



universität
wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„The prediction of functional bacterial secretion systems
based on comparative genomics“

verfasst von / submitted by

Valerie Eichinger, BSc

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

Master of Science (MSc)

Wien, 2016 / Vienna 2016

Studienkennzahl lt. Studienblatt /
degree programme code as it appears on
the student record sheet:

A 066 910

Studienrichtung lt. Studienblatt /
degree programme as it appears on
the student record sheet:

Computational Science

Betreut von / Supervisor:

Univ.-Prof. Mag. Dr. Thomas Rattei

Abstract

Bacterial secretion systems, as crucial virulence factors, can be of major interest, when analysing new genomic or metagenomic data. As the amount of generated genomic data increases heavily, a sophisticated way of analysing (meta-) genomic sequences is of high importance. Thus, we have to find some way of a “high-throughput-analysis” for genomic data to gain information out of the cavalcade of provided data. As there is currently a lack of secretion-related genome annotation, secondary databases, as the one presented in this work, have to provide sophisticated tools and carefully curated data. Thus, we designed an approach, based on machine learning techniques. It is a computational method that uses binary classification to make a clear decision, whether there is a functional secretion system or not.

Therefore we used the PICA framework, based on a support vector machine, which only uses the sequence information, to train one model each for the types III, IV and VI of the bacterial secretion systems. As the major questions of our work, we had to find on the one hand, a sufficient number of already verified secretion system occurrences, to train our models correctly and on the other hand, we had to look for a sophisticated genotype representation. Within our approach, the genomic information is represented by clusters of orthologous groups (COGs). Calculating the models in such way, gives us the opportunity to understand which COGs contribute mostly to the classification as well as, which (unexpected) organisms show the specific trait.

To provide not only the created models for the secretion system prediction, but a sophisticated and comprehensive web tool, we embedded our approach under the name EffectiveS346 within the database EffectiveDB (<http://effectivedb.org>). Additionally, we achieved several adaptations of the already existing effective website. There was an update of the training datasets of the effector predicting applications EffectiveT3 and EffectiveELD. Furthermore, the new type III effector prediction method EffectiveCCBD, based on the detection of a conserved 1B chaperone binding domain was implemented. To expand the databases scope to type IV secreted effectors, we also integrated the existing web tool T4SEpre. Finally, to complete a coherent and extensive analysis of secretion specific information, we added the Predotar web tool, which indicates possible target sites of the effectors within the eukaryotic host. All of these predictions were applied on 1677 genomes, available from the EggNOG 4.0 database, on beforehand and their results are stored and provided to the users.

Although our predictions are not free of all inconsistencies, just like taxonomic bias or partly imbalanced training datasets, we are able to predict the occurrence of bacterial secretion systems in more than 90% (standard deviation below 4.5%) of the cases.

With our tool, users will be able to gain extensive information about intact secretion systems and putative secreted proteins of more than 1677 genomes on the one hand and secondly be able to interpret the prediction results of their own sequence data, easily.

Kurzfassung

Bakterielle Sekretionssysteme sind wichtige Virulenzfaktoren und als solche von großem Interesse bei der Analyse von neuen Genom- oder Metagenomdaten. Da die Menge an produzierten Genomdaten stetig zunimmt, ist es von großer Wichtigkeit, eine zielgerichtete und intelligente „Hochdurchsatzmethode“ zu finden um mit der Flut an Daten umzugehen und daraus die nötige Information zu generieren. Nachdem es momentan keine Bestrebungen oder Richtlinien zur sekretionsbezogenen Genomannotation gibt, ist es umso wichtiger, dass Datenbanken, wie die Vorgestellte, ausgefeilte Werkzeuge und sorgfältig kuratierte Daten zur Verfügung stellen. Mit unserem Vorhersageansatz, basierend auf dem Konzept des maschinellen Lernens, haben wir eine computerbasierte Methode gefunden, die, mittels binärer Klassifikation, eine eindeutige Entscheidung trifft, ob ein intaktes Sekretionssystem vorliegt, oder nicht.

Zur Umsetzung nutzten wir das PICA framework, basierend auf einer support-vector-machine, welche ausschließlich die Sequenzinformation verwendet, um je ein Modell für die Typen III, IV und VI der bakteriellen Sekretionssysteme zu entwickeln. Dabei stellten sich, als größte Fragen im Modellierungsprozess, auf der einen Seite die Suche nach einer ausreichenden Anzahl an belegten Vorkommen von Sekretionssystemen und andererseits, die sinnvolle Repräsentation des Genotyps, dar. In unserem Ansatz entschieden wir uns für die Darstellung des Genotyps, in Form von „clusters of orthologous groups“ (COGs). Diese Umsetzung ermöglichte es uns herauszufinden, welche COGs eine maßgebliche Rolle bei der Klassifikation spielten, beziehungsweise, welche, unter Umständen unerwarteten, Organismen den gesuchten Phänotyp aufweisen.

Um nicht nur die besagten Vorhersagemodelle für Sekretionssysteme zu veröffentlichen, sondern außerdem auch ein umfangreiches Webtool zur Verfügung zu stellen, wurden die Vorhersagemodelle unter dem Anwendungsnamen EffectiveS346 in die Datenbank EffectiveDB (<http://effectivedb.org>) eingebettet. Zusätzlich wurden einige Adaptionen an der bereits existierenden effective-Website durchgeführt. Es wurden die Trainingsdatensätze der Effektorvorhersagemethoden EffectiveT3 und EffectiveELD upgedatet. Außerdem haben wir die neue Typ III-Effektorvorhersage EffectiveCCBD implementiert. Diese basiert primär auf der Erkennung einer hoch-konservierten Bindestelle für 1B Chaperone. Um das Spektrum der Datenbank auch in Bezug auf Typ IV sezernierte Proteine zu erweitern, wurde das bereits verfügbare Werkzeug T4SEpre ebenso integriert. Letztlich, um die umfassende Analyse sekretionsspezifischer Daten abzurunden, wurde die Predotar-Anwendung, welche mögliche Zielorganellen der Effektoren in der eukaryotischen Wirtszelle angibt, eingefügt. Jede der erwähnten Vorhersagemethoden wurde vorab an 1677, via EggNOG 4.0 verfügbaren Genomen angewandt und die Vorhersageresultate dem User zur Verfügung gestellt.

Obwohl unsere Vorhersagen nicht gänzlich frei von Inkonsistenzen, wie taxonomischer Fehlinterpretationen oder nicht ganz ausgewogenen Trainingsdatensätzen, sind, ist es uns möglich, funktionelle bakterielle Sekretionssysteme mit einer Genauigkeit von über 90% (Standardabweichung unter 4.5%) vorherzusagen.

Unsere Datenbank, ermöglicht es dem User, umfangreiche Informationen über funktionelle Sekretionssysteme, sowie mögliche sezernierte Proteine, für mehr al 1677 Genome zu erhalten. Außerdem hat er die Möglichkeit der Vorhersage und einfachen Interpretation seiner eigenen Sequenzdaten.

Content

Abstract	3
Kurzfassung	5
Content.....	7
List of Figures.....	9
List of Tables.....	11
1 Introduction.....	13
1.1 Host-pathogen interaction	13
1.2 Properties of the distinct secretion systems	13
1.2.1 Structure proteins	14
1.2.2 Type III secretion system	16
1.2.3 Type IV secretion system.....	17
1.2.4 Type VI secretion system.....	17
1.2.5 Secreted proteins – “effectors”	18
1.3 Sources and Methods for the prediction of bacterial secretion systems	18
1.3.1 Metagenomics	19
1.3.2 Data mining and machine learning methods	19
1.4 Genome based prediction of phenotypes.....	20
1.4.1 Support vector machine	20
1.4.2 Cross-validation	22
1.4.3 Genotype abstractions	22
1.5 Scientific relevance.....	23
2 Methods	27
2.1 Model design	27
2.1.1 Training.....	27
2.1.2 Model application.....	30
2.2 Application implementation.....	30
3 Results and Discussion	37

3.1	PICA models.....	37
3.1.1	Finding sufficient Training Data.....	37
3.1.2	Genotype representation	39
3.1.3	Final models.....	46
3.1.4	Limitations and advantages of using a support vector machine for phenotype prediction 47	
3.2	EffectiveDB	50
3.2.1	Database design	51
3.2.2	Workflow	55
3.2.3	Database content description	61
3.3	Comparison to existing prediction methods	69
3.3.1	SecReT4 and SecReT6.....	69
3.3.2	T346Hunter.....	71
3.4	Pipeline application on metagenomic data from sponges.....	72
4	Conclusion and Outlook	75
	Bibliography.....	77
	Supplement	83
	Training datasets	83
	Top 20 ranks SVM rank files	94
	Genotype profile with PFAM architectures.....	98
	Shell-script EffectiveDB pipeline	99
	Recommended genome completeness.....	109

List of Figures

Figure 1 Components of the bacterial secretion systems type III, IV and VI.	14
Figure 2 Three different machine learning techniques.	20
Figure 3 Concept of a support vector machine.	21
Figure 4 Theoretical approach abstraction.	22
Figure 5 Concept and workflow of the PICA approach.	27
Figure 6 Taxonomy of positive dataset run 16_97_200.	40
Figure 7 COG occurrence concerning T6SS.	47
Figure 8 The new EffectiveDB Logo.	53
Figure 9 Homepage of EffectiveDB.	54
Figure 10 EffectiveDB workflow.	55
Figure 11 Main input.	59
Figure 12 Enabling/Disabling.	59
Figure 13 Choosing the genome mode.	60
Figure 14 Identified bacterial secretion systems.	61
Figure 15 Comparison of type III effector prediction methods.	62
Figure 16 Overview Escherichia coli O157:H7 str. EDL933.	63
Figure 17 Input parameters Escherichia coli O157:H7 str. EDL933.	64
Figure 18 Summary Escherichia coli O157:H7 str. EDL933.	64
Figure 19 Protein based results Escherichia coli O157:H7 str. EDL933.	65
Figure 20 Genome based results Escherichia coli O157:H7 str. EDL933.	66
Figure 21 Summary Burkholderia pseudomallei K96243.	67
Figure 22 Genome based results Burkholderia pseudomallei K96243.	67
Figure 23 Genome repository T346Hunter.	71

Figures without any mentioned references within their captions are made by myself.

List of Tables

Table 1 Structure proteins of the bacterial secretion systems type III, IV and VI.	15
Table 2 Stepwise pipeline description.	31
Table 3 Comparison of expected vs. observed COGs of T3SS.	42
Table 4 Comparison of expected vs. observed PFAMs of T3SS.	43
Table 5 Comparison of expected vs. observed PFAM-architectures of T3SS.	44
Table 6 Results feature selection for optimised models.	45
Table 7 Overview of the three finally used, optimised models of the PICA approach.	46
Table 8 True-negatives containing models.	48
Table 9 Comparison of the type three dependent results of run 15_51_100 and 17_51_102.	49
Table 10 Use cases.	51
Table 11 Required pipeline preconditions.	55
Table 12 Pipeline time estimation.	58
Table 13 Classifying COGs Burkholderia pseudomallei K96243.	67
Table 14 Genome repositories SecReT.	70
Table 15 Results of EffectiveS346 and EffectiveELD on 18 metagenomic bins.	74

1 Introduction

This chapter should give the reader a brief overview of the scientific area of bacterial secretion in general and insights into the usage of computational methods in regard to phenotype prediction.

There will be explained why we think that our work relevantly contributes to the current state of scientific research. We raise the question of using machine learning techniques for phenotype prediction and try to introduce the reader to our approach of giving the answer.

1.1 Host-pathogen interaction

Bacterial secretion is a key virulence factor of pathogenicity in general. Especially, the interaction between a bacterium and its host is closely related to this attribute (1). Thus, one needs to understand how the manifold protein apparatuses work in interaction with both, the bacterium and the eukaryotic host. Additionally, the secreted proteins itself can give insights into the processes of secretion.

Within the group of gram-negative bacteria there are, up to now, eight different types of secretion system plus the chaperone-usher pathway, known (2, 3). Although those secretion systems are quite well conserved among different bacterial genera, they are difficult to identify due to their diversity and genetic modularity. Moreover it is getting more and more obvious that our knowledge is quite limited in this field, yet (3). Nevertheless, we have to find a way of extracting the essential information of our current knowledge to make a meaningful phenotype prediction of the different types of secretion systems. Therefore, it is of crucial importance to understand as many facts and attributes of the distinct secretion systems as possible, to consider them for our prediction-models.

Whereas the types II, V and VIII, plus the already mentioned usher pathway, always depend on the presence of the so called “Sec-protein complex” or the “twine arginine transporter”, to transfer substances across the inner cell membrane, all the others can act without those transporters (2), ((4):277). In other words, the type I, III, IV, VI and VII secretion systems have the ability to cross both bacterial cell membranes (or the cell membrane plus the mycomembrane, respectively) within a single step to export effector proteins or in some cases even DNA, for example (5). Within this group of secreting protein complexes, our work focusses specifically on the types III, IV and VI.

1.2 Properties of the distinct secretion systems

The three already mentioned types III, IV and VI, commonly share the ability to not only bridge the inner and outer membrane of the microbe within a single step. They are also able to inject the secreted substance directly into the cytosol of the host organism (1) (Figure 1).

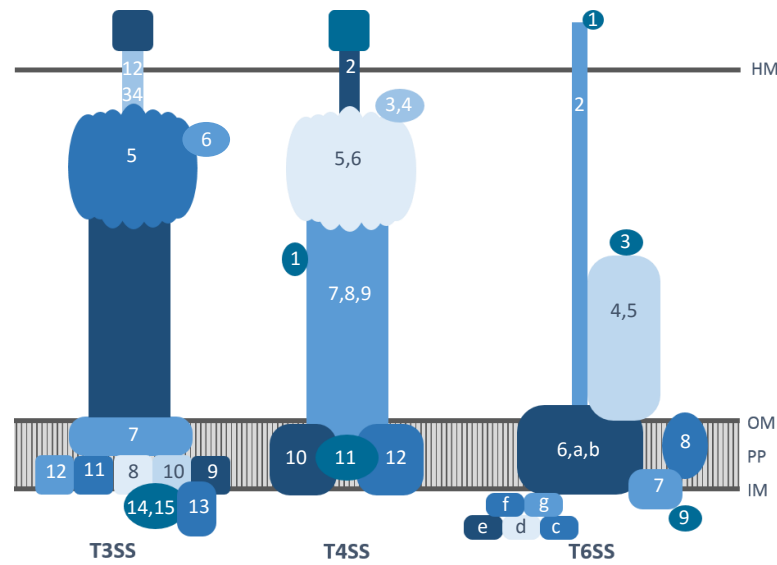


Figure 1 Components of the bacterial secretion systems type III, IV and VI. The three types of bacterial secretion systems, our work is focusing on, are able to not only span the inner membrane (IM), the periplasm (PP) and the outer membrane (OM) of the bacterium, but also the host membrane (HM) to inject putative effectors directly into the hosts cytosol. The numbers and letters, shown in the figure, correspond to the compound descriptions in Table 1.

Beside this connecting feature, there are major differences between those three systems. It is crucial to understand the features and abilities that characterise each system individually, to be able to create an analytical approach as ours.

1.2.1 Structure proteins

It seems to be obvious that only those secretion systems, which contain the most important structure giving proteins, can have the ability of a functional effector secretion. Therefore, we had a look at the structure proteins suggested by KEGG (6). The most of them could be “translated” into the format of our genotypic representation called COGs (1.4.3 Genotype abstractions). We would expect those proteins, or classifying features, within our predictions. Table 1 shows a list of the KEGG structure proteins for each of the three systems and the assigned COG number and description. The numbers in the second column are also mentioned in the above introduced Figure 1. Although, those presented proteins are of crucial importance, they are not the only classifying factor. Within our approach it is necessary to find features within a phenotypic trait that are very typical and unique. The DotU protein, for example, is a factor of both, the type IV and the type VI secretion system. This makes it hard to distinguish between those two types just by the presence or absence of proteins like DotU. Thus, we are also dependent on further features like effectors, regulatory factors or other associated proteins. Regarding the T6SS, I want to explicitly mention the presence of the so called “putative cytoplasmic components”. Although, they actually do not belong to the structure of the protein complex itself, they are very much associated with it (7). Within the presented Table 1 and Figure 1 they are marked by the

letters “a” to “g” to clearly highlight them, compared to the KEGG structure proteins. Because of the strong interaction of those proteins with the secretion process, this group of proteins turned out to be very important for our classification. Finally, one can say that finding the structure proteins of the distinct systems within the group of classifying features, gives a strong indication of the correctly found trait (=secretion system). On the other hand, it does not need to be problematic, if the list of features lacks the structure proteins in the top ranks, as they are not necessarily the most important factors for the correct classification.

Table 1 Structure proteins of the bacterial secretion systems type III, IV and VI. This table shows the different structure proteins of the three secretion system apparatuses and their corresponding COGs and COG-descriptions. We would expect to find those within our predictions. Having an in depth look at table, we can say that if we find those type specific and typical proteins (or more precise: their assigned COGs) within the list of features that contribute to our classifying decision, we have a serious indication that the trait was correctly captured.

	#	Protein	COG/NOG	Description
T3SS	1	YscF	COG0659	Sulfate transporter
	2	YscO	-	-
	3	YscP	-	-
	4	YscX	-	-
	5	YscC	COG1450	General secretion pathway protein D
	6	YscW	-	-
	7	YscJ	COG4669	Type iii secretion
	8	YscR	COG4790	Type III secretion system protein
	9	YscS	COG4794	Protein secretion
	10	YscT	COG4791	Type iii secretion
	11	YscU	COG4792	Type iii secretion
	12	YscV	COG4789	Type III secretion
	13	YscN	COG1157	Flagellum-specific ATP synthase
	14	YscQ	COG1157	Flagellum-specific ATP synthase
	15	YscL	COG1317	Flagellar assembly protein
T4SS	1	VirB1	-	-
	2	VirB2	COG3838	Conjugal transfer protein, TrbC
	3	VirB3	COG3702	Type IV secretion system protein
	4	VirB5	COG3701	Conjugal transfer protein
	5	VirB7	-	-
	6	VirB9	COG3504	Transfer protein TrbG
	7	VirB6	COG3704	Type IV secretion system protein
	8	VirB8	COG3736	Type IV secretion system protein VirB8

T6SS	9	VirB10	COG2948	Conjugation trbi family protein
	10	VirB4	COG3451	Type IV secretion system protein
	11	VirB11	COG0630	Type II secretion system protein
	12	VirD4	COG3505	Trag family
	1	VgrG	COG3501	Rhs element vgr protein
	2	Hcp	COG3157	Type VI secretion system effector, hcp1 family
	3	Lip	COG3521	Type VI secretion
	4	IcmF	COG3523	Type VI secretion protein (IcmF)
	5	DotU	COG3455	Type IV VI secretion system protein, DotU family
	6	ClpV	COG0542	ATP-dependent CLP protease, ATP-binding subunit
	7	Fha1	COG3456	Fha domain containing protein
	8	PpkA	COG0515	Serine Threonin protein kinase
	9	PppA	COG0631	Phosphatase
	a	TssB	COG3516	Type VI secretion protein, VC_A0107 family
	b	TssC	COG3517	Type VI secretion protein, EvpB VC_A0114 family
	c	TssK	COG3522	Type VI secretion protein, VC_A0114 family
	d	TssG	COG3520	Type VI secretion protein, VC_A0111 family
	e	TssF	COG3519	Type VI secretion protein,
	f	TssA	COG3515	Type VI secretion-associated protein,
	g	TssE	COG3518	Type VI secretion system, lysozyme-related protein

1.2.2 Type III secretion system

Bacteria which show the very frequently observed type III secretion system (T3SS) can be found among the group of gram-negatives. They can interact with their eukaryotic hosts, plants as well as animals, as pathogens or as mutualists (8).

Originally, the so called “injectisome” is homologous to the flagellum, which also acts as a kind of T3SS (9). Difficulties occur, in regard to the precise identification of the non-flagellar T3SS apart from the flagellum. Furthermore, it is getting more complicated, because the type III injectisome requires up to 25 different proteins, of which only nine are highly conserved. Eight out of those nine are also conserved within the flagellar apparatus (8). It appears to be obvious that a clear distinction between those two complexes is very difficult. Structurally, the needle complex is organised in different partitions (base, needle, inner rod, inner membrane export apparatus, cytosolic components, needle tip and the translocon) which are assembled in a gradual manner (10). Especially the so called “cytosolic components” might be of interest in connection to our approach. Although their

organisation and their interaction is only poorly understood, the genotypic information is highly conserved across many different T3SS (10). As our approach is not limited to structural proteins, those components might provide crucial information and therefore, essentially contribute to the decision of classification.

1.2.3 Type IV secretion system

Whereas T3SS are closely related to the flagellum (9), type IV secretion systems (T4SS) are assembled from core components of the conjugational machine (11). Moreover, it can be said that this type, among the other bacterial secretion systems, is the most variable and versatile one, regarding its interaction partners or putatively secreted proteins (12). It can be further categorised into three different subfamilies:

1. Conjugational system

It is the major subgroup of T4SS and can be found within both, the gram-positive and gram-negative bacteria as well (12) and even within some archaea (13). As it is responsible for the transfer of DNA-protein substrates into eukaryotes like fungi, plants or even human cells (14), it is a crucial factor for drug resistance (15). Other researches have shown that the conjugational system also has an influence on biofilm formation (16).

2. DNA uptake and release system

This subfamily also provides DNA exchange, which contributes to genetic plasticity, as the first one, mentioned above. Opposed to the conjugational system, it is independent of direct cell contact (12).

3. Effector translocation system

The third family seems to have the biggest impact on pathogenicity as it has the ability to directly inject potential pathogenic proteins into plant or mammalian host cells (12) via an injectisome, which is similar to the already mentioned T3SS.

1.2.4 Type VI secretion system

The secretion systems, we already dealt with above, all evolved from non-virulent bacterial compartments. Whereas type III and IV derived from flagellum and conjugational apparatus, respectively, the type VI secretion system (T6SS) seem to be evolutionarily related to the phage injection machinery (17). The gene clusters for T6SS contain between 12 and more than 20 genes and there can be more than one copy of a cluster (18).

Opposed to the T4SS, only gram-negative bacteria seem to have the ability to express a T6SS (19). Furthermore, this type of secretion system does not only show virulent abilities. It is suggested by other studies that type VI secretion is also required by plants for efficient root colonisation and nitrogen fixation, for example (20).

1.2.5 Secreted proteins – “effectors”

Effectors are proteins, which are transported out of the bacterial cytosol into the cell environment or even directly into the host, mediated by bacterial secretion systems (21). As the name already indicates, they can cause manifold effects within a host, such as a change of the immune response, for example (22). Although it is quite difficult to characterise these proteins, due to their high diversity, it is possible to detect them, if secreted by a T3SS, by searching for a signal peptide within their N-terminal region (23).

Type III secreted effectors can also show a specific class 1B chaperone binding site within their N-terminal 70 amino acids. Such 1B chaperones mediate the unfolding of putatively secreted proteins, to enable their transport through the narrow type III injectisome and therefore, are of crucial importance for a successful secretion (24).

Opposed to type III dependent secreted proteins, we also had a look at type IV assigned effectors. Although, there has been found quite a number of type IV effectors experimentally, it is difficult to identify them because most of them are spread over the genome, rather than being clustered in a specific genomic region. Nevertheless, the combination of the information extracted from the sequence of the C-terminal region, the position and composition of amino-acids and the motifs and structure, seems to make it possible to identify type IV secreted effectors (25).

Another interesting possibility of identifying effectors, is the secretion system type independent approach of eukaryotic-like domains. In that case, effectors might show the presence of protein domains which are usually found within eukaryotes (e.g.(26)). Those protein domains show a higher frequency within bacteria that are host-associated, compared to those which do not interact with a eukaryote and thus, seem to be an indication of functional secretion. As every protein domain, which is found as well within eukaryotes as within pathogens and at the same time is absent in non-pathogens is a potential “eukaryotic-like domain”, it is quite difficult to identify them. Computational similarity-based approaches just as EffectiveELD (3.2.2 Workflow) therefore, are of crucial importance(23).

1.3 Sources and Methods for the prediction of bacterial secretion systems

By asking if there is a scientific relevance of our work, we have to find interesting biological sources for our data. Thus the field of metagenomics offers us an interesting source of genome data. Facing the large amount of data, generated by metagenomic approaches, we have to use data mining techniques to gain new information.

1.3.1 Metagenomics

Beside the host-pathogen interaction, another interesting application area is the field of metagenomics. This topic firstly came up in 1998 and describes a function based and culture independent way of analysing biological samples (27). It was a ground-breaking new approach, as currently, only less than one percent of the microbial biodiversity can be cultured (28). Finding a way of bypassing the culturing step within sample processing, opened the way for understanding the interaction of different taxa with each other and their environment. Considering the knowledge of metagenomics for almost 20 years, might raise the question why our approach should be of any scientific interest or significance, yet. Answering this, I want to point out the major achievements in bioinformatics over the last years. Nowadays it is possible to extract almost complete genomic bins from a metagenomic sample, due to improvements in genome sequencing, assembling and annotation (29, 30). Consequently, the extreme wide range of possible habitats and the huge amount of genomic data makes it difficult to extract meaningful information about functionality, attributes or lifestyle, just by using the sequence information (30). It seems to be obvious that this gain of information cannot be achieved by analysing the data manually. In fact, there is an inevitable need of computational methods!

1.3.2 Data mining and machine learning methods

During the last few decades of history, people got overwhelmed with data and there is no end in sight. As computers getting more powerful and memory capacities become larger and larger, we gained the ability to produce practically never ending data streams and save everything. But simply accumulating and saving data does not imply any gain of information. Especially, the field of genetics, or more specific “genomics”, emerged through the last years to a scientific area with great achievements in DNA-sequencing for example. What appears to be crucial to handle such comprehensive data, is a sophisticated pattern search, we call “data mining” ((31):5). It is not possible any longer to simply look at some data to find out what they might tell us. We rely on the calculative power of computers to discover those patterns within the data.

“Learning”, on the other hand, is a term which is a little bit difficult to explain in regard to computers. Machine learning in our comprehension is nothing else but *“techniques for finding and describing structural patterns in data, as a tool for helping to explain the data and make predictions from it”* (28:8). It means that the algorithm does not give a clear definition of the specific trait one is interested in, on beforehand. The machine learning algorithm learns by examples to find its own rules and definitions through “experience”.

To render the term of machine learning more precisely, we want to state the following three techniques of machine learning in a nutshell (Figure 2):

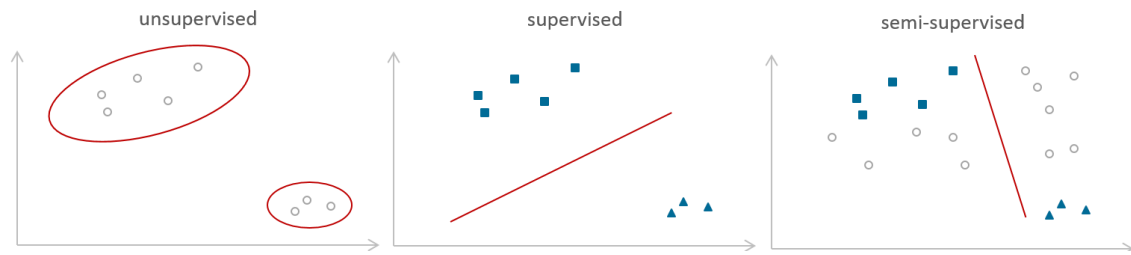


Figure 2 **Three different machine learning techniques.** The unsupervised learning does not require any labelling information to solve clustering problems. In supervised learning, we need the a priori classification of samples, to classify unknown samples. Semi-supervised learning kind of extends the supervised learning. Therefore, it additionally contains unlabeled data.

Unsupervised learning

Unsupervised learning is practically used to handle typical clustering problems. It means that one does not know any labeling of the data, but can group it by the parameters of their occurrence. ((32):3)

Supervised learning

In supervised learning, each training sample is labeled explicitly to a certain class. Most supervised learning problems follow the concept of classification. It is their aim to learn certain rules from the classification of the labeled instances and afterwards precisely assign a new, unlabeled instance to a specific class (29:179). The rules are generated by summarising and comparing the information beyond the attributes of the known samples. ((32):3)

Semi-supervised learning

Semi-supervised learning is a kind of abstraction of the supervised learning technique. For such classification problems you know the labels of some of your data and additionally you also have unlabeled data. As it can be seen in Figure 2, this might lead to another separation border for classification as if supervised learning is used. ((31):294)

1.4 Genome based prediction of phenotypes

Supervised learning is exactly what our approach does, in regard to bacterial secretion systems. Let us face the particular problem, very briefly, by introducing the terms “support vector machine”, “cross-validation” and “clusters of orthologous groups”.

1.4.1 Support vector machine

The support vector machine (SVM) is one possible application of the concept of supervised learning. The classification by using an SVM requires samples, which we call “instances” (=strains of bacterial species) that are unambiguously labeled, which means they belong to exactly one of two possible classes (at least in regard to binary classification). In our particular example it means that they show

the specific trait or phenotype (=bacterial secretion system) or they lack it. Every instance is represented by a number of features (=COGs). Knowing the features of the labeled instances, allows the SVM to learn certain rules for the correct classification of new unlabeled instances. To cut a long story short, one can say that a sufficient number of correctly labeled data allows the SVM to learn how to classify unlabeled data. ((32):326)

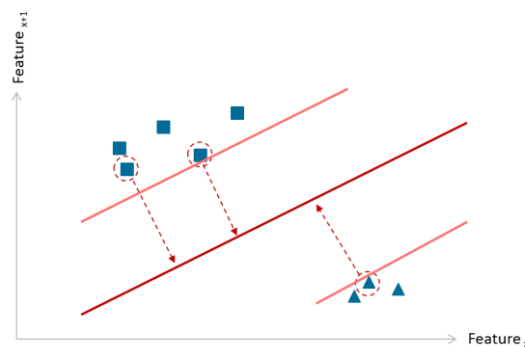


Figure 3 Concept of a support vector machine. It demonstrates the way of how a support vector machine comes to a classifying decision. The dark-red line is the dividing hyperplane that lies within a margin, as brought as possible. The instances, lying closest to the hyperplane are the so called “support vectors”.

Figure 3 shows a theoretical abstraction of the principle of operation of an SVM. It should be noticed that each sample is an instance (blue triangle or square) but each feature represents one dimension. In other words, this approach is highly dimensional in its computational effort. The depicted hyper plane is linear, because of the linearly chosen kernel of our SVM. Although it is not mandatory to use a linear kernel with our approach ((33):45ff), it offers us a very helpful opportunity: The arrows in Figure 3, pointing from the most critical, red highlighted instances, we call “support vectors”, to the dividing hyper plane, indicate the position of the instance within the feature space. As already pointed out, each sample is composed of features that are dimensions. By using a linear Kernel, we now have the opportunity to calculate back to the feature with the highest influence, quite easily. Therefore, we use the distance of each sample to the hyper plane, in every dimension. At this point, we created the so called SVM rank file (3.1.2 Genotype representation), which is a crucial output of our approach. It helps us to understand, which of the available features affords the biggest contribution to the decision of classification.

Finally, the conclusive output of the SVM, of our approach, is a binary classification decision (Phenotype available? → YES/NO)! This explicit result was a crucial factor, why we decided to use this method. In order to offer this approach to biologists, who usually do not have sophisticated experience or additional skills with computer scientific methods, we were dependent on providing results that do not have to be post-processed furthermore.

1.4.2 Cross-validation

But how can we estimate, how well the created model of the SVM works? One possibility would be to apply the model on a new dataset that was not part of the model training, but which is already labeled. Although this would be an elegant solution, it is quite uncommon in practice. When working with an SVM, each single instance is of crucial importance. Leaving instances out is a luxury, one cannot afford. Therefore, we performed the concept of cross-validation to get a hint of the quality of our models.

In cross-validation, the dataset gets divided into equally sized parts. All parts, except one, are used for model training. The remaining unused division is used for testing. To improve the output, this procedure is repeated several times, until at least, each part was once used for testing ((32):33). At the end, one gets a clue about the instances, which are classified correctly in the different runs by interpreting the mean balanced accuracy and its standard deviation.

$$BalAcc = \frac{1}{2} * \left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP} \right)$$

We used the balanced accuracy to compensate the disequilibrium among our datasets, which usually do not contain an equal number of positive and negative samples.

1.4.3 Genotype abstractions

It is our aim to predict a phenotype, in our case the presence of a bacterial secretion system, by using an analytical approach. Therefore, it is necessary to somehow find a way of describing and quantifying the trait, we are interested in. First of all, we know, as this is a supervised learning approach, if the phenotype is present or not (=“class” or “label”). This information can be represented by a vector consisting of zeroes (=absence) and ones (=presence) (Figure 4, right side of the equation).

	COG1	COG2	...	COGn		Phenotype
Spe1	1	1	...	0	Modell =	0
Spe2	0	1	...	0		0
⋮	⋮	⋮	⋮	⋮		⋮
Spen	0	0	...	1		1

Figure 4 **Theoretical approach abstraction.** This very abstract description of our approach depicts the connection between genotype and phenotype represented by a matrix and a vector.

A much more difficult question is it, to find a representation of the genotype (3.1.2 Genotype representation). There are different levels of depth and different kinds of representation for describing the genomic information. Thus, it is crucial to find a mode that provides all the information that is needed to make a sophisticated prediction on the one hand, but restricts nonessential data on the other hand. Additionally, it is necessary to find a notation which allows us to unambiguously identify and directly compare features from different samples. By using the so called “clusters of orthologous

groups” (COGs) we found an abstraction of the genome that fulfills these preconditions. As “*Orthologs are direct evolutionary counterparts related by vertical descent*” (34), an orthologous group consists of sets of proteins that all belong to one common ancestor (35). The various proteins contained in a COG can be found within taxonomically widely spread species due to several speciation events. Usually, orthologous proteins share a very similar architecture and function. Therefore, it is possible to transfer the attributes of well-studied organisms to those that are not so commonly known, if they share the same COG-profile (34). Theoretically, this hypothesis leads to the idea that the sum of all available COGs is limited in its size by the complete genome of a “last universal common ancestor” (LUCA), which was firstly termed by Patrick Forterre (36), ((37):254f).

Furthermore, the EggNOG 4.0 (38) database, we use, also provides NOGs beside the already mentioned COGs. Those “non-supervised orthologous groups” build some kind of extension to the COGs, as they are fully automatically created from those genes, that cannot be assigned to already known COGs (39). Summarising this, we can say that each COG (or NOG) represents one feature of a sample and thus, is one dimension of our approaches feature space.

As an alternative representation of the genotypic information, we used the so called “Protein Families” (PFAMs). In this case, proteins are grouped by the domains they contain. It is assumed that the function of a domain within different proteins is very similar and therefore might give insights into the behavior of the protein (40). Although, the resolution level of PFAMs is more coarse grained compared to COGs, we wanted to try, if there can be a sophisticated phenotype prediction by using this genotypic representation. As an extension to this kind of genome abstraction, there is also the possibility of grouping proteins by their domain organisations, the so called “architectures”. Those groups can contain more than one kind of PFAM, or more copies of the same PFAM. As the combination of different attributes might lead to a more precise result, we also tried to apply this kind of genome description to our approach (3.1.2 Genotype representation).

Additionally, we were interested in providing our results to a broad community of users, such as biologists, geneticists or computer scientifically skilled method developers, as there is a strong need in secretion related genome annotation.

1.5 Scientific relevance

After explaining our aims and the approach in the chapters above, we have to face the question, if there is a serious need in those data. It would not make any sense to repeat data that is already available for the user. As there are several databases and (online) tools available, which deal with the topic of bacterial secretion, we analysed their offers and tools carefully to avoid useless redundancies. Regarding the secretion systems, the databases SecReT4 (41) and SecReT6 (42) show specific insights into the particular secretion system types. Within their predictions, these databases are focusing on

the presence or absence of structure proteins. Additionally the user gets a sophisticated physical map containing information about the number and position of the secretion system. Very similarly T346Hunter (43) covers the prediction of the same three types of bacterial secretion systems as we do. This database additionally gives a percentage of the found components. Although this might offer a hint for interpretation, it is in the users responsibility to interpret the results correctly. Neither of those three databases gives a binary decision of the prediction. They do not give any indication about the intactness or functionality of the identified secretion systems.

To go beyond the scope of the secretion system focused databases, EffectiveDB also takes the secreted proteins, or effectors, into account. Although, there are already existing databases like BEAN 2.0 (44) or T3SEdb (45) which deal with, at least effectors of T3SS, we also included effector specific information to provide comprehensive insights into the whole process of secretion. Additionally, the former effective database already contained the tools EffectiveT3 and now called EffectiveELD (46) to make sophisticated predictions. We updated and implemented those tools into the new EffectiveDB. There is also an additional way of identifying type III secreted proteins, by finding a specific sequence pattern within their chaperone binding domain. Although, this prediction method was already found in the year 2012 (24), it can provide additional information because of its distinguishable approach. Type IV effector prediction is a more complicated topic as the T4SS, with its various subtypes, is highly modular. Nevertheless, the combination of the amino acid composition, the dipeptide composition, a position-specific scoring matrix composition and the auto covariance transformation of a position-specific scoring matrix can lead to a useful prediction of type IV assigned effectors (47). Another source for type IV effector prediction is T4SEpre (25). This tool focusses on the C-terminal domain of a putatively secreted protein and combines various features. We included parts of this method into our choice of prediction tools. When understanding bacterial secretion as a putative virulence mechanism and factor of pathogenicity, it is also interesting to have a look at online sources like PATRIC (48). This database provides various data concerning bacterial behavior and pathogenicity. It covers fields like antibiotic resistance or diverse occurrences of protein-protein interactions. Nevertheless, the combination of information about functional secretion systems and corresponding effectors are currently not available by a single, freely available source.

Unfortunately, currently, there is a lack of secretion specific annotation. Although, it would be of major impact to include this automatically into genome annotations, there is no indication of such efforts, yet. Moreover, there are no generally valid standards scheduled (49).

Therefore, it is very important to provide secretion specific databases that carefully curate, offer and maintain their data. Complementing the offer of already existing databases, we faced the challenge of

providing a “one-stop-shop” user-friendly prediction-tool, embedded in a comprehensive database for functional secretion systems and their related effectors.

Finally, being aware of the attributes of secretion systems, the features and advantages of machine learning methods, especially the support vector machine and the awareness of the need of this data leads us to the following scientific question, this work raises:

“Is it possible to make a clear an easily understandable prediction of bacterial secretion systems as a phenotypic trait, by using machine learning techniques, especially a support vector machine?”

2 Methods

This chapter introduces the actual design, practices and applied methods to the reader.

2.1 Model design

To provide a sophisticated prediction of bacterial secretion systems, we used the PICA framework ((50), Version 1.0.1) and its support vector machine (SVM) including the adaptations of Roman Feldbauer (33). The whole process of the approach design consists of a training phase and the actual application of the model on new data to predict the occurrence of secretions systems within a group of unclassified bacterial strains (Figure 5).

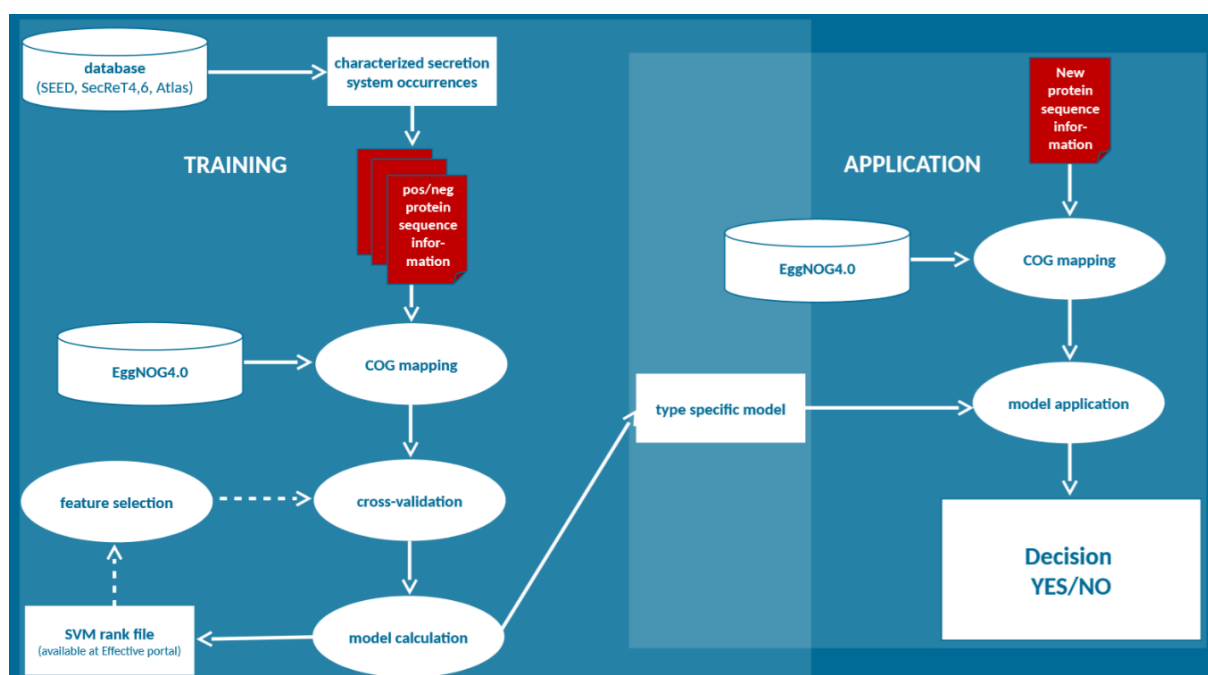


Figure 5 **Concept and workflow of the PICA approach.** This figure shows the modelling workflow of the PICA approach. The left side focusses on the training phase, which includes the data acquisition, its processing and the model construction. On the right side one can see the application of the approach that leads to a binary decision.

2.1.1 Training

It is of major importance to find reliable and well characterised input training data. Therefore we used the information of the following databases (3.1.1 Finding sufficient Training Data):

- SEED (51)
- T3SEdb (45)
- SecReT4 (41)
- AtlasT4SS (52)
- SecReT6 (42)

Additionally, we used up to 18 strains of Chlamydiales regarding T3SS. Their sources come from the literature. The complete training datasets can be found in Suppl.-Table 1.

All of the above mentioned samples are part of the positive dataset, which means that they contain at least one functional secretion system. But using an SVM for binary classification also requires a negative dataset. Finding verified training data, which lack the phenotype of interest, is unlikely more complicated. Obviously, we do not have any database that provides data regarding missing secretion systems. Therefore, we decided to compose our negative training dataset by restriction. Thus, we used all bacterial genomes, available in EggNOG 4.0 (38) and restricted them by all genera, which are contained in the positive set. In regard to type III and VI, we only used the gram-negative bacteria as basic set, as those two types do not appear within gram-positives. Moreover, we only used randomly chosen samples from the resulting pool of data. To further process the input data and create a valid list ("phenotype.txt"), which can be used by PICA, the phenotypic labels of the different samples were assigned in the following manner:

```
taxid    T3SS
1016998  YES
115713   YES
155864   YES
688270   NO
272630   NO
391008   NO
```

Thereby, it is crucial to assign the "name" of the phenotype in the header of the second, tab-separated column.

To achieve a useful genotype-representation, we used clusters of orthologous groups (COGs) (1.4.3 Genotype abstractions). Those clusters can be directly downloaded for all genomes, available in EggNOG 4.0. Samples which are not included in EggNOG, had to be manually calculated. Their COG-profiles were generated by COGnitor (35). Combining all the distinct genotypes, results in a file containing one genome per line. The first item always needs to be a kind of identifier, in our case we used the taxonomy id (53). Afterwards, each single COG is tab-separated listed within the same line ("genotype.txt").

```
502801  COG0001 COG0002 COG0004 COG0005
454166  COG0001 COG0002 COG0003 COG0004
454169  COG0001 COG0002 COG0004 COG0005
```

Using the prepared data as representation of phenotype and genotype, enables us to do a five-fold cross-validation (1.4.2 Cross-validation). Results of over 90% mean balanced accuracy and a standard deviation below 4.5% were used for further model building calculations.

Therefore, the PICA script "crossvalidate.py" was used the following way:

```
python crossvalidate.py -s genotype.txt -c phenotype.txt -t "T3SS" -o
crossvalidation.output > crossvalidation.log
```

To run the script correctly, it is important to write the name of the trait after the switch “-t” exactly the same way as in the header of the phenotype.txt file.

Models of the three described traits (1.2.2 Type III secretion system - 1.2.4 Type VI secretion system) were generated by an SVM of the type “classification” with the default parameters $C=5$, $\gamma=0$ and a linear kernel, done by train.py. The resulting final models can be found in 3.1.3 Final models.

```
python train.py -s genotype.txt -c phenotype.txt -t T3SS -o model.txt
```

With the help of the constructed models, we additionally calculated the feature ranking, to verify if the SVM has captured the correct phenotypic trait (Suppl.-Table 2 - Suppl.-Table 4).

Therefore, the extending script svmFeatureRanking.py by Roman Feldbauer ((33):38) was used. For a correct run of the script one has to define the correct model and a file (in our case: NOG.description.txt) which assigns the COG/NOG identifiers to the descriptions.

```
python svmFeatureRanking.py model.txt -d NOG.description.txt > svm_rank.txt
```

PFAMs

Using PFAMs (1.4.3 Genotype abstractions) as an alternative form of a possible genotype representation, decreased the calculation time considerably. This happens due to a smaller number of features on the one hand and additionally, because the PFAM-profile is already created within the calculations of EffectiveELD (3.2.2 Workflow). In that case the very time consuming step of generating a COG/NOG-profile by COGnitor (35) is omitted. Instead, a PFAM-profile, created by Interproscan ((54), version 5.11-51.0), is generated in the following manner:

```
1002804 PF00239 PF00248 PF01051 PF01420
1005048 PF00004 PF00005 PF00006 PF00009
1005090 PF00004 PF00005 PF00006 PF00009
```

PFAM-architectures

Additionally, we used a third type of genotype representation, the PFAM-architectures. Those components can contain one or more PFAMs in different copy numbers. This combination might lead to a higher specificity and therefore create a more precise description than the single PFAMs.

Therefore, the genotypic profile was created by using the script genotype_pfam_architecture.py (Suppl.-Script 2 **Source code for EffectiveDB pipeline**) to generate an input-file of the following format:

```
1002804 PF11074 PF02661 PF01656 PF00248
1005048 PF10590|PF01243 PF02779|PF00456|PF02780 PF00691|PF04972 PF01865
1005090 PF00456|PF02779|PF02780 PF00146 PF01197 PF03099
```

Regardless of the actually used kind of representation, the process of model training remains the same as PICA does not give any restrictions to the type of a feature format.

Feature selection

Another option of seriously reducing the calculation time, was to decrease the problems dimensionality, drastically. Therefore, we tried to restrict the feature space to a sufficient number of features instead of using all the features that represent the whole genome.

To achieve a proper selection of the most important features, we firstly had to train an optimal model, as it is shown in the first paragraph of this chapter. Afterwards, we used the optimised models 18_93_202 (T3SS), 22_57_100 (T4SS) and 28_151_200 (T6SS) which can be found online at the EffectiveDB web-portal. To restrict the size, we chose only the top members of the ranked features. The model of T3SS was restricted to the 500, T4SS to 5000 and T6SS to 2000 top ranked COGs. Those COGs/NOGs were again used to train a new model for each phenotype. Thereby, the genotype files contained only the preselected COGs/NOGs.

Because there were no satisfying results of the runs using PFAMs (detailed results can be found in 3.1.2 Genotype representation), we decided to only use the COG/NOG-profiles for our further work. Unfortunately, also the feature selection did not lead to the simplification, we hoped for (3.1.2 Genotype representation). Nevertheless, we had to handle the time consuming step of the COG/NOG profile creation. Therefore, we decided to do the calculations of submitted jobs with the help of a queuing system on our grid (3.2.2 Workflow).

2.1.2 Model application

To apply the created models on new datasets, we again had to create the necessary genotypic profiles by using COGnitor(35) in our case. After that, the PICA script “test.py” was performed, using the new genotype.txt file and the already provided, model of choice (model.txt is an output of the former performed train.py).

```
python test.py -s genotype.txt -m model.txt -t T3SS > test.txt
```

2.2 Application implementation

The three models created in the chapter above are part of the EffectiveS346 application for the prediction of functional secretion systems. This tool is embedded in a database and web application EffectiveDB, which is exclusively designed for secretion specific investigations of bacterial genomes. This chapter should give the reader insights in the actual implementation of the procedural pipeline and the different applications. As the code is mainly written by Dr. Thomas Nussbaumer, I just want to give an overview of the distinct steps of the pipeline and a brief description of the EffectiveS346 application.

Beside the construction of the new prediction pipeline, the database and the already existing applications EffectiveT3 and EffectiveELD were updated. Therefore, new training datasets were

generated. This work was mainly done by Dr. Alexander Platzer, Dr. Thomas Nussbaumer and Marc-André Jehl.

The actual pipeline consists of the following ten steps, which are displayed to the user by a progress bar. If an Error occurs, the progress bar automatically switches to step 10 and the run is finished. Because of the occasionally long computing time, especially when enabling the genome mode, the user has the option to leave his e-mail address and receives a message, when the job is finished. After submitting a job, it is assigned to a unique RUN-ID and stored for the chosen steps of processing.

The following table (Table 2) should provide a structured overview of the ten different steps of the prediction pipeline. Each section should give the answer to the following questions:

- What happens and which shell-script is used that for?
- Which input information (files, parameters, etc.) is necessary to successfully run the current step?
- What kind of errors are handled? Which error-information is provided to the user?
- Which results (files, information, etc.) are produced to go on with further steps or to contribute to a meaningful final result.

Table 2 Stepwise pipeline description. The table shows the sequential workflow of the EffectiveDB pipeline.

Step 1 - Multiple FASTA file check

<i>What happens</i>	• Check if any input sequence is provided. • Check if the typical FASTA format is fulfilled by running <code>check_seq.py</code>
<i>Input Parameters/Files</i>	Multiple FASTA input file
<i>Errors raised</i>	• ERROR: Wrong format of FASTA, Multiple Fasta Sequences • ERROR: Proteins are provided in the wrong format (e.g. *.fasta)
<i>Output</i>	• status.TXT • PROTEINS.FILTERED • PROTEINS.IGNORED

Step 2 - Predotar

<i>What happens</i>	Effector targeting by running <code>Predotar.jar</code>
<i>Input Parameters/Files</i>	• Enabled Predotar • animal (default) or plant mode • PROTEINS.FILTERED
<i>Errors raised</i>	ERROR: Modus is invalid either 'p' (plant) or 'a' (animal) mode
<i>Output</i>	• Predotar_output_modelANIMAL.TXT • Predotar_output_modelPLANT.TXT

Step 3 – EffectiveT3

<i>What happens</i>	Type III effector prediction by using java TTSS_GUI-latest.jar
<i>Input Parameters/Files</i>	• Enabled EffectiveT3 • minimal score • New (default) or old model • PROTEINS.FILTERED
<i>Errors raised</i>	• ERROR: Invalid value for EffectiveT3 new model • ERROR: Invalid value for EffectiveT3 old model
<i>Output</i>	• status.TXT • EffectiveT3.NEW.OUT • EffectiveT3.OLD.OUT

Step 4 - EffectiveCCBD

<i>What happens</i>	Type III effector prediction by running find_CCBD.py
<i>Input Parameters/Files</i>	• Enabled EffectiveCCBD • PROTEINS.FILTERED
<i>Errors raised</i>	-
<i>Output</i>	• status.TXT • PROTEINS.CHAPERONES.TXT

Step 5 – T4SEpre

<i>What happens</i>	Type IV effector prediction by running predict_T4SS.py
<i>Input Parameters/Files</i>	• Enabled T4SEpre • score
<i>Errors raised</i>	ERROR: Parameter in T4SEpre has to be a float-number
<i>Output</i>	• status.TXT

Step 6 - EffectiveELD

<i>What happens</i>	Effector prediction by running eld_filter.py
<i>Input Parameters/Files</i>	• Enabled EffectiveELD • Z-score • PROTEINS.FILTERED • iprscan_pfam.tsv (created by running mask_gi.py and interproscan.sh on beforehand)
<i>Errors raised</i>	ERROR: Invalid value for EffectiveELD
<i>Output</i>	• status.TXT • PROTEINS.ELD.TXT

Steps seven to nine are only relevant, if the genome mode is enabled!

Step 7 – CheckM

<i>What happens</i>	Check for genome completeness by running <code>parse_checkM.py</code>
<i>Input Parameters/Files</i>	• Enabled genome mode • Enabled CheckM • CheckM_COMPLETENESS (threshold of completeness, default=80) • PROTEINS.FILTERED
<i>Errors raised</i>	• ERROR: checkM – genome completeness check failed • ERROR: Invalid value for CheckM-COMPLETENESS parameter
<i>Output</i>	• <code>status.TXT</code> • <code>checkM_output.TXT</code>

Step 8 COGnitor and Step 9 EffectiveS346

```

519      # -----
520
521      if [ "${RUN_EffectiveS346}" = "T" ]
522      then
523
524          if [ -z ${GENOTYPE} ]
525          then
526              echo "COGnitor run..."
527
528              ./COGNITOR/cognitor_V2.sh ${RUN_ID}
529              echo -e "8\tStep 8 finished COGnitor" >>
530  ${RUN_ID}/status.TXT
531              python EffectiveS346/extract_protein_COG_pairs.py
532  ${RUN_ID} > ${RUN_ID}/PROTEINS.genotype.txt
533
534          else
535              cp ${GENOTYPE} ${RUN_ID}/PROTEINS.genotype.txt
536              echo -e "INFO\tThank you for providing a genotype,
537  COGnitor run is not needed here"
538              fi
539              echo "EffectiveS346..."
540
541              COG_FILE=$(mktemp)
542              less ${RUN_ID}/PROTEINS.genotype.txt | cut -f 1 | uniq | sed
543  '1 i\genome' | sed ':a;N;$!ba;s/\n/\t/g' > ${COG_FILE}
544              python EffectiveS346/application/scripts/automate_SecPred.py
545  ${RUN_ID} ${COG_FILE}
546              if [ $? = 1 ]
547              then
548                  python EffectiveS346/compare_top100.py ${RUN_ID}
549                  python EffectiveS346/compare_kegg.py ${RUN_ID}
550              fi
551              show_time $date1
552              cp ${COG_FILE} ${RUN_ID}/COG_FILE.CHECK
553              rm ${COG_FILE}
554
555              echo -e "9\tStep 9 finished EffectiveS346" >>
556  ${RUN_ID}/status.TXT
557              fi
558
559      # -----

```

In regard to the steps 8 and 9, I want to describe the source code a little bit more in depth because this section deals with the prediction of the secretion systems, which is the main part of this work. The secretion system prediction is the major section of the so called genome mode, which can be obtained when submitting an almost complete genome to the prediction pipeline.

Enabling the genome-mode and also EffectiveS346 (line 521), automatically starts the COGnitor run (line 528), if the almost complete genome is provided as protein sequence. In this case, COGnitor creates a COG/NOG-profile of the whole genome, to run EffectiveS346 successfully. After that, step eight is finished.

Alternatively, it is possible to provide an own genotype file containing a COG/NOG profile (line 535). In this case, the time consuming COGnitor run (step 8) is omitted. Additionally, the provided data is checked for validity by running `check_genotype.py`. In case of an invalid input, the message “ERROR: Invalid Genotype file” is thrown and the submitted job is directly led to step 10.

Independently of the chosen source of the genotype file, EffectiveS346 can be started by running `automate_SecPred.py` (line 545). This simple shell script uses the created models and checks sequentially for gram-negativity, T3SS, T4SS and finally T6SS. All the results are summarised and displayed in a simple table. Within step 9 the results are processed by running `compare_top100.py` (line 548) and `compare_kegg.py` (line 549). Within these two comparisons, the occurrence of typical structure proteins, as suggested by KEGG and of the top 100 COGs/NOGs, of the formerly calculated SVM rank-file, are made.

Step 10 finish Effective run

<i>What happens</i>	• Provide results and finish job by running <code>merge_results.py</code> and <code>summary.py</code> • Send email if address is stated
<i>Input Parameters/Files</i>	• EMAIL (optional)
<i>Errors raised</i>	-
<i>Output</i>	• <code>status.TXT</code> • <code>merge_results.html</code> • <code>summary.TXT</code>

After finishing Step 10, the user is directly guided to the results page and the finally processed output-files are downloadable in diverse formats. It is also possible to return to the results page of a specific job ID later on.

3 Results and Discussion

This section deals with the results of this work in the focus of the actual PICA models, their application EffetiveS346 and its web-oriented background and the comparison of our method to existing methods provided by others.

3.1 PICA models

Regarding the design of our experiment, we firstly had to deal with two major questions. On the one hand, we had to find a sufficient number of reliable training data, on the other hand it was essential to find the optimal way of a genotypic representation.

3.1.1 Finding sufficient Training Data

The first step of our work was to train a proper model for each phenotype we are interested in. Therefore, we needed samples consisting of two components. On the one hand we need to know, if a bacterial strain (=sample) shows the specific phenotype. Secondly, it is essential to have the actual genome sequence of each sample. Ideally, more than 30 (personally experienced threshold of Roman Feldbauer) samples of each, positively and negatively proven phenotypic expressions, can be found. To find such samples, there are a few different approaches:

- Using the data of working groups, specialised in the field of interest
- Text-mining, for example within the PubMed database (53)
- Using specialised databases

It went out to be challenging, to find a sufficient number of experimentally proven data within the literature and additionally, the proper sequence data, just by manually searching. Doing an automatic text search approach would have probably also led to a satisfying result, as already done for the SecReT databases (41, 42), but was not tried by us. Our search was terminated by the usage of established databases. Nevertheless, we had to choose our sources responsibly. It might not have been sufficient to use in-silico predictions of other databases or prediction tools as “verified” data. Additionally, it appears to be almost impossible to find verified negative data. There is no database that contains any data about species that were proven to not contain a specific secretion system. On the other hand, the fact that one did not find any secretion system, might not be evidence enough for the real absence of such a system. This is the reason why we selected our negative dataset by blacklisting the positive genera from the species that are available in EggNOG. To generate a reliable positive dataset, we decided to use samples from the following databases. Please note that some of the samples are found not only by one, but several sources.

SEED

Within SEED (51) the annotation is done manually by an expert curator. He uses subsystems that are found across many different genomes. Afterwards, protein families are extracted from those subsystems and furthermore used as so called “core components” for automated annotation. Although, the use of subsystems does not give us any indication of functionality in general, we decided to use this data because of the work of the expert curators. We assume that the experts from the field have the knowledge to assign only those genomes to be functional, which really show the trait of interest. Using only the proteins from those subsystems to find new ones, should give a true indication of functionality, even though we cannot find any evidence in literature.

In total, we found 173 positive samples (for all three systems) in the SEED database. (Suppl.-Table 1)

T3SEdb

T3SEdb (45) gives a clear indication whether a sample is experimentally validated or not. To identify such proven data, they did an NCBI Entrez Protein database (55) text search. Afterwards the different samples were manually checked and assigned to their experimental validation status. Although this database is developed in regard to type III secreted effectors, we used the data as some kind of indirect indication of a functional secretion system. We made the assumption that effector specific experiments can only be made, if the species of interest has the ability to functionally secrete. We used 17 samples of the status “E” for the positive dataset of T3SS.

SecReT4

SecReT4 (41) is a database which is specialised to T4SS. In this case a text mining approach of PubMed entries was the underlying source of data. The very detailed references of SecReT4 (as well as SecReT6) to sources in the literature gave us the indication of reliable data that can be used as valid positive samples. Although we only used the experimentally validated samples to build our training data set, SecReT4 also makes own predictions. In such cases, all of the 18 previously identified core components have to be identified. If there was a core component availability of at least 30% the distinct sample was manually annotated. 35 of our type IV specific positive training samples derived from SecReT4. Species that were predicted by SecReT4 to have a secretion system, were used for comparison with our predictions (3.3.1 SecReT4 and SecReT6).

AtlasT4SS

In regard to T4SS we had to face that only very few samples are currently known. Thus, we also added samples of this database. Unfortunately AtlasT4SS (52) does not give any indication about the completeness of a secretion system. It is rather focusing on the presence of individual components than on functionality. Nevertheless it is manually curated. We added 19 samples that show a minimal threshold of 20 components.

SecReT6

SecReT6 (42) works very similarly to SecReT4. The only major difference is that a full set of T6SS describing components consists of 13 COGs. The threshold for manual curation is defined by a minimum set of five COGs. As already mentioned in relation to SecReT4, the accurate literature based search of experimentally verified samples let us assume real functionality. We gained 40 samples from this database.

At the end, the combination of the different input data sources enabled us, to create sufficient and diverse datasets to train our models.

3.1.2 Genotype representation

The second, quite challenging, part of modelling our traits, was to find an optimal representation of the genotypic information. A genome can be represented in various levels of depth. It is possible to describe it for example by its nucleotide sequence, proteome or by theoretically constructed entities like protein families or orthologous groups, for example. The following three different representations were tested by us (1.4.3 Genotype abstractions):

- COGs/NOGs
- PFAMs
- PFAM-domain architectures

It was our aim to find a way of preserving the genotypic information we need, but contrariwise reduce the amount of data efficiently, to optimise the computational effort, by using those different entities.

To verify the quality of the representation, we made use of two features:

- Mean balanced accuracy and standard deviation of each calculated model (1.4.2 Cross-validation)
- Occurrence of already known structure proteins (1.2.1 Structure proteins)

SVM rank-files

To verify, if the chosen depth of representation is sufficient to transport the classifying information, we made use of the so called “SVM rank files”. Those output files of PICA, which are a representation of the contributing classifying features, are of crucial importance in regard to the optimisation of the classification models. Investigating the ranking of the features gives us insights, if the SVM has “captured” the trait we were looking for. If so, we would expect to find the most important structure proteins (1.2.1 Structure proteins) within the top ranked features. It happened several times that our cross-validation showed a very promising mean balanced accuracy, but further analysis of the rank-file showed, that the trait of interest was not found. (e.g. run 16_97_200: mean balanced accuracy 87,2%;

top 20 of SVM rank-file can be found in Suppl.-Table 5). This Phenomenon might occur due to imbalanced datasets. In already mentioned run 16_97_200 we added 18 samples of the order Chlamydiales. Unfortunately, seven samples belong to the species *Chlamydia trachomatis* (Figure 6). This experiment indicates that a lack of far related, widely spread organisms, causes some bias. In this case the SVM did not find the specific trait (as it can be seen in the rank file), but focused on basic features that are independent of secretion.

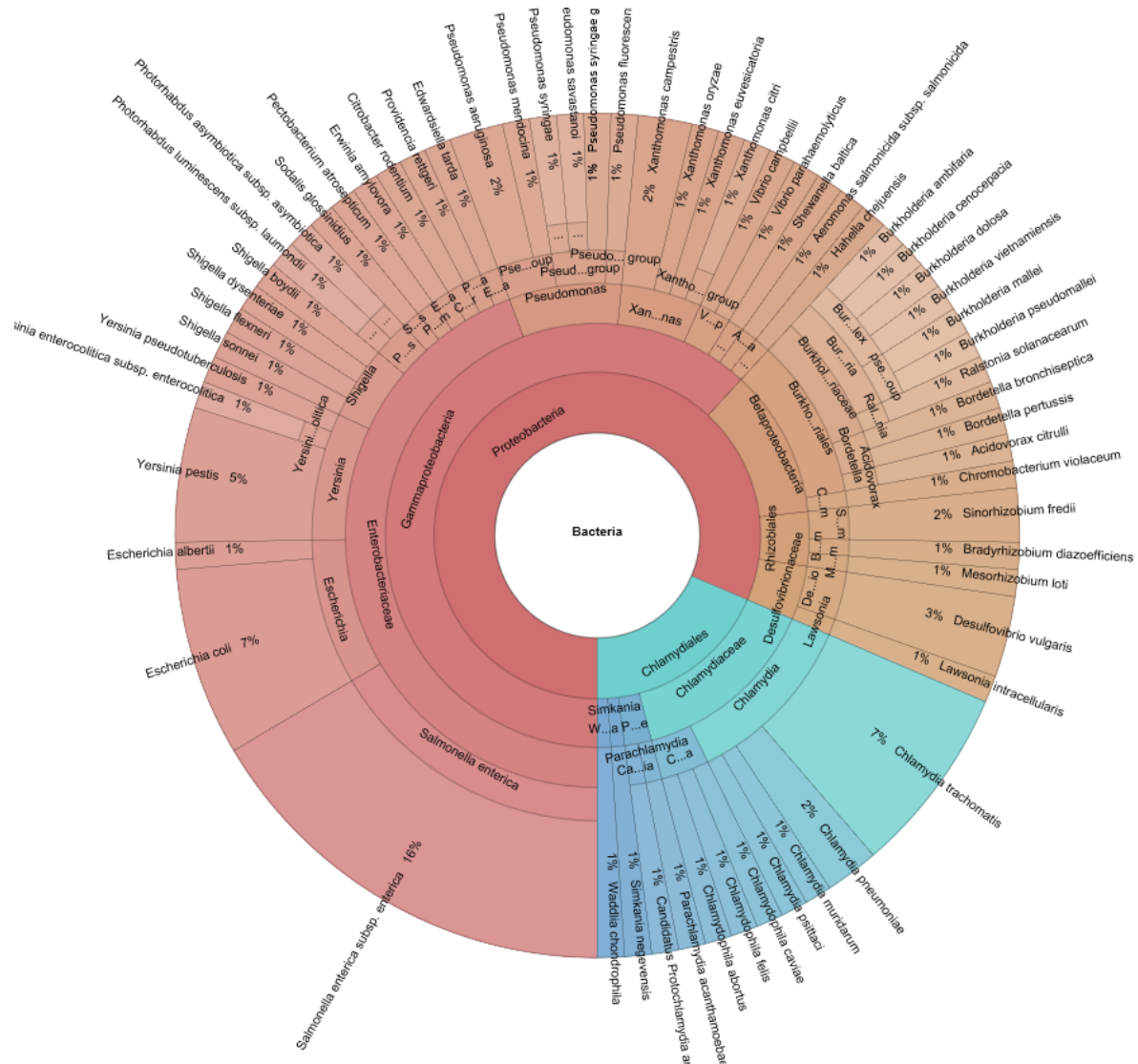


Figure 6 **Taxonomy of positive dataset run 16_97_200.** This krona-chart (75) shows a taxonomical overview of the positive dataset of run 16_97_200. It can be seen that almost 20% of the dataset belongs to the Chlamydiales (blueish part). Furthermore, within this order 39% of the samples even belong to the same species *Chlamydia trachomatis*.

Another very important aspect, which is offered by the SVM rank file, is the identification of proteins that have not been assigned to the specific trait, yet. Within the top 10 ranks of an SVM rank file, we typically find the COGs/NOGs that can be assigned to the particular system (those COG/NOG, PFAMs and PFAM-domains are depicted in the “expected”-columns of the following three Tables). The presence of those features gives us the indication that the correct trait was found. The score, which is stated in the fourth column shows the relative importance of the feature in the classification. In other

words, the feature that has the highest score (= rank 1), plays the most important role for the classifying prediction. Although the score becomes lower and lower when moving down the list, those features do also contribute to the decision of classification. Such lower sorted features can represent chaperones, for example, or other supporting proteins. But even proteins which are unknown or which have never been associated with the trait, might be found in this manner. Those unexpected occurrences might be an interesting starting point for further investigation. Especially in regard to the exact specification and separation of different secretion systems, their subsystems, or even the detection of new secretion systems and their contributing factors, those features are of major importance.

Moreover, it can happen that features are assigned to a negative score in the rank-file, which indicates the absence of the specific feature. This can also be a revealing information. For instance the prediction of obligate intracellular lifestyle, investigated by Roman Feldbauer (33), significantly relies on such negative scores.

Using the SVM rank files as a kind of describing tool, we tested the following types of genome representations:

COG/NOG

In general, the SVM of the PICA framework can deal with manifold types of information. There is no specific need of using COG/NOG profiles. Nevertheless, previous experiences of our colleges, for example the work of Roman Feldbauer (56) showed that the usage of COGs and NOGs seem to be at least a meaningful way of starting our experiments. Additionally, within the PICA framework originally EggNOG2.0 (57) was used, which gave us the indication that using COGs/NOGs might work with our approach too. Nevertheless we are aware of the fact that an alternative source of input data might not only be possible, but could even improve the results. For example it would be interesting if the usage of the OMA database (58), for instance, might show different results.

It turned out, when using COGs/NOGs, the very highly ranked, most significant classification features, belong to the group of manually curated COGs. The SVM rank-file of classifying features, which is part of the automatically produced output of PICA, also contains the in-silico generated NOGs (e.g. observed rank #8 in Table 3), but in our case, those clusters do not seem to have such a big influence on the classification decision.

We used the structure proteins for secretion systems, suggested by KEGG (6, 59), to get an impression of expectable COGs. Seven of the type III assigned, corresponding COGs were found within the top ten ranked features (Table 3).

Table 3 **Comparison of expected vs. observed COGs of T3SS.** The table gives a descriptive overview of the top ten ranks of the SVM rank file for a COG-represented genotype.

COG/NOG T3SS run 18_93_202

rank	exp	obs	Score	Group_description
1	COG4791	COG4791	0.0451327093871	type iii secretion
2	COG4669	COG4669	0.03911983819	type iii secretion
3	COG4789	COG4789	0.037949732826	Type III secretion
4	COG4790	COG4790	0.0340518828083	Type III secretion system protein
5	COG4794	COG4794	0.0304019478839	type iii secretion
6	COG4792	COG4792	0.0269867899847	type iii secretion
7	COG1450	COG1450	0.0191699922235	general secretion pathway protein D
8	COG0659	NOG00650	0.0180774117106	major facilitator superfamily
9	COG1157	COG2032	0.0158158211414	Destroys radicals which are normally produced within the cells and which are toxic to biological systems (By similarity)
10	COG1317	COG0791	0.0155378363758	NLP P60 protein

Single PFAM-domains

Especially in regard to the implementation of the PICA models into the online application of EffectiveDB, we had to face that calculating the whole COG-profile, by COGnitor (35) for each uploaded genome sample, would be of quite high computing effort. To find a possibility of bypassing this step, we were interested in using the protein family domains, created by Pfam 28.0 (40) as representing format of our genotype. Not only the decreasing degree of dimensionality of the SVM seemed to be convincing. Another simplifying fact was that the PFAM-domains are already calculated within EffectiveDB to provide the eukaryotic-like domains as part of EffectiveELD. Thus, using Pfam-domains would not have caused additional computational effort.

Nevertheless, the usage of Pfam-domains instead of COGs did not show the results we hoped for. Run 31_90_71 was a small test run, based on the optimised model 18_93_202, containing 161 strains altogether. We wanted to see, if there is a significant pattern of Pfam-domains within the SVM rank-file. If so, we would have been able to refine the model and its samples to optimise the output. Unfortunately, the rank file did not show any connection to secretion systems. The mean balanced accuracy of over 89%, and the quite small standard deviation below 6% suggested a proper result, whereas the rank-file at least showed quite the opposite (Table 4). Using the COG profiles showed us that secretion systems in general are more likely to be found by the presence of specific COGs than by

their absence. If we take a look at Table 4, we see that 6 out of the first 10 PFAM-domains are tagged to be absent (indicated by “-”), to “predict” a T3SS.

If we have an in depth look at the specific, highly ranked PFAMs, our impression gets confirmed. Some of the top ten features can be assigned to very general helicases. Rank 9 is of unknown function.

We explained this result because PFAMs are quite general in their description and annotation. So we decided to repeat the experiment by using PFAM-domain architectures instead of single domains. This could probably show us a more precise result and a better resolution regarding the secretion systems, we are looking for.

Table 4 Comparison of expected vs. observed PFAMs of T3SS. The table gives a descriptive overview of the top ten ranks of the SVM rank file for a PFAM-represented genotype.

PFAM T3SS run 31_90_71

rank	Exp	obs	Score	PFAM Description of observed feature
1	PF01311	-PF01051	-0.121285762356	Initiator Replication protein
2	PF01514	PF03677	0.106714168712	Uncharacterised protein family (UPF0137)
3	PF00771	PF00772	0.0923447703281	DnaB helicase N-terminal domain
4	PF00813	PF03796	0.0923447703281	DnaB helicase C-terminal domain
5	PF01313	-PF07724	-0.0903603395021	AAA domain (Cdc48 subfamily)
6	PF01312	-PF02086	-0.0901953813472	TrkA-C domain
7	PF00263	-PF05973	-0.0854245868448	Phage derived protein Gp49-like (DUF891)
8	PF09392	PF05475	0.0850469983972	Chlamydia virulence protein PGP3-D
9	PF01052	-PF11074	-0.0836164979248	Domain of unknown function(DUF2779)
10	PF06188	-PF01856	-0.0824338815473	Helicobacter outer membrane protein

PFAM-domain architectures

To generate a sufficient map of PFAM-domain architectures and proteins, we mapped all the associated PFAM-domains to the different proteins of all input samples. In other words, each occurred protein was then represented by one or more PFAM-domains. (Source code can be found in the Supplement. Suppl.-Script 2 **Source code for EffectiveDB pipeline**) The PICA run 32_90_71 was based on the same dataset as the run of single PFAM-domains (31_90_71), except the “new” genotypic profile.

Again, the results were not satisfying. Almost the same domains were ranked up highly. Additionally, complex architectures were not that relevant and again, single PFAMs came first (Table 5).

Although run 32_90_71 shows a proper accuracy of 89.9% and a quite low standard deviation below 5%, we could not find any connection between the PICA results and the trait of a T3SS.

Both experiments with Pfam-domains led to the assumption that the bad results can be explained by the higher variability of domains compared to the orthologous groups. It seems that within the same ortholog there can be different domain architectures that might lead to a different genotypic profile.

To verify this hypothesis, we had a deeper look in the results of the two strains *Desulfovibrio vulgaris* Hildenborough (taxid: 882) and *Desulfovibrio vulgaris* 'Miyazaki F' (taxid: 883).

Both of them share 1292 domain architectures and there are additional 685 more architectures that are only present in one of them. It caught our attention that the shared domains in most cases (874) are single-protein domains. Opposed to that, 406 out of 685 specific architectures consist of at least two domains.

Compared to the genotypic profile generated by COGs we see again that there are 1492 shared COGs/NOGs and 1016 specific orthologous groups. It seems that again there is a similar property of shared and specific items which makes us believe that the crucial difference lies within the domains and architectures itself.

Table 5 Comparison of expected vs. observed PFAM-architectures of T3SS. The table gives a descriptive overview of the top ten ranks of the SVM rank file for a PFAM-architecture-represented genotype.

PFAM-architecture T3SS run 32_90_71

rank	exp	Obs	Score	PFAM-architecture description of observed feature
1	PF01311 PF01391	-PF1051	-0.12351134544	Initiator Replication protein
2	PF01514 08345	PF00589	0.116888965041	Phage integrase family
3	PF00771 PF00771	PF03677	0.10896169784	Uncharacterised protein family (UPF0137)
4	PF00813	-PF02086	-0.0962120973865	DNA methyltransferase
5	PF01313	PF03796 PF00772	0.0954470478667	DNA helicase
6	PF01312	-PF07804	-0.095205230544	HipA-like C-terminal domain
7	PF00263	-PF11074	-0.0923076368499	Domain of unknown function(DUF2779)
8	PF09392	-PF01856	-0.0915418766171	Helicobacter outer membrane protein
9	PF01052 PF01052	-PF07804 PF16948	-0.0915418766171	HipA-like C-terminal domain protein
10	PF06188	-PF09491	-0.0915418766171	AlwI restriction endonuclease

Feature selection

A completely different approach, with the aim of reducing the computational time, was to only use the most important and not all of the classifying features. This results in a much shorter problem dimensionality.

Therefore, in regard to T3SS, we only used the top 500 features of model 18_93_202. After testing species of interest with the restricted genotype of 500 items per maximum, in 100% of the cases, the prediction turned out to be “NO”. We explained this by the expectancy of the SVM that a sample should contain about 20.000 features. If this is not the case the SVM, with its global view on the genotype, will respond with a not fulfilled expectancy and classifies the sample as “NO”.

As a second experiment, we used the 500 highest ranks to retrain a model that only uses the restricted pool of features. After testing the restricted genomes with the “smaller” model, we received the astonishing cross-validation result of a mean balanced accuracy of 99.9% and a standard deviation of only 0.5%.

We expanded this experiment on the types IV and VI and therefore used different numbers of restricted features. Regarding type VI our best model 28_151_200 was restricted to only 200 items. Firstly, it showed the resulting accuracy of 98.1% and a standard deviation of only 1.4%. Unfortunately, having a deeper look at the SVM rank-file does not show a connection to our trait.

Obviously, using only 200 items is far too less for a significant prediction. At the end, we found the following minimum numbers that still enable a reliable classification (Table 6).

Table 6 Results feature selection for optimised models. Overview of the accuracies and standard deviations for the final, optimised prediction models and assigned feature selection runs.

Secretion system	Run	Mean balanced accuracy	Standard deviation	Number of selected features	Mean balanced accuracy (fs)	Standard deviation (fs)
T3SS	18_93_202	91.5%	4.2%	500	99.9%	0.5%
T4SS	22_57_100	92.4%	4.5%	5000	97.8%	2.9%
T6SS	28_151_200	90.7%	2.9%	2000	98.9%	1.2%

As one can see within the table above, the mean balanced accuracies of the cross-validations are much higher after feature selection than concerning the complete models. Although this might lead to the assumption of very good models, we have to explain this high accuracy by other disturbing phenomena. One explanation might be the reduced problem complexity because of a smaller number of features. This idea is underlined by the results of Roman Feldbauer ((33):48).

Nevertheless an increase from 91.5% to 99.9% seems to be too high to be ascribed only to this behaviour. Another explanation might be that within the SVM, the independent variables are given a higher weight ((33):38). By restricting the feature space, we created an artificial independence of the different features that might additionally cause such a high accuracy.

The feature selection experiment showed a significant decrease of dimensions for T3SS. In order to T6SS and T4SS we found out, that it is necessary to have at least 2000 and 5000 features respectively, available. This reduction will on the one hand still allow a sophisticated prediction model, on the other hand the decrease of the computing time will not be sufficient to use those methods for the integration of our application into EffectiveDB.

Finally, we used the original COG/NOG-profiles from EggNOG 4.0 as genotypic representations. To handle the challenge of the quite high computing time, we decided to calculate the genotypic profiles of online submitted samples on our grid system (3.2.2 Workflow).

3.1.3 Final models

The finally used and optimized models can be overviewed in Table 7. All of them show a mean balanced accuracy of over 90% and a standard deviation below 4.5%. Additionally, their trait specific affiliation gets explained by their SVM rank-files. Detailed information about the training datasets and the rank-files can be found in

Suppl.-Table 1 - Suppl.-Table 4. The models can be obtained from the EffectiveDB web portal.

Table 7 Overview of the three finally used, optimised models of the PICA approach. All of the accuracies are above 90% and the standard deviations below 4.5%. The top ten ranks are compared to the expected structure proteins from KEGG (6, 59). The dark-red colour highlights mismatches of expected and observed COGs. T6SS consist of only 9 structure proteins, which is the reason for the empty field for rank number 10.

runs	T3SS		T4SS		T6SS	
	18_93_202		22_57_100		28_151_200	
	m bal acc	std dev	m bal acc	std dev	m bal acc	std dev
rank	91.5%	4.2%	92.4%	4.5%	90.7%	2.9%
	exp	obs	Exp	obs	exp	obs
	1	2	3	4	5	6
1	COG4791	COG4791	COG3505	COG3505	COG3523	COG3516
2	COG4669	COG4669	COG3451	COG3451	COG3501	COG3517
3	COG4789	COG4789	COG2948	COG2948	COG3157	COG3522
4	COG4790	COG4790	COG3736	COG736	COG3455	COG3455
5	COG4794	COG4794	COG3504	COG3504	COG0515	COG3520
6	COG4792	COG4792	COG3702	COG2919	COG0542	COG3522
7	COG1450	COG1450	COG3704	COG1125	COG3456	COG3519
8	COG0659	NOG00650	COG3838	COG1174	COG0631	COG3515
9	COG1157	COG2032	COG3701	COG1732	COG3501	COG3501
10	COG1317	COG0791	COG0630	COG0630	-	COG3518

Concerning the types III and IV we see a quite good overlap of the expected and the observed COGs. The observed ranks 7, 8 and 9 of T4SS belong to ABC-Transporter which are indirectly connected to bacterial secretion.

What appears to be suspicious is the low overlap (7 mismatches out of 9 COGs) of observed and expected COGs in regard to T6SS. Moreover, even the very most important features in the top 3 ranks are not matching with KEGG structure proteins (1.2.1 Structure proteins). This can be explained by having a deeper look into the descriptions of the specific COGs (Suppl.-Table 4). Five of them belong to a complex so called the “putative cytoplasmic components”, the two remaining features COG3516 (TssB) and COG3517 (TssC) can be found in a complex together with the ClpV protein which is represented by COG0542 (7) (Figure 7) Thus, we can directly assign our results to the type VI secretion, although the structure proteins of KEGG do not have such a significant influence on the classification.

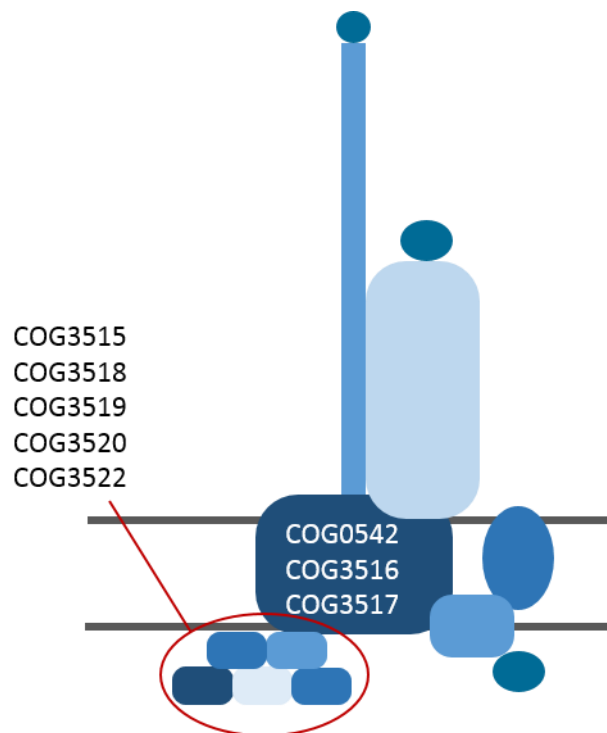


Figure 7 COG occurrence concerning T6SS. The T6SS interacts with proteins called „putative cytoplasmic components“. The five COGs representing them, seem to have crucial influence on the classifying decision of the SVM. Additionally, we found COGs of the proteins TssB (COG3516) and TssC (COG3517) that form a complex together with ClpV.

3.1.4 Limitations and advantages of using a support vector machine for phenotype prediction

Although each of the models shows a mean balanced accuracy of more than 90% and a standard deviation below 4.5% (Table 7), we have to face the challenges that occur, when using an SVM. Regarding the training data it is absolutely crucial to find independent data for an optimal model building. Obviously it is impossible to find completely independent data in regard to taxonomic

relationships between the distinct chosen strains. Furthermore there should be some far related strains, to create a more or less broad representation of the overall taxonomy and not only a short part of it. Additionally it is also important to have positive and negative samples of closely related strains. Those samples are important to train the SVM for the slight differences within the genotypes of taxa that show the phenotype or not. If there is a lack of such closely related samples, the approach might only predict the taxonomic relation itself and not the presence or absence of the trait of interest.

To omit these kind of mispredictions, we also created datasets that contain “true negatives”, found in the literature (Table 8). Unfortunately, those runs did not show the results we hoped for. It went out that although these models showed the specific trait within their rank files, they showed a rather bad accuracy of their cross-validations and additionally a higher standard deviation than the optimised models.

Table 8 True-negatives containing models. Comparison of the mean balances accuracy and the standard deviation of the finally used models to the models containing manually added “true negative” samples that do not show the trait of interest.

Secretion system	Run	M. bal. acc.	Std dev	Run incl. TN	M bal acc incl. TN	Std dev incl. TN
T3SS	18_93_202	91.5%	4.2%	12_54_813	82.1%	7.5%
T4SS	22_57_100	92.4%	4.5%	30_57_107	83.4%	6.4%
T6SS	28_151_200	90.7%	2.9%	29_149_305	87.1%	4.9%

This effect might have occurred because of false “true negatives”, which means that they have the specific phenotype, but it is currently unknown. Another interpretation might be that there were too less examples of them and the training data was not well-balanced. In this case the few additional examples would have disturbed the actual negative dataset. Although this actions should work against the probable taxonomic over-fitting of the training data, we were not able to include them because they would have confused the classifying parameters too much. On the other hand, we showed that it is possible to minimise the over-fitting of the training data, focusing on one single species in the dataset.

Correct classification of virulent and non-virulent Escherichia coli strains

For the model of T3SS, we wanted to find out, if it is possible to reduce the taxonomic bias just for one single species. If so, we might be able to reduce the effect of predicting the taxonomic relation instead of the trait of interest, at least for this specific taxon. This experiment worked quite well in regard to Escherichia coli. We added two, well known non-pathogenic strains to the negative dataset (run 17_51_102 (incl. taxa 199310, 316385)). Recognising a decrease of the balanced accuracy from 92.3 % (run 15_51_100, same datasets except TN) to 90.6 % (Table 9), all, except one (taxid469008), of the expected nonpathogenic samples of E. coli from the EggNOG test set, were classified correctly.

Table 9 Comparison of the type three dependent results of run 15_51_100 and 17_51_102. The number of positive predictions of E. coli decreased slightly. Whereas in run 15_51_100 all of the eight E. coli strains were tagged “YES”, in run 17_51_102 there is a separation between a positive and a negative fraction. All samples of the K-12 strain were correctly predicted as “NO”. Only 3 out of 8 samples remained being predicted as “YES”.

Taxid name	15_51_100	17_51_102	expected
155864 <i>Escherichia coli</i> O157:H7 str. EDL933 51	YES	YES	YES
199310 <i>Escherichia coli</i> CFT073 77	YES	NO	NO
316385 <i>Escherichia coli</i> str. K-12 substr. DH10B 8	YES	NO	NO
316407 <i>Escherichia coli</i> str. K-12 substr. W3110 167	YES	NO	NO
362663 <i>Escherichia coli</i> 536 48	YES	NO	NO
469008 <i>Escherichia coli</i> BL21(DE3)	YES	YES	NO
481805 <i>Escherichia coli</i> ATCC 8739 44	YES	YES	YES
511145 <i>Escherichia coli</i> str. K-12 substr. MG1655	YES	NO	NO

Although the experiment showed a satisfying result, we did not extend it on the models for type IV and VI, because it would lower the mean balanced accuracy and its standard deviation and additionally influence the overall independency of the approach. This is the reason why, for instance in regard to E. coli K-12 DH10B the results of applying our models and therefore also the output of EffectiveDB show the correct result “NO” in regard to T3SS but the probably incorrect prediction “YES” concerning T4SS and T6SS.

Difficulties in testing the approach

An additional disadvantage of the SVM is that each single instance is absolutely crucial for an optimised model training. Therefore we integrated every well proven, positive sample into our training dataset. Unfortunately this means that there is no independent, well known dataset left to test the created model. This makes it, for obvious reasons, quite difficult to verify the models quality properly. Although, we did a 5-fold cross-validation (1.4.2 Cross-validation) with each of our models, there was no actual testing on unused data.

To gain a hint of the predicting power of our method, we tried to find a way to compare our results to those of other, already existing tools. This cannot be interpreted as a typical testing of an SVM-model. Therefore we would have to know the correct answers of the prediction on beforehand. What we did is to compare our results to the predicted results of others. The results can be found within section Comparison to existing prediction methods.

Advantages of the support vector machine

Being aware of all the challenges when working with an SVM might lead us to scrutinise the method itself. Why should we need an SVM as classification method? This question can be answered by the

fact that the result of this supervised learning approach is a clear YES/NO prediction. Our aim was it to find a method that makes it easy for the user (mostly biologists, not computer scientists!) to handle the gained results. Offering a probability threshold or a number of available proteins is already sophisticatedly provided by other databases such as T346Hunter(43). Unfortunately, such results do not make any statement in regard to functionality or completeness of a secretion system. Showing a clear answer if the trait occurs or not, makes it much easier for the user to handle the results of the prediction.

Furthermore, there is no limitation in regard to the chosen trait. The PICA framework is designed for phenotype prediction in general and, as already shown by Feldbauer et al. (56) the traits can be manifold. The only precondition that needs to be satisfied, is to find a sufficient number of independent training data. Once achieving this, there seems to be practically no limitation.

Another important advantage is the proper calculation of the feature ranking. Using a linear kernel, as we did, makes it quite easy to make conclusions regarding the influence of the contributing features.

Choosing a support vector machine as predictive tool for phenotype expression, gives us the opportunity to provide intuitively understandable results and additional information about the importance of the classifying features.

3.2 EffectiveDB

Bacterial secretion is a crucial factor of pathogenicity within the interaction of pathogen and host. Thus, investigations according both, the secreted proteins, as well as the secretion systems might open new perspectives to researchers. Thereby, bioinformatic approaches, especially those, using machine learning techniques, enlarge the scientists' scope of action by disclosing information, gained from large amounts of data. The challenge we faced, is to bridge the gap between the biological problem and the computer scientific answer. It became our aim to enable biologists to use those computational approaches without any further knowledge. Additionally, it is crucial for us to provide intuitively understandable results that are easy to interpret. Although there are a few very interesting and freely available online tools that deal with the different topics of secretion, we wanted to provide a one-stop-shop to the user. Until the update of EffectiveDB, there was no tool that showed an overview of the secretome, investigated by distinct, each other complementing, applications, the presence and functionality of different types of secretion systems, and the targeting of putative secreted proteins within the eukaryotic host cell. All of these features are combined within our database and thus create a new package of information.

To make our approach freely available for others, we decided to integrate it into <http://www.EffectiveDB.org> (formerly known under "<http://www.effectors.org>"). Until September

2015, this database was specialised to effector secretion prediction within bacterial cells. Adding the PICA approach under the applications name “EffectiveS346” broadened the databases scope in regard to bacterial secretion systems. Additionally, we made some further dataset updates for the already existing methods and also integrated additional prediction features such as EffectiveCCBD, for example. The new EffectiveDB is a comprehensive and intuitively usable database that enables biologists to use the advantages of computer scientific tools without further knowledge.

3.2.1 Database design

As already mentioned, we had to ask ourselves what kind of information is needed by biologists on the one hand and by computer scientists as users, on the other hand. To understand the different needs in a better way, we build our conceptual design based on the following four use cases:

Table 10 Use cases. The four described use cases helped us to verify the needs and requirements of our database.

Use Case	Use genome mode prediction
<i>Actor</i>	Biologist
<i>Description</i>	A biologist uploads his own genome information via the web interface and submits it for predictional processing.
<i>Pre-conditions</i>	The data needs to be in multi-FASTA protein format.
<i>Post-conditions</i>	The user is directed to the results-page of his calculations.
<i>Main Success Scenario</i>	1. The user selects „Job submission“. 2. A form opens and the user uploads his data by selecting “Durchsuchen”. 3. Diverse features can be en/-disabled, “Genome mode” MUST be enabled. 4. User submits the job, which is stored under a unique calculation-ID, for processing 5. The system forwards the submitted job to the grid system.
<i>Extensions</i>	• The user can alternatively upload a precalculated COG-profile to decrease the calculation time considerably. • The user can state his e-mail address to get informed when the job is finished.
<i>Priority</i>	High
<i>Performance Target</i>	Complete full prediction pipeline within hours

Use Case	Use protein mode prediction
<i>Actor</i>	Biologist
<i>Description</i>	A biologist uploads his own protein information (single proteins or groups of proteins) via the web interface and submits it for predictional processing.
<i>Pre-conditions</i>	The data needs to be in (multi-)FASTA protein format.
<i>Post-conditions</i>	The user is directed to the results-page of his calculations.

<i>Main Success Scenario</i>	1. The user selects „Job submission“. 2. A form opens and the user uploads his data by copy-and-paste. 3. Diverse features can be en/-disabled, “Genome mode” MUST be disabled. 4. User submits the job, which is stored under a unique calculation-ID, for processing 5. The system forwards the submitted job to the grid system.
<i>Extensions</i>	The user can state his e-mail address to get informed when the job is finished
<i>Priority</i>	High
<i>Performance Target</i>	Complete full prediction pipeline within minutes

Use Case	Browse database entries
<i>Actor</i>	Method developer
<i>Description</i>	A method developer uses the precalculated and provided data to use it for his purposes in finding new, or improving existing methods in bioinformatics.
<i>Pre-conditions</i>	none
<i>Post-conditions</i>	none
<i>Main Success Scenario</i>	1. The user selects „Browse“ 2. The user selects the genome of interest by text search and follows the link. 3. Sample-specific information is displayed 4. Additional files can be downloaded 5. The obtained data can be saved
<i>Extensions</i>	The user also has the opportunity to download additional data concerning training sets and source code.
<i>Priority</i>	normal
<i>Issues</i>	How to handle inconsistent results?

Use Case	Provide and Maintain the database
<i>Actor</i>	Administrator
<i>Description</i>	The administrator takes care of the currentness of the data and applications. He conducts updates and informs users about them.
<i>Pre-conditions</i>	The administrator must have logged-in his account
<i>Post-conditions</i>	The changes and updates are completed successfully.
<i>Main Success Scenario</i>	1. Do changes regarding static texts 2. Supply database with updated datasets 3. Provide system updates
<i>Extensions</i>	The administrator can also send a newsletter and manage a mailing list. Additionally, the administrator has the possibility to submit a test-job to monitor the automatic deletion of the provided jobs.
<i>Priority</i>	high

Facing those four scenarios, we made the following design decisions:

Java vs. Drupal

When extending the diverse applications and updating the training data sets of the EffectiveDB web-portal, we also considered about the usability, appearance and maintenance of the website.

One of the major adaptations was the switch from JSP (“JavaServer Pages”); a software development technology, created by Sun Microsystems (Oracle Corporation) to Drupal (Drupal is a registered trademark of Dries Buytaert.), an open source content management platform.

This comes along with advantages in regard to the website administration. Whereas Java needs to be compiled to a binary file and afterwards run on the JVM, with Drupal no source code compilation is necessary. This means that static parts of a website, such as paragraphs of text, etc. can be adapted very fast, directly within the source file. Furthermore the web-based administration interface enables the designer to implement a newsletter or mailing list, very easily. Those advantages seem to fulfill the administrative requirements a lot better than the originally used JSP.

Layout and Logo

After the new layout of our divisions website in June 2015, we decided to adapt the new EffectiveDB in the same style. A new logo was created, based on the CUBE-logo. It shows a microbial cell, containing a secretion system on its surface. On top, some secreted proteins are signified (Figure 8).



Figure 8 The new EffectiveDB Logo. It is compared to the former version of the databases logo (source: old webpage www.effectors.org, June, 5th 2015) and pictures an effector secreting, bacterial cell.

Additionally we changed the colouring scheme of the whole webpage. The combination of a medium shade of gray, petrol blue and dark red should create a better readability and higher contrast than the yellowish design of the old version.

Another adaptation was the decision to change the old name “Effective” to “EffectiveDB”. This should allow an easier detection of the webpage on the internet. Additionally, we could obtain the new URL: www.effectivedb.org which reflects the name much better than the old www.effectors.org.

Structure

The new EffectiveDB web-portal consists of three major partitions (Figure 9).

- “Browse”, represents the database section of EffectiveDB. In this part, the results of 1,677 precalculated genomes from EggNOG 4.0 are stored. All of them went through several of our prediction tools. Moreover, there are ongoing additions of genomes, derived from NCBI RefSeq (60).
- “Job submission”, contains the complete predicting pipeline including several offered features. All of our web applications can be obtained exclusively or combined in this section.
- “Methods” shows all the important information about the algorithms and theoretical considerations of the distinct tools.

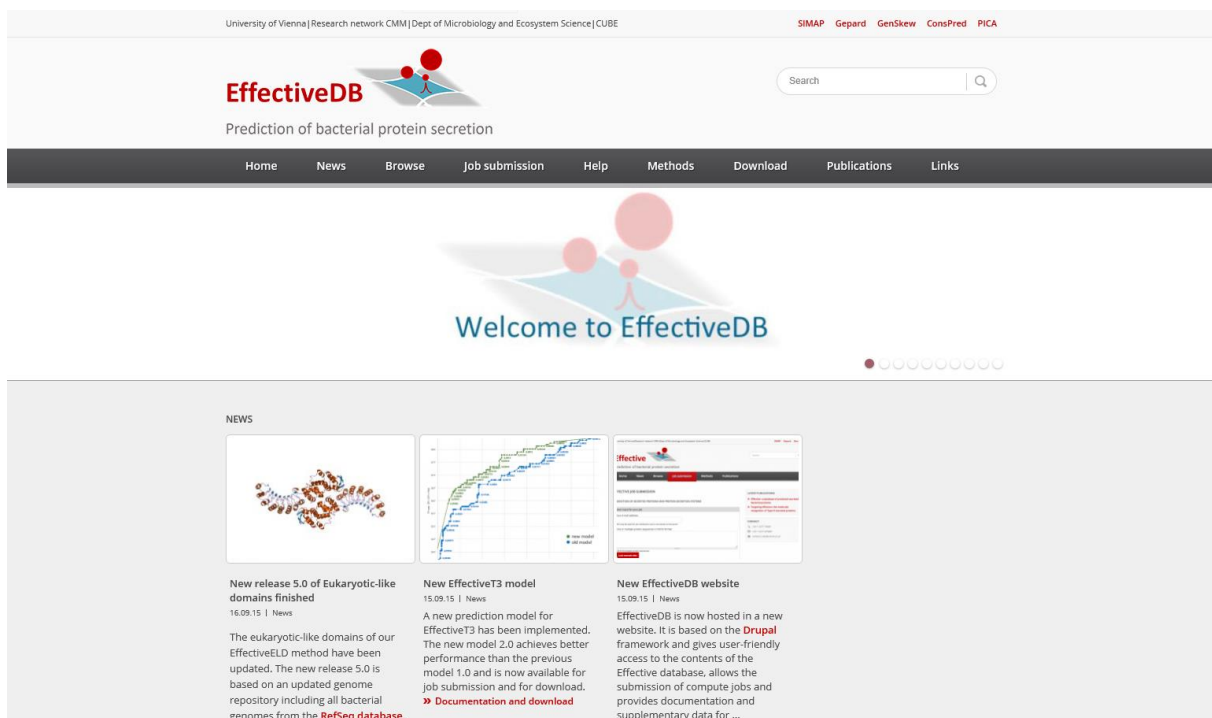


Figure 9 **Homepage of EffectiveDB.** This screenshot shows the structure and design of the web-portal (source: www.effectivedb.org, October 10th 2015).

Moreover, we also provide a download section where the user has access to supplementary information, such as training datasets, models, source code and archived results, respectively, and a “Help” section containing brief tutorials.

3.2.2 Workflow

Figure 10 depicts the overall workflow of the EffectiveDB prediction pipeline. It consists of two major parts, the genome mode, if almost complete genomes are uploaded and the protein mode, as part of the genome mode, which also works for smaller datasets or even single proteins.

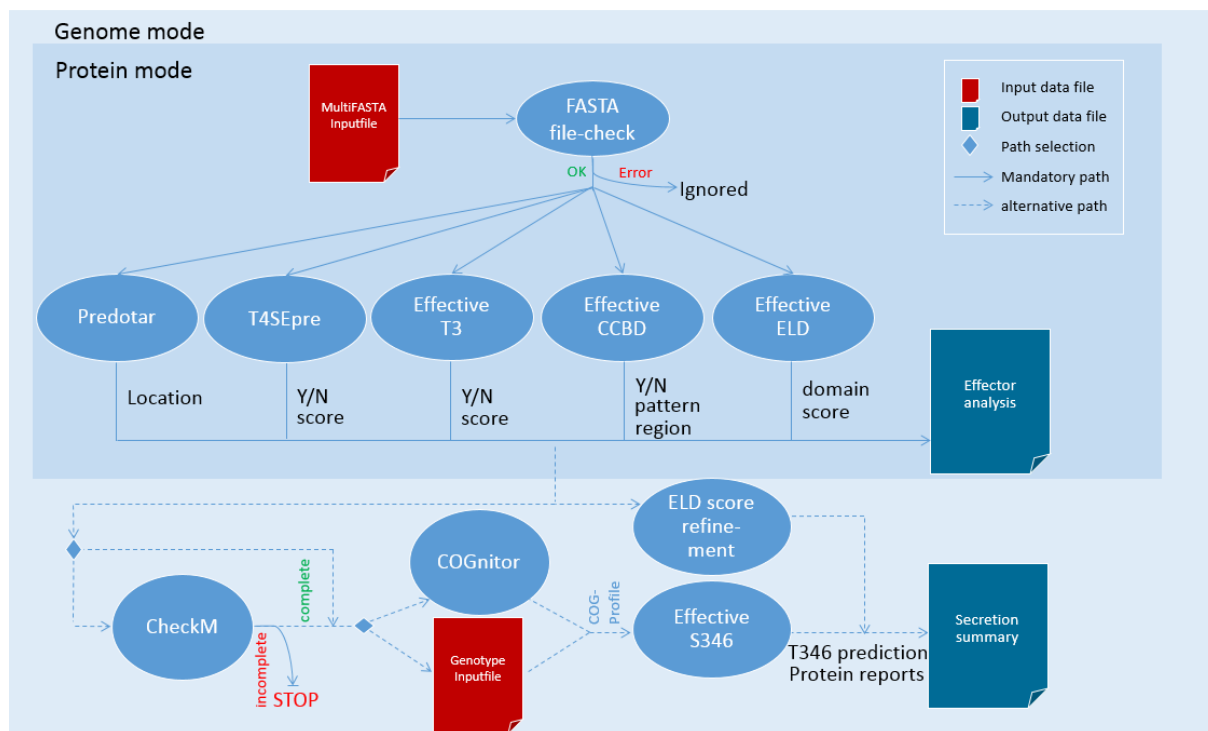


Figure 10 EffectiveDB workflow. The user can upload a multiFASTA file with his set of proteins. Immediately the pipeline will check it for validity in regard to FASTA formatting. Proteins that do not fulfill the requirements will be stored transparently within an “ignored” file. All of the other proteins can be checked with further applications. In the protein mode there are three differently focused programs (EffectiveT3, EffectiveCCBD and EffectiveELD) that will predict putative secreted proteins for type III secretion. Additionally we added an application (T4SEpre) that predicts type IV secretion assigned effectors. If a protein is predicted to be secreted, it is possible to track it within the host-cell by Predotar. Furthermore, users can also upload an (almost) complete genome and enable the genome mode. In this case, beside the applications of the protein mode, the input file firstly will be checked for completeness (CheckM). Afterwards a COG profile will be generated by COGnitor. At last, EffectiveS346 calculates, if the sample contains secretion systems. The ELD score will be then calculated for all putative sequences instead of using only significantly enriched ones.

Components

Table 11 shows the diverse calculations that can be done after submitting a proper genome file. Additionally all of those results are already available for the precalculated strains that can be found in the “Browse” section.

Table 11 Required pipeline preconditions. The table shows the different applications of EffectiveDB and their preconditions

application	T3	T4	T6
gram positive		●	
gram negative	●	●	●
EffectiveS346	●	●	●
EffectiveELD	●	●	●

<i>EffectiveCCBD</i>	●		
<i>EffectiveT3</i>	●		
<i>Predotar</i>	○	○	○
○...only significant if protein is secreted			

The first part of our prediction pipeline is the sample division into gram-positive and gram-negative bacteria, respectively. This can be achieved by a PICA model of Roman Feldbauer (33) that shows a very valuable balanced accuracy of 98.9% and standard deviation of only 1.4%. Primarily this step was not included in our prediction pipeline, but the secretion system prediction was done right at the beginning. The results we got with this strategy were not comprehensible from a biological point of view. It happened several times that T3SS or T6SS were identified within the group of gram-positives, which seems to be impossible from the current state of knowledge. To avoid this kind of misprediction, we included the gram-staining prediction before the secretion system prediction. Considering this, it is possible to predict T4SS within the gram-positives and the gram-negatives and T3SS and T6SS exclusively within the group of gram-negatives.

Only if we are able to identify at least one single type of secretion system, we start further calculations about putative secreted proteins. In this case we have the opportunity to make predictions by four differently focused calculation procedures (*EffectiveT3*, *EffectiveCCBD*, *EffectiveELD*, *T4SEpre*), each described in the following section.

The last part of our extensive calculation is the possibility of tracking putative effectors into the target cell, by using *Predotar*. This tool might help users to analyse and understand the host-bacterium interaction a lot better. For a correct interpretation of the result it is crucial to assume that the protein of interest is secreted, which can be checked by the already mentioned applications. The *Predotar* tool does not give any indication about the possibility of secretion itself.

- Secretion System prediction

- *EffectiveS346* (61)

This novel application opens the user the opportunity to predict functional bacterial secretion systems of the types III, IV and VI. The most remarkable feature is the prediction by binary classification. This is expressed in a YES/NO decision that should unburden the user when being confronted with possible difficulties of the results interpretation. Further details regarding the experimental design or the modelling can be found within 2.1 Model design and 3.1 PICA models, respectively.

- Prediction of putative secreted proteins

- *EffectiveT3* (23)

EffectiveT3 is one of two possible tools for the prediction of putative secreted proteins exclusively in regard to type III secretion. It was also available on the “old” Effective webpage. This method searches for N-terminal signal peptides. It is now updated by the usage of a new, much bigger training data set.

- EffectiveCCBD (24)

This application was designed with the help of the work of Costa et al., 2012. (24). It looks for a specific “conserved chaperone binding domain” consisting of the following sequence: (LMIF)1XXX(IV)5XX(IV)8X(N)10

The underlying idea is that the injection tube of the T3SS is quite narrow and lots of secreted proteins seem to be too big to pass through, without being unfolded. The chaperones of type IB, we are interested in, help such proteins to become unfolded and thus are able to get through the tunnel.

Although this can be a useful tool in regard to find putative secreted proteins, our results show less hits than expected, which leads to the assumption that the sequence might be too specific to identify all putative effector candidates.

- EffectiveELD ((23), Version 5.0)

Opposed to the three other effector predicting methods provided by EffectiveDB, EffectiveELD is independent of the type of the secretion system. Its predictions are only based on the occurrence of eukaryotic-like domains (ELDs), which can be found in eukaryotes and host-associated bacteria.

There is a difference between the application of EffectiveELD in genome- or protein-mode. Regarding a small set of proteins, EffectiveELD only reports those ELDs that show a significant enrichment in one or more of the host-associated genomes that can be found in the genome collection of EffectiveDB. On the other hand, if users upload a whole genome to be investigated, the enrichment score of ELDs in several proteins of the genome, is calculated. This can show new, even unexpected ELDs to be predicted.

- T4SEpre (25)

This prediction tool is specialised on the identification of putative type IV secreted proteins by their C-terminal features. In its original version it contains multiple models which represent sequential and position specific amino acid compositions, motifs, and structure based features. The integration of T4SEpre in EffectiveDB does not contain the latter one for calculation time optimising reasons.

- Effector targeting

- Predotar (62)

Predotar is an external webtool, based on neural networks, we integrated into our database. Although the tool was updated in 2003 for the last time, there is currently no other comparable application to identify putative N-terminal targeting sequences. It has already been suggested that organelles might act as targets of bacterial secreted proteins(63). Thus Predotar offers the user the opportunity to check after gaining information about putative secreted proteins, if they might be located whether in mitochondria, endoplasmic reticulum or plastids of a eukaryote (plant or animal). This might elucidate the specific host-bacterium-interaction a little bit more.

Job submission

The following paragraph should give the reader insights on the actual handling of the website. It also deals with practical instructions of how to submit a valid job on the online submission form.

Parameters and sequence files that are submitted by the user are stored under a unique calculation identifier. The shell script `effective_pipeline.sh` (Suppl.-Script 2) covering all program commands is run on a Sun Grid Engine (SGE) batch queue. If the particular node is busy due to other calculations, those jobs are arrested and restarted after the preferred EffectiveDB-job is finished. The programs for secretion system prediction, effector prediction and target sight identification are run in serial and the results files are displayed on the webpage after all steps are finished. Running the whole genome mode, which includes all predictions might take several ours, whereas the protein mode is finished within a few minutes (Table 12).

Table 12 Pipeline time estimation. The table depicts the time measurement of submitted jobs with different sizes and calculation modes, always using default parameters

<i>E. coli O157:H7 str. EDL933</i>	Sample size	Accession number/ number of proteins	time	note
<i>Protein mode</i>	Single protein	WP_061069282.1	2 minutes	
<i>Protein mode</i>	Whole genome (5.6394Mb)	5530	39 minutes	
<i>Genome mode</i>	Whole genome (5.6394Mb)	5530	12 hours, 55 minutes	Including: • CheckM • Cognitor • EffectiveS346

The first box of the submission form (Figure 11) contains the main input information for the calculations. With EffectiveDB it is possible to have a single multi-FASTA protein file as one and only input data to run all provided predicting methods.

EFFECTIVE JOB SUBMISSION

PREDICTION OF SECRETED PROTEINS AND PROTEIN SECRETION SYSTEMS

Main input for your job

Your e-mail address:

Will only be used for job notification and is not stored on the server

One or multiple protein sequences in FASTA format:

FASTA formatted protein sequences

Load example data

Protein file

Durchsuchen...

Upload a FASTA formatted protein sequence file

Figure 11 **Main input**. The option of stating an e-mail address is useful, especially when enabling the time consuming “genome mode”. The major box in the center should contain the data that is going to be analysed. It can be either a single protein, a group of proteins and/or proteins from a whole genome in multi-FASTA protein format. Source: www.effective-db.org, October 22nd 2015.

Within the workflow of EffectiveDB, there are two major parts (protein mode / genome mode), which can be enabled either separately or combined. It depends on the number of submitted proteins, which of the two modes is appropriate.

If one submits a single protein or a small group of proteins, the protein mode should be used. EffectiveDB is then able to predict whether the uploaded proteins are putatively secreted (EffectiveT3, T4SEpre, EffectiveCCBD, and EffectiveELD). Additionally, if enabled by the user, Predotar, provides information about the target location in the specific host-cell. Each of the prediction methods can be manually en/disabled, according to the interests of the user (Figure 12)

EffectiveCCBD: predicts Type III secreted proteins based on their secretion chaperon binding site

☒ Enabled

Figure 12 **Enabling/Disabling**. Functions like EffectiveCCBD, for instance, can be manually en-/disabled. Source: www.effective-db.org, October 22nd 2015.

The genome mode is disabled by default, due to its long calculation time. If it is enabled, the user must upload the (whole) genomic information of an organism.

Genome mode: for protein sequences from (nearly) complete genomes

☒ **Enabled**
Select if your proteins represent a genome. This will enable Z-score refinement in **EffectiveELD** and allows the activation of **EffectiveS346**.

☐ Run **CheckM** to estimate genome completeness

Optional: COG/NOG assignment file

Durchsuchen...

Alternative to **COGnitor**: upload a file listing all **eggNOG 4.0** COGs/NOGs that are present in your genome (one COG/NOG accession per line; only NOGs on root level - no bactNOGs, gproNOGs etc.). As second column, delimited by tab, the protein name can be provided (optional). **Example file**

☐ Run **EffectiveS346**
Predicts Type III, IV, VI secretion systems for protein sequences from (nearly) complete genomes. If no COG/NOG assignment file is given, orthologous groups will be predicted by using **COGnitor** and **eggNOG 4.0**

Figure 13 **Choosing the genome mode**. It must be manually enabled. It searches for type III, IV and VI bacterial secretion systems within the uploaded genome. Additionally the user can check the input data for genomic completeness, done by CheckM. When enabling the genome mode, automatically the Z-score of EffectiveELD will be refined. Source: www.effectedb.org, October 22nd 2015.

Although it is more time consuming than secretion prediction, also predictions of bacterial secretion systems (EffectiveS346) can be obtained. The genome mode has to be enabled for that purpose. The genome mode assumes that the submitted proteins represent an entire genome, which is as complete as possible. This can optionally be checked by CheckM (64). Although the user has the opportunity to change the minimal threshold of genomic completeness, our recommendation is 85% (Suppl.-Figure 1).

As an additional feature, to reduce the calculation time substantially, there is the option to upload an already prepared genotype-file containing the COG/NOG-profile of the organism of interest (arrow in Figure 13). It is crucial to upload it in the proper format:

- Only COGs/NOGs from EggNOG 4.0 that are present in the interested genome are valid input data
- All COGs/NOGs must be on root level. It is not possible to upload bactNOG, for example.
- Only one single COG/NOG accession per line is allowed.

There can be an optional second column in the genotype-file that contains the protein names. In this case, the possible number of secretion systems can be estimated from the proteins associated to the most relevant COGs/NOGs.

If a valid COG/NOG profile is uploaded, the time consuming COGnitor run is omitted automatically. Thus, no additional disabling is necessary.

3.2.3 Database content description

Overview of predictions

- Bacterial secretion system predictions

The final models (3.1.3 Final models) of the three different bacterial secretion systems were applied on the dataset of all bacterial genomes, which can be found in EggNOG 4.0. The results came up with our expectations. We found 164 T3SS and 266 T6SS within the gram-negative bacteria. Figure 14 does not show any predictions for those two types within the gram-positives. As this group of bacteria does not seem to have the ability of expressing such secretion systems, we excluded them on beforehand. Therefore, we prohibited such cases, by previously predicting gram-negativity itself. This was done by using the same PICA framework and a model, predicting gram-negativity, trained by Roman Feldbauer (33).

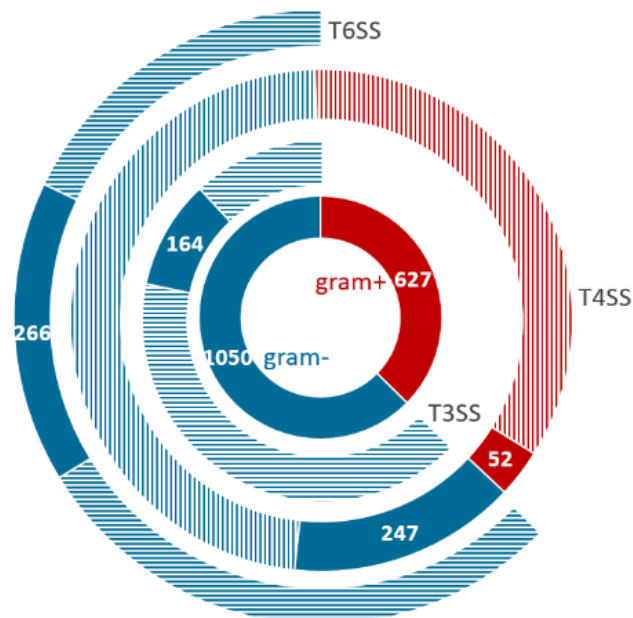


Figure 14 **Identified bacterial secretion systems.** This figure shows the 729 bacterial secretion systems, predicted by PICA models, which were applied on the 1677 bacterial genomes of EggNOG 4.0.

In regard to T4SS we included the gram-positive dataset because of the possible occurrence of type IV subsystems, such as those for conjugation or DNA-uptake and –release (1.2.3 Type IV secretion system). In fact we predicted 52 T4SS within those groups.

- Comparison of the three types of type III effector-prediction

If we compare the three different tools for the prediction of secreted proteins, we recognise that they complement each other due to their different approaches. All three applications were used to analyse the 1,677 different bacterial genomes, contained by EggNOG4.0 (Figure 15).

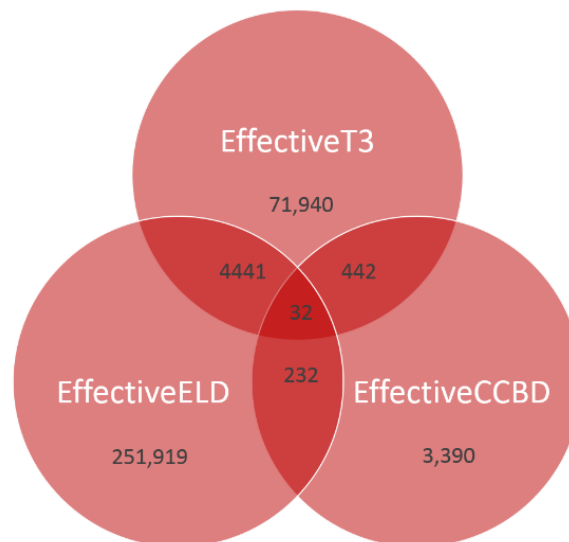


Figure 15 **Comparison of type III effector prediction methods.** The Venn-diagram shows the overlap of the three different methods for effector prediction in type III secreted samples. 32 secreted proteins can be detected by all of them.

The comparison of the different numbers, underlines the suspicion that the pattern search of EffectiveCCBD is highly specific and that the 3,390 hits do not show the entire secretome of the investigated species.

Sample specific predictions for Escherichia coli O157:H7 str. EDL933

To give an insight of how the results can look from the users perspective, the following section deals with E. coli O157/H7 str. EDL933 as an example of use.

BROWSE EFFECTIVEDB

EffectiveDB provides precalculated results for all 1677 bacterial genomes in the **EggNOG 4.0** database and additional 23 bacterial genomes from the NCBI RefSeq database. In the near future we will switch to the upcoming EggNOG 5.0 database, derived from STRING 11 (about 10.000 genomes). RefSeq genomes not contained in STRING will be continuously added. We have used the following settings:

- Genome mode: enabled, genotypes from EggNOG, EffectiveS346 enabled (Type III and VI secretion systems only predicted for gram negative bacteria)
- EffectiveT3: enabled if EffectiveS346 predicted a type III secretion system; model 2.0.1
- EffectiveCCBD: enabled if EffectiveS346 predicted a type III secretion system
- T4SEpre: enabled if EffectiveS346 predicted a type IV secretion system, minimal score=0.5
- EffectiveELD: enabled, minimal Z-score=4
- Predotar: enabled (both animal and plant models are used)

Show entries

Search: ✕

Genome 	T3SS 	T4SS 	T6SS 
Escherichia albertii TW07627	YES	YES	YES
Escherichia coli 536	NO	YES	YES
Escherichia coli ATCC 8739	YES	YES	YES
Escherichia coli BL21(DE3)	YES	YES	YES
Escherichia coli CFT073	NO	YES	YES
Escherichia coli O157:H7 str. EDL933 	YES	YES	YES

Figure 16 Overview *Escherichia coli* O157:H7 str. EDL933. Selection of *Escherichia coli* O157:H7 str. EDL933 within the "Browse" section.

The overview provided by EffectiveDB immediately shows, which among the three different types of secretion systems, is available (Figure 16, arrow). Users have the opportunity to gain deeper insights of the results by following the organisms link. To understand which of the calculations are done, we have a closer look at the input parameters, which are depicted at the very bottom of the results page (Figure 17).

All of the applications of EffectiveDB, including EffectiveS346 from the genome mode are enabled in this particular case, which can be seen by the capital letter T (=TRUE) (lower arrow, Figure 17). In our pre-calculations we used the COG/NOG-profile that was extracted from the EggNOG 4.0 database. All of the application specific parameters are set to their default values. In detail, this means that the results of EffectiveT3 are selectively chosen by a minimal score of 0.9999. In other words, only hits with a very high secretion probability are selected. The results of Predotar are shown with respect to the interaction with animals as hosts (upper arrow, Figure 17). Results for plant-hosts are shown in the

output within brackets "()". The check for completeness, done by CheckM, was disabled, as the genomes of the organisms for our pre-calculations are complete.

PARAMETERS:

```
=====
EFFECTIVE_T3_OLD=selective
EFFECTIVE_T3_NEW=selective
PREDOTAR_ANIMAL_OR_PLANT=a
PROTEINS=INPUT PROTEINS
MODUS=genome
RUN_PREDOTAR=T
RUN_EFFECTIVE_T3=T
RUN_T4SEpre=0.5
RUN_EFFECTIVE_CCBBD=T
RUN_EFFECTIVE_ELD=4
CheckM_COMPLETENESS=
RUN_EffectiveS346=T
GENOTYPE=COG Profile
EMAIL=
=====
```

Figure 17 *Input parameters Escherichia coli O157:H7 str. EDL933*. The figure depicts the default input parameters which were used for the precalculated predictions of *Escherichia coli* O157:H7 str. EDL933. Source: www.effectedb.org, January 4th 2016.

Using the above-described parameters for the calculations, leads to the following results (Figure 18).

EFFECTIVE RESULTS FOR ESCHERICHIA COLI O157:H7 STR. EDL933 (EGGNOG40)

SUMMARY

Number of putative secreted proteins: 956
 - Predicted by **EffectiveT3**: 464 (high confidence), 230 (low confidence)
 - Predicted by **EffectiveCCBD**: 33 within, 23 before, 32 behind expected region 25..70aa
 - Predicted by **T4SEpre**: 101
 - Predicted by **EffectiveELD**: 41 eukaryotic-like domains, contained in 125 proteins (40 high, 85 medium Z-score)
 Predicted protein secretion systems: Type III, Type IV, Type VI

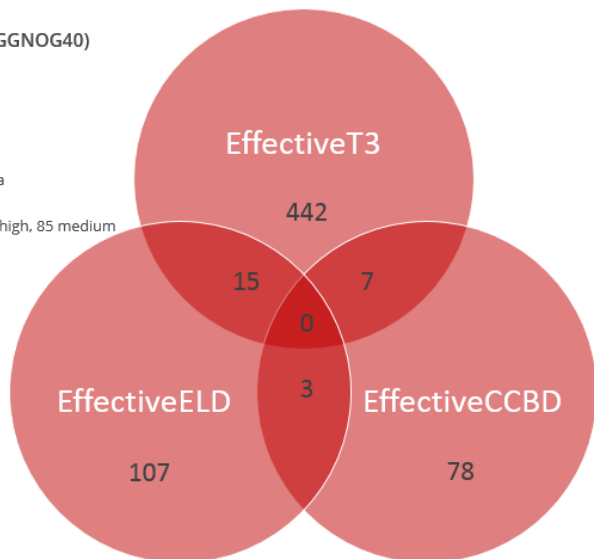


Figure 18 *Summary Escherichia coli O157:H7 str. EDL933*. Combination of the pre-calculated results for *Escherichia coli* O157:H7 str. EDL933. Source: www.effectedb.org, January 4th 2016.

Primarily, the user gets confronted with an overview of the different calculations. The first line shows the overall amount of putative secreted proteins. This is not necessarily the same number as summing up the numbers of all of the effector predicting methods. There might be some proteins that are detected by more than one method. By providing different approaches for effector predictions, we want to find as many putative effectors as possible. One should be aware of the fact that not every secreted protein is detected by all methods!

PROTEIN BASED RESULTS

Putative secreted proteins (weak evidence indicated by parentheses)

Show entries

Protein	EffectiveT3	EffectiveCCBD	T4SEpre	EffectiveELD	Predotar
155864.Z3464	+	+			
155864.Z5788	+	+			
155864.Z1336	+	(<)			(ER)
155864.Z5450	+	(<)			(plastid)
155864.Z2428	+	(<)			
155864.Z0678	+	(<)			
155864.Z1412	+	(>)			
155864.Z2816	+		+		(mitochondrial)
155864.Z5140	+		+		(mitochondrial)
155864.Z5801	+		+		

Showing 1 to 10 of 956 entries

◀ Previous Next ▶

Figure 19 **Protein based results Escherichia coli O157:H7 str. EDL933**. The example of *Escherichia coli* O157:H7 str. EDL933 shows that the protein 155864.Z3464 for example, is predicted to be secreted by EffectiveT3 and EffectiveCCBD. The results are ranked by their significance. Proteins that are predicted to be secreted by more than one method are ranked on top. Source: www.effectedb.org, January 4th 2016.

Figure 19 shows the protein-based results of the EffectiveDB predictions. Again it can be seen that for instance Protein 155864.Z3464 is predicted to be an effector not only by EffectiveT3, but also by EffectiveCCBD. Proteins that are not predicted to be secreted are not contained in the output table.

To further analyse the diverse results of EffectiveDB, the user may also download the combined results as .csv or .html format, listed below the table. Moreover, the specific results of each protein-based prediction can be obtained separately as downloadable files.

Regarding the results of the genome based approach of EffectiveS346, it is firstly summarised if there are one or several intact secretion systems predicted within the genomic sequence, or not (Figure 20).

In this genome, intact Type III, IV and VI secretion systems have been predicted. Otherwise the prediction would be **NO** (not available) or **N.D.** (not defined). The latter might occur within the group of gram-positive bacteria. Those organisms are known to lack the type III or VI secretion systems at all. Therefore, we disabled the respective models for gram-positive bacteria.

GENOME BASED RESULTS

Summary

- Gram negative: **YES** (FDR = 0.5 %)
- Functional Type III secretion system: **YES** (FDR = 12.5 %)
Estimated copy number = 2.2 (22/10 KEGG proteins)
- Functional Type IV secretion system: **YES** (FDR = 7.2 %)
Estimated copy number = 0.0 (0/10 KEGG proteins)
- Functional Type VI secretion system: **YES** (FDR = 7.7 %)
Estimated copy number = 1.25 (10/8 KEGG proteins)

Individual results for each secretion system

- **List of 100 most important COGs/NOGs for prediction of functional Type III secretion system (T3_top100_COGS.xls)**
- **List of 100 most important COGs/NOGs for prediction of functional Type IV secretion system (T4_top100_COGS.xls)**
- **List of 100 most important COGs/NOGs for prediction of functional Type VI secretion system (T6_top100_COGS.xls)**
- **List of COGs assigned to Type III secretion system in KEGG (T3SS_KEGG_cmp.xls)**
- **List of COGs assigned to Type IV secretion system in KEGG (T4SS_KEGG_cmp.xls)**
- **List of COGs assigned to Type VI secretion system in KEGG (T6SS_KEGG_cmp.xls)**

Figure 20 Genome based results *Escherichia coli* O157:H7 str. EDL933. The Output of genome based calculations contain a short summary if the distinct secretion systems are available and diverse download options. Source: www.effective-db.org, January 4th 2016.

Within the brackets, beside the actual prediction, the FDR (=false discovery rate) gives the user an additional indication of the reliability of the result.

The estimated copy number can assign the actual number of structural proteins (as represented in KEGG) to the significant COGs. This can be an indication of how many secretion systems of this type are present in the particular sample.

It is very important to conscientiously further investigate these numbers, as they do not directly belong to our actual prediction. Therefore we recommend to use the database T346Hunter. For genomes that are covered by T346Hunter and EffectiveDB, we provide a direct link to their results at the end of our results page.

Sample specific predictions for Burkholderia pseudomallei K96342

As a second, interesting example from the genome repository of EffectiveDB, *Burkholderia pseudomallei* K96342 should be mentioned. The results of the protein mode can be viewed in Figure 21. We find several putatively type III secreted proteins, detected by EffectiveT3 and EffectiveCCBD. Additionally there are 102 hits of T4SEpre that indicate putative type IV secreted effectors.

EFFECTIVE RESULTS FOR BURKHOLDERIA PSEUDOMALLEI K96243 (EGGNOG40)

SUMMARY

Number of putative secreted proteins: 1157

- Predicted by **EffectiveT3**: 560 (high confidence), 297 (low confidence)
 - Predicted by **EffectiveCCBD**: 22 within, 10 before, 37 behind expected region 25..70aa
 - Predicted by **T4SEpre**: 102
 - Predicted by **EffectiveELD**: 52 eukaryotic-like domains, contained in 177 proteins (13 high, 164 medium Z-score)
- Predicted protein secretion systems (parentheses indicate weak positive predictions lacking more than 50% of core proteins of a secretion system): Type III, (Type IV), Type VI

Figure 21 Summary Burkholderia pseudomallei K96243. This excerpt from the precalculated results for Burkholderia pseudomallei K96243 gives an overview of the protein based prediction results. EffectiveT3 and EffectiveCCBD are specialised to find type III secreted proteins, T4SEpre identified 102 type IV effectors and the results of EffectiveELD are independent of the type of secretion system. Source: www.effectedb.org, January 4th 2016.

Thus, at first, it does not seem to be surprising that the genome-based prediction for T4SS is “YES” Figure 22.

Being aware of that fact, it is even more suspicious that the estimated copy number for the KEGG proteins is 0.0. In other words, there was not a single protein found within the sequence that has a

GENOME BASED RESULTS

Summary

- Gram negative: **YES** (FDR = 0.5 %)
- Functional Type III secretion system: **YES** (FDR = 12.5 %)
Estimated copy number = 2.9 (29/10 KEGG proteins)
- Functional Type IV secretion system: **YES** (FDR = 7.2 %)
Estimated copy number = 0.0 (0/10 KEGG proteins)
- Functional Type VI secretion system: **YES** (FDR = 7.7 %)
Estimated copy number = 3.875 (31/8 KEGG proteins)

Figure 22 Genome based results Burkholderia pseudomallei K96243. Prediction of the presence of all three types of secretion systems. Interestingly, in regard to type IV, we do not have identified a single KEGG Protein although the prediction is “YES”. Source: www.effectedb.org, January 4th 2016.

structure building function as it is suggested by KEGG(6, 59).

To understand these circumstances a bit better, we had an in depth look at the corresponding features and their descriptions (Table 13).

Table 13 Classifying COGs Burkholderia pseudomallei K96243. This table shows the essential features (COGs) for the T4SS prediction. In Burkholderia pseudomallei K96243 the COGs for the ranks number six to nine are present (indicated by “1”).

rank	Conserved Orthologous Group	SVM prediction score	COG present in provided sequence	COG description
1	COG3505	0.0232919560811	0	trag family
2	COG3451	0.022727598531	0	type IV secretion system protein
3	COG2948	0.0192692357999	0	conjugation trbi family protein

4	COG3736	0.0191766768433	0	Type IV secretion system protein VirB8
5	COG3504	0.0190196608164	0	transfer protein TrbG
6	COG2919	0.0184093890533	1	Essential cell division protein. May link together the upstream cell division proteins, which are predominantly cytoplasmic, with the downstream cell division proteins, which are predominantly periplasmic (By similarity)
7	COG1125	0.018117026252	1	(ABC) transporter
8	COG1174	0.0177722193227	1	(ABC) transporter
9	COG1732	0.0177722193227	1	(ABC) transporter
10	COG0630	0.0176452172704	0	type II secretion system protein

The SVM rank file might give an indication, why there is a positive prediction of T4SS, whereas none of the KEGG structure proteins can be found. As already mentioned in chapter 3.1.3 Final models, those structure proteins are not the only features that contribute to the classifying decision. The support vector machine takes a lot more features into account. Especially ABC transporters (ranks #7, #8 and #9), which mediate the protein transport through the cell membrane seem to play a crucial role for our prediction of T4SS. Those three examples, as well as a couple of other crucial proteins are present within the sequence. On the other hand those proteins do not belong to the actual structure proteins mentioned by KEGG. Nevertheless, the evidence for a functional T4SS is quite thin. This result might be false positive. This is also underlined by the results of T346Hunter(43). This database identifies different types of T3SS and T6SS on two different loci, whereas there was no T4SS found on any of them.

This phenomenon leads us to a critical discussion of a putative refinement of our method. We thought about the implementation of weights to the different features. This could be a way of telling the SVM which features are of major importance. For example the structure protein representing COGs could be assigned as crucial features. One can assume that there cannot be any successful secretion, if the structure of the secretion system is not present. Nevertheless, in my opinion, such a restriction of a completely analytical approach as ours, would influence the independency of the results in a manipulating manner. Facing an analytically created result, gives the researcher the opportunity to critically interpret and analyse the data, even if the result seems to be wrong. A weighting of features would create an intransparency of the approach and results that cannot be handled seriously.

What can be an option of improving the approach, is the usage of semi-supervised machine learning techniques. Especially in regard to T4SS we face the problem of a very limited sample space. Adding unlabeled samples to the training datasets might offer new possibilities.

3.3 Comparison to existing prediction methods

As an implicit characteristic of an SVM, the result consists of an explicit binary prediction (YES/NO). There is no numeric factor of predictional accuracy. The only measurable values we can deal with, are the mean balanced accuracy and the standard deviation, obtained from the model creation. Obviously this value is the same for each of the predicted instances.

To our knowledge, EffectiveS346 of EffectiveDB is the only prediction method for bacterial secretion systems that provides such a clear, easily interpretable result. Therefore, it is difficult to compare our results to those of other databases, which provide a threshold or a number of occurring proteins, without any indication of functionality.

Nevertheless we tried to compare at least the overlap of the different predictions with each other, always lacking the information whose prediction is “correct”. It must be said insistently that this section does not try to “test” the created models of EffectiveS346. We do not have any evidence for the in-silico predictions of the other databases, or at least of ours. It should only give an overview of the distinct applications and a hint of the predictional overlap of the different methods.

Additionally, we have to say that our way of predicting secretion systems has limitations in regard to the localisation of the predicted clusters. We neither give any information about the distinct chromosomes (if there are more than one), nor the explicit location of a secretion system within a chromosome. As this kind of prediction was not part of our scope of providing a user-friendly, database that shows an overview of many aspects of secretion, we truly recommend the additional usage of other databases such as T346Hunter. This database, as well as the two SecReT (41, 42) databases, give very good insights into detailed mapping of secretion systems.

3.3.1 SecReT4 and SecReT6

The SecReT4 ((41), Version 1.0) and SecReT6 ((42), Version 2.0) databases contain a broad collection of information concerning the bacterial T4SS and T6SS, respectively. They present a pool of experimentally validated examples from the literature and additionally in-silico predictions found by their approach (3.1.1 Finding sufficient Training Data). Within a “Browse-section” each genome can be directly tracked to the NCBI Taxonomy database (53, 65) via its taxonomy-ID.

Furthermore, a list of putative effectors, if available, is presented. SecReT6 also gives an additional column to indicate putative immunity proteins.

Moreover, the user can find not only the number of found secretion systems but also a label if the specific secretion system is experimentally validated. The results are not only presented tabularly, but can also be found as a genome-map containing the locations of the distinct proteins.

Beside the functions of the lists in “Browse”, users also have the opportunity to “search” directly for effectors, immunity and component proteins or a specific species within the database.

The tools which are used for the detection, search and prediction can be separately used directly within the “Tools-section”. Protein and gene sequence information can be downloaded in FASTA format.

The information of those two databases was also used as input for our training datasets. Only those samples that are marked as “data derived from experimental literature” are used for the model training. All of the other samples, occurring in SecReT4 and SecReT6 are predicted to have at least one bacterial secretion system of the specific type. In other words, the SecReT databases do not make a binary classification decision, but all of the samples that can be found in the databases are predicted to show one or more secretion systems.

If we have a look at the general overlap of the genome repositories of those two databases with ours (Table 14), we see that only a quite small number of genomes is shared. This discrepancy is explained by the underlying data. Whereas EffectiveS346 is based on the genomes of EggNOG 4.0, the SecReT databases use text-mining as their input information (41, 42). Afterwards the putative samples are manually curated.

Concerning the databases overlap, it has to be noted that the 1677 genomes of EffectiveDB contain positive and negative predictions, whereas SecReT4 identifies 444 hits and the SecReT6, 515 hits of only positively predicted samples. Negatively predicted samples are not shown within those two databases. Within the shared genomes between SecReT4 and EffectiveDB, 60% of the predictions for T4SS are found commonly by both detection methods. In regard to T6SS, altogether 74% can be found by both tools. Nevertheless, there are several predictions of SecReT that cannot be found by EffectiveS346 and vice versa.

Table 14 Genome repositories SecReT. Comparison of the databases EffectiveDB, SecReT4 and SecReT6.

	SecReT4	SecReT6
<i>Contained genomes</i>	444	515
<i>Shared genomes</i>	235	203
<i>Shared positive predictions</i>	141	151
<i>Pos. predictions unfound by S346</i>	94	52
<i>Additional pos. predictions by S346</i>	158	115

This might be explained by the different algorithms that are used. In SecReT4 only 30% of 18 core-components must be found to further inspect the genomes by manual curation. In regard to T6SS, SecReT6 expects 5 of 13 COGs for further investigation. It might be the case that an untypical candidate

of the specific secretion system might be found because of the in depth analysis of an expert. Regarding our systematical approach it seems to be quite unlikely that such a sample could be identified as “YES”.

On the other hand, EffectiveS346 reaches a bigger genome repository. Thus there are 273 additional putative secretion systems type IV and VI predicted, which are outside the scope of SecReT. This also means that the predictional overlap which seemed to be quite low at first sight, actually is not so bad. 47% of the type IV predictions and 57% of those concerning type VI, found by EffectiveS346 are also found by the SecReT databases.

3.3.2 T346Hunter

The direct comparison of our database and T346Hunter(43) is much more difficult. Actually it is more or less impossible as the authors of T346Hunter explicitly underline that the final decision, if there is a secretion system or not, completely stays in the users responsibility (43).

Nevertheless, it might be of interest that 405 genomes are shared by both databases (Figure 23). T346Hunter gives a very good insight into the secretion systems of the contained genomes. It separates the information into the results for 2,997 chromosomes and 2,164 plasmids. Predicted clusters can be directly accessed by choosing a species of interest. Users have a direct insight into a genomic map, in which secretion systems of all three types are shown in different colors. Moreover there is a lot of further information regarding the genomic context and the found clusters.

In our opinion, the information provided by T346Hunter complements the results of EffectiveS346 in a reasonable manner. Therefore, we included direct links between the shared genomes of the two databases. Users now have the opportunity to profit from both sources in an optimal way.



Figure 23 **Genome repository T346Hunter.** The overlap of the genome repositories of T346Hunter and EffectiveDB shows 405 shared genomes.

Although, all of the above described databases provide sophisticated and often detailed insights into the topic of secretion systems, EffectiveDB plays a more complementing than competing role. It was our focus to simplify not only the results format but also to enable the users to run all the distinct

prediction tools just by providing a simple multi-FASTA protein file. Of course there must be some restrictions in regard to the resolution of the results. With our method it is not possible to give any information about the location of the secretion system or any mapping of specific proteins to the specific secretion system. Nevertheless, we provide a one-stop shop to the user, where it is possible to gain extensive information of the secretome of the sample. Afterwards, users might know where to look for further information or how to start an in depth investigation.

3.4 Pipeline application on metagenomic data from sponges

Metagenomes, as a currently highly investigated topic in biology, can be an interesting application of our prediction pipeline. In classical biological approaches, samples are chosen due to a specific interesting feature and cultivated for further investigations, regarding possible habitats, for example. In many cases, biologists have a lot of information about their samples a priori. Unfortunately, only a very few species can be cultivated efficiently. On the other hand, metagenomic data confronts scientists with a completely new situation. Usually, the habitat is more or less the only information that is available from metagenomes. There is neither information about the species that live in it, nor about the attributes they have. It also can happen that there are species present in the metagenome, which are already unknown. Having insights on the potential occurrence of pathogens, for example, within the metagenome could be of major importance. Additionally, it can be interesting to somehow make a high throughput analysis with all traits that are already modelled to get an overview of putative sample features.

As one of the first examples of using our approach in context with metagenomic data, the EffectiveDB pipeline was applied on 18 bins of the sponge *Rhopaloeides odroabile* from the Great Barrier Reef. This sponge is an example of a holobiont (a Greek term, which describes the physical association between a eukaryotic host and its integrated biont organisms and their microbiota (66)). Applying our pipeline might provide new information regarding the interactions between the different symbionts in such a stable symbiont community.

Rhopaloeides odroabile specifically hosts very high microbial diversity (67). The 18 most complete bins that were generated from the metagenome of three biological replicates of *Rhopaloeides*, are investigated here. In general, applying the Effective pipeline on metagenomic data, can give insights into the interaction of putative hosts and symbionts. In this particular case we wanted to know, which of the bacterial symbionts (represented by the 18 bins in Table 15) is interacting with the sponge via bacterial secretion.

There are five bins highlighted in dark-red, which did not pass the completeness check with a minimal threshold of 75%. Within the sufficient complete genomes, there is one gram-positive bin (metabat_very-sensitive_12aug2015.23). In this case only T4SS prediction was done. The results of the

secretion system prediction, done by EffectiveS346 did not show any bin to have any functional secretion system of the three types at all. According to that, no additional effector prediction (EffectiveT3, EffectiveCCBD or T4SEpre) was done. Regarding the refined identification of eukaryotic-like domains, as it is done by EffectiveELD, in the genome mode, there are several PFAMs that showed a very high score of 10000. Those PFAMs are listed in the very last column of the table. None of them shows a connection to secretion. In the EffectiveELD database we only find PF01222 as a domain connected to pathogenicity in general.

A specific search after PF09619 “Type III secretion system lipoprotein chaperone (YscW)”, PF06541 “Putative ABC-transporter type IV” and PF14113 “Type VI secretion system (T6SS), amidase effector protein 4”, as examples for typical secretion specific ELDs also remained unsuccessfully.

Finally, we can say that there is no indication for type III, IV or VI secretion within those data. Secretion system prediction, as well as the investigation of eukaryotic-like domains, did not show positive results in any of the bins.

Table 15 Results of EffectivesS346 and EffectiveELD on 18 metagenomic bins. Samples highlighted in dark-red did not pass the completeness check with a minimum threshold of 75%.

Bin's name	Completeness	Contamination	Taxon used	Taxon ID	EffectiveS346	ELD
<i>metabat_bins43.fa</i>	75.64%	2.12%	Trueperaceae	332247	NO	-
<i>metabat_veryensitive_12aug2015.38</i>	95.37%	1.82%	Caldilineaceae	475964	NO	PF10223, PF08925
<i>metabat_superspec_12aug2015.21</i>	94.51%	2.56%	Gemmatimonadetes	142182	NO	PF08498
<i>metabat_veryensitive_12aug2015.21</i>	94.22%	0.85%	Nitrospinae	1293497	NO	PF01222, PF08498, PF01222, PF08498
<i>metabat_veryensitive_12aug2015.6</i>	89.55%	1.51%	Bacteroidetes	976	NO	-
<i>metabat_veryensitive_12aug2015.50</i>	87.58%	4.27%	Bacteria	2	NO	PF00530, PF04106, PF00530
<i>metabat_superspec_12aug2015.48</i>	66.24%	0.91%	Acidobacteria	57723	-	-
<i>metabat_veryensitive_12aug2015.23</i>	84.73%	0.60%	Actinobacteria	1760	NO	PF08498, PF01222, PF08498, PF01222
<i>metabat_veryensitive_12aug2015.56</i>	84.90%	0.00%	Anaerolineales	292629	NO	PF13870
<i>metabat_veryensitive_12aug2015.67</i>	78.66%	2.20%	Gemmatimonadetes	142182	NO	PF13970
<i>metabat_veryensitive_12aug2015.1</i>	78.35%	0.93%	Acidobacteria	57723	NO	PF08498, PF01222, PF01222, PF08498
<i>metabat_superspec_12aug2015.37</i>	73.39%	2.73%	Caldilineaceae	475964	-	-
<i>metabat_bins8.fa</i>	86.49%	4.40%	Gemmatimonadetes	142182	NO	PF07014, PF08498
<i>metabat_superspec_12aug2015.24</i>	89.36%	0.91%	Caldilineaceae	475964	NO	PF08925
<i>metabat_superspec_12aug2015.5</i>	72.91%	5.62%	Deltaproteobacteria	28221	-	-
<i>rhopa123db2_bin_127.fna</i>	74.52%	0.23%	Poribacteria	265317	-	-
<i>metabat-bins61.fa</i>	73.71%	4.09%	Chloroflexi	32061	-	-
<i>metabat_veryensitive_12aug2015.25</i>	96.20%	3.75%	Rhodobacteraceae	31989	NO	PF01222, PF08498

4 Conclusion and Outlook

With the presented work, we showed that it is possible to make a sophisticated phenotype prediction of bacterial secretion systems, only based on the sequence information. Although bacterial secretion systems are highly modular and diverse protein complexes, a phenotype prediction that uses comparative genomics shows a satisfying result. By using the machine learning technique of a classifying support-vector machine, we provide a binary classification decision whether the phenotypic trait, in our case the bacterial secretion systems of the types III, IV and VI, is present or not. Although the algorithm should be improved furthermore, by specifying the training datasets more precisely and most of all enlarging them, the already achieved predictions show an accuracy of at least 90% (standard deviation below 4.5%) in all three cases.

With our method it is possible to not only find new species that were not thought to show the trait of interest. Due to the clear depiction of the weights of the classifying features, we are also able to identify unexpected proteins that contribute to the identification of a particular phenotype.

The updated EffectiveDB provides extensive secretion specific information for more than 1677 genomes, yet. It provides data regarding secreted effector proteins, the absence or presence of three different types of bacterial secretion systems and putative effector target sites within the eukaryotic host cell. The user has the opportunity to analyse and compare the precalculated results or to submit his own sequence data for processing. As secretion specific genome annotation is currently not considered, our database and tools give the opportunity to facilitate the research of host-pathogen-interactions for example.

With our approach it is not only possible to investigate single genomes. The PICA framework has already been tested successfully on metagenomic data of a biogas fermenter by Roman Feldbauer ((33):71), for example. He tested his data for several different phenotypes. Additionally, the whole Effective pipeline was applied on a sponge metagenome, provided by Nicole Webster (67).

The exploding amount of genomic data, passed through lots of methodic improvements during the past years. Thus, it is possible to create (meta-) genomic data with a high quality of almost complete bins that make a sophisticated phenotypic prediction possible. Additionally, our method can be the basis for a high-throughput analysis of newly generated sequence data. It does no longer need to be an idealistic dream of the future, to automatically analyse uncovered data for phenotypic traits. This is also not limited to bacterial secretion systems. As Roman Feldbauer already showed in his work (33), manifold phenotypes can be investigated by our approach. The only requirement is to find data of a sufficient number and quality to train the model on beforehand. Afterwards, the increase of data makes it easy to improve the training datasets furthermore.

To not only hold the current state of accuracy, but also increase the quality furthermore, it is essential to maintain and thus also update the databases content, frequently. As it is already started, it clearly makes sense to go on with integrating more and more genomes from RefSeq database (55), for the section of the precalculations. Otherwise, it would be an avoidable limitation of our databases scope to only use the currently quite limited number of genomes from EggNOG 4.0 (38). Secondly, I would insistently recommend to frequently update and enlarge the training datasets of the distinct models. As research from the field of secretion (or any other trait) moves on and provides new data, the number of genomes from the training dataset can be enlarged and even refined. At last, there is the possibility of integrating new information about putative structure proteins within an updated version. On the one hand it can be illuminating if a protein of currently unknown function is understood in a better way and can be taken into account when analysing the feature rank file. On the other hand it might make sense to integrate some kind of weight to specific features, if they are known as crucial predicting features. Summarising the manifold options of updating and enlarging this approach, a frequent maintenance appears to be obvious to preserve the predictional power and scientific relevance.

Bibliography

1. Tseng,T.-T., Tyler,B.M. and Setubal,J.C. (2009) Protein secretion systems in bacterial-host associations, and their description in the Gene Ontology. *BMC Microbiol.*, **9 Suppl 1**, S2.
2. Gawarzewski,I., Smits,S.H.J., Schmitt,L. and Jose,J. (2013) Structural comparison of the transport units of type V secretion systems. *Biol. Chem.*, **394**, 1385–98.
3. Costa,T.R.D., Felisberto-Rodrigues,C., Meir,A., Prevost,M.S., Redzej,A., Trokter,M. and Waksman,G. (2015) Secretion systems in Gram-negative bacteria: structural and mechanistic insights. *Nat. Rev. Microbiol.*, **13**, 343–59.
4. Schlegel,H.-G., Eitinger,T., Fuchs,G., Heider,J., Kemper,B., Kothe,E., Schink,B., Schneider,E. and Unden,G. (2006) Allgemeine Mikrobiologie 8. Auflage. Fuchs,G. (ed) Georg Thieme Verlag, Stuttgart.
5. Rêgo,A.T., Chandran,V. and Waksman,G. (2010) Two-step and one-step secretion mechanisms in Gram-negative bacteria: contrasting the type IV secretion system and the chaperone-usheer pathway of pilus biogenesis. *Biochem. J.*, **425**, 475–88.
6. Kanehisa,M. and Goto,S. (2000) KEGG: kyoto encyclopedia of genes and genomes. *Nucleic Acids Res.*, **28**, 27–30.
7. Coulthurst,S.J. The Type VI secretion system - a widespread and versatile cell targeting system. *Res. Microbiol.*, **164**, 640–54.
8. Cornelis,G.R. (2006) The type III secretion injectisome. *Nat. Rev. Microbiol.*, **4**, 811–25.
9. Abby,S.S. and Rocha,E.P.C. (2012) The non-flagellar type III secretion system evolved from the bacterial flagellum and diversified into host-cell adapted systems. *PLoS Genet.*, **8**, e1002983.
10. Galán,J.E., Lara-Tejero,M., Marlovits,T.C. and Wagner,S. (2014) Bacterial type III secretion systems: specialized nanomachines for protein delivery into target cells. *Annu. Rev. Microbiol.*, **68**, 415–38.
11. Christie,P.J. and Vogel,J.P. (2000) Bacterial type IV secretion: conjugation systems adapted to deliver effector molecules to host cells. *Trends Microbiol.*, **8**, 354–360.
12. Cascales,E. and Christie,P.J. (2003) The versatile bacterial type IV secretion systems. *Nat. Rev. Microbiol.*, **1**, 137–49.
13. Lipps, Georg (Institute of Biochemistry,U. of B. (2006) Plasmids and viruses of the thermoacidophilic crenarchaeote *Sulfolobus*. - PubMed - NCBI.
14. Waters,V.L. (2001) Conjugation between bacterial and mammalian cells. *Nat. Genet.*, **29**, 375–6.
15. de la Cruz,F. and Davies,J. (2000) Horizontal gene transfer and the origin of species: lessons from bacteria. - PubMed - NCBI. *Trends Microbiol.*
16. Ghigo,J.-M. (2001) Natural conjugative plasmids induce bacterial biofilm development. *Nature*, **412**, 442–445.
17. Records,A.R. (2011) The type VI secretion system: a multipurpose delivery system with a phage-

- like machinery. *Mol. Plant. Microbe. Interact.*, **24**, 751–7.
18. Cascales,E. (2008) The type VI secretion toolkit. *EMBO Rep.*, **9**, 735–41.
19. Bingle,L.E., Bailey,C.M. and Pallen,M.J. (2008) Type VI secretion: a beginner’s guide. *Curr. Opin. Microbiol.*, **11**, 3–8.
20. Black,M., Moolhuijzen,P., Chapman,B., Barrero,R., Howieson,J., Hungria,M. and Bellgard,M. (2012) The genetics of symbiotic nitrogen fixation: comparative genomics of 14 rhizobia strains by resolution of protein clusters. *Genes (Basel)*, **3**, 138–66.
21. Rodrigues,C.D. and Enninga,J. (2010) The ‘when and whereabouts’ of injected pathogen effectors. *Nat. Methods*, **7**, 267–9.
22. Diacovich,L. and Gorvel,J.-P. (2010) Bacterial manipulation of innate immunity to promote infection. *Nat. Rev. Microbiol.*, **8**, 117–128.
23. Jehl,M.-A., Arnold,R. and Rattei,T. (2011) Effective--a database of predicted secreted bacterial proteins. *Nucleic Acids Res.*, **39**, D591–5.
24. Costa,S.C.P., Schmitz,A.M., Jahufar,F.F., Boyd,J.D., Cho,M.Y., Glicksman,M.A. and Lesser,C.F. (2012) A new means to identify type 3 secreted effectors: functionally interchangeable class IB chaperones recognize a conserved sequence. *MBio*, **3**.
25. Wang,Y., Wei,X., Bao,H. and Liu,S.-L. (2014) Prediction of bacterial type IV secreted effectors by C-terminal features. *BMC Genomics*, **15**, 50.
26. Ponting,C.P., Aravind,L., Schultz,J., Bork,P. and Koonin,E. V (1999) Eukaryotic signalling domain homologues in archaea and bacteria. Ancient ancestry and horizontal gene transfer. *J. Mol. Biol.*, **289**, 729–45.
27. Handelsman,J., Rondon,M.R., Brady,S.F., Clardy,J. and Goodman,R.M. (1998) Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products. *Chem. Biol.*, **5**, R245–9.
28. Hugenholtz,P., Goebel,B.M. and Pace,N.R. (1998) Impact of culture-independent studies on the emerging phylogenetic view of bacterial diversity. *J. Bacteriol.*, **180**, 4765–74.
29. Burton,J.N., Liachko,I., Dunham,M.J. and Shendure,J. (2014) Species-Level Deconvolution of Metagenome Assemblies with Hi-C-Based Contact Probability Maps. *G3: Genes/Genomes/Genetics*, **4**, 1339–1346.
30. Kumar,S., Krishnani,K.K., Bhushan,B. and Brahmane,M.P. (2015) Metagenomics: Retrospect and Prospects in High Throughput Age. *Biotechnol. Res. Int.*, **2015**, 121735.
31. Witten,I.H., Frank,E. and Hall,M.A. (2011) Data mining: practical machine learning tools and techniques 3rd ed. Morgan Kaufmann Publishers.
32. Bishop,C.M. (2007) Pattern Recognition and Machine Learning 1st editio. Springer.
33. Feldbauer,R.V. (2015) Predicting microbial genotype-phenotype relations with machine learning techniques.
34. Tatusov,R.L., Galperin,M.Y., Natale,D.A. and Koonin,E. V (2000) The COG database: a tool for

- genome-scale analysis of protein functions and evolution. *Nucleic Acids Res.*, **28**, 33–6.
35. Tatusov, R.L., Koonin, E. V and Lipman, D.J. (1997) A genomic perspective on protein families. *Science*, **278**, 631–7.
 36. Forterre, P. (1999) Displacement of cellular proteins by functional analogues from plasmids or viruses could explain puzzling phylogenies of many DNA informational proteins. *Mol. Microbiol.*, **33**, 457–65.
 37. Koonin, E. V and Galperin, M.Y. (2003) *Sequence - Evolution - Function* Kluwer Academic.
 38. Powell, S., Forslund, K., Szklarczyk, D., Trachana, K., Roth, A., Huerta-Cepas, J., Gabaldón, T., Rattei, T., Creevey, C., Kuhn, M., *et al.* (2014) eggNOG v4.0: nested orthology inference across 3686 organisms. *Nucleic Acids Res.*, **42**, D231–9.
 39. Jensen, L.J., Julien, P., Kuhn, M., von Mering, C., Muller, J., Doerks, T. and Bork, P. (2007) eggNOG: automated construction and annotation of orthologous groups of genes. *Nucleic Acids Res.*, **36**, D250–D254.
 40. Finn, R.D., Bateman, A., Clements, J., Coghill, P., Eberhardt, R.Y., Eddy, S.R., Heger, A., Hetherington, K., Holm, L., Mistry, J., *et al.* (2014) Pfam: the protein families database. *Nucleic Acids Res.*, **42**, D222–30.
 41. Bi, D., Liu, L., Tai, C., Deng, Z., Rajakumar, K. and Ou, H.-Y. (2013) SecReT4: a web-based bacterial type IV secretion system resource. *Nucleic Acids Res.*, **41**, D660–5.
 42. Li, J., Yao, Y., Xu, H.H., Hao, L., Deng, Z., Rajakumar, K. and Ou, H.-Y. (2015) SecReT6: a web-based resource for type VI secretion systems found in bacteria. *Environ. Microbiol.*, **17**, 2196–202.
 43. Martínez-García, P.M., Ramos, C. and Rodríguez-Palenzuela, P. (2015) T346Hunter: a novel web-based tool for the prediction of type III, type IV and type VI secretion systems in bacterial genomes. *PLoS One*, **10**, e0119317.
 44. Dong, X., Lu, X. and Zhang, Z. (2015) BEAN 2.0: an integrated web resource for the identification and functional analysis of type III secreted effectors. *Database (Oxford)*, **2015**, bav064.
 45. Tay, D.M.M., Govindarajan, K.R., Khan, A.M., Ong, T.Y.R., Samad, H.M., Soh, W.W., Tong, M., Zhang, F. and Tan, T.W. (2010) T3SEdb: data warehousing of virulence effectors secreted by the bacterial Type III Secretion System. *BMC Bioinformatics*, **11 Suppl 7**, S4.
 46. Jehl, M.-A., Arnold, R. and Rattei, T. (2011) Effective--a database of predicted secreted bacterial proteins. *Nucleic Acids Res.*, **39**, D591–5.
 47. Zou, L., Nan, C. and Hu, F. (2013) Accurate prediction of bacterial type IV secreted effectors using amino acid composition and PSSM profiles. *Bioinformatics*, **29**, 3135–42.
 48. Wattam, A.R., Abraham, D., Dalay, O., Disz, T.L., Driscoll, T., Gabbard, J.L., Gillespie, J.J., Gough, R., Hix, D., Kenyon, R., *et al.* (2014) PATRIC, the bacterial bioinformatics database and analysis resource. *Nucleic Acids Res.*, **42**, D581–91.
 49. Cochrane, G., Karsch-Mizrachi, I., Takagi, T. and Sequence Database Collaboration, I.N. (2015) The International Nucleotide Sequence Database Collaboration. *Nucleic Acids Res.*, **44**, D48–50.

50. MacDonald,N.J. and Beiko,R.G. (2010) Efficient learning of microbial genotype-phenotype association rules. *Bioinformatics*, **26**, 1834–40.
51. Overbeek,R., Begley,T., Butler,R.M., Choudhuri,J. V, Chuang,H.-Y., Cohoon,M., de Crécy-Lagard,V., Diaz,N., Disz,T., Edwards,R., *et al.* (2005) The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes. *Nucleic Acids Res.*, **33**, 5691–702.
52. Souza,R.C., del Rosario Quispe Saji,G., Costa,M.O.C., Netto,D.S., Lima,N.C.B., Klein,C.C., Vasconcelos,A.T.R. and Nicolás,M.F. (2012) AtlasT4SS: a curated database for type IV secretion systems. *BMC Microbiol.*, **12**, 172.
53. Sayers,E.W., Barrett,T., Benson,D.A., Bryant,S.H., Canese,K., Chetvernin,V., Church,D.M., DiCuccio,M., Edgar,R., Federhen,S., *et al.* (2009) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res.*, **37**, D5–15.
54. Jones,P., Binns,D., Chang,H.-Y., Fraser,M., Li,W., McAnulla,C., McWilliam,H., Maslen,J., Mitchell,A., Nuka,G., *et al.* (2014) InterProScan 5: genome-scale protein function classification. *Bioinformatics*, **30**, 1236–40.
55. Wheeler,D.L., Barrett,T., Benson,D.A., Bryant,S.H., Canese,K., Church,D.M., DiCuccio,M., Edgar,R., Federhen,S., Helmberg,W., *et al.* (2005) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res.*, **33**, D39–45.
56. Feldbauer,R., Schulz,F., Horn,M. and Rattei,T. (2015) Prediction of microbial phenotypes based on comparative genomics. *BMC Bioinformatics*, **16**, S1.
57. Muller,J., Szklarczyk,D., Julien,P., Letunic,I., Roth,A., Kuhn,M., Powell,S., von Mering,C., Doerks,T., Jensen,L.J., *et al.* (2010) eggNOG v2.0: extending the evolutionary genealogy of genes with enhanced non-supervised orthologous groups, species and functional annotations. *Nucleic Acids Res.*, **38**, D190–5.
58. Altenhoff,A.M., Škunca,N., Glover,N., Train,C.-M., Sueki,A., Piližota,I., Gori,K., Tomiczek,B., Müller,S., Redestig,H., *et al.* (2015) The OMA orthology database in 2015: function predictions, better plant support, synteny view and other improvements. *Nucleic Acids Res.*, **43**, D240–9.
59. Kanehisa,M., Goto,S., Sato,Y., Kawashima,M., Furumichi,M. and Tanabe,M. (2014) Data, information, knowledge and principle: back to metabolism in KEGG. *Nucleic Acids Res.*, **42**, D199–205.
60. McEntyre,J. and Ostell,J. (2002) The NCBI Handbook.
61. Eichinger,V., Nussbaumer,T., Platzer,A., Jehl,M.-A., Arnold,R. and Rattei,T. (2015) EffectiveDB- updates and novel features for a better annotation of bacterial secreted proteins and Type III, IV, VI secretion systems. *Nucleic Acids Res.*, 10.1093/nar/gkv1269.
62. Small,I., Peeters,N., Legeai,F. and Lurin,C. (2004) Predotar: A tool for rapidly screening proteomes for N-terminal targeting sequences. *Proteomics*, **4**, 1581–90.
63. Lobet,E., Letesson,J.-J. and Arnould,T. (2015) Mitochondria: a target for bacteria. *Biochem. Pharmacol.*, **94**, 173–85.
64. Parks,D.H., Imelfort,M., Skennerton,C.T., Hugenholtz,P. and Tyson,G.W. (2015) CheckM: assessing

- the quality of microbial genomes recovered from isolates, single cells, and metagenomes. *Genome Res.*, **25**, 1043–1055.
65. Benson,D.A., Karsch-Mizrachi,I., Lipman,D.J., Ostell,J. and Sayers,E.W. (2009) GenBank. *Nucleic Acids Res.*, **37**, D26–31.
 66. Guerrero,R., Margulis,L. and Berlanga,M. (2013) Symbiogenesis: the holobiont as a unit of evolution. *Int. Microbiol.*, **16**, 133–43.
 67. Webster,N.S., Taylor,M.W., Behnam,F., Lückner,S., Rattei,T., Whalan,S., Horn,M. and Wagner,M. (2010) Deep sequencing reveals exceptional diversity and modes of transmission for bacterial sponge symbionts. *Environ. Microbiol.*, **12**, 2070–82.
 68. Bulir,D.C., Waltho,D.A., Stone,C.B., Liang,S., Chiang,C.K.W., Mwawasi,K.A., Nelson,J.C., Zhang,S.W., Mihalco,S.P., Scinocca,Z.C., *et al.* (2015) Chlamydia Outer Protein (Cop) B from Chlamydia pneumoniae possesses characteristic features of a type III secretion (T3S) translocator protein. *BMC Microbiol.*, **15**, 163.
 69. Voigt,A., Schöfl,G. and Saluz,H.P. (2012) The Chlamydia psittaci genome: a comparative analysis of intracellular pathogens. *PLoS One*, **7**, e35097.
 70. Horn,M., Collingro,A., Schmitz-Esser,S., Beier,C.L., Purkhold,U., Fartmann,B., Brandt,P., Nyakatura,G.J., Droege,M., Frishman,D., *et al.* (2004) Illuminating the evolutionary history of chlamydiae. *Science*, **304**, 728–30.
 71. Collingro,A., Tischler,P., Weinmaier,T., Penz,T., Heinz,E., Brunham,R.C., Read,T.D., Bavoil,P.M., Sachse,K., Kahane,S., *et al.* (2011) Unity in variety--the pan-genome of the Chlamydiae. *Mol. Biol. Evol.*, **28**, 3253–70.
 72. de Barsy,M. and Greub,G. (2013) Waddlia chondrophila: from biology to pathogenicity. *Microbes Infect.*, **15**, 1033–41.
 73. Pilhofer,M., Aistleitner,K., Ladinsky,M.S., König,L., Horn,M. and Jensen,G.J. (2014) Architecture and host interface of environmental chlamydiae revealed by electron cryotomography. *Environ. Microbiol.*, **16**, 417–29.
 74. Peters,J., Wilson,D.P., Myers,G., Timms,P. and Bavoil,P.M. (2007) Type III secretion à la Chlamydia. *Trends Microbiol.*, **15**, 241–51.
 75. Ondov,B.D., Bergman,N.H. and Phillippy,A.M. (2011) Interactive metagenomic visualization in a Web browser. *BMC Bioinformatics*, **12**, 385.

Supplement

Training datasets

Suppl.-Table 1 Training datasets of all three types.

TAXID	T3SS	SOURCE	TAXID	T4SS	SOURCE	TAXID	T6SS	SOURCE
1016998	YES	T3SEdb	163164	YES	SecReT4	155864	YES	SEED
115713	YES	(68)	190486	YES	SecReT4	160488	YES	SEED
155864	YES	SEED	205920	YES	SecReT4	176299	YES	SecReT6
182082	YES	SEED	208963	YES	SecReT4	177416	YES	SecReT6
187410	YES	SEED	211110	YES	SecReT4	187272	YES	SEED
190485	YES	SEED	212042	YES	SecReT4	187410	YES	SEED
190486	YES	SEED	224308	YES	SecReT4	190486	YES	SEED
198214	YES	T3SEdb	224914	YES	SecReT4	199310	YES	SecReT6
205918	YES	SEED	227377	YES	SecReT4	203122	YES	SEED
208963	YES	SEED	234826	YES	SecReT4	205918	YES	SecReT6
208964	YES	SEED	257313	YES	SecReT4	205922	YES	SEED
209261	YES	T3SEdb	272624	YES	SecReT4	208964	YES	SEED
214092	YES	SEED	283165	YES	SecReT4	214092	YES	SecReT6
216591	YES	SEED	283166	YES	SecReT4	216591	YES	SecReT6
216595	YES	SEED	297245	YES	SEED	216595	YES	SEED
218491	YES	SEED	297246	YES	SecReT4	216596	YES	SEED
218497	YES	(69)	314565	YES	SecReT4	216895	YES	SEED
220341	YES	SEED	340100	YES	SecReT4	218491	YES	SEED
223283	YES	SEED	340184	YES	SEED	220341	YES	SEED
223926	YES	SEED	351745	YES	SecReT4	220664	YES	SEED
224911	YES	SEED	382640	YES	SecReT4	223283	YES	SecReT6
227941	YES	(69)	391295	YES	SecReT4	223926	YES	SecReT6
243160	YES	SEED	400673	YES	SecReT4	224911	YES	SEED
243265	YES	SEED	430066	YES	SecReT4	234267	YES	SEED
243365	YES	SEED	434131	YES	SecReT4	235279	YES	SecReT6
257310	YES	SEED	434922	YES	SecReT4	243090	YES	SEED
257313	YES	SEED	434923	YES	SecReT4	243160	YES	SecReT6
264201	YES	(70)	434924	YES	SecReT4	243231	YES	SEED
264202	YES	(69)	454169	YES	SEED	243265	YES	SEED
264730	YES	SEED	484021	YES	SecReT4	243277	YES	SecReT6

266835	YES	SEED	512562	YES	SecReT4	243365	YES	SEED
267608	YES	SEED	529507	YES	SecReT4	246197	YES	SecReT6
269482	YES	SEED	563041	YES	SecReT4	252305	YES	SEED
272560	YES	SEED	570508	YES	SecReT4	257310	YES	SecReT6
295319	YES	SEED	66084	YES	SecReT4	264198	YES	SEED
300267	YES	T3SEdb	661367	YES	SecReT4	264730	YES	SEED
300268	YES	T3SEdb	85962	YES	SecReT4	266264	YES	SEED
300269	YES	T3SEdb	85963	YES	SecReT4	266835	YES	SEED
314565	YES	SEED	160492	YES	AtlasT4SS	267608	YES	SecReT6
316273	YES	SEED	176299	YES	AtlasT4SS	269482	YES	SEED
321314	YES	SEED	190485	YES	AtlasT4SS	269799	YES	SEED
325240	YES	SEED	205918	YES	AtlasT4SS	272559	YES	SecReT6
331113	YES	(71)	216596	YES	AtlasT4SS	272560	YES	SecReT6
331636	YES	(69)	223283	YES	AtlasT4SS	272620	YES	SEED
338187	YES	SEED	224911	YES	AtlasT4SS	272943	YES	SEED
339670	YES	SEED	266264	YES	AtlasT4SS	290398	YES	SEED
342109	YES	T3SEdb	266265	YES	AtlasT4SS	298386	YES	SEED
343509	YES	SEED	266779	YES	AtlasT4SS	306254	YES	SEED
349521	YES	SEED	266834	YES	AtlasT4SS	312309	YES	SEED
349746	YES	T3SEdb	266835	YES	AtlasT4SS	314230	YES	SEED
350701	YES	SEED	272560	YES	AtlasT4SS	314262	YES	SEED
363253	YES	SEED	279238	YES	AtlasT4SS	314264	YES	SEED
382245	YES	SEED	288000	YES	AtlasT4SS	314278	YES	SEED
391774	YES	SEED	323097	YES	AtlasT4SS	314282	YES	SEED
393305	YES	T3SEdb	347834	YES	AtlasT4SS	314283	YES	SEED
394	YES	T3SEdb	357244	YES	AtlasT4SS	314292	YES	SEED
397945	YES	SEED	391896	YES	AtlasT4SS	316273	YES	SEED
399739	YES	SEED	400668	NO		316275	YES	SEED
423368	YES	T3SEdb	281689	NO		318167	YES	SEED
439843	YES	T3SEdb	484018	NO		318586	YES	SEED
454166	YES	T3SEdb	262724	NO		338187	YES	SEED
454169	YES	T3SEdb	391595	NO		338963	YES	SEED
471472	YES	T3SEdb	563192	NO		339670	YES	SEED
471473	YES	T3SEdb	504474	NO		345073	YES	SEED
476213	YES	T3SEdb	408672	NO		349163	YES	SEED

481805	YES	T3SEdb	331636	NO	349521	YES	SEED
498217	YES	T3SEdb	762903	NO	350701	YES	SEED
502347	YES	T3SEdb	755731	NO	351348	YES	SEED
502801	YES	T3SEdb	221109	NO	351746	YES	SEED
521000	YES	T3SEdb	561230	NO	360104	YES	SEED
544404	YES	T3SEdb	515620	NO	362663	YES	SEED
550537	YES	T3SEdb	269796	NO	376619	YES	SecReT6
550538	YES	T3SEdb	396513	NO	376686	YES	SecReT6
553480	YES	T3SEdb	365044	NO	380703	YES	SEED
554290	YES	T3SEdb	665029	NO	382245	YES	SEED
568708	YES	T3SEdb	314275	NO	384676	YES	SEED
573235	YES	T3SEdb	596151	NO	390235	YES	SEED
574521	YES	T3SEdb	875328	NO	392500	YES	SEED
580049	YES	T3SEdb	568817	NO	395019	YES	SEED
585395	YES	T3SEdb	525254	NO	395495	YES	SEED
585396	YES	T3SEdb	62977	NO	397945	YES	SecReT6
588858	YES	T3SEdb	498217	NO	398578	YES	SEED
634464	YES	T3SEdb	880072	NO	399739	YES	SEED
637382	YES	T3SEdb	535289	NO	399741	YES	SEED
637385	YES	T3SEdb	525146	NO	400668	YES	SEED
637910	YES	T3SEdb	1111676	NO	438753	YES	SEED
672161	YES	T3SEdb	596324	NO	448385	YES	SEED
716544	YES	(72)	745277	NO	498217	YES	SecReT6
765952	YES	(73)	393480	NO	529507	YES	SecReT6
882	YES	SEED	37692	NO	575788	YES	SEED
883	YES	SEED	314256	NO	62928	YES	SecReT6
887712	YES	(74)	411902	NO	62977	YES	SecReT6
99287	YES	SEED	445974	NO	634503	YES	SecReT6
199310	NO		634956	NO	716541	YES	SecReT6
316385	NO		317655	NO	76869	YES	SEED
688270	NO		398767	NO	857087	YES	SecReT6
272630	NO		207559	NO	99287	YES	SEED
391008	NO		641112	NO	150340	YES	SEED
298386	NO		272943	NO	196600	YES	SEED
404589	NO		388413	NO	208963	YES	SecReT6

575614	NO	263358	NO	209261	YES	SEED
351348	NO	290402	NO	216592	YES	SecReT6
556263	NO	340099	NO	216597	YES	SecReT6
517722	NO	706434	NO	216599	YES	SEED
696125	NO	391589	NO	229193	YES	SEED
395494	NO	349520	NO	269483	YES	SEED
983544	NO	718255	NO	271848	YES	SecReT6
1005048	NO	715225	NO	272563	YES	SEED
240016	NO	330214	NO	273123	YES	SEED
224324	NO	688269	NO	273526	YES	SecReT6
742821	NO	713887	NO	291331	YES	SEED
398580	NO	1074889	NO	295319	YES	SEED
59374	NO	547042	NO	300269	YES	SEED
865938	NO	871585	NO	320372	YES	SEED
259536	NO	702438	NO	320373	YES	SEED
679190	NO	376686	NO	320388	YES	SEED
1008459	NO	940190	NO	320389	YES	SEED
546268	NO	411477	NO	321314	YES	SEED
272943	NO	306537	NO	331109	YES	SEED
228410	NO	888052	NO	331111	YES	SEED
553219	NO	981222	NO	331112	YES	SEED
314271	NO	688246	NO	331271	YES	SEED
290339	NO	903510	NO	331272	YES	SEED
485918	NO	754027	NO	340184	YES	SEED
679191	NO	189518	NO	342109	YES	SEED
252305	NO	262768	NO	349101	YES	SEED
300852	NO	756067	NO	349746	YES	SEED
395965	NO	290315	NO	349747	YES	SecReT6
254945	NO	680198	NO	360094	YES	SEED
439497	NO	649638	NO	360102	YES	SEED
759914	NO	491915	NO	364106	YES	SEED
655815	NO	196164	NO	377628	YES	SEED
572544	NO	626523	NO	381754	YES	SEED
434131	NO	633147	NO	386656	YES	SEED
225937	NO	451757	NO	400667	YES	SecReT6

743721	NO	411467	NO	401614	YES	SecReT6
317655	NO	679197	NO	405955	YES	SEED
469610	NO	762983	NO	412022	YES	SEED
355276	NO	439493	NO	423368	YES	SEED
547042	NO	122586	NO	439843	YES	SecReT6
58051	NO	553175	NO	439851	YES	SecReT6
391603	NO	342451	NO	454166	YES	SEED
387093	NO	886872	NO	454169	YES	SEED
638303	NO	879305	NO	502800	YES	SecReT6
592015	NO	445973	NO	550538	YES	SecReT6
163164	NO	394503	NO	554290	YES	SEED
498742	NO	469371	NO	557600	YES	SEED
221988	NO	237727	NO	557722	YES	SEED
669262	NO	522306	NO	585034	YES	SEED
190650	NO	517418	NO	588858	YES	SecReT6
382464	NO	525244	NO	1266738	YES	SecReT6
575588	NO	577650	NO	595536	NO	
292414	NO	552531	NO	484019	NO	
314260	NO	338963	NO	439235	NO	
997352	NO	537007	NO	366602	NO	
709797	NO	553219	NO	402881	NO	
272624	NO	360105	NO	1036674	NO	
335543	NO			638300	NO	
633131	NO			526224	NO	
258594	NO			156889	NO	
381764	NO			393595	NO	
585543	NO			634500	NO	
435590	NO			316058	NO	
570417	NO			489653	NO	
742159	NO			367737	NO	
709032	NO			983545	NO	
273121	NO			342610	NO	
416591	NO			546262	NO	
1046627	NO			582744	NO	
553171	NO			198214	NO	

391587	NO	258594	NO
795359	NO	717785	NO
543728	NO	506534	NO
313603	NO	481448	NO
531844	NO	314607	NO
315456	NO	688269	NO
1033810	NO	666681	NO
380749	NO	888741	NO
487797	NO	592316	NO
743722	NO	309807	NO
702113	NO	391600	NO
62928	NO	867900	NO
983328	NO	579138	NO
509190	NO	381764	NO
649349	NO	522373	NO
261292	NO	236097	NO
287752	NO	444612	NO
323097	NO	292805	NO
228399	NO	595537	NO
484019	NO	316057	NO
349163	NO	269796	NO
406817	NO	396595	NO
645512	NO	757424	NO
354242	NO	471854	NO
472759	NO	290315	NO
158190	NO	445932	NO
393480	NO	391615	NO
395495	NO	641118	NO
483215	NO	709032	NO
519441	NO	983920	NO
281689	NO	367336	NO
619693	NO	643562	NO
156889	NO	242619	NO
745277	NO	388413	NO
675812	NO	266779	NO

457424	NO	352165	NO
888741	NO	323098	NO
573065	NO	222891	NO
755732	NO	313590	NO
340101	NO	349124	NO
491952	NO	1082931	NO
644107	NO	880070	NO
761193	NO	281689	NO
504728	NO	326442	NO
693979	NO	205914	NO
352165	NO	224914	NO
521045	NO	600809	NO
85962	NO	638303	NO
357244	NO	335284	NO
744979	NO	363253	NO
314724	NO	886377	NO
706436	NO	426355	NO
335283	NO	572477	NO
388739	NO	272951	NO
545695	NO	888832	NO
582402	NO	272843	NO
575611	NO	264462	NO
637616	NO	634499	NO
487796	NO	760011	NO
573413	NO	741091	NO
391595	NO	583355	NO
290317	NO	658187	NO
937774	NO	653733	NO
485915	NO	866771	NO
987059	NO	760192	NO
756272	NO	688270	NO
123214	NO	391009	NO
331104	NO	243274	NO
525373	NO	331635	NO
283165	NO	629741	NO

714995	NO	880072	NO
526218	NO	237727	NO
313596	NO	283942	NO
553174	NO	537011	NO
1002804	NO	706191	NO
749222	NO	246200	NO
270374	NO	743525	NO
207949	NO	403833	NO
394221	NO	443254	NO
426355	NO	926562	NO
469613	NO	526227	NO
216432	NO	553177	NO
1005090	NO	525373	NO
375451	NO	398580	NO
314266	NO	546268	NO
399795	NO	557723	NO
1045858	NO	888057	NO
391598	NO	264731	NO
306263	NO	366394	NO
506534	NO	717959	NO
760142	NO	869210	NO
391589	NO	420662	NO
679897	NO	319225	NO
391596	NO	355276	NO
638300	NO	566466	NO
293614	NO	573413	NO
471881	NO	384765	NO
367737	NO	545695	NO
1085623	NO	1005057	NO
349124	NO	553171	NO
222891	NO	619693	NO
575590	NO	596323	NO
596320	NO	414684	NO
323848	NO	637910	NO
575587	NO	765910	NO

857571	NO	469595	NO
272843	NO	911045	NO
469618	NO	501479	NO
563192	NO	869209	NO
391624	NO	189518	NO
522373	NO	762903	NO
714943	NO	272561	NO
177437	NO	411154	NO
319795	NO	592015	NO
338969	NO	76114	NO
272947	NO	207954	NO
868864	NO	565045	NO
234826	NO	700598	NO
572477	NO	391587	NO
471854	NO	445987	NO
411154	NO	523791	NO
880072	NO	643867	NO
1000565	NO	324925	NO
661367	NO	316056	NO
862515	NO	762983	NO
		641491	NO
		313603	NO
		314260	NO
		344747	NO
		983548	NO
		667128	NO
		887898	NO
		243275	NO
		290512	NO
		545694	NO
		565034	NO
		633131	NO
		205920	NO
		158190	NO
		313598	NO

	59196	NO
	634504	NO
	257363	NO
	313594	NO
	198628	NO
	590409	NO
	56780	NO
	87626	NO
	795359	NO
	457405	NO
	387092	NO
	649761	NO
	316055	NO
	296591	NO
	314287	NO
	760142	NO
	331869	NO
	1007104	NO
	445970	NO
	580340	NO
	156578	NO
	862719	NO
	557598	NO
	1002339	NO
	290338	NO
	555217	NO
	521010	NO
	992406	NO
	404589	NO
	595494	NO
	313628	NO
	227941	NO
	547045	NO
	582899	NO
	1045855	NO

	392499	NO
	331104	NO
	575611	NO
	317025	NO
	243161	NO
	52598	NO

Top 20 ranks SVM rank files

Suppl.-Table 2 Excerpt from SVM rank file of run 18_93_200. Classifying features of final model for T3SS prediction.

GROUP_ID	SCORE	CLASS	GROUP_DESCRIPTION
COG4791	0.0451327093871	YES	type iii secretion
COG4669	0.03911983819	YES	type iii secretion
COG4789	0.037949732826	YES	Type III secretion
COG4790	0.0340518828083	YES	Type III secretion system protein
COG4794	0.0304019478839	YES	type iii secretion
COG4792	0.0269867899847	YES	type iii secretion
COG1450	0.0191699922235	YES	general secretion pathway protein D
NOG00650	0.0180774117106	YES	major facilitator superfamily
COG2032	0.0158158211414	YES	Destroys radicals which are normally produced within the cells and which are toxic to biological systems (By similarity)
COG0791	0.0155378363758	YES	NLP P60 protein
NOG01689	0.0146552131488	YES	Integrase
COG1232	0.014347805546	YES	protoporphyrinogen oxidase
COG2172	0.014051688392	YES	anti-sigma regulatory factor serine threonine protein kinase
COG5613	0.013671493175	YES	secretion system, effector
COG4643	0.0132207300428	YES	DNA primase
COG4146	-0.0127433592072	NO	solute carrier family 5 (sodium glucose cotransporter), member
COG2208	0.0125923720554	YES	protein serine threonine phosphatase
COG1773	0.0124415108768	YES	rubredoxin
COG3524	-0.0123493465423	NO	Capsule polysaccharide export
COG1324	0.0122919844008	YES	tolerance protein

Suppl.-Table 3 Excerpt from SVM rank file of run 22_57_100. Classifying features of final model for T4SS prediction.

GROUP_ID	SCORE	CLASS	GROUP_DESCRIPTION
COG3505	0.0232919560811	YES	trag family
COG3451	0.022727598531	YES	type IV secretion system protein
COG2948	0.0192692357999	YES	conjugation trbi family protein
COG3736	0.0191766768433	YES	Type IV secretion system protein VirB8
COG3504	0.0190196608164	YES	transfer protein TrbG

COG2919	0.0184093890533	YES	Essential cell division protein. May link together the upstream cell division proteins, which are predominantly cytoplasmic, with the downstream cell division proteins, which are predominantly periplasmic (By similarity)
COG1125	0.018117026252	YES	(ABC) transporter
COG1174	0.0177722193227	YES	(ABC) transporter
COG1732	0.0177722193227	YES	(ABC) transporter
COG0630	0.0176452172704	YES	type II secretion system protein
COG2902	0.017544830854	YES	Dehydrogenase
COG1388	-0.0164846038448	NO	domain protein
COG1286	0.0159094564474	YES	Colicin v production protein
COG0648	-0.0158704228352	NO	Endonuclease IV plays a role in DNA repair. It cleaves phosphodiester bonds at apurinic or apyrimidinic sites (AP sites) to produce new 5'-ends that are base-free deoxyribose 5-phosphate residues. It preferentially attacks modified AP sites created by bleomycin and neocarzinostatin (By similarity)
COG3443	0.0153618038197	YES	ABC superfamily ATP binding cassette transporter, binding protein
COG3568	0.0153393749118	YES	Endonuclease Exonuclease Phosphatase
NOG01101	0.015111658312	YES	Transporter
COG0420	-0.0149420778259	NO	Exonuclease
COG1002	0.0148689950218	YES	Type II restriction enzyme, methylase
COG2021	-0.0147511978223	NO	homoserine O-acetyltransferase

Suppl.-Table 4 Excerpt from SVM rank file of run 28_151_200. Classifying features of final model for T6SS prediction.

GROUP_ID	SCORE	CLASS	GROUP_DESCRIPTION
COG3516	0.0425164462853	YES	Type VI secretion protein, VC_A0107 family

COG3517	0.0418079769225	YES	type VI secretion protein, EvpB VC_A0108 family
COG3522	0.0397014013243	YES	type VI secretion protein, VC_A0114 family
COG3455	0.0377501211729	YES	Type IV VI secretion system protein, DotU family
COG3520	0.0374393778864	YES	type VI secretion protein, VC_A0111 family
COG3523	0.0363881503917	YES	Type VI secretion protein (IcmF)
COG3519	0.0361119585846	YES	type VI secretion protein
COG3515	0.0343441617219	YES	type VI secretion-associated protein
COG3501	0.0304316610475	YES	Rhs element vgr protein
COG3518	0.0293999907274	YES	type VI secretion system, lysozyme-related protein
COG3157	0.0236613954176	YES	type VI secretion system effector, hcp1 family
COG3456	0.0200054698176	YES	fha domain-containing protein
COG2055	-0.0197827667087	NO	malate L-lactate dehydrogenase
COG4104	0.0196263923158	YES	PAAR repeat-containing protein
COG2156	0.0173677809751	YES	One of the components of the high-affinity ATP-driven potassium transport (or KDP) system, which catalyzes the hydrolysis of ATP coupled with the exchange of hydrogen and potassium ions. The C subunit may be involved in assembly of the KDP complex (By similarity)
NOG15161	-0.0172684862238	NO	amino acid AbC transporter
COG2216	0.0172084674859	YES	One of the components of the high-affinity ATP-driven potassium transport (or KDP) system, which catalyzes the hydrolysis of ATP coupled with the exchange of hydrogen and potassium ions (By similarity)
COG3521	0.0169361072211	YES	Type VI secretion
COG2268	0.0164988058383	YES	Band 7 protein

COG3049 | 0.0164156848883 YES Choloylglycine hydrolase

Suppl.-Table 5 Excerpt from SVM rank file of run 16_97_200.

GROUP_ID	SCORE	CLASS	GROUP_DESCRIPTION
COG0267	0.134637148888	YES	50S ribosomal protein L33
COG0222	0.132302194399	YES	Seems to be the binding site for several of the factors involved in protein synthesis and appears to be essential for accurate translation (By similarity)
NOG195154	-0.131993358027	NO	mce related protein
NOG119397	0.129860390097	YES	
NOG146408	-0.118384703116	NO	
NOG82409	0.102694281854	YES	
NOG65661	0.0997831268883	YES	
NOG52272	-0.0951958804896	NO	
COG0462	-0.092391285244	NO	Phosphoribosyl pyrophosphate synthase
COG2973	-0.0862628987616	NO	This protein is an aporepressor. When complexed with L- tryptophan it binds the operator region of the trp operon (5'- ACTAGT-'3') and prevents the initiation of transcription. The complex also regulates trp repressor biosynthesis by binding to its regulatory region (By similarity)
COG0134	-0.0850245451428	NO	indole-3-glycerol phosphate synthase
COG0547	-0.0847424387036	NO	Anthranilate phosphoribosyltransferase
COG3844	-0.0844843410104	NO	Catalyzes the cleavage of L- kynurenine (L-Kyn) and L-3- hydroxykynurenine (L-3OHKyn) into anthranilic acid (AA) and 3-

			hydroxyanthranilic acid (3-OHAA), respectively (By similarity)
COG0133	-0.0813352879225	NO	The beta subunit is responsible for the synthesis of L-tryptophan from indole and L-serine (By similarity)
COG0159	-0.0801606054135	NO	The alpha subunit is responsible for the aldol cleavage of indoleglycerol phosphate to indole and glyceraldehyde 3-phosphate (By similarity)
COG0135	-0.080076454393	NO	N-(5'-phosphoribosyl)anthranilate isomerase
NOG14237	-0.0758925846894	NO	Massive surface protein
NOG114019	-0.0746281601891	NO	perforin
COG0001	0.0688058503482	YES	Glutamate-1-semialdehyde aminotransferase
NOG37965	0.0575957403065	YES	mce related protein

Genotype profile with PFAM architectures

Suppl.-Script 1 Source code to create a genotypic profile with PFAM architectures

```
#!/usr/bin/python
import os
path =
"/scratch/eichinger/PhenPred/Interpro/Pfam_architecture/T3SS/mapped_Pfam"
genotypePfamDict={}
##reads in .faa.tsv files
directory=os.listdir(path)
for file in directory:
    if file.endswith(".faa.tsv"):
        proteinPfamDict={}
        inputData=open (file, "r")
        print("inputfile "+file+" has been read")
        ##reads proteins and pfams into dictionary
        for line in inputData:
            columns = line.split("\t")
            protein= columns[0]
            pfam=columns[4]
            if protein in proteinPfamDict:
                proteinPfamDict[protein].add(pfam)
            else:
                proteinPfamDict[protein] = set([pfam])
```

```

    ##finds taxid from filename
    base=os.path.basename(file)
    taxidExt=os.path.splitext(base)[0]
    taxid=os.path.splitext(taxidExt)[0]
    genotypePfamDict[taxid] =set()
    for protein in proteinPfamDict:

genotypePfamDict[taxid].add(frozenset(proteinPfamDict[protein]))
taxids=sorted(list(genotypePfamDict))
##creates string for outputlist
taxidPfamPair=""
for taxid in taxids:
    taxidPfamPair+=taxid+"\t"
    sorted(genotypePfamDict[taxid])
    print("create output for "+taxid)
    for pfams in genotypePfamDict[taxid]:
        sorted(pfams)
        for pfam in pfams:
            taxidPfamPair+=pfam+"|"
        taxidPfamPair=taxidPfamPair[:-1]#substring removes last char
        taxidPfamPair+="\t"
    taxidPfamPair+="\n"
##writes outputfiles
print("write outputfile")
genotypePfamFile = open("genotype_pfam_architecture.txt", "w")
genotypePfamFile.write(taxidPfamPair)
genotypePfamFile.close()

```

Shell-script EffectiveDB pipeline

Suppl.-Script 2 Source code for EffectiveDB pipeline.

```

# -----
. /etc/profile
# for Predotar
module unload java
module load java/1.8.0_51
while getopts "A:B:C:D:E:FGHI:J:KL:M:N:O:" opt; do
    case $opt in
        A)
            EFFECTIVE_T3_OLD=$OPTARG
            ;;
        B)
            EFFECTIVE_T3_NEW=$OPTARG
            ;;
        C)
            PREDOTAR_ANIMAL_OR_PLANT=$OPTARG
            ;;
        D)
            PROTEINS=$OPTARG
            ;;
        E)
            MODUS=$OPTARG
            ;;
    esac
done

```

```

F)
  RUN_PREDOTAR=T
;;
G)
  RUN_EFFECTIVE_T3=T
;;
H)
  RUN_EFFECTIVE_CCBD=T
;;
I)
  RUN_EFFECTIVE_ELD=$OPTARG
;;
J)
  CheckM_COMPLETENESS=$OPTARG
;;
K)
  RUN_EffectiveS346=T
;;
L)
  GENOTYPE=$OPTARG
;;
M)
  OUTPUTFILE=$OPTARG
;;
N)
  EMAIL=$OPTARG
;;
O)
  RUN_T4SEpre=$OPTARG
esac
done

# -----

if [ -z ${RUN_T4SEpre} ]
then
  echo -e "INFO: T4SEpre run disabled"
else
  echo ${RUN_T4SEpre} | grep "^[0-9]*[.][0-9]*$"
  val=`echo $?`
  if [[ $val == 0 ]]
  then
    echo "INFO: T4SEpre parameter is valid"
  else
    echo -e "10\tStep 10 ERROR - Parameter in T4SEpre has to be
a float-number" >> ${RUN_ID}/status.TXT
    error_msg
    exit
  fi
fi

if [ -z ${OUTPUTFILE} ]
then
  RANDOMSTRING=`strings /dev/urandom | grep -o '[[[:alnum:]]]' | head -
n 30 | tr -d '\n';`

```

```

        RUN_ID=${RANDOMSTRING}
else
    RUN_ID=${OUTPUTFILE}
fi
echo -e "Data is stored in following directory: $RUN_ID"
mkdir -p $RUN_ID
cp $PROTEINS ${RUN_ID}/PROTEINS
touch ${RUN_ID}/status.TXT
# -----
error_msg(){
    if [ -z ${EMAIL} ]
    then
        echo "INFO: No email will be sent"
    else
        ./notify_error.py ${EMAIL} $(basename ${RUN_ID})
        echo "INFO: Email is sent to ${EMAIL}"
    fi
}
# -----
if [ -z $PROTEINS ]
then
    echo "ERROR: No Proteins were provided"
    echo -e "10\tStep 10 ERROR - No Proteins were provided" >>
${RUN_ID}/status.TXT
    error_msg
    exit;
fi
if [ -z $MODUS ]
then
    echo "ERROR: No Modus selected (protein-genome)"
    echo -e "10\tStep 10 ERROR - No Modus selected (protein-genome)" >>
${RUN_ID}/status.TXT
    error_msg
    exit;
fi
# -----
-----
if [ "$MODUS" = "protein" ] || [ "$MODUS" = "genome" ]
then
    echo "INFO: Calculation of model: $MODUS"
else
    echo "ERROR: Modus is invalid either 'protein' or 'genome' mode"
    echo -e "10\tStep 10 ERROR - Modus is invalid either 'protein' or
'genome' mode" >> ${RUN_ID}/status.TXT
    error_msg
    exit;
fi
# -----
-----
if [ -z "$PREDOTAR_ANIMAL_OR_PLANT" ]
then
    echo "INFO: No Predotar run will be performed"
else
    if [ "$PREDOTAR_ANIMAL_OR_PLANT" = "p" ] || [
"$PREDOTAR_ANIMAL_OR_PLANT" = "a" ]

```

```

        then
            echo "INFO: Calculation of Predotar model:
$PREDOTAR_ANIMAL_OR_PLANT"
        else
            echo "ERROR: Modus is invalid either 'p' (plant) or 'a'
(animal) mode"
            echo -e "10\tStep 10 ERROR - Modus is invalid either 'p'
(plant) or 'a' (animal) mode" >> ${RUN_ID}/status.TXT
            error_msg
            exit;
        fi
    fi
fi
# -----
-----
if [ -z ${EFFECTIVE_T3_OLD} ]
then
    echo "INFO: No EffectiveT3 old calculation"
else
    if [ "$EFFECTIVE_T3_OLD" = "sensitive" ] || [ "$EFFECTIVE_T3_OLD" =
"selective" ]
    then
        echo "INFO: Calculation of EffectiveT3 Old:
$EFFECTIVE_T3_OLD"
    else
        if [ "$EFFECTIVE_T3_OLD"" -eq "$EFFECTIVE_T3_OLD"" ]
2>/dev/null
        then
            EFFECTIVE_T3_OLD=cutoff=$EFFECTIVE_T3_OLD
        else
            echo "ERROR: Invalid value for EffectiveT3 old
model"
            echo -e "10\tStep 10 ERROR - Invalid value for
EffectiveT3 old model" >> ${RUN_ID}/status.TXT
            error_msg
            exit;
        fi
    fi
fi
# -----
-----
if [ -z ${EFFECTIVE_T3_NEW} ];
then
    echo "INFO: No EffectiveT3 new calculation"
else
    if [ "$EFFECTIVE_T3_NEW" = "sensitive" ] || [ "$EFFECTIVE_T3_NEW" =
"selective" ]
    then
        echo "INFO: Calculation of EffectiveT3 New:
$EFFECTIVE_T3_NEW"
    else
        if [ "$EFFECTIVE_T3_NEW"" -eq "$EFFECTIVE_T3_NEW"" ]
2>/dev/null
        then
            EFFECTIVE_T3_NEW=cutoff=$EFFECTIVE_T3_NEW
        else

```

```

                                echo "ERROR: Invalid value for EffectiveT3 new
model"
                                echo -e "10\tStep 10 ERROR - Invalid value for
EffectiveT3 new model" >> ${RUN_ID}/status.TXT
                                error_msg
                                exit;
                        fi
                fi
        fi
# -----
-----
if [ "$MODUS" = "genome" ]
then
        if [ -z ${CheckM_COMPLETENESS} ]
        then
                echo "INFO: No CheckM here will be performed"
        else
                if [ "$CheckM_COMPLETENESS" = "T" ]
                then
                        CheckM_COMPLETENESS=80
                else
                        if [ "$CheckM_COMPLETENESS" -eq
"$CheckM_COMPLETENESS" ] 2>/dev/null
                        then
                                echo "INFO: At least ${CheckM_COMPLETENESS}
% of genes are needed"
                        else
                                echo "ERROR: Invalid value for
CheckM_COMPLETENESS parameter"
                                echo -e "10\tStep 10 ERROR - Invalid value
for CheckM_COMPLETENESS parameter" >> ${RUN_ID}/status.TXT
                                error_msg
                                exit;
                        fi
                fi
        fi
fi
# -----
-----
if [ -z ${RUN_EFFECTIVE_ELD} ]
then
        echo "INFO: No EffectiveELD run will be performed"
else
        if [ "$RUN_EFFECTIVE_ELD" -eq "$RUN_EFFECTIVE_ELD" ] 2>/dev/null
        then
                echo "INFO: EffectiveELD run is enabled"
        else
                echo "INFO: Invalid value for EffectiveELD"
                echo -e "10\tStep 10 ERROR - Invalid value for
EffectiveELD" >> ${RUN_ID}/status.TXT
                error_msg
                exit;
        fi
fi

```

```

if [ ${MODUS} == "genome" ]
then
    if [ -z ${RUN_EffectiveS346} ]
    then
        echo "INFO: No EffectiveS346 run will be performed"
    else
        if [ -z ${GENOTYPE} ]
        then
            echo -e "INFO: No Genotype file was provided"
        else
            python check_genotype.py ${GENOTYPE}
            if [ $? = 0 ]
            then
                echo "ERROR: Invalid Genotype file"
                echo -e "10\tStep 10 ERROR - Invalid
Genotype file" >> ${RUN_ID}/status.TXT
                error_msg
                exit;
            fi
            echo "INFO: Genotype file - consistency check
successful"
        fi
    fi
fi
# -----
show_time(){
    date2=$(date +%s")
    diff=$((date2-$1))
    echo "  $((diff / 60)) minutes and $((diff % 60)) seconds
elapsed."
}
# -----
WEB_DIR=$(basename $(dirname ${PROTEINS}))
WEB_FILE=$(basename ${PROTEINS})
WEBFILE="<a
href='http://effectors.org/sites/eff/files/effective/${WEB_DIR}/${WEB_FILE}
'>Input proteins</a>"
if [ -z ${GENOTYPE} ]
then
    echo "INFO: No Link to define for Genotype file"
else
    WEB_DIR_GENOTYPE=$(basename $(dirname ${GENOTYPE}))
    WEB_FILE_GENOTYPE=$(basename ${GENOTYPE})
    WEBFILE_GENOTYPE="<a
href='http://effectors.org/sites/eff/files/effective/${WEB_DIR_GENOTYPE}/${
WEB_FILE_GENOTYPE}'>COG profile</a>"
fi
echo -e "===== " >
${RUN_ID}/Parameters.TXT
echo -e "EFFECTIVE_T3_OLD=${EFFECTIVE_T3_OLD}" >> ${RUN_ID}/Parameters.TXT
echo -e "EFFECTIVE_T3_NEW=${EFFECTIVE_T3_NEW}" >> ${RUN_ID}/Parameters.TXT
echo -e "PREDOTAR_ANIMAL_OR_PLANT=${PREDOTAR_ANIMAL_OR_PLANT}" >>
${RUN_ID}/Parameters.TXT
echo -e "PROTEINS=${WEBFILE}" >> ${RUN_ID}/Parameters.TXT
echo -e "MODUS=${MODUS}" >> ${RUN_ID}/Parameters.TXT

```

```

echo -e "RUN_PREDOTAR=${RUN_PREDOTAR}" >> ${RUN_ID}/Parameters.TXT
echo -e "RUN_EFFECTIVE_T3=${RUN_EFFECTIVE_T3}" >> ${RUN_ID}/Parameters.TXT
echo -e "RUN_T4SEpre=${RUN_T4SEpre}" >> ${RUN_ID}/Parameters.TXT
echo -e "RUN_EFFECTIVE_CCBD=${RUN_EFFECTIVE_CCBD}" >>
${RUN_ID}/Parameters.TXT
echo -e "RUN_EFFECTIVE_ELD=${RUN_EFFECTIVE_ELD}" >>
${RUN_ID}/Parameters.TXT
echo -e "CheckM_COMPLETENESS=${CheckM_COMPLETENESS}" >>
${RUN_ID}/Parameters.TXT
echo -e "RUN_EffectiveS346=${RUN_EffectiveS346}" >>
${RUN_ID}/Parameters.TXT
if [ -z ${GENOTYPE} ]
then
    echo "INFO: No Genotype information in the Parameters file"
else
    echo -e "GENOTYPE=${WEBFILE_GENOTYPE}" >> ${RUN_ID}/Parameters.TXT
fi
#echo -e "OUTPUTFILE=${OUTPUTFILE}" >> ${RUN_ID}/Parameters.TXT
echo -e "EMAIL=${EMAIL}" >> ${RUN_ID}/Parameters.TXT
echo -e "===== " >>
${RUN_ID}/Parameters.TXT
# -----
date1=$(date +%s")
# -----
# FASTA-Validation
echo "FASTA-Validation..."

if [ $(mimetype -b "${RUN_ID}/PROTEINS") == "text/plain" ]
then
    echo "INFO: Protein provided in correct format"
else
    echo "ERROR: Proteins are provided in the wrong format (e.g.
*.fasta)"
    echo -e "10\tStep 10 ERROR - Proteins are provided in the wrong
format (e.g. *.fasta)" >> ${RUN_ID}/status.TXT
    error_msg
fi
python check_seq.py ${RUN_ID}/PROTEINS ${RUN_ID}/PROTEINS.FILTERED
${RUN_ID}/PROTEINS.IGNORED
if [ $? = 1 ]
then
    echo "ERROR: Wrong format of FASTA, Multiple Fasta Sequences"
    echo -e "10\tStep 10 ERROR - Wrong format of FASTA, Multiple Fasta
Sequences" >> ${RUN_ID}/status.TXT
    error_msg
    exit
fi
echo -e "1\tStep 1 finished Multiple FASTA file check" >>
${RUN_ID}/status.TXT
show_time $date1
# -----
if [ -z ${RUN_PREDOTAR} ]
then
    echo "INFO: No Predotar run"
else

```

```

        echo "Predotar..."
        if [[ "${PREDOTAR_ANIMAL_OR_PLANT}" = "a" ]];
        then
            java -cp "./Predotar" Predotar ${PREDOTAR_ANIMAL_OR_PLANT}
            ${RUN_ID}/PROTEINS.FILTERED > ${RUN_ID}/Predotar_output_modelANIMAL.TXT
        else
            java -cp "./Predotar" Predotar ${PREDOTAR_ANIMAL_OR_PLANT}
            ${RUN_ID}/PROTEINS.FILTERED > ${RUN_ID}/Predotar_output_modelPLANT.TXT
        fi
        echo -e "2\tStep 2 finished Predotar" >> ${RUN_ID}/status.TXT
        show_time $date1
    fi
# -----
if [ -z ${RUN_EFFECTIVE_T3} ]
then
    echo "INFO: No EffectiveT3 run"
else
    echo "EffectiveT3"
    cd EffectiveT3
    if [[ "${RUN_ID}" = /* ]];
    then
        F_A=${RUN_ID}/PROTEINS.FILTERED
        F_B=${RUN_ID}/PROTEINS.NEW.EFFECTIVET3
        F_C=${RUN_ID}/PROTEINS.OLD.EFFECTIVET3
        F_D=${RUN_ID}/status.TXT
        F_E=${RUN_ID}/EffectiveT3.NEW.OUT
        F_F=${RUN_ID}/EffectiveT3.OLD.OUT
    else
        F_A=./${RUN_ID}/PROTEINS.FILTERED
        F_B=./${RUN_ID}/PROTEINS.NEW.EFFECTIVET3
        F_C=./${RUN_ID}/PROTEINS.OLD.EFFECTIVET3
        F_D=./${RUN_ID}/status.TXT
        F_E=./${RUN_ID}/EffectiveT3.NEW.OUT
        F_F=./${RUN_ID}/EffectiveT3.OLD.OUT
    fi
    if [ -z ${EFFECTIVE_T3_NEW} ]
    then
        echo "INFO: No EffectiveT3 run with new model"
    else
        java -Xmx1024m -jar TTSS_GUI-latest.jar -f ${F_A} -m
        TTSS_COMBINED-latest_2015_08_27.jar -t ${EFFECTIVE_T3_NEW} -o ${F_E} -q >
        ${F_B}
    fi
    if [ -z ${EFFECTIVE_T3_OLD} ]
    then
        echo "INFO: No EffectiveT3 run with old model"
    else
        java -Xmx1024m -jar TTSS_GUI-latest.jar -f ${F_A} -m
        TTSS_STD-latest.jar -t ${EFFECTIVE_T3_OLD} -o ${F_F} -q > ${F_C}
    fi
    echo -e "3\tStep 3 finished EffectiveT3" >> ${F_D}
    cd ..
    show_time $date1
fi
# -----

```

```

if [ -z ${RUN_EFFECTIVE_CCBD} ]
then
    echo "INFO: No EffectiveCCBD run"
else
    echo "EffectiveChaperone..."
    python EffectiveCCBD/find_CCBD.py ${RUN_ID}/PROTEINS.FILTERED >
${RUN_ID}/PROTEINS.CHAPERONES.TXT
    echo -e "4\tStep 4 finished EffectiveCCBD" >> ${RUN_ID}/status.TXT
    show_time $date1
fi
if [ -z ${RUN_T4SEpre} ]
then
    echo "INFO: No T4SEpre run"
else
    echo "T4SEpre..."

    python T4SEpre/predict_T4SS.py ${RUN_ID} ${RUN_T4SEpre}
    echo -e "5\tStep 5 finished T4SEpre" >> ${RUN_ID}/status.TXT
    show_time $date1
fi
# -----
if [ -z ${RUN_EFFECTIVE_ELD} ]
then
    echo "INFO: No EffectiveELD run"
else
    echo "EffectiveELD..."
    IPRSEQTMP=$(mktemp)
    cat ${RUN_ID}/PROTEINS.FILTERED | EffectiveELD/mask_gi.py
>$IPRSEQTMP
    /localmirror/monthly/interpro/interproscan-*/interproscan.sh -appl
Pfam -f TSV -i $IPRSEQTMP -o ${RUN_ID}/iprscan_pfam.tsv -T /tmp
    rm -f $IPRSEQTMP
    if [ "$MODUS" = "genome" ]
    then
        cat ${RUN_ID}/iprscan_pfam.tsv | EffectiveELD/eld_filter.py
g ${RUN_EFFECTIVE_ELD} > ${RUN_ID}/PROTEINS.ELD.TXT
    else
        cat ${RUN_ID}/iprscan_pfam.tsv | EffectiveELD/eld_filter.py
p ${RUN_EFFECTIVE_ELD} > ${RUN_ID}/PROTEINS.ELD.TXT
    fi
    echo -e "6\tStep 6 finished EffectiveELD" >> ${RUN_ID}/status.TXT
    show_time $date1
fi
# -----
if [ $MODUS = "genome" ]
then
    # -----
    -
    if [ -z "$CheckM_COMPLETENESS" ]
    then
        echo "INFO: No CheckM run will be performed"
    else
        mkdir ${RUN_ID}/BIN_IN
        cp ${RUN_ID}/PROTEINS.FILTERED
${RUN_ID}/BIN_IN/PROTEINS.faa

```

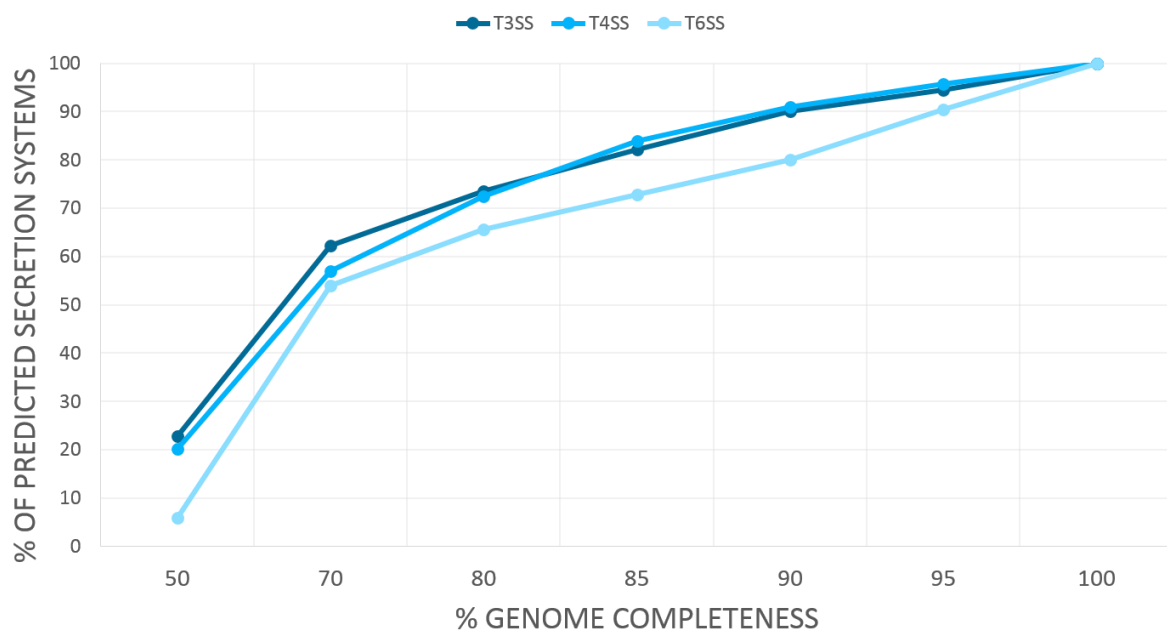
```

        ./checkM/run.sh ${RUN_ID}/BIN_IN ${RUN_ID}/BIN_OUT
${RUN_ID}/checkM_output.TXT
        python ./checkM/parse_checkM.py ${RUN_ID}/checkM_output.TXT
${CheckM_COMPLETENESS}
        if [ $? = 1 ]
        then
                echo -e "ERROR\tcheckM - genome completeness check
failed"
                echo -e "10\tStep 10 ERROR - checkM - genome
completeness check failed" >> ${RUN_ID}/status.TXT
                error_msg
                exit;
        fi
        echo -e "7\tStep 7 finished CheckM" >> ${RUN_ID}/status.TXT
fi
# -----
-
if [ "${RUN_EffectiveS346}" = "T" ]
then
        if [ -z ${GENOTYPE} ]
        then
                echo "COGnitor run..."
                ./COGNITOR/cognitor_V2.sh ${RUN_ID}
                echo -e "8\tStep 8 finished COGnitor" >>
${RUN_ID}/status.TXT
                python EffectiveS346/extract_protein_COG_pairs.py
${RUN_ID} > ${RUN_ID}/PROTEINS.genotype.txt
        else
                cp ${GENOTYPE} ${RUN_ID}/PROTEINS.genotype.txt
                echo -e "INFO\tThank you for providing a genotype,
COGnitor run is not needed here"
        fi
        echo "EffectiveS346..."
        COG_FILE=$(mktemp)
        less ${RUN_ID}/PROTEINS.genotype.txt | cut -f 1 | uniq |
sed '1 i\genome' | sed ':a;N;$!ba;s/\n/\t/g' > ${COG_FILE}
        python
EffectiveS346/application/scripts/automate_SecPred.py ${RUN_ID} ${COG_FILE}
        if [ $? = 1 ]
        then
                python EffectiveS346/compare_top100.py ${RUN_ID}
                python EffectiveS346/compare_kegg.py ${RUN_ID}
        fi
        show_time $date1
        cp ${COG_FILE} ${RUN_ID}/COG_FILE.CHECK
        rm ${COG_FILE}
        echo -e "9\tStep 9 finished EffectiveS346" >>
${RUN_ID}/status.TXT
        fi
        # -----
-
fi
# -----
python merge_results.py ${RUN_ID} > ${RUN_ID}/merge_results.html
python summary.py ${RUN_ID} > ${RUN_ID}/summary.TXT

```

```
# -----
echo -e "\nAnalysis has been finished!"
if [ -z ${EMAIL} ]
then
    echo "INFO: No email will be sent"
else
    ./notify.py ${EMAIL} $(basename ${RUN_ID})
    echo "INFO: Email is sent to ${EMAIL}"
fi
echo -e "10\tStep 10 finished Effective run" >> ${RUN_ID}/status.TXT
```

Recommended genome completeness



Suppl.-Figure 1 **Recommended genome completeness.** We recommend a genome completeness of at least 85% as the number of identified secretion systems decreases slightly (especially for T6SS), when staying beyond that limit.