BIOINFORMATICS ORIGINAL PAPER
Vol. 23 no. 1 2007, pages 77–83
doi:10.1093/bioinformatics/btl511

Systems biology

DASS: efficient discovery and *p*-value calculation of substructures in unordered data

Jens Hollunder[1], Maik Friedel[1], Andreas Beyer[1,2], Christopher T. Workman[2] and Thomas Wilhelm[1,*]

[1]Department of Theoretical Systems Biology, Leibniz Institute for Age Research—Fritz-Lipmann-Institute e. V. (former IMB Jena), Beutenbergstrasse 11, D-07745 Jena, Germany and [2]Department of Bioengineering, University of California, San Diego, La Jolla, CA 92093, USA

Received on May 31, 2006; revised on September 28, 2006; accepted on October 1, 2006
Advance Access publication October 10, 2006
Associate Editor: Martin Bishop

ABSTRACT

Motivation: Pattern identification in biological sequence data is one of the main objectives of bioinformatics research. However, few methods are available for detecting patterns (substructures) in unordered datasets. Data mining algorithms mainly developed outside the realm of bioinformatics have been adapted for that purpose, but typically do not determine the statistical significance of the identified patterns. Moreover, these algorithms do not exploit the often modular structure of biological data.

Results: We present the algorithm DASS (Discovery of All Significant Substructures) that first identifies all substructures in unordered data ($DASS_{Sub}$) in a manner that is especially efficient for modular data. In addition, DASS calculates the statistical significance of the identified substructures, for sets with at most one element of each type ($DASS_{P_{set}}$), or for sets with multiple occurrence of elements ($DASS_{P_{mset}}$). The power and versatility of DASS is demonstrated by four examples: combinations of protein domains in multi-domain proteins, combinations of proteins in protein complexes (protein subcomplexes), combinations of transcription factor target sites in promoter regions and evolutionarily conserved protein interaction subnetworks.

Availability: The program code and additional data are available at http://www.fli-leibniz.de/tsb/DASS

Contact: wilhelm@fli-leibniz.de

Supplementary information: Supplementary information is available at *Bioinformatics* online.

1 INTRODUCTION

Pattern recognition is one of the most fundamental themes in bioinformatics (Ouzounis and Valencia, 2003). It comprises different problem classes, such as binding site discovery (Stormo, 2000; Tompa *et al*., 2005), multiple sequence alignment [e.g. hidden Markov models (Eddy, 1998)], classification of proteins [e.g. SCOP (Andreeva and Srikant, 1994), SUPERFAMILY (Madera *et al*., 2004)], characterization of protein complexes (Aloy *et al*., 2004; Hollunder *et al*., 2005a) and functional annotation of proteins [e.g. FunSpec (Robinson *et al*., 2002)]. Numerous algorithms exist to analyze biological sequences [e.g. FASTA (Pearson and Lipman, 1988), BLAST (Altschul *et al*., 1990) and PSI-BLAST (Altschul *et al*., 1997)]. However, only few methods deal with non-sequential (unordered) data.

Although lacking sequential order, such data nevertheless may contain hidden structure and hierarchically organized features. Hierarchical organization and modularity are common in biology and can be observed at many levels of cellular organization (Gavin and Superti-Furga, 2003). For instance, protein complexes are frequently composed of different protein subcomplexes, also called protein modules or molecular machines (Wilhelm *et al*., 2003; Hollunder *et al*., 2005a). By combining proteins and protein modules into specific complexes the cell achieves a higher degree of versatility, flexibility and robustness. In addition, the proteins used in these complexes are themselves composed of different protein domains (Apic *et al*., 2001; Vogel *et al*., 2004) that are specifically combined to define a protein's structure and function. Other examples are the combinatorial transcriptional gene regulation (Ihmels *et al*., 2004; Beyer *et al*., 2006) and modules of interacting proteins conserved across species that can be used as building blocks of biological networks and pathways (Li *et al*., 2004; Sharan *et al*., 2005). Therefore, it is important to develop efficient methods for the investigation of such modular and hierarchical structures.

Here we present algorithms for the detection and scoring of structures in data that can be represented as sets, e.g. sets of proteins or sets of protein domains. Most existing pattern recognition algorithms for unordered data are variants of the APRIORI algorithm (Agrawal and Srikant, 1994). Here a breadth-first search is used to iterate through the items (elements) of each set, considering a threshold for the item frequencies in the result sets. In the first iteration, all items below this frequency threshold are removed and all combinations of the remaining items are tested (two-itemset candidates). Frequent itemsets at the given level are extended by more items to generate candidate sets for the next iteration and infrequent subsets are discarded. Recent approaches use similar search strategies with item-based data structures to reduce the search space. Three classes of algorithms can be distinguished: those finding all frequent patterns (sets), e.g. APRIORI (Agrawal and Srikant, 1994), those determining only maximal frequent sets (a set with no frequent superset containing this set), e.g. MAFIA (Burdick *et al*., 2001), and algorithms finding frequent closed sets [a set is 'closed' if there exists no superset with the same frequency (Pei *et al*., 2000)], e.g. CHARM (Zaki and Hsiao, 2002), Close (Pasquier *et al*., 1999), and CLOSET (Pei *et al*., 2000). All of

*To whom correspondence should be addressed.

these approaches require a user-specified threshold for the set frequency. We are not aware of any algorithm quantifying the statistical significance of the observed patterns. Depending on the frequency of the constitutive elements, abundant patterns might not be statistically significant, whereas patterns with a comparably low abundance may be significant.

In Section 2, we present an approach that detects patterns in unordered hierarchical data and estimates the statistical significance of these patterns without requiring a pattern frequency threshold. DASS (Discovery of All Significant Substructures) comprises two independent algorithms: the first ($DASS_{Sub}$, Section 2.2.1) performs a fast search to find all closed sets. Closed sets provide a concise representation of the data, because the number of closed sets is typically much smaller than the number of all contained patterns. In contrast to existing algorithms, which iterate through the set elements, $DASS_{Sub}$ iterates through the sets (structures) themselves. This approach allows for efficient pruning when analyzing hierarchically structured data and provides a superior performance compared with other algorithms. In the second algorithm (Section 2.2.2) the statistical significance of all closed sets is calculated. Two cases are distinguished: (1) sets with unique elements and (2) sets with multiple occurrence of the same elements (multi-sets). In each case DASS calculates the probability (p-value) that the considered closed set is found in a randomized dataset with the same or higher frequency.

The DASS algorithm has numerous potential applications. Section 3 outlines applications to protein domain combinations, protein complexes, and transcriptional modules. As an additional application, a new method to find conserved subnetworks is presented in the Supplementary information 2.4.

2 ALGORITHM

2.1 Problem formulation and preliminaries

Let $\Sigma = \{e_1, e_2, \ldots, e_{|\Sigma|}\}$ be a finite alphabet of elements e_k. We consider sets $S_i = \{(E, m), E \subseteq \Sigma\}$ (e.g. measured protein complexes), where E is the (unordered) list of elements e_k contained in S_i and $m : E \to \mathbb{N}^+$ is a function mapping the e_k onto their frequencies in S_i. Thus, $m(e_k)$ is the frequency of an element e_k in S_i and $n_i = \sum_{\forall k: e_k \in S_i} m(e_k)$ is the size of S_i. If e_k may occur more than once, S_i is a multi-set with $m(e_k) \geq 1$. The dataset $\mathcal{D} = \{S_1, S_2, \ldots, S_{|\mathcal{D}|}\}$ comprises a finite set of structures S_i (e.g. Fig. 1a). For instance, $\mathcal{D}$ could be a collection of measured protein complexes, where each structure S_i comprises a set of proteins and contains all different proteins. The frequency of an element e_k in $\mathcal{D}$ is defined as $F_k = \sum_{\forall i: S_i \in \mathcal{D}} m(e_k)$. Thus, we can compute the relative frequency of e_k as $f_k = F_k/N$, where N is the total number of all elements in $\mathcal{D}$. See Figure 1b for an example.

Let the substructure $M_j \subseteq S_i$ be a non-empty set or multi-set of $n_j = \sum_{\forall k: e_k \in M_j} m(e_k)$ elements and $\Lambda(M_j) \subseteq \mathcal{D}$ be the subset of all structures from $\mathcal{D}$ containing the specific substructure M_j ('host structures'). The frequency $F_j = |\Lambda(M_j)|$ of a substructure M_j is the number of structures in the dataset $\mathcal{D}$ containing M_j (e.g. Fig. 1c). If each S_i corresponds to an independent measurement, a high frequency F_j indicates a large number of measurements confirming the existence of the substructure M_j.

DEFINITION 1 *(closed set). A substructure M_X is called a closed set if there exists no substructure M_Y such that $M_X \subset M_Y$ and $F_X = F_Y$. In other words, M_X is called closed if there exists no superset $M_Y \supset M_X$ with the same frequency.*

(a) the dataset
$\mathcal{D} = \{ \; S_1 = \{a, a, b, c, c, d\}, \; S_2 = \{a, a, b, d, f\}, \; S_3 = \{a, b, d, e\} \; \}$

(b) alphabet, probabilities, and sizes
$\Sigma = \{a, b, c, d, e, f\}$
$n_1 = 6,\ n_2 = 5,\ n_3 = 4, \quad N = 15$
$f_a = \frac{5}{15},\ f_b = \frac{3}{15},\ f_c = \frac{2}{15},\ f_d = \frac{3}{15},\ f_e = \frac{1}{15},\ f_f = \frac{1}{15}$

(c) example for host structures
$\Lambda(\{a, a, b, d\}) = \{ \; S_1 = \{a, a, b, c, c, d\}, \; S_2 = \{a, a, b, d, f\} \; \}$
$F_{\{a,a,b,d\}} = 2$

(d) all closed sets
$\mathcal{M} = \{$
$M_1 = \{a, a, b, c, c, d\} = S_1,$
$M_2 = \{a, a, b, d, f\} = S_2,$
$M_3 = \{a, b, d, e\} = S_3,$
$M_4 = \{a, a, b, d\} = S_1 \cap S_2,$
$M_5 = \{a, b, d\} = S_1 \cap S_3, S_2 \cap S_3, \text{or } S_1 \cap S_2 \cap S_3 \; \}$

(e) closed sets with frequency $F_j = |\Lambda(M_j)| > 1$

$M_4 = \{a, a, b, d\}$	$F_4 = 2$	$p(M_4) = 0.12465$
$M_5 = \{a, b, d\}$	$F_5 = 3$	$p(M_5) = 0.08549$

Fig. 1. Example. (**a**) the dataset $\mathcal{D} = \{S_1, S_2, S_3\}$; (**b**) the used alphabet Σ, the size of each structure and the relative frequencies of each element; (**c**) host structures for the closed set $\{a, a, b, d\}$; (**d**) all closed sets from $\mathcal{D}$; and (**e**) the p-values $p(M_j)$ of all closed sets M_j which occur more than once in $\mathcal{D}$.

Obviously, each structure S_i itself is a closed set.

$\mathcal{M} = \{M_1, M_2, \ldots, M_{|\mathcal{M}|}\}$ is the set of all closed sets of the dataset $\mathcal{D}$ (e.g. Fig. 1d). The first part of the DASS algorithm finds $\mathcal{M}$ for a given dataset $\mathcal{D}$ and the second part computes the statistical significance of each M_j based on its frequency (e.g. Fig. 1e).

2.2 The DASS Algorithm

2.2.1 Finding all substructures The algorithm $DASS_{Sub}$ finds $\mathcal{M}$ in a given dataset $\mathcal{D}$ by systematically intersecting every structure with all other structures. Figure 2 shows the pseudocode for the algorithm (without frequency calculations for clarity). The algorithm starts with an empty set $\mathcal{M}$ (Fig. 2, line 1) and works by using two loops (i-loop and j-loop), and the helper-set H. The i-loop iterates through the structures S_i (lines 2–12), the j-loop iterates through the current $\mathcal{M}$ (lines 5–9). H is reinitialized in every i-loop with the current structure S_i (line 4) and then collects all intersections of S_i with all substructures M_j in the current $\mathcal{M}$ (line 7). After finishing the j-loop, $\mathcal{M}$ is updated with H (line 10). Finally, $\mathcal{M}$ contains all closed sets from $\mathcal{D}$ (line 13). This can be proven by taking into account that all structures S_i are closed sets, all intersections of two closed sets are closed again, and the complete intersection tree (all combinations of intersections of structures) is traversed by the algorithm. Figure 3 outlines the algorithm for the example of Figure 1.

As mentioned above, this algorithm iterates through the structures S_i and not through the elements of Σ. Compared with other data mining algorithms (Zaki and Hsiao, 2002; Pasquier *et al.*, 1999; Pei *et al.*, 2000), $DASS_{Sub}$ has the advantage of finding all closed sets without needing to test that the sets are closed. Generally, given $\mathcal{D} = \{S_1, S_2, \ldots, S_{|\mathcal{D}|}\}$ with $|\mathcal{D}|$ structures,

$DASS_{Sub}$ algorithm

Input: $\mathcal{D} = \{S_1, S_2, ..., S_{|\mathcal{D}|}\}$
Output: $\mathcal{M} = \{M_1, M_2, ..., M_{|\mathcal{M}|}\}$

```
1: M ← {∅}
2: for all i : S_i ∈ D do
3:     if S_i ∉ M then
4:         H ← {S_i}
5:         for all j : M_j ∈ M do
6:             if {S_i ∩ M_j} ∉ M and {S_i ∩ M_j} ∉ H then
7:                 H ← H ∪ {S_i ∩ M_j}
8:             end if
9:         end for
10:        M ← M ∪ H
11:    end if
12: end for
13: return M
```

Fig. 2. The DASS_{Sub} algorithm—a set-wise enumeration algorithm for finding all closed sets M_j in a given dataset $\mathcal{D}$.

	i	j	if-check	H	$\mathcal{M}$
	0				
$\mathcal{M}$:					$\{\}$
	1	0	$S_1 = \{a, a, b, c, c, d\}$	S_1	
update $\mathcal{M}$:					$\{S_1\}$
	2	0	$S_2 = \{a, a, b, d, f\}$	S_2	
		1	$S_2 \cap S_1 = \{a, a, b, d\}$	$S_2 \cap S_1$	
update $\mathcal{M}$:					$\{S_1, S_2, S_1 \cap S_2\}$
	3	0	$S_3 = \{a, b, d, e\}$	S_3	
		1	$S_3 \cap S_1 = \{a, b, d\}$	$S_3 \cap S_1$	
		2	$S_3 \cap S_2 = \{a, b, d\}$	–	
		3	$S_3 \cap S_1 \cap S_2 = \{a, b, d\}$	–	
update $\mathcal{M}$:					$\{S_1, S_2, S_1 \cap S_2, S_3, S_3 \cap S_1\}$

Fig. 3. Illustration of the DASS_{Sub} algorithm considering the example of Figure 1.

$\mathcal{M} = \{S_1, S_2, \ldots, S_{|\mathcal{D}|}, S_1 \cap S_2, S_1 \cap S_3, \ldots, S_1 \cap S_2 \cap S_3, \ldots, S_1 \cap S_2 \cap \ldots \cap S_{|\mathcal{D}|-1} \cap S_{|\mathcal{D}|}\}$ contains $2^{|\mathcal{D}|} - 1$ sets in the worst case.

However, DASS_{Sub} is particularly fast if newly added substructures M_j in $\mathcal{M}$ are often identical to already existing sets, because this reduces the number of intersections in subsequent j-loops. Fortunately, hierarchically organized biological data have exactly this property: the frequency of substructures (i.e. modules) is quite large compared with the number of different substructures. This vastly reduces the computation time of DASS_{Sub}. For instance, $\mathcal{M}$ in Figure 1d contains only five different sets, because $S_1 \cap S_3 = S_2 \cap S_3 = S_1 \cap S_2 \cap S_3$. Algorithms iterating through elements from Σ do not exploit the hierarchical organization of biological data.

Finally, we compute the frequency F_j of each substructure $M_j \in \mathcal{M}$.

2.2.2 Statistical significance of all substructures The calculation of the statistical significance of substructures can be divided into two different problems: (1) structures with unique elements (simple sets) ($\text{DASS}_{\text{P}_{\text{set}}}$) and (2) structures with possible multiple occurrence of the same elements ($\text{DASS}_{\text{P}_{\text{mset}}}$). The appropriateness of the respective approach depends on the dataset and on the biological question.

In each case (simple set or multi-set) our underlying null model assumes that all elements e_k from Σ are randomly assigned to the structures S_i, taking into account their relative frequencies f_k. The exact way of determining the statistical significance is to generate all possible permutations of a given dataset $\mathcal{D}$ [Supplementary information 1.1.1 (a)]. However, this is only practicable for very small datasets. A good approximation for the exact model is provided by shuffling the data. This procedure maintains all structure sizes and re-assigns all elements from the 'element pool' (containing all e_k with their frequencies f_k) [Supplementary information 1.1.1 (b)]. After each randomization the frequencies of all substructures are counted. The p-value of a given structure is the number of randomizations containing the substructure at least as often as its frequency in original data $\mathcal{D}$ divided by the total number of randomizations [Supplementary information 1.1.1 (b)—Model IIA]. For very improbable substructures this procedure is very costly, because many randomizations are needed. The corresponding p-value calculation can be improved by fitting for a given substructure its frequency distribution [Supplementary information 1.1.1 (b)—Model IIB]. However, because this procedure is still computationally demanding, we developed another shuffling model suited for fast analytical calculations, which is described in the following. It is based on the average probability $\hat{p}$ to find a substructure in the randomized data. The p-value is calculated with a binomial distribution using this average probability and the observed frequency of the substructure. The binomial distribution $(k, n, \hat{p})$ is an exact test to calculate the probability of at least k successes in n Bernoulli experiments with the probability $\hat{p}$ of success. A detailed discussion of all these models and a validation of our final analytical model is given in the Supplementary information 1.1.

Both $\text{DASS}_{\text{P}_{\text{set}}}$ and $\text{DASS}_{\text{P}_{\text{mset}}}$ estimate the probability, p_j (n_i), of finding M_j in a random set of size n_i for all structure sizes $n_i \geq n_j$ contained in the dataset $\mathcal{D}$. The average probability $\hat{p}_j$ of finding M_j in such a random set is

$$\hat{p}_j = \frac{1}{K_j} \sum_{\forall i: n_i \geq n_j} k_i \cdot p_j(n_i), \tag{1}$$

where k_i is the number of structures of size n_i and $K_j = \sum_{\forall i: n_i \geq n_j} k_i$ is the total number of structures with size $\geq n_j$. The product $k_i \cdot p_j(n_i)$ is the expectation value to find Mj in the structures of size n_i. The average probability $\hat{p}_j$ and the observed frequency F_j of M_j are used to compute the statistical significance based on a binomial distribution. The probability of observing M_j at least F_j times in a randomized dataset D_{random} thus reads:

$$p(M_j) = \sum_{t=F_j}^{K_j} \binom{K_j}{t} \cdot \hat{p}_j^t \cdot (1-\hat{p}_j)^{K_j - t}. \tag{2}$$

Next, we show how the probability $p_j(n_i)$ is calculated.

(i) $\text{DASS}_{\text{P}_{\text{set}}}$. If each element e_k can occur at most once in a set [$\forall k : e_k \in M_j$ is $m(e_k) = 1$], the number of different permutations in a random set S_r of n_j elements is $n_j!$ and the probability of finding a substructure M_j in a random set S_r of n_j elements is:

$$p_j(n_j) = n_j! \prod_{\forall k: e_k \in M_j} f_k. \tag{3}$$

Note that this equation corresponds to a sampling with replacement. Of course, this assumption is violated for a small number of elements. In such a case one should use a more exact model as

indicated above. However, biological datasets typically have a large number of elements, justifying the use of Equation (3) [see Supplementary information 1.1.1 (d)].

Next, the probability p_j of finding a substructure M_j in a random set S_r with $n_r \geq n_j$ can be calculated as 1 minus the probability of not finding M_j in $C_{n_j}^{n_r} = \binom{n_r}{n_j}$ random experiments:

$$p_j(n_r) = 1 - (1 - p_j(n_j))^{C_{n_j}^{n_r}}. \quad (4)$$

The $p_j(n_r)$ are calculated for all structure sizes $\geq n_j$ contained in the given dataset.

(ii) $\text{DASS}_{\text{P}_{\text{mset}}}$. We now consider the case where the M_j are multi-sets. The probability $p_j(n_r)$ of finding M_j in a random set S_r of size $n_r \geq n_j$ can be calculated by a summation of the probabilities of all possible outcomes containing M_j at least once. A brute force algorithm would generate all $|\Sigma|^{n_r}$ possible outcomes to calculate this probability. Of course, this is applicable only for a small number of structures and a small alphabet of elements. We present an efficient, iterative algorithm to estimate the probability of finding M_j in S_r for successively increasing set sizes. Intermediate results of previous iteration steps are efficiently re-used to avoid unnecessary recalculations.

The number of different permutations of a given substructure M_j is:

$$\text{Perm}(M_j) = n_j! \left(\prod_{\forall k: e_k \in M_j} m(e_k)! \right)^{-1}. \quad (5)$$

The probability that n_j positions are randomly filled with elements from M_j is

$$\text{Prob}(M_j) = \prod_{\forall k: e_k \in M_j} f_k^{m(e_k)} \quad (6)$$

The probability $p_j(n_j)$ of finding a given substructure M_j in a random set of n_j elements is:

$$p_j(n_j) = \text{Perm}(M_j) \cdot \text{Prob}(M_j). \quad (7)$$

Note that (Equation 7) is a generalization of (Equation 3).

If $n_r > n_j$ Equation (7) has to be expanded by an additional factor $\mathcal{F} > 1$, accounting for the higher probabilities for the occurrence of M_j in larger sets. $\mathcal{F}$ monotonically increases with the number of free positions. For the general case $n_r \geq n_j$ the probability $p_j(n_r)$ is calculated by

$$p_j(n_r) = \text{Perm}(M_j) \cdot \text{Prob}(M_j) \cdot \mathcal{F}(M_j, n_r), \quad (8)$$

with $\mathcal{F}(M_j, n_r \equiv n_j) = 1$.

All random sets S_r containing M_j have n_j fixed elements. The remaining $n_d = n_r - n_j$ positions can be filled with any element from Σ. The purpose of $\mathcal{F}$ in Equation (8) is to account for the different ways in which these free positions can be filled. For instance, if $n_d = 1$ then $p_j(n_r) = p_j(n_j + 1)$ can be calculated by summing up all possible outcomes with one additional element, which may be an element of M_j or not. A brute force algorithm for the generation of all outcomes is very costly. To make the calculation more efficient, we recursively calculate $\mathcal{F}$ as a function of its predecessors, i.e. $\mathcal{F}(M_j, n_r) = f[\mathcal{F}(M_j, n_r - 1)]$. Additionally, we use an efficient 'factor out technique', where the subtotals are re-used extensively throughout the procedure (intermediate values are stored in a matrix SubT). Figure 4 shows the pseudo-code for the complete algorithm $\text{DASS}_{\text{P}_{\text{mset}}}$. The code also demonstrates the efficient calculation of the factor $\mathcal{F}$.

$DASS_{P_{mset}}$ algorithm

Input: substructure $M_j \in \mathcal{M}$ with size n_j and a structure size $n_r \geq n_j$
Output: probability $p_j(n_r)$ to find M_j in a random set S_r

1: $Prob(M_j) = \prod_{\forall k: e_k \in M_j} f_k^{m(e_k)}$
2: $Perm(M_j) = n_j! \left(\prod_{\forall k: e_k \in M_j} m(e_k)! \right)^{-1}$
3: **if** $n_j == n_r$ **then**
4: **return** $p_j(n_r \equiv n_j) = Prob(M_j) \cdot Perm(M_j)$
5: **end if**
6: $M_j' \leftarrow M_j \cup \{\bar{e}\}$ with $f_{\bar{e}} = 1 - \sum_{\forall k: e_k \in M_j} f_k$ and $m(\bar{e}) = 0$
7: **for all** $pos \leftarrow 1 \ldots n_d$ **do**
8: **for all** $k : e_k \in M_j'$ **do**
9: $temp = 1.0$
10: **for all** $s \leftarrow 1 \ldots pos - 1$ **do**
11: $temp = temp \cdot f_k \cdot (m(e_k) + pos - s + 1)^{-1} + SubT[s][k-1]$
12: **end for**
13: $SubT[pos][k] = SubT[pos][k-1] + temp \cdot f_k \cdot (m(e_k) + 1)^{-1}$
14: **end for**
15: **end for**
16: $\mathcal{F}(M_j, n_r) = \frac{n_r!}{n_j!} \cdot SubT[pos][k]$
17: **return** $p_j(n_r) = Prob(M_j) \cdot Perm(M_j) \cdot \mathcal{F}(M_j, n_r)$

Fig. 4. The algorithm $\text{DASS}_{\text{P}_{\text{mset}}}$ determines the probability $p_j(n_r)$ to find M_j in a random structure S_r with possible multiple occurrence of the same elements in the general case $n_r \geq n_j$. SubT$[x][0] = 0$ for $1 \leq x \leq n_d = n_r - n_j$.

At first $\text{DASS}_{\text{P}_{\text{mset}}}$ computes the probability $p_j(n_j)$ (Fig. 4, lines 1–5). Next, $\text{DASS}_{\text{P}_{\text{mset}}}$ defines a composed set $M_j = M_j \cup \{\bar{e}\}$ containing all elements of a given substructure M_j and the pseudo-element $\bar{e} \notin M_j$ where $\bar{e}$ is a placeholder for all elements $e_k \in \Sigma \backslash M_j$. Thus, the relative frequency of $\bar{e}$ in M_j' is $p_{\bar{e}} = 1 - \sum_{\forall k: e_k \in M_j} f_k$ (line 6).

In an outer loop (lines 7–15) the algorithm iterates through the n_d free positions, which can be occupied by any element from Σ. In the inner k-loop (lines 8–14) the algorithm iterates through the different elements of M_j' always in a predefined fixed order and stores all subtotals in a matrix SubT (line 13). The matrix SubT contains the corresponding subtotal for each possible free position and element $e_k \in M_j'$ and is used in later steps for determining the factor $\mathcal{F}$. The most inner loop (lines 10–12) re-uses the subtotals in SubT to compute the subtotal for the given element e_k. This loop is only used if $n_d > 1$. Finally, $\mathcal{F}$ (line 16) and the corresponding probability $p_j(n_r)$ (line 17) are computed.

Due to the pre-calculation of partial results, the computation time is polynomial $\left[O\left(\frac{n_d^3}{2} |\Sigma|\right) \right]$, instead of the exponential time required for the brute force algorithm $[> O(|\Sigma|^{n_d})$: $O(n_d)$ (loop of line 6), $O(|\Sigma|)$ (loop of line 7) and $O(\frac{n_d \cdot (n_d - 1)}{2})$ (loop of line 9)].

As in the simple set case above, also in the multi-set case the $p_j(n_r)$ calculation is only an approximation (sampling with replacement). However, in most cases it leads to a conservative estimate of the exact p-value [see Supplementary information 1.1.1 (d)], the accuracy increases with an increasing number of elements in Σ and would be exact for an infinite number of elements. In most biological applications the number of elements in Σ is much larger than the largest structure S_i. Thus, $\text{DASS}_{\text{P}_{\text{mset}}}$ is generally a very good approximation even if Σ is finite.

We illustrate the algorithm (Fig. 4) with help of the following example: we calculate the probability of finding the substructure $M_4 = \{a, a, b, d\}$ (cf. Fig. 1) in random structures of size four, five and six (Fig. 5). The values of SubT denote the cumulative subtotals. The 'free positions' are filled with the indicated elements

(a) Probability of finding $M_4 = \{a, a, b, d\}$ in a random structure S_r of size $n_r = 4$:
1. $Prob(M_4) = \prod_{\forall k: e_k \in M_j} f_k^{\,m(e_k)} = (\frac{1}{3})^2 \cdot \frac{1}{5} \cdot \frac{1}{5} = \frac{1}{225}$ (Fig. 4, line 1)
2. $Perm(M_4) = n_j! \left(\prod_{\forall k: e_k \in M_j} m(e_k)!\right)^{-1} = \frac{4!}{2!1!1!} = 12$ (line 2)
3. $p_4(4) = Perm(M_4) \cdot Prob(M_4) = \frac{4}{75}$ (line 4)

(b) Probability of finding $M_4 = \{a, a, b, d\}$ in a random structure S_r of size $n_r = 5$:
1. $Prob(M_4) = \frac{1}{225}$ (Fig. 4, line 1)
2. $Perm(M_4) = 12$ (line 2)
3. $M_4' = M_4 \cup \{\,\bar{e}\}$, where $\bar{e} \in \Sigma \setminus M_4 = \{c, e, f\}$ (line 6)
4. iteration through 1 $(= 5 - 4)$ free position (pos-loop of line 7)
5. iteration in the fixed order: $a, b, d, \bar{e}$ (k-loop, line 8)

$$
\begin{array}{lcll}
SubT[1][1] & = & \frac{f_a}{3} & = \frac{f_a}{3} & = \frac{1}{9} \\
SubT[1][2] & = & \frac{f_b}{2} + \frac{f_a}{3} & = \frac{f_b}{2} + SubT[1][1] & = \frac{1}{10} + \frac{1}{9} \\
SubT[1][3] & = & \frac{f_d}{2} + \frac{f_a}{3} + \frac{f_b}{2} & = \frac{f_d}{2} + SubT[1][2] & = \frac{1}{10} + \frac{19}{90} \\
SubT[1][4] & = & \frac{f_{\bar{e}}}{1} + \frac{f_a}{3} + \frac{f_b}{2} + \frac{f_d}{2} & = \underbrace{\frac{f_{\bar{e}}}{1} + SubT[1][3]}_{\text{line 13}} & = \frac{1}{15} + \frac{14}{45}
\end{array}
$$

6. $\mathcal{F}(M_4, 5) = \frac{5!}{4!} \cdot SubT[1][4] = 5 \cdot \frac{26}{45} = \frac{26}{9}$ (line 16)
7. $p_4(5) = Perm(M_4) \cdot Prob(M_4) \cdot \mathcal{F}(M_4, 5) = \frac{104}{675}$ (line 17)

(c) Probability of finding $M_4 = \{a, a, b, d\}$ in a random structure S_r of size $n_r = 6$:
1. $Prob(M_4) = \frac{1}{225}$ (Fig. 4, line 1)
2. $Perm(M_4) = 12$ (line 2)
3. $M_4' = M_4 \cup \{\,\bar{e}\}$, where $\bar{e} \in \Sigma \setminus M_4 = \{c, e, f\}$ (line 6)
4. iteration through 2 $(= 6 - 4)$ free positions (pos-loop of line 7)
5. iteration in the fixed order: $a, b, d, \bar{e}$ (k-loop, line 8)

$SubT[1][1]$, $SubT[1][2]$, $SubT[1][3]$, and $SubT[1][4]$ are the already calculated values of (b)

$$
\begin{array}{lcll}
SubT[2][1] & = & \frac{f_a}{3}(\frac{f_a}{4}) & = \frac{1}{9} \cdot \frac{1}{12} \\
SubT[2][2] & = & \frac{f_b}{2}(\frac{f_b}{2} + \frac{f_a}{3}) + SubT[2][1] & \\
 & = & \underbrace{\frac{f_b}{2}(\frac{f_b}{3} + SubT[1][1])}_{\text{temp}} + SubT[2][1] & = \frac{1}{10}(\frac{1}{15} + \frac{1}{9}) + \frac{1}{108} \\
SubT[2][3] & = & \frac{f_d}{2}(\frac{f_d}{3} + \frac{f_b}{2} + \frac{f_a}{3}) + SubT[2][2] & \\
 & = & \frac{f_d}{2}(\frac{f_d}{3} + SubT[1][2]) + SubT[2][2] & = \frac{1}{10}(\frac{1}{15} + \frac{19}{90}) + \frac{73}{2700} \\
SubT[2][4] & = & \frac{f_{\bar{e}}}{1}(\frac{f_{\bar{e}}}{2} + \frac{f_d}{2} + \frac{f_b}{2} + \frac{f_a}{3}) + SubT[2][3] & \\
 & = & \underbrace{\frac{f_{\bar{e}}}{1}(\underbrace{\frac{f_{\bar{e}}}{2} + SubT[1][3]}_{\text{loop of line 10}}) + SubT[2][3]}_{\text{line 13}} & = \frac{4}{15}(\frac{2}{15} + \frac{14}{45}) + \frac{37}{675}
\end{array}
$$

6. $\mathcal{F}(M_4, 6) = \frac{6!}{4!} \cdot SubT[2][4] = 30 \cdot \frac{13}{75} = \frac{26}{5}$ (line 16)
7. $p_4(6) = Perm(M_4) \cdot Prob(M_4) \cdot \mathcal{F}(M_4, 6) = \frac{104}{375}$ (line 17)

Fig. 5. Example for the algorithm $DASS_{P_{mset}}$. Calculation of the probability to find the substructure M_4 (Fig. 1) in a random structure. (**a**) of size four, (**b**) five and (**c**) six. Note, that re-using SubT[1][1], SubT[1][2], SubT[1][3], and SubT[1][4] from (b) in (c) reduces the time complexity.

(Fig. 5b and c). For instance, using the fixed order a, b, d, $\bar{e}$: SubT[1][1] refers to the case of an additional 'a' at the first free position, $SubT$[1][2] to the case of finding either 'a' or 'b' at the first free position, and so on. SubT[2][1] in Figure 5c refers to the case that both free positions are filled with 'a', and SubT[2][2] that both free positions are either filled with only 'a' or 'b' or are filled with 'a' and 'b' and so on. In the Supplementary information 1.2 it is shown that $DASS_{P_{mset}}$ generates the same results than a brute force algorithm.

3 APPLICATIONS

The following examples demonstrate the power and versatility of DASS for a broad range of applications. Of course, many more applications are conceivable, e.g. a new method for the identification of conserved protein interaction networks is outlined in the Supplementary information 2.4.

3.1. Protein domain combinations

Multi-domain proteins are composed of different protein domains that determine the function of the whole protein. Vogel *et al*. (2004) discussed combinations of two and three domains that recur in different proteins together with different partner domains in a particular functional and spatial relationship. Using the $DASS_{P_{mset}}$ algorithm we analyze the corresponding general problem of protein domain combinations of any size. Domain assignments of the proteins were taken from the SUPERFAMILY database (Madera *et al*., 2004). In a future work, we will elaborate on the functions of significant domain combinations together with their geometrical arrangement and their distribution in different species. Here we identify different protein domain combinations containing the SH2 domain (Src-homology-2) and/or the PDZ domain (PSD-95, discs large and ZO-1, see Supplementary information 2.1). SH2 domains have previously been identified in a wide range of signaling proteins, often together with other signaling domains (e.g. SH3 domain, Pleckstrin homology domain, protein tyrosin phosphatase domain) or together with scaffold domains (Yaffe, 2002). PDZ domains play a key role in organizing diverse cell signaling assemblies and occur often as multiple copies and together with other signaling domains (Fan and Zhang, 2002). We identified more than 100 significant PDZ modules ($P < 10^{-5}$) containing PDZ and other domains, which can be classified into different subclasses (Fan and Zhang, 2002). Proteins directly associated with the plasma membrane (ion channels, receptors and cytoskeleton proteins) are among the major cellular targets of PDZ domains. In contrast to normal tissue, in some human tumors (e.g. colon and breast cancer), proteins containing PDZ and LIM domains are overexpressed (Kang *et al*., 2000). PDZ domains are abundant in animal proteins, yet scarce in yeast, bacteria and plants (Ponting, 1997). We identified similar significant PDZ modules in different multicellular species, suggesting conserved signaling mechanisms in these organisms (see Supplementary information 2.1).

3.2. Protein subcomplexes in *S.cerevisiae* and *E.coli*

One level above combinations of protein domains in single proteins are combinations of whole proteins in protein complexes which act as highly specialized cellular molecular machines.

We separately analyzed datasets of the two model organisms *S.cerevisiae* (Hollunder *et al*., 2005a) and *E.coli* (Hollunder *et al*., 2005b) and identified the underlying modular composition of protein complexes. We argue that subcomplexes represent more reliable protein assemblies than the originally measured complexes. Using the DASS algorithm we identified well-characterized protein assemblies with known functions, for instance, Casein kinase II, 19/22S (regulator of the proteasome), eIF2B and Arp2/3 in *S.cerevisiae* and the DNA-dependent RNA polymerase complexes and the 2-oxoglutarate dehydrogenase complex in *E.coli*.

On the other hand, we also identified previously unknown protein assemblies that may fulfill special cellular functions. We characterized the subcomplexes and subcomplex proteins in detail and found that subcomplexes have specific properties that underline their distinct role. For instance, subcomplexes are enriched for essential proteins and the expression of subcomplex proteins is more correlated than the expression of complex proteins in general. Subcomplexes of *S.cerevisiae* and *E.coli* are also characterized by a more homogeneous spatial and functional composition, compared with that of the measured host complexes. This property is exploited to propose functions and sub-cellular localizations of unannotated proteins. Proteins of unknown function belonging to known subcomplexes or to complexes with homogeneous

functional or spatial composition get assigned the function or localization of this subcomplex.

The result of the protein subcomplex identification analyzing the recently published MALDI-TOF MS data (Krogan *et al*., 2006) is shown in the Supplementary information 2.2.

3.3. Transcription factor modules in *S.cerevisiae*

In an accompanying study we analyzed the interaction of transcription factors (TF) for the combinatorial regulation of target genes (Beyer *et al*., 2006). In that study we developed a scoring system for robustly assigning TFs to their target genes and we subsequently used $DASS_{P_{set}}$ to determine which TFs cooperatively regulate a significant number of target genes. The analysis of these TF modules provides novel insights into combinatorial transcriptional regulation. However, many TFs bind more than once in the upstream regions of their regulated target genes and it is assumed that repeated occurrence of the same TF increases the repressing or activating signal (Harbison *et al*., 2004). Thus, in addition to simply identifying sets of TFs acting together, here we also analyze the number of repeated occurrences of TFs in these modules using $DASS_{P_{mset}}$. For this analysis we selected all highly significant TF—target interactions in the yeast *S.cerevisiae* from Beyer *et al*. (2006). Interactions with a log-likelihood score larger than six were selected for this study. These 3820 interaction pairs were used to assign TF-binding sites to their target genes. The number of repeats was determined based on the number of binding sites within the first 1000 bp upstream of the start codon. Binding sites were determined using ANN-Spec (Workman and Stormo, 2000) and assuming a significance threshold of $P < 10^{-4}$.

$DASS_{P_{mset}}$ found 702 TF-modules, 438 of which were significant with $P < 10^{-4}$ (see Supplementary information 2.3). $DASS_{P_{set}}$ finds only 200 significant TF-modules using the same interactions and requiring the same level of significance. This result shows that the number of binding sites carries significant additional information.

Figure 6 shows frequency distributions for selected TFs. It can be seen that some TFs have a strong preference for low-redundancy (e.g. Gat1p), whereas others have a broad distribution of binding site frequencies (e.g. Skn7p). Some TFs always have repeated binding elements. Examples are the drug resistance regulators Pdr1p and Pdr3p, which almost always have at least two binding sites in the upstream regions of their targets. Although the two TFs are assumed to be homologous they do not always regulate the same genes (DeRisi *et al*., 2000; Zhu and Xiao, 2004). Hence, the distributions of their binding sites in Figure 6 are not identical. However, both TFs primarily require either two or four binding sites, while triple binding sites appear less frequently. This does not come as a surprise, because Pdr1p and Pdr3p bind the palindromic *cis*-element 5′-TCCGCGGA-3′ as dimers (Mamnun *et al*., 2002). Accordingly, the TF module Pdr1p/Pdr1p/Pdr3p/Pdr3p is highly significant (20 targets, $P = 2 \times 10^{-55}$), whereas a TF module with single copies of the two TFs (i.e. Pdr1p/Pdr3p) does not exist at all. Thus, the TF-modules correctly reveal the stoichiometry of regulatory complexes.

Another interesting TF module is Cbf1p/Cbf1p/Tye7p/Tye7p. To our knowledge, it has not been reported that Cbf1p and Tye7p form heterodimers and no interaction between these TFs was found [iHOP (Hoffman and Valencia, 2004); SGD (Christie *et al*., 2004); MIPS (Mewes *et al*., 2004)]; however, the two TFs bind to identical chromosomal locations. Cbf1p's binding motif (8 bases long) is completely contained in Tye7's binding motif (10 bases long). Also, Cbf1p and Tye7p have a significant number of common targets (31 targets, $P = 10^{-43}$). Hence, in this case it is more likely that Cbf1p and Tye7p regulate their targets in a mutually exclusive way. Further studies would be required to validate the presence of a competitive interaction. It is known that Cbf1p physically interacts with Met4p (Kuras *et al*., 1996) and that Met31p and Met32p may replace the role of Cbf1p in the regulatory complex (Blaiseau and Thomas, 1998). In support of this, we find a number of significant TF-modules ($P < 10^{-5}$) involving Cbf1p together with Met4p, Met31p and Met32p. These examples demonstrate that DASS is able to reveal known combinatorial interactions and to discover unknown regulatory modules.

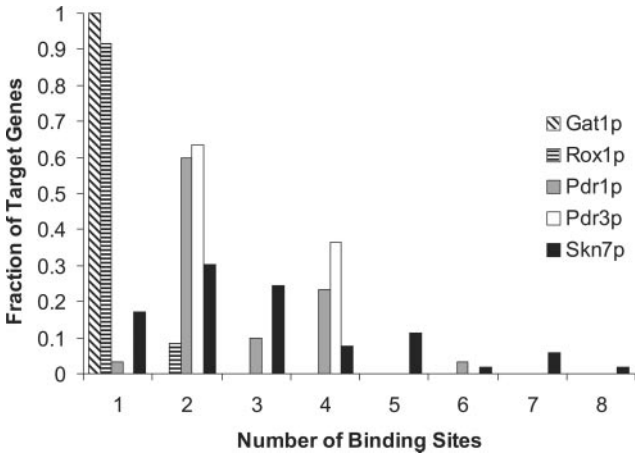

Fig. 6. Histogram of binding site frequencies for selected yeast transcription factors.

4 CONCLUSION

The DASS algorithm represents an efficient method to calculate the statistical significance of substructures in unordered data. It is composed of two independent parts. The first sub-algorithm $DASS_{Sub}$ finds all patterns in unordered data. It is especially efficient for hierarchically organized data, a typical property of large datasets in molecular biology. In the second part, the statistical significance of all identified patterns is calculated ($DASS_P$). The algorithms $DASS_{P_{set}}$ and $DASS_{P_{mset}}$ are developed to handle the two different cases: simple sets or multi-sets. These algorithms perform approximated estimations of statistical significances, valid for large datasets. The Supplementary information 1.1 contains additional detailed discussions of four other models better suited for smaller datasets.

The DASS algorithm has a very broad range of applications. Here we demonstrated the application of DASS to four different types of biological datasets. Modules with low *p*-values are overrepresented, whereas high *p*-values indicate underrepresentation. Interestingly, all closed sets in protein complex data (protein subcomplexes) have low *p*-values (see Supplementary information 2.2). In contrast, in the analyzed TF binding site data are many closed sets (TF modules), which are significantly underrepresented (see Supplementary information 2.3). All resulting modules with significant *p*-values represent new hypotheses requiring further

experimental verification. These analyses will help to better understand the hierarchical and combinatorial nature of cellular processes.

ACKNOWLEDGEMENTS

The authors thank S. Nikolajewa and M. Hoffmann for discussing statistical aspects and the anonymous referees for important comments. This work has been funded by the Federal Ministry of Education and Research, Germany (grant 0312704E).

Conflict of Interest: none declared.

REFERENCES

Altschul,S.F. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.

Altschul,S.F. *et al.* (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.

Agrawal,R. and Srikant,R. (1994) Fast algorithms for mining association rules. In *Proceedings of 20th International Conference on Very Large Data Bases*, 487–499.

Aloy,P. *et al.* (2004) Structure-based assembly of protein complexes in yeast. *Science*, **303**, 2026–2029.

Andreeva,A. *et al.* (2004) SCOP database in 2004: refinements integrate structure and sequence family data. *Nucleic Acids Res.*, **32**, D226–D229.

Apic,G. *et al.* (2001) Domain combinations in archaeal, eubacterial and eukaryotic proteomes. *J. Mol. Biol.*, **310**, 311–325.

Beyer,A. *et al.* (2006) Integrated assessment and prediction of transcription factor binding. *PLOS Comput. Biol.*, **2**, e70.

Blaiseau,P.L. and Thomas,D. (1998) Multiple transcriptional activation complexes tether the yeast activator Met4 to DNA. *EMBO J.*, **17**, 6327–6336.

Burdick,D. *et al.* (2001) MAFIA: a maximal frequent itemset algorithm for transactional databases. In *Proceedings of the 17th International Conference on Data Engineering (ICDE 2001)*, pp. 443–452.

Christie,K.R. *et al.* (2004) *Saccharomyces* Genome Database (SGD) provides tools to identify and analyze sequences from *Saccharomyces cerevisiae* and related sequences from other organisms. *Nucleic Acids Res.*, **32**, D311–D314.

DeRisi,J. *et al.* (2000) Genome microarray analysis of transcriptional activation in multidrug resistance yeast mutants. *FEBS Lett.*, **470**, 156–160.

Eddy,S.R. (1998) Profile hidden Markov models. *Bioinformatics*, **14**, 755–763.

Fan,J.S. and Zhang,M. (2002) Signaling complex organization by PDZ domain proteins. *Neurosignals*, **11**, 315–321.

Gavin,A.-C. and Superti-Furga,G. (2003) Protein complexes and proteome organization from yeast to man. *Curr. Opin. Chem. Biol.*, **7**, 21–27.

Harbison,C.T. *et al.* (2004) Transcriptional regulatory code of a eukaryotic genome. *Nature*, **431**, 99–104.

Hoffmann,R. and Valencia,A. (2004) A gene network for navigating the literature. *Nat. Genet.*, **36**, 664.

Hollunder,J. *et al.* (2005a) Identification and characterization of protein subcomplexes in yeast. *Proteomics*, **5**, 2082–2089.

Hollunder,J. *et al.* (2005b) Exploiting combinatorial complexity—searching for new functional entities in the cell. In *Proceedings of Foundation of System Biology in Engineering (FOSBE 2005)*, pp. 363–366.

Ihmels,J. *et al.* (2004) Defining transcription modules using large-scale gene expression data. *Bioinformatics*, **20**, 1993–2003.

Kang,S. *et al.* (2000) PCD1, a novel gene containing PDZ and LIM domains, is overexpressed in several human cancers. *Cancer Res.*, **60**, 5296–5302.

Krogan,N.J. *et al.* (2006) Global landscape of protein complexes in the yeast *Saccharomyces cerevisiae*. *Nature*, **440**, 637–643.

Kuras,L. *et al.* (1996) A heteromeric complex containing the centromere binding factor 1 and two basic leucine zipper factors, Met4 and Met28, mediates the transcription activation of yeast sulfur metabolism. *EMBO J.*, **15**, 2519–2529.

Li,S. *et al.* (2004) A map of the interactome network of the metazoan *C. elegans*. *Science*, **303**, 540–543.

Madera,M. *et al.* (2004) The SUPERFAMILY database in 2004: additions and improvements. *Nucleic Acids Res.*, **32**, D235–D239.

Mamnun,Y.M. *et al.* (2002) The yeast zinc finger regulators Pdr1p and Pdr3p control pleiotropic drug resistance (PDR) as homo- and heterodimers *in vivo*. *Mol. Microbiol.*, **46**, 1429–1440.

Mewes,H.W. *et al.* (2004) MIPS: analysis and annotation of proteins from whole genomes. *Nucleic Acids Res.*, **32**, D41–D44.

Ouzounis,C.A. and Valencia,A. (2003) Early bioinformatics: the birth of a discipline—a personal view. *Bioinformatics*, **19**, 2176–2190.

Pasquier,N. *et al.* (1999) Discovering frequent closed itemsets for association rules. In *Proceedings of the 7th International Conference on Database Theory (ICDT 1999)*, *Lecture Notes in Computer Science,* Vol. **1540**, Springer, pp. 398–416.

Pei,J. *et al.* (2000) CLOSET: an efficient algorithm for mining frequent closed itemsets. In *Proceedings of ACM SIGMOD International Workshop on Data Mining and Knowledge Discovery*, *(DMKD 2000),* pp. 21–30.

Pearson,W.R. and Lipman,D.J. (1988) Improved tools for biological sequence comparison. *Proc. Natl Acad. Sci. USA*, **85**, 2444–2448.

Ponting,C.P. (1997) Evidence for PDZ domains in bacteria, yeast, and plants. *Protein Sci.*, **6**, 464–468.

Robinson,M.D. *et al.* (2002) FunSpec: a web-based cluster interpreter for yeast. *BMC Bioinformatics*, **3**, 35.

Sharan,R. *et al.* (2005) Conserved patterns of protein interaction in multiple species. *Proc. Natl Acad. Sci. USA*, **102**, 1974–1979.

Stormo,G.D. (2000) DNA binding sites: representation and discovery. *Bioinformatics*, **16**, 16–23.

Tompa,M. (2005) Assessing computational tools for the discovery of transcription factor binding sites. *Nat. Biotechnol.*, **23**, 137–144.

Vogel,C. *et al.* (2004) Supra-domains. Evolutionary units larger than protein domains. *J. Mol. Biol.*, **336**, 809–823.

Wilhelm,T. *et al.* (2003) Physical and functional modularity of the protein network in yeast. *Mol. Cell. Prot.*, **2.5**, 292–298.

Workman,C.T. and Stormo,G.D. (2000) ANN-Spec: a method for discovering transcription factor binding sites with improved specificity. *Pac. Symp. Biocomput.*, **5**, 467–478.

Yaffe,M.B. (2002) Phosphotyrosine-binding domains in signal transduction. *Nat. Rev. Mol. Cell Biol.*, **3**, 177–186.

Zaki,M.J. and Hsiao,C.-J. (2002) CHARM: an efficient algorithm for closed itemset mining. In *Proceedings of the 2nd SIAM International Conference on Data Mining (SDM 2002)*, pp. 457–473.

Zhu,Y. and Xiao,W. (2004) Pdr3 is required for DNA damage induction of MAG1 and DDI1 via a bi-directional promoter element. *Nucleic Acids Res.*, **32**, 5066–5075.