Bioinformatics Advances, 2023, vbad005
https://doi.org/10.1093/bioadv/vbad005
Advance Access Publication Date: 2 February 2023
Original Article

Software

Snekmer: a scalable pipeline for protein sequence fingerprinting based on amino acid recoding

Christine H. Chang[1], William C. Nelson[1], Abby Jerger[1], Aaron T. Wright [2], Robert G. Egbert [1] and Jason E. McDermott [1,3,*]

[1]Biological Sciences Division, Pacific Northwest National Laboratory, Richland, WA 99352, USA, [2]Department of Biology, Baylor University, Waco, TX 76798, USA and [3]Department of Molecular Microbiology and Immunology, Oregon Health & Science University, Portland, OR 97239, USA

*To whom correspondence should be addressed.
Associate Editor: Cecilia Arighi

Received on March 14, 2022; revised on December 16, 2022; editorial decision on January 13, 2023; accepted on February 1, 2023

Abstract

Motivation: The vast expansion of sequence data generated from single organisms and microbiomes has precipitated the need for faster and more sensitive methods to assess evolutionary and functional relationships between proteins. Representing proteins as sets of short peptide sequences (kmers) has been used for rapid, accurate classification of proteins into functional categories; however, this approach employs an exact-match methodology and thus may be limited in terms of sensitivity and coverage. We have previously used similarity groupings, based on the chemical properties of amino acids, to form reduced character sets and recode proteins. This amino acid recoding (AAR) approach simplifies the construction of protein representations in the form of kmer vectors, which can link sequences with distant sequence similarity and provide accurate classification of problematic protein families.

Results: : Here, we describe Snekmer, a software tool for recoding proteins into AAR kmer vectors and performing either (i) construction of supervised classification models trained on input protein families or (ii) clustering for *de novo* determination of protein families. We provide examples of the operation of the tool against a set of nitrogen cycling families originally collected using both standard hidden Markov models and a larger set of proteins from Uniprot and demonstrate that our method accurately differentiates these sequences in both operation modes.

Availability and implementation: Snekmer is written in Python using Snakemake. Code and data used in this article, along with tutorial notebooks, are available at http://github.com/PNNL-CompBio/Snekmer under an open-source BSD-3 license.

Contact: jason.mcdermott@pnnl.gov

Supplementary information: Supplementary data are available at *Bioinformatics Advances* online.

1 Introduction

The ability to rapidly and inexpensively sequence diverse biological samples has driven a sharp increase in the amount of sequence information available. However, analysis of functional annotation assignment has revealed that, on average, only ~50–60% of genes are assigned any functional annotation using current standard annotation techniques, and only about half of those receive specific functional assignment (Lobb *et al.*, 2020; Salzberg, 2019). Development of new methods providing improved and expanded functional annotations, including user-driven exploration of sequence space and automated construction of functional prediction models, have significantly trailed the huge increases in sequence information becoming available. Sequence similarity (e.g. via BLAST) is commonly used to assign functional annotations, and hidden Markov models (HMMs) provide a probabilistic approach to modeling protein families and motifs. However, traditional sequence similarity methods are unable to accurately capture distantly related sequences in many cases, and HMMs can be time consuming to build, in part due to the requirement for building multiple sequence alignments from families (Bateman *et al.*, 2000; Eddy, 1998).

Previously, for machine learning and rapid comparison, proteins have been represented as vectors of short peptide sequences (kmers). At least one major annotation resource [Rapid Annotation using Subsystem Technology (RAST)] uses the kmer approach for protein annotation (Edwards *et al.*, 2012; Overbeek *et al.*, 2014) and multiple newer tools, such as MMseqs2 (Mirdita *et al.*, 2019; Steinegger and Söding, 2017) and DIAMOND (Buchfink *et al.*, 2021) use kmers to increase the speed and sensitivity of searches in various ways. We previously extended the protein kmer approach via amino acid recoding (AAR), which uses chemical similarities between amino acids to simplify the sequence space (McDermott *et al.*,

2019). We found that a simple AAR grouping of amino acids by hydrophobicity or hydrophilicity performed well in classifying disparate families of ubiquitin ligase effector mimics from viral and bacterial pathogens (McDermott *et al.*, 2019). This work demonstrated that the kmer AAR approach can be used as the basis for machine learning models capable of classifying sets of functionally related proteins with little sequence similarity. We concluded that the AAR allows greater flexibility in sequence representation and captures similarity between sequences that would not be detected via conventional methods, and these observations have been borne out in other studies as well (Hauswedell *et al.*, 2014; Liang *et al.*, 2022). Thus, AAR creates simplified representations of proteins in the form of flexible feature vectors, which can then be used to construct machine learning models for functional prediction, explore sequence similarities in newly sequenced datasets (e.g. from metagenomes), and identify similarities between sets of sequences that may be undetected using other methods.

Our previous study included code implementing the kmer AAR approach for the specific application, but this code was neither easily applied to new protein families nor compatible with high-performance computing to build models for large numbers of sequences. Here we describe Snekmer (Fig. 1), an open-source Python tool that expands the AAR approach into a flexible, modular pipeline incorporating multiple recoding schemes, model training and automatic evaluation. Snekmer generates AAR features from input proteins, trains models automatically from input sets of proteins and clusters proteins based on kmer similarity assessment. Snekmer can also be used as a general tool to cluster moderately sized sets of proteins (10–100k) to determine protein sequence and function similarity using unsupervised clustering. Users can alternatively supply their own sets of proteins to build kmer-based models, assess those models for performance and apply the models to novel sets of sequences.

To evaluate Snekmer and the AAR approach, we applied Snekmer to a set of protein families associated with the nitrogen cycle and to a larger set of disparate proteins from Uniprot. Our results demonstrate excellent fidelity with the HMMs that were used to identify these families, across a range of AAR and kmer lengths, and good agreement with MMSeqs2 for clustering performance. We anticipate Snekmer will empower users to rapidly develop their own prediction models based on sets of input sequences, enabling computationally efficient exploration of the sequence landscape in large collections of sequence information from microbiomes.

2 Results

2.1. Implementation and availability

2.1.1. Software stack

Snekmer is written in Python (version 3.6+) and uses the Snakemake [version 6.0+, (Koster and Rahmann, 2018)] workflow management system. Key dependencies include the Scikit-Learn library [sklearn; (Pedregosa *et al.*, 2011)] for building machine learning models, Bioconda (Gruning *et al.*, 2018), NumPy for matrix operations (Harris *et al.*, 2020), Pandas for general data manipulations (McKinney, 2010), HDBSCAN (McInnes *et al.*, 2017) and UMAP (McInnes *et al.*, 2018). Optionally, the Blazing Signature Filter (Lee *et al.*, 2018) can be installed to facilitate clustering on larger datasets. The implementation of Snekmer via Snakemake provides several advantages, including scalability, reproducibility and Python compatibility. Because Snakemake determines dependencies between steps in a workflow and queues each step sequentially, Snekmer easily scales to high-performance computing (HPC) environments, e.g. supercomputer and cloud computing clusters, and executes each step as its own job script. As a result, multiple input files can be processed simultaneously, in parallel.

2.1.2. Execution modes

Snekmer can be operated in three distinct modes: *model* (supervised) or *cluster* (unsupervised) and *search* to apply trained models to new sequences. In the supervised mode, users supply a series of FASTA files, each containing sequences from a single family, to construct models. For each family, Snekmer generates features from all the proteins, constructs feature vectors and calculates a probability score for each kmer feature based on its representation in that family versus other families and, optionally, in user-defined background sequences. Snekmer then builds a logistic regression classification model based on each sequence's scores for in-family versus out-of-family assignment for each family (see Section 2.1.4 below for details). Classification performance is evaluated using K-fold cross-validation. The resulting models can further be applied to new sequences that have been processed via Snekmer with matching AAR parameters.

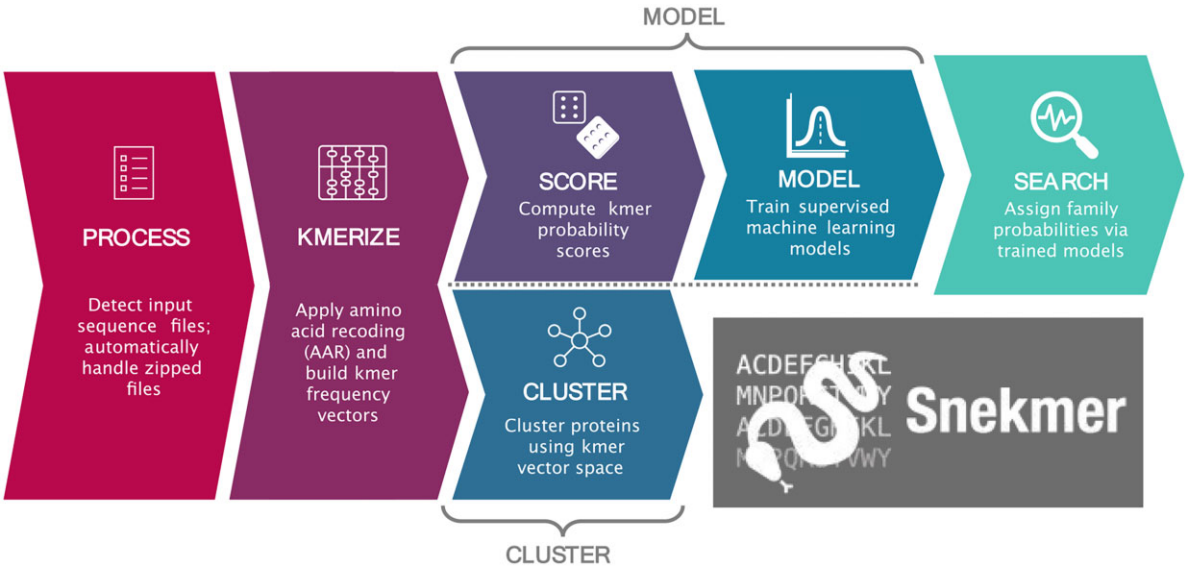

Fig. 1. Workflow for the Snekmer pipeline. Steps are implemented as individual Snakemake rules. This figure provides an overview of the main Snekmer capabilities the supervised model mode, and the unsupervised cluster mode. For both modes preliminary steps include preprocessing to filter out duplicates and clean input data, generation of kmers from input sequences and vectorization of kmer signatures to build frequency vectors. The model mode then applies a scoring algorithm to the kmers and builds a predictive model for each family. The cluster mode generates clusters based on user input parameters. A third mode, search mode, is used to apply trained models to new input sequences

In the unsupervised mode, Snekmer uses clustering to identify similarities between protein sequences in a given FASTA file or files. Snekmer generates features from all protein sequences, constructs feature vectors, and calculates the similarity between the vectors based on standard metrics (see Section 2.2.2 below for details) and can apply multiple clustering methods to produce related sets of proteins. Optionally, the similarity matrix can be output if a more granular inspection of sequence similarity is desired.

An example is included in the software (demo_example) which can be used to run each of the different modes on a small set of nitrogen cycle families. Additionally, a Jupyter notebook outlining each step of each Snekmer mode is included.

Once a Snekmer analysis has been applied, the results are summarized in a report (in HTML format) which is produced in the top-level output directory. Each report provides a short description of the results, diagnostic plots from the run if appropriate and links to the relevant output files.

2.1.3. Feature generation

Following the Snakemake framework, Snekmer accepts sequence files as input and performs a predetermined set of processing steps—known as rules in Snakemake—on each input file until the desired output file has successfully been generated. Input parameters and pipeline steps are defined by the user in a configuration (YAML) file; an example configuration file is provided in the code repository. By default, Snekmer handles a number of FASTA-type input file formats (FASTA, FAA, FNA and FA), but the user can specify acceptable file extensions in the configuration file. Snekmer also automatically decompresses files compressed with gzip. Once input files are read, Snekmer can optionally screen for duplicate sequences.

The standard Snekmer pipeline first processes the input file to generate kmer features. The desired recoding scheme (alphabet) and kmer length (k) are specified by the user. Six recoding alphabets are included (Table 1) (Arnold *et al.*, 2009; Bacardit *et al.*, 2009; McDermott *et al.*, 2019; Yamada and Tomii, 2014), or the user may specify no recoding at all. Snekmer uses the specified alphabet and kmer length to calculate the potential kmer space of all possible kmer combinations. Then, Snekmer recodes all sequences in the input file according to the desired alphabet recoding and generates a vector counting all kmers that occur in the sequence. Features that are not observed in at least one input sequence are removed from the final feature space, though this threshold can be adjusted by the user.

The 'min_filter' parameter allows users to specify a threshold for inclusion of kmers in models or the clustering process to reduce the complexity of the kmer matrix. This threshold can be either (i) an integer number of observations of a kmer, or (ii) a percentage of the sequences the kmer must be observed in. Setting a higher threshold will reduce the memory footprint of the operation and increase execution speed, but may degrade model or cluster quality if the filter is set too stringently.

2.1.4. Scoring

If supervised mode is selected, a probability score is calculated for each input protein family group. The family probability of a kmer, F_{kmer}, is the ratio of the number of family members containing the kmer to the total number of family members:

$$F_{\text{kmer}} = \frac{n_{\text{kmer}}}{N}.$$

Here, n_{kmer} is the number of family members containing the kmer and N is the total number of family members. If sequences for additional families are provided, a modified family probability score is generated by subtracting the summed fractional family probabilities for the other G families:

$$F'_{\text{kmer}} = F_{\text{kmer}} - \sum_{i=1}^{G} \frac{1}{G} \times \frac{n_{\text{kmer},i}}{N_i}.$$

For many applications, the ability to compare sequences against a background set of sequences representative of a larger sequence space is useful. User-supplied background sequences are processed using a similar method as described above in Section 2.1.3; however, to minimize computational requirements, Snekmer only evaluates the background sequences for the kmer feature space defined by the input target families. Snekmer then accounts for the prevalence of

Table 1. AAR schemes showing the groups of amino acids used for each scheme included in Snekmer

Recoding scheme	States	Groupings
No recoding	20	A C D E F G H I K L M N P Q R S T V W Y
MIQS observed substitution	11	A C D E N F W Y G H I L M Q V K R P S T
Chemical properties	7	A G I L M V C D E F W Y H P K R N Q S T
Solvent accessibility	3	A G H S T C F I L M V W Y D E K N P Q R
Hydrophobicity/charge	3	A G I L M V D E K R C F H N P Q S T W Y
Hydrophobicity/structure-breaker	3	A I L M V G P C D E F H K N Q R S T W Y
Hydrophobicity	2	A G I L M V C D E F H K N P Q R S T W Y

kmers in the background set in calculating the overall kmer probability scores.

The overall score S_{kmer} combines the modified family probability and the probability of the kmer appearing in general protein space. The background set is provided by the user and should be chosen carefully to minimize bias and maximize both sequence diversity and protein family representation.

$$S_{\text{kmer}} = \frac{F'_{\text{kmer}}}{\max(F')} \times \left(1 - \frac{P_{\text{kmer}}}{\max(P)}\right),$$

where P_{kmer} is the fraction of sequences in the background data set that contain the kmer.

The resulting method assigns a score between $(-1, 1)$, where a maximum score of 1 indicates that the kmer is found in all sequences within a given group and in no sequences outside of the group, whereas a minimum score of -1 indicates that the kmer is found in none of sequences within the group but found in all sequences outside of the group. A score of zero indicates the kmer is perfectly agnostic toward in-group or out-of-group assignment.

Because kmers with scores near zero contribute little to in-family assignment scores, these kmers are relatively unimportant to final in-group assignment. Thus, users can reduce the number of kmers used in scoring. For instance, kmers can be restricted to the top N, or top n%, of kmers by absolute score. Thus, a final score can be tailored to selectively include kmers with the strongest positive or negative association with a group assignment.

2.1.5. Model building

In supervised mode, models for input protein groups are automatically trained and evaluated. Snekmer employs the method described above to score each pre-labeled sequence based on its substituent kmers. The in-group scores are then used to train an in-group versus out-of-group classification model using logistic regression. In the process, a model is built and evaluated for each input protein family using the other input family sequences and an optional background set of sequences as the negative examples for the model. This means that model results may vary based on the number of input families modeled, as well as the relative numbers and types of sequences in those families.

Currently, trained models are stored in the Python-standard pickle binary format. Installation for Snekmer using conda will ensure that these are forward compatible with libraries used, but in future versions of Snekmer, we plan to explore alternatives to pickle for file storage.

2.1.6. Clustering

To evaluate Snekmer's ability to identify useful clusters of functionally related sequences, we clustered a 10-genome set of sequences (46 908 proteins) from Uniprot using Snekmer and the Many-against-Many sequence searching (MMseqs2) suite (Mirdita *et al.*, 2019) easy-clust mode with default parameters.

2.1.7. Evaluation

Model performance was evaluated via K-fold cross-validation and calculation of accuracy, specificity, sensitivity, area under the receiver-operator characteristic curve (AUROC or AUC ROC) and area under the precision-recall curve (AUPR or PRAUC). For each fold, kmers were rescored using the same methodology as described in Section 2.1.4, but with family probabilities calculated separately for each fold using each training set. Thus, a separate scoring basis and model were developed, trained and evaluated for a given fold. Results from each cross-validation split are stored in tabular form and summarized graphically (Fig. 2). Descriptions of each evaluation metric are given in Table 2.

The AUROC (also known as AUC ROC or ROC AUC) is commonly used to evaluate binary classifiers. However, for imbalanced datasets, the AUROC alone may be insufficient in summarizing classifier performance in instances where the number of negative examples far outweighs the number of positive examples (Davis and Goadrich, 2006). The nitrogen cycling family dataset presents such a case, since the number of positive examples (i.e. members of one family) is much smaller than the number of negative examples (i.e. members of all the remaining families) used to evaluate each model. Thus, we compared both AUROC and AUPR for all classifiers trained from the nitrogen cycling family dataset.

To evaluate results from the Snekmer and MMseqs2 clustering runs, we used available Uniprot annotations from EggNog (COG), Pfam, Interpro, GO and TIGRFams and evaluated clusters using homogeneity, completeness and the V score metrics from Scikit-Learn. Briefly, given an existing set of annotations for clustered proteins, completeness assesses the likelihood that an annotation label can be found in only one cluster and homogeneity assesses how likely a cluster is to have just one annotation label assigned to its members. The V score is the harmonic mean between homogeneity and completeness. We report average metrics across all clusters.

2.1.8. Availability

Snekmer and associated example data and code are available at http://github.com/PNNL-CompBio/Snekmer. Documentation for Snekmer is available at https://snekmer.readthedocs.io.

2.2. Example applications

To illustrate the utility of Snekmer, we tested two distinct applications: (i) supervised mode, which involves automatically building and evaluating models for nitrogen cycling families and (ii) unsupervised mode, which involves *de novo* clustering of similar proteins. For the unsupervised mode, we additionally clustered a larger dataset of ~50 000 protein sequences comprising 10 complete genomes.

2.2.1. Data description

For both applications delineated above, we applied Snekmer to an example dataset of sequences belonging to 33 nitrogen cycling families. Each family contains between 13 and 390 sequences for a total of 5530 sequences (Supplementary Table S1). We used 33 previously described HMMs (Haft *et al.*, 2013; Mistry *et al.*, 2021; Nelson *et al.*, 2020) and the hmmer package (Eddy, 2011) to identify proteins involved in nitrogen transport or transformation from a set of 1835 representative and reference complete genomes from the NCBI RefSeq database (generated in 2018) (O'Leary *et al.*, 2016) (Supplementary Table S1). To determine nitrogen cycle family genes, we searched the NCBI using HMMer (version 3.3.2) (Eddy, 2011) with trusted cut-off scores from Pfam (Mistry *et al.*, 2021). All protein sequences used for modeling are included with the code as an example application.

We also analyzed a dataset comprised of 10 complete genomes from the SwissProt/UniProt database (Duvaud *et al.*, 2021). The list of genomes can be found in Supplementary Table S2. The dataset contains 46 908 protein sequences. Uniprot annotations (GO, Pfam, Interpro, TIGRFams, COG and OrthoDB) were obtained for each genome to evaluate clustering results.

2.2.2. Example Application 1: Automated training and evaluation of models for nitrogen cycling protein families

We evaluated Snekmer's supervised operation mode in automatically generating kmer-based classification models for protein families in the nitrogen cycling protein set. The supervised mode is useful when available sets of sequences are known to be functionally and/or evolutionarily related. The nitrogen cycling families used for these examples were identified using traditional sequence similarity methods and thus represent a useful example set. Our previous work has shown that AAR can be used to develop models for families which are functionally related, but where members lack significant sequence similarity with each other (McDermott *et al.*, 2019). Resulting models can then be applied to sequence collections (e.g. genomes or metagenomes) to provide predictions for the models generated in the training phase.

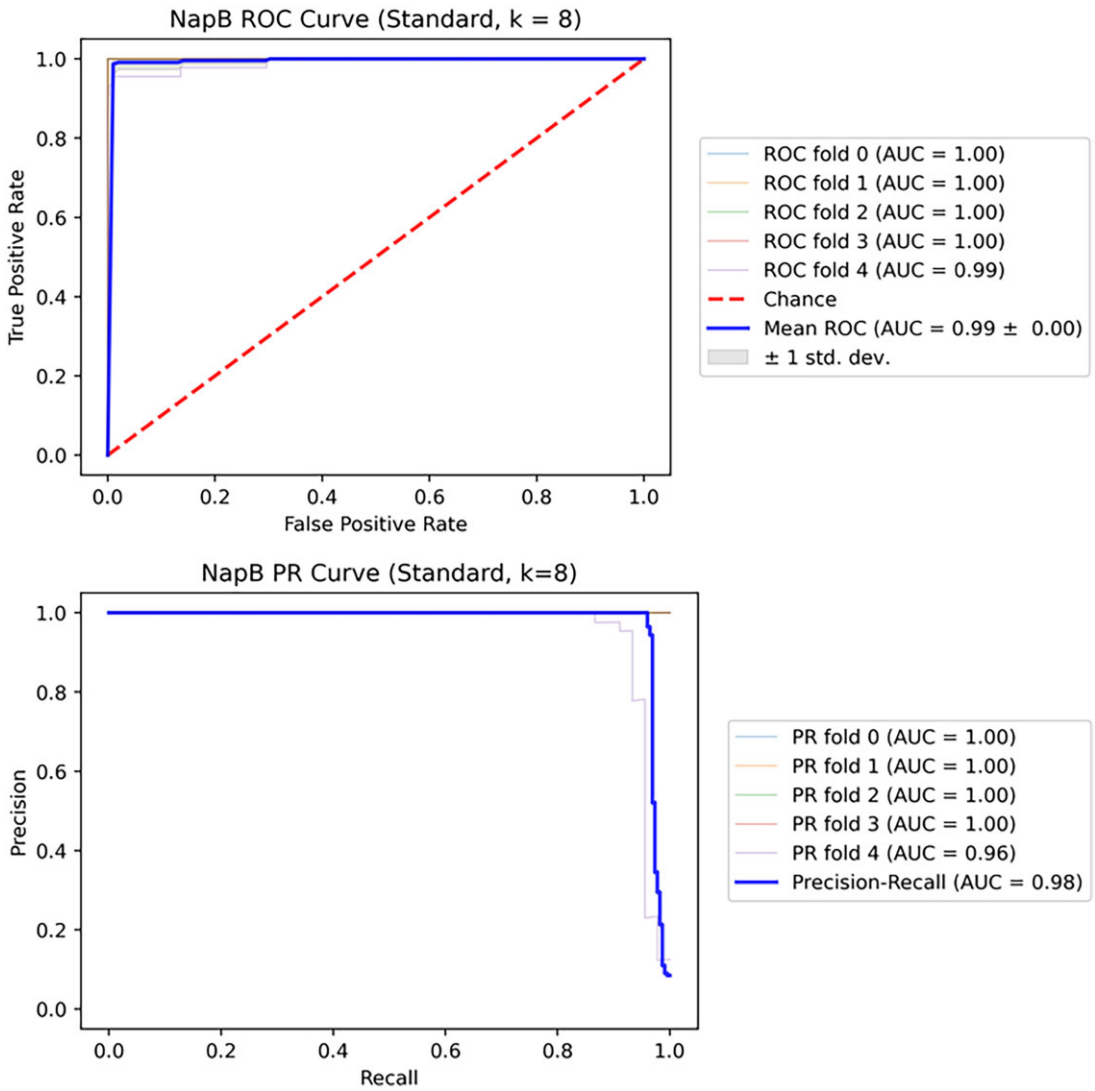

Fig. 2. AUROC (left) and AUPR (right) curves, with 5-fold cross-validation, for a NapB protein family classification model using AAR with the standard alphabet and kmer length of 8

Table 2. Description of metrics describing the performance of binary classification models

Metric	Method
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$
Sensitivity (true positive rate or recall)	$\frac{TP}{TP+FN}$
Specificity (true negative rate)	$\frac{TN}{TN+FP}$
False positive rate	$\frac{FP}{TN+FP}$
Precision	$\frac{TP}{TP+FP}$
Area under receiver-operator characteristic curve	Plot: true positive rate versus false positive rate
Area under precision-recall curve	Plot: precision versus recall

Note: Note that P indicates positive examples and N indicates negative examples. T and F represent true and false assignments, respectively. Thus, TP indicates true positives assigned by the model.

2.2.2.1 Model construction. We used Snekmer to evaluate all included recoding alphabets (Table 1) for kmers of lengths between 4 and 16. Probability scores were then used to train logistic regression binary classification models for in-family versus out-of-family assignment for each of the 31 protein families.

2.2.2.2. Model evaluation. We evaluated the performance of Snekmer for automatically constructing classifiers for the 33 protein families using 5-fold cross-validation. The results of this analysis are shown for a range of AAR schemes and lengths of k in Figure 3. The corresponding overall AUC PR plots and plots showing the performance in individual families are included as Supplementary Figure S1.

For most of the protein families, the models developed via Snekmer perform well, but performance of individual families varied (Supplementary Fig. S1). The AAR alphabets with higher complexity (i.e. greater number of encodings)—Standard, MIQS and no encoding—performed best overall, showing consistently high PR AUC and AUC ROC across multiple k values (Fig. 3). We note that overall, the classifiers perform well with most alphabet and k combinations, speaking to the highly specific and differentiated nature of the original dataset.

However, while the per-alphabet differences in AUC are pronounced at low k values, as k increases the AUCs are similar between simple (Hydro, Hydrocharge and Hydrostruct) and complex alphabets (Standard, MIQS and no encoding). This indicates a tradeoff in specificity, with the simpler alphabets requiring longer kmers to attain specificity for the families, whereas the more complex alphabets display peak performance at short kmer lengths. In other words, higher-complexity alphabets paired with longer kmer

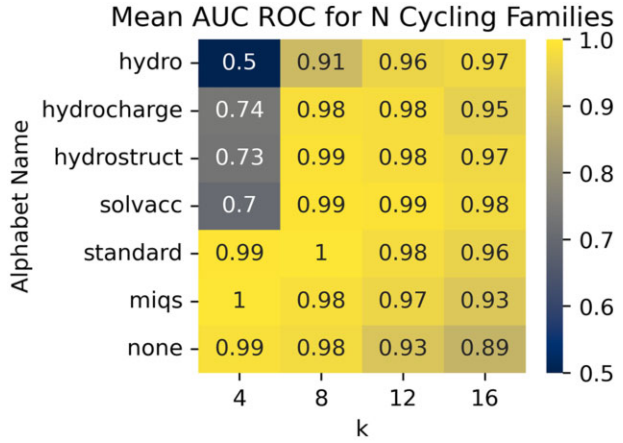

Fig. 3. Overall performance of automatically generated supervised models from Snekmer. Models were generated by Snekmer as described and 5-fold cross validated. The mean AUC ROC across all 33 families for each alphabet/k combination is shown

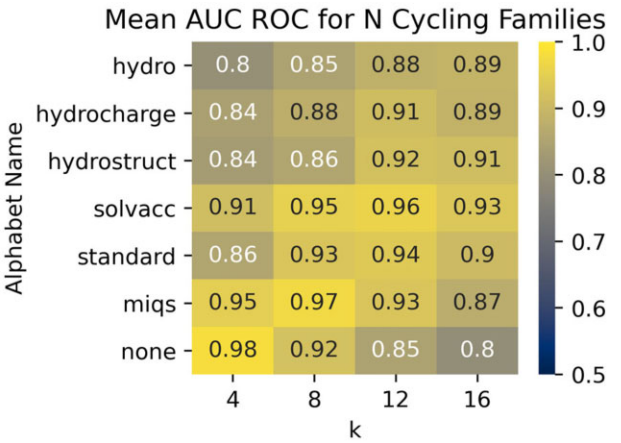

Fig. 4. Overall performance of *de novo* determined relationships between nitrogen cycling family members from Snekmer. Feature vectors generated by Snekmer were used to determine similarity between proteins. The AUC ROC was calculated by considering each similarity relationship as positive if it links two members of the same family and negative if it links members of different families. The mean AUC ROC over all 33 families is shown

lengths are more prone to overfitting, and conversely lower-complexity alphabets paired with shorter kmer lengths result in an overabundance of detected kmers with little diagnostic value.

The best-performing combination across all protein families that we examined was k of 4 with MIQS AAR or a k of 8 for the standard chemistry encoding, for which the majority of the models had an ROC AUC of 1.0 (perfect classification). However, we note that many encoding/k combinations performed in the range of 0.98–1.0 over the families, indicating many possible options for encoding that may work well across different families (Supplementary Fig. S2). We include the complete results of our analysis as Supplementary Table S1. Our results indicate that while for the studied protein families, a shorter k with a more complex alphabet may be the optimal choice, other alphabet combinations can achieve similar classification performance. Overall, as observed previously (McDermott *et al.*, 2019), certain families present challenges to this simple approach and may necessitate more flexible AARs for classification.

One significant difference between parameters, however, is the computational requirement. For instance, models built from MIQS recoded 4-mers exhibit comparable performance as the unencoded 4-mers, but the feature generation step for the MIQS recoded sequence is nearly 4 times faster (Supplementary Table S2). The differences in calculation time scale with k-mer length, presenting a significant obstacle for large datasets.

2.2.3. Example Application 2: Unsupervised clustering of nitrogen cycling protein families

For the second example application, we used Snekmer to cluster protein sequences in an unsupervised manner based on AAR kmer profile similarity. This application can enable exploration of the protein space for largely uncharacterized sequences, such as metagenome sequences. Resulting protein clusters should represent similar functional groups for further evaluation.

2.2.3.1. Unsupervised operation. We first generated kmer features from input protein sequences belonging to the nitrogen cycle families used in Example 1 and created feature vectors for each protein encompassing all kmers occurring more than once in the dataset. We then calculated Pearson correlation coefficients between all pairs of feature vectors.

2.2.3.2. Unsupervised operation evaluation. To evaluate Snekmer's capacity to identify clusters that match functional families, we used the original nitrogen cycle family designations to calculate the ROC AUC for each family using the correlation values between proteins in the same family ('positive' predictions) compared to correlation values between a protein in the family and one not in the family ('negative' predictions). Since, the original labels used were generated using HMM models, the average AUC values here represent agreement with those models. As can be seen in Figure 4, with results from individual families shown as Supplementary Figure S3, the Snekmer unsupervised mode performs quite well, with average AUC scores of >0.9 in more than half of the examined alphabet and kmer length combinations. Thus Snekmer-generated clusters match the performance of the original HMMs, and roughly mirror results from the Snekmer-generated models in Example 1 in terms of AUCs for this controlled clustering example.

Though the example clusters perform well, we highlight that the well-annotated test dataset are not representative of a typical dataset necessitating the unsupervised mode, which would likely contain little-to-no previous understanding of the protein family structure. Therefore, we used the correlation matrix to generate edges with a minimum correlation value threshold and visualized the resulting network, coloring nodes by their respective families (Fig. 5). The network is clearly organized into clusters that faithfully represent the original nitrogen cycle families and could be easily separated by applying a community detection algorithm to the network. We varied the correlation threshold and show that at a more conservative correlation value of 0.5 the families are distinct, unconnected subnetworks (Supplementary Fig. S4). We also performed the same clustering using the solvent accessibility alphabet and a k of 12, which performed well in the correlation analysis (Fig. 4) and show the resulting network with a correlation threshold of 0.1 as Supplementary Figure S5. These results show that the AARs produce similar clustering results as the unencoded kmers, but at a significantly reduced computational cost.

2.2.4. Example Application 3: Unsupervised clustering of multiple genomes

Though Snekmer performed well for the nitrogen cycle families example, we were interested in evaluating Snekmer's clustering performance on larger datasets that more closely mirror datasets of interest to potential users, and benchmarking our method versus state-of-the art sequence clustering methods. Accordingly, we gathered 46 908 protein sequences from 10 randomly selected genomes from the SwissProt/UniProt database (Duvaud *et al.*, 2021). We compared the performance of different parameter selections by assessing the homogeneity and completeness of the resulting clusters when evaluated against annotations available from Uniprot,

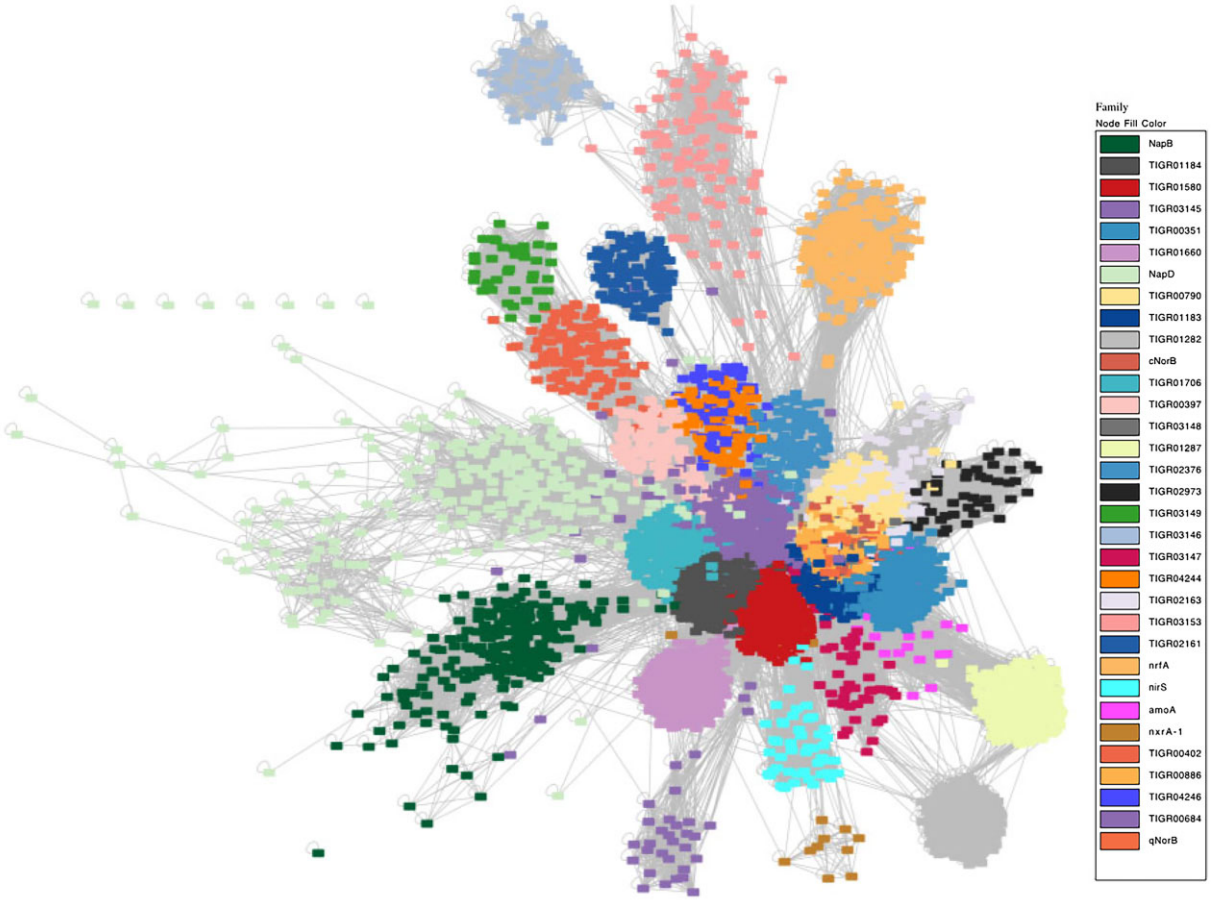

Fig. 5. Network representation of the *de novo* similarity relationships determined by Snekmer ($k = 4$, no encoding) showing the nitrogen cycling families as different colors (legend). Nodes represent proteins and edges represent similarity relationships with a correlation of 0.1 or greater as determined by comparing kmer vectors for both proteins

including Gene Ontology (GO), Pfam, Interpro and Clusters of Orthogolous Genes (COG).

Using parameter combinations derived from the best-performing clusters from our second example (Fig. 4), we chose k = 4 and the MIQS alphabet, $k = 8$ with the solvent accessibility alphabet, and $k = 14$ with the hydrophobicity alphabet. We applied Snekmer in cluster mode to the sequences using Jaccard similarity to determine the distance matrix and Scikit-Learn's agglomerative clustering using complete linkage. After some parameter exploration, we settled on a distance threshold for the agglomerative clustering of 95. To compare our results with an existing clustering method, we also applied MMSeqs2 to the same protein set using the 'easy cluster' mode with the default sensitivity (Mirdita *et al.*, 2019). The average V score across all clusters shows that Snekmer clustering has the best performance when $k = 14$ with a hydrophobic AAR. These results are somewhat lower than those with MMseqs2 (Table 3), though still show Snekmer can perform reasonably well for annotation purposes.

To further investigate the differences between Snekmer clustering and MMseqs2, we examined the overlap between proteins assigned to a cluster containing at least two sequences as determined by each method. Surprisingly, we found that although each method clustered approximately 50–60% of the proteins, roughly 60% of the proteins clustered were not shared between the two (Fig. 6). Integrating Snekmer and MMSeqs2 clustering results increases the fraction of total proteins assigned to clusters from ~50% to 60% by either method alone, to 85% using both methods together.

The results we present are compared with MMSeqs2 clustering run with default parameters but see little difference when we alter sensitivity of MMSeqs2. MMSeqs2 clustering using sensitivities in the range of 1 (least) to 7.5 (most) sensitive show that the results vary little either in terms of protein coverage, from a minimum of 20 240 to a maximum of 28 977, or clustering performance, with a Pfam V score minimum of 0.90 and a maximum of 0.92.

Table 3. V-score performance of clustering on 46 908 proteins

	Snekmer			
Annotation	k4, MIQS	k8, SolvAcc	k14, Hydro	MMSeqs2
GO	0.599	0.671	0.702	0.783
Pfam	0.759	0.820	0.854	0.919
InterPro	0.729	0.790	0.824	0.883
TIGRFAMs	0.807	0.899	0.921	0.966
COG	0.816	0.874	0.853	0.922
OrthoDB	0.850	0.900	0.894	0.943

To evaluate Snekmer's ability to predict annotations for unannotated proteins, we evaluated the number of clusters containing at least one annotated protein and at least one unannotated protein. These clusters would be candidates for transitive annotation based on the annotated protein association. Notably, though the overall performance metrics for Snekmer and MMseqs2 are similar, Snekmer identifies nearly twice as many clusters as MMseqs2, and the proportion of mixed clusters is higher, indicating an increased number of annotation predictions for Snekmer (Table 4). Determining whether these predictions are true positives or false positives is the subject of ongoing work.

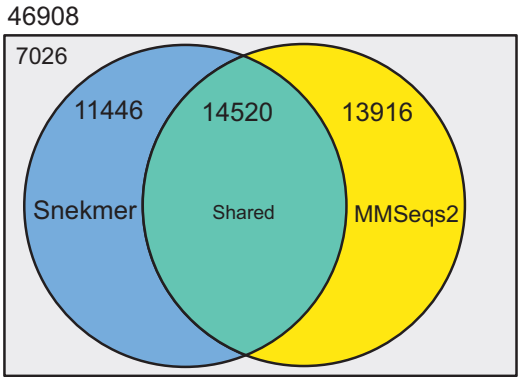

Fig. 6. Coverage of clustering methods on a multiple genome protein set. The entire set (box) is indicated and the coverage of the Snekmer clustering (left circle), MMSeqs2 (right circle) and shared (middle area) are shown with numbers indicating the number of proteins in the groups that are unique to each method or shared between the two

Table 4. Summary of clusters with annotations

	Snekmer		MMSeqs2	
Annotation	Annotated	Mixed	Annotated	Mixed
GO	9645	4235	4918	571
Pfam	10 327	4030	5406	362
InterPro	10 782	3500	5709	135
TIGRFAMs	3833	2574	1891	677
COG	1851	1579	1395	1243
OrthoDB	2265	2050	2109	2017

Note: Annotated indicates the number of clusters with at least one annotation of that type. Mixed indicates the number of clusters with at least one annotation, and one unannotated protein in the same cluster.

3. Limitations and future directions

Snekmer is a tool designed to be modular and flexible in order to enable easy development of (i) predictive models for protein families and (ii) clusters of moderate-sized protein sets. Its integration of Snekmer further allows simple deployment of Snekmer on high-performance computing platforms for the training of large sets of models. However, Snekmer currently demonstrates some limitations in speed and memory usage, particularly in clustering very large datasets. In future development of Snekmer, we plan to optimize clustering and model-building to improve Snekmer's ability to model and cluster large datasets. We also plan to provide more support for user-associated metadata for trained models.

4. Conclusions

We describe Snekmer, a Snakemake pipeline to perform AAR and kmer representation for protein sequences. We have previously used this approach to develop a machine-learning model for ubiquitin ligase mimics from pathogens (McDermott *et al.*, 2019), as well as related machine-learning methods to predict type III secreted effectors (McDermott *et al.*, 2011; Samudrala *et al.*, 2009), and multi-drug efflux transporters (McDermott *et al.*, 2015) in bacteria. The current work enables broader adoption of AAR-based workflows and high-throughput kmer-based sequence analysis. The Snekmer pipeline is available on GitHub (http://github.com/PNNL-CompBio/Snekmer).

Previously, we demonstrated that the simpler AAR schemes (e.g. separating amino acids into hydrophobic and hydrophilic groups) outperformed the native protein sequence kmers for classification of the sequence-diverse ubiquitin ligase proteins (McDermott *et al.*, 2019). In the current work, using Snekmer to apply the AAR approach to a variety of protein families in an automated fashion, we found that many protein families are accurately classified using the native protein sequence kmers. However, some of families are classified more accurately using AAR, indicating its utility. Because AAR captures greater flexibility in sequence space (McDermott *et al.*, 2019; Yamada and Tomii, 2014), the approach will likely be more robust to introduction of new sequences into a family.

We describe two applications of Snekmer workflows: (i) supervised automatic construction of classification models for input protein families and (ii) unsupervised clustering of protein sequences based on recoded kmers. In both cases, Snekmer produces high-quality results, either in the form of automatically generated models that can predict the function of protein families with high accuracy or identified protein clusters that reproduce the known family structure of the underlying sequences. For clustering, Snekmer can generate clusters of comparable quality to MMseqs2, but that covers a different subset of protein space, demonstrating the value of our approach as a supplement to the widely used MMseqs2. We note that while the flexible architecture of Snekmer currently limits the size of input protein files that can be reasonably executed on standard computing environments, especially for the clustering mode, future development will be aimed toward enabling faster clustering for larger protein files. We believe researchers studying novel genomes and microbial communities will find Snekmer valuable in building predictive models for functionally related proteins.

Funding

This work was supported by the Department of Energy (DOE) Office of Biological and Environmental Research (BER) and is a contribution of the Scientific Focus Area 'Persistence Control of Engineered Functions in Complex Soil Microbiomes', the DOE/BER 'Improved Protein Annotation in KBase Using Machine Learning, Multi-Omics Data Integration, and Structural Prediction' project, the DOE/BER 'Machine-Learning Approaches for Integrating Multi-Omics Data to Expand Microbiome Annotation' project, NSF [1856556], the Defense Threat Reduction Agency, under Interagency Agreement [DTRA13081–37739], and the Proxy Applications for Converged Workloads (PACER) project, a component of the Laboratory Directed Research and Development Program at Pacific Northwest National Laboratory. PNNL is operated for the DOE by Battelle Memorial Institute under Contract [DE-AC05-76RL01830].

Conflict of Interest: none declared.

Data availability

Example data as well as code are available at http://github.com/PNNL-CompBio/Snekmer.

References

Arnold,R. *et al.* (2009) Sequence-based prediction of type III secreted proteins. *PLoS Pathog.*, **5**, e1000376.

Bacardit,J. *et al.* (2009) Automated alphabet reduction for protein datasets. *BMC Bioinformatics*, **10**, 6.

Bateman,A. *et al.* (2000) The Pfam protein families database. *Nucleic Acids Res.*, **28**, 263–266.

Buchfink,B. *et al.* (2021) Sensitive protein alignments at tree-of-life scale using DIAMOND. *Nat. Methods*, **18**, 366–368.

Davis,J. and Goadrich,M. (2006) The relationship between precision-recall and ROC curves. In: *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, pp. 233-240. Association for Computing Machinery, New York, NY.

Duvaud,S. *et al.* (2021) Expasy, the Swiss Bioinformatics Resource Portal, as designed by its users. *Nucleic Acids Res.*, **49**, W216–W227.

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

Eddy,S.R. (2011) Accelerated profile HMM searches. *PLoS Comput. Biol.*, **7**, e1002195.

Edwards,R.A. *et al.* (2012) Real time metagenomics: using k-mers to annotate metagenomes. *Bioinformatics*, **28**, 3316–3317.

Gruning,B. *et al.*; The Bioconda Team. (2018) Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat. Methods*, **15**, 475–476.

Haft,D.H. *et al.* (2013) TIGRFAMs and genome properties in 2013. *Nucleic Acids Res.*, **41**, D387–395.

Harris,C.R. *et al.* (2020) Array programming with NumPy. *Nature*, **585**, 357–362.

Hauswedell,H. *et al.* (2014) Lambda: the local aligner for massive biological data. *Bioinformatics*, **30**, i349–i355.

Koster,J. and Rahmann,S. (2018) Snakemake-a scalable bioinformatics workflow engine. *Bioinformatics*, **34**, 3600.

McInnes, L. *et al.* (2018) UMAP: uniform manifold approximation and projection. *J. Open Source Softw.*, **3**, 861.

McInnes, L. *et al.* (2017) hdbscan: hierarchical density based clustering. *Open J.*, **2**, 205.

Lee,J.Y. *et al.* (2018) Blazing signature filter: a library for fast pairwise similarity comparisons. *BMC Bioinformatics*, **19**, 221.

Liang,Y. *et al.* (2022) Research progress of reduced amino acid alphabets in protein analysis and prediction. *Comput. Struct. Biotechnol. J.*, **20**, 3503–3510.

Lobb,B. *et al.* (2020) An assessment of genome annotation coverage across the bacterial tree of life. *Microb. Genom.*, **6**, e000341.

McDermott,J.E. *et al.* (2015) Prediction of multi-drug resistance transporters using a novel sequence analysis method. *F1000Res.*, **4**, 60.

McDermott,J.E. *et al.* (2011) Computational prediction of type III and IV secreted effectors in gram-negative bacteria. *Infect. Immun.*, **79**, 23–32.

McDermott,J.E. *et al.* (2019) Prediction of bacterial E3 ubiquitin ligase effectors using reduced amino acid peptide fingerprinting. *PeerJ*, **7**, e7055.

McKinney,W. (2010) Data structures for statistical computing in Python. In: van der Walt, S.J. and Millman, J., (eds) *Proceedings of the 9th Python in Science Conference*, Austin, TX. pp. 56-61.

Mirdita,M. *et al.* (2019) MMseqs2 desktop and local web server app for fast, interactive sequence searches. *Bioinformatics*, **35**, 2856–2858.

Mistry,J. *et al.* (2021) Pfam: the protein families database in 2021. *Nucleic Acids Res.*, **49**, D412–D419.

Nelson,W.C. *et al.* (2020) Distinct temporal diversity profiles for nitrogen cycling genes in a hyporheic microbiome. *PLoS One*, **15**, e0228165.

O'Leary,N.A. *et al.* (2016) Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res.*, **44**, D733–745.

Overbeek,R. *et al.* (2014) The SEED and the rapid annotation of microbial genomes using subsystems technology (RAST). *Nucleic Acids Res.*, **42**, D206–214.

Pedregosa,F. *et al.* (2011) Scikit-learn: machine learning in python. *J. Mach. Learn. Res.*, **12**, 2825–2830.

Salzberg,S.L. (2019) Next-generation genome annotation: we still struggle to get it right. *Genome Biol.*, **20**, 92.

Samudrala,R. *et al.* (2009) Accurate prediction of secreted substrates and identification of a conserved putative secretion signal for type III secretion systems. *PLoS Pathog.*, **5**, e1000375.

Steinegger,M. and Söding,J. (2017) MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.*, **35**, 1026–1028.

Yamada,K. and Tomii,K. (2014) Revisiting amino acid substitution matrices for identifying distantly related proteins. *Bioinformatics*, **30**, 317–325.