text{Degree}(j) \geq 1 \)
14: end while
15: \( \text{Degree}(i) \leftarrow \text{Degree}(i) - 1 \)
16: \( E' \leftarrow E' \cup \{(j, i)\} \)
17: end while
18: return \( G'(L, R, E') \)

The most important advantage of our method for generating the random graph is that it doesn’t require any parameter. There exist some other algorithms that can generate random graphs with the same structure as the original graph. For example, the method in Gionis et al. (2007) is more simple and easier to understand than our method because only local swaps are performed in generating a random graph. However, this method needs to specify the number of local swaps and it is difficult to decide which parameter value is the best.

To make the protein inference results comparable, the graph randomization procedure should keep all the features of the original graph unchanged except the matches between proteins and peptides. The features of a graph include the number of nodes, the number of edges, the number of connected components and so on. Since our method only ensures that the individual degree of each node (protein or peptide) is preserved, one may wonder if the number of connected components will change substantially. In the supplemental Figure S1, we plot the distribution of the number of connected components in random graphs. It shows that the difference between the random graph and the real graph with respect to the number of connected components is very small. Therefore, protein inference is comparable between the original graph and random graphs although the numbers of connected components of these graphs are not exactly the same.

### 2.2 Predicting the protein scores on random graphs

It is very time-consuming and inconvenient to execute some protein inference algorithms for thousands of times. Therefore, we build a linear regression model to predict the protein scores reported by these algorithms.

We use a response variable \( y_i \in R \) to denote the identification score of protein \( i \) and a predictor vector \( x_i \in R^m \) to represent the set of associated peptide probabilities or scores. The element \( x_{ij} \) in vector \( x_i \) is constructed by the following way:

\[
x_{ij} = \begin{cases} 
\text{probability of peptide } j, & (j, i) \in E \\
0, & (j, i) \notin E 
\end{cases}
\]

That is, if there is an edge between protein \( i \) and peptide \( j \) in the bipartite graph, the probability of peptide \( j \) will be used as the corresponding predictor value. Since there are \( m \) candidate proteins, we have \( m \) observation pairs \((x_i, y_i)\).

Essentially, all existing protein inference algorithms use the assumption that each protein matches an identified peptide by chance is true, then there is no significant difference between the score of each protein in the original graph and those calculated from the random graphs. Therefore, we can calculate the protein scores generated from this algorithm. From a machine learning perspective, it is possible to “learn” a similar model that can be used to perform protein inference in the future. As a result, we can apply our method to simulate the identification results of any protein inference algorithm without knowing its algorithmic details even when its executable program is not available to us. Based on this motivation, here we build a linear regression model to realize this goal. In this approach, we search a coefficient vector \( \beta \) to minimize the residual sum of squares:

\[
\min_{\beta} f(\beta) = \min_{\beta} \sum_{i=1}^{m} (y_i - x_i^T \beta)^2.
\]

For protein \( i \), when we have a new peptide contribution vector \( \tilde{x}_i \), with the estimated coefficient vector \( \hat{\beta} \), we can easily use the linear model to get the identification score \( \tilde{y}_i \):

\[
\tilde{y}_i = \tilde{x}_i^T \hat{\beta}.
\]

### 2.3 Calculating the permutation \( p \)-value and FDR

After obtaining the protein scores calculated on these random graphs, we can compute the permutation \( p \)-value and FDR.

If our null hypothesis that each protein matches an identified peptide by chance is true, then there is no significant difference between the score of each protein in the original graph and those calculated from the random graphs. Therefore, we can calculate the permutation \( p \)-value of one protein as the percent of random graphs that produce a larger score than its score generated on the original graph. More precisely, the \( p \)-value of the \( i \)th protein is computed as:

\[
p_i = \frac{\# \{ y_i' > y_i \} }{K},
\]

where \( y_i \) is the original score of the \( i \)th protein and \( y_i' \) is the score of the \( i \)th protein produced from the \( r \)th random graph, \( \# \{ y_i' > y_i \} \) denotes the number of random graphs on which protein \( i \) has a larger score than \( y_i \) and \( K \) is the total number of random bipartite graphs.

Based on these \( p \)-values, we can calculate the FDR and the pFDR value using the method described in Storey and Tibshirani (2003):

\[
\tilde{FDR}(\gamma) = \frac{\pi_0 \gamma}{Pr(P \leq \gamma)},
\]

\[
pFDR(\gamma) = \frac{\pi_0 \gamma}{Pr(P \leq \gamma)(1 - (1 - \pi_0)m)},
\]

where \( \gamma \) denotes the rejection threshold and \( \pi_0 \) is the proportion of false positive. \( Pr(P \leq \gamma) \) represents the probability that a \( p \)-value
pFDR (positive False Discovery Rate) is a modified version of the FDR. As $\gamma$ becomes small, FDR is driven by the fact that the rejection threshold is small rather than the fact that the ‘rate that discoveries are false’ is small. In contrast, pFDR can avoid this disadvantage. Therefore, in this article, we use pFDR as the final estimated value.

We estimate $\hat{\pi}_0$ by the following equation:

$$\hat{\pi}_0(\lambda) = \frac{\# \{ p_i > \lambda \}}{(1 - \lambda)^m},$$

where $p_i$ is the $p$-value of each candidate protein, $\lambda$ is a parameter and $\# \{ p_i > \lambda \}$ denotes the number of proteins whose $p$-values are larger than $\lambda$. And we use the method in Storey (2002) to pick $\lambda$.

A natural estimate of $P_r(P \leq \gamma)$ is:

$$\hat{P}_r(P \leq \gamma) = \frac{\# \{ p_i \leq \gamma \}}{m} = \frac{R(\gamma)}{m},$$

where $R(\gamma) = \# \{ p_i \leq \gamma \}$ represents the number of candidate proteins with $p$-value $\leq \gamma$.

With the estimated $\hat{\pi}_0(\lambda)$ and $\hat{P}_r(P \leq \gamma)$, we can easily get the pFDR value at a given threshold $\gamma$.

3 EXPERIMENT

3.1 Data sets

In our experiments, we use six public available data sets: 18 mixtures (Klimmek et al., 2008), Sigma49 (Tabb et al., 2007), Yeast (Ramakrishnan et al., 2009a), DME (Brunner et al., 2007), HumanMD (Ramakrishnan et al., 2009b) and HumanEKC (Ramakrishnan et al., 2009a). The first three data sets have the ground-truth protein reference sets and another three have no such reference sets. The details of these data sets are described in the supplementary Table S1 and S2.

We use XTandem (David and Cottrell, 2004) as the peptide identification method. Since we compare our method with MAYU (Reiter et al., 2009), all MS/MS data are searched against both target and decoy protein databases. The decoy database is generated using the Trans-Proteomic Pipeline (TPP) v4.6 software. In reality, it is not necessary to search a decoy database when using our method. During the database search, we use default search parameters and accept the peptides reported by PepSubmitted 2015-10-14 by guest on July 8, 2015 http://bioinformatics.oxfordjournals.org/DPIIPIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIIPIIII Downloaded from http://bioinformatics.oxfordjournals.org/ by guest on July 8, 2015

4 CONCLUSION

In this article, we propose a novel protein-level FDR estimation method. We assume that each candidate protein matches an identified peptide totally at random. Then we use random permutation for assessing the statistical significance of each protein.
Fig. 2: Performance on 18 mixtures, Sigma49 and Yeast when ProteinLP (upper) and ProteinProphet (lower) are used as the target protein inference algorithm, respectively. The difference with the benchmark $= |\text{the estimated FDR} - \text{true FDR}|$. For each threshold, the smaller the difference, the better the performance. For 18mixtures, 18 standard proteins and 15 contaminants are marked as the ground-truth. For Sigma49, all accessions in the ABRF2007 BIC Final Correct Proteins List are used as the ground-truth.

Table 2. The running time of our method and MAYU on six data sets. Note that MAYU can only report the FDR value at one threshold in each run, whereas our method can estimate a series of FDR values at all the thresholds. The execution time listed here for MAYU is the average running time over different thresholds. The number of random graphs in our method is fixed to be 10000.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>MAYU</th>
<th>Our method (ProteinLP)</th>
<th>Our method (ProteinProphet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>18mixtures</td>
<td>3 s</td>
<td>2 s</td>
<td>2 s</td>
</tr>
<tr>
<td>Sigma 49</td>
<td>30 s</td>
<td>2 s</td>
<td>2 s</td>
</tr>
<tr>
<td>Yeast</td>
<td>12 s</td>
<td>106 s</td>
<td>90 s</td>
</tr>
<tr>
<td>HumanEKC</td>
<td>41 s</td>
<td>35 s</td>
<td>35 s</td>
</tr>
<tr>
<td>DME</td>
<td>35 s</td>
<td>30 s</td>
<td>30 s</td>
</tr>
<tr>
<td>HumanMD</td>
<td>41 s</td>
<td>13 s</td>
<td>13 s</td>
</tr>
</tbody>
</table>

Table 1. Real $\pi_0$ and the estimated $\hat{\pi}_0$ on six data sets. $\hat{\pi}_{0,1}$ and $\hat{\pi}_{0,2}$ are estimated when ProteinLP and ProteinProphet are used as the target protein inference algorithm, respectively.

<table>
<thead>
<tr>
<th>Data sets</th>
<th>real $\pi_0$</th>
<th>estimated $\hat{\pi}_{0,1}$</th>
<th>estimated $\hat{\pi}_{0,2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>18mixtures</td>
<td>0.698</td>
<td>0.024</td>
<td>0.090</td>
</tr>
<tr>
<td>Sigma 49</td>
<td>0.333</td>
<td>0.068</td>
<td>0.048</td>
</tr>
<tr>
<td>Yeast</td>
<td>0.331</td>
<td>0.248</td>
<td>0.196</td>
</tr>
<tr>
<td>DME</td>
<td>0.308</td>
<td>0.195</td>
<td>0.222</td>
</tr>
<tr>
<td>HumanEKC</td>
<td>0.232</td>
<td>0.202</td>
<td>0.180</td>
</tr>
<tr>
<td>HumanMD</td>
<td>0.063</td>
<td>0.177</td>
<td>0.121</td>
</tr>
</tbody>
</table>

in terms of $p$-value and calculate the final FDR. The main advantage is that our method can calculate FDR without searching a decoy database. Experimental results on six proteomics data sets demonstrate the superiority of our method.

In the future work, we will extend our method to validate the inference results of the algorithms that do not report protein probabilities or scores. We plan to train a logistic regression model, which can assign a corresponding probability to each reported protein. Meanwhile, since $\pi_0$ can affect the FDR estimation results significantly, we will find more accurate $\pi_0$ estimation method in the future.
Fig. 3: Performance on DME, HumanMD and HumanEKC when ProteinLP (upper) and ProteinProphet (lower) are used as the target protein inference algorithm, respectively.

ACKNOWLEDGEMENT

This work was supported by the Natural Science Foundation of China under Grant No. 61003176.

REFERENCES


Gionis, A. et al. (2007) Assessing data mining results via swap randomization. ACM Transactions on Knowledge Discovery from Data, 1 (3).


Fig. 4: The influence of different numbers of the random graphs on three data sets with reference sets.