MASTERARBEIT / MASTER’S THESIS

Titel der Masterarbeit / Title of the Master‘s Thesis

„PICA-to-go: A fast microbial phenotype investigation pipeline“

verfasst von / submitted by

Florian Piewald BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2018 / Vienna 2018

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 875
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Bioinformatik
Betreut von / Supervisor:	Prof. Dr. Thomas Rattei

Inhaltsverzeichnis

Zusammenfassung

Nachdem die Anzahl an komplett sequenzierten bakteriellen Spezies steigt, wird die Analyse der Phänotypen dieser Spezies zu einem Bottleneck innerhalb der Wissenschaft. In der Vergangenheit wurden Machine Learning-Tools verwendet, um diesem Problem Herr zu werden. Das PICA Framework, das mit Support Vector Maschinen arbeitet, ist ein Beispiel für solch ein Tool. Allerdings benötigt PICA Informationen bezüglich der 'cluster of orthologous groups' (COGs) in jeder Spezies (auch Bin genannt), welche für das Training bzw. für die Vorhersage verwendet wird. In unserer Arbeitsgruppe wurde für diesen Zweck in vergangenen Projekten HMMER verwendet, um die eggNOG Datenbank zu durchsuchen. Diese Herangehensweise ist allerdings limitiert für Wissenschaftler, welche Zugriff zu einem Hochleistungs-Computer-Cluster haben und benötigt zudem einen beträchtlichen Zeitaufwand. Ich stelle eine neue Herangehensweise vor (PICA-to-go), welche das Clustering-Toolset von MMSeqs2 in Verbindung mit dem PICA Framework nutzt. Ein Modell für Ciprofloxacin-Resistenz in Acinetobacter baumannii kann so mit einer Genauigkeit von 0.93 in weniger als zehn Minuten auf einem gewöhnlichen PC trainiert werden.

Abstract

As the number of completely sequenced bacterial species grows, analyzing the phenotypes of these species becomes a bottleneck in science. Machine learning tools have been used in the past to cope with this problem. The PICA framework is an example of such a tool, using support vector machines. PICA, however, needs information about the clusters of orthologous groups (COGs) in each species (also known as bin) to be trained/predicted. In previous work of our group, HMMER searching in the eggNOG database, was used for this purpose. The usage of this approach is limited to researchers with access to a high performance computing cluster and takes a considerably amount of time. I present a new approach (PICA-to-go) using the clustering suite of MMSeqs2 together with the PICA framework. A model for ciprofloxacin resistance in Acinetobacter baumannii with a balanced accuracy of 0.93 can be trained in less than 10 minutes on an ordinary desktop machine.

1 Introduction

1.1 Related Projects and PICA-to-go

Effective DB EffectiveDB is an application and database, with a user friendly web interface, for the prediction of protein secretion systems in bacteria. It is based on the PICA framework using support vector machines (SVMs). As features for machine learning, orthologous groups are used (from eggNOG 4.0). (Eichinger et al., 2016)

PhenDB PhenDB is a currently ongoing (and yet unpublished) project of a phenotype prediction application. In contrast to EffectiveDB, the predictions are not limited to secretion systems, but cover a wide range of different phenotypes. It is built as a user friendly web interface, where researchers can upload their bins and obtain a wide range of different phenotype predictions together with informations about completeness/contamination and taxonomic predictions. The application as well as information about the tool and current news are available at `phendb.org/`. The author of this master thesis has also been involved in building the PhenDB application.

PICA training pipeline for PhenDB The existing pipeline used to train PICA models for PhenDB uses Prodigal (Hyatt et al., 2010) for gene finding and HMMER (Wheeler and Eddy, 2013) to annotate all genes to the eggNOG 4.5 database (Huerta-Cepas et al., 2016). The eggNOG id, which refers to a cluster of orthologous groups (COG), is then used as feature to train a PICA model.

1.2 The idea behind PICA-to-go

Unlike the related projects mentioned above PICA-to-go is a desktop software that provides the user with the possibility to train his own machine learning models as well as to perform predictions using these models. The tool is usable on a normal desktop machine without the access to a computer cluster. The internal pipeline used for PhenDB is able to train as well as to predict PICA models, however it is not suitable to be used on a desktop computer as it requires to map a sequence to the eggNOG database using HMMER. This is not only a speed limiting step, but also requires a mirror of the eggNOG database on the hard drive. Other steps in the pipeline, however, such as Prodigal for gene prediction or PICA itself are quite fast and can be performed on a desktop machine. Therefore, alternatives for the HMMER/eggNOG step have to be found with the aim to replace this step using different tools. Moreover, conversion steps may be required between this steps as the output of a certain tool may differ from the input format required by the succeeding tool in the pipeline.

Evaluation To test if PICA-to-go performs as well as the existing pipeline, all PICA models used in PhenDB are trained using PICA-to-go and the mean balanced accuracy is then compared to the existing pipeline. As a runtime requirement, it should be possible to run most models on a regular laptop on a single workday (8h).

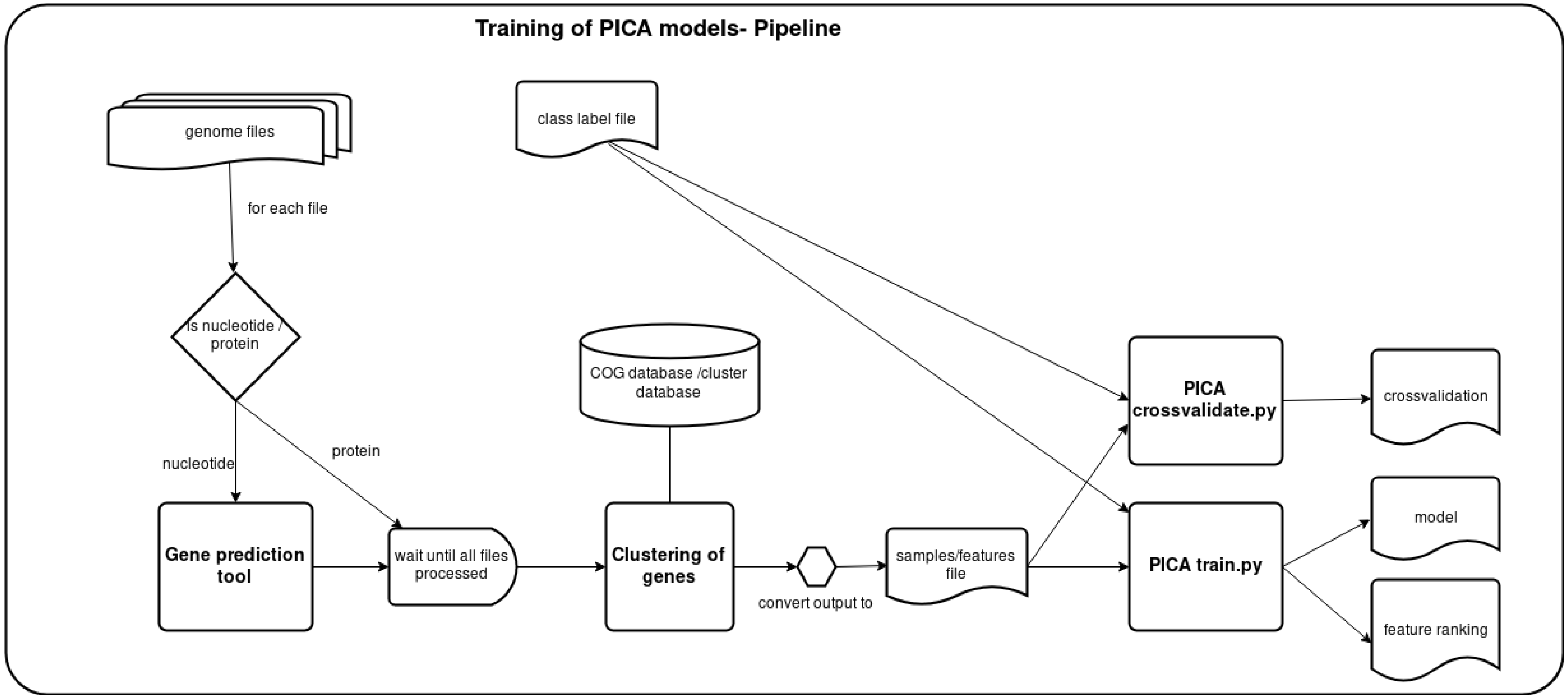


Figure 1: Workflow of training a PICA model

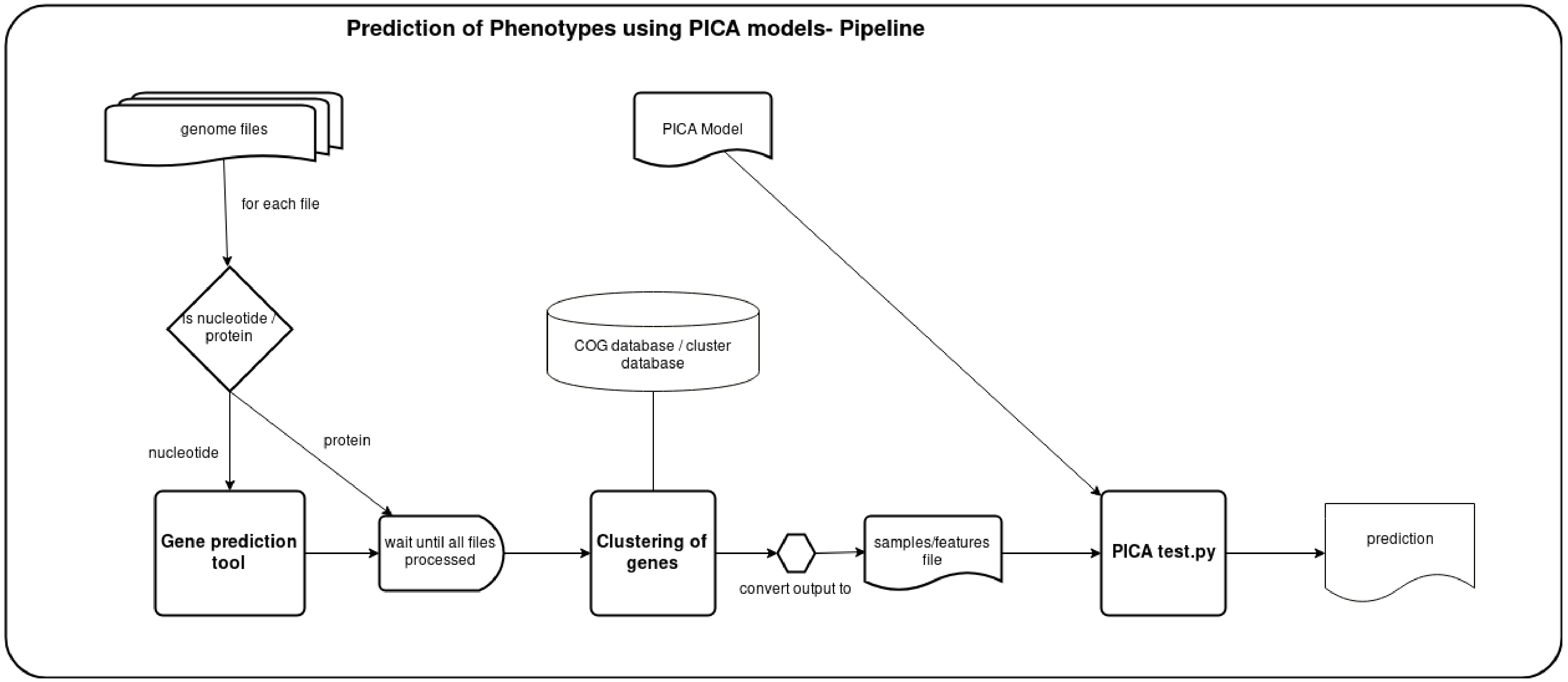


Figure 2: Workflow of performing a prediction using a PICA model

1.3 Relevance of phenotype prediction

1.3.1 The age of metagenomics

For more than a century, microbial research was mainly done by cultivating microbes as pure cultures on agar plates. Such a setup easily allows the testing of growth requirements or the sensitivity towards environmental conditions (e.g. temperature, presence of a certain antibiotic) of a certain bacterium. However, it is estimated that more than 99 percent of all prokaryotic species cannot be cultured, which has been a major problem in microbial research (Schloss and Handelsman, 2005). As the cost of sequencing is decreasing, the metagenomic approach is used to deal with this problem. DNA is directly obtained from environmental samples and sequenced. To reconstruct the original sequence, all reads are assembled using an existing backbone (mapping) or by de-novo assembly. Before that, to avoid wrongly assembled reads, each read is separated on a taxonomical level into different groups using binning algorithms. Each species gained by this process is called bin. (Kunin et al., 2008) While certain tools exist for predicting the taxonomical classification of a certain bin (e.g. Kraken (Wood and Salzberg, 2014)), it is still not trivial to assign bins to a phenotype.

1.3.2 Phenotypes

A phenotype is defined as observable physical properties of an organism. For bacterial organisms, this could be the growth requirements such as the usage of methane as energy source (Bochner, 2008). Different phenotypes can be a result of a single nucleotide polymorphism (Read and Massey, 2014), the presence or absence of a gene or even by changes in the non-coding region of the genome (e.g. a mutation in a promotor region or in a region of a non coding regulatory RNA (Gottesman, 2005)). In some cases, marker genes can be used to predict a certain phenotype based on whole genome data. An example for this approach is the ammonia monooxygenase structural gene amoA (Rotthauwe, Witzel, and Liesack, 1997). However, this approach is limited to species that have been well investigated. Furthermore, this idea follows the hypothesis that a trait is controlled by a single locus, which is true for some genetic traits, while a high amount of phenotypes is controlled by a complex circuit involving multiple genes (Plomin, Haworth, and Davis, 2009). To address this problem, genome wide association studies (GWAS) are used in human genetics. GWAS associate a single nucleotide polymorphisms to a characteristic with a certain probability (e.g. enhanced probability of a particular disease). This is a purely statistical approach that differs from machine learning techniques (see 1.4). For prokaryotic organisms, however, GWAS is not as common yet (Lees and Bentley, 2016) (Brynildsrud et al., 2016).

To predict phenotypes, some approaches use curated databases containing genes and/or SNPs, which cause a specific phenotype, collected from scientific literature. This approach is commonly used by tools predicting antibiotic resistance. An example of such a database is ResFinder (Zankari et al., 2012) or CARD (Comprehensive Antibiotic Resistance Database) (McArthur et al., 2013). Some of these tools are able to work with pre-assembled reads. SRST2 uses Bowtie2 to map a read to potential resistance genes of a reference database (Inouye et al., 2014). Other tools, such as Mykrobe predictor, rely on the comparison of De Bruijn graphs and do not only provide information about antibiotic resistance, but also on virulence (Bradley et al., 2015). In conclusion, most of these tools are intended to be used in the field of medical microbiology and rely on curated databases. This clearly distinguishes these tools from our PICA-based tools such as PhenDB or PICA-to-go, which are intended as general purpose tool for phenotype prediction. As PICA relies on machine learning, a curated database of genes relevant for a certain phenotype is not needed. Instead, it finds relevant genes automatically in the process of the training of the model. This information is then not only used to generate a model to perform predictions, but can also be used to identify relevant genes for a certain phenotype, which have not been associated with this phenotype before. Traitar is a similar tool, which also relies on machine learning (support vector machines). It supports 67 models that have been trained using training data from the proprietary

GIDEON database. The intention behind Traitar is not to allow the user to train its own models, but just to provide him with trained models. As features for machine learning, functional annotations from the Pfam database are used. (Weimann et al., 2016) On the other hand, PICA-to-go is not shipped with trained models, but is intended to provide the user with the option of building his own models.

1.4 Machine learning

1.4.1 Definition and Comparison to other fields

Machine learning is a term in computer science used for algorithms that learn and improve from experience (Jordan and Mitchell, 2015). It is closely related to the field of statistics (Breiman, 2001). The latter, however, mainly focuses on describing data (descriptive statistics) or drawing a conclusion (inferential statistics). An example for descriptive statistics would be information about the average income in a country (median, arithmetic average, variance etc.). On the other hand, an example of inferential statistics would be a classical hypothesis test ("Is cancer drug X more effective than cancer drug Y?"). (Pagano, 2012) Machine learning, however, tries to make a prediction using a set of data ("Is this bacteria AEROBE?"). There are two different approaches in machine learning: Supervised machine learning and unsupervised machine learning. Supervised machine learning uses data from the past, that is assigned to certain labels (e.g. AEROBE YES/NO). General rules are then extracted from the dataset and a model is build upon these rules (training). After the training, the extracted rules (= the model) can be used to predict the label of a new sample. By contrast, unsupervised machine learning tries to find structures in unlabelled data. An example for unsupervised machine learning is the clustering of data. (Bastanlar and Ozuysal, 2014)

1.4.2 Supervised Machine learning: Training of a model

The training data used in supervised machine learning typically consists of a list of samples that are assigned to a certain class (YES/NO or multiple classes). Each sample consists of a vector of features. A feature is an individual measurable property (Bastanlar and Ozuysal, 2014). For phenotype prediction a sample would be a bacterial strain and a feature a certain property associated with this strain. This could be the presence/absence of genes or SNPs, the GC count or the amino acid composition. The accuracy of a model highly depends on a useful selection of features (Bastanlar and Ozuysal, 2014). As the GC count or the amino acid composition may not provide much information about the phenotype compared to SNPs or the presence/absence of genes, it may not be very useful to use these as features.

1.4.3 Bias

One major problem in machine learning is overfitting, which means that the model fits too close to a limited number of data points. This is especially a problem for datasets with a high amount of noise (high variance). Overfitted models may lead to an accurate prediction, when a sample from the trainingdata is used for prediction, but may have a low accuracy when a prediction is performed on an unknown sample. This means that the model does not generalize well. (Bastanlar and Ozuysal, 2014) The chance of overfitting also depends on the used machine learning approach, certain approaches (e.g. tree learners) tend to overfit and additional techniques have to be applied to reduce the amount of overfitting(e.g. pruning or boosting) (Mueller and Guido, 2016). Furthermore an increase of the number of samples as well as a reduction of features may be used as strategies against overfitting. To get an unbiased estimate of the models accuracy, a crossvalidation can be performed. The trainingdata is randomly splitted into k equal sized folds. While last fold is used to perform predictions after the model is build (predict-set), the first ones (0...k-1) are used for training (train-set). The splitting in train and predict set should avoid being assigned too optimistic values (due to overfitting). The number of false positives, false negatives, true positives and true negatives (confusion matrix) is counted. The training of the model using 0...k-1 sets is done k times using all combinations of train- and predict-sets. As the initial split into k-folds may be

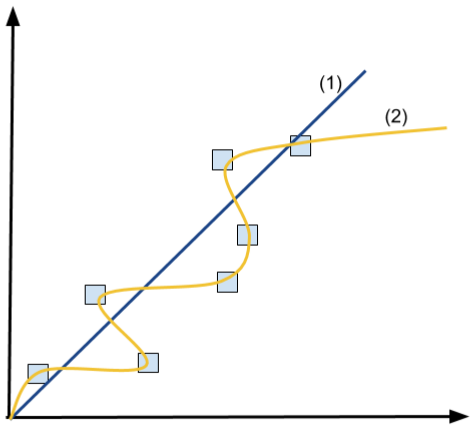


Figure 3: An example of overfitting. (2) matches all data points exactly, but may be a inaccurate model, as potential noise is incorporated in the model (overfitting); (1) may be a closer model to reality, even it does not match all data points exactly

biased, the whole procedure is repeated r times using a random split. Eventually (the desired) statistical values such as sensitivity, specificity, the raw accuracy or the balanced accuracy are calculated for each model (number of models = r*k). The balanced accuracy has the advantage that true positives and true negatives equally contribute regardless how the data is structured (e.g. if a label is predominant, the raw accuracy will be biased). The mean and the variance is then calculated for each statistical measure. (Marsland, 2014)

Besides overfitting, another bias can be introduced in the selection of the samples used to train the model. Supervised machine learning algorithms try to find rules to separate datasets based on differences in the features to achieve an optimal separation between the used labels. For example, if in a model on amoxicillin resistance, the samples labelled as YES are all of the species Staphylococcus aureus, while the samples labelled as NO are of various bacterial species (but mainly not of Staphylococcus aureus), the model will be trained to detect whether a bacterium belongs to Staphylococcus aureus, but won't be a useable model to detect amoxicillin resistance.

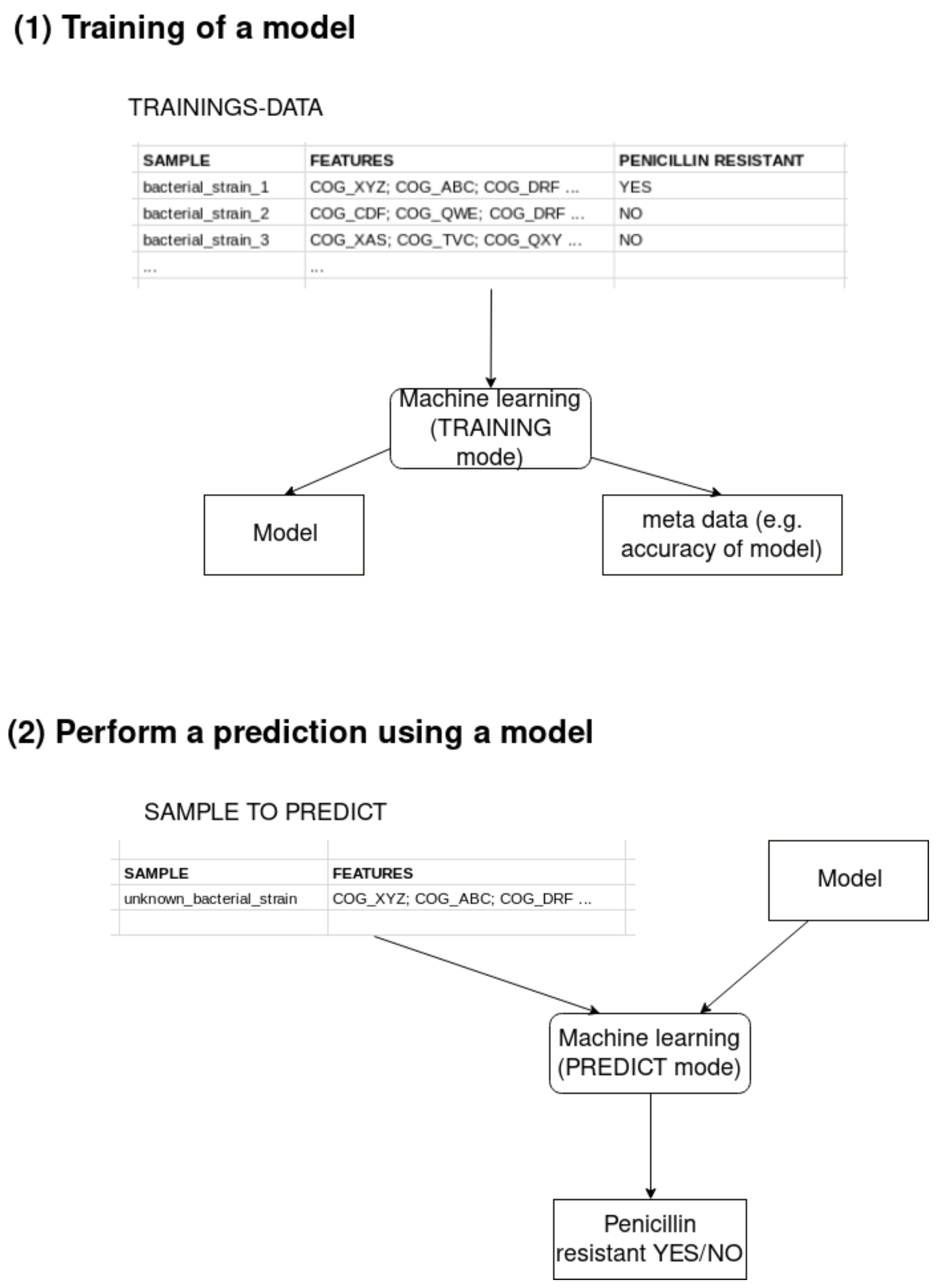
1.4.4 Machine learning tools for phenotype prediction

For the process of machine learning itself, different techniques are available, each of which with several advantages or disadvantages. PICA was originally built using the CPAR (Predictive Association Rules) algorithm (MacDonald and Beiko, 2010). Furthermore, PICA also offered a plugin that relied on support vector machines (SVM), which has been extended by Feldbauer and colleagues. This approach led to a similar balanced accuracy and has the advantage of a lower runtime. (Feldbauer et al., 2015) Besides PICA, another similar tool for phenotype prediction is Traitar, which also relies on support vector machines. (Weimann et al., 2016)

The PICA framework basically consists of three different tools (train.py, test.py and crossvalidate.py). The PICA train mode requires a class label file and a samples/features file, which consists of a tab separated list of all features for each bin. As output, the program delivers a machine readable model file that is required to predict phenotypes. Furthermore, the tool is able to provide a feature ranking that states the relevance of each feature (in this case each feature is a gene cluster) for the model. To gain relevant information about the accuracy of the model, there is the PICA crossvalidate mode that provides the user with informations about accuracy. The required input files are the same as for the PICA train program. PICA test is able to predict phenotypes using a model file (output of PICA-train) and a samples/features file. For the samples/features file, the idea is to use functional equivalent clusters of genes as features. Typically, clusters of orthologous groups were used for this before.

1.4.5 Support vector machines

Support vector machines (SVMs) are a supervised machine learning technique that is mainly used for two-class classification problems (e.g. Is my bacterium AEROBE YES/NO) and are quite robust (do not tend to overfit easily). SVMs try to find an optimal split (hyperplane) in a multidimensional feature space. In the case of phenotype prediction the feature space is a binary vector (e.g. presence/absence of COGs). Finding an optimal split means that the distance from the hyperplane to the nearest data point representing the first class (e.g. AEROBE YES) equals the distance of nearest data point representing the second class (e.g. AEROBE NO) to the hyperplane. In this regard, the SVM differs from the perceptron machine learning technique, which finds a split between data point, but without the requirement of an optimal split. Another problem regarding SVMs is that the data points are, not always linearly separable. This is solved by an optimization problem that tries to minimize the number of wrongly split data points as much as possible. (Marsland, 2014) (Mueller and Guido, 2016) The optimal hyperplane can be linear or multidimensional. For the latter, the kernel trick is used to avoid computationally expensive calculations. With multidimensional kernels more complex relationships between the features can be modelled; however, the results may not be comprehensible by humans any more and the SVM basically becomes a black box. On the other hand, the usage of linear kernels allows to extract the individual weight for each feature (Chang and Lin, 2008). This makes it possible to detect which COGs are the most relevant for the prediction (feature ranking). Furthermore, not only the overall accuracy of a model can be evaluated (using crossvalidation), but also the reliability of a single prediction can be evaluated using Platt scaling (Platt, 2000), which considers the distance between the individual data point (e.g. a certain bin in a multidimensional feature space) and the hyperplane. (Marsland, 2014) (Mueller and Guido, 2016)


(1) Training of a model
TRAININGS-DATA
Machine learning (TRAINING mode)
Model
meta data (e.g. accuracy of model)
(2) Perform a prediction using a model
SAMPLE TO PREDICT
Model
Machine learning (PREDICT mode)
Penicillin resistant YES/NO

SAMPLE	FEATURES	PENICILLIN RESISTANT
bacterial_strain_1	COG_XYZ; COG_ABC; COG_DRF ...	YES
bacterial_strain_2	COG_CDF; COG_QWE; COG_DRF ...	NO
bacterial_strain_3	COG_XAS; COG_TVC; COG_QXY ...	NO
...	...	

SAMPLE	FEATURES
unknown_bacterial_strain	COG_XYZ; COG_ABC; COG_DRF ...

Figure 4: The process of machine learning

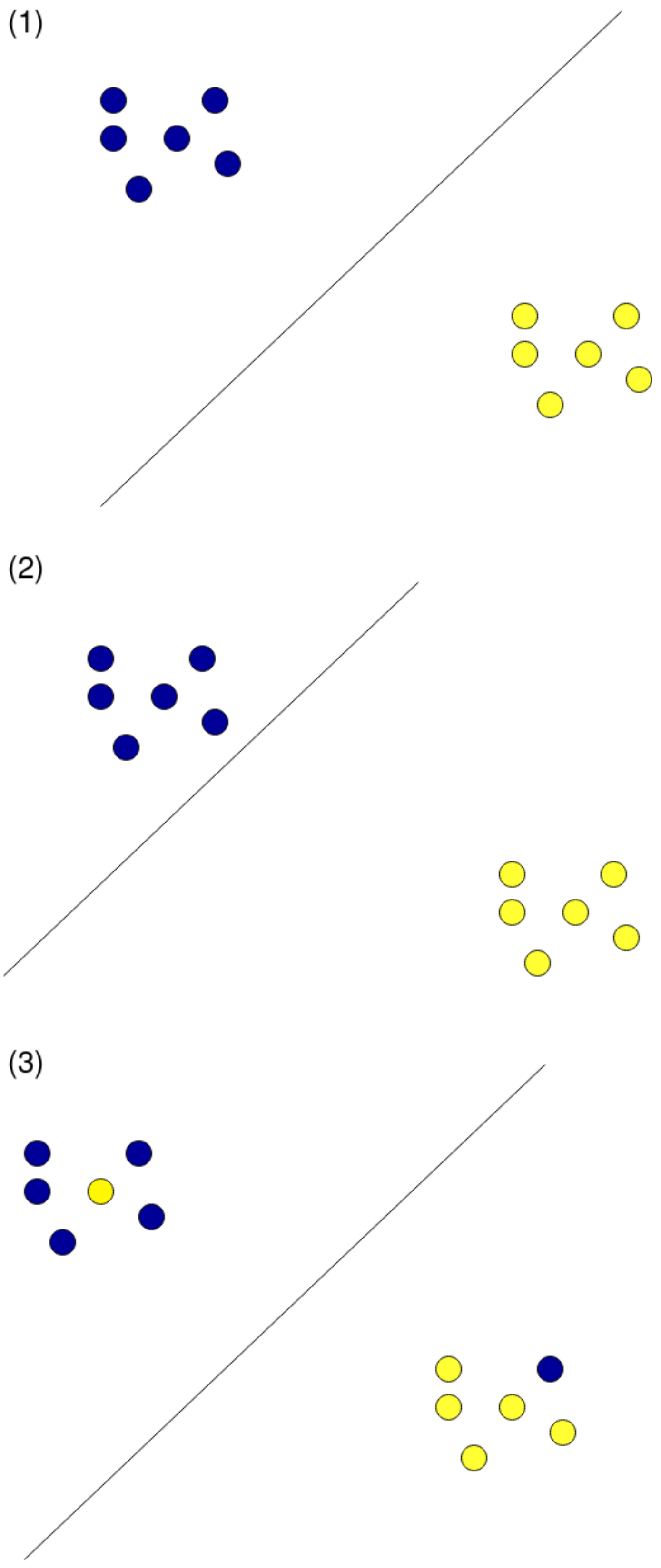


Figure 5: (1) Optimal split, (2) Splits all data points correctly, but split is not optimal, (3) Optimal split, but data points not separable

1.5 Functional equivalent clusters

To train a machine learning model using PICA, a list of functional equivalent clusters for each bin has to be provided. In this section, different possibilities for clustering genes to gain functional equivalent clusters are reviewed.

Clusters of orthologous groups The term homologous is used for genes of the same evolutionary origin. Homology is not a gradual feature, but a binary decision, e.g. two genes cannot be 50 percent homologous. There are two subtypes of homology: Orthologs and paralogs. Besides these two subtypes, there are xenologs, which result from horizontal gene transfer. While orthologous groups originate from speciation events, paralogs arise from duplication. (Gerlt and Babbitt, 2000) Clusters of orthologous groups (COGs) are often suggested to be used as features in machine learning for phenotype prediction, such as in the PICA framework (MacDonald and Beiko, 2010). The underlying idea behind this, is the orthology-function conjecture. It is suggested that orthologs tend to fulfil a similar function in different species, while paralogs tend to fulfil different functions. This view is widespread and some authors even define orthologs as "homologs in different species that catalyze the same reaction" (Gerlt and Babbitt, 2000). However, the idea of conserved functions across orthologs has been questioned and there are examples of orthologs, that falsify this claim (Studer and Robinson-Rechavi, 2009). Even the corresponding claim that paralogs functional diverge is not always true, as there are cases, where paralogy contribute to protein dosage modulation. Nevertheless, the orthology-function conjecture seems to hold true as a statistical trend and COG databases such as eggNOG were built on the idea to determine function from orthology. (Gabaldón and Koonin, 2013)

To use orthologous groups as features in machine learning, it is necessary to cluster all genes on a orthologous level. This task is not trivial, as orthology is defined evolutionarily and concepts such as sequence similarity/identity are not always useful for determining whether two gene belongs to the same orthogroup as orthologs may evolve at different rates in different lineages. Furthermore, the classification in paralog and ortholog genes is not always straightforward as paralogy does not only apply to genes within a species, but also occurs between species in complex scenarios (e.g. differential gene loss). (Gabaldón and Koonin, 2013)

Identification of orthologous groups Hypothetically, orthologs and paralogs could be identified by using a perfect phylogenetic tree. However, the construction of phylogenetic trees is not trivial and furthermore computationally expensive. Therefore, most approaches for the identification of orthologous groups rely on comparing all genes with all genes using tools like Blast followed by the identification of best bidirectional hits between each species. (Zallot et al., 2016) OrthoMCL, which is a popular tool to identify orthologous groups, assigns the similarity of each best bidirectional hit to the edges of a graph and performs a clustering using graph based clustering (MCL algorithm). (L. Li, Stoeckert, and Roos, 2003)

The COG database by Eugene Koonin uses a similar approach, but clusters the genes by detecting and merging triangles of mutual best bidirectional hits in a manually curated procedure to a cluster (Tatusov et al., 2000). Another popular database is the eggNOG database, which is an approach similar to the COG database, but does not need any manual curation. Furthermore it introduces the idea of taxonomic levels regarding orthologous groups. For example two genes may be considered as orthologs on the family level, but not on the level of the last universal common ancestor (Jensen et al., 2008).

OrthoFinder, on the other hand, extends the approach used by OrthoMCL, as it uses length normalized hits (as the score of an alignment is dependent on the length, which might introduce a bias) (Emms and Kelly, 2015).

The approaches mentioned above can be distinguished in two classes: Standalone tools that cluster sequences based on orthology (e.g. OrthoFinder) and approaches that rely on the construction of a central database and a potential query is mapped to the database using tools like HMMER (e.g. eggNOG database). This distinction, is, however, not always possible as OrthoMCL is available as stand-alone tool, but there is also a central database that is constructed using OrthoMCL (OrthoMCL DB (Chen et al., 2006)).

Clustering genes without any special regard to orthologous groups Besides methods for finding orthologous groups, there are tools that cluster genes based on similarity and/or the e-value as a cut off. These tools do not claim to identify orthologous groups as gene similarity alone may only be a weak classifier for orthologous groups compared to best bidirectional hits. This leads to the case that (more similar) paralogs may be present in the same cluster, while distant orthologs are in different clusters. However, some authors suggest that gene function correlates well with gene similarity (Joshi and Xu, 2007). For clustering, several tools have been developed such as CD-HIT (W. Li and Godzik, 2006), USEARCH (Edgar, 2010) or MMSeqs (Hauser, Steinegger, and Söding, 2016).

MMSeqs / MMSeqs2 The MMSeqs clustering suite seems especially promising as the alignment mode is 1000 times faster than BLAST and provides a similar sensitivity. Per default, the tool performs a three-step cascaded clustering. It implements an all vs. all sequence comparison, which performs an alignment on all sequence pairs that pass a prefilter module. In the first step, it uses a low sensitivity setting. After the all vs. all sequence comparison, the sequence with the largest number of neighbours (sequences that are above a certain threshold e.g. sequence identity) is merged with its neighbours to a cluster. Now the sequence with the second largest number of neighbours is merged and so on until all sequences belong to a cluster(greedy clustering). In the second and in the third step, the representative sequences of each cluster are clustered with each other using a higher sensitivity each time. Using this so called cascaded clustering workflow, a higher sensitivity can be achieved by an acceptable speed. (Hauser, Steinegger, and Söding, 2016)

MMSeqs2, an advanced version of the original MMSeqs tool (Steinegger and Söding, 2017), offers a tool for clustering in linear time (linclust) that allows to cluster 1.6 billion sequences in 10h on a single server and is thereby more than 1000 times faster than similar tools. Linclust avoids an all vs. all sequence prefilter/alignment step. For each sequence a defined number of k-mers (per default 20) is selected based on the lowest hash function value (to achieve that similar k-mers are selected per sequence). For each k-mer it finds a set of sequences that share this k-mer. The longest sequence of each set is then selected as "center sequence". All sequences in the set are then aligned to the center sequence and are connected by edges in a graph if certain requirements have been met (e-value, sequence identity and coverage). Eventually greedy incremental clustering is performed. All sequences are sorted based on their length and the longest sequence is merged with all sequences connected by an edge to a cluster and with the longest sequence as representative sequence. The MMSeqs2 package provides linclust as a standalone tool (here referred to as MMSeqs2/linclust). However, it also included linclust in their 'conventional' cluster algorithm as first step in a cascaded clustering workflow (referred to as MMSeqs2/cluster). Additionally, it should be noted that all clusters were analysed for consistency by comparing the functional annotation of all sequences in each cluster. This was done using Gene Ontology- and Pfam annotations. For Gene Ontology annotations a subset of experimentally inferred annotations was additionally used. Both MMSeqs2/cluster and MMSeqs2/linclust were found to generate more consistent clusters than UCLUST and CD-Hit. (Steinegger and Söding, 2018)

Suitable clustering approach for PICA-to-go As said before, for Pica-to-go it is essential to avoid the necessity of mirroring a large database and of mapping sequences to this database. MMseqs2/linclust and MMSeqs2/cluster have the advantage of being incredibly fast. Therefore MMSeqs2/cluster and MMSeqs2/linclust were evaluated to be used in Pica-to-go. OrthoFinder was also initially used, but was then dropped, as it did not satisfy the runtime requirements (see Result section).

Comparison of different clusterings To investigate how the clusterings gained by eggNOG/HMMER and by alternative tools used in Pica-to-go differ from each other, several clustering evaluation techniques can be used. A distinction has to be made between internal and external evaluation. While internal evaluation means that the evaluation is based on information that is retrieved from a single clustering itself, external evaluation means to compare the outcome of a clustering to a "gold standard" clustering (Rezaei et al., 2016). Besides that, another option would be to manually evaluate the cluster to investigate if these clusterings are useful, which however is only feasible when the dataset is not too large. When using different clustering algorithms that produce different results when the clusterings are used as feature for machine learning, algorithms for external evaluation could be used to compare these clusterings. Most of these algorithms return a value between 0 and 1, with a value of 1 meaning that the clusterings are identical. The used algorithms compares the clusters in every set with each other. Some of these indices are not symmetric (e.g. Meila-Heckerman index) as they would return a different result when comparing clusterset1 with clusterset2 and comparing clusterset2 with clusterset1 (S. Wagner and D. Wagner, 2007). These indices are therefore difficult to interpret and to apply to certain problems. The Meila Heckermann index (MH) compares every cluster of a clustering to the cluster in the second clustering having the maximal intersection. The intersection is summed up and divided by the number of items (n). (Meila, 2007) While this approach seems pretty straightforward, problems can arise from certain clusterings. Let's imagine a radical case of a clustering consisting of 5 items for which every item forms its own cluster. If we compare this clustering with a clustering that consists of a single cluster with all 5 items in them and compare them with each other, a value of 1 or a value of 0 will be returned depending on whether clustering1 is compared to clustering2 or the other way around. The Van Dongen measure tries to overcome the problem of asymmetry by considering both cases. Each cluster is matched to another cluster with whom it has a maximal intersection. (S. Wagner and D. Wagner, 2007) (Rezaei et al., 2016) The Van Dongen measure was used in a modified version together with the Meila-Heckerman index for the comparison of clusters (see 4.8).

2 Results

All computations were performed on an ordinary desktop machine (4 cores, 8GB RAM; see Methods section for hardware specifications) if not stated otherwise.

First of all, the selection of a suitable clustering tool to be used in PICA-to-go is discussed, as HMMER/eggNOG is not suitable to be run on a common desktop machine in a reasonable amount of time (see Introduction section). In the following sections, the accuracy of various models trained with the PICA-to-go pipeline is evaluated and also compared to the existing pipeline (which uses HMMER/eggNOG for the clustering step). PICA crossvalidate.py performs a 5 fold crossvalidation with 10 replicates (n=5*10). A two-sided t-test for unequal variances (Welch's test) is performed to test if there are any significant differences regarding the accuracy of the models when different methods (e.g. clustering tools) are used. For all significant differences, the effect size (Cohen's d) is calculated as well as the raw difference between the means (RMD) to not only evaluate if there is a significant difference, but also provide some information about the magnitude.

2.1 Selection of a clustering tool

OrthoFinder OrthoFinder has to perform an all vs. all pairwise sequence alignment in its workflow. This is usually done by BLAST (in the following named OrthoFinder/blast), however OrthoFinder also offers the possibility of using DIAMOND for this purpose (OrthoFinder/diamond). To evaluate the runtime of OrthoFinder/blast, a set of predicted and translated genes gained from 3, 6 or 9 bins, is clustered using OrthoFinder/blast. All genes were predicted and translated using Prodigal (Hyatt et al., 2010). The used bins were randomly taken from the PEN_180 model. As the BLAST all vs. all step is speed limiting, a runtime of $O(n^2)$ is assumed.

By comparing the runtime of OrthoFinder, which uses the predicted genes of the previously randomly bins as input, the runtime seems to follow roughly the formula $(0.6x)^2 = runtime(minutes)$, where x is the number of bins. This means that

n (number of bins)	runtime (minutes)
3	3.41
6	13.52
9	28.43

Table 1: The runtime of OrthoFinder/blast using random bins from the PEN_180 model

clustering all protein files of the trainingdata of the PEN_180 model (180 bins) would take >190 hours. On the Life Science Compute Cluster (see Methods for specifications on the hardware), OrthoFinder/blast was run with the trainingdata of the PEN_180 model on four cores to test if this rough estimate is valid. The job was cancelled after four days as the runtime requirement (clustering in less than 8 hours on a desktop machine) was clearly exceeded. Therefore, the usage of OrthoFinder/blast is not an alternative to replace HMMER/eggNOG in the pipeline.

A rough runtime estimation of OrthoFinder/diamond was done in the same way as for OrthoFinder/blast, however a set of 6 and 12 bins was used. Assuming that the runtime follows roughly the formula $(0.3x)^2 = runtime(minutes)$, the runtime of OrthoFinder with 180 bins (number of bins in the PEN_180 model) would be nearly 60 hours.

Per default OrthoFinder uses DIAMOND with the –more-sensitive option, which offers a better sensitivity at cost of a higher runtime. This can be changed in a configuration file. OrthoFinder was run with DIAMOND without the –more-sensitive attribute (noted as OrthoFinder/diamond(default)) on predicted genes of 12 bins. Assuming a quadratic big O (DIAMOND all vs. all) and comparing the runtime of OrthoFinder using the randomly selected bins, the runtime would be $(0.19x)^2 = runtime(minutes)$ and 180 bins should be finished in about 20 hours. Though the runtime requirement of clustering the predicted genes in less than 8 hours was not fulfilled, OrthoFinder/diamond(default) was run on the Life Science Compute Cluster with four cores (simulation of the ordinary desktop machine used before) to see if the rough estimation about the runtime is valid and to use the clusterings as input for PICA-crossvalidate.py (to get a mean balanced accuracy of

n (number of bins)	runtime (minutes)
6	4.27
12	15.63

Table 2: The runtime of OrthoFinder/diamond(–more-sensitive) using random bins from the PEN_180 model

n (number of bins)	runtime (minutes)
12	4.64

Table 3: The runtime of OrthoFinder/diamond(default) using random bins from the PEN_180 model

the resulting model). The job took about one day and the mean balanced accuracy of PICA-crossvalidate.py was 0.87.

MMSeqs2 As OrthoFinder does not fulfil the requirements regarding the runtime, the usage of MMSeqs2/cluster and MMSeqs2/linclust was evaluated. While both tools have the same default cut-off regarding coverage and e-value, the cut-off for the minimum sequence identity is considerably different (0.9 for MMSeqs2/linclust and 0.0 for MMSeqs2/cluster). Therefore, MMSeqs2/linclust is tested with the default cut-off as well as with a minimal sequence identity cut-off of 0.0 as both tools may be more comparable with identical cut-offs. When the tool is used with the default cutoffs, it is named MMSeqs2/linclust here, while it is called MMSeqs2/linclust-msi0 if it is used with a minimum sequence identity cutoff of 0.0.

MMSeqs2/linclust and MMSeqs2/linclust-msi0 managed to cluster all genes of the 180 bins of the PEN_180 model in less than one minute (0.75 minutes for MMSeqs2/linclust and 0.53 minutes for MMSeqs2/linclust-msi0), while MMSeqs2/cluster needed less than two minutes (1.72 minutes) for the same dataset.

To test if the clusterings gained by MMSeqs2/cluster and MMSeqs2/linclust are meaningful for our purpose, PICA-crossvalidate.py was run on the clustering output. With a mean balanced accuracy of 0.87 for MMSeqs2/linclust, 0.91 for MMSeqs2/linclust-msi0, 0.89 for OrthoFinder/diamond(default) and 0.90 for MMSeqs2/cluster, OrthoFinder does not seem to have any advantage over the usage of MMSeqs2 regarding the mean balanced accuracy. Considering the improvements

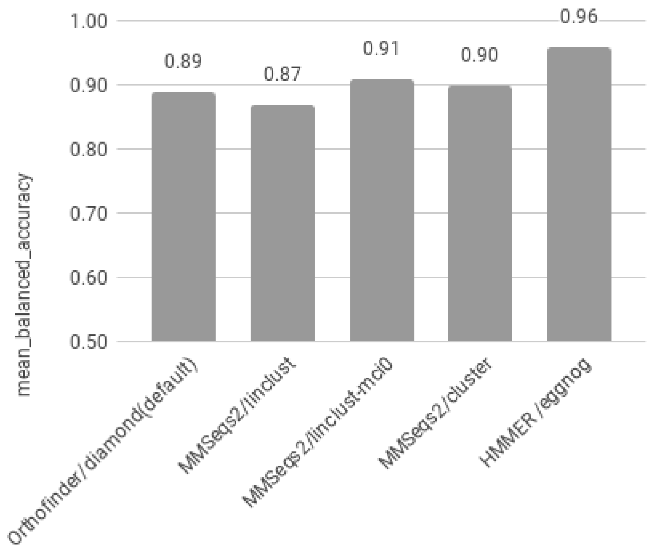


Figure 6: Comparison of the mean balanced accuracy of PICA-crossvalidate.py using the clustering output of different tools as input

concerning the runtime, the choice of the clustering tool was therefore in favour of MMSeqs2. As MMSeqs2/cluster and MMSeqs2/linclust share the same interface, both tools were incorporated in PICA-to-go. The MMSeqs2 software consists furthermore of a program (MMSeqs2/clusterupdate) that easily allows updating an existing clustering database with new genes. This is essential for the predict mode of PICA-to-go (see Methods section).

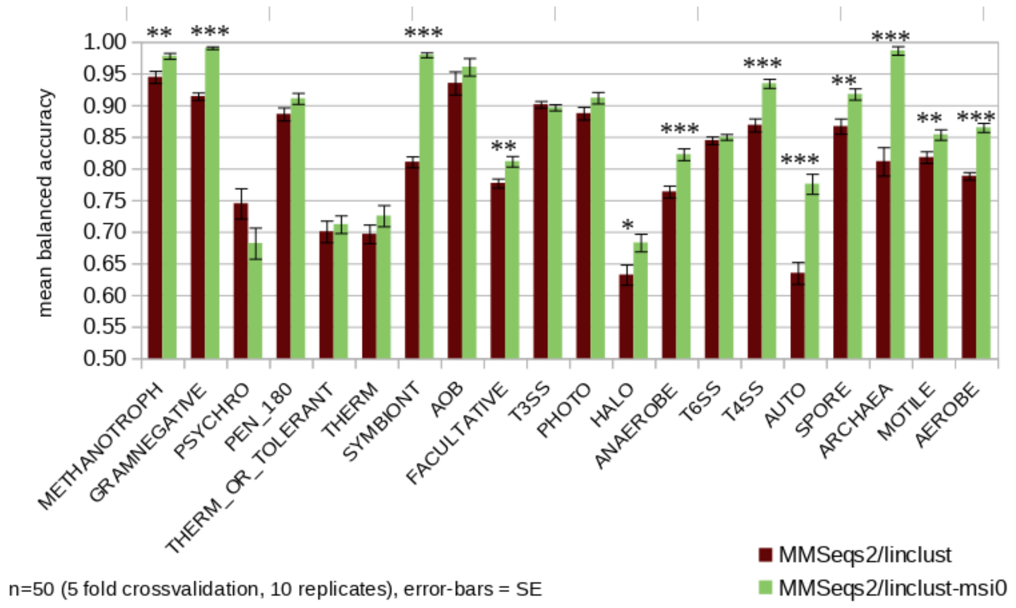
2.2 Computational requirements

The PICA-to-go pipeline was run with all models specified in the Methods section using MMSeqs2/cluster as well as MMSeqs2/linclust. It was possible to run all models (except the ARCHAEA model) on an ordinary laptop (4 cores, 8GB RAM; see Methods section). The runtime of the whole PICA-to-go pipeline ranged from a few minutes to a few hours. In none of the cases did the runtime exceed the requirement of eight hours. For some models with a higher amount of trainingdata, a temporary folder on the hard-disk had to be specified (instead of having all temporary files in RAM). The ARCHAEA model had to be run on the Life Science Compute Cluster as amount of trainingdata (1,800 bins) is higher than for most other models (which consist of maximal 450 bins).

2.3 MMSeqs2/linclust vs. MMSeqs2/linclust-msi0

In the previous section, it was shown that the usage of a cutoff of 0.0 for the minimal sequence identity lead to a slightly higher balanced accuracy than the default cutoff of 0.9 in MMSeqs2/linclust for the PEN_180 model. In the following, the PICA-to-go pipeline is run on all 45 PICA models that we currently use (including a re-run for the PEN_180 model [1]). For the clustering step, MMSeqs2/linclust was used and different cut-offs (0.0 and 0.9) for the minimal sequence identity were specified. In 23 cases (out of 45 models in total), there was a significant difference ($p<0.05$) between the usage of MMSeqs2/linclust and MMSeqs2/linclust-msi0. In all of these cases, the latter performed better. In five cases (AUTO, SYMBIONT, ARCHAEA, cnta, coos), this was especially pronounced with a mean balanced accuracy that differed by > 0.10. In conclusion, it therefore seems that a cut-off of 0.0 for the minimal sequence identity leads to better results than the default value of 0.9.

[1]Note that the mean balanced accuracy is not deterministic and is subject to variations as the samples for the test set are chosen randomly by PICA



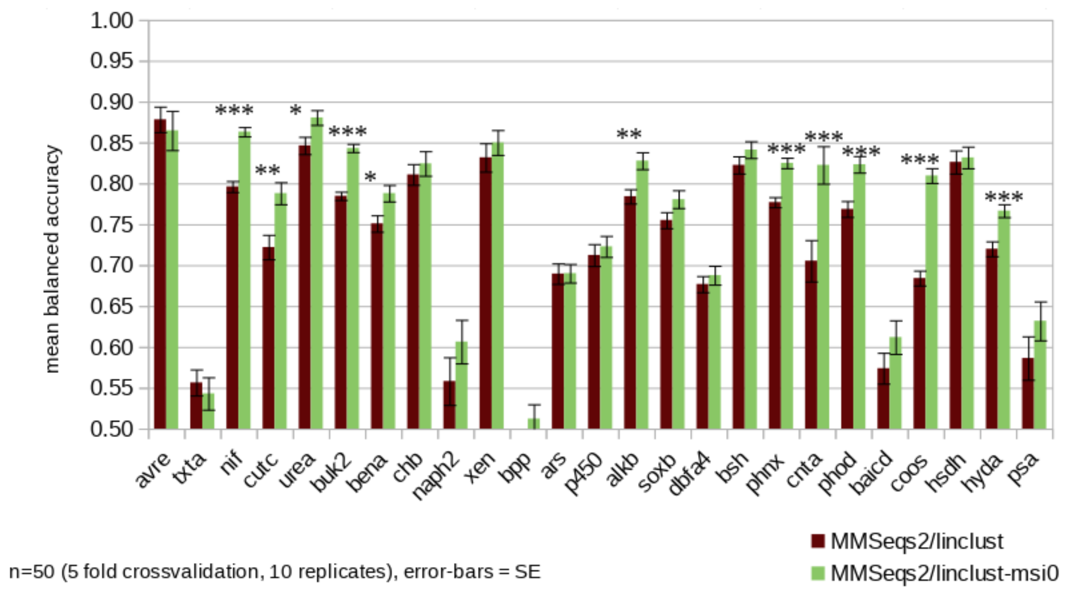
	MMSeqs2/linclust	MMSeqs2/linclust-msi0	significance
METHANOTROPH	0.94	0.98	**
GRAMNEGATIVE	0.91	0.99	***
PSYCHRO	0.74	0.68	
PEN_180	0.89	0.91	
THERM_OR_TOLERANT	0.70	0.71	
THERM	0.70	0.73	
SYMBIONT	0.81	0.98	***
AOB	0.94	0.96	
FACULTATIVE	0.78	0.81	**
T3SS	0.90	0.90	
PHOTO	0.89	0.91	
HALO	0.63	0.68	*
ANAEROBE	0.76	0.82	***
T6SS	0.84	0.85	
T4SS	0.87	0.93	***
AUTO	0.63	0.78	***
SPORE	0.87	0.92	**
ARCHAEA	0.81	0.99	***
MOTILE	0.82	0.85	**
AEROBE	0.79	0.86	***

Figure 7: Comparison of the mean balanced accuracy of different models using MMSeqs2/linclust with different cut-offs (0.9 or 0.0 for minimal sequence identity). (1)

* = significant ($\alpha < 0.05, \alpha > 0.01$)

** = highly significant ($\alpha < 0.01, \alpha > 0.001$)

*** = very highly significant ($\alpha < 0.001$)



	MMSeqs2/linclust	MMSeqs2/linclust-msi0	significance
avre	0.88	0.86	
txta	0.56	0.54	
nif	0.80	0.86	***
cutc	0.72	0.79	**
urea	0.85	0.88	*
buk2	0.78	0.84	***
bena	0.75	0.79	*
chb	0.81	0.82	
naph2	0.56	0.61	
xen	0.83	0.85	
bpp	0.48	0.51	
ars	0.69	0.69	
p450	0.71	0.72	
alkb	0.78	0.83	**
soxb	0.76	0.78	
dbfa4	0.68	0.69	
bsh	0.82	0.84	
phnx	0.78	0.83	***
cnta	0.71	0.82	***
phod	0.77	0.82	***
baicd	0.57	0.61	
coos	0.68	0.81	***
hsdh	0.83	0.83	
hyda	0.72	0.77	***
psa	0.59	0.63	

Figure 8: Comparison of the mean balanced accuracy of different models using MMSeqs2/linclust with different cut-offs (0.9 or 0.0 for minimal sequence identity). (2)

* = significant ($\alpha < 0.05, \alpha > 0.01$)

** = highly significant ($\alpha < 0.01, \alpha > 0.001$)

*** = very highly significant ($\alpha < 0.001$)

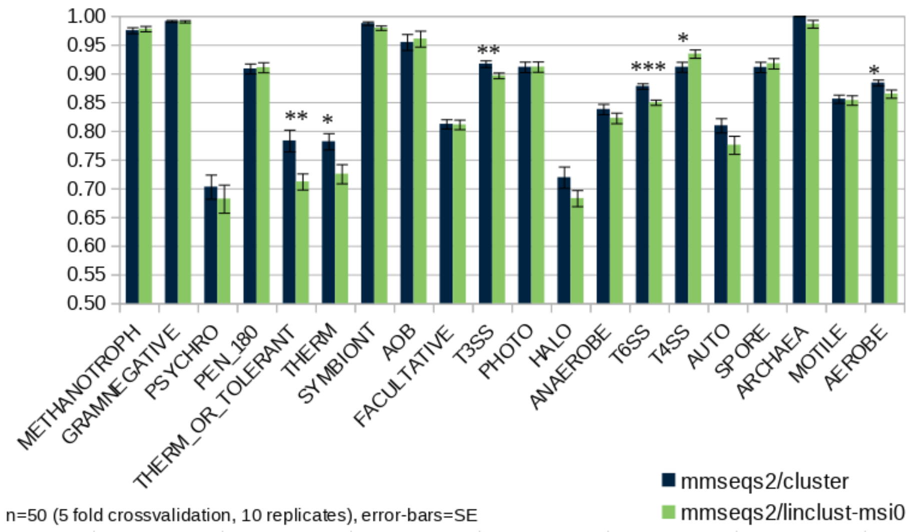
	MMSeqs2/linclust	MMSeqs2/linclust-msi0	significance	Cohen's d	RMD
METHANOTROPH	0.94	0.98	**	**	
ANAEROBE	0.76	0.82	***	***	*
GRAMNEGATIVE	0.91	0.99	***	***	*
T4SS	0.87	0.93	***	***	*
nif	0.80	0.86	***	***	*
alkb	0.78	0.83	**	**	*
cutc	0.72	0.79	**	**	*
urea	0.85	0.88	*	*	
AUTO	0.63	0.78	***	***	**
SYMBIONT	0.81	0.98	***	***	**
SPORE	0.87	0.92	**	**	*
ARCHAEA	0.81	0.99	***	***	**
buk2	0.78	0.84	***	***	*
MOTILE	0.82	0.85	**	**	
phnx	0.78	0.83	***	***	*
cnta	0.71	0.82	***	**	**
FACULTATIVE	0.78	0.81	**	**	
bena	0.75	0.79	*	**	
phod	0.77	0.82	***	**	*
AEROBE	0.79	0.86	***	***	*
HALO	0.63	0.68	*	*	*
coos	0.68	0.81	***	***	**
hyda	0.72	0.77	***	**	*

Table 4: Comparison of the mean balanced accuracy of different models using MMSeqs2/linclust with different cut-offs (0.9 or 0.0 for minimal sequence identity): Detailed inspection of all models with significant differences using the effect size (Cohen's d) and the raw mean difference (RMD).

for Cohens d:
* = small effect ($d > 0.2, d < 0.5$)
** = medium effect ($d > 0.5, d < 0.8$)
*** = large effect ($d > 0.8$)
for RMD:
* = considerable difference ($\Delta > 0.04, \Delta < 0.10$)
** = large difference ($\Delta > 0.10, \Delta < 0.20$)
*** = very large difference ($\Delta > 0.20$)

2.4 MMSeqs2/cluster vs. MMSeqs2/linclust-msi0

As MMSeqs2/cluster uses cascaded clustering, it is expected to be significantly slower, but should lead to a more accurate clustering than MMSeqs2/linclust (Steinegger and Söding, 2018). To investigate if the accuracies of the PICA models are affected by this, the usage of MMSeqs2/linclust-mci0 in PICA-to-go is compared with the usage of MMSeqs2/cluster. In 12 cases out of 45 models in total, there was a significant ($\alpha < 0.05$) difference regarding the mean balanced accuracy between the usage of MMSeqs2/cluster and MMSeqs2/linclust-mci0. Of these 12 models with significant differences, three were considered as "very highly significant" ($\alpha < 0.001$) and three as "highly significant". Ten cases with significant differences were in favour of MMSeqs2/cluster, while in two cases (T4SS, cutc) MMSeqs2/linclust-mci0 lead to a better balanced accuracy. This effect was, however, very small for T4SS (0.91 vs. 0.93), but more considerable regarding cutc (0.74 vs. 0.79). All in all, in only four cases, the raw mean difference is larger than 0.04. So in conclusion, MMSeqs2/linclust-mci0 lead to a worse performance than MMSeqs2/cluster. Nevertheless, the effect itself was not very pronounced.



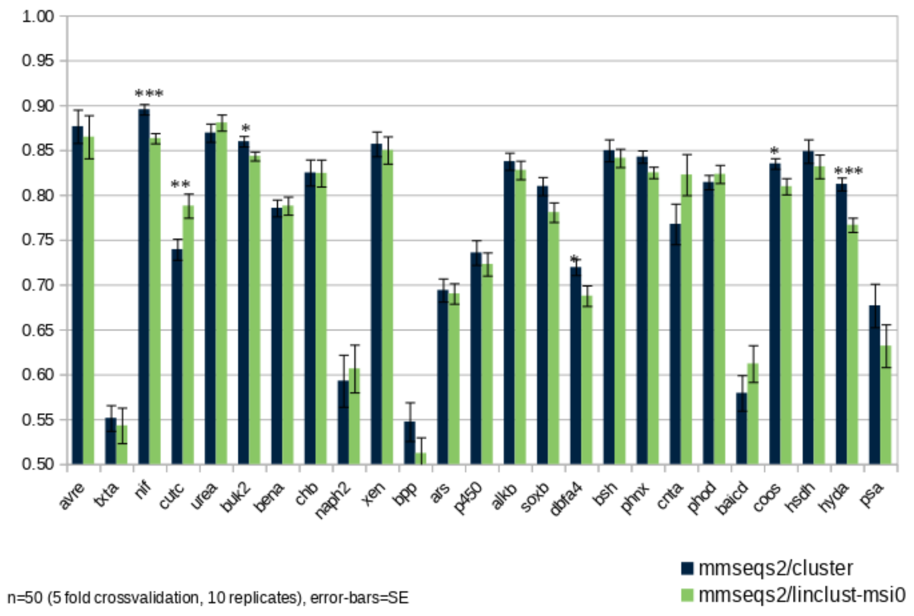
	MMSeqs2/cluster	MMSeqs2/linclust-mci0	significance
METHANOTROPH	0.98	0.98	
GRAMNEGATIVE	0.99	0.99	
PSYCHRO	0.70	0.68	
PEN_180	0.91	0.91	
THERM_OR_TOLERANT	0.78	0.71	**
THERM	0.78	0.73	*
SYMBIONT	0.99	0.98	
AOB	0.95	0.96	
FACULTATIVE	0.81	0.81	
T3SS	0.92	0.90	**
PHOTO	0.91	0.91	
HALO	0.72	0.68	
ANAEROBE	0.84	0.82	
T6SS	0.88	0.85	***
T4SS	0.91	0.93	*
AUTO	0.81	0.78	
SPORE	0.91	0.92	
ARCHAEA	1.00	0.99	
MOTILE	0.86	0.85	
AEROBE	0.88	0.86	*

Figure 9: Comparison of the mean balanced accuracy of each model using MMSeqs2/cluster or MMSeqs2/linclust, as clustering step. (1)

* = significant ($\alpha < 0.05, \alpha > 0.01$)

** = highly significant ($\alpha < 0.01, \alpha > 0.001$)

*** = very highly significant ($\alpha < 0.001$)



	MMSeqs2/cluster	MMSeqs2/linclust	significance
avre	0.88	0.86	
txta	0.55	0.54	
nif	0.90	0.86	***
cutc	0.74	0.79	**
urea	0.87	0.88	
buk2	0.86	0.84	*
bena	0.79	0.79	
chb	0.83	0.82	
naph2	0.59	0.61	
xen	0.86	0.85	
bpp	0.55	0.51	
ars	0.69	0.69	
p450	0.74	0.72	
alkb	0.84	0.83	
soxb	0.81	0.78	
dbfa4	0.72	0.69	*
bsh	0.85	0.84	
phnx	0.84	0.83	
cnta	0.77	0.82	
phod	0.81	0.82	
baicd	0.58	0.61	
coos	0.84	0.81	*
hsdh	0.85	0.83	
hyda	0.81	0.77	***
psa	0.68	0.63	

Figure 10: Comparison of the mean balanced accuracy of each model using MMSeqs2/cluster or MMSeqs2/linclust, as clustering step. (2)
* = significant ($\alpha < 0.05, \alpha > 0.01$)
** = highly significant ($\alpha < 0.01, \alpha > 0.001$)
*** = very highly significant ($\alpha < 0.001$)

	MMSeqs2/cluster	MMSeqs2/linclust-msi0	significance	Cohen's d	RMD
dbfa4	0.72	0.69	*	*	
buk2	0.86	0.84	*	*	
THERM_OR_TOLERANT	0.78	0.71	**	**	*
T6SS	0.88	0.85	***	***	
T4SS	0.91	0.93	*	*	
T3SS	0.92	0.90	**	**	
AEROBE	0.88	0.86	*	*	
nif	0.90	0.86	***	**	
THERM	0.78	0.73	*	**	*
coos	0.84	0.81	*	*	
cutc	0.74	0.79	**	**	*
hyda	0.81	0.77	***	***	*

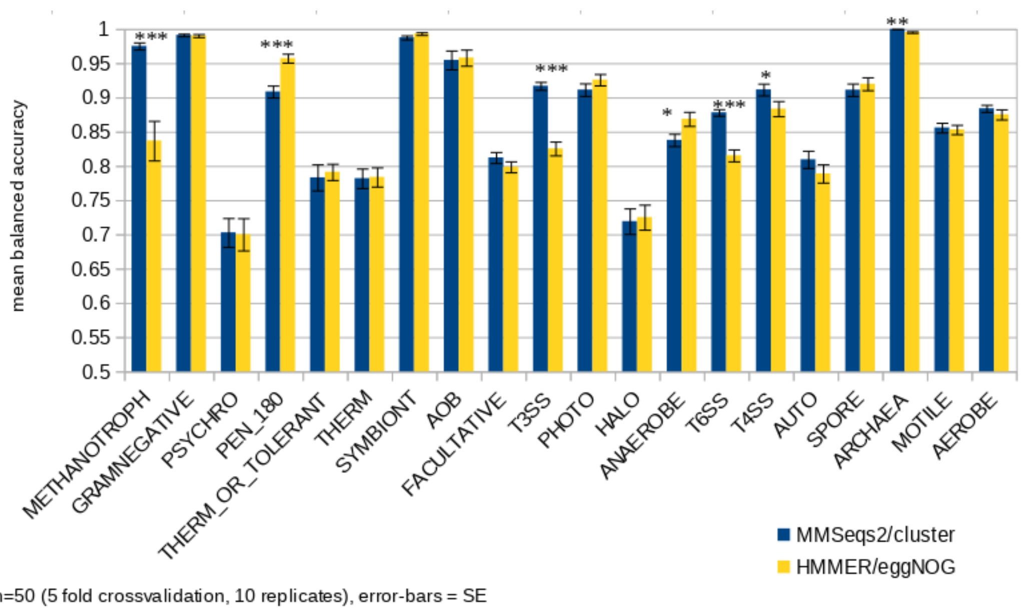
Table 5: Pica-to-go (using MMSeqs2/cluster) vs. PICA-to-go (using MMSeqs2/linclust): Detailed inspection of all models with significant differences using the effect size (Cohen's d) and the raw mean difference (RMD).
for Cohens d:
* = small effect ($d > 0.2, d < 0.5$)
** = medium effect ($d > 0.5, d < 0.8$)
*** = large effect ($d > 0.8$)
for RMD:
* = considerable difference ($\Delta > 0.04, \Delta < 0.10$)
** = large difference ($\Delta > 0.10, \Delta < 0.20$)
*** = very large difference ($\Delta > 0.20$)

2.5 Comparison of PICA-to-go to the existing pipeline

The mean balanced accuracy of the models resulting from applying MMSeqs2/cluster was compared to the performance of the existing pipeline. MMseqs2/linclust is not listed in the table as it does not perform as well as MMSeqs2/cluster for most models(see previous section).

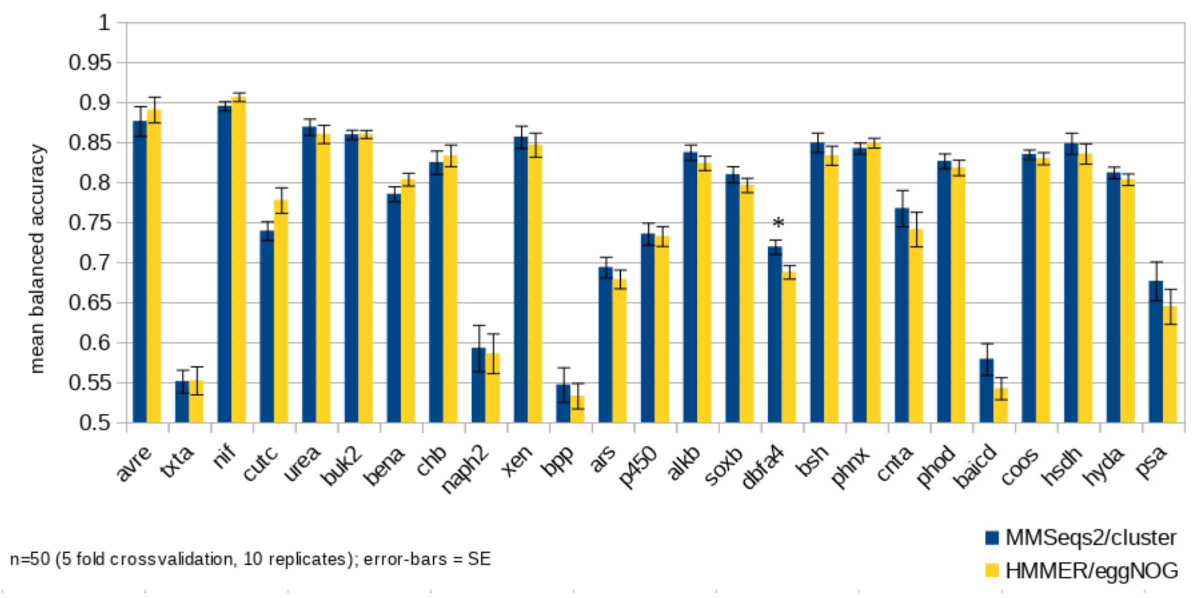
In six cases, the usage of PICA-to-go (using MMSeqs2/cluster) lead to a better mean balanced accuracy than the existing pipeline (using HMMER/eggNOG). These cases were the METHANOTROPH, ARCHAEA, dbfa4, T6SS, T4SS and the T3SS model. For two cases (ANAEROBE, PEN_180), the existing pipeline performed better. The differences regarding the mean balanced accuracy were especially pronounced in the METHANOTROPH model (0.84 vs. 0.98 mean balanced accuracy).

All in all, PICA-to-go seems to perform as well (or slightly better) as the existing pipeline with only two cases were PICA-to-go performed marginally worse.



	MMSeqs2/cluster	HMMER/eggNOG	significance
METHANOTROPH	0.98	0.84	***
GRAMNEGATIVE	0.99	0.99	
PSYCHRO	0.70	0.70	
PEN_180	0.91	0.96	***
THERM_OR_TOLERANT	0.78	0.79	
THERM	0.78	0.78	
SYMBIONT	0.99	0.99	
AOB	0.95	0.96	
FACULTATIVE	0.81	0.80	
T3SS	0.92	0.83	***
PHOTO	0.91	0.93	
HALO	0.72	0.73	
ANAEROBE	0.84	0.87	*
T6SS	0.88	0.82	***
T4SS	0.91	0.88	*
AUTO	0.81	0.79	
SPORE	0.91	0.92	
ARCHAEA	1.00	1.00	**
MOTILE	0.86	0.85	
AEROBE	0.88	0.88	

Figure 11: Comparison of the mean balanced accuracy of the existing pipeline (using HMMER/eggNOG) and PICA-to-go using MMSeqs2/cluster (1).
* = significant ($\alpha < 0.05, \alpha > 0.01$)
** = highly significant ($\alpha < 0.01, \alpha > 0.001$)
*** = very highly significant ($\alpha < 0.001$)



	MMSeqs/cluster	HMMER/eggNOG	significance
avre	0.88	0.89	
txta	0.55	0.55	
nif	0.90	0.91	
cutc	0.74	0.78	
urea	0.87	0.86	
buk2	0.86	0.86	
bena	0.79	0.80	
chb	0.83	0.83	
naph2	0.59	0.59	
xen	0.86	0.85	
bpp	0.55	0.53	
ars	0.69	0.68	
p450	0.74	0.73	
alkb	0.84	0.82	
soxb	0.81	0.80	
dbfa4	0.72	0.69	*
bsh	0.85	0.83	
phnx	0.84	0.85	
cnta	0.77	0.74	
phod	0.83	0.82	
baicd	0.58	0.54	
coos	0.84	0.83	
hsdh	0.85	0.84	
hyda	0.81	0.80	
psa	0.68	0.65	

Figure 12: Comparison of the mean balanced accuracy of the existing pipeline (using HMMER/eggNOG) and PICA-to-go using MMSeqs2/cluster (2).

* = significant ($\alpha < 0.05, \alpha > 0.01$)

** = highly significant ($\alpha < 0.01, \alpha > 0.001$)

*** = very highly significant ($\alpha < 0.001$)

	MMSeqs2/cluster	HMMER/eggNOG	significance	Cohen's d	RMD
METHANOTROPH	0.98	0.84	***	***	**
ANAEROBE	0.84	0.87	*	*	
ARCHAEA	1.00	1.00	**	**	
dbfa4	0.72	0.69	*	**	
PEN_180	0.91	0.96	***	***	*
T6SS	0.88	0.82	***	***	*
T4SS	0.91	0.88	*	*	
T3SS	0.92	0.83	***	***	*

Table 6: Pica-to-go (using MMSeqs2/cluster) vs. HMMER/eggNOG: Detailed inspection of all models with significant differences using the effect size (Cohen's d) and the raw mean difference (RMD).

for Cohens d:

* = small effect ($d > 0.2, d < 0.5$)

** = medium effect ($d > 0.5, d < 0.8$)

*** = large effect ($d > 0.8$)

for RMD:

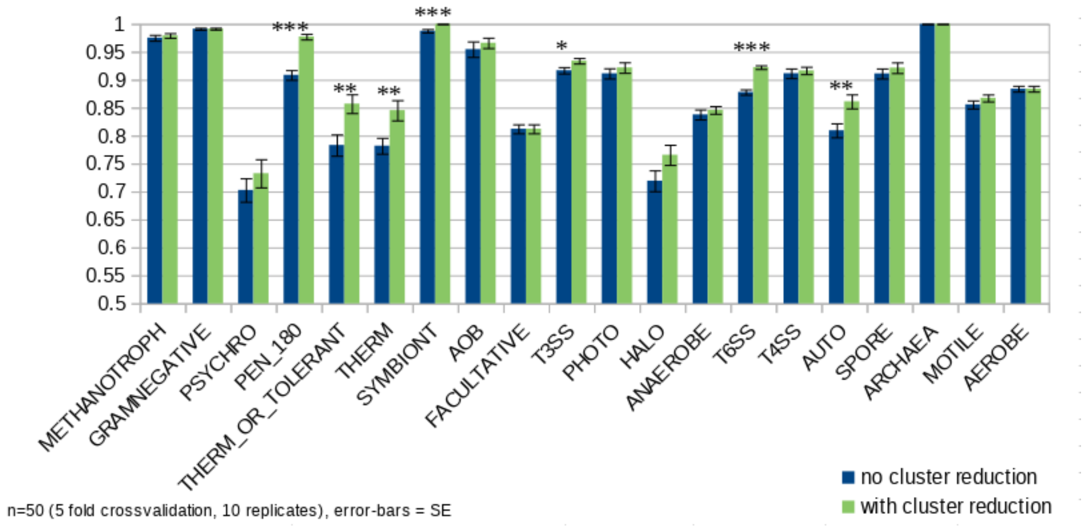
* = considerable difference ($\Delta > 0.04, \Delta < 0.10$)

** = large difference ($\Delta > 0.10, \Delta < 0.20$)

*** = very large difference ($\Delta > 0.20$)

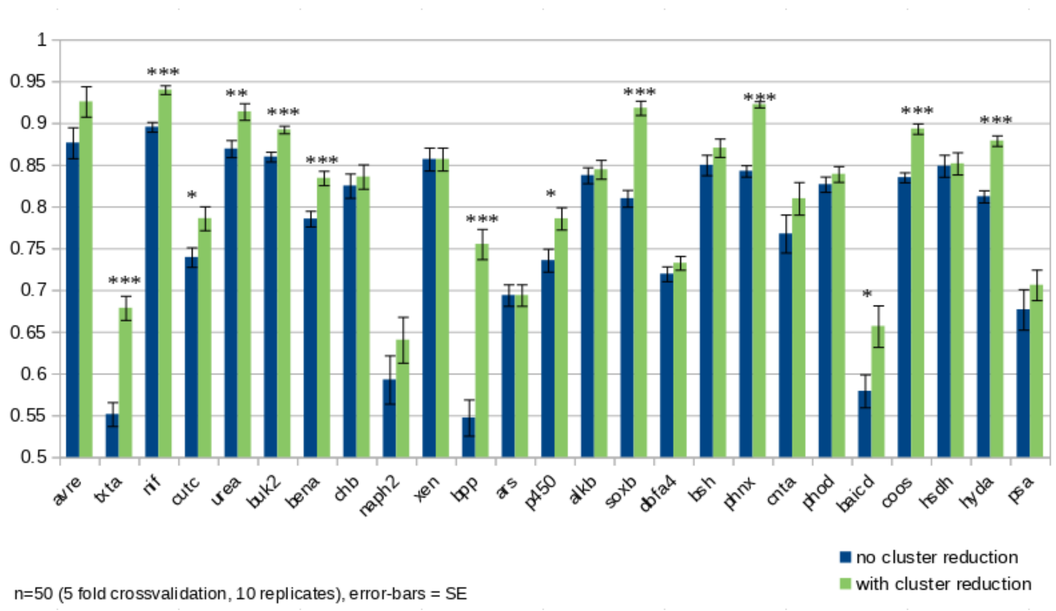
2.6 The effect of feature reduction based on cluster size

As specified in the Methods section, the -filter option in PICA-to-go performs a feature reduction step based on cluster size ("filtering"). All models were trained with this mode and the mean balanced accuracy was compared with and without the feature reduction step. All calculations were completed using the MMSeqs2/cluster mode of PICA-to-go. In about half of the models (20/45), the filtering step led to an significantly increased mean balanced accuracy ($\alpha < 0.05$). In 16 cases, the increase was even highly significant or very highly significant ($\alpha < 0.01$). The difference was especially pronounced in the bpp model (0.55 to 0.76), in the txta model (0.55 to 0.68) as well as in the soxb model (0.81 to 0.92).



	MMSeqs2/cluster	MMSeqs2/cluster + filtering	significance
METHANOTROPH	0.98	0.98	
GRAMNEGATIVE	0.99	0.99	
PSYCHRO	0.70	0.73	
PEN_180	0.91	0.98	***
THERM_OR_TOLERANT	0.78	0.86	**
THERM	0.78	0.85	**
SYMBIONT	0.99	1.00	***
AOB	0.95	0.97	
FACULTATIVE	0.81	0.81	
T3SS	0.92	0.93	*
PHOTO	0.91	0.92	
HALO	0.72	0.77	
ANAEROBE	0.84	0.85	
T6SS	0.88	0.92	***
T4SS	0.91	0.92	
AUTO	0.81	0.86	**
SPORE	0.91	0.92	
ARCHAEA	1.00	1.00	
MOTILE	0.86	0.87	
AEROBE	0.88	0.88	

Figure 13: Comparison of the mean balanced accuracy of PICA-to-go using MMSeqs2/cluster with and without feature reduction based on cluster size ("filtering")
* = significant ($\alpha < 0.05, \alpha > 0.01$)
** = highly significant ($\alpha < 0.01, \alpha > 0.001$)
*** = very highly significant ($\alpha < 0.001$)



	MMSeqs2/cluster	MMSeqs2/cluster + filtering	significance
avre	0.88	0.93	
txta	0.55	0.68	***
nif	0.90	0.94	***
cutc	0.74	0.79	*
urea	0.87	0.91	**
buk2	0.86	0.89	***
bena	0.79	0.83	***
chb	0.83	0.84	
naph2	0.59	0.64	
xen	0.86	0.86	
bpp	0.55	0.76	***
ars	0.69	0.69	
p450	0.74	0.79	*
alkb	0.84	0.84	
soxb	0.81	0.92	***
dbfa4	0.72	0.73	
bsh	0.85	0.87	
phnx	0.84	0.92	***
cnta	0.77	0.81	
phod	0.83	0.84	
baicd	0.58	0.66	*
coos	0.84	0.89	***
hsdh	0.85	0.85	
hyda	0.81	0.88	***
psa	0.68	0.71	

Figure 14: Comparison of the mean balanced accuracy of PICA-to-go using MMSeqs2/cluster with and without feature reduction based on cluster size ("filtering")
* = significant ($\alpha < 0.05, \alpha > 0.01$)
** = highly significant ($\alpha < 0.01, \alpha > 0.001$)
*** = very highly significant ($\alpha < 0.001$)

	MMSeqs2/cluster	MMSeqs2/cluster (+filtering)	significance	Cohen's d	RMD
bpp	0.55	0.76	***	***	***
PEN_180	0.91	0.98	***	***	*
THERM_OR_TOLERANT	0.78	0.86	**	**	*
T6SS	0.88	0.92	***	***	*
txta	0.55	0.68	***	***	**
nif	0.90	0.94	***	***	*
p450	0.74	0.79	*	**	*
THERM	0.78	0.85	**	**	*
cutc	0.74	0.79	*	**	*
urea	0.87	0.91	**	**	*
AUTO	0.81	0.86	**	**	*
SYMBIONT	0.99	1.00	***	***	
soxb	0.81	0.92	***	***	**
buk2	0.86	0.89	***	***	
phnx	0.84	0.92	***	***	*
bena	0.79	0.83	***	**	*
baicd	0.58	0.66	*	*	*
T3SS	0.92	0.93	*	*	
coos	0.84	0.89	***	***	*
hyda	0.81	0.88	***	***	*

Table 7: Pica-to-go (using MMSeqs2/cluster) with and without feature reduction based on cluster size (filtering): Detailed inspection of all models with significant differences using the effect size (Cohen's d) and the raw mean difference (RMD).

for Cohens d:

* = small effect ($d > 0.2, d < 0.5$)

** = medium effect ($d > 0.5, d < 0.8$)

*** = large effect ($d > 0.8$)

for RMD:

* = considerable difference ($\Delta > 0.04, \Delta < 0.10$)

** = large difference ($\Delta > 0.10, \Delta < 0.20$)

*** = very large difference ($\Delta > 0.20$)

2.7 Detailed analysis of selected models

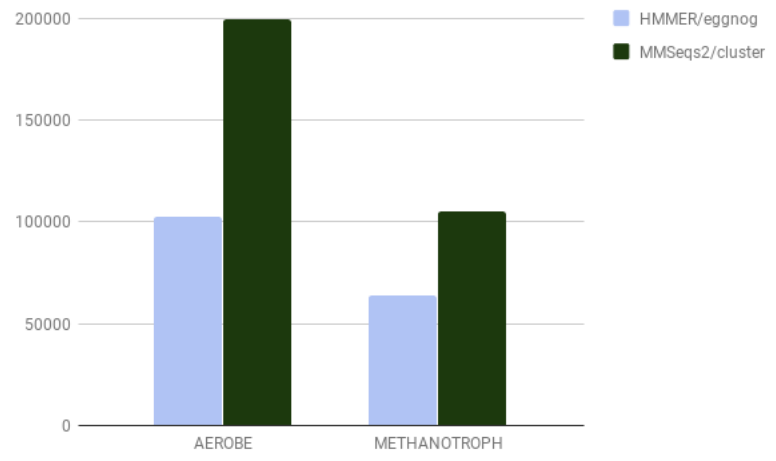
2.7.1 Comparison of clusterings

Some models were chosen to be examined in more detail as specified in the Methods section. The selection was in favour of the AEROBE model, as the mean balanced accuracy from the existing pipeline corresponds well with the results from PICA-to-go. Furthermore, the METHANOTROPH model was chosen, as it clearly outperforms the existing pipeline regarding the mean balanced accuracy. The main hypothesis is that an analysis of the difference of these clusterings should reveal that clusterings of a certain model that performed equally good as features for PICA, should be more similar to each other, than clusterings with a more significant difference in the mean balanced accuracy. So, I expect that the clusterings of HMMER/eggNOG and MMSeqs2/cluster using the AEROBE dataset are more similar than when this comparison is done using the METHANOTROPH dataset. For all comparisons the "unfiltered clustering set" (clustering set without any feature reduction based on cluster size) is taken, as these clustering comparison algorithms are only described comparing clusterings with the same size of items.

However, using the reverse Normalized Van Dongen (rNVD) measure, which returns a value between 0 and 1 (the latter for an identical clustering) we observe a value of 0.76 when comparing the clustering of HMMER/eggNOG and MMSeqs2/cluster using the dataset of the AEROBE model. As the rNVD index is symmetric and closely related to the asymmetric Meila-Heckerman measure (MH), the values of this index is shown too. Contrary to the hypothesis stated before, these values did not really differ from the values obtained using the METHANOTROPH dataset when comparing the clustering of HMMER/eggNOG to MMSeqs2/cluster.

AEROBE	
rNVD	0.76
MH (first vs. second)	0.77
MH (second vs. first)	0.75
METHANOTROPH	
rNVD	0.77
MH (first vs. second)	0.79
MH (second vs. first)	0.76

Table 8: Comparing the clustering of HMMER/eggNOG to MMSeqs2/cluster using rNVD and MH



	AEROBE	METHANOTROPH
HMMER/eggNOG	102197	63636
MMSeqs2/cluster	199268	104850

Table 9: Comparing the cluster size of the clusterings gained by HMMER/eggNOG and MMSeqs2/cluster

2.7.2 Relevant features

As PICA-to-go is not intended to serve as a "black box" for phenotype prediction, but to provide comprehensible predictions, the output of the feature ranking for the PEN_180 model (penicillin resistance) is examined. The output depicted here is not identical to the output of PICA-to-go as the column with the name of the representative sequence is not shown (for information about raw data, see 4.10). The annotations for the features of the PEN_180 model were gained from the remote blast option in PICA-to-go. For the PEN_180 model, the most relevant features are a beta-lactamase and the methicillin resistance proteins MecI and mecR1.

feature rank	contributing to	accession/version	annotation
0.73	YES	P00807.1	Beta-lactamase
0.70	YES	P68261.1	Methicillin resistance regulatory protein MecI
0.42	YES	P0A0A9.1	Methicillin resistance mecR1 protein
0.28	YES	O32225.1 | Q899J5.2	Uncharacterized membrane protein | Probable 2-phosphosulfolactate phosphatase
0.26	YES	P0A4U2.1	Cadmium resistance transcriptional regulatory protein CadC
0.24	YES	Q6AMN0.1	Fructose-16-bisphosphatase class 3
-0.22	NO	O31991.1	SPBc2 prophage-derived uncharacterized protein YolD
0.16	YES	Q7A3J7.1	Fibronectin-binding protein A
0.14	YES	C0H423.1	Uncharacterized membrane protein YozV
0.14	YES	P94356.2	Uncharacterized protein YxkC

Table 10: The top 10 features contributing to the PEN_180 model, '|' indicated that two clusters were pooled into one group in PICA as they always co-occurred

2.8 The workflow of PICA-to-go explained on a newly trained model

To show the application field of PICA-to-go a machine learning, a model for ciprofloxacin resistance in Acinetobacter baumannii was trained. The training data was gained from the Pathogen Browser (https://www.ncbi.nlm.nih.gov/pathogens/). All entries that are known to be classified as sensitive or resistant (intermediate was not considered) using antimicrobial susceptibility testing methods (AST), and for which a genome assembly is available, were considered. To get a balanced train set (same number of YES/NO labels), all instances of the less abundant class are considered, while a random sample of the same size was taken for the class with the higher abundance. Using this process, a training set of 224 samples (112 YES, 112 NO) was obtained for the ciprofloxacin model. PICA-to-go was able to train this model (using MMSeqs2/cluster, without the feature-reduction option) in less than ten minutes on a normal desktop machine. 4GB of RAM was needed. The mean balanced accuracy was 0.93. All raw data is available in the corresponding Github repository of PICA-to-go.

3 Discussion

3.1 PICA-to-go to link genes to a certain phenotype

PICA-to-go provides the user with a list of features (gene clusters) stating the relevance for each feature for the prediction with a value between 0 and 1 (see 4.4.4). This not only makes predictions more comprehensible, as PICA-to-go does not act as a black box, but also allows for linking genes to a certain phenotype. This could provide new insights if this gene has not been linked to this phenotype before. As shown in 2.7.2, three genes known to cause antibiotic resistance, are the most relevant features for the PEN_180 model. Some other relevant proteins, however, were not clearly linked to antibiotic resistance before. PICA-to-go may provide a clue to further investigate these genes. However, a correlation of a gene with a certain phenotype does not always have to be the cause of a certain phenotype. For the PEN_180 model, an example would be the cadmium resistance gene that is linked with penicillin resistance. The correlation between heavy metal tolerance and antibiotic resistance is well known and seems to be caused by co-selection (Baker-Austin et al., 2006). Moreover, PICA-to-go cannot replace classical statistical techniques for correlating genes to a certain phenotype as the aim of support vector machines is to find a perfect split between data to train a model to make certain predictions and not to test for statistical significance.

3.2 Relevance for the field of bioinformatics

PICA-to-go is the first tool for the training of machine learning models for phenotype prediction that is intended to be used on a normal desktop machine and is able to to run most of our machine learning models (except ARCHAEA) in an appropriate amount of time (ranging from a few minutes to a few hours). Furthermore, it outperforms previous approaches using HMMER/eggNOG not only regarding the ultra fast speed, but in some cases also regarding the accuracy. Moreover, it comes with a user friendly interface and can be easily deployed as it is offered as Singularity container. However, PICA-to-go is not limited to desktop users, but may also be beneficial to use on a computer cluster due to its advantages over other approaches.

3.3 Interpretation of the results

Orthologous groups vs. gene clusters The accuracy of PICA-to-go is closely linked to the identification of clusters of functional units in the genome, which are used as features for machine learning. Previous studies using PICA relied on the usage of clusters of orthologous groups, which is based on the idea that orthologous groups are functionally conserved. My results show, that a simple gene clustering, without the ambition to identify orthologous groups, lead to comparable results regarding the accuracy of the machine learning models as the usage of orthologous groups (eggNOGs).

Cascaded clustering Moreover, the difference between the usage of MMSeqs2/linclust vs. MMSeqs2/cluster seems to be comparably low. The difference between both tools is the usage of cascaded clustering in the latter tool, which seems to lead to slightly better results for some models at the cost of a longer runtime.

Used cutoffs MMSeqs2 uses three cutoffs based on sequence identity (0.0), coverage (0.8) and e-value (0.001). For MMSeqs2/linclust two different cut-offs for sequence identity were tested (0.9 and 0.0). The results showed that the cut-off of 0.9 lead to worse predictions. It seems likely that a cut-off of 0.9 leads to the case that functional similar genes are separated into different clusters. A cut-off of 0.0 for sequence identity on the other hand could be still able to lead to a necessary separation of clusters due to the fact that there is still a cut-off for the e-value and for the coverage. As further different cut-off settings were not tested, however, it is still difficult to make an assumption about optimal settings. In machine learning, the automatic, exhaustive, variation of various cut-offs in other to gain a better prediction for a single model is called grid-search. However, this term is commonly applied to different default settings of the machine learning algorithm itself (e.g. for SVM: used kernel, C value) rather than for the process of the selection of features (Mueller and Guido, 2016). Such a method may lead to better prediction and is similar to the optional feature reduction step (called LinearFiltering), however, it applying such an approach would drastically decrease the performance and is therefore counter-intuitive to the original purpose of PICA-to-go

3.4 Further improvements

In the PICA-to-go pipeline, PICA itself is in general not a performance bottleneck, except for large models (such as ARCHAEA). Besides libSVM, which is used in PICA, other libraries for support vector machines are available. However, libSVM is one of the most used libraries for that purpose. In the past, several scientific tools have been developed that use GPU computing, which is, nevertheless, not a trivial task. In addition, GPU acceleration may only be useful for certain computations (e.g. matrix operations). Moreover, the portability may be limited to certain hardware types, if hardware-specific tools such as CUDA are used instead of cross-platform tools (such as OpenCL). Furthermore GPU computing is not always useful as not every computer cluster is equipped with graphic cards. For a program that is primarily designed to be used on a normal desktop machine, this approach may be quite useful (as personal computers normally consist of a GPU). ThunderSVM is an GPU accelerated SVM implementation which is over 100 times faster than libSVM according to the authors with Python bindings available. (Wen et al., 2018) To further improve the performance of the PICA-to-go pipeline, exchanging PICA with a modified PICA version that uses ThunderSVM could be considered.

4 Methods

4.1 Definition of the task

The following business rules have been extracted as requirements for PICA-to-go:

- PICA-to-go should be able to train machine learning models (training mode)
- Besides the training mode, there should be a phenotype prediction mode using models that have been trained before
- It should be able to perform all calculations on a modern desktop machine (no computer cluster needed)
- The program should work on most major Linux distributions
- The program should accept protein files as well as nucleotide files
- Clusters of sequences used as machine learning feature should be comprehensible by the user

- The program should be able to deliver as good predictions as the existing pipeline when used with the same training data

4.2 Workflow to find a replacement for HMMER/eggNOG

Overcoming the bottleneck of using HMMER/eggNOG would require clustering all sequences in the training-data with each other and thereby generating a small database of clusters, instead of mapping them to an existing, universal database. Certain tools exist for clustering sequences by identifying homologous relationships (see Introduction/Orthologous groups). Furthermore, for the predict mode of PICA-to-go, it is necessary to find a method to cluster new sequences to an existing clustering database (that has been generated in the PICA-to-go train mode).

To select a suitable tool to use for PICA-to-go, three requirements have to be met: (1) The tool has to run on a desktop machine with a suitable runtime (it should be possible to train most of our models in less than 8 hours). (2) Furthermore, the clusterings should be comprehensible. (3) A method has to be found for clustering new sequences to an existing clustering database.

To test if these requirements are met by any existing tool, the following procedure was considered: (1) My personal computer (Acer Aspire E15 E5-571G-50K9) will be used as a reference for all speed testing, as its hardware performance (8GB RAM, 4 Cores) may be a good representation of an ordinary personal computer. (2) At first a small set of translated genomes will be used to determine the runtime of the program. (3) A suitable regression model will be applied to get a runtime estimation when used with a larger set of translated genomes. This may only be a rough estimation with limited meaningfulness. (4) If the runtime estimation is roughly within the limit for 200 bins, the translated genomes of the PEN_180 model, which were also used for PhenDB, will be clustered. (5) To test if these clusterings are comprehensible, they are used as input for PICA crossvalidate. The mean balanced accuracy of the output is then compared to the mean balanced accuracy in the existing pipeline (HMMER/eggNOG) that was used for PhenDB. (6) It is necessary to find a method to map new sequences to an existing database (if the tool itself does not provide such a method) (7) The steps 4 and 5 are repeated with all other models used in PhenDB. (8) For selected models, the clusters of the used clustering tool are directly compared with the clusters of eggNOG/HMMER (genomes with the same eggNOG id form a cluster).

OrthoFinder, MMSeqs2/linclust and MMSeqs2/cluster were evaluated with this procedure and finally MMSeqs2/cluster and MMSeqs2/linclust were chosen to be used in the pipeline.

4.3 The existing pipeline

The PICA pipeline uses HMMER to map certain genes to the eggNOG database. The pipeline returns the best hit (best e-value) if the e-value is below a threshold of 0.05. Eventually, a crossvalidation is performed using PICA crossvalidate.py and the model is trained using PICA train.py.

4.4 The PICA-to-go pipeline

PICA-to-go offers two different modes: train and predict depending if a model is trained or an existing model is used to perform predictions on bins. The PICA-to-go pipeline consists of three major components: Gene finding, gene clustering and the training/prediction module. All these functions are covered by external components (Prodigal, MMSeqs2/cluster or MMSeqs2/linclust and PICA). These components are called from the PICA-to-go pipeline written in Java 8 providing a user friendly interface and some additional minor components. Moreover, it acts as a compatibility layer between these components as the required output and input format differs between these components and may require extensive parsing. Additional minor components include (1) file sanity checking (including detection if a file is a protein or a nucleotide file), (2) generating a json file (gene_clusters.json) containing the id of the reference genome as key and all gene clusters as value (seperated by whitespace). Furthermore, (3) a filtering module named LinearFiltering is provided that performs feature reduction based on cluster size. The feature ranking module (4) provides the option to give certain features (=clusters) a meaningful annotation. Besides this minor features mentioned, basic functionality such as the management of temporary files, logging, useful error messages, a debug option and similar features are provided through the PICA-to-go pipeline.

4.4.1 Implementation of PICA-to-go

Java 8 (openjdk) was used for implementing the PICA-to-go pipeline. Furthermore the following external Java libraries were used Apache Commons Lang3 3.0, Apache Commons IO 2.5, Apache Commons CLI 1.4, Apache Commons Math3 3.6, Google Guava 25, Google Gson 2.8.2, Biojava-ws 4.2.11 and MapDB 3.0.7. As build management tool, Maven 3.5.1 is used. All necessary stand-alone tools (Prodigal, MMSeqs2, PICA) are distributed with PICA-to-go. The integration of these tools is done using the internal ProcessBuilder class in Java. Calling external stand-alone programs is generally avoided when there is another solution (e.g. library) available as this might be a possible source of errors. In a few cases, however, awk and similar standard UNIX tools were called directly in Java as some of these programs provided simple solutions for problems that required a complex instructions in Java instead. The program is started with a run script written in bash that checks if all dependencies that are not directly shipped with the program are available at the system (e.g. Java version 8 or higher, Python 2, basic UNIX tools). Furthermore, it chooses a suitable MMSeqs2 binary as there are two different binaries available depending on the hardware architecture. The user can either chose to download distribution consisting of the bash script, a jar file containing PICA-to-go and all used Java libraries and all needed stand-alone tools (Prodigal, MMSeqs2, PICA) or the user can decide to use a provided Singularity container. As PICA-to-go is free software (GPL 3), the user can also download the source code and build the jar file himself using Maven. The program is available under: https://github.com/FloFlo93/PICA-to-go

4.4.2 Version/changes of used programs

PICA The develop branch of the PICA repository of CUBE was cloned (commit 8a9c8c9). PICA crossvalidate.py was extended with an additional command line option that allows to write the summary statistic to a file. In the class ClassificationResults, the "Probability" was changed to "Prediction Confidence". This name is used as a column name in the results of PICA test.py, describing the confidence of a classification (using Platt scaling). As this is not a probability in a statistical sense, the name was changed. This is consistent with the PhenDB project, which also uses the term "Prediction Confidence" for the result of Platt scaling instead of "Probability". All changes were submitted to the PICA repository owned by CUBE as pull requests and have been accepted.

MMSeqs2 PICA-to-go is shipped with version 2-23394 of MMSeqs2 (released on 5th of March 2018).

Prodigal The version 2.6.3 of Prodigal was used. A static flag was added to the make-file in order to gain a portable binary that is shipped with PICA-to-go. The binary was compiled on Fedora 27 using GCC version 7.3.1.

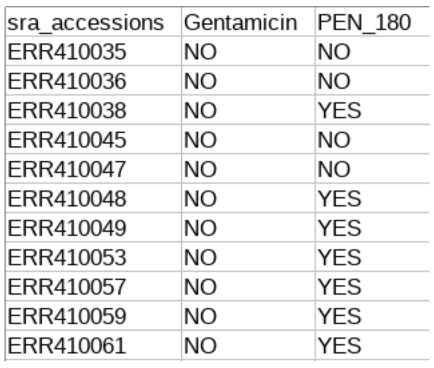
4.4.3 Parameters

Prodigal is called with the default parameters, except if a different translation table (default: 11) is specified in the command-line options of PICA-to-go. PICA train.py and crossvalidate.py are used with the default options. This means that support vector machines are used as a machine learning algorithm using a linear kernel (with a C parameter of 5). All parameters can be changed in the config file, which allows to add parameters to PICA train.py, PICA test.py and PICA crossvalidate.py. This should, however, be used with caution. For MMSeqs2/linclust and MMSeqs2/cluster, the default values for the e-value cutoff (0.001) and the minimal coverage (0.8) are used. For the minimal sequence id, a value of 0.0 is used for both tools (which is the default value for MMSeqs2/cluster, but not for MM-

Seqs2/linclust). Every single cut-off can be changed individually in the config file. As for the PICA tools test, train and crossvalidate, adding additional parameters is also possible for MMSeqs2/cluster and MMSeqs2/linclust. Currently, it is not possible for svmFeatureFanking.py and Prodigal to add additional command-line options. Moreover, the config file allows to set a maximal heap size for Java and allows to add additional arguments to the Java virtual machine. Per default, this is set to 6GB, which was sufficient to train all models (except ARCHAEA). As the real size for the Java process was considerably smaller in the training of the majority of the models, it might be possible to set a lower value.

4.4.4 Provided functionality and inputs/outputs

PICA-to-go train: required parameters For each training, a directory has to be provided containing all bins to be used for the training. All bins must be provided as flat files and in the fasta format. Furthermore, gzipped files (but not compressed directories) are supported. The files can be provided as nucleotides or proteins. PICA-to-go can automatically distinguish between these two file types based on the used alphabet in the file. Mixing both types is also allowed (e.g. provide some protein files and some nucleotide files in the directory). A gene prediction is then performed on all files that are considered to be nucleotide files. Any suffix can be chosen for the files as the processing is independent of the suffix. Unknown characters in each fasta file are automatically converted to 'X' or 'N'. A phenotype file has to be provided for the training. This is a tab separated file. The first line contains the names of the phenotypes to be examined. All the other lines contain an identifier for each bin in the first column and a YES/NO classification in the following columns depending if this bin has a certain phenotype or not.



sra_accessions	Gentamicin	PEN_180
ERR410035	NO	NO
ERR410036	NO	NO
ERR410038	NO	YES
ERR410045	NO	NO
ERR410047	NO	NO
ERR410048	NO	YES
ERR410049	NO	YES
ERR410053	NO	YES
ERR410057	NO	YES
ERR410059	NO	YES
ERR410061	NO	YES

Figure 15: Example of a phenotype file

Furthermore, the name of the phenotype to be trained has to be specified (as a phenotype file might contain information about more than a single phenotype) and an output folder for all results has to be stated.

PICA-to-go train: additional parameters Moreover, the number of threads can be specified (per default number of threads = number of cores) as well as a temporary directory (per default temporary files are written to /tmp/ and deleted after the program terminates). The latter option may be useful for large models and limited RAM, as the /tmp/ folder is often situated in RAM per default and a high amount of files written to /tmp/ may therefore lead to a high RAM usage.

The program furthermore provides a debugging option that prints the stacktrace to stderr in case an exception occurs and it prevents the deletion of the temporary folder as useful information such as log files for each external tool is stored in the temporary folder together with needed flat files itself that are required during different stages of the program and may be corrupted in case of an exception. Per default PICA-to-go uses MMSeqs2/cluster with default settings (cut-offs for clustering: e-value 0.001, coverage:0.8, minimal sequence identity 0.0). Additionally, MMSeqs2/linclust is supported, which does not use cascaded clustering and is therefore faster but less accurate (see Introduction section).

MMSeqs2/linclust is called with the same cut-offs as MMSeqs2/cluster. These options are not identical to the default options of MMSeqs2/linclust (which uses an minimal sequence identity cut-off of 0.9 instead of 0.0; see Introduction section). All these options can be changed in a config file, which also allows to add additional options to each program.

PICA-to-go also allows to specify a translation table for gene prediction (per default 11 is used).

The program offers an additional feature reduction step that is able to provide more accurate models at the cost of a longer runtime (see 4.5).

Moreover, the program allows specifying a list of annotated genomes (default: all genomes are considered to be annotated) for the feature ranking step (see 4.6) or allows performing a remote blast on the NCBI server for a few top features.

PICA-to-go train: output The program outputs a zipped folder named "MODELNAME.picamodel". This zipped folder is not intended to be read by humans, but is needed when a prediction is performed using this model. The same is true for another zipped folder named "database.mmseqs". In the prediction mode the database files are needed to perform a clusterupdate with new sequences (the bins that are to be predicted). Information about the model's accuracy are provided in a json file (pica-crossvalidation-no-filtering.json). If a feature reduction has been performed, an additional file is present stating the "new accuracy" (pica-crossvalidation.json). In this case all output files only refer to the "feature reduced" model, except for the pica-crossvalidation-no-filtering.json file that is included stating the original accuracy without any feature reduction. This is intended to give the user an idea how much the accuracy is improved by the feature reduction step. Furthermore, a json file (gene_clusters.json) is provided that contains information about all gene clusters. The representative sequence of each cluster is used as key, while the value contains all genes of this cluster (using the internal gene names, see 4.4.5). Furthermore two tab separated files are provided containing a feature ranking that corresponds to the model. This provides valuable information which gene clusters are most relevant for the prediction. While "MODELNAME.rank.raw.tsv" is the raw output of the corresponding PICA tool (svmFeatureRanking.py), "MODELNAME.rank.annotated.tsv introduces an additional column that contains an annotation for a sequence in the cluster (see section 4.6).

PICA-to-go predict: required parameters PICA-to-go predict requires the PICA-model as well as the clustering database. Both files are provided as output of the PICA-to-go training ("MODELNAME.picamodel", "database.mmseqs"). Furthermore, the name of the phenotype to predict has to be added. Moreover, a directory containing the bins to predict has to be provided. As for the train mode, the directory can contain nucleotide files as well as protein files.

PICA-to-go predict: additional parameters Similar to PICA-to-go train, it is possible to specify a directory for temporary files, the number of threads or a translation table. The program also consists of a debug option.

PICA-to-go predict output The output consists of a gene_clusters.json file like PICA-to-go train, but includes the genes from the bins to be predicted. Furthermore, the prediction itself is delivered in a tab separated file named outputPica.txt. It states a YES/NO prediction on each bin and includes a confidence value (Platt scaling, see Introduction section) between 0 and 1 for each prediction.

4.4.5 Internal gene names

In the PICA-to-go pipeline each gene has to be uniquely identifiable. Furthermore, it should be possible to track each gene name to a certain bin and to a fasta header easily. The internal naming therefore relies on a combination of the bin name (name of the fasta file in the input directory without the suffices fasta, faa, fna and fa) and of the fasta header. For the fasta header only the identifier is considered (the first part of the header before any whitespace). The bin name and the identifier are combined by ^_.

4.5 Feature reduction

It is well known that a high number of irrelevant features is able to decrease the accuracy of a model, especially if the number of samples is low (Hua et al., 2005). However, determining which gene clusters are relevant or irrelevant for a model is not trivial. The general idea behind the feature reduction step is that small clusters consisting of few genes may be not as relevant as larger cluster. To find an optimal cut-off regarding the cluster size, an exhaustive search is performed. Based on the maximal cluster size, 100 + 1 equal spaced cut-offs are used. For example if the maximal cluster size is 1000, the corresponding cut-offs will be 0,10,20,...,1000. For each cut-off a 5-fold crossvalidation (with 10 replicates) will be performed. The cut-off with the highest mean balanced accuracy is then used. While this kind of feature reduction (here named LinearFiltering) is available for PICA-to-go, it is not done per default, due to the fact that it may drastically increase the runtime of large models and it does not always lead to considerably better results (see result section). As a single crossvalidation already needs 50 PICA trainings to be performed, 100 crossvalidation will need 5000 PICA trainings.

4.6 Annotation of features in the feature ranking

As mentioned before, a file containing the importance of each feature (gene cluster) for the prediction is outputted by PICA-to-go train. The purpose of this file is to make the prediction more comprehensible. Furthermore, it allows identifying genes, associated with a phenotype, that have not been associated with this phenotype before. Each gene cluster is, however, named by its representative sequence in PICA-to-go (e.g. 196164^_WP_006768639.1). This is a disadvantage compared to the existing PICA training pipeline that outputs an eggNOG id, which may be more useful than the name of a representative sequence as the eggNOG database provides a functional annotation for each eggNOG id. To overcome this disadvantage, the user can specify a list of fully annotated genomes in PICA-to-go train. If a gene of a cluster originates from a fully annotated genome that was specified before, the corresponding annotation is used. If no annotated genomes are specified, it is assumed that all genomes are annotated and the fasta header of the representative sequence is used as annotation. In case no annotation is available, there is the option of performing a remote blast on the protein sequence of the representative sequence. This is done by using the NCBI remote service that Bio-Java provides. Per default this is limited to the top ten features, but it is possible to manually change this limit.

4.7 The PICA models

All used models are part of the PhenDB-project and can be downloaded from `phendb.org` (`phendb.org/reports/modeloverview`). The training data consists of the original PICA publication by MacDonald and co-workers (MacDonald and Beiko, 2010); moreover additional data and models were added in a manual process. As the trainings data is updated constantly and all models are retrained with the newest version of this data, the data used in my project (and provided via the link above) may not be identical with the latest models used in PhenDB. The AEROBE, ANAEROBE, SPORE, GRAMNEGATIVE and MOTILE models were recently superseded with trainings data from text mining in PhenDB. This data was not used in PICA-to-go, but the old models. In order make all the results reproducible, all the raw data, including a list of training data used for each model, are available in the official Github Repository of PICA-to-go.

Model name	Model description	training date
alkb	Organism is capable of degrading alkanes	2018/05/08
ars	Organism is capable of arsenic detoxification	2018/05/08
avre	Organism is a plant pathogen (based on AvrE virulence factor)	2018/05/08
baicd	Organism is capable of producing secondary bile acids	2018/05/08
bena	Organism is capable of degrading benzoate via hydroxylation	2018/05/08
bpp	Organism is capable of hydrolyzing phytate	2018/05/08
bsh	Organism is capable of deconjugating glycine and taurine from bile salts	2018/05/08
buk2	Organism is capable of producing butyrate	2018/05/08
chb	Organism is capable of degrading chitine	2018/05/08
cnta	Organism is capable of aromatic ring hydroxylation	2018/05/08
coos	Organism is capable of assimilating CO	2018/05/08
cutc	Organism is capable of producing trimethylamine via choline	2018/05/08
dbfa4	Organism is capable of performing dibenzofuran degradation	2018/05/08
hsdh	Organism is capable of degrading bile acids	2018/05/08
hyda	Organism is capable of producing H2 gas	2018/05/08
naph2	Organism is capable of degrading naphtalene	2018/05/08
nif	Organism is capable of fixing N2	2018/05/08
p450	Organism is capable of degrading fatty acids	2018/05/08
phnx	Organism is capable of hydrolyzing phosphonate	2018/05/08
phod	Organism is capable of recycling organic phosphorus	2018/05/08
psa	Organism is capable of synthesizing capsular polysaccharide	2018/05/08
soxb	Organism is capable of oxidizing thiosulfate	2018/05/08
txta	Organism is a plant pathogen (based on thaxtomin synthesis capability)	2018/05/08
urea	Organism is capable of degrading urea	2018/05/08
xen	Organism is capable of reducing various alpha,beta-unsaturated and nitro compounds	2018/05/08
AEROBE	Organism requires oxygen to grow (obligant aerobe)	2018/04/05
ANAEROBE	Organism requires the absence of oxygen to grow (obligant anaerobe)	2018/04/10
AOB	Organism is part of the clade of ammonia-oxidizing bacteria	2018/04/10
ARCHAEA	Organism is an archaeon	2018/05/14
AUTO	Organism is capable of growth with CO2 as sole carbon source	2018/05/14
FACULTATIVE	Organism is a facultative anaerobe	2018/04/10
GRAMNEGATIVE	Organism is gram negative	2018/04/10
HALO	Organism has a halophilic lifestyle	2018/04/10
METHANOTROPH	Organism is capable of growth with methane as the sole carbon source	2018/04/06
MOTILE	Organism is capable of self-propelled motio	2018/04/10
PEN_180	Organism is resistant to penicillin	2018/05/08
PHOTO	Organism is capable of photon capture for energy acquisition	2018/04/10
PSYCHRO	Organism has a psychrophilic lifestyle	2018/04/10
SPORE	Organism is capable of producing endospores for persistence	2018/04/06
SYMBIONT	Organism has an obligate intracellular lifestyle	2018/05/08
T3SS	Organism expresses a Type III secretion system	2018/04/10
T4SS	Organism expresses a Type IV secretion system	2018/04/10
T6SS	Organism expresses a Type VI secretion system	2018/04/10
THERM	Organism has a thermophilic lifestyle	2018/04/10
THERM_OR_TOLERANT	Organism has a thermophilic lifestyle or is thermotolerant	2018/04/10

Table 11: Description of all used models

4.8 Cluster-Comparison

As specified before, it is necessary to evaluate some selected models in more detail (if there is a major difference in the mean balanced accuracy compared to the existing pipeline). As the main difference between the existing pipeline and PICA-to-go is the usage of a different gene clustering tool, it may be useful to compare the outcomes of the clusterings to the clustering outcome of the existing pipeline (see Introduction section). As cluster comparison methods using pair counting (e.g. Rand Index) are computationally unfeasible considering the size of points and the computational complexity, set matching algorithms are chosen instead. In contrast to other set matching algorithms, the Van Dongen measure (VD) is symmetric (VD(clusterset1, clusterset2) = VD(clusterset2, clusterset1)) (S. Wagner and D. Wagner, 2007). Every cluster of clustering A is compared to every cluster in clustering B. For every cluster in clustering A only the maximal intersection to a cluster in clustering B is considered and summed up. Then the procedure is repeated comparing every cluster of clustering B is compared to every cluster in clustering A. Both values are then summed up and the sum is substracted from 2n with n being the number of all items. So, the VD index returns a value between 0 and 2n (0 = identical clusterings). Normalization of the VD index is described in the literature (Rezaei et al., 2016) dividing the index by 2n, which leads to a value between 0 and 1, however a value of 0 would point to an identical clustering. As this is counterintuitive as a value of 1 indicates an identical clustering in most other indices (e.g. Rand index), the VD index was normalized using to return 1 for an identical clustering. To avoid any confusion, I chose to call this index reverse Normalized Van Dongen measure (rNVD).

$$VD = 2n - \sum_{i=1}^{K} max_{j=1}^{K'} Intersection_{ij} + \sum_{j=1}^{K'} max_{i=1}^{K} Intersection_{ij}$$

$$rNVD = \frac{\sum_{i=1}^{K} max_{j=1}^{K'} Intersection_{ij} + \sum_{j=1}^{K'} max_{i=1}^{K} Intersection_{ij}}{2n}$$

Furthermore the asymmetric Meila-Heckerman index (MH) (Meila, 2007) for both cases (clusterset1 vs. clusterset2 and clusterset2 vs. clusterset1) is additionally used for the comparison of clusterings. The MH is closely related to the rNVD, as the arithmetic mean of the MH for both cases (clusterset1 vs. clusterset2, clusterset2 vs. clusterset1) leads to the rNVD.

$$MH = \frac{\sum_{i=1}^{K} max_{j=1}^{K'} Intersection_{ij}}{n}$$

A tool using the orthogroups.json output of PICA-to-go as input was implemented to calculate the rNVD index as well as the MH index. To get a maximal performance, the tool was implemented in C++ using OpenMP for parallelism, which is typically used in high performance computing. Parallelism using multiple cores was achieved by parallelizing the outermost for loop in the algorithm. Furthermore SIMD parallelization is used for the calculation of the maximum intersection, which allows to perform a single instruction with multiple data in parallel on a single core. This is explicitly requested using OpenMP. Furthermore, the whole program was compiled with gcc using the "-O3" flag, which optimizes the code for speed, including vectorizing instructions and running them using SIMD. The intersection itself is calculated by the C++ instruction std::set_intersection (which uses a sorted vector as datastructure). The tool is licensed under LGPLv3 license and the source code as well as an install script is available under [https://github.com/FloFlo93/rNVD].The program has been tested on CentOS 7.5 and on Fedora 27.

4.9 Statistical evaluation

For each model, a 5-fold crossvalidation with 10 replicates is performed using crossvalidate.py. The balanced accuracy of each model is then compared instead of using the raw accuracy. This is done due to the fact that the raw accuracy can be biased in case the YES/NO labels are not balanced for the model. On the other hand, the balanced accuracy considers both labels to an equal degree. PICA crossvalidate.py further returns the standard deviation of each model. This information is then used to perform significance tests. All significance tests were calculated in Java using the Apache Math3 library. The tTest method of the class TTest was used, performing a two-sample, two-tailed t-test with unequal variances (Welch test). All models with significant differences between different methods (e.g. HMMER/eggNOG vs. MMSeqs2/cluster) were further evaluated in order to gain information about the magnitude of this effect. For this task, Cohen's d was used together with the raw differences between the means.

$$Cohen's \quad d = \frac{\mu_1 - \mu_2}{\sigma}, \qquad \sigma = \sqrt{(\sigma_1^2 + \sigma_2^2)/2}$$

4.10 Raw data

For each model, a MODELNAME.training.txt file is provided, which consists of the taxon id, the assembly id or a similar identifier. Furthermore, for every tested method (existing pipeline using HMMER/eggNOG, MMSeqs2/cluster, MMSeqs2/linclust, MMSeqs2/linclust-msi0), a crossvalidation-file is provided. No database-, picamodel or fasta files of the training data are provided.

References

Baker-Austin, Craig et al. (2006). "Co-selection of antibiotic and metal resistance". In: *Trends in Microbiology* 14.4, pp. 176–182. ISSN: 0966-842X. DOI: 10.1016/j.tim.2006.02.006.

Bastanlar, Yalin and Mustafa Ozuysal (2014). "Introduction to machine learning". In: *Methods Mol. Biol.* 1107, pp. 105–28. ISSN: 1940-6029. DOI: 10.1007/978-1-62703-748-8_7.

Bochner, Barry R. (2008). "Global phenotypic characterization of bacteria". In: *FEMS Microbiol. Rev.* 33.1, pp. 191–205. ISSN: 0168-6445. DOI: 10.1111/j.1574-6976.2008.00149.x.

Bradley, Phelim et al. (2015). "Rapid antibiotic-resistance predictions from genome sequence data for Staphylococcus aureus and Mycobacterium tuberculosis". In: *Nat. Commun.* 6, p. 10063. ISSN: 2041-1723. DOI: 10.1038/ncomms10063.

Breiman, Leo (2001). "Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author)". In: *Statistical Science* 16.3, pp. 199–231. ISSN: 0883-4237. DOI: 10.1214/ss/1009213726.

Brynildsrud, Ola et al. (2016). "Rapid scoring of genes in microbial pan-genome-wide association studies with Scoary". In: *Genome Biol.* 17.1, p. 238. ISSN: 1474-760X. DOI: 10.1186/s13059-016-1108-8.

Chang, Yin-Wen and Chih-Jen Lin (2008). "Feature Ranking Using Linear SVM". In: *Proceedings of the Workshop on the Causation and Prediction Challenge at WCCI 2008*. Ed. by Isabelle Guyon et al. Vol. 3. Proceedings of Machine Learning Research. Hong Kong: PMLR, pp. 53–64. URL: http://proceedings.mlr.press/v3/chang08a.html.

Chen, Feng et al. (2006). "OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups". In: *Nucleic Acids Res.* 34.Database, issue. ISSN: 1362-4962. DOI: 10.1093/nar/gkj123.

Edgar, Robert C. (2010). “Search and clustering orders of magnitude faster than BLAST”. In: *Bioinformatics* 26.19, pp. 2460–2461. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btq461.

Eichinger, Valerie et al. (2016). “EffectiveDB—updates and novel features for a better annotation of bacterial secreted proteins and Type III, IV, VI secretion systems”. In: *Nucleic Acids Res.* 44.D1, pp. D669–D674. ISSN: 0305-1048. DOI: 10.1093/nar/gkv1269.

Emms, David M. and Steven Kelly (2015). “OrthoFinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy”. In: *Genome Biol.* 16.1, p. 157. ISSN: 1474-760X. DOI: 10.1186/s13059-015-0721-2.

Feldbauer, Roman et al. (2015). “Prediction of microbial phenotypes based on comparative genomics”. In: *BMC Bioinformatics* 16.14, S1. ISSN: 1471-2105. DOI: 10.1186/1471-2105-16-S14-S1. URL: https://doi.org/10.1186/1471-2105-16-S14-S1.

Gabaldón, Toni and Eugene V. Koonin (2013). “Functional and evolutionary implications of gene orthology”. In: *Nat. Rev. Genet.* 14.5, pp. 360–366. ISSN: 1471-0056. DOI: 10.1038/nrg3456.

Gerlt, John A. and Patricia C. Babbitt (2000). “Can sequence determine function?” In: *Genome Biol.* 1.5, reviews0005.1–reviews0005.10. ISSN: 1465-6906. DOI: 10.1186/gb-2000-1-5-reviews0005.

Gottesman, Susan (2005). “Micros for microbes: non-coding regulatory RNAs in bacteria”. In: *Trends Genet.* 21.7, pp. 399–404. ISSN: 0168-9525. DOI: 10.1016/j.tig.2005.05.008.

Hauser, Maria, Martin Steinegger, and Johannes Söding (2016). “MMseqs software suite for fast and deep clustering and searching of large protein sequence sets”. In: *Bioinformatics* 32.9, pp. 1323–1330. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw006.

Hua, Jianping et al. (2005). “Optimal number of features as a function of sample size for various classification rules”. In: *Bioinformatics* 21.8, pp. 1509–1515. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bti171.

Huerta-Cepas, Jaime et al. (2016). “eggNOG 4.5: a hierarchical orthology framework with improved functional annotations for eukaryotic, prokaryotic and viral sequences”. In: *Nucleic Acids Res.* 44.D1, pp. 286–293. ISSN: 1362-4962. DOI: 10.1093/nar/gkv1248.

Hyatt, Doug et al. (2010). “Prodigal: prokaryotic gene recognition and translation initiation site identification”. In: *BMC Bioinf.* 11.1, p. 119. ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-119.

Inouye, Michael et al. (2014). “SRST2: Rapid genomic surveillance for public health and hospital microbiology labs”. In: *Genome Med.* 6.11, p. 90. ISSN: 1756-994X. DOI: 10.1186/s13073-014-0090-6.

Jensen, Lars Juhl et al. (2008). “eggNOG: automated construction and annotation of orthologous groups of genes”. In: *Nucleic Acids Res.* 36.suppl$_1$, pp. D250–D254. ISSN: 0305-1048. DOI: 10.1093/nar/gkm796.

Jordan, M. I. and T. M. Mitchell (2015). “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349.6245, pp. 255–260. ISSN: 0036-8075. DOI: 10.1126/science.aaa8415.

Joshi, Trupti and Dong Xu (2007). “Quantitative assessment of relationship between sequence similarity and function similarity”. In: *BMC Genomics* 8, p. 222. ISSN: 1471-2164. DOI: 10.1186/1471-2164-8-222.

Kunin, Victor et al. (2008). “A Bioinformatician’s Guide to Metagenomics”. In: *Microbiol. Mol. Biol. Rev.* 72.4, pp. 557–578. ISSN: 1092-2172. DOI: 10.1128/MMBR.00009-08.

Lees, John A. and Stephen D. Bentley (2016). “Bacterial GWAS: not just gilding the lily”. In: *Nat. Rev. Microbiol.* 14, p. 406. ISSN: 1740-1534. DOI: 10.1038/nrmicro.2016.82.

Li, Li, Christian J. Stoeckert Jr., and David S. Roos (2003). “OrthoMCL: Identification of Ortholog Groups for Eukaryotic Genomes”. In: *Genome Res.* 13.9, pp. 2178–2189. ISSN: 1088-9051. DOI: 10.1101/gr.1224503.

Li, Weizhong and Adam Godzik (2006). “Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences”. In: *Bioinformatics* 22.13, pp. 1658–1659. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btl158.

MacDonald, Norman J. and Robert G. Beiko (2010). “Efficient learning of microbial genotype-phenotype association rules”. In: *Bioinformatics* 26.15, pp. 1834–1840. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btq305.

Marsland, Stephen (2014). *Machine Learning: An Algorithmic Perspective, Second Edition (Chapman & Hall/Crc Machine Learning & Pattern Recognition)*. Taylor & Francis Inc. ISBN: 978-146658328-3. URL: https://www.amazon.de/Machine-Learning-Algorithmic-Perspective-Recognition/dp/1466583282.

McArthur, Andrew G. et al. (2013). “The Comprehensive Antibiotic Resistance Database”. In: *Antimicrob. Agents Chemother.* 57.7, pp. 3348–3357. ISSN: 0066-4804. DOI: 10.1128/AAC.00419-13.

Meila, Marina (2007). “Comparing clusterings—an information based distance”. In: *Journal of Multivariate Analysis* 98.5, pp. 873–895. ISSN: 0047-259X. DOI: 10.1016/j.jmva.2006.11.013.

Mueller, Andreas C. and Sarah Guido (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O’Reilly Media. URL: https://www.amazon.de/Introduction-Machine-Learning-Python-Scientists-ebook/dp/B01M0LNE8C.

Pagano, R. R. (2012). *Understanding Statistics in the Behavioral Sciences*. Cengage Learning. ISBN: 978-113371393-7. URL: https://books.google.at/books?id=8UsKAAAAQBAJ&printsec=frontcover&dq=descriptive+statistics+inferential+statistics&hl=de&sa=X&ved=0ahUKEwij5uu0n5_dAhURLFAKHZWaB5kQ6AEIRTAE#v=onepage&q=descriptive%20statistics%20inferential%20statistics&f=false.

Platt, John C. (2000). “Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods”. In: *Adv. Large Margin Classif* 10.3. URL: https://www.researchgate.net/publication/2594015_Probabilistic_Outputs_for_Support_Vector_Machines_and_Comparisons_to_Regularized_Likelihood_Methods.

Plomin, Robert, Claire M. A. Haworth, and Oliver S. P. Davis (2009). “Common disorders are quantitative traits”. In: *Nat. Rev. Genet.* 10.12, p. 872. ISSN: 1471-0064. DOI: 10.1038/nrg2670.

Read, Timothy D. and Ruth C. Massey (2014). “Characterizing the genetic basis of bacterial phenotypes using genome-wide association studies: a new direction for bacteriology”. In: *Genome Med.* 6.11, p. 109. ISSN: 1756-994X. DOI: 10.1186/s13073-014-0109-z.

Rezaei, Mohammad et al. (2016). “Set Matching Measures for External Cluster Validity”. In: *IEEE Trans. Knowl. Data Eng.* 28.8, pp. 2173–2186. ISSN: 1041-4347. DOI: 10.1109/TKDE.2016.2551240.

Rotthauwe, J. H., K. P. Witzel, and W. Liesack (1997). “The ammonia monooxygenase structural gene amoA as a functional marker: molecular fine-scale analysis of natural ammonia-oxidizing populations.” In: *Appl. Environ. Microbiol.* 63.12, pp. 4704–4712. ISSN: 0099-2240. URL: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC168793.

Schloss, Patrick D. and Jo Handelsman (2005). “Metagenomics for studying unculturable microorganisms: cutting the Gordian knot”. In: *Genome Biol.* 6.8, p. 229. ISSN: 1474-760X. DOI: 10.1186/gb-2005-6-8-229.

Schürch, Anita C. and Willem van Schaik (2017). “Challenges and opportunities for whole-genome sequencing-based surveillance of antibiotic resistance”. In: *Ann. N. Y. Acad. Sci.* 1388.1, pp. 108–120. ISSN: 1749-6632. DOI: 10.1111/nyas.13310.

Steinegger, Martin and Johannes Söding (2017). “MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets”. In: *Nat. Biotechnol.* 35, p. 1026. ISSN: 1546-1696. DOI: 10.1038/nbt.3988.

Steinegger, Martin and Johannes Söding (2018). “Clustering huge protein sequence sets in linear time”. In: *Nat. Commun.* 9.1, p. 2542. ISSN: 2041-1723. DOI: 10.1038/s41467-018-04964-5.

Studer, Romain A. and Marc Robinson-Rechavi (2009). “How confident can we be that orthologs are similar, but paralogs differ?” In: *Trends Genet.* 25.5, pp. 210–216. ISSN: 0168-9525. DOI: 10.1016/j.tig.2009.03.004.

Tatusov, Roman L. et al. (2000). “The COG database: a tool for genome-scale analysis of protein functions and evolution”. In: *Nucleic Acids Res.* 28.1, pp. 33–36. ISSN: 0305-1048. DOI: 10.1093/nar/28.1.33.

Wagner, Silke and Dorothea Wagner (2007). *Comparing Clusterings - An Overview.* [Online; accessed 26. Aug. 2018]. URL: https://publikationen.bibliothek.kit.edu/1000011477.

Weimann, Aaron et al. (2016). “From Genomes to Phenotypes: Traitar, the Microbial Trait Analyzer”. In: *mSystems* 1.6, pp. 00101–16. ISSN: 2379-5077. DOI: 10.1128/mSystems.00101-16.

Wen, Zeyi et al. (2018). “ThunderSVM: A Fast SVM Library on GPUs and CPUs”. In: *Journal of Machine Learning Research* 19, pp. 1–5.

Wheeler, Travis J. and Sean R. Eddy (2013). “nhmmer: DNA homology search with profile HMMs”. In: *Bioinformatics* 29.19, pp. 2487–2489. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btt403.

Wood, Derrick E. and Steven L. Salzberg (2014). “Kraken: ultrafast metagenomic sequence classification using exact alignments”. In: *Genome Biol.* 15.3, R46. ISSN: 1465-6906. DOI: 10.1186/gb-2014-15-3-r46.

Zallot, Remi et al. (2016). “Functional Annotations of Paralogs: A Blessing and a Curse”. In: *Life (Basel).* 6.3, p. 39. ISSN: 2075-1729. DOI: 10.3390/life6030039.

Zankari, Ea et al. (2012). “Identification of acquired antimicrobial resistance genes”. In: *J. Antimicrob. Chemother.* 67.11, pp. 2640–2644. ISSN: 0305-7453. DOI: 10.1093/jac/dks261.