# **Supplementary material for:**

**DDAP: docking domain affinity and biosynthetic pathway prediction tool for type I polyketide synthases**

Tingyang Li1, Ashootosh Tripathi2,3, Fengan Yu2, David H. Sherman2,3,4 and Arvind Rao1,\*

1Department of Computational Medicine and Bioinformatics, 2Life Sciences Institute, 3Department of Medicinal Chemistry, 4Department of Chemistry, Department of Microbiology and Immunology, University of Michigan, Ann Arbor, Michigan, USA

\*To whom correspondence should be addressed.

# **1 Methods**

## 1.1 Positive and Negative Samples

The PKS pathway data used in this study was obtained from two sources: (i) the work reported by Thattai *et al.*, 2007, (ii) a manual literature review, guided by the MIBiG database (Version 1.4 and 1.3) (Medema *et al.*, 2015). PKS sequences were downloaded from the NCBI database (Benson *et al.*, 2017). Docking domain (DD) sequences were extracted from the N-/C- terminus of the PKS proteins. One hundred amino acid (AA) residues at the C-terminus of each PKS protein were extracted as the Head (or C-terminal DD) except the last protein in the pathway (which includes the thioesterase domain). Fifty AAs at the N-terminus of each PKS protein were extracted as the Tail (or N-terminal DD) except the first protein in the pathway (which includes the loading module). The lengths of the Heads and Tails were decided based on: (i) the implementation of antiSMASH, where 50 AAs at the N-terminus and 100 AAs at the C-terminus are extracted as DDs (Blin *et al.*, 2019), (ii) the work reported by Broadhurst et al. where it was revealed that the C-terminal DDs usually comprise 80-100 residues and N-terminal DDs usually comprise 30-40 residues (Broadhurst *et al.*, 2003). Previous publications indicated that C-terminal DDs of class II and class III are shorter than that of class I DD (Whicher *et al.*, 2013; Moss *et al.*, 2019). Considering that machine learning methods are in general tolerable for redundant information, we believe that extracting 100 AAs as C-terminal DD and 50 AAs as N-terminal DD fits our application.

In this study, the ground truth for docking domain (DD) affinity was generated based on the polyketide synthase (PKS) pathway information. Positive samples (encoded as 1) were defined as interacting Head-Tail (HT) pairs (**Figure 1A**). Negative samples (encoded as 0) were defined as Head-Tail pairs that were in the same pathway but not interacting. This notion was also adopted in the study by Thattai et al. (Thattai *et al.*, 2007). Since the total number of negative samples greatly exceeded the total number of positive samples, we randomly subsampled the negative samples to match the size of positive samples to ensure balance.

## 1.2 Model Selection

During the development stage of DDAP, the dataset was divided into two portions: the training data (85/92 pathways, 714/764 DDs) and the hold-out testing data (7/92 pathways, 50/764 DDs). The training data came from MIBiG Version 1.3 and the Thattai *et al.* dataset (Thattai *et al.*, 2007), which consists of PKS pathways reported before 2016. The hold-out testing data came from MIBiG Version 1.4, which consists of PKS pathways reported between 2016 and 2018. In this study, the names of the PKS pathways were coded as BGC\_n, where n was an integer starting from 1. BGC\_1 through BGC\_85 were used for model selection. BGC\_86 through BGC\_92 were the hold-out testing samples. Eventually, the best performing model was trained using all available data (92/92 pathways, 764/764 DDs) and was integrated into DDAP. A mapping scheme from the above IDs to the sequences can be found in the DDAP database.

Model selection was performed through repeated five-fold cross-validation (CV). The area under the receiver operating characteristic (ROC) curve (AUC) was used to assess the DD affinity prediction performance. In CV, the training dataset was randomly partitioned into five approximately equally sized sets, where HT pairs in the same pathway were always allocated to the same set. The five sets were then sequentially used as the validation set, while the remaining four sets were used as the training set. The algorithm learns from the training set and uses the validation set to check the accuracy by calculating the AUC. The above process was repeated five times for each candidate model. In total, AUC scores were calculated. The mean of 25 AUCs was used to represent the performance of the candidate model. The Random Forest algorithm implemented by the python library scikit-learn (0.19.1) (Pedregosa *et al.*, 2011) was used as the base learner. Parameters were kept as default, except “n\_estimators” (set to 200), “random\_state” (set to 0), and “max\_depth” (a hyperparameter). In the following text, the CV results for the grid search of “max\_depth” is omitted.

## 1.3 Model 1: Sequence Similarity Based Model

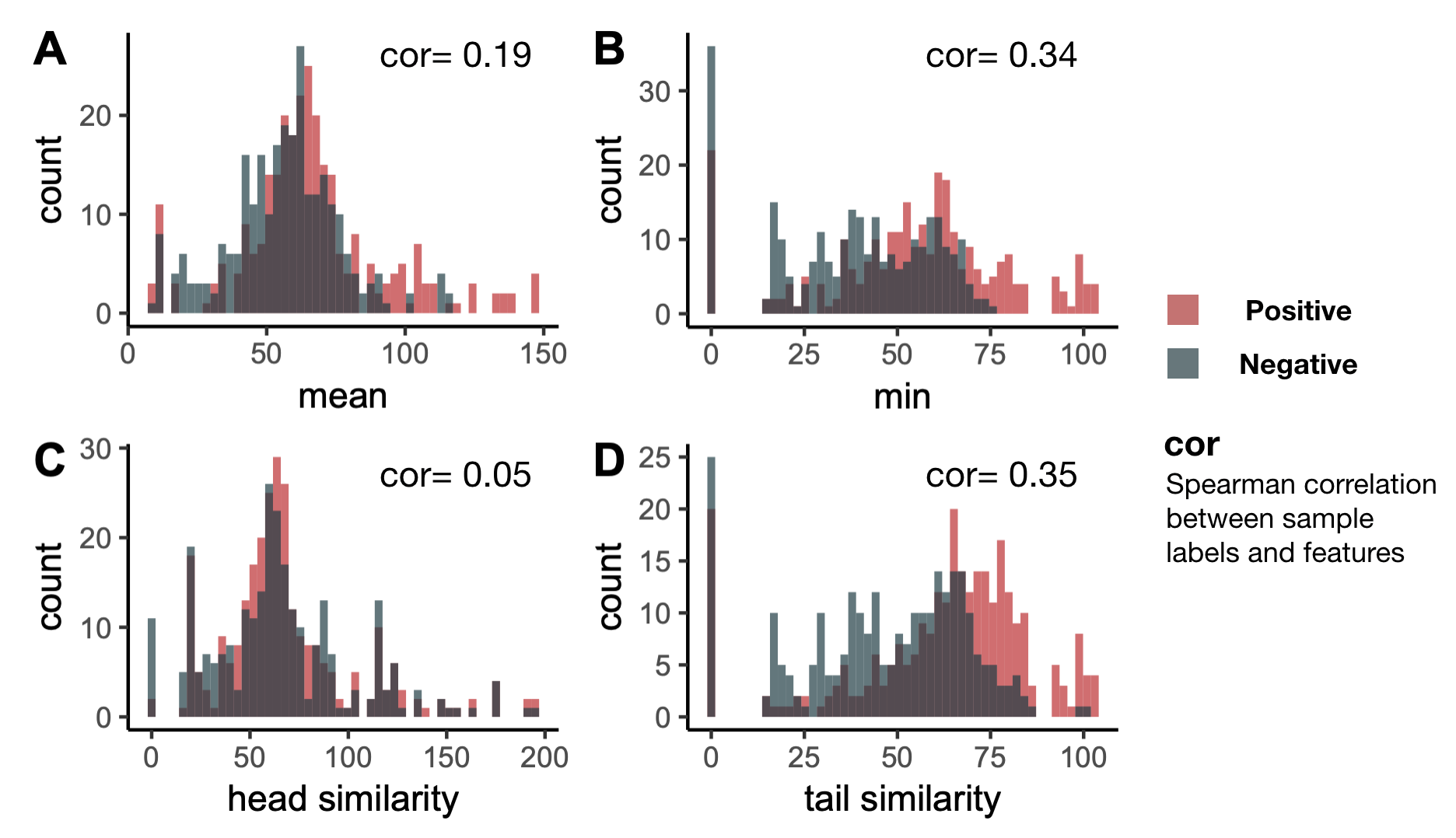
### 1.3.1 Method

The underlying assumption of Model 1 is that if an unknown HT pair is similar to an interacting HT pair, the unknown HT pair is then more likely to interact. This assumption is supported by the study of Thattai et al. (Thattai *et al.*, 2007), where it was found that Heads and Tails could be grouped into mutually incompatible clusters based on phylogenetic analysis. For each testing HT pair, Model 1 first calculates the sequence similarity between the unknown Head and all the Heads in the training set, and the similarity between the unknown Tail and all the Tails in the training set (**Figure 1B**). In this study, we used bit-scores calculated by “blastp” in BLAST 2.7.1 (Camacho *et al.*, 2009) to represent sequence similarity. Bit-score is a normalized alignment score that can be used as a measurement for sequence similarity (Altschul *et al.*, 1997). Higher bit-scores represent higher similarity. For each Head-Tail pair in the training set, we further calculated the mean and minimum of the Head-Head similarity and the Tail-Tail similarity. Each interacting HT pair (i.e., each positive sample) in the training set is therefore associated with four scores: (a) the Head-Head similarity score, (b) the Tail-Tail similarity score, (c) the average of (a) and (b), (d) the minimum of (a) and (b).

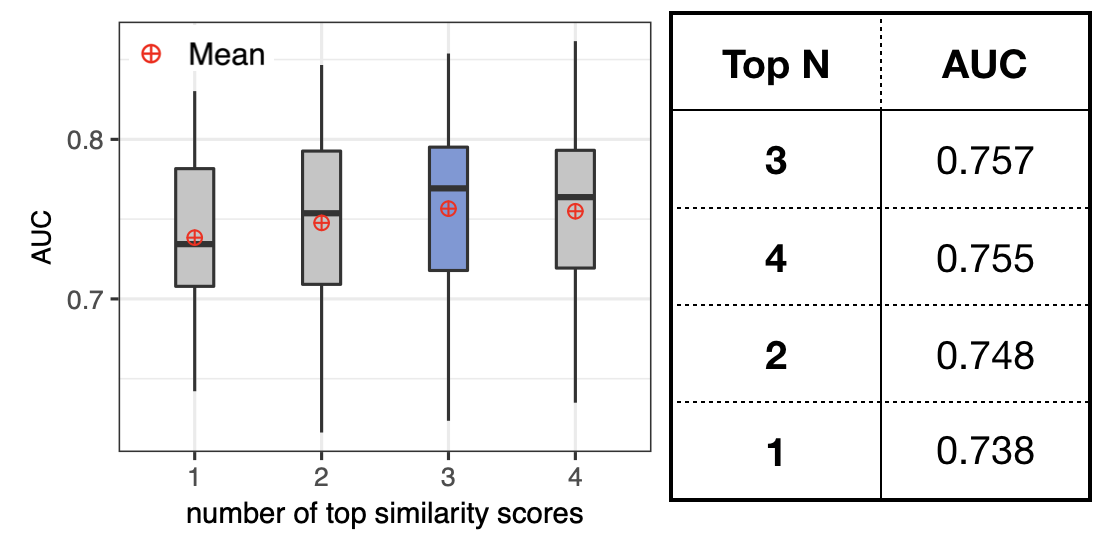
After obtaining the four metrics, we ranked training HT pairs according to (c) in descending order. HT pairs at the top of the list were considered most similar to the unknown (i.e., testing) HT pair. We then used (a), (b), (c) and (d) scores associated with the top HT pairs as features (**Figure 1B**). Each testing sample was thus represented by numerical features, where was a hyperparameter optimized through cross-validation. When we calculated features for the training samples, each pathway was treated as a unit, which means that the within-pathway similarity scores were not considered. Considering only inter-pathway similarity ensures that the distribution of features in the training set resembles that of the testing set.

### 1.3.2 Model 1 Cross-Validation Results

The Spearman correlation between the four metrics calculated in Model 1 and the ground truth is shown in **Figure S1**. It is observed that the minimum of two similarity scores and the Tail-Tail similarity score have relatively stronger correlation with the ground truth, where positive samples (interacting HT pairs) tend to have higher similarity values. The cross-validation results for Model 1 are shown in **Figure S2**, where we tested . Eventually, received the highest performance. It is expected that more features receive higher performance. In this case, the improvement from to is insignificant, therefore, we consider that the model performance converges at and .



**Figure S1. Histograms of the four metrics calculated in Model 1. A.** Histogram of the mean of the Head-Head similarity and Tail-Tail similarity. **B.** Histogram of the minimum of the Head-Head similarity and Tail-Tail similarity. **C.** Histogram of Head-Head similarity **D.** Histogram of Tail-Tail similarity. “Positive” refers to interacting HT pairs. “Negative” refers to non-interacting HT pairs.

****

**Figure S2. A boxplot of the cross-validation results for Model 1.** Each box shows a five-number summary of the cross-validation result of one value of . The AUC in the table is the mean of AUCs in the cross-validation.

## 1.4 Model 2: Cognate Residue Based Model

### 1.4.1 Method

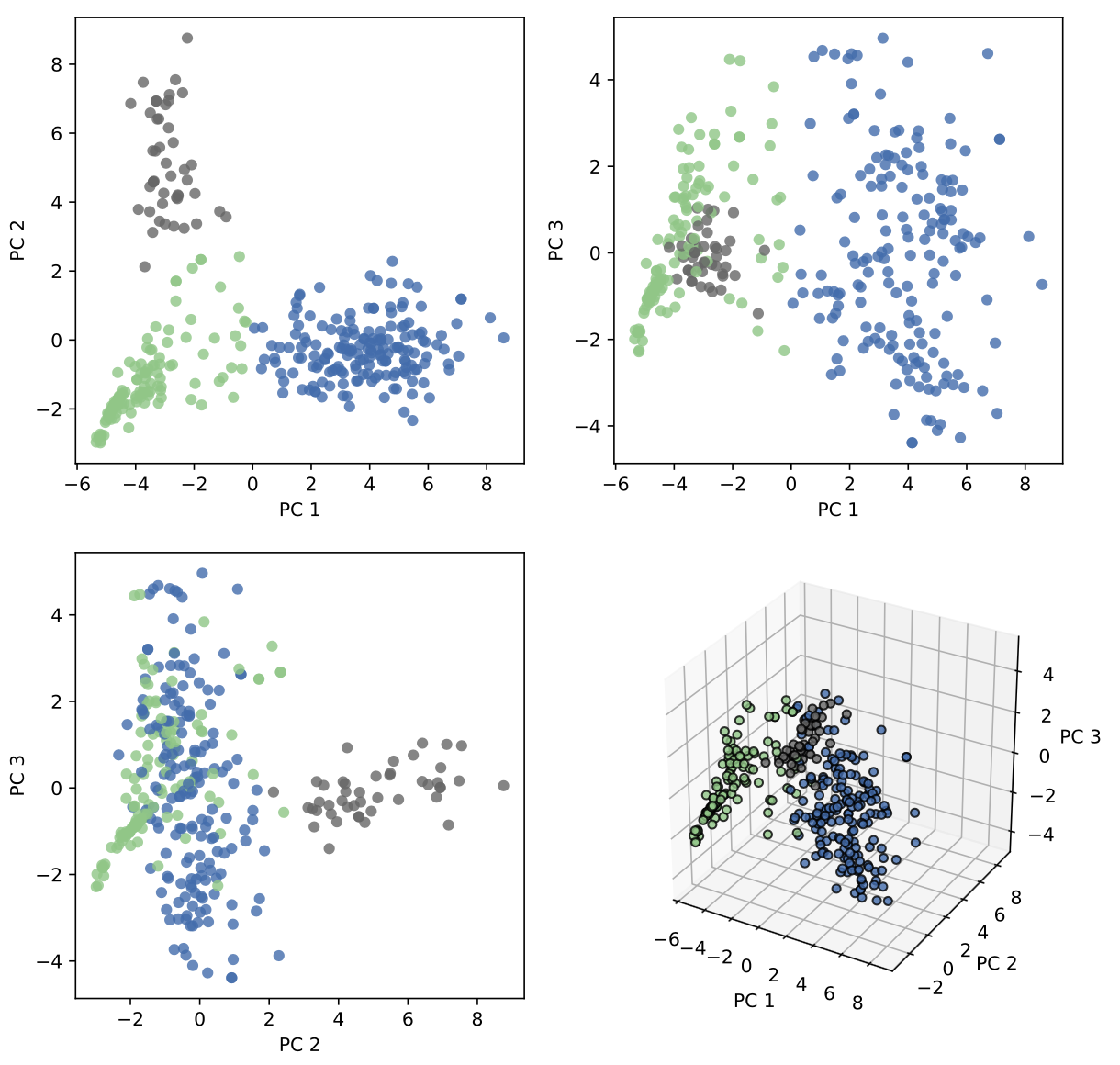
Previous structural studies on docking domains revealed that key residues (or cognate residues) at the interface of two interacting DD proteins are playing important roles in determining the specificity of the interaction and stabilizing the bond (Broadhurst *et al.*, 2003; Buchholz *et al.*, 2009). Yadav *et al.* incorporated the above knowledge into their method, assuming the structure of DEBS DD could be generalized to all types of DDs (Yadav *et al.*, 2009). Later on, the crystal structure of type II DD was resolved (Whicher *et al.*, 2013), which was found to have cognate residues at different positions compared to type I DD. On the other hand, Model 2 strives to make use of the knowledge about cognate residues without assuming the locations of the cognate residues. Model 2 contains the following assumptions: (a) cognate residues in Heads and Tails determine the specificity of Head-Tail interactions; (b) different classes of DDs have different cognate residues; (c) for an interacting HT pair, correct cognate residues in the Head “match” with cognate residues in the Tail.

In this study, we assume a favorable “match” happens when a positively charged amino acid (AA) meets a negatively charged AA. As shown in **Figure 1C**, matched and unmatched residue pairs can be represented by 1 or 0 in a 2-D matrix, where “1” marks the locations of matched residue pairs. For example, “1” at location means the second AA in the Head matches with the fifth AA in the Tail. After generating the 2-D representation of each pair of DDs, we used k-means to cluster these 2-D matrices into k clusters (k is a hyperparameter) to address assumption (b). The 2-D matrices are turned into 1-D vectors before the k-means clustering.

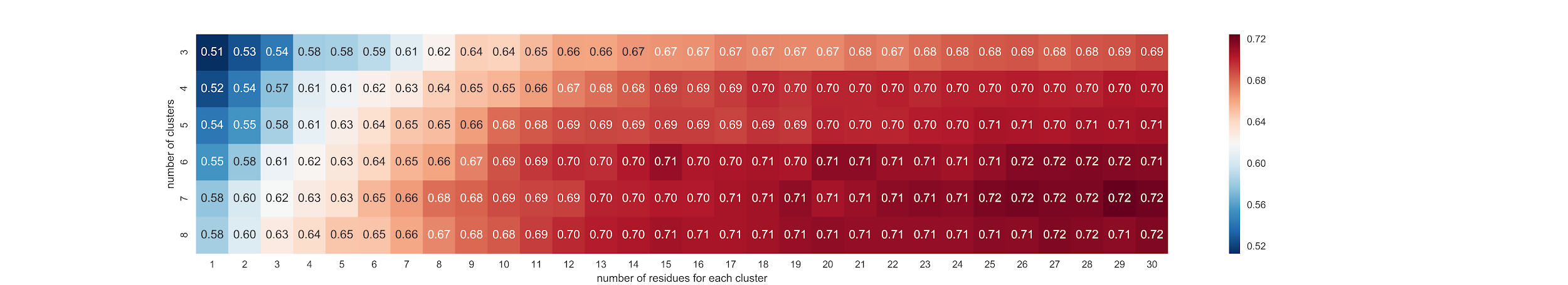
After clustering, HT pairs with different cognate residues were allocated to different clusters. In order to find cognate residues within each cluster, we ranked each location in the 2-D matrix by the frequency of observing a “1” at that position, which can be calculated as:  
, where N is the total number of HT pairs. The locations with the highest frequencies were picked as cognate residue pairs for that particular cluster. The total number of cognate residues in a cluster is a hyperparameter. We used a combination of cognate residue pairs from all clusters as the features for Model 2. Repeated cognate residue pairs were removed.

### 1.4.2 Model 2 Cross-Validation Results

We used Principal Component Analysis (PCA) to visualize the clustering result in 2-D and 3-D space (**Figure S3**). Although our approach is different from the phylogenetic approach that Thattai et al. utilized and we used a different dataset, we still observed that the docking domain pairs formed three large clusters. **Figure S4** shows the cross-validation result for the grid search of the number of HT pair clusters () and the number of cognate residues in each cluster (). As the two parameters increase, the mean AUC slowly converges to 0.72. We picked and as the parameters for the final model. Aside from k-means, we also tested DBSCAN, k-modes and spectral clustering (implemented by scikit-learn). These three methods did not outperform the proposed method (results not shown).



**Figure S3. Clustering result for k = 3.** Principal Component Analysis (PCA) was used to generate the 2-D and 3-D visualization of the clustering result. In this figure, each dot represents a HT pair. The data used as the input for PCA represents the features generated by Model 2. The color of the dots indicates cluster membership.



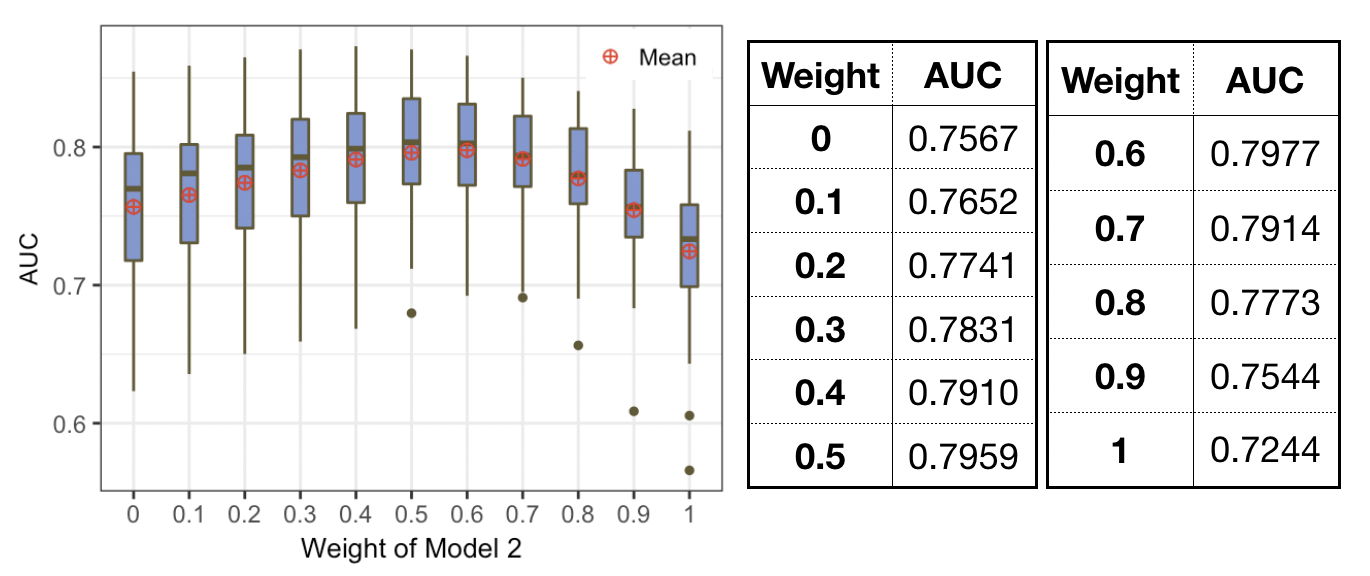
**Figure S4. Cross-validation results for the grid search of hyperparameters in Model 2.** The value and color in each grid cell represent the average AUC for a specific hyperparameter combination. The vertical axis is the number of clusters. The horizontal axis is the number of cognate residues picked for each cluster.

## 1.5 The Stacked Model

We then stacked the prediction results of Model 1 and Model 2, in order to reduce biases and boost the prediction accuracy. As stated above, Model 1 and Model 2 are two independent DD affinity prediction models. Each testing sample (i.e. each unknown HT pair) will receive two predicted affinity scores from the two models. To combine the two scores, we used weighted averaging, which is a widely used ensemble method for machine learning models. The weights are tuned through CV (**Figure S5**). The highest AUC (0.7977) was achieved when the weight of Model 2 and the weight of Model 1 . Aside from weighted averaging, Support Vector Machines (SVM) and Random Forest (RF) were employed as alternative ensemble methods (results not shown). We also tried combining features generated by Model 1 and Model 2 into one feature matrix and training RF with the entire matrix (results not shown). Weighted averaging performed best among all candidate methods and thus was used in the final model.

## 1.5 Pathway Prediction

We then used the predicted DD affinity to infer PKS protein assembly orders in the biosynthetic pathway. For a given PKS gene cluster, we first create an exhaustive list of putative pathways generating permutations of PKS proteins. Then for each candidate pathway we calculate average affinity of all interacting HT pairs in the pathway: , where function calculates the affinity between and ; is the total number of proteins in the pathway; is the C-terminal DD of the -th protein in the putative pathway; is the N-terminal DD of the -th protein in the putative pathway. The likelihood score associated with each pathway always ranges from 0 to 1, where 1 indicates high likelihood. Eventually, we rank the putative pathways according to the likelihood. Since a brute-force approach is used for pathway prediction, the running time grows exponentially with the number of genes/proteins in the biosynthetic pathway. Because of limited time and resources, in cross-validation we only predicted pathways for PKSs with ten or less genes. BGC\_18, BGC\_27 and BGC\_29 were omitted because they have more than ten genes. Pathway prediction performance in the CV of the final model is shown in a pie chart in **Figure S6**. The Mean Reciprocal Rank (MRR) of the true pathway is 0.63 (95% CI: 0.59-0.67). Pathway prediction performance on the hold-out testing dataset is shown in **Table S1**.

****

**Figure S5. Cross-validation results for the stacked model.** This figure summarizes CV results when the weight of Model 2 () takes different values. The weight of Model 1 is . In this plot, is equivalent to using only Model 2; is equivalent to using only Model 1. Among all weights, achieved the highest average AUC and was used in the final model.



**Figure S6. Cross-validation results of the final model for pathway prediction.** This figure shows the frequency for the true pathway ranking #1, #2, etc., in our ranking of putative pathways. In the outer pie chart, the “Top 10” category includes #4 through #10. “Top 20” includes #11 through #20. “Others” refers to the cases where the ranks of true pathways are beyond #20. In the inner pie chart, “Top 3” includes #1 through #3; “Others” refers to ranks beyond #3.

## 1.6 Comparing with the State-of-the-Art

The rule-based method that Yadav *et al.* developed is the current state-of-the-art for T1PKS DD affinity prediction and pathway prediction (Yadav *et al.*, 2009). However it has several defects: (i) this algorithm assumes that the structure of DEBS DD can be applied to other DDs, however, it has been proven that the DD structure of DEBS does not apply to class II or class III DDs (Whicher *et al.*, 2013; Moss *et al.*, 2019); (ii) this algorithm typically yields a large number of tied ranks in pathway prediction, which greatly affects its prediction accuracy; (iii) the score of the pathway is calculated as a sum of the scores of all interacting DDs in the pathway, therefore, the upper/lower limits of the pathway scores are different for PKSs with different numbers of genes, which makes the pathway scores difficult to interpret. DDAP has overcome all the above limitations.

We used antiSMASH 4.2.0 web server (Blin *et al.*, 2017) for the pathway prediction performance comparison. By default, for each PKS system with more than three and less than 11 PKS genes, antiSMASH returns a maximum of 1000 putative pathways and the raw scores associated with each putative pathway. For PKSs with or genes, collinearity will be assumed, meaning that the order of PKS proteins in the pathway is assumed to be the same as the order of their coding genes in the genome. The antiSMASH tool also assumes colinearity when non-PKS genes (e.g. non-ribosomal peptide synthetase) are present in the gene cluster. Therefore, we removed all non-PKS genes before uploading the sequences to antiSMASH server to ensure that the pathway prediction was performed.

We first attempted to use the hold-out testing data to compare the performance (see **Table S1**) and calculated the Mean Reciprocal Rank (MRR) for antiSMASH and our algorithm. The antiSMASH method typically introduces tied scores to the results. Therefore, we used a tie-aware MRR to measure the performance of antiSMASH (McSherry and Najork, 2008). A conventional MRR was used for the evaluation of our method, which is equivalent to using the tie-aware MRR with the number of tied scores set to 1. For the hold-out testing data, MRR of the antiSMASH method was 0.64; MRR of our method was 0.67.

**Table S1. Pathway prediction results of DDAP and antiSMASH on hold-out testing dataset.** The relationship between the total number of possible pathways () and the number of genes in the pathway () is . In the third column, means that possible pathways are returned by antiSMASH, but the true pathway is not among these putative pathways. When , it could be because: (i) the predicted PKS genes are incorrect, (ii) antiSMASH uses a separate method to locate the first and last module before generating permutations of the PKS proteins. If the first/last module is incorrectly predicted, the result will consist of a non-exhaustive list of putative pathways that do not include the true pathway. The number of tied pathways (pathways that have the exact same score as the true pathway) is shown in the fourth column.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **BGC ID** | **rank of true pathways**  **(DDAP)** | **rank of true pathways**  **(antiSMASH)** | **tied pathways**  **(antiSMASH)** | **predicted or assumed colinear**  **(antiSMASH)** | **total number of possible pathways** | **number of genes in the pathway** |
| **BGC\_86** | 1 | 1 | 1 | assumed colinear | 6 | 3 |
| **BGC\_87** | 1 | 1 | 2 | predicted | 24 | 4 |
| **BGC\_88** | 26 | 1000+ | / | predicted | 40320 | 8 |
| **BGC\_89** | 1 | 1 | 2 | predicted | 120 | 5 |
| **BGC\_90** | 1 | 1 | 1 | assumed colinear | 6 | 3 |
| **BGC\_91** | 6 | 120+ | / | predicted | 720 | 6 |
| **BGC\_92** | 2 | 1 | 1 | assumed colinear | 6 | 3 |

Since the hold-out testing set is limited in size for a robust performance comparison, we further used training pathways to compare antiSMASH and DDAP. The antiSMASH 4.2.0 web server was used to make predictions for the training pathways (i.e., the pathways used for CV of our method). We excluded 15 samples (see reasons in **Table S2**). Eventually, 70 samples were used for the performance comparison. To calculate MRR for DDAP on these 70 samples, we extracted all predictions made for these samples in the cross-validation of our best performing model. Eventually, our method achieved MRR=0.62 (95% CI: 0.57-0.66), while the antiSMASH method achieved MRR=0.48 (95% CI: 0.38-0.57). Since PKSs with assumed colinearity do not directly reflect the performance of the prediction algorithm, we calculated MRR a second time after removing the PKSs that were assumed to be colinear (**Table S2**). Eventually, antiSMASH received MRR=0.31 (95% CI: 0.23-0.40), and our method received MRR=0.54 (95% CI: 0.49-0.59) (**Figure S7**). Note that during CV, the DDAP algorithm was trained using 80% of the training data (about 74% of all available data). This result shows that even with 74% of data, DDAP still outperforms antiSMASH in most cases.



**Figure S7. Pathway prediction results of DDAP and antiSMASH on CV dataset.** The red bars represent the cases where DDAP outperforms antiSMASH. The average rank for antiSMASH is calculated as: , where is the number of tied pathways (); is the highest possible rank of the true pathway. The average rank for DDAP is simply the average of the predicted ranks in the CV results. All PKSs with assumed colinearity are removed from this plot. BGC\_13 is not included in this plot because the average ranks predicted by antiSMASH and DDAP both exceed 1000.

## 1.7 Alternative Methods

A commonly used approach for protein-protein interaction (PPI) prediction is coevolution analysis (de Juan *et al.*, 2013). RaptorX (Källberg *et al.*, 2012) is one of the most acknowledged tools in this field. The “Complex Contact Prediction” function in RaptorX uses coevolution analysis and deep learning to predict the interfacial contacts between two potentially interacting proteins. We used the RaptorX webserver to obtain contact maps of both positive and negative samples (i.e., interacting and non-interacting DDs) in the training set, and used deep neural networks implemented by Google AutoML Vision to predict DD affinity based on the contact maps. We found this method achieved an accuracy of around 50%. Therefore, this approach was not included in the final model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **BGC ID** | **rank of true pathways** | **tied pathways** | **include or exclude for performance comparison** | **notes** |
| **BGC\_14** | 1 | 1 | include | assumed colinear |
| **BGC\_17** | 1 | 1 | include | assumed colinear |
| **BGC\_23** | 1 | 1 | include | assumed colinear |
| **BGC\_24** | 1 | 1 | include | assumed colinear |
| **BGC\_28** | 1 | 1 | include | assumed colinear |
| **BGC\_38** | 1 | 1 | include | assumed colinear |
| **BGC\_4** | 1 | 1 | include | assumed colinear |
| **BGC\_41** | 1 | 1 | include | assumed colinear |
| **BGC\_44** | 1 | 1 | include | assumed colinear |
| **BGC\_50** | 1 | 1 | include | assumed colinear |
| **BGC\_59** | 1 | 1 | include | assumed colinear |
| **BGC\_6** | 1 | 1 | include | assumed colinear |
| **BGC\_62** | 1 | 1 | include | assumed colinear |
| **BGC\_67** | 1 | 1 | include | assumed colinear |
| **BGC\_68** | 1 | 1 | include | assumed colinear |
| **BGC\_7** | 1 | 1 | include | assumed colinear |
| **BGC\_74** | 1 | 1 | include | assumed colinear |
| **BGC\_58** | 7 | 12 | include | predicted |
| **BGC\_10** | 3 | 22 | include | predicted |
| **BGC\_11** | 1 | 1 | include | predicted |
| **BGC\_12** | 3 | 10 | include | predicted |
| **BGC\_13** | 1000+ | / | include | predicted |
| **BGC\_15** | 5 | 10 | include | predicted |
| **BGC\_16** | 3 | 2 | include | predicted |
| **BGC\_19** | 1 | 12 | include | predicted |
| **BGC\_20** | 871 | 134 | include | predicted |
| **BGC\_25** | 2 | 1 | include | predicted |
| **BGC\_26** | 5 | 6 | include | predicted |
| **BGC\_30** | 1 | 4 | include | predicted |
| **BGC\_31** | 49 | 624 | include | predicted |
| **BGC\_33** | 1 | 2 | include | predicted |
| **BGC\_35** | 1 | 4 | include | predicted |
| **BGC\_36** | 1 | 1 | include | predicted |
| **BGC\_37** | 1000+ | / | include | predicted |
| **BGC\_39** | 275 | 526 | include | predicted |
| **BGC\_40** | 1 | 6 | include | predicted |
| **BGC\_42** | 17 | 8 | include | predicted |
| **BGC\_45** | 63 | 148 | include | predicted |
| **BGC\_46** | 10 | 5 | include | predicted |
| **BGC\_47** | 2 | 1 | include | predicted |
| **BGC\_48** | 1 | 2 | include | predicted |
| **BGC\_49** | 2+ | / | include | predicted |
| **BGC\_5** | 1 | 6 | include | predicted |
| **BGC\_52** | 10 | 19 | include | predicted |
| **BGC\_53** | 6+ | / | include | predicted |
| **BGC\_54** | 25 | 102 | include | predicted |
| **BGC\_55** | 1 | 2 | include | predicted |
| **BGC\_56** | 2 | 3 | include | predicted |
| **BGC\_57** | 1 | 4 | include | predicted |
| **BGC\_60** | 1 | 2 | include | predicted |
| **BGC\_61** | 24+ | / | include | predicted |
| **BGC\_63** | 1 | 4 | include | predicted |
| **BGC\_64** | 1 | 4 | include | predicted |
| **BGC\_65** | 169 | 168 | include | predicted |
| **BGC\_66** | 4 | 9 | include | predicted |
| **BGC\_69** | 9 | 6 | include | predicted |
| **BGC\_70** | 1 | 2 | include | predicted |
| **BGC\_71** | 2 | 1 | include | predicted |
| **BGC\_73** | 1 | 4 | include | predicted |
| **BGC\_75** | 1 | 4 | include | predicted |
| **BGC\_76** | 1 | 6 | include | predicted |
| **BGC\_77** | 2 | 3 | include | predicted |
| **BGC\_78** | 13 | 4 | include | predicted |
| **BGC\_79** | 5 | 2 | include | predicted |
| **BGC\_80** | 5 | 16 | include | predicted |
| **BGC\_81** | 1 | 3 | include | predicted |
| **BGC\_82** | 7 | 11 | include | predicted |
| **BGC\_83** | 1 | 1 | include | predicted |
| **BGC\_84** | 51 | 31 | include | predicted |
| **BGC\_85** | 115 | 6 | include | predicted |
| **BGC\_2** | / | / | exclude | hybrid |
| **BGC\_21** | / | / | exclude | hybrid |
| **BGC\_22** | / | / | exclude | hybrid |
| **BGC\_3** | / | / | exclude | hybrid |
| **BGC\_32** | / | / | exclude | hybrid |
| **BGC\_43** | / | / | exclude | hybrid |
| **BGC\_8** | / | / | exclude | hybrid |
| **BGC\_34** | / | / | exclude | hybrid |
| **BGC\_18** | / | / | exclude | no results from CV |
| **BGC\_27** | / | / | exclude | no results from CV |
| **BGC\_29** | / | / | exclude | no results from CV |
| **BGC\_1** | / | / | exclude | wrong region |
| **BGC\_51** | / | / | exclude | wrong region |
| **BGC\_72** | / | / | exclude | wrong region |
| **BGC\_9** | / | / | exclude | wrong region |

**Table S2. The antiSMASH pathway prediction results.** In the second column, means that possible pathways are returned by antiSMASH, and the true pathway is not among these putative pathways. The third column is the number of pathways tied with the true pathway (including the true pathway itself). The fourth column indicates whether the PKS is considered for performance comparison. In the fifth column, “predicted” means that the pathway is predicted by antiSMASH; “assumed colinear” means the number of PKS genes is not within the range of 3 to 11, and the pathway is generated assuming colinearity; “hybrid” means antiSMASH assumes colinearity because hybrid genes are present in the gene cluster; “wrong region” means that the PKS regions predicted by antiSMASH are inaccurate; “no results from CV” means that these PKSs did not receive pathway predictions in our CV.

# **2 References**

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

Benson,D.A. *et al.* (2017) GenBank. *Nucleic Acids Research*, **45**, D37–D42.

Blin,K. *et al.* (2017) antiSMASH 4.0—improvements in chemistry prediction and gene cluster boundary identification. *Nucleic Acids Research*, **45**, W36–W41.

Blin,K. *et al.* (2019) antiSMASH 5.0: updates to the secondary metabolite genome mining pipeline. *Nucleic Acids Research*.

Broadhurst,R.W. *et al.* (2003) The structure of docking domains in modular polyketide synthases. *Chemistry & Biology*, **10**, 723–731.

Buchholz,T.J. *et al.* (2009) Structural basis for binding specificity between subclasses of modular polyketide synthase docking domains. *ACS Chemical Biology*, **4**, 41–52.

Camacho,C. *et al.* (2009) BLAST+: architecture and applications. *BMC Bioinformatics*, **10**, 421.

de Juan,D. *et al.* (2013) Emerging methods in protein co-evolution. *Nature Reviews Genetics*, **14**, 249–261.

Källberg,M. *et al.* (2012) Template-based protein structure modeling using the RaptorX web server. *Nature Protocols*, **7**, 1511–1522.

McSherry,F. and Najork,M. (2008) Computing information retrieval performance measures efficiently in the presence of tied scores. *European Conference on Information Retrieval*. Springer, pp. 414–421.

Medema,M.H. *et al.* (2015) Minimum information about a biosynthetic gene cluster. *Nature Chemical Biology*, **11**, 625–631.

Moss,N.A. *et al.* (2019) Nature’s combinatorial biosynthesis produces vatiamides A-F. *Angewandte Chemie International Edition*.

Pedregosa,F. *et al.* (2011) Scikit-learn: machine learning in python. *Journal of Machine Learning Research*, **12**, 2825–2830.

Thattai,M. *et al.* (2007) The origins of specificity in polyketide synthase protein interactions. *PLOS Computational Biology*, **3**, e186.

Whicher,J.R. *et al.* (2013) Cyanobacterial polyketide synthase docking domains: a tool for engineering natural product biosynthesis. *Chemistry & Biology*, **20**, 1340–1351.

Yadav,G. *et al.* (2009) Towards prediction of metabolic products of polyketide synthases: an in silico analysis. *PLOS Computational Biology*, **5**, e1000351.