Sequence analysis

Probability-based pattern recognition and statistical framework for randomization: modeling tandem mass spectrum/peptide sequence false match frequencies

Jian Feng¹, Daniel Q. Naiman¹ and Bret Cooper²,∗

¹Department of Applied Mathematics and Statistics, The Johns Hopkins University, Baltimore, and ²Soybean Genomics and Improvement Laboratory, USDA-ARS, Beltsville, Maryland, USA

Received on March 6, 2007; revised on April 9, 2007; accepted on May 9, 2007
Advance Access publication May 17, 2007
Associate Editor: Limsoon Wong

ABSTRACT

Motivation: In proteomics, reverse database searching is used to control the false match frequency for tandem mass spectrum/peptide sequence matches, but reversal creates sequences devoid of patterns that usually challenge database-search software.

Results: We designed an unsupervised pattern recognition algorithm for detecting patterns with various lengths from large sequence datasets. The patterns found in a protein sequence database were used to create decoy databases using a Monte Carlo sampling algorithm. Searching these decoy databases led to the prediction of false positive rates for spectrum/peptide sequence matches. We show examples where this method, independent of instrumentation, database-search software and samples, provides better estimation of false positive identification rates than a prevailing reverse database searching method. The pattern detection algorithm can also be used to analyze sequences for other purposes in biology or crypology.

Availability: On request from the authors.

Contact: Bret.Cooper@ars.usda.gov

Supplementary information: http://bioinformatics.psb.ugent.be/

1 INTRODUCTION

Tandem mass spectrometry (MS/MS) is a basic method used to identify peptides (Hunt et al., 1986). A peptide tandem mass spectrum reflects the amino acid sequence of the peptide which can be used to indicate the presence of a protein in the starting sample (Yates et al., 1995). Although tandem mass spectra can be interpreted by hand, most researchers rely on software to deduce the amino acid sequence information. Several types exist, such as those that scan for mass differentials corresponding to amino acid losses (Tabb et al., 2003) (Liska et al., 2005). More prevalent are algorithms such as Sequest (Eng et al., 1994) and Mascot (Perkins et al., 1999) that compare a spectrum to hypothetical spectra derived from a protein sequence database. Scores indicate match confidence and output is a peptide sequence from the database.

The various algorithms that perform spectrum/peptide sequence matching can make false or incorrect matches. Sample variation, limited instrument mass resolution, low fragment ionization and unpredicted posttranslational or chemical modification of the peptides create conditions whereby observed tandem mass spectra may deviate from hypothetical spectra (Nesvizhskii and Aebersold, 2005; Nesvizhskii et al., 2006). As a result, a match can be made to a peptide sequence other than the correct one. Furthermore, as large-sized databases with more protein sequences are used to detect a wide variety of possible candidate proteins in a sample, the chance for making false matches increases (Cargile et al., 2004; Resing et al., 2004).

Good spectrum/peptide sequence matching is dependent on factors indicating a quality match versus the chance that the match is false. Consequently, it is important to control the false positive rates; else there is little indication which matches are true, even if match scores are high. Some of the first researchers to attempt to measure the false positive rates did so by searching against a version of the database with reversed protein sequences (Higdon et al., 2005; Peng et al., 2003; Qian et al., 2005). The rationale was that any match made to a sequence from the reverse database would be false since none of those sequences existed in nature, except for the rare palindromes. The information was then used to gauge the false discovery rate and to subsequently set spectrum/peptide sequence match scores at a threshold whereby only a small percentage of the matches would be false. This is now a standard practice and one of the rigorous measurements imposed on sequences inferred from any database-search algorithm. It also provides a common metric that has been used to gauge scores produced by different algorithms (Kapp et al., 2005; Tabb et al., 2007).

Generally, a reverse database is a logical decoy. It has the same number of protein records, the same number of amino acids, and a similar number of candidate peptides (based on molecular weight) as the original forward sequence database. However, reversal removes most small biologically relevant peptides that scan for mass differentials corresponding to amino acid losses.

*To whom correspondence should be addressed.
sequences that frequently appear in nature whose presence in the forward database can naturally lead to false matches. As a result, a reverse database may be too artificial for modeling. We hypothesize that the reverse database search misestimates false positive rates.

Here, we present a new statistical algorithm that detects patterns in sequence strings found in written forms of alphabetic languages and in DNA or protein sequences. The patterns found in protein sequence databases can be used by a Monte Carlo sampling algorithm to populate a decoy database comprising the same number of records with the same amino acid frequency, and similar number of candidate peptides. By comparing searches made against our experimental decoy databases containing amino acid patterns randomly dispersed at a frequency similar to the forward database, we show that reverse database searching has generally been inaccurate for modeling false matches.

2 METHODS

2.1 Definitions and assumptions

The term alphabet refers to a finite set of characters or letters denoted by $\mathcal{Z}$. As examples, the English alphabet contains 26 letters while the protein alphabet contains 20 letters for 20 amino acids. The term pattern or word refers to finite sequence $w$ of characters from that alphabet. A word of English text can be seen as a pattern. A finite collection of patterns makes up a dictionary denoted as $D$.

Streams of dictionary words picked at random from the dictionary one at a time are concatenated to form a stream of characters. Each new word $w$ is selected with some probability $p_w$ independently of the words that have been selected thus far. In statistical parlance, word selection is a series of multinomial trials. Sequence means a stream concatenated using the finite number of full dictionary words. A string is a part of a stream of full patterns. A string does not have to start with or end in a full pattern.

Imagine that we observe the stream of characters, we do not know what words produced that particular sequence, as the words are not delimited in any way, and there can be ambiguities in how the stream is decomposed into words. Thus, there is a ‘hidden’ unobserved process in which the words pop out of a black box and get concatenated to the stream, while on the other hand there is the ‘observed’ process which is simply the resulting stream of characters.

To illustrate with a concrete example, suppose the dictionary is $D = \{ \text{‘the’}, \text{‘red’}, \text{‘ark’}, \text{‘there’}, \text{‘dark’} \}$ and ‘theredark’ is a sequence. Focusing on the hidden word generation process, there are two ways to obtain the character stream of ‘theredark’. One of these corresponds to three consecutive words entering the character stream that are ‘the’, ‘red’ and ‘ark’. The other way to obtain ‘theredark’ corresponds to streaming of consecutive words ‘there’ and ‘dark’.

2.2 The optimal decomposition of a sequence

If the words ‘the’, ‘red’, ‘ark’, ‘there’ and ‘dark’ in $D$ have the probabilities 0.3, 0.3, 0.3, 0.05 and 0.05, respectively, then the probability of consecutive words being ‘the’, ‘red’ and ‘ark’ as in the first decomposition of ‘theredark’ is $(0.3)^3 = 0.027$. The probability of two consecutive words being ‘there’ and ‘dark’ which corresponds to the second decomposition is $(0.05)^2 = 0.0025$. Therefore, given the words and probability distribution in $D$, the first decomposition is optimal because it has the highest probability.

Enumeration of all the possible decompositions of a sequence is usually not applicable when a dataset consists of a large amount of patterns/words or when the sequences are long, such as protein sequences. In order to efficiently find the optimal decomposition of a sequence into all possible patterns, a generalized dynamic algorithm is applied as follows:

Suppose sequence, $S = y_1 y_2 \ldots y_n$, where $y_i \in \mathcal{Z}$. A new numerical sequence $F = f_1 f_2 \ldots f_n$, where $f_i$ for every $i$ is the probability that the sequence $y_1 y_2 \ldots y_i$ can be observed using the optimal decomposition. The initial values of $f_i$ are set to be 0, and $f_0$ is defined as 1. In general, if $f_i$ is not 0, then $y_1 y_2 \ldots y_i$ can be segmented into full patterns, which means the sequence starts with a full pattern and also ends with a full pattern, and thus a new pattern starts at position $i+1$. Suppose we are at the $i+1$st step, which means we are checking $y_{i+1}$. If $f_i$ is 0, we continue to the next step and check $y_{i+2}$. Since $y_{i+1}$ cannot be the first letter of a pattern in this case. Otherwise, $D$ is checked if string $y_1 y_2 y_3 \ldots y_i$ is a pattern for some $j$ such that $i+1 \leq j \leq n$. If it is, $f_j$ is compared with $f_i \cdot P(w_{i+1} w_{i+2} \ldots w_n)$. If $f_j$ is smaller, which implies a better decomposition is found, $f_j$ gets the new value $f_j \cdot P(w_{i+1} w_{i+2} \ldots w_n)$. The step $i+1$ is done when all are checked. Then we go to the step $i+2$ to check the next position. More details are in Supplementary Material. In the algorithm, a log function is applied to $f_j$ to maintain as many significant digits as possible.

2.3 The probability of observing a string

For a sequence $S = y_1 y_2 \ldots y_n$, we have the following denotations:

\[
\begin{align*}
  y_1 y_2 y_3 \ldots y_{i-1} y_i y_{i+1} \ldots y_n \\
  y_1 y_2 y_3 \ldots y_{i-1} y_i y_{i+1} \ldots y_n
\end{align*}
\]

where, $y_{i}$ is the letter (an entry of the alphabet $\mathcal{Z}$) at the $i$-th position of the sequence, $x_i$ represents the pattern in $D$ that the letter $y_i$ came from and $u_{i}$ is the position of letter $y_i$ in $x_i$. Both $x_i$ and $u_{i}$ are random variables. Suppose we randomly choose a position $T$ of the sequence $s$, where the probability distribution of $T$ is uniform, i.e. any two positions have the same chance to be chosen. Then, the goal is to calculate the probability that a given $m$-letter long string $t$ can be observed starting at the position $T$ when $n$ goes to infinity.

2.3.1 Probability that $x_T$ is pattern $w$

When randomly choosing a letter in a long sequence, the probability that the letter belongs to pattern $w$ is proportional to the product of the probability and the length of $w$. A normalization factor $NF$ is defined as \(\sum_{w} p_w \cdot |w|\), where $|w|$ is the length of the pattern $w$. Thus,

\[
\text{Prob}(x_T = w) = \frac{p_w \cdot |w|}{NF}
\]

If we assume \(\text{Prob}(x_T = |X_{TF} = w|) = \text{Prob}(x_T = |X_{TF} = w|) = \ldots = \text{Prob}(x_T = |X_{TF} = w|)\), then for a given integer $T(1 \leq T \leq |w|)$,

\[
\text{Prob}(x_T = w, u_T = T) = \frac{p_w \cdot |w|}{NF} \times \frac{1}{|w|} = \frac{p_w}{NF}
\]

2.3.2 Spelling of $t$ A spelling $r$ of $t$ is a sequence of patterns, i.e.

\[r = (\text{substring of } w_1) w_2 \ldots (\text{substring of } w_k)\]

where, a substring can be the whole string but cannot be an empty string. There is a position of $w_1$ at which the string $r$ begins. This position is $r_1$ and $S_r$ is the set of spellings of $t$. 

Downloaded from https://academic.oup.com/bioinformatics/article-abstract/23/17/2210/260201 by guest on 12 February 2019
2.3.3 Probability that a spelling \( r \) occurs at position \( T \) Using Equation (3), we conclude:

\[
\text{Prob}(\text{spelling } r \text{ occurs at position } T) = \text{Prob}(w_1 \text{ starts at position } T - r + 1),
\]
\[
\ldots
\]
\[
= \text{Prob}(w_1 \text{ starts at position } T - r + 1 + |w_1|) + \text{Prob}(w_2 \text{ starts at position } T - r + 1 + |w_1| + |w_2|) + \ldots
\]
\[
= \text{Prob}(w_1 \text{ starts at position } T - r + 1) + \frac{1}{NF} \prod_{j=1}^{k} p_{w_j}
\]

2.3.4 Probability that the string \( t \) starts at position \( T \) By summing over all spellings in \( S_t \) that may generate the string \( t \),

\[
\text{Prob}(t \text{ starts at position } T) = \sum_{r \in S_t} \text{Prob}(\text{spelling } r \text{ starts at position } T)
\]
\[
= \frac{1}{NF} \sum_{r \in S_t} \prod_{i=1}^{k} p_{w_i}
\]

where \( w_1w_2 \ldots w_k \) are the words of the spelling \( r \). It should be noted that the number \( k \) here depends on spelling \( r \) in \( S_t \).

2.4 The mean and variance of the numbers of appearance of a string in the sequence dataset

We assume that sequences are much longer than the lengths of the patterns and chosen strings. As a rough approximation, the numbers of appearance for each particular string \( t \) in the sequence dataset can be seen as a random variable that satisfies binomial distribution with mean \( \text{Prob}(T=t) \sum_{i=1}^{L} \text{length}(S_i) - M + 1 \) and variance \( \text{Prob}(T=t)(1-\text{Prob}(T=t)) \sum_{i=1}^{L} \text{length}(S_i) - M + 1 \) in which \( M \) is the number of sequences in the dataset and \( \text{Prob}(T=t) \) is the probability that this string \( t \) can be observed in the scenario of 2.3.4.

2.5 PTTRNFNDR

Processes and equations in 2.2–2.4 are incorporated into an algorithm called PTTRNFNDR. Several parameters need to be set before starting it. One is the significance threshold for the strings. If a string appears significantly more often in the sequence dataset than expected based on the present \( D \), the string is a good candidate for \( D \). This significance is the difference between the observed numbers of appearance of the string and its mean, divided by the SD from 2.4. Another parameter is the least number of appearances of the pattern candidates in the sequence dataset to be inserted into \( D \).

The algorithm iteratively finds patterns in different lengths. Each iteration corresponds to a particular length of strings to be investigated. For example, the algorithm counts how many times every three-letter-long string appears in the sequence dataset in the third iteration. When the algorithm finishes the investigation of the \( L \)-letter-long strings, it checks the convergence of the search. If it has found new patterns or the probabilities of the patterns in \( D \) have changed during the last \( L \) iterations, the algorithm restarts to check the one-letter-long strings based on the present \( D \). Otherwise the algorithm stops.

The strategy to use iterations to first check short patterns, then check long patterns and then recheck the short patterns is based on the following observation: some short patterns might be parts of longer patterns and they will disappear from \( D \) after the longer patterns have been found. By rechecking the short patterns, the algorithm could find the true short patterns (not parts of longer patterns) that are not found in the previous iterations due to their insignificance compared to those false short patterns that are parts of longer patterns. The details of the algorithm are as follows.

2.5.1 Initialization An empty \( D \) is initialized by inserting all letters in \( E \). Their frequencies of appearance are their initial probabilities. In this step, we set variable \( l \) to be 1, where \( l \) denotes the length of patterns we are trying to find.

2.5.2 Search of new patterns The variable \( l \) increases by 1 if \( l < L \). The goal of this step is to find all the candidate patterns in the sequence dataset that are \( l \) letters long. A string appearing more often than determined by its probability is regarded as a new pattern candidate. In this step, the whole sequence database is scanned to find how many times each \( l \)-letter-long string appears. Candidate strings are paired with their probability estimates and inserted into \( D \). If the pattern is already in \( D \), its probability is updated instead of being reinserted. After checking the convergence of \( D \) at the beginning of this step (if \( l = L \)), the algorithm stops if no change has been made to \( D \) in the last \( L \) iterations, which implies convergence of the algorithm. Otherwise, \( l \) is set to 1 and the probabilities of one letter patterns are modified.

2.5.3 Normalization of the dictionary The new patterns inserted into \( D \) in the last step cause the sum of the probabilities over all patterns in \( D \) to be more than 1. By dividing all pattern probabilities by this sum, the probabilities are normalized.

2.5.4 Fragmentation of the sequences All sequences in the dataset are decomposed into patterns based on the present \( D \) using the method shown in Section 2.2. As a consequence, the frequency of every pattern appearing in the sequences is derived, which should approximate the probability of the pattern when the sequence dataset is large based on the law of large numbers.

2.5.5 Update of pattern probabilities If the difference between the frequencies calculated in the last step and the probabilities of the patterns measured by some norm is within the preset error range, the algorithm returns to step 2.5.2 to search for new patterns. Otherwise, the probabilities in \( D \) are replaced by these frequencies and the algorithm proceeds with the last step—step 2.5.4—to decompose the sequence dataset again based on the new probabilities in the updated \( D \).

2.6 Monte Carlo database regeneration

For the purposes of this study, the patterns from various (forward) protein sequence databases were modeled with PTTRNFNDR. After the dictionaries were created, decoy databases were regenerated from these patterns based on the probabilities associated with them. Monte Carlo sampling was used to populate the decoy database with patterns and the patterns were concatenated to resemble sequences in pattern composition and length in the forward database.

2.6.1 Protein sequence databases Three protein sequence databases from yeast (26 November 2003 release of the Saccharomyces cerevisiae protein database; 6479 sequences; ftp://ftp.ncbi.nih.gov/blast/db/), Arabidopsis thaliana, (V6.0; 30,862 sequences; ftp://ftp.arabidopsis.org/home/tair/Sequences/blast_databases/) and rice (V3.0; 61,250 sequences; http://rice.tigr.org/) were used. Six bovine protein sequence records were appended to each database (catalase, apotransferrin, carbamic anhydrase, glutamate dehydrogenase, serum albumin and lactoperoxidase). These databases were analyzed by PTTRNFNDR.
2.6.2 Randomly choose a pattern from the given dictionary. The guide table algorithm was used to compose sequences using the pattern strings and their probabilities. The guide table algorithm (Chen and Asau, 1974) is efficient in that the expected number of comparisons for every sampling is at most 2.

2.6.3 Concatenate the patterns into artificial protein sequences. To create a database containing protein sequences in similar lengths as sequences in the forward database, every sequence is composed such that the expected length of the composed sequence is equal to the length of the corresponding protein sequence in the selected database. The difference between the artificial sequence and the real one is less than the maximal length of all patterns in the D.

For each protein sequence $S_i$, its length is $|S_i|$. To compose an artificial sequence $R_i$, pattern strings are randomly chosen from the D using the method delineated in 2.6.2. Before the chosen string (denoted by $w$) is appended to $R_i$, which is empty at the beginning, the sum of the length of $R_i$ and the length of $w$ is checked against $|S_i|$. If it does not exceed it, $w$ is appended to $R_i$, and then the next pattern is chosen and the operation repeated. If the sum is larger than $|S_i|$, a new $U$ is sampled from uniform distribution in $[0,1]$. If the following holds,

$$U < \frac{|S_i| - |R_i|}{|w|} \tag{7}$$

then the pattern $w$ will be appended to the present sequence $R_i$ and the new $R_i$ is written to the file (decoy database). Otherwise, $R_i$ is output as the artificial sequence without $w$. The expected length of the artificial sequence $R_i$ is equal to $|S_i|$ when (7) is used. The decoy database is created when this procedure is finished for all protein sequences in the forward database.

2.7 Mass spectrometry and Mascot searching

Tryptic digests of bovine catalase, apotransferrin, carbonic anhydrase, glutamate dehydrogenase, serum albumin and lactoperoxidase were prepared according to manufacturer instructions (Michrom Bioresources, Auburn, CA, USA). The peptide mixtures were separated in a 12-step process and analyzed on an LCQ-Deca XP ion trap mass spectrometer (Cooper et al., 2006). MS/MS were searched with Mascot 2.1 against the forward protein sequence databases described in 2.6.1, reversed versions of these databases (reversed from start to stop) and the various decoy databases created in 2.6.3.

3 RESULTS

The goal of this project was to determine whether protein sequence databases populated with sequence patterns at a frequency similar to forward databases served as better decoys than reverse databases when trying to evaluate false positive discovery rates. PTTRNFDNR was developed to model probabilities associated with the appearance of real patterns in sequences. To demonstrate its capabilities, we first examined English text and DNA sequences for the presence of real patterns.

3.1 Natural language sequence processing

As in Bussemaker et al. (2000), we tested our algorithm on the sequence dataset acquired from the first 10 chapters of Moby Dick (Bussemaker et al., 2000). If we treat root words having different forms as unique words, then these chapters contain 4214 unique English words. For example, ‘whale’ and ‘whales’ are counted as two unique words for simplicity. Among those words, 1584 appear more than once and these words are the target set since our algorithm relies on repetition to find the frequency of patterns. All spaces, punctuation marks, and number digits were deleted, and all lower case letters were converted to capital letters.

The maximal length of the pattern candidates was set at 12 when analyzing the English text. We required a string to have at least two copies in the text in order to be accepted as a pattern candidate. The significance threshold was set to be 3 SDs. 5377 patterns were inserted into the final dictionary. The top 20 patterns in the dictionary were {S, A, AND, E, IN, I, THE, T, OF, TO, THAT, D, BUT, IT, O, Y, IS, ED, HE, AS}. It is not surprising that ‘S’ is the top pattern because of the plural forms of nouns. However, patterns like ‘ed’, or ‘ing’, ‘er’, and ‘non’ are also valid findings due to their common use in English. The top five patterns containing 12 letters in the dictionary were {FATHERMAAPPLE, HARPOONEERIS, LANDLORDSAID, CIRCUMSTANCE and ASTONISHMENT}. The first three are combinations of real words. Combinations were detected because the algorithm does not consider syntax. Some other 12-letter patterns found were ‘CONGREGATION’, ‘HEREANDTHERE’, ‘INDIFFERENCE’ and ‘THEWILLOFGOD’. If we only look at the top 1800 patterns (to make a comparison to Bussemaker et al., 2000), 937 were real words and 491 were combinations of real words as defined by their calculated probabilities. The remaining were word segments such as ‘ERPENDICULAR’.

In this case, since the maximal length of the patterns was set at 12, the real word ‘PERPENDICULAR’ was not detected. A subsequent test where the maximal length of patterns was set at 13 found ‘PERPENDICULAR’. In sum, this demonstrates the capability of PTTRNFDNR to detect patterns in sequence sets since the real and combined real words are good patterns in that they usually appear more often than other character strings.

3.2 DNA sequence processing

PTTRNFDNR was applied to a dataset consisting of all non-overlapping sequences (800 maximum) upstream of coding regions in the yeast genome (downloaded from http://rsat.ulb.ac.be/rsat/). No data were removed from the dataset. There are 50 known motifs defined by Kellis et al. (2003) that were used as the target set. Most of the motifs are binding sites of transcription factors and were originally found by way of cross-species genomic sequence comparisons or through biological experimentation.

It should be stated that the concept of ‘motif’ when referring to DNA sequences is not exactly the same as our concept of ‘pattern’. A motif can be matched to several different strings. For example, ‘CGGCNCGG’ can contain A, C, G or T in place of N. A pattern, however, can only be matched to one unique string, i.e. itself. Furthermore, the context of a sequence, which is inherent to patterns, is relevant to language and biology but not resolved by the algorithm. Using English as an example, real patterns like ‘A’ or ‘AND’ reveal little of the meaning of the text. The words ‘CIRCUMSTANCE’ and ‘ASTONISHMENT’, on the other hand, are contextually more descriptive. In biology, motifs provide context. PTTRNFDNR
does have the capability to find important patterns but also finds patterns that are less biologically interesting with no defined context. For both English and DNA, relevant contextual patterns are usually longer than background patterns.

For analysis, the maximal length of the pattern candidates was set at 19 and every pattern string had to appear at least two times in the sequence dataset. The significance threshold was set to be 20 SDs due to the large size of the dataset compared to the size of the alphabet. PTTRNFNDR created a dictionary with 5662 patterns whose frequency of appearance was calculated not to be random. Patterns were ranked by decreasing probability.

By our rules, a pattern had to be at least as long as the target motif, but not shorter. We detected real patterns representing 38 out of 50 motifs, and these motifs were found in the top 800 patterns in the dictionary (Supplementary Material 2). PTTRNFNDR did not detect motifs as listed but did detect patterns represented by them. For example, TCGGGCGGC TDW was identified by the three patterns TCGGCGGCTAA (2×), CTCGGCGGCTAAATT (2×), and GTCGCGG GCTATT (2×). Considering that the dictionary consists of patterns whose lengths range from 1 letter to 19 letters, this result can be seen as a positive sign of PTTRNFNDR’s capability of detecting patterns in biological sequences.

3.3 Protein decoy database creating

Once the effectiveness of PTTRNFNDR was determined, three protein sequence databases appended with six sequences of bovine protein standards were analyzed for patterns. The databases for yeast, A.thaliana, and rice were chosen because they are different in size and complexity. The databases contain records with homology only to bovine catalase and glutamate dehydrogenase, but these records do not share identical tryptic peptide sequences. Mascot always assigned the highest score to the spectrum matched to a bovine standard. Combined with the bovine sequences, the yeast, A.thaliana and rice sequences represent different sized databases containing records to which true and false matches could be made.

A string needed to appear at least three times in the sequence database in order to be inserted into the dictionary as a pattern. The significance threshold was set to be 3 SDs. Because the rate of false positive spectrum/peptide sequence matches can depend on the size of the database versus the length of the peptide sequence, three different pattern maximal lengths (6, 8 and 12 amino acids; a.a.) were examined. Based on those patterns found in these forward databases (Supplementary Material), a Monte Carlo method was used to generate an artificial sequence database, herein called a decoy database, by randomly recombining the patterns according to their distributions. For each set of parameters, we created ten decoy databases that are the same size and have the same number of records as the corresponding forward sequence databases. In total, 90 decoy databases were created for this test.

3.4 Decoy and reverse database searching

Spectra generated from the standard proteins were searched against forward, reverse and our decoy databases using Mascot. Every match to a peptide sequence not associated with any of the six standard proteins was treated as a false match. Rather than evaluate the total numbers of peptide sequence matches, we compared the 12 highest false-match scores from a forward and reverse database search and the 12 highest scores averaged over 10 decoy searches. Since the difference of the Ions score and the Identity score is one measure for the quality of a spectrum/peptide sequence match, the 12 false matches that had the highest value for Ions score minus Identity score were also considered.

Differentials between scores from reverse and decoy searches with respect to the false forward match scores are indicative of how well a database performs for modeling false matches. If a search generates a set of scores close to the false forward database scores, then the database is deemed to be good for modeling false positive rates. If the difference is large, false positive rates deduced from searching the reverse or decoy database are not as accurate. Differentials were measured as distances calculated by Euclidean norm where each set of highest scores was treated as a vector. If the top \( i \) scores were greater than the reverse database scores were (Fig. 1; Supplementary Material). Across the sets of all 12 top scoring false matches, the average scores from decoys were closer to the false forward scores than were the reverse database scores. This was true if the database was populated with ≤6, 8 or 12 a.a. patterns. The differentials for the reverse database scores were nearly twice that for the average decoy scores, which indicates that the reverse database is not as well suited as the decoys.

Different results were obtained when using the A.thaliana database to model false positive rates. For some of the decoys, the scores were closer to the false forward database scores than were the reverse database scores, while for others, the distances were greater than the reverse database scores were (Fig. 1; Supplementary Material). For the 8 and 12 a.a. patterned decoys, more averaged decoy scores were closer to the false forward scores than were the reverse database scores. The clearest indication that the A.thaliana decoys are better suited for false positive rate modeling can be seen by comparing the top false match scores from all the searches. The top match (highest) false score is most likely to be mistaken as a true match. The fact that the differentials for the top decoy scores were smaller than for the top reverse scores is a positive indication that the decoy databases are more suitable for false match modeling.

When the rice database was used, half of all of the derived decoy databases performed better than the reverse database. Average scores for the 6 a.a. patterned decoys were slightly better than the reverse scores, but 8 and 12 a.a. average scores were slightly worse (Fig. 1; Supplementary Material). Nevertheless, as was seen in the A.thaliana decoys, the
differentials for the top decoy scores were smaller than the differentials for the top reverse scores. Because many researchers use the Ions–Identity score relationship to measure the quality of a spectrum/peptide sequence match, we analyzed these score differences. The results are similar to what was previously observed. For the yeast and the rice 6 a.a. decoys, the averaged Ions–Identity relationships were closer to the false forward Ions–Identity relationships than were the reverse relationships (Fig. 1; Supplementary Material 4). Again for the yeast decoys, the differentials for the reverse Ions–Identity relationships were two times that of the averaged decoys. Despite the averaged Ions–Identity score relationships being closer to the false forward relationships than the reverse score relationships were, the differentials for the lower ranking scores were still pronounced. Because the Identity score is a product of the number of peptides in the database with similar mass to charge ratio as the precursor ion for the spectrum, these results suggest that the decoy databases contain a similar number of candidate peptides as the forward database. This means that there was very little mass bias created during database regeneration.

As a final test, we searched another bovine protein spectral dataset against the forward, reverse and 6 a.a. pattern decoys and compared the number of false matches found for a range of Ions score thresholds. Such analysis is commonly used to say that a peptide is positively identified if its Ions score is higher than the prescribed threshold (cutoff). The results are consistent with the previous analysis. Generally, the number of false positives obtained from the forward database searches are better estimated by the decoy database searches than the reverse database (Fig. 1; Supplementary Material 5). However, for the A.thaliana and rice databases, thresholds exist for which reverse databases perform better than the decoys.

4 DISCUSSION

An accurate estimation of false positive rates is necessary to produce a confidence measure for tandem mass spectrum/peptide sequence matches. False positive rates can change as samples, experimental protocols, instruments, databases or spectrum/peptide sequence match score thresholds change. Researchers have used reverse database searching as an easy way to estimate false positive rates. But the reverse database has inherent problems.

For one, even though reverse databases contain the same number of sequences, amino acids and tryptic peptides, the biologically relevant patterns, patterns which are conserved through evolution due to their importance in molecules’ basic functions and structures, are lost because most are not symmetric. Therefore, comparing observed spectra acquired from natural protein sequences to spectra predicted from decoy (gray) databases. The databases searched and maximum lengths of patterns found from the forward database and used in decoys are shown. Averages for 10 decoy database trials and SDs are shown. Euclidean distances (relative to forward scores) between reverse and average decoy score data represented in the bar graphs are plotted next to each bar graph. The last graph compares false positives versus Ions score cutoffs.
artificial protein sequences with reverse patterns could give rise to a potential risk of introducing biases for the estimation of false positive identification rates.

Another problem with the reverse database search method is that there is only one true reverse database for every forward database and only one such comparison can be made between the forward and reverse database. Multiple observations are required to collect statistical information, such as mean and variance to obtain a better understanding of randomness hidden behind an event. As a randomization approach, the estimate of a false positive rate acquired from a reverse database search can be higher or lower than the real false positive rate, but there is no way to predict how far it could be from the real value since there is only one value available.

Here, we propose a new approach to create decoy protein sequence databases for the estimation of false positive rates. These decoys contain real patterns found in the forward database. The probability distribution of the patterns is based on their frequencies appearing in the optimally decomposed forms of these sequences. Since a Monte Carlo sampling method is then used to create a decoy database, many different decoy databases can be made so statistical information can be generated.

We searched a set of tandem mass spectra from purified proteins against three forward databases, the corresponding three reverse databases and 90 decoy databases created by our algorithm. In many cases, the high scoring matches to peptides from the decoy databases were closer to the scores of false peptide matches from the forward database than were the scores of matches to peptides in the corresponding reverse database.

Why, then, is it better that our decoy match scores were closer to forward false-match scores than the reverse match scores were? The answer lies in the estimates made to gauge real false positive rates. If the reverse search scores are not close to the forward false-match scores from the forward database, then the estimated false positive rate will not be close to the real false positive rate. Thus, better estimates are obtained from the decoys whose match scores approach forward false-match scores.

In our studies, the best estimates were measured by performing searches against the decoys created from the yeast database. Improvements were also evident when the A. thaliana database was used, but less improvement was obtained with long patterns from the rice database. This order of performance appears to coincide with the order of the complexity of proteins encoded by the genomes of the species and size of the corresponding protein sequence databases. It is possible that it might be easier for our algorithm to find the patterns in the simpler yeast protein databases, which could result in a pattern set (dictionary) with better quality. An analogy would be the difference in word complexity between the texts of Green Eggs and Ham and King Lear. Since the realm of underlying patterns in protein sequence databases is not well understood or defined, further research on amino acid patterns may be required to achieve the best simulation performance for different databases.

Before the decoy databases created by our algorithm can be broadly applied to experimental datasets to estimate false identification rates, several issues need to be resolved. The first is that due to the nature of the algorithm, the decoy database, when virtually digested, has peptides that can be found in the forward database. Automatic methods need to be used to find and/or get rid of these peptides from the database search results. A second issue is that searching multiple decoy databases results in variant score distribution which can be considered with score means to roughly predict the confidence interval of the false positive rates acquired from the forward database search. However, it is not straightforward how to manage all of these data. One solution might be to apply Oracle or MySQL. This method requires some changes of the pipeline used in MS/MS data processing, but it will make false positive rate estimation easier for future applications.

A new annotation of the rice genome has been described by Itoh et al. (Genome Res. 17:175-183, 2007). This new version contains half of the number of protein sequences as the version we used.

ACKNOWLEDGEMENTS

This work was funded by a specific cooperative agreement between the Johns Hopkins University and USDA-ARS, Beltsville, MD and in part by a National Research Initiative grant award to B.C. from USDA-CSREES (no. 2005-35605-15392).

Conflict of Interest: none declared.

REFERENCES


