universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

probeBase - an online resource for rRNA-targeted oligonucleotide probes and primers: new features 2016

verfasst von / submitted by

Daniel Greuter, BSc

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

Master of Science (MSc)

Wien, 2017 / Vienna 2017

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 862
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Chemie
Betreut von / Supervisor:	Univ.-Prof. Mag. Dr. Thomas Rattei
Mitbetreut von / Co-Supervisor:	

You have to hit it with the seed ...

— Anonymous —

Acknowledgment

First I would like to thank my thesis advisor Prof. Dr. Thomas Rattei who gave me the opportunity to realise this project and to explore my ideas. His valuable input, patience and knowledge steered me into the right direction whenever I needed guidance.
My genuine thanks also go to Prof. Dr. Matthias Horn and Prof. Dr. Alexander Loy of the Department of Microbial Ecology. Their collaboration helped me to complete this thesis and to publish my work successfully.

With a special mention of my former colleagues Oliver and Thomas. We had a lot of fun during all these sleepless nights talking and drinking beer together.

I also want to express my deepest gratitude to my mother for her encouragement throughout my life. Without you, I would never have made it this far.

Special thanks to my girlfriend Claudia, for her patience and continuous help. This accomplishment would not have been possible without you. Thank you.

Contents

Abstract

Modern technologies in molecular biology, such as next generation sequencing, allow researchers to perform more accurate and advanced analysis of organisms. Today, scientists can identify microorganisms via their unique genetic fingerprint. With the occurrence of these new methods, the number of sequenced microbial genomes has exploded over the last decades.

To gather and preserve all available information regarding those sequences, online databases, like probeBase, have been established. ProbeBase was created in 2002 by the Microbial Ecology Group staff at the Lehrstuhl für Mikrobiologie of the Technische Universität München. Over the years, the database grew and more people used it for their research. The popularity pushed it to its technical limits and made a reimplementation unavoidable. This thesis deals with the reimplementation and describes all the necessary concepts, processes and technical aspects which made the new probeBase possible.

The new web resource uses Drupal, a popular open-source content management system, as its foundation. Such frameworks allow maintainers to manage web applications more efficiently. Additional features can be added via a web interface and the core functions can be extended with custom modules. All functionalities the new probeBase offers have been implemented in such extensions.

The new probe matching feature is powered by VMATCH. This tool utilises enhanced

suffix arrays to perform fast sequencing matching and allows up to two mismatches between probes and user-submitted sequences. Besides the reimplementation of the original toolset, new features and enhancements like the proxy matching have been added. This function identifies near full-length rRNA equivalent for short query sequences. Found sequences are then matched with all oligonucleotides available in the probeBase database to get possible probe candidates of the original user submission.

Kurzfassung

Moderne Technologien in der molekularen Biologie, wie "Next Generation Sequencing" Methoden, erlauben Wissenschaftlern verbesserte und vor allem genauere Analysen von Organismen durchzuführen. Heute ist es Forschern möglich, Mikroorganismen anhand ihres einmaligen genetischen Fingerabdrucks zu identifizieren. Dieser Fortschritt ließ die Zahl an sequenzierten mikrobiellen Genomen in den letzten Jahrzehnten explodieren. Um Informationen rund um diese Sequenzen zu sammeln und aufzubereiten wurden zahlreiche online Datenbanken, wie die probeBase, aufgebaut. Die probeBase wurde im Jahr 2002 von der Forschungsgruppe für mikrobielle Ökologie am Lehrstuhl für Mikrobiologie der Technischen Universität München ins Leben gerufen. Über die Jahre wuchs die Datenbank stetig und wurde von immer mehr Personen in ihrer tägliche Arbeit genutzt. Diese Popularität brachte das System an seine technischen Grenzen und machte eine Neuimplementierung unumgänglich. Die vorliegende Masterarbeit beschäftigt sich mit diesem Prozess und beschreibt alle dafür wichtigen Konzepte sowie die notwendigen technischen Details.
Als Basis für die neue probeBase kommt Drupal, ein populäres Open Source Content Management System, zum Einsatz. Frameworks dieser Art erleichtern Webadministratoren die Wartung von Webapplikationen. Zusätzliche Features können über eine Weboberfläche freigeschalten werden. Des Weiteren können Erweiterungen durch Module hinzugefügt werden. Alle probeBase spezifischen Funktionen sind in solchen

Programmkomponenten realisiert.
Das neue "probe matching" setzt im Hintergrund auf VMATCH. Diese Software nutzt sogenannte "enhanced suffix arrays" um einen möglichst effizienten Vergleich von Sequenzen zu ermöglichen. Außerdem sind damit Sequenzvergleiche mit bis zu zwei "mismatches" zwischen Sonden und nutzerspezifischer Sequenzen möglich.
Neben etablierten Funktionen ist auch an neuen Features, wie der Proxy Suche, gearbeitet worden. Diese erlaubt es dem User, für kurze Sequenzen die dazugehörigen Volllängen-rRNA-Sequenzen zu finden. Mit deren Hilfe kann in der Datenbank für die ursprüngliche Sequenz nach passenden Sonden gesucht werden.

CHAPTER 1

Introduction

Microorganisms constitute about 60% of earth's total biomass. They influence all forms of life and are essential for ecological processes, such as promoting plant growth or decomposing organic matter. It is therefore a high-priority task in microbial ecology to understand the role of these microbial communities in nature. For a long time, the only method to study microorganisms was to cultivate them in the laboratory. Unfortunately, only a small fraction of all microbes are cultivable. Because of this, researchers in this field had just a limited view of the microbial world. New methods in molecular biology, like next generation sequencing techniques, made it possible to identify and to study microbial communities via their unique genetic structure without the need of cultivation. These advances opened the door to a new era of microbiology, allowing experiments with much higher accuracy. Microorganisms from various habitats can now be identified and studied directly from environmental samples [1]. The next sections address the basic theory behind those new methods and how they are used in the biosciences today.

1.1 DNA

Deoxyribonucleic acid, or DNA, is one of the key macromolecules in biology. It stores information which is encoded by four different organic molecules: cytosine, guanine, adenine and thymine. Sequences of these molecules hold the blueprints for various types of proteins, which are essential for all organisms on earth. Proteins perform many tasks like catalysing biochemical reactions or transporting other molecules.

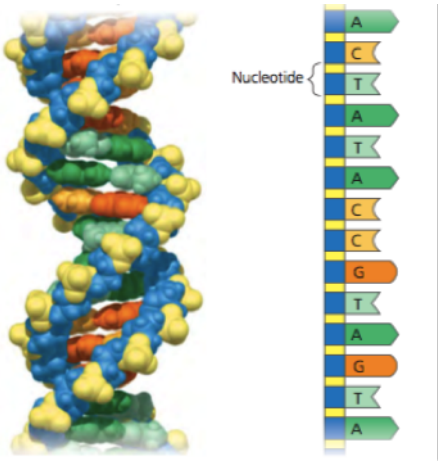


Figure 1.1: Left: DNA double helix. Right: a single strand of DNA. Adapted from Campbell Biology 10th Edition (page 5), J. Jane, B. Reece; Neil A. Campbell. 2014.

The transformation of the encoded information into the final protein is accomplished in two steps: the transcription step and the translation step.

- The first step is called transcription and replicates the information stored in the DNA into an intermediate called messenger RNA, or mRNA.
- After that, in the second step, the mRNA gets attached to a larger molecular machinery, the ribosome. This complex molecule turns the information encoded into the mRNA into the final protein [2].

This flow of information is a central concept in biology and was first described by Francis Crick in 1956 [3]. Although Crick's original hypothesis did not consider the reverse flow of information, it still was essential for the understanding of the genetic flow in organisms [4].

Figure 1.2: Flow of genetic information. Adapted from Campbell Biology 10th Edition (page 337), J. Jane, B. Reece; Neil A. Campbell. 2014.

1.2 The Ribosome

As described before, the ribosome plays a fundamental role in the protein synthesis. This cell organelle contains dozens of different proteins and RNAs and is divided into two distinct parts, differing in size and function. These two components are known as the large subunit, or LSU, and the small subunit, or SSU. The LSU is vital for the creation of the actual peptide bonds between the amino acids which are the building blocks of proteins [2]. One of the responsibilities of the SSU is to ensure that each codon in the messenger RNA is pairing with the anticodon in the transfer RNA. After the translation process is complete, the polypeptide chain and the mRNA are detached from the ribosome [5].

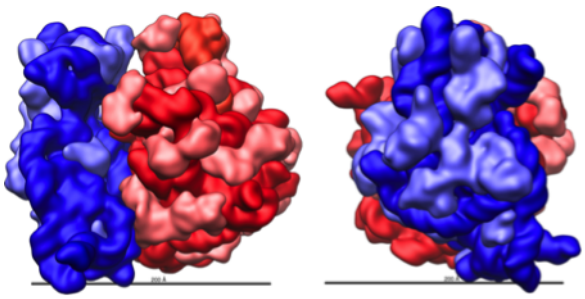


Figure 1.3: Large subunit (red) and small subunit (blue). Adapted from Wikipedia (October 2016). Retrieved from https://upload.wikimedia.org/wikipedia/commons/thumb/4/42/Ribosome_shape.png/1920px-Ribosome_shape.png.

For further classification of the ribosome and its sub-components, a unit called the Svedberg [S] is used. It describes the sedimentation rate of molecules in a centrifuged density gradient. The sedimentation behaviour of a particle depends on its mass, density, and shape. The higher the Svedberg number is, the higher is the sedimentation rate of a particle [6]. Eukaryotes have an 80S Ribosome which is subdivided

into a 60S and a 40S subunit, while prokaryotes have a 70S Ribosome, which consists of a 50S and a 30S subunit. Each of those ribosomal subunits contains characteristic ribosomal RNAs, or rRNAs, shown in table 1.1. Ribosomes are ancient biomolecules and many gene sequences embodied in those molecules are strongly conserved. That means these sequences did not significantly change over time and since all life forms have a similar organisation of rRNA genes, they are an excellent way to identify organisms. Gene sequences of the 16S rRNA are commonly used for the identification of unknown sequence samples.

	Prokaryotes	**Eukaryotes**
Total size	70 S	80 S
Size of LSU	50 S	60 S
Size of SSU	30 S	40 S
rRNAs in LSU	23 S, 5 S	23 S, 5.8 S, 5 S
rRNAs in SSU	16 S	18 S
Proteins in LSU	34 proteins	49 proteins
Proteins in SSU	21 proteins	33 proteins

Table 1.1: Composition of eukaryotic and prokaryotic ribosomes [5].

1.3 Identification of rRNA

For the identification of rRNA genes, complementary sequences are needed. These oligonucleotides are called probes or primers and act as markers by pairing with their counterpart. The hybridisation process is proportional to the probe length and typical lengths range from 18 to 50 nucleotides (nt). Long sequences lead to a higher hybridisation time and result in a reduced yield. On the other hand, if the oligonucleotide is too short, its specificity is lower. Other parameters, such as the occurrence of mismatches between the oligonucleotide and its target sequence also influence the hybridisation process. Mismatches are positions where the probe base and the corresponding target base are not complementary [7]. Every mismatch decreases the stability of the probe-target complex and the shorter the oligonucleotide is, the more noticeable this effect is. Experiments show that mismatches between pyrimidine have

a more significant impact on the stability of the complex than mismatches between purines.

$$G:T \approx G:G > G:A > A:A \approx T:T > A:C \approx T:C > C:C$$

Figure 1.4: Relative hybridisation stabilities of nucleotide pairs.

The position of mismatches within a sequence is also crucial for the magnitude of this destabilisation effect [8]. Mismatches in the centre of the oligonucleotide have a more profound impact on the hybridisation stability than those at either end. Table 1.2 shows some popular universal probes and primer sequences [9].

Name	**Application**	**Target**	**Sequence**	**Reference**
ARCH915	FISH	Most Archea	GTGCTCCCCC GCCAATTCCT	Stahl, Amann(1991)
EUB338	FISH	MostBacteria	GCTGCCTCCC GTAGGAGT	Amann et al.(1900)
BAC8F	PCR	MostBacteria	AGAGTTTGAT CMTGGCTC	Godon et al.(1997)
ARC344F	PCR	Most Archea	ACGGGGCGCA GCAGGCGCGA	Raskin et al.(1994)

Table 1.2: Examples of some universal 16S rRNA-targeted FISH probes and primers [10].

1.3.1 Fluorescence in-situ oligonucleotide hybridisation

For the detection of DNA or RNA sequences in tissues and other sample types, a technique called Fluorescence in-situ oligonucleotide hybridisation (FISH) is used. If a specific gene is present in a sample, probes are hybridising with it, followed by a fluorescence reaction to make it visible (see Figure 1.5).

Probes or primers used for FISH can either be directly labelled with so-called fluorophores or be enhanced by a target for fluorescently labelled antibodies. FISH is extensively used in research and diagnostics since it offers some advantages over other in-situ hybridisation methods.

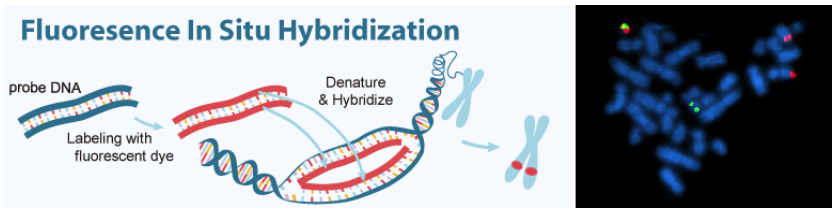


Figure 1.5: Genes visualised using fluorescence in-situ hybridisation. Adapted from Wikipedia (October 2016). Retrieved from https://upload.wikimedia.org/wikipedia/commons/d/dc/Bcrablmet.jpg. and http://www.abnova.com/images/content/support/Banner_fish.gif.

The labelled probes have a better spatial resolution than radioactive or immunocytochemical labelled ones. Additionally, they are more stable over a longer period and offer the possibility to detect multiple targets simultaneously [11].

1.3.2 Probe databases

Protocols of probes and their usage are scattered throughout literature and the world wide web. To ease the information retrieval, online databases have been established. These systems try to collect and preserve all knowledge of the corresponding oligonucleotide. New sequences can be added by a curator, by user submissions or by a combination of both. Established sequence archives, such as Genbank, include sequences from many sources [12]. These sequence records are mostly unreviewed and may be of low quality. In contrast to that are databases like RDP, SILVA or probeBase which offer up-to-date curated data as well as complete taxonomic classifications of every entry. The Ribosomal Database Project (RDP) was started in 1992 by Carl Woese [13] [14]. Over the years it grew from a few hundred to several million aligned and annotated rRNA gene sequences. The sequences are obtained from major sequence archives, such as GenBank or the European Nucleotide Archive (ENA), or through direct submissions to RDP. To support scientist in their efforts, the database provides tools for the identification of related sequences or probe as well as a primer matching services [15]. The SILVA database is a collaboration between the Microbial Genomics Group at the Max Planck Institute for Marine Microbiology in Bremen, Germany, the Lehrstuhl für Mikrobiologie at the Technische Universität München,

and Ribocon GmbH. The database contains quality-controlled datasets of aligned small (16S/18S; SSU) and large (23S/28S; LSU) ribosomal RNA sequences for all three domains of life [16] [17] [18]. SILVA can be accessed via a taxonomy browser and through search forms. The browser offers a hierarchical view of the data stored in the database, visualising any of the taxonomies available. It is also possible to evaluate probes or primers with the tools *TestProbe* and *TestPrime* [18]. The sequences in the database are available in four different quality categories [19]:

1. Parc: Comprehensive quality checked SSU & LSU sequences.
2. Ref: Highquality SSU & LSU data bases only including sequences longer than 1200/1900 nucleotides.
3. Ref NR 99: Nonredundant SSU Ref database with a 99% identity criterion.
4. Manually curated taxonomy and guide trees.

1.3.3 ProbeBase - Origins

ProbeBase was established in 2002 by the Microbial Ecology Group staff at the Lehrstuhl für Mikrobiologie of the Technische Universität München. When the database went online, it held approximately 700 published rRNA oligonucleotide probes, offering information on the probe sequences, target organisms, target molecules, target sites, and thermodynamics. It contained background information on the correct probe usage as well as a reference to the original publication. Various features helped users to discover all relevant information of these oligonucleotides. They could either be found by their target, by their target site or directly by the name of the probe itself. A matching tool allowed researchers to upload their sequence data to check for probes matching those sequences. Additionally, all available oligonucleotides in the database can be matched with external rRNA sequence databases. The project grew steadily over time. More features were added to the database to meet the user requirements. A list-functionality, allowed users to retrieve complete lists of all probes with specific characteristics, like the successful application in FISH experiments or the usage in microarrays. A comment system gave users the opportunity to comment

on sequences and helped the team around probeBase to further improve the overall database quality by reducing errors in entries and by adding additional information on oligonucleotides. The user count and the number of available probes increased dramatically over the years. Today it holds about 2900 probes, 200 primers, and 16 microarrays. More than 400 paper references can be found in the database too [20]. The popularity of the original web resource pushed it to its technical limits. As a result, features like the probe matching tool had to be disabled since it could not handle the vast amount of users simultaneously. Also, using Microsoft Access as the database system for the resource was not the best choice and had a significant impact on the user experience. Managing the database became harder for the maintainers the more popular the database grew over the years. Flaws in the design of the database model made the maintenance more and more time-consuming. These issues made an update of the original probeBase necessary.

1.3.4 Goal of this thesis

In this thesis, I describe how the probeBase web resource has been reimplemented. I start by explaining some of the commonly used bioinformatic tools today. By sketching out the advantages and disadvantages these applications offer, I justify why particular programs have been chosen for the new probeBase. In chapter 3, I give a more detailed picture on the implementation. I describe all relevant technical topics, ranging from the export and cleaning of the original data to all aspects regarding programming and enhancements made to improve the usability of the former resource. In the second last chapter, I demonstrate how probeBase users can work with the database via typical use-cases. The last chapter reflects on the whole project, discussing challenges I faced and plans which have been successfully or not so successfully realised during the development. Finally, I give an outlook on potential challenges the resource may face in the future.

CHAPTER 2

Bioinformatic tools and methods

Over the last decades, biology has become a highly data-centric area of research. Due to the advances in sequencing and the decreased costs of this process, the amount of biological data has exploded. To handle and analyse this quantity of information, researchers nowadays need more than expert knowledge in their specific field. It is mandatory that scientists also have a solid understanding of computing concepts and so it is not surprising that programming classes are taught in many undergraduate biology programs today. One important concept in computer science, which is extensively used in biology, are strings. A string is an ordered list of a finite set of symbols like characters or numbers [21]. Bioscientists use them to store the sequence information of molecules like proteins or DNA (see Figure 2.1).

(A) ATGAATCATCGAAAATTCAGCATCA
(B) IIMIRLCTIVLIAAGVAVALYLSLYGY

Figure 2.1: (A) Example of a DNA sequence. (B) Example of a protein sequence.

All public sequence databases, organised within the International Nucleotide Sequence Database Collaboration (INSDC), offer the complete genome of many species

stored in the form of long strings. In these genomes, researchers try to find specific gene motifs by using string matching algorithms. The importance of this process for biology lies in the fact that genes encode proteins which fulfil essential tasks in an organism. Finding a known gene pattern in an unknown genome, helps researchers to decipher and better understand genomes.
Looking for sequence patterns in large genomes is like trying to find a needle in a haystack. Without the availability of modern computers, the proper software tools and methods, it would be impossible to perform those matching tasks.
String matching algorithms fall into two classes: exact string matching and similarity-based string matching. Consider two strings T and P. In the exact string matching approach only a pattern P that perfectly matches a subset of T is considered as a match (Figure 2.2 left). In contrast to this are approximate based methods. They allow the occurrence of mismatches between P and T (Figure 2.2 right).

Figure 2.2: Left: Exact string matching approach. Right: Approximate string matching approach.

There are many different algorithms available for both types. The next part of this chapter illustrates the basic ideas behind some of these algorithms.

2.1 Exact string matching algorithms

2.1.1 Naive string matching

The naive algorithm is the most straightforward technique for exact string matching. In this approach a pattern P is aligned with a target string T. Then the pattern is shifted character by character towards the end of the target string to check if the pattern motif appears within it (Figure 2.3). This algorithm is easy to implement and to understand. Unfortunately, it is not an efficient method because, in the worst possible case, it has to check all positions of T for an appearance of P. For large

strings, like in genome analysis, this is not applicable since it would take to long to finish the matching task [22].

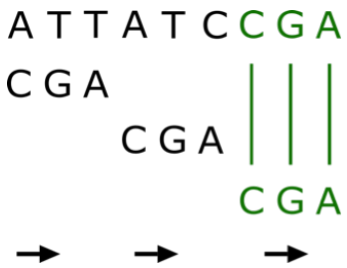


Figure 2.3: Example of the naive string matching approach.

2.1.2 Boyer-Moore algorithm

The Boyer-Moore algorithm was developed by Robert S. Boyer and J. Strother Moore in 1977. The advantage of this approach is that pointless alignments between the pattern P and the target T are skipped. This reduces the matching time drastically. The first step in this method is to align P with the target string T. Starting from the rightmost character in the pattern string, P and T are compared character by character. If two characters are not equal, the algorithm skips further checking and a new pattern-target alignment is calculated. The shift-calculation can be done in two ways: via the Good Suffix Shift or the Bad Character Shift [23].

The Good Suffix Shift

The Good Suffix Shift is also known as the Good Suffix Heuristic. If a mismatch occurs between P and T, the pattern P gets shifted to the right until the next subpattern "GA" in P is aligned with T (Figure 2.4 left). If there is no further sub-pattern in P available, the whole pattern P gets shifted beyond the position of the last pattern character (Figure 2.4 right).

The Bad Character Shift

The Bad Character Shift is also known as the Bad Character Heuristic. If a character in a pattern P does not match with the one in the string T, the pattern is shifted, so

Figure 2.4: Examples of Good Suffix Shifts.

the "Bad Character" in the target string is aligned with the next occurrence of this character in P (Figure 2.5).

Figure 2.5: Example of a Bad Character Shift.

If the "Bad Character" of T is not available in P, then the pattern is shifted beyond the current position of the rightmost pattern character. The Boyer-Moore algorithm is a widely used approach in text editors or command line tools like GNU Grep [24]. If the search pattern has a large alphabet, Boyer-Moore is considered as one of the best-suited approaches in pattern matching [25].

2.1.3 Suffix Trees

In contrast to Boyer-Moore are data structures like suffix trees. They pre-process the target strings and store them in an optimised form to gain a better matching performance. As the name implies, the suffix tree has a tree-like shape. Building the tree takes some time first, but it only needs to be done once. This approach is useful for scenarios where the target strings are relatively constant over time, like in genomes [22]. There are many different algorithms available to construct a suffix tree.

A popular one is the Ukkonen's algorithm. The idea behind Ukkonen's approach is to build the tree by adding the suffixes starting with the smallest suffix and progressing towards the longest suffix of the string. It is a so-called "on-line" method because all processed characters are immediately available in the suffix tree [26].

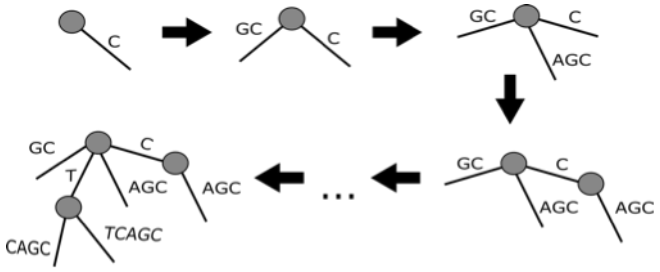


Figure 2.6: Construction of a suffix tree.

The algorithm starts with the smallest suffix "C" (see Figure 2.6). "C" is added to the root node. Then the suffix "GC" is evaluated. Since there is no branch starting with "G", this suffix becomes a new path within the tree. The next suffix is "AGC". There was no branch previously added that starts with "A" and so this suffix is added as a new side branch. "CAGC" is the following suffix. A "C" branch was already added in step one but since this branch does not have any further sub-branches, "AGC" gets added to the "C" branch. This process is repeated until all suffixes are added to the tree. The highest possible number of branches for any given node in this example is four because there are only four different nucleotides in the DNA.

String matching in a suffix tree

Looking for a motif within a suffix tree is accomplished by travelling through the structure starting from the root node and proceeding down the branches, character by character until a possible match is found (Figure 2.7).

A disadvantage of this data structure is the space it consumes in memory, especially for large sequence strings, common in genome analysis. Fortunately, the suffix tree has inspired many other data structures which have overcome this drawback. An alternative to the suffix tree with an optimised space consumption are suffix arrays [27].

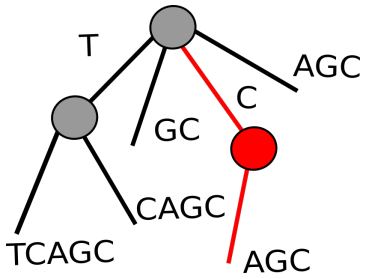


Figure 2.7: Example of string matching in the suffix tree for the suffix CAGC.

2.1.4 Suffix Array

In 1990, Manber and Myers introduced suffix arrays as space efficient alternatives to suffix trees. Suffix arrays are data structures which store all the suffixes of a string in a lexicographically sorted array. The construction of the suffix array is accomplished in two steps. At first, all the suffixes are listed and ordered by the length of the strings. Then all the suffixes are sorted lexicographically (Figure 2.8).

i	SA[i]	
0	3	AGC
1	5	C
2	2	CAGC
3	4	GC
4	1	TCAGC
5	0	TTCAGC

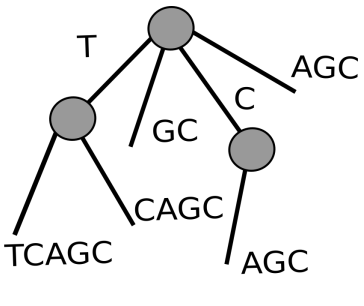


Figure 2.8: Left: Example of a suffix array. Right: Corresponding suffix tree.

Binary search in a suffix array

The first and the last element of the array serve as the boundaries L and R. Then the suffix at position $floor(\frac{L+R}{2})$ is compared with the search pattern. If no matching suffix is found and if the suffix at this location is lexicographically lower ranked then the search pattern, the second half of the suffix list is discarded and the first half is used for the next iteration. If it is lexicographically higher ranked than the search pattern, the whole procedure is performed vice versa. Again, the outer two elements

of this sub-list serve as the borders for the next search interval and the following suffix at position $floor(\frac{L+R}{2})$ is compared with the search pattern. These steps are repeated until the pattern is found or the suffix list is empty (Figure 2.9).

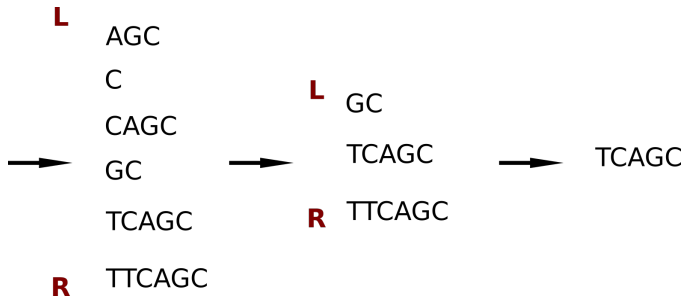


Figure 2.9: Example of a binary search in a suffix array.

By using only the basic form of a suffix array, it is not possible to replace all the algorithms using suffix trees. Some features which suffix trees offer, such as bottom-up or top-down traversal, are heavily used in many string matching problems and are not available in basic suffix arrays. To overcome this obstacle, so-called "enhanced suffix arrays" have been developed.

2.1.5 Enhanced Suffix Array

Enhanced Suffix Array are data structures consisting of the suffix array and additional table entries. The enhanced suffix array contains the suffix array, an LCP array, and a child table. LCP stands for **L**ongest **C**ommon **P**refix. The values in these columns represent the length of the longest prefix which two adjacent string prefixes have in common. Mohamed Ibrahim et al. showed that with the help of an LCP-interval tree, it is possible to solve all problems with enhanced suffix arrays that are usually solved by a bottom-up traversal done in suffix trees [28]. The child table contains three values per index: up, down, and nextlIndex. With the help of these values, an optimal top-down traversal can be achieved [29]. These additional tables are the reasons why all algorithms which are based on suffix trees, can be replaced by enhanced suffix arrays [27]. Using this data structure is favourable because algorithms which are based on them are easier to implement and more space efficient than their

tree counterpart. Additionally, experiments have shown that the running times for these algorithms are much better than those of suffix trees.

2.2 Approximate string matching methods

The problem of exact string matching is that only 100% identical sequences are considered as matches. Mismatches between two sequences can be the result of evolutionary reasons like mutations but can also be the result of technical errors which have occurred during the sequencing process. That's the reason why in biology matching only partly similar sequences is far more important than matching completely identical sequences. To evaluate two "similar" sequences, approximate string matching methods are needed which can handle mismatches between a pattern and a target string. One way to solve the approximate string matching problem is through dynamic programming algorithms. These types of algorithms try to solve complex problems by breaking them down into a smaller collection of simpler independent sub-problems. Two prominent examples of this approach are the "Smith-Waterman" and the "Needleman-Wunsch" algorithms.

2.2.1 Smith-Waterman algorithm

The Smith-Waterman algorithm is a local alignment method. It tries to find regions of similarity between two strings. To obtain the local alignment with this technique, a matrix of two given string sequences is initialised. The matrix is then filled with scoring values, according to a predefined scoring scheme. In the final step, a so-called traceback is performed. The purpose of the traceback is to find all possible alignments between the two strings. It starts with the highest score and traces the path of the alignment in the matrix until the scoring value reaches zero. The process is then repeated with the second highest score, then with the third highest score and so on until all possible paths are calculated (Figure 2.10) [30].

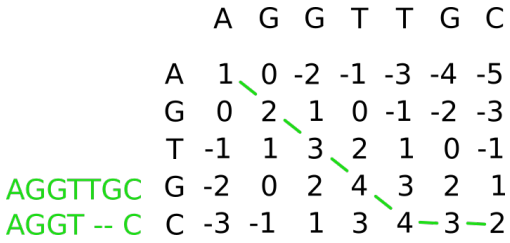


Figure 2.10: Pairwise alignment between two sequences with the Smith-Waterman algorithm.

2.2.2 Needleman-Wunsch

The Needleman-Wunsch approach is an example of a global alignment method. These methods try to find the best end-to-end alignment between two given strings. The algorithm is similar to the Smith-Waterman algorithm. Like Smith-Waterman, the method starts with generating a sequence matrix of two strings. Then it is filled with scoring values according to some predefined scoring scheme. As the last step, a traceback is derived to find the optimal alignments.

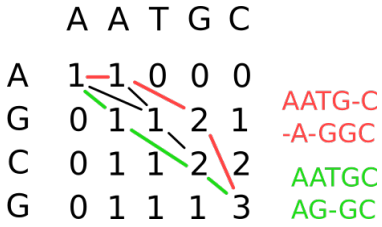


Figure 2.11: Pairwise alignment between two sequences with the Needleman-Wunsch algorithm.

The difference between the two algorithms is, that the Smith-Waterman algorithm sets negative scoring values to zero, so no negative values appear in the matrix. This makes the local alignments visible (Figure 2.11). Both approaches work well with smaller strings, but the computational costs for large sequences make them unfavourable for genome analysis. To handle larger and more complex sequences, heuristics are used.

Heuristic methods do not guarantee to find the optimal or best solution of a problem but the results should be sufficient and the performance of the string matching process is much higher. Tools like BLAST or BLAT, implement such heuristics and allow much faster execution times.

2.2.3 BLAT – Blast Like Alignment Tool

BLAT was created 2002 and is designed to achieve fast and accurate alignments of both DNA and protein sequences. It was created for the UCSC Genome Browser, as a faster alternative to BLAST and is mainly used for genome size sequences. Behind the scenes, BLAT uses a different indexing approach for the sequences than BLAST. In BLAST, the target database is a set of GenBank sequences. In BLAT, the database is an index which is derived from the assembly of the complete genome. This index provides a huge query speed improvement over BLAST [31].
While the BLAST algorithm scans through the database sequences for each query sequence, BLAT only browses through the query sequences which is faster. This is ideal for interactive web resources where users submit sequences and expect to receive results in a reasonable time (Figure 2.12) [32]. The speed optimisation through the BLAT approach comes at the expense of abandoning weak homology [33]. BLAST can find much more remote matches than BLAT and is, therefore, the recommended tool when searching more distantly related sequences [34].

2.2.4 VMATCH

VMATCH was created by Stefan Kurtz from the centre of Bioinformatics in Hamburg and was built for large-scale matching tasks. It pre-processes a set of sequence strings into a persistent index over a collection of several files. The index represents all sub-strings of a pre-processed sequence and results in high matching performance, independent of its size. It utilises enhanced suffix arrays, which have the advantage of reduced space requirements and an optimised processing time. With enough system resources, the 32-bit version of the tool can handle up to 400 million symbols per sequence. The 64-bit version can process even larger sequences [36]. The tool is a non-heuristic aligner, which means that in contrast to heuristic tools like BLAT

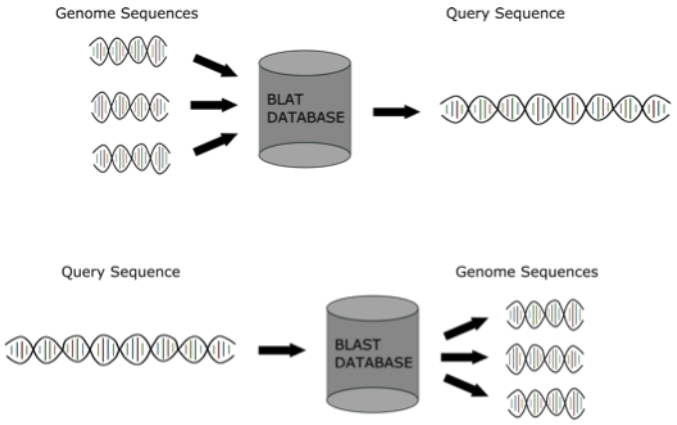


Figure 2.12: Top: The BLAT approach. A pre-computed index of the genome is stored in the database and compared to the user submitted sequences. Bottom: The BLAST approach. The query sequence is first indexed and then the whole genome is analyzed [35].

or BLAST it guarantees to find all available hits which fulfil certain criteria. One particularity of VMATCH is that it is independent of the alphabet of the strings. The creators of the sequence aligner call this concept "symbolic mapping". The users can define a custom alphabet, not larger than 250 symbols, to predefine which characters or groups of characters in the matching process should be considered as a match. Therefore, VMATCH can be considered as a "universal" matching tool, not restricted to DNA or Protein sequences [37].

2.2.5 LAST

LAST is designed for genome-scale sequencing applications. Like BLAST it utilises a "seed-and-extend" approach but instead of contiguous seeds, it uses so-called "spaced seeds" which are seeds containing gaps (Figure 2.13). Contiguous seeds with a fixed length perform well on sequences with a uniform distribution of nucleotides. The problem is that biological sequences like primate genomes contain various copies of certain elements [38]. In BLAST-like tools, this leads to high seed matching numbers. To overcome this, a possible solution would be the usage of larger seeds, but they would fail to detect weaker similarities. The concept of spaced seeds was introduced by Ma et al. (2002). By using this type of seeds, a better sensitivity than with fixed-

length seeds can be achieved as well as a lower running time [39]. This is the reason why LAST can handle large sequence data, like two vertebrate genomes, without becoming overwhelmed by an abundant number of hits [40].

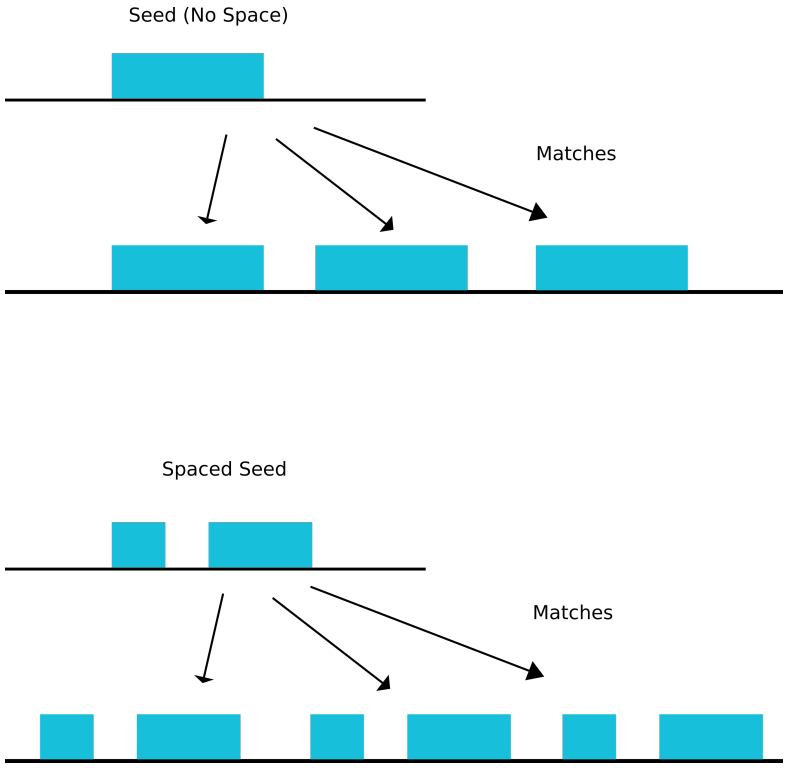


Figure 2.13: Example of spaced seeds and Non-Spaced seeds approach. Adapted from sevenbridges.com (March 2017).

CHAPTER 3

Implementation

This chapter deals with the implementation of the new probeBase. Here I describe all concepts and design decisions made during this process and I reveal all implementation details of the project which led to the final product. When I started my thesis project, I obtained the latest version of the source code and a database dump from the probeBase maintainers. The original database system which was used for the web resource was Microsoft Access. The updated probeBase uses a different database system and it was, therefore, necessary to export the dump into the open XML format. XML is the abbreviation for eXtensible Markup Language and describes a human- and machine-readable data format to store and transport any digital information. This file type has a tree-like structure, starting with a root node and branching out to leaves.

```
<root>
        <child>
        <subchild>...</subchild>
        </child>
</root>
```

Code 3.1: XML syntax example.

XML files are text-based and it is possible to inspect them in any text editor. After I exported the database dump to XML, I compiled a list of the data and reviewed

it together with the probeBase team. The purpose of this was to better understand the available data. Shortly after revision, it was obvious that some of the original data have become out-of-date while others were not needed anymore. I excluded those from the database transition process. Further analysis also revealed some disadvantages in the database design. An example of this is the probe specificity, which indicates the target organism of a probe. In the original data model, the specificity information was kept in three distinct data fields: "specificity", "taxonomy_down" and "taxonomy_up". The field "specificity" describes the species targeted by the probe while "taxonomy_up" and "taxomomy_down" contain lists of taxa, determining possible probe targets in higher or lower taxonomic ranks. The drawback of this approach is that maintainers always had to consider all three fields if they wanted to change the specificity information of an existing probe or if they wanted to add new oligonucleotides. It is obvious that this process was error-prone. To improve this situation, taxonomy IDs were introduced to the database. The concept behind these IDs is discussed in section 3.1.9.
Another design flaw was that the number of specific probe features, such as the probe references was "hardcoded" into the database schema. It was impossible to add a higher number of reference entries to a probe than predefined by the database model. Moreover, if not all feature fields are required by a probe entry, the record contained unused data fields. To allow probes to have an arbitrary number of feature fields, the model has been enhanced by a separate table containing these features. This external data table is connected with the probe table via a foreign key table, holding the probe ID and the feature ID. That tuple relates every probe with its corresponding feature and prevents empty fields in the database (see Figure 3.1).

3.1 The new database design

After several meetings with the probeBase team, I prepared a catalogue with all the data that was still relevant and necessary. This collection was the basis for the new database schema. As figure 3.1 illustrates, "probe" is the central table in the database.

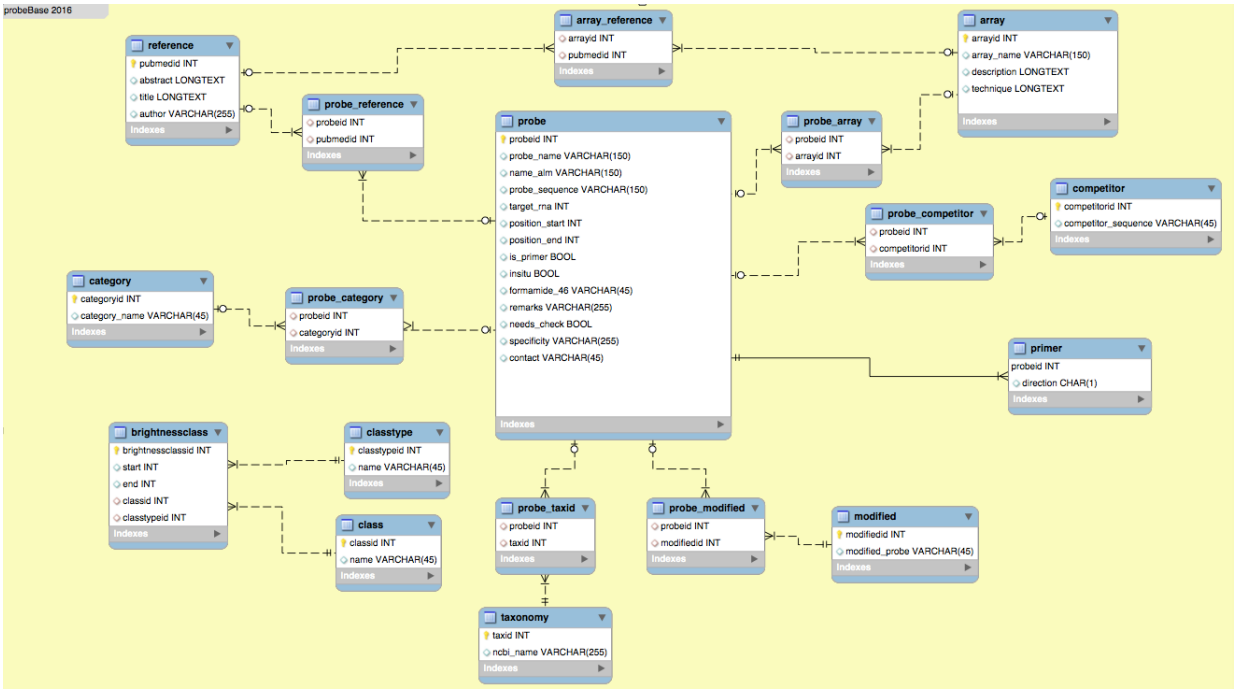


Figure 3.1: Full probeBase database model.

It contains the core information of every probe like the name, the sequence or the target RNA. Every probe gets a unique probe ID in the data table for identification. Aside from this table, there are additional ones attached to it.

3.1.1 Category table

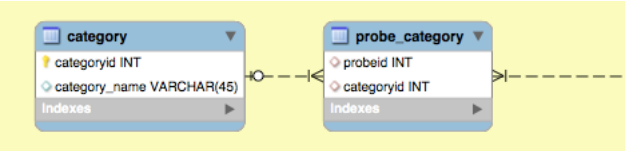


Figure 3.2: "Category" and "probe_category" tables.

To group probes with certain commonalities, the table "category" was added to the schema. This table contains two columns: "category_name" and "categoryid". In total, there are 29 category names and every one of them has a unique ID. Probes

are associated with their corresponding categories via a foreign key table, holding the probe IDs and the category IDs.

3.1.2 Primer table

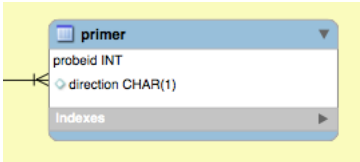


Figure 3.3: "Primer" table.

If a probe is also a primer, a new row in the primer table is added, containing the probe ID as the foreign key and the orientation of the primer represented by a single character. Reverse primers have the character 'r' stored in this data field while forward primer store the character 'f'.

3.1.3 Competitor table

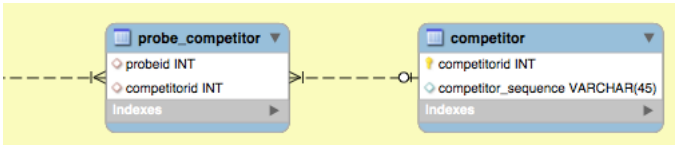


Figure 3.4: "Competitor" and "probe_competitor" table.

To suppress unspecific or undesired probe bindings with non-target sequences, competitor sequences are used. Known competitors are held in a separate table called "competitor". Every sequence has a unique competitor ID. The probe table is connected to the competitor table via a foreign key table, holding the probe ID and the competitor ID.

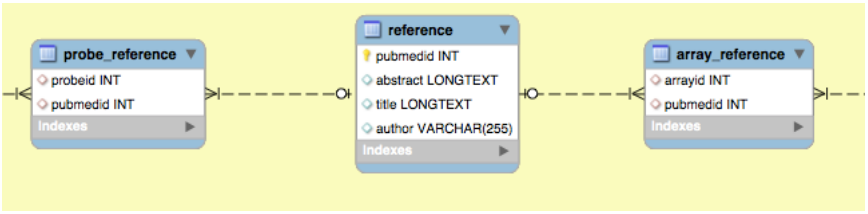


Figure 3.5: "Reference", "array_reference" and "probe_reference" table.

3.1.4 Reference table

The table "reference" holds all the references a probe or a microarray is related to. If available, a PubMed ID of the reference is used as the unique primary key in this table. It also holds the title, the authors and the abstract of the corresponding reference. The table "probe_reference" relates every reference with its probe. Respectively, the table "probe_array" associates every array with its corresponding paper.

3.1.5 Array table

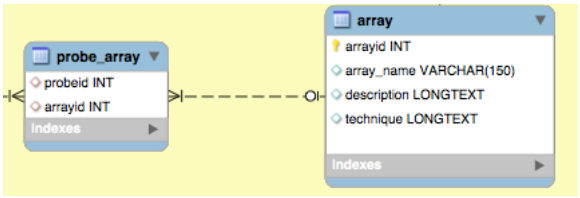


Figure 3.6: "Array" and "probe_array" table.

The database table "array" holds information on microarrays. It contains detailed information of microarrays such as the name, the description, and the used technique. Every microarray has a unique array ID. To associate a probe which is used within a microarray, an entry in the foreign key table "probe_array" is added. This table holds the probe ID and the array ID.

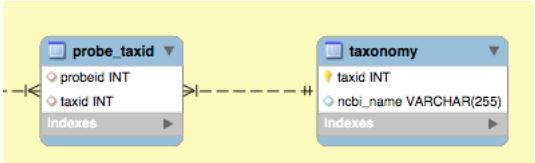


Figure 3.7: "Taxonomy" and "probe_taxonomy" table.

3.1.6 Taxonomy table

The "taxonomy" table holds the NCBI name of the target organism of the probe. The primary key in this table is the taxonomy ID from NCBI taxonomy database. A foreign key table connects every probe with its corresponding target organism with the probe ID and the taxonomy ID.

3.1.7 Modified table

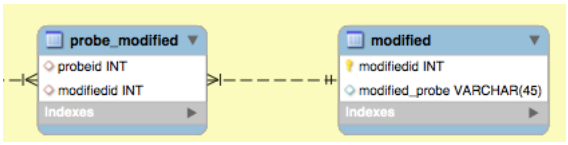


Figure 3.8: Taxonomy and "probe_modified" foreign table.

The table "modified" holds all probe IDs which are alternative versions of the probe. The oligonucleotide and its different versions are connected via their unique IDs in a foreign key table.

3.1.8 Brightnessclass tables

The Accessibility of oligonucleotides is varying across rRNA target sites. To classify this probe characteristic, experiments by Behrens and Fuchs were made [41] [42]. In probeBase, this important information is stored in three tables: "brightness-class", "class" and "class type". The table "brightnessclass" holds all positions of the Behrens and the Fuchs classification, together with the actual positions in the rRNA. The table "class" holds the available brightness classes ranging from "I" -

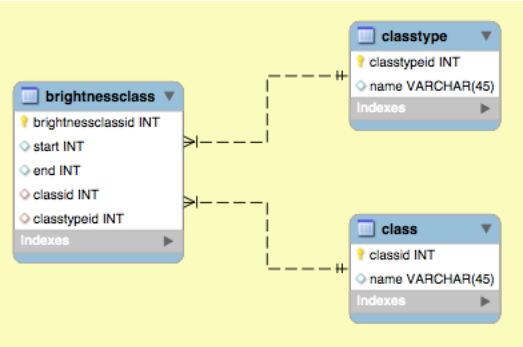


Figure 3.9: "Brightnessclass", "classtype" and "class" table.

"VI". "I" represents the highest class with the brightest fluorescence signal while "VI" is the lowest class with almost no detectable fluorescence signal. The table "class type" holds all the available class types, such as Fuchs or Behrens. Aside from the probe data tables, there is also a local copy of the NCBI Taxonomy Database in the new probeBase. The information stored in these tables is primarily used for the autocompletion of input fields and the dynamic reconstruction of the lineage from any given taxonomy ID.

3.1.9 The NCBI Taxonomy Database

The goal of the NCBI Taxonomy Database project is to provide a consistent nomenclature and classification for all the organisms available in the sequence databases. Every organism gets its unique taxonomic identifier, which makes it possible to determine the species and to retrieve the exact lineage for any given organism. The latest taxonomy database dump is always available under ftp.ncbi.nih.gov/pub/taxonomy/ in the form of a compressed tar archive file. The data is partitioned into two tables: "names" and "nodes". "Names" holds the taxonomy IDs, names, unique names and name classes. The "nodes" table stores information like the rank, comments, the parent taxonomy ID and other fields. To import this data, the schemas of two tables need to be added to the database first: "ncbi_names" and "ncbi_nodes". Both are created by issuing the following two commands:

```
DROP TABLE IF EXISTS `ncbi_names`;
CREATE TABLE `ncbi_names` (
        `tax_id` mediumint(11) unsigned NOT NULL default '0',
        `name_txt` varchar(255) NOT NULL default '',
        `unique_name` varchar(255) default NULL,
        `name_class` varchar(32) NOT NULL default '',
        KEY `tax_id` (`tax_id`),
        KEY `name_class` (`name_class`),
        KEY `name_txt` (`name_txt`)
) TYPE=MyISAM;
```

Code 3.2: Creating the "ncbi_names" table.

```
DROP TABLE IF EXISTS `ncbi_nodes`;
CREATE TABLE `ncbi_nodes` (
        `tax_id` mediumint(11) unsigned NOT NULL default '0',
        `parent_tax_id` mediumint(8) unsigned NOT NULL default '0',
        `rank` varchar(32) default NULL,
        `embl_code` varchar(16) default NULL,
        `division_id` smallint(6) NOT NULL default '0',
        `inherited_div_flag` tinyint(4) NOT NULL default '0',
        `genetic_code_id` smallint(6) NOT NULL default '0',
        `inherited_GC_flag` tinyint(4) NOT NULL default '0',
        `mitochondrial_genetic_code_id` smallint(4) NOT NULL default '0',
        `inherited_MGC_flag` tinyint(4) NOT NULL default '0',
        `GenBank_hidden_flag` smallint(4) NOT NULL default '0',
        `hidden_subtree_root_flag` tinyint(4) NOT NULL default '0',
        `comments` varchar(255) default NULL,
        PRIMARY KEY (`tax_id`),
        KEY `parent_tax_id` (`parent_tax_id`)
) TYPE=MyISAM;
```

Code 3.3: Creating the "ncbi_nodes" table.

After all tables have been created, the following command is executed to import all the data into the tables:

```
LOAD DATA INFILE 'names.dmp'
INTO TABLE ncbi_names
FIELDS TERMINATED BY '\t|\t'
LINES TERMINATED BY '\t|\n'
(tax_id, name_txt, unique_name, name_class);
```

Code 3.4: Importing all data to the "ncbi_names" table.

The same command is used to import the nodes table dump:

```
LOAD DATA INFILE 'nodes.dmp'
INTO TABLE ncbi_nodes
FIELDS TERMINATED BY '\t|\t'
LINES TERMINATED BY '\t|\n'
(tax_id, parent_tax_id, rank, embl_code, division_id, inherited_div_flag,
genetic_code_id, inherited_GC_flag, mitochondrial_genetic_code_id,
inherited_MGC_flag,GenBank_hidden_flag, hidden_subtree_root_flag, comments);
```

Code 3.5: Importing all data to the "ncbi_nodes" table.

3.2 Mapping taxonomy IDs to probes

Based on the data stored in the original dataset, all probes are mapped against the NCBI taxonomy IDs to derive a distinct target organism for each oligonucleotide in the database. Three different approaches are pursued to accomplish the mapping task:

1. A NCBI taxonomy ID is used, if sufficient information is available and the mapping of the corresponding NCBI taxonomy ID is clear.

2. If a probe is targeting multiple taxa, two cases are considered. The first one is, if one taxon is predominantly covered by the probe, the ID of this taxon is preferred and used. If the former approach is not possible, the taxonomy ID of the next higher taxonomic rank is used, since it also covers all taxa of the lower ranks.

3. If it is impossible to assign any taxonomy ID to a given probe, the root ID 1 of the NCBI taxonomy database is used instead. The information of possible targets is still available in the specificity field in the probe database table.

Many of the original probe records include the same specificity information. To avoid unnecessary work, the first task was to create a data subset, containing all unique specificity entries. After this pre-processing step, the dataset was reduced by 28% to approximately 1986 entries. To automate the mapping process, a custom python script has been developed. It successfully assigned 1511 of the oligonucleotides to the corresponding taxonomy ID. For 401 probes it was necessary to map them manually due to ambiguities in the specificity, like "Candidatus Burkholderiacalva (AY277697)", "Burkholderia cepacia (AY512825)". From this 401 probes, only 104 remain without any distinct taxonomy ID. They had not enough information in their original data record to associate them with any target organism. The reason for this are descriptions like "filaments from hydrothermal vents near White Point" which do not hold any valuable information. These probes have been assigned to the root taxonomy ID 1, which is not related to any organism. But since the original specificity data is kept, it is still possible to find hints regarding the specificity. After setting

up the database and importing all the data, the actual work on the probeBase web resource began.

3.3 Choosing the Web framework

A disadvantage of the original probeBase was that it did not use any content management system (CMS). This type of systems allows webmasters to maintain their websites without any knowledge of programming. New features can conveniently be added through a web interface in just a matter of seconds, like a newsletter system. With the help of a CMS, the administration of a website can be shared among multiple maintainers through fine-grained roles and permissions. It is possible that some users have full access while other users are only allowed to work on certain areas of a web page. This tightens up security and ensures the integrity of a website. The sheer amount of content management systems available makes it hard to decide which one to use for a project. After reviewing some of the available open source CMS, I selected the following three systems for further investigation: DjangoCMS, Drupal, and Joomla.

3.3.1 DjangoCMS

DjangoCMS is the most recent CMS project of all three. It is a python based framework and has a clear and easy to use API. It offers an intuitive and user-friendly interface which makes it easy for maintainers to edit content. The main problem with this platform is that the available resources in the form of tutorials or books on this CMS are limited. Additionally, when I reviewed DjangoCMS the community behind the platform was still small and the risk for API breaks was pretty high due to the early stage of the system.

3.3.2 Joomla

Joomla is a PHP-based content management system. It has a large and active user base and is heavily customizable because of many available extensions. When the thesis started, the Joomla platform had many resources available for the branch 2.X

but the newer branch 3.X was already released. Using the outdated version was not an option for this thesis. In version 3.X the API has changed drastically. So a significant amount of the existing tutorials and books were not valid anymore which was a huge drawback for this platform.

3.3.3 Drupal

Drupal 7 was the last CMS taken into further consideration. The system is written in PHP and is also very extensible. Due to its mature state, Drupal also offers a highly stable API and many resources are available for this platform. But there is one disadvantage compared to Joomla and DjangoCMS and that is the complexity of the API. It was obvious, that if it were chosen for the thesis project, it would be necessary to invest a considerable amount of time to understand this platform. Nonetheless, after discussing these content management systems with my thesis supervisor, I decided to use Drupal for the updated probeBase web resource. The well-written documentation, the stable API and the in-house experience with Drupal at CUBE were the drivers behind this decision.

3.4 Choosing the right sequence matching tools

The goal of the probeBase database is to support researchers in their efforts to find information on specific probes and primers. Search results are the basis for further research. Therefore, accuracy and plausibility are essential qualities of the software tools used in this web service. To meet these requirements, only established and actively maintained tools, like the ones discussed in chapter 2, have been considered for the reimplementation of the database. The web resource needs to be able to deliver results for a search query within a reasonable amount of time. It is not acceptable that the information retrieval takes hours before completion. Furthermore, the website has to handle a high number of user queries simultaneously. After examining a couple of different sequence alignment tools, two have been chosen for the backend of the new probeBase.
To match short probe or primer sequences stored in the database with user-submitted

sequences, VMATCH is used. The tool is a non-heuristic aligner and guarantees to find the best possible probe candidates for a given input sequence available in the database. Tests showed that VMATCH outperforms BLAST and BLAT in retrieval times. BLAST generates seeds by creating k-long words, also known as k-mers, of the sequence stored in the database. These k-mers are indexed in a table. The k-mers of a query sequence are then used to retrieve the location of a particular k-mer in the database sequence. A different approach to create seeds is to extend k-mers until a mismatch occurs. These so-called maximal extended matches (MEMs) have the advantage of returning only a single seed for every stretch of exact matches and are therefore computationally favourably. The classical approach of finding MEMs between two sequences is to create a concatenated sequence S#P, indexing the resulting string in a suffix tree, and searching for maximal repeats that span the special character # (Gusfield, 1997). Indexing both strings is costly regarding space consumption because a significant amount of memory is occupied by the suffix tree. Recent work has focused on using the SA, a space-efficient alternative to the suffix tree, to further decrease the memory occupied by the index (Manber and Myers, 1993). Augmented with additional information, the 'enhanced' suffix array (ESA) provides the same functionality as the suffix tree and can be used to find MEMs between sequences (Abouelhoda et al., 2004).

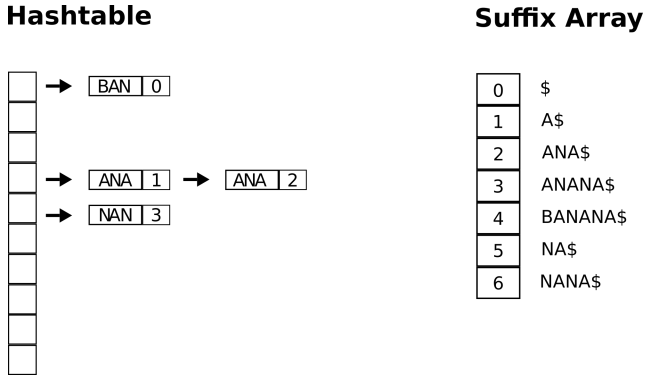


Figure 3.10: Comparision of the search strategies between BLAST (left) and VMATCH (right). BLAST generates seeds by creating k-long words, also known as k-mers, of the sequence stored in a hashtable. VMATCH stores all suffixes of a sequence in a suffix array. To find suffixes, a binary search is done through this data structure.

Furthermore, it supports standard formats for the input sequences, like FASTA or Genbank, and the search output can be received either as an easy to parse report or as an XML file. This provides a lot of flexibility for post-processing the data. VMATCH also allows specifying the exact number of the permitted mismatches between two strings, enabling users more fine-grained database queries.
In microbiology, amplicon sequencing is a widely adopted approach to study microbial communities. Amplicons are DNA or RNA fragments. Through next-generation sequencing techniques, large numbers of these reads can be collected simultaneously. One major drawback of sequencing techniques like Illumina MiSeq is that they produce reads with less than 500 nucleotides in length. These short rRNA reads are unfavourable for the probe matching process because they limit the selection of complementary oligonucleotides. To overcome this limitation a new experimental feature, the proxy match, is added to probeBase. This tool tries to find the near full-length rRNA for submitted amplicons. If successful, the corresponding sequence is used as a proxy for them. ProbeBase then tries to find all suitable probes or primers for this proxy in the database. It turned out that using VMATCH for this task is not applicable since this aligner is not optimised for sequences of this size. Among all tested tools, LAST is the best choice. It is sensitive enough to locate only partly matching amplicon reads with full-length rRNA sequences. Furthermore, the tool is fast, it can find proxy candidates in under one minute and the overall quality of the results is outstanding. This makes the tool ideal for the new proxy feature.

3.5 Setting up the Drupal environment

The whole development was done in an isolated virtual Linux environment with a running web-stack, consisting of Apache, PHP and MariaDB and was provided by the CUBE team. To set up a new Drupal instance, I downloaded the latest compressed archive of Drupal 7 from the project website http://www.drupal.org. Then I uncompressed and moved it to its destination on the web server. Before continuing with the actual installation, it was necessary to create a new empty database for Drupal. I also had to create the file settings.php in the corresponding sites folder which holds all relevant database configuration settings.

```
$databases = array (
'default' =>
array (
'default' =>
        array (
                'database' => 'pro_drupal',
                'username' => 'pbDrupal123',
                'password' => 'password123',
                'host' => 'localhost',
                'port' => '',
                'driver' => 'mysql',
                'prefix' => '',
        ),
),
);
```

Code 3.6: Example of a settings.php file. All databases necessary for the Drupal installation are configured here.

A template of this file can be found in the default sub-folder, named default.settings.php. I moved it to the corresponding sites folder and renamed it to settings.php. It was necessary to change the permissions of this file temporarily to writeable. In the final step, before continuing with the installation procedure, the directory "files" is created. After these steps, the Drupal 7 installation script can be executed to start the installation by pointing the browser to the Drupal installation, e.g. http://www.example.com or http://www.example.com/drupal.

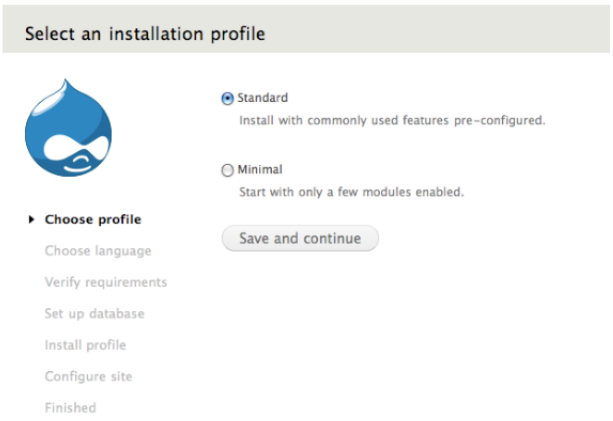


Figure 3.11: Drupal installation page.

3.6 Module development

It was planned to implement the new probeBase in the form of various Drupal modules passing data back and forth to perform different tasks. Drupal modules are used to customise and extend the functionality of the framework. Every default Drupal installation comes with a core set of modules, providing the base functionalities of the CMS. Furthermore, it is possible to create custom modules to enhance Drupal for specific needs. There are three different types of modules [43]:

1. Core modules: These are the default modules shipped with Drupal and are approved by the core developers and the community.
2. Contributed modules: They are written by the Drupal community and shared under the GNU Public License (GPL).
3. Custom modules: The modules are built for a particular use-case and therefore specific to the site they are installed on.

3.6.1 Modules structure

Drupal modules are organised in folders and contain at least two files: a ".module" and a ".info" file. The ".module" file holds all the logic while ".info" files provide all meta information, like name, version, description etc. This information is used to identify and describe the module in the administration interface of Drupal. If a new one is added, it appears in the modules list of the admin interface. To make it available to the web page it needs to be activated. All modules, like Drupal itself, are written in the programming language PHP. Before programming the modules, I analysed all the requirements and identified the following five features the application should provide:

- Search function: Users want to find information on probes based on sequence, name, authors, references, etc.
- Matching function: Users want to match submitted sequences with probes and primers stored in the database.

- Proxy matching function: Users submit short reads and try to find proxy sequences which are then used to detect suitable probes.

- List functionality: Users want to request dynamic lists of probes meeting specific criteria, e.g. probes successfully used for FISH.

- Administration: The probeBase staff needs to maintain the resource.

The probeBase performs two core tasks, as the concept above indicates: requesting information from the database directly via search forms or sequence aligner and displaying the found results to the user. These two tasks are vital in my module concept. I decided to build all components around this.
A search module provides all input fields so the users can query the database for specific oligonucleotides. Furthermore, via the probe matching tool or the new proxy tool researchers can find probes, matching their submitted sequence data. To a certain degree, all the results, coming from the web resource share commonalities. Therefore, I decided to develop a module whose only purpose is to display results data to the users. Aside from the lower code complexity, this also avoids unnecessary code duplication.
The search module, the proxy tool and the probe matching tool perform different tasks and depend on various toolsets in the backend. Hence, these features have been moved into separate modules. The same applies to the submission, the export and the administration module.
Giving users a detailed report view on probes is also a detached module. Even though it would be possible to include this function into the results module, it appeared more reasonable to me, assigning this task to a separate module. To share methods among all probeBase modules, a particular kind of module has been created. It serves as a small function library and gives shelter to those functions which conceptually do not fit in any other module. To pass data between these components, session variables are used. They exist only as long, as the user session lasts. Even though they are not declared in the module itself, they can be used as a local variable.

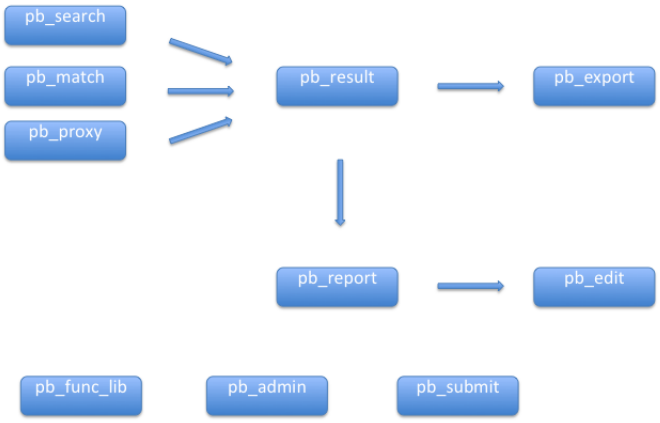


Figure 3.12: Overview of all developed Drupal modules and how they are connected to each other.

3.6.2 Hooks

Modules do not only need to be able to communicate with each other, but they also need to interact with the Drupal core system. This is accomplished with so-called "hooks". "Hooks" are PHP functions with a defined set of parameters and a specific return type. There are hundreds of different "hooks" available in the Drupal core API. The two most important hooks used for the new probeBase are: hook_menu and hook_form. hook_menu allows modules to define URL paths which are used to trigger callback function if certain URLs are requested. For example, to register a path "pb_search" the following code is used:

```
function pb_search_menu() {
        $items = array();
        $items['pb_search'] = array(
                'title' => t('Search probeBase'),
        'page callback' => 'pb_search_form',
        );
        ...
return $items;
}
```

Code 3.7: Example of a URL path registered in "hook_menu".

If "http://probebase.csb.univie.ac.at/pb_search" is requested, a defined callback, "pb_search_form", is invoked and returns a search form to the user. If the callback function depends on input parameters, the following pattern is used:

```
function pb_search_menu() {
        $items = array();
        $items['pb_search/%'] = array(
        'title' => t('Search probeBase'),
        'page callback' => 'pb_search_form',
        'page arguments' => array(1),
        );
        ...
return $items;
}
```

Code 3.8: Example of "hook_menu" passing the callback function one argument.

The '%' part of the path is a wildcard. If "http://probebase.csb.univie.ac.at/pb_search/123" is requested, the value 123 is passed as a parameter to the callback function. To allow certain paths to be accessed only by logged-in users, the attribute 'access callback' is defined within the item block:

```
function pb_admin_menu() {
        $items = array();
        $items['pb_admin/'] = array(
        'title' => t('Search probeBase'),
        'page callback' => 'pb_search_form',
        'access callback' => 'user_is_logged_in'
);
...
return $items;
}
```

Code 3.9: Use of "access callback" attribute in hook_menu.

Many of the modules built for the new probeBase contain some kind of forms. The hook "hook_form" provides functions to process and present them.

```
function pb_search_form() {
        return drupal_get_form('pb_search_form');
}
```

Code 3.10: Usage of "hook_form" in the "pb_search module".

The hook returns a form array with all elements the form contains. The actual definitions of them are done in the function "pb_search_form" which has the following structure:

```
function pb_search_my_form($form, &$form_state) {
                $form = array();
                        $form['search_sequence']['seq_query'] = array(
                                '#type' => 'textfield',
                                '#title' => t('probe/primer sequence'),
                                '#attributes' => array(
                                'placeholder' => t("Enter sequence in 5'-3' orientation"),
                                )
                );
                        ...
                return $form;
}
```

Code 3.11: Structure of "pb_search_form".

This method returns an array containing all the form elements. In the example above an input textfield ('#type' => 'textfield') with the label 'probe/primer sequence' ('#title' => t('probe/primer sequence')) and with the placeholder text 'Enter sequence in 5'-3' orientation' ('#attributes' => array('placeholder' => t("Enter sequence in 5'-3' orientation")) is defined.

3.6.3 Database API

To query the database, two different approaches are used in probeBase: Static queries with *db_select()* and dynamic queries with *db_select()*. *db_select()* offers an abstraction-layer for a unified database access which is built dynamically by Drupal using a query object.

```
db_set_active('probebase');
$query = db_select('probe', 'p');
$query->fields('p',array('probe_sequence'));
$query->condition('probe_sequence', $submission['probe_sequence'], '=');
$results = $query->execute()->fetchAll();
db_set_active();
```

Code 3.12: Dynamic database query.

In the code snippet above, the active database is changed to 'probebase'. Then the field "probe_sequence" in table probe is selected. A condition statement drops all probes in the probe table except for those matching the sequence in the array "$submission". After fetching all rows the active database is switched back to the default database. *db_select()*, on the other hand, executes a query string to get data from the database.

```
db_query('select probe_sequence from probe where probe_sequence = :probe_sequence',
                array(':probe_sequence' => $submission['probe_sequence'])
);
```

Code 3.13: The same query as in the *db_select()* example above. This time in form of a query string.

Using *db_select()* is an elegant and convenient way to create database queries. On the other hand, benchmarks showed, that queries made with *db_query()* are faster than those with *db_select()*. I decided to limit the use of *db_query()* to performance-critical areas of probeBase.
All databases need to be configured in the settings.php to access them. Drupal always uses the active database regardless which method is used. The default active database is the one defined during the Drupal installation [44].

```
        $databases = array (
                'default' =>
                        array (
                                'default' =>
                                        array (
                                                'database' => 'pro_drupal',
                                                'username' => 'pbDrupal123',
                                                'password' => 'password123',
                                                'host' => 'localhost',
                                                'port' => '',
                                                'driver' => 'mysql',
                                                'prefix' => '',
                                        ),
                        ),
                'database2' =>
                        array (
                                'default' =>
                                        array (
                                                'database' => 'myDB',
                                                'username' => 'user123',
                                                'password' => '1234',
                                                'host' => 'localhost',
                                                'driver' => 'mysql',
                                                'prefix' => '',
                                        ),
                        ),
        );
```

Code 3.14: Example of a settings.php file containing the configuration for two different databases: "default" and "database2".

To activate "database2", in the listing above, developers need to call the function *db_set_active()* with the database name as the argument:

```
db_set_active(''database2'');
```

Code 3.15: Switching to "database2".

By invoking *db_set_active()* without any arguments it possible to change back to the default database.

3.6.4 ProbeBase modules

This section describes all modules which were created for this thesis. By explaining the purpose and the logic behind every component, an in-depth look into the internals of probeBase is provided. Each module description begins with a list of its URL paths, used to access its functionality, followed by the explanation of the module.

pb_admin

URL Path	Description
pb_admin_main	Overview of the admin interface.
pb_admin_probes	The admin interface to manage all oligonucleotides.
pb_admin_arrays	The admin interface of all microarrays.
pb_admin_categories	The admin interface of all categories.
category_edit/CATEGORYID	admin interface to edit a selected category.
array_edit/ARRAYID	The admin interface to edit a selected array.

Table 3.1: The table contains all URL paths of the "pb_admin" module. CATEGORYID is the unique category ID of the selected category. ARRAYID is the unique array ID of the selected array.

In the past, maintainers had to edit data within the database system which was not convenient. The new admin interface allows data manipulation directly on the probeBase web page and makes the overall data management easier. It can be accessed through the main menu and is split into several independent sub-pages. A drop-down menu gives users the opportunity to choose between them.

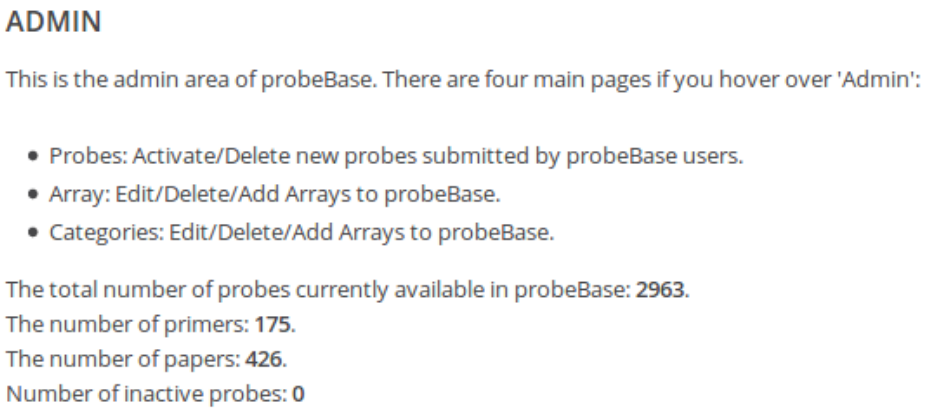


Figure 3.13: Main admin area.

The main admin area provides a small description of all functionalities available in the admin interface. Furthermore, it also supplies the status not only of all probes but also of the database itself.

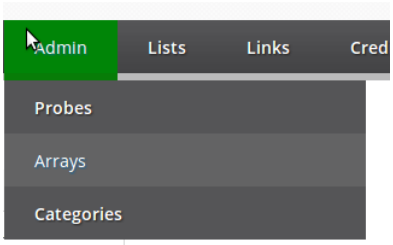


Figure 3.14: Admin menu.

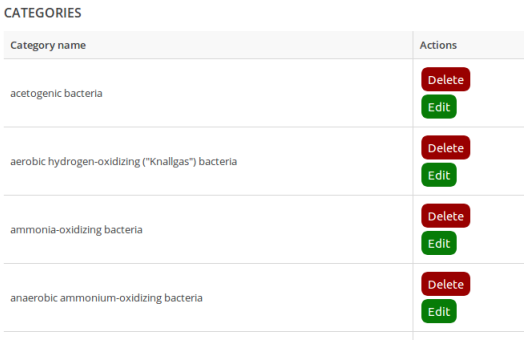


Figure 3.15: Categories admin area.

The "categories" section provides functions to add new categories and to delete or modify all available ones. The same is possible for arrays and probes in the probeBase.

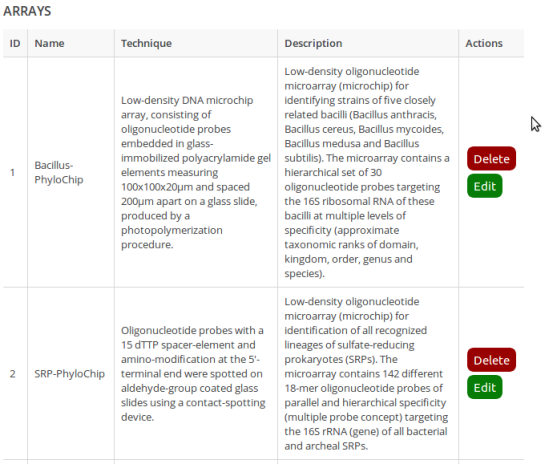


Figure 3.16: Arrays admin area.

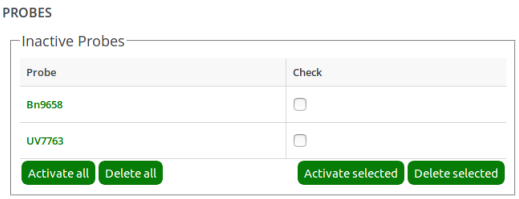


Figure 3.17: Probes admin area.

All of these sub-sections are accessed via URL paths which are registered in the "hook_menu" function:

```
function pb_admin_menu() {
        $items = array();
        // administration page
        $items['pb_admin_main'] = array(
                'type' => MENU_CALLBACK,
                'page callback' => 'drupal_get_form',
                'page arguments' => array('pb_admin_main'),
                ...
        );
        // administration of probes
        $items['pb_admin_probes'] = array(
                'type' => MENU_CALLBACK,
                'page callback' => 'drupal_get_form',
                'page arguments' => array('pb_admin_probes'),
                ...
        );
        // view available categories in probeBase
        $items['pb_admin_categories'] = array(
                'type' => MENU_CALLBACK,
                'page callback' => 'drupal_get_form',
                'page arguments' => array('pb_admin_categories'),
                ...
        );
        ...
```

Code 3.16: Registered URLs in the hook menu.

If the path "pb_admin_categories" is requested, the corresponding callback, in this case, "pb_admin_categories", is executed and returns the administration interface for the categories. This function has the following structure:

```
function pb_admin_categories($form, &$form_state) {
        if(isset($form_state['storage']['confirm'])) {
        return confirm_form($form,
        'Please Confirm!','pb_admin_categories',
        'Do you really want to delete this category?',
        'Yes!',
        'Cancel');
        }
        // switch to probeBase database
        db_set_active("probebase");
        // call db_select() to get all categories in database
        ...
```

```
        $cat_query = db_select('category', 'c');
        $cat_query->fields('c',array('categoryid','category_name'));
        $cat_query->orderBy('category_name','ASC');
        $cat_results = $cat_query->execute()->fetchAll()
        ...
        // loop over all returned categories
        foreach($cat_results as $category){
                // create html code for table row
                $form['category_entry_'.$categoryid] = array(
                        '#type' => 'markup',
                        '#markup' => "<tr><td>".$category_name."</td>",
                );
                $form['delete_category_'.$categoryid] = array (
                        '#type' => 'submit',
                        '#value' => t('Delete'),
                        '#prefix' => '<td><div class=danger_button>',
                        '#suffix' => '</div>',
                        '#name' => $categoryid,
                        '#submit' => array('pb_admin_categories_submit'),
                );
                $form['edit_category_'.$categoryid] = array (
                        '#type' => 'submit',
                        '#value' => t('Edit'),
                        '#name' => $categoryid,
                        '#suffix' => '</td></tr>',
                        '#submit' => array('pb_admin_categories_submit'),
                );
        }
        ...
        return $form;
}
```

Code 3.17: Categories administration callback function.

All other admin areas, such as probes or microarrays, are implemented in the same manner. To prevent unwanted deletion of data, a modal dialogue is added via the Drupal "confirm_form" function in all admin forms. The function is triggered through the 'if(isset($form_state['storage']['confirm']))' condition in the module code. This persistent variable is set 'TRUE' when the "Submit" button is pressed for the first time. Then the form gets rebuilt. This time, because of the '$form_state['rebuild'] = TRUE' statement, a confirmation dialogue appears. If the user approves it, all changes are applied to the database. Otherwise, if it is cancelled, a second callback function is executed, like "pb_admin_categories" in the example above.

```
$items['pb_admin_probes'] = array(
   ...
   'access callback' => 'user_is_logged_in',
   ...
);
```

Code 3.18: Example for using "access callback" attribute in the probes administration path.

Only logged in users can see and access the admin menu which is accomplished by using the "access callback" attribute for every registered URL path (see Listing 3.18).

pb_report

URL Path	Description
pb_report/DATATYPE/PROBEID	Shows detailed description of selected probe.

Table 3.2: All available paths for the module "pb_report". DATATYPE is the name of the data type to request. PROBEID is the unique probeid of the selected probe.

This module creates reports of probes, references and microarrays. Only one path is available in this module which requires two input arguments. The first one is the requested data type. Allowed values are "probe", "reference", and "probemicroarray". The second argument is the database ID of this entity to identify the corresponding data row in the table.

```
http://probebase.csb.univie.ac.at/pb_report/probe/43
```

Code 3.19: Example of requests for the report view of the probe with ID 43.

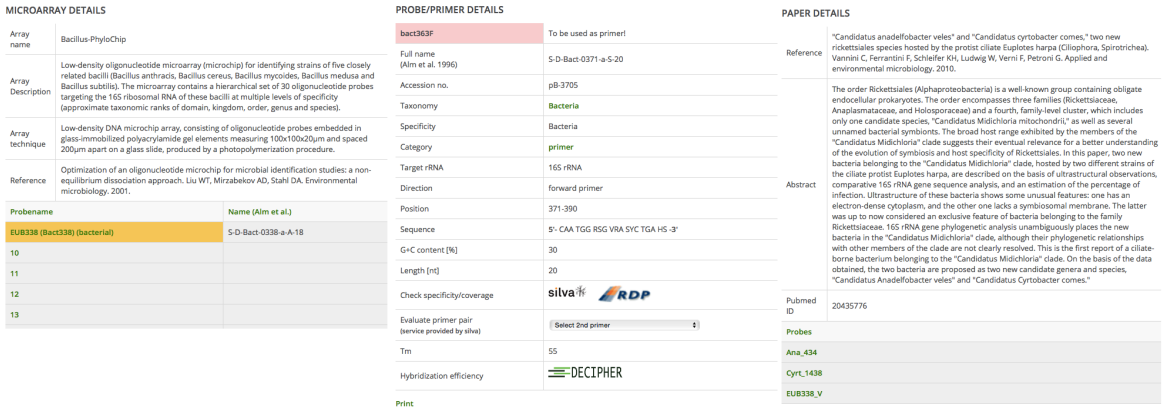


Figure 3.18: Examples of all report types (left to right): "microarrays", "probes and primers" and "references".

Microarray reports contain more detailed information, including name, description, technique and references. A list at the bottom holds all oligonucleotides used on

this microarray. Probe and primer reports reveal information on any selected oligonucleotide in the database. Aside from its name, sequence, target specificity or position in the RNA, the report view also offers the possibility to request external services like SILVA, RDP or Decipher. These services are accessed by passing parameters in URLs.

```
http://decipher.cee.wisc.edu/ProbeMelt.html?probe=CTAATGCACCC&target=GGTTGTTGGGT&autosubmit=yes
```

Code 3.20: Example request URL for the Decipher service.

The bottom of every probe report holds the link to a more printer friendly version of the page. This is accomplished by the Drupal print module by passing the report path and the probe ID to a specific URL.

```
http://DRUPAL_WEBSITE/print/pb_report/probe/3
```

Code 3.21: Passing the report page of the probe with ID 3 to the print module.

The references report provides information on all oligonucleotide papers stored in the database. The table contains the citation, the abstract as well as the PubMed ID, of the corresponding reference. It also lists all probes and primers in the database, which are related to the selected paper.

pb_edit

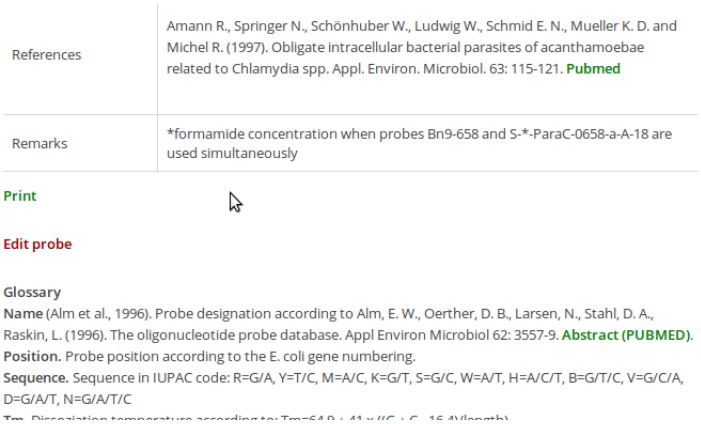


Figure 3.19: Edit link on probe report page.

URL Path	Description
pb_edit/TYPE/PROBEID	Edit the corresponding probe.
pb_edit/TYPE/PROBEID/delete	Delete the corresponding probe.

Table 3.3: All URLs "pb_edit" provides. Allowed values for TYPE: 'probes' or 'category'. PROBEID is the unique probeid of the selected probe.

This module enhances the "pb_report" module. Only logged-in users have an edit link on the bottom of the probe report view. Through it, the maintainer can access the edit form of the oligonucleotide directly within the report. This allows probeBase staff to quickly correct or enhance probe entries when they notice errors or missing data. Like in the "pb_admin" module before, a modal dialogue was added via the Drupal "confirm_form" function to prevent deletion of probes by accident.

```
//request callback function with probe ID as argument
  $items['pb_edit/%/%'] = array(
                . . .
  );
//delete probe callback function with probe ID as argument
  $items['pb_edit/%/%/delete'] = array(
                . . .
  );
```

Code 3.22: Registered URL paths.

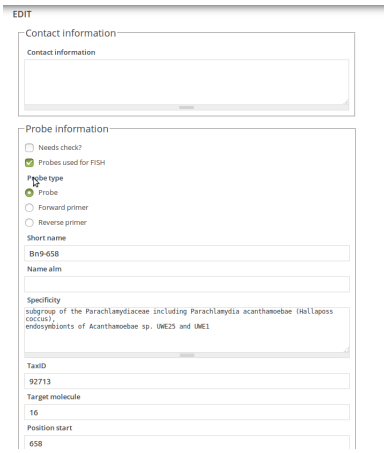


Figure 3.20: Probe edit form.

The module accepts two input arguments. The first argument is the type of data to edit, such as "probes". The second one is the probe ID.

pb_func_lib

This is the only probeBase module without any URL path. It is simply a collection of helper functions and global variables and also contains binaries and database files used by other modules.

File/Folder	Description
js/	This folder contains all java script files used throughout probeBase.
last-604/	This folder contains the binaries and all related files for the LAST alignment tool.
silva/	This folder contains a copy of the SILVA rRNA sequence database.
TransDNA	The symbol mapping file, used by VMATCH.
vmatch	The binary for the VMATCH alignment tool.
vmatch.lic	The licence key for the VMATCH binary.
mkvtree	Binary for the mkvtree tool. Used by VMATCH to create the persistend index files.
pbFaidx.py	Python helper class to read FASTA-files. Used by pbProxyMatch.py script.
pbProbeMatch.py	Python helper script. Reads user input and executes VMATCH. Returns parsed VMATCH results.
pbProxyMatch.py	Pytohn helper script. Reads user input and executes LAST and VMATCH. Returns parsed results.
pbSequenceInput.py	Python helper class. Reads user input. Cleans and translates it to valid input for the pbProbeMatch and pbProxyMatch scripts.
pb_func_lib.info	.info file for the module
pb_func_lib.module	Module source code.
pB_batch_example_file.txt	Example file for batch upload.

Table 3.4: Files and folders in the module "pb_func_lib".

The helper functions in the .module file solve many tasks, such as rebuilding the suffix array for VMATCH, generating the reverse sequence complement of a sequence, fetching PubMed entries, etc. The getReverseComplement() function, for example, is used in the reports module. It can be invoked like a local function in the report module:

```
...
$reverse_complement = getReverseComplement(PROBE_SEQUENCE);
...
```

Code 3.23: Invoking the getReverseComplement function.

Many modules have the dependency on individual paths or files in common. To avoid redundancy throughout the code base, "pb_func_lib" contains global variables, storing paths or often used HTML patterns, which can be re-used by other components. An example of such a global variable is the pager HTML code used by the tablesorter module. This module renders all data tables in probeBase and needs this HTML code to append the pagination functionality and control elements to every table.

```
<div id=tablesorter_pager>
        <a class='first'/><<</a>
        <a class='prev'/><</a>
        <span class='pagedisplay'></span>
        <select class='pagesize' title='Select page size'>
                <option selected='selected' value='50'>50</option>
                <option value='100'>100</option>
                <option value='150'>150</option>
        </select>
        <a class='next'/>></a>
        <a class='last'/>>></a>
</div>";

//Store it as a global variable
variable_set('pager',$pager);
```

Code 3.24: Tablesorter pager HTML code.

If developers want to access this variable, the following function call is made:

```
$pager = variable_get('pager');
```

Code 3.25: Call global variables in Drupal

The module also contains additional resources, like the VMATCH suffix array files, the SILVA database files, all the javascript code used on the web page as well as the binaries for VMATCH and LAST. Furthermore, two python scripts are stored in this folder: pbProbeMatch.py and pbProxyMatch.py. The pbProbeMatch.py script is used for the probe matching function. It first checks and cleans the user input and executes VMATCH with all passed in arguments. The results are parsed and returned to the caller.

To perform its task, the script depends on four arguments:

```
$ pbProbeMatch.py sequenceFilename editDistance probeDBOption rootPath
```

Code 3.26: pbProbeMatch script command signature.

"sequenceFilename" defines the filename of the file containing the query sequences. "editDistance" is the number of allowed mismatches. "probeDBOption" allows to choose between probes or primers. "rootPath" configures the path to the "pb_func_lib" module. The pbProxyMatch.py script, on the other hand, is used for the proxy matching feature. Like the pbProbeMatch script, it also checks and cleans the user input first. Then it executes LAST and prepares a FASTA-file of non-redundant proxy sequences. For every sequence in this file, it performs a probe match query and returns matched probes to the caller. This script depends on six arguments:

```
$ pbProxyMatch.py equenceFilename editDistance  silvaDB probeDBOption proxyThreshold rootPath
```

Code 3.27: pbProxyMatch script command signature.

"sequenceFilename" is the filename of the file containing the query sequences. The "editDistance" determines the number of allowed mismatches. "silvaDB" is the name of the SILVA database used for the proxy match. "probeDBOption" allows to choose between probes or primers. "proxyThreshold" configures the minimum number of percentage matching between submitted sequence and proxy candidate. "rootPath" is the path to the "pb_func_lib" module.

pb_match

URL Path	Description
pb_match	Access probeBase match form.

Table 3.5: pb_match URL paths.

This module contains the logic of the probe match feature. Users can upload a sequence file or they can paste sequence data into the text field on the page. It is essential that the sequence data is submitted in the FASTA-format.

The user can determine the maximum number of mismatches between the entered sequences and the matching oligonucleotides. They can also restrict the query to

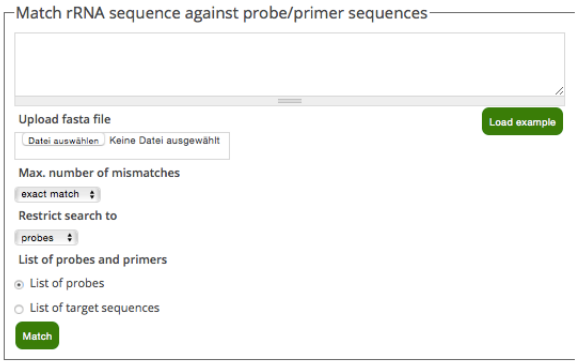


Figure 3.21: ProbeBase match form.

probes or primers exclusively. Finally, it is possible to configure how the results should be presented to the user. There are two options available. The first one groups per found probe. Every probe or primer contains a list of the submitted sequence which is targeted by this oligonucleotide. The second option groups the results list per user sequences. Every submitted sequence contains a list of suitable probes. If the probe matching process is started, the following steps are performed:

1. The module collects all user data from the form.
2. All the sequence data is saved into a temporary file on the web server. If a user decides to paste sequence data into the text field and to upload a sequence file, all sequences get merged.
3. The "pbProbeMatch.py" helper script is executed. It takes all input data and settings, cleans the sequence data and runs VMATCH.
4. The results from VMATCH are stored in a local variable.
5. "pb_match" loops through this variable and transforms it into a nested associative array. Its structure depends on the grouping option the user did choose in the match form. After this step is completed, the array is saved into a session variable.
6. Finally, "pb_results" is called to display the data.

pb_proxy

URL Path	Description
pb_proxy	Access the probeBase proxy match form.

Table 3.6: "pb_proxy" URL paths.

This module contains the code for a brand new probeBase feature which has the purpose of finding the near full-length rRNA equivalent of every query sequence provided by the user. Discovered sequences are then matched against the probeBase database to find suitable oligonucleotides.

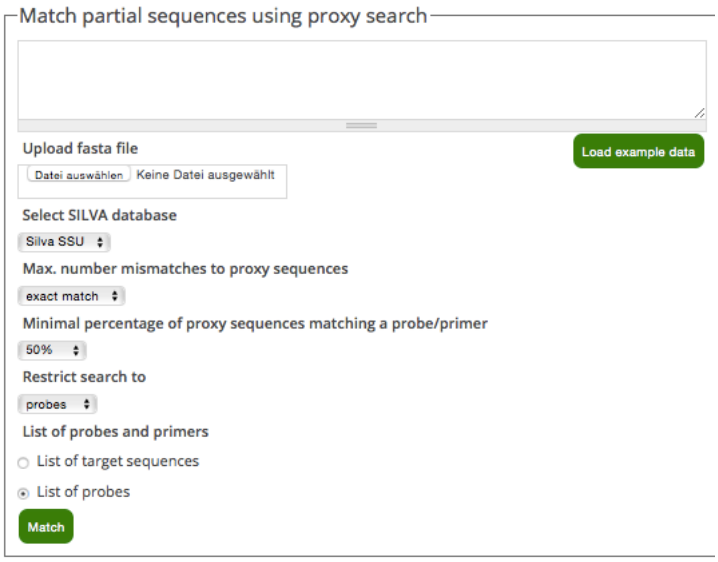


Figure 3.22: ProbeBase proxy match form.

The structure of this module is similar to the "pb_match" module. A user provides sequence data by pasting it into the text field or by uploading a file containing the sequence data. Again the FASTA-format is mandatory. To determine the SILVA database, which contains possible proxy candidates, users can choose between SILVA LSU, the large subunit, or SILVA SSU, the small subunit. It is also possible to specify a maximum number of mismatches between the proxy and probe sequence. The third drop-down menu determines the minimum number of percentage a proxy sequence

has to match a submitted one. Users have to set their search to either probes or primers. The last option in the form configures how the results should be grouped. If the proxy match run is finally executed, the following steps are performed:

1. First all form data is collected.
2. Next, all sequences, either pasted into the text field or uploaded as a file, are saved into a temporary file on the web server.
3. The "pbProxyMatch.py" script gets executed. It takes all sequence data, cleans it and executes LAST with the options selected in the proxy match form. All found proxy sequences are then used as the input data for a probe match query to get suitable oligonucleotides.
4. The results are saved in a local variable.
5. "pb_proxy" then loops through this variable and transforms the data into a nested associative array. The structure depends on the grouping option the user selected in the proxy match form. To pass the data to the results module, this array is saved in a SESSION variable.
6. At the end, "pb_results" is called to display the results data.

pb_results

URL Path	Description
pb_results/TYPE	Data tables depending on passed TYPE string.

Table 3.7: Available paths for "pb_results". Possible values for TYPE: 'pbproxy', 'pbmatch', 'silva', 'probespapers', 'targetsites', 'taxon', 'targetorganism', 'sequence', 'categories', 'listall', 'listarray', 'listprimer', 'listreferences', 'listinsitu'. See table 3.8 for a detailed description.

This module generates all data tables in probeBase. The URL parameter determines which of them is requested. There are 14 different target tables available, each with a specific structure and appearance. The actual data is either passed in via session variables or retrieved within the module itself.

type	description
pbproxy	Results list for proxy match query.
pbmatch	Results list for probe match query.
silva	List of all probes matching sequence or name.
probespapers	Lists all papers for a certain probe.
targetsites	List of all probes or primers targeting a certain RNA region.
taxon	All probes with a certain taxon in their lineage.
targetorganism	Table of all oligonucleotides with a certain specificity.
sequence	Oligonucleotides with a certain sequence.
listarray	Lists all available microarrays in the database.
listprimer	Lists all available primers in the database.
listall	Lists all probes and primers in the database.
listinsitu	Lists all FISH approved oligonucleotides available in the database.
listreferences	Lists all available references in the database.
categories	Lists all available categories in the database.

Table 3.8: All available data table types in "pb_results".

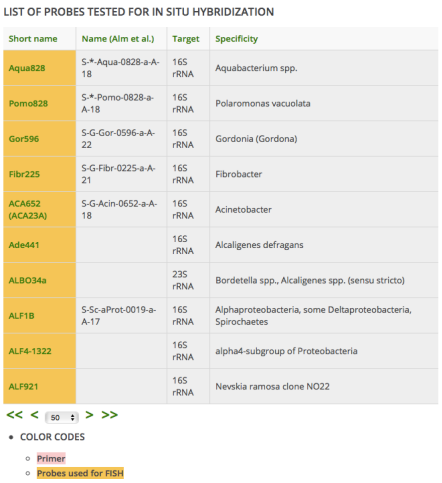
```
function pb_results_menu() {
        $items['pb_results/%'] = array(
                'page callback' => 'pb_results_list',
                'page arguments' => array(1),
                'access arguments' => array('access content'),
                'type' => MENU_CALLBACK,
        );
  return $items;
}

function pb_results_list($arg1, $arg2) {
        ...
        if($arg1 == 'pbproxy'){
                ...
                foreach($unique_queryseq as $seq){
                        // prepare html code for data table
                }
                ...
        }
        if ($arg1 == 'pbmatch'){
                ...
                foreach($unique_queryseq as $seq){
                        // prepare html code for data table
                }
                ...
        }
        ...
}
```

Code 3.28: "pb_results" code structure.

Within these control statements, the corresponding HTML tables are generated.

Not all requests made to the module depend on the same input data. The "pbproxy" and "pbmatch" targets are used for the probe matching and proxy matching feature. The actual work of these tools is performed externally and the results get passed in as arrays.



LIST OF PROBES TESTED FOR IN SITU HYBRIDIZATION

Short name	Name (Alm et al.)	Target	Specificity
Aqua828	S-*-Aqua-0828-a-A-18	16S rRNA	Aquabacterium spp.
Pomo828	S-*-Pomo-0828-a-A-18	16S rRNA	Polaromonas vacuolata
Gor596	S-G-Gor-0596-a-A-22	16S rRNA	Gordonia (Gordona)
Fibr225	S-G-Fibr-0225-a-A-21	16S rRNA	Fibrobacter
ACA652 (ACA23A)	S-G-Acin-0652-a-A-18	16S rRNA	Acinetobacter
Ade441		16S rRNA	Alcaligenes defragans
ALBO34a		23S rRNA	Bordetella spp., Alcaligenes spp. (sensu stricto)
ALF1B	S-Sc-aProt-0019-a-A-17	16S rRNA	Alphaproteobacteria, some Deltaproteobacteria, Spirochaetes
ALF4-1322		16S rRNA	alpha4-subgroup of Proteobacteria
ALF921		16S rRNA	Nevskia ramosa clone NO22

Figure 3.23: Example of a list of probes tested in FISH experiments.

By looping through them, the final data table is created and returned to the user. The target "probespapers" is responsible for handling the data from the "probe/primer names and papers" search. This allows users to search for probe names in combination with author names or keywords used in the references. The concatenation of these terms is done via two boolean operators: AND and OR. Users can choose between these two operators in the search form. The "pb_results" module considers the user choice and adapts the database query accordingly. Drupal provides two functions to assemble the queries with the corresponding operators: db_or() or db_and().

```
$query
        ->condition(db_or()
        ->condition('r.title', '%' . db_like($author_title_abstract) . '%', 'LIKE')
        ->condition(db_or()
        ->condition('r.abstract', '%' . db_like($author_title_abstract) . '%', 'LIKE')
));
```

Code 3.29: Example for a database query using db_or(). 'r' is the alias for reference table in database.

"Targetorganism" is used to process all the data passed in from the corresponding search form. In a first step, the entered term needs to be translated into the proper taxonomy ID. This is done via the function "getTaxID" which is available in the "pb_func_lib" module. The returned ID is then used to find all probes stored in the database targeting this exact taxonomy ID. The name of the target organism is used to find probes containing it in the specificity field of the probe table in the database. Furthermore, if a user also checks the tax up and tax down option in the search form, the complete taxonomic lineage of the taxonomy ID is assembled. This returns a collection of taxonomy IDs which are then used to query the database for all probes targeting any of these IDs. A particular path is "silva" for which the mandatory parameter is either the name or the sequence of a probe. This target is used by the external SILVA website to find probes by names or sequence. A sequence has to contain at least six characters and only a limited alphabet is allowed. Otherwise, the passed parameter will be treated as a probe name.

```
http://probebase.csb.univie.ac.at/pb_results/silva/atgcagtt
```

Code 3.30: Call "pb_results" with sequence data.

```
http://probebase.csb.univie.ac.at/pb_results/silva/Burkho
```

Code 3.31: Call "pb_results" with probe name.

The target site search form takes a location in the rRNA in the form of an integer number. It looks for any probe, targeting this position within a distance of 20 nt in both sides.
"Taxon" simply takes the name of the passed in taxonomic term and returns a list of probes targeting it. This is used in the oligonucleotides report page. By clicking any taxon in the created lineage, a list of all probes matching this taxon is returned.
Probes stored in the probeBase database can have two states: active or inactive. This state is determined in the probe table field "needs_check". If a probe is inactive, this value is set to 'TRUE'. Probes are excluded from results by checking the probe by the isNull() function from Drupal.

```
http://probebase.csb.univie.ac.at/pb_results/taxon/Bacillus-cereus-group
```

Code 3.32: Call "pb_results" - "taxon" endpoint with taxonomic term.

```
db_set_active('probebase');
$query = db_select('probe', 'p');
$query->fields('p',
        array('probeid','probe_name','name_alm','probe_sequence','target_rna',
              'position_start','position_end','needs_check','is_primer','specificity','insitu'));
$query->isNull('p.needs_check');
$query->condition('insitu', '1', '=');
$results = $query->execute();
db_set_active();
```

Code 3.33: Example for requesting all FISH probes in the database.

The sequence path maps the results from a VMATCH query to the corresponding probes by their IDs. All the data is then saved in a session variable, which is further used by the module "pb_result" to create the results data table to the users.

```
...
$query->condition('p.probeid',$_SESSION['search_sequence_results'],'IN');
$results = $query->execute();
...
```

Code 3.34: Database query with all probe IDs stored in the session variable.

pb_search

URL Path	Description
pb_search	Access probeBase search form.
autocomplete_pb_search	Autocomplete callback.

Table 3.9: "pb_search" URL paths.

This module contains the primary search form of probeBase. By requesting "pb_search", a page with four different search forms is returned, a target organism search form, a form to search a combination of probe names, authors and references, a probe sequence search form and a form to search for possible oligonucleotide target sites. If a query is issued, all form data and selected options are passed to the "pb_results" module, which then gets information from the database, compiles a data table and returns it to the user. This is true for all except one search form: the probe sequence search. Here the input data, which is a probe sequence, is used to align it with all the probe sequences in the database via VMATCH.

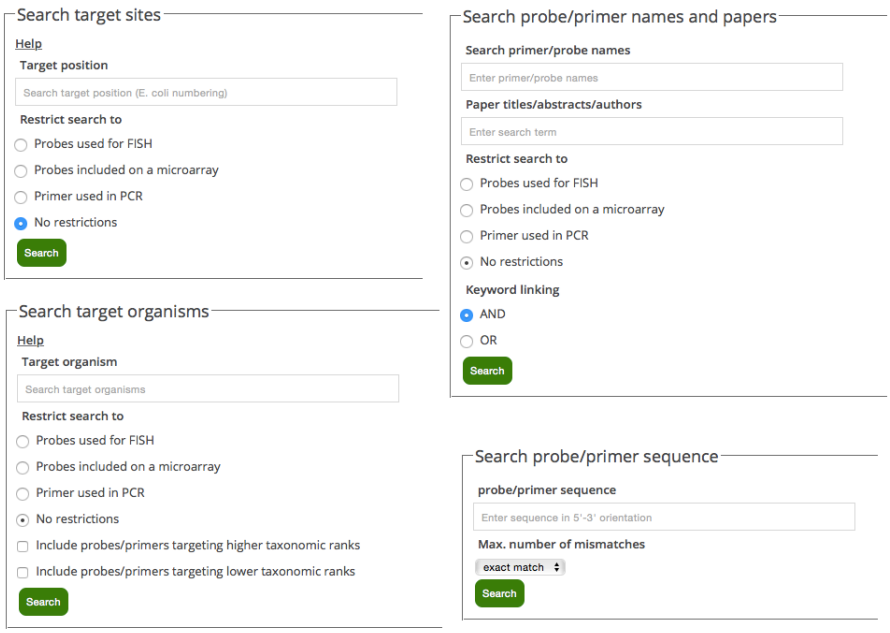


Figure 3.24: All available search forms.

After this is done, the results are also passed and processed in the "pb_results" module. The component "autocomplete_pb_search" is used by the target organism search form for the autocomplete function.

```
$form['search_targetorganisms']['targetorganism_term'] = array(
  '#type' => 'textfield',
  '#title' => t('Target organism'),
  '#attributes' => array(
    'placeholder' => t('Search target organisms'),
  ),
  '#autocomplete_path' => 'autocomplete_pb_search',
);
```

Code 3.35: Target organism input field with auto-complete callback function.

The input text field uses the 'autocomplete_path' attribute. Every time a user enters a character into the corresponding input field, the callback function of this path is executed. To avoid flooding the database with short, unspecific terms, a minimum input length was defined. The length of the entered string has to be larger than four before probeBase makes any database calls and returns suggestions to the user. All taxa containing this pattern are returned. The full taxon list is returned as a JSON formatted string. The taxonomic database contains two IDs corresponding

to "Bacteria". The ID 2 is referring to the superkingdom "Bacteria", while 629395, refers to "Bacteria Latreille", also known as a stick insect. That ambiguity can cause problems since probeBase is a microbial database. Therefore all database queries use exclusively the taxonomy ID 2.

pb_submit

URL Path	Description
pb_submit	Access probeBase submission form.
autocomplete_pb_submit	Autocomplete callback.

Table 3.10: pb_submit URL paths.

This form gives users the opportunity to contribute to the probeBase project by uploading new probes to the database. Two upload options are available, a single probe and a batch upload option. If the upload is successful, all submitted data is added to the database. In the table "probe" the field "needs_check" is set to 1, which means, that the corresponding probe is not yet active. To make it available to the probeBase users, maintainers need to activate it in the admin section. The batch upload parses the uploaded CSV file and adds the probe to the database. Again, to make the probe available to the public, it needs to be activated first.

pb_export

URL Path	Description
pb_export/TYPE	Get data from the corresponding table as in format TYPE.

Table 3.11: TYPE: 'xls', 'tsv'.

This module was created to export data from the "pb_results" data tables. ProbeBase users have the choice between two data formats: XLS-files or tab-separated-values. The data which is exported is passed in via session variables. Then this module simply returns an empty page with the corresponding HTTP-header set. For XLS-file the MIME-TYPE is "application/vnd.ms-excel" and for tsv-files it is "text/tab-separated-values". Finally, it uses the echo function to return the export data.

```
header("Content-type: application/vnd.ms-excel");
header("Content-Disposition: attachment;
        filename=pb_export.xls");
echo $export;
```

Code 3.36: Header content set in "pb_export" module.

3.6.5 3rd party modules

tablesorter

By adding the class "tablesorter" to the corresponding HTML table, it is transformed into a sortable, paginated table. The module automatically detects the data types, such as strings, dates, integers etc. of every column. Through pagination, long tables are split into several sub-tables, each accessible via the control elements at the bottom.

prevent js alterts

Prevent JS alerts suppress all javascript error messages on a Drupal web page. The autocomplete callback, which is used to enhance the user experience, throws errors in the form of alert dialogue messages if API calls fail to succeed. Since this is not important for the probeBase users and has a negative effect on the user experience, this module was installed to hide these messages.

print

The module print generates printer optimised versions of a web page. These optimized HTML pages are created via simple URL calls, e.g.:

```
www.example.com/print/pb_report/probe/3
```

Code 3.37: This generates a printer-friendly HTML site of the probe with the ID 3.

This module is used in the "pb_report" module. By providing this link on the bottom of every probe report, users can request a printer friendly report for every oligonucleotide.

probeBase
an online resource for
rRNA-targeted oligonucleotide probes
Published on *probeBase* (http://localhost/probebase)

Home >

Probe/primer details

Bn9-658	Tested for in situ hybridization.
Accession no.	pB-596
Taxonomy	Parachlamydiaceae; Chlamydiales; Chlamydiia; Chlamydiae; Chlamydiae/Verrucomicrobia group; Bacteria
Specificity	subgroup of the Parachlamydiaceae including Parachlamydia acanthamoebae (Hallaposs coccus), endosymbionts of Acanthamoebae sp. UWE25 and UWE1
Category	bacteria of medical or hygienical relevance
Target rRNA	16S rRNA
Position	658-675
Sequence	**5'-** TCC GTT TTC TCC GCC TAC **-3'**
G+C content [%]	56
Length [nt]	18

Figure 3.25: Printer friendly probe report.

3.6.6 ProbeBase theme

The theme for the probeBase resource was created by "pion websystems" and is adapted to meet all the requirements of the web service.

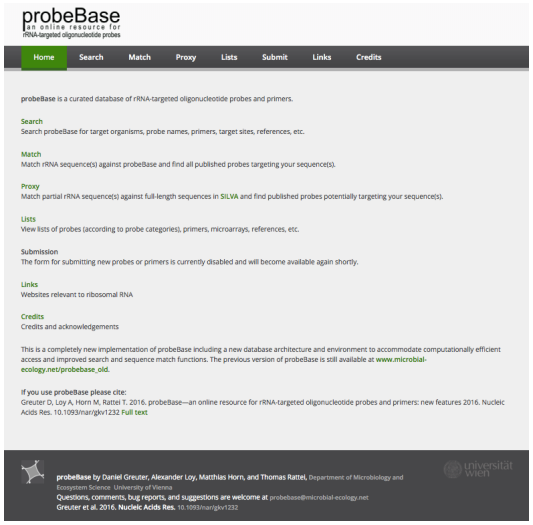


Figure 3.26: ProbeBase theme.

The layout of the theme has the following structure: on top of the page is the project logo. The main menu is below, followed by the content of the currently active menu item. Additionally, every page has also a footer at the bottom, providing links

to references and citation information. The theme is responsive and it, therefore, adapts to the available display size of the device the page is viewed with.

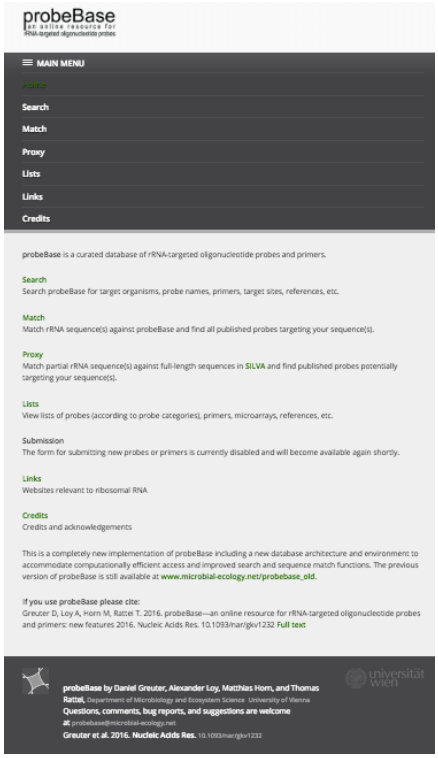


Figure 3.27: ProbeBase web site on a mobile phone. To save space, the main menu and the page content is stacked horizontally.

To customize the pion theme, the CSS-files in the theme directory have been extended with additional selectors and properties.

CHAPTER 4

Working with the new probeBase

The following chapter provides step-by-step guidelines on how researchers can use the probeBase web-service in their daily work. Typical use-cases, like searching for oligonucleotides within a specific target site or a specific target organism, are described as well as the enhanced probe matching tool. The chapter continues with the description of the brand new proxy matching feature and ends with a tutorial on how users can submit new probes to the resource. Every use-case is illustrated with the placeholder characters Alice and Bob since this is more convenient than repeating the phrase 'A user' in every application example.

4.1 Search the probeBase

The main feature probeBase offers is the search form. It helps its user-base to query the database for oligonucleotides and filter the results by certain requirements. These can either be specific target organisms, names, authors, references, sequences or target sites. The search form also contains detailed descriptions for two of the four search forms: target organisms and target sites. These explanations provide background information on the corresponding search form, how to optimise search queries and how

the users should interpret the results. The other two subforms, target sequence and probe names and authors, are self-explanatory and do not need further explanation. The following four sections demonstrate how to use this feature.

4.1.1 Probes targeting a certain organism

Use-case: *Bob wants to find all probes targeting the genus "parachlamydia" and all taxa above it. In addition to that, he wants to restrict the results to oligonucleotides which have been successfully used in FISH experiments.*

The form for the target organism search is located under "Search" in the main menu and is the first form on the page.

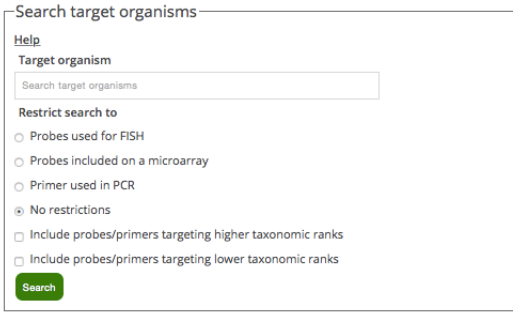


Figure 4.1: Target organism search form.

Initially, Bob has to enter the taxonomic name of the target organism in the input field. An auto-completion function supports him in his efforts and after entering the first few characters into the field, a drop-down list appears suggesting possible taxonomic names. He can choose the correct term from this list and continue with the selection of the restriction options.

Since Bob is only interested in probes which have been successfully applied in FISH experiments, he selects "Probes used for FISH" in the form. Furthermore, he also wants to include all probes, targeting taxa in higher taxonomic ranks as the one he is looking for. So he has to choose "Include probes/primers targeting higher taxonomic ranks".

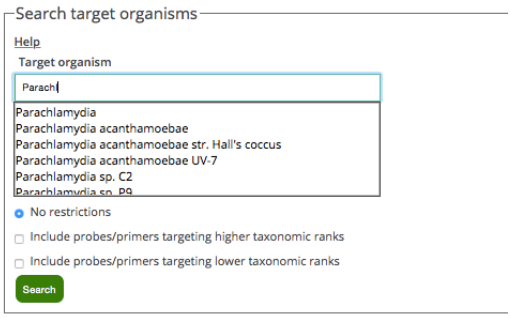


Figure 4.2: Auto-completion in the target organism search form.

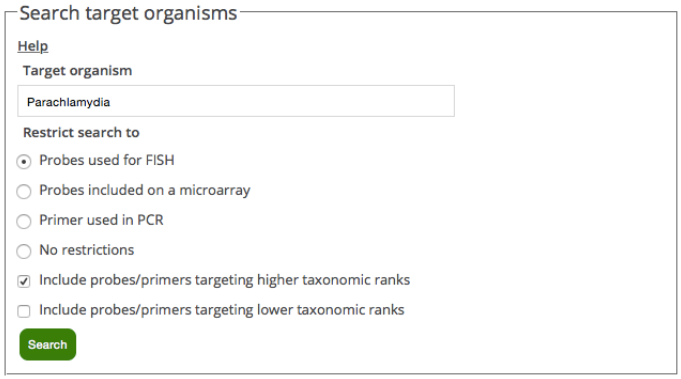
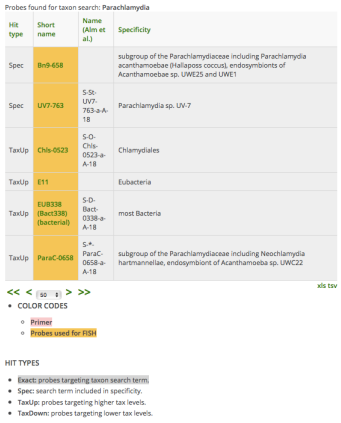


Figure 4.3: Complete user input of Bob's parachlamydia search query.

After clicking "Search", Bob receives a list of all probes matching his requirements. The results table, shown in figure 4.4, contains six probes with two different hit types, "Spec" and "TaxUp". "Spec" means, that the manually curated "specificity" field in the probe entry indicates this organism as a possible target. It suggests that these probes target certain strains or some organisms within the subgroup "Parachlamydiaceae". All probes with hit type "TaxUp" are oligonucleotides targeting taxa above the genus "parachlamydia". Since the search query is restricted to "Probes used for FISH", all probes except for those successfully used in FISH experiments are excluded. To highlight the FISH probes, a different colouring scheme is applied. All colour codes and hit types are described at the bottom of the results page. By clicking on the name of the oligonucleotide, researchers can also access a more detailed description of each probe. Information like the name, the sequence, the remarks or the


Probes found for taxon search: Parachlamydia

Hit type	Short name	Name (Alm et al.)	Specificity
Spec	Bn9-658		subgroup of the Parachlamydiaceae including Parachlamydia acanthamoebae (Hallaposs coccus), endosymbionts of Acanthamoebae sp. UWE25 and UWE1
Spec	UV7-763	S-St-UV7-763-a-A-18	Parachlamydia sp. UV-7
TaxUp	Chls-0523	S-O-Chls-0523-a-A-18	Chlamydiales
TaxUp	E11		Eubacteria
TaxUp	EUB338 (Bact338) (bacterial)	S-D-Bact-0338-a-A-18	most Bacteria
TaxUp	ParaC-0658	S-*-ParaC-0658-a-A-18	subgroup of the Parachlamydiaceae including Neochlamydia hartmannellae, endosymbiont of Acanthamoeba sp. UWC22

xls tsv

<< < 50 > >>

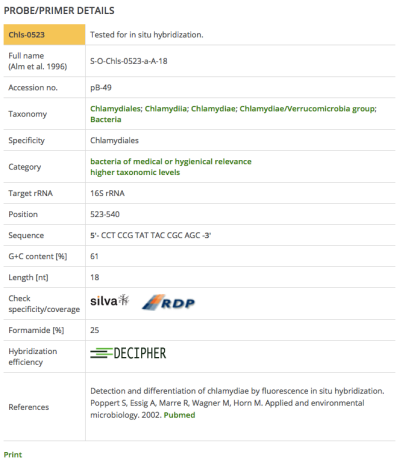
- COLOR CODES
  - Primer
  - Probes used for FISH

HIT TYPES

- Exact: probes targeting taxon search term.
- Spec: search term included in specificity.
- TaxUp: probes targeting higher tax levels.
- TaxDown: probes targeting lower tax levels.

Figure 4.4: Results table of the parachlamydia search query.

references are listed there and, if available, an external link to the original publication is also added to the report. In addition to that, all probes can be further analysed by external tools like RDP, SILVA or Decipher through a simple click on the banner of the corresponding tool.


PROBE/PRIMER DETAILS

Chls-0523	Tested for in situ hybridization.
Full name (Alm et al. 1996)	S-O-Chls-0523-a-A-18
Accession no.	pB-49
Taxonomy	Chlamydiales; Chlamydiia; Chlamydiae; Chlamydiae/Verrucomicrobia group; Bacteria
Specificity	Chlamydiales
Category	bacteria of medical or hygienical relevance higher taxonomic levels
Target rRNA	16S rRNA
Position	523-540
Sequence	5'- CCT CCG TAT TAC CGC AGC -3'
G+C content [%]	61
Length [nt]	18
Check specificity/coverage	silva RDP
Formamide [%]	25
Hybridization efficiency	DECIPHER
References	Detection and differentiation of chlamydiae by fluorescence in situ hybridization. Poppert S, Essig A, Marre R, Wagner M, Horn M. Applied and environmental microbiology. 2002. Pubmed

Print

Figure 4.5: Detailed probe report for probe "Chls-0523".

If a user decides to use such a service, a new browser tab opens and the corresponding tool is preconfigured with the required input data of the original probe. Via the hyperlink "Print" in the lower left corner of the probe report, Bob can also request a more printer friendly version of this report page.

4.1.2 Searching for probe names, keywords and authors

Below the target organism search form is the second search option. It gives researchers the opportunity to look for probes with certain names and combine this with keywords found in the abstract of the reference literature or the author's list of the corresponding oligonucleotide.

Use-case: *Alice wants to find all probes names containing the pattern "UNIV". Furthermore, she only wants probes by the author "Jakobsen".*

This search form contains two input fields. In the first one, Alice has to enter the name of the probe. The field below is for the name of the author or for a specific keyword found within the probe literature.

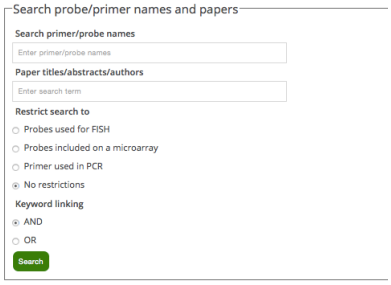


Figure 4.6: ProbeBase probe name and probe author search form.

Since she only looks for probes with the name pattern "UNIV" and which are from the author "Jakobsen", she can keep the default "keyword linking option" value "AND". This ensures that both search terms are mandatory in the results. After pressing the "Search" button, the database looks for all probes matching these criteria.

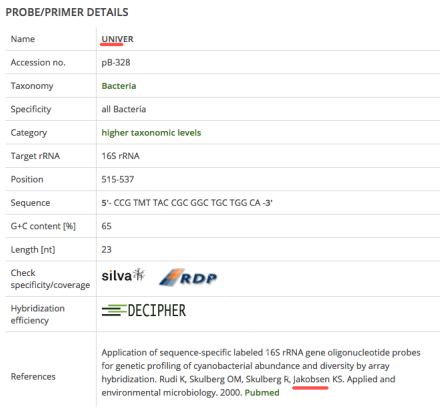


Figure 4.7: Detailed view of the found probe.

4.1.3 Sequence search

The third search option offers the opportunity to look for oligonucleotides containing a specific sequence pattern.

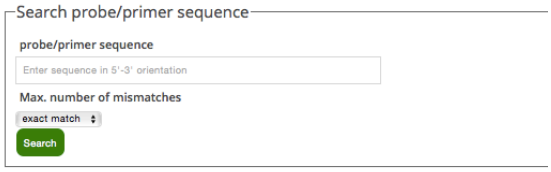


Figure 4.8: ProbeBase sequence search.

Use-case: *Alice queries the database for probes containing the following sequence pattern: GCTAGACCTT. No mismatches are allowed.*

The sequence search form is another sub-form in the "Search" area. Alice needs to enter the nucleobase-pattern into the input field. Since she wants to find the exact pattern, Alice can keep the settings for mismatches to "exact match". By clicking the "Search" button, the sequence search starts and when the search is completed, all matching probes are returned. In this use-case, no suitable probe has been found. To perform a less strict sequence search, Alice starts another run, by setting the mismatch count to two. Now one match is found in the database.

Probes/primers found matching: **GCTAGACCTT**

Short name	Mismatches	Sequence	Specificity
Bnix-64	1	GCTAGACCTGTTACCGCT	Candidatus Endonucleobacter bathymodioli

Figure 4.9: Probes containing the sequence pattern.

4.1.4 Search for specific target sites

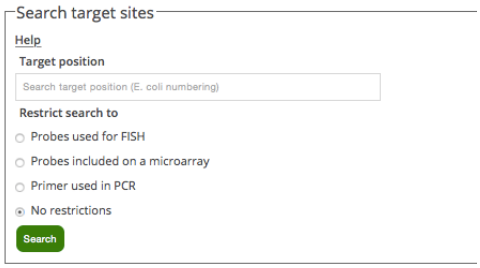


Figure 4.10: ProbeBase target site search form.

In the last search form, researchers can examine the oligonucleotide accessibility within the 16S RNA.

Use-case: *Bob needs to find all primers in the database targeting the 16S rRNA near position 60.*

Bob has to enter the target site position into the input field of the search form. Since he only wants to receive proper primers, he needs to restrict the results to "Primer used in PCR". After clicking "Search", the web-service checks the database for probes targeting the 16S RNA near the requested position.

In addition to the found probes, the results page also contains a table which provides an overview of the quality of the fluorescence signal at certain positions near the requested target site. The classification is done with classes, ranging from "I" to "VI". "I" being the best and "VI" the worst signal. This way, researchers can determine if the oligonucleotide is suitable for any FISH experiments. Furthermore, the results page also contains all the necessary reference information so Bob can gain a better understanding of target sites.

Probe name	Target	Position
Pra46F	16S rRNA	46-60
64F	16S rRNA	44-63
68F	16S rRNA	49-68
P63F	16S rRNA	42-62

PROBE BRIGHTNESS CLASSES NEAR POSITION 60 xls tsv

I: highest class (brightest fluorescence signal); VI: lowest class (almost no fluorescence signal, not suitable for FISH). Consensus: consensus brightness classes using data from E.coli, Pirellula sp., and Metallosphaera sedula.

E. coli 16S RNA	Consensus	E. coli 23S RNA
32-47 Class: III		
38-54 Class: II	20-37 Class: II	19-36 Class: III
48-65 Class: II	38-55 Class: II	37-54 Class: II
55-72 Class: III	**55-72 Class: II**	**55-72 Class: III**
60-77 Class: II	109-126 Class: II	73-90 Class: III
63-80 Class: II	127-144 Class: II	91-108 Class: IV
66-83 Class: II		

References:

- Fuchs B. M., Wallner G., Beisker W., Schwippl I., Ludwig W., Amann R. (1998). Flow cytometric analysis of the in situ accessibility of Escherichia coli 16S rRNA for fluorescently labeled oligonucleotide probes. Appl Environ Microbiol. 64: 4973-82. Abstract
- Fuchs B. M., Syutsubo K., Ludwig W., Amann R. (2001). In situ accessibility of Escherichia coli 23S rRNA to fluorescently labeled oligonucleotide probes. Appl Environ Microbiol. 67: 961-8. Abstract
- Behrens S., Ruhland C., Inacio J., Huber H., Fonseca A., Spencer-Martins I., Fuchs B.M., Amann R. (2003). In situ accessibility of small-subunit rRNA of members of the domains Bacteria, Archaea, and Eucarya to Cy3-labeled oligonucleotide probes. Appl Environ Microbiol. 69: 1748-58. Abstract

Figure 4.11: ProbeBase target site search form.

4.2 Probe matching

The probe matching tool, which can be found under "Match" in the main menu, is used to detect oligonucleotides matching any submitted user sequence. Users can either insert their sequences directly into the form on the page or upload them in a separate text file. It is mandatory that all sequence data is encoded in the FASTA-format. FASTA-files are text-based files. The first line starts with > and continues without any space with the name or the ID of the sequence. The next lines are reserved for the actual sequence. Additional sequences are after the last entry. Examples are provided in the probe matching form. If users submit their data in both ways, all sequences are merged into one file.

Use-case: *Bob needs to find suitable probes for a FASTA-file he received from a colleague. He accepts up to two mismatches between the submitted sequences and the probes in the database. Furthermore, he wants all found probes to be grouped in the results table per submitted sequence.*

A detailed description of the tool can be accessed by clicking the "Help" link on top of the page. This will reveal further explanation on how to use this tool, typical applications and its limits. Since Bob received the sequences in the form of a FASTA-file, he can upload it by clicking on the button below the large text field in the match form. In the file dialogue, he has to select the file that he wants to upload from his computer. If Bob accidentally uploads a non-FASTA conform file, the web resource dumps the file and returns an empty results table.

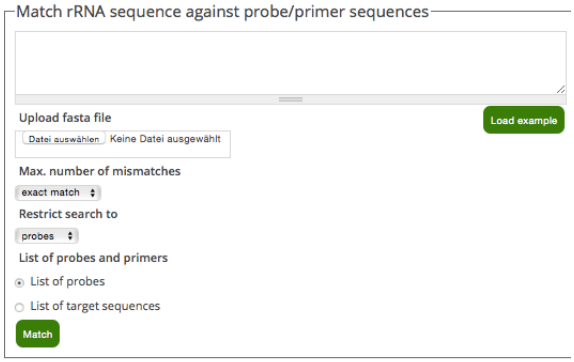
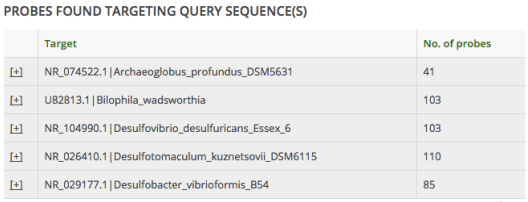


Figure 4.12: Probe matching form.

After that, Bob needs to select the number of allowed mismatches, which is in this use-case, two. Under "Restrict search to" he can keep the default value "probes". To group the results list per submitted sequence, Bob has to select "List of target sequences" in the last form section. To start the probe matching process, Bob needs to click on the "Match" button. Depending on how many sequences are in the uploaded file, it can take a while. When the matching process is finished, Bob gets redirected to the results page.

As shown in figure 4.13, all found probes are grouped per submitted sequences. The column "Target" holds all their names from the submitted file and "No. of


PROBES FOUND TARGETING QUERY SEQUENCE(S)

	Target	No. of probes
[+]	NR_074522.1 | Archaeoglobus_profundus_DSM5631	41
[+]	U82813.1 | Bilophila_wadsworthia	103
[+]	NR_104990.1 | Desulfovibrio_desulfuricans_Essex_6	103
[+]	NR_026410.1 | Desulfotomaculum_kuznetsovii_DSM6115	110
[+]	NR_029177.1 | Desulfobacter_vibrioformis_B54	85

Figure 4.13: Results of a probe match query.

probes" shows the total number of probes matching them. The table is fully sortable. This is especially useful if users are interested in the sequences with the most or least probe hits. By clicking on the "plus" symbol, the list gets expanded and all the probes matching the corresponding sequence appear. If Bob is also interested in the alignment between the probe and a particular sequence, he can further expand the list by clicking on "Show alignment". By clicking on the probe name, the user is redirected to the probe report page.

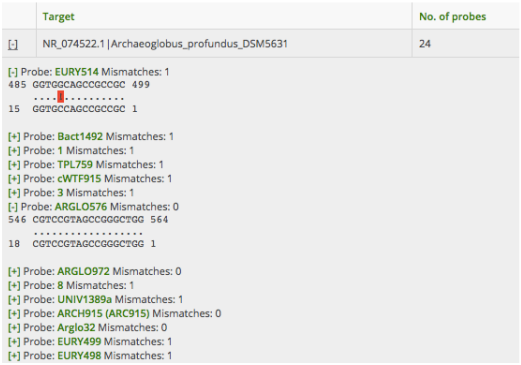


Figure 4.14: Extended probe match results.

4.3 Proxy matching

Use-case: *Alice has a file with MiSeq reads from 16S rRNA environmental samples. For these, she wants to find suitable primers in the probeBase. The sequences are too short for a regular probe match query, and so Alice decides to try the new proxy matching tool. She wants at least 90% matching between the submitted sequences and the proxy sequence candidate. In addition to that, Alice does not want any mismatches to occur between the proxy sequence and the oligonucleotide(s). The results should be grouped per submitted sequence.*

By clicking the "Help" hyperlink, Alice can reveal a description of the purpose and the usage of this tool. If a non-FASTA conform file is submitted to the web resource, it gets dumped and an empty results list is returned. Alice has to upload the sequence data first by clicking on the "File upload" button and selecting the local file. Since she wants to analyse 16S rRNA environmental data, the default setting for the SILVA database, "Silva SSU", is retained.

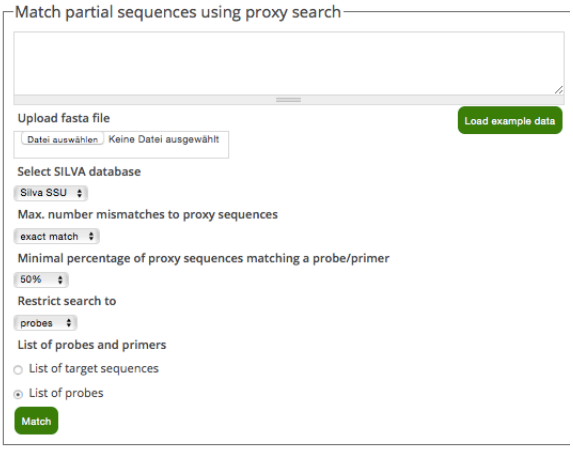
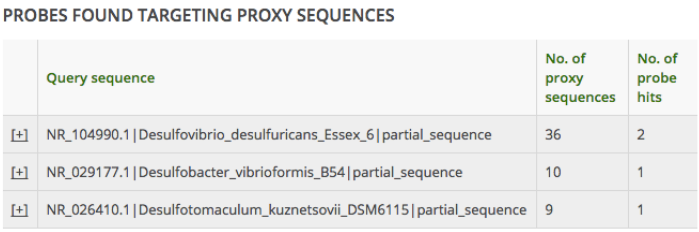


Figure 4.15: Proxy matching form.

Alice does not want any mismatches to occur between the proxy sequence and the primers. Therefore the settings for mismatches is set to "exact match". Furthermore, she wants at least 90% coverage between a possible proxy candidate and the submitted sequence. Aside from that, Alice is only interested in primers and so the setting

"Restrict search to" is changed to "primer". Like before, the results setting is set to "List of target sequences" to group the results list accordingly.

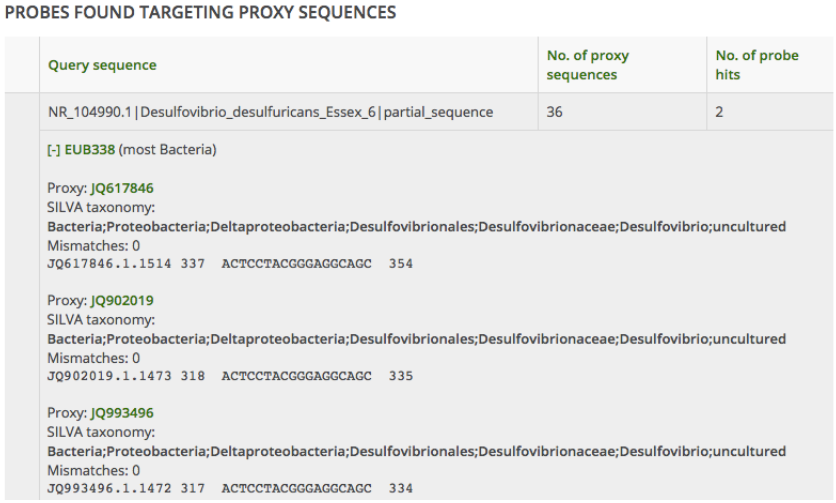


PROBES FOUND TARGETING PROXY SEQUENCES

	Query sequence	No. of proxy sequences	No. of probe hits
[+]	NR_104990.1 | Desulfovibrio_desulfuricans_Essex_6 | partial_sequence	36	2
[+]	NR_029177.1 | Desulfobacter_vibrioformis_B54 | partial_sequence	10	1
[+]	NR_026410.1 | Desulfotomaculum_kuznetsovii_DSM6115 | partial_sequence	9	1

Figure 4.16: Results of a proxy match.

The results table contains three columns: the submitted sequence, the number of the found proxies for these user sequences and the total number of probe hits for all the discovered proxies. By clicking on the "plus" symbol of each submitted sequence, a list with all possible oligonucleotides appear. Through a mouse click on the second plus symbol of each found probe, Alice can further examine all proxy candidates matching this oligonucleotide.



PROBES FOUND TARGETING PROXY SEQUENCES

Query sequence	No. of proxy sequences	No. of probe hits
NR_104990.1 | Desulfovibrio_desulfuricans_Essex_6 | partial_sequence	36	2

[-] EUB338 (most Bacteria)

Proxy: JQ617846
SILVA taxonomy:
Bacteria;Proteobacteria;Deltaproteobacteria;Desulfovibrionales;Desulfovibrionaceae;Desulfovibrio;uncultured
Mismatches: 0
JQ617846.1.1514 337 ACTCCTACGGGAGGCAGC 354

Proxy: JQ902019
SILVA taxonomy:
Bacteria;Proteobacteria;Deltaproteobacteria;Desulfovibrionales;Desulfovibrionaceae;Desulfovibrio;uncultured
Mismatches: 0
JQ902019.1.1473 318 ACTCCTACGGGAGGCAGC 335

Proxy: JQ993496
SILVA taxonomy:
Bacteria;Proteobacteria;Deltaproteobacteria;Desulfovibrionales;Desulfovibrionaceae;Desulfovibrio;uncultured
Mismatches: 0
JQ993496.1.1472 317 ACTCCTACGGGAGGCAGC 334

Figure 4.17: Extended results for proxy match.

If she wants a more detailed view of the proxy sequences, she needs to click on the name of the proxy. A new browser tab opens and the corresponding proxy report on the SILVA database project page appears.

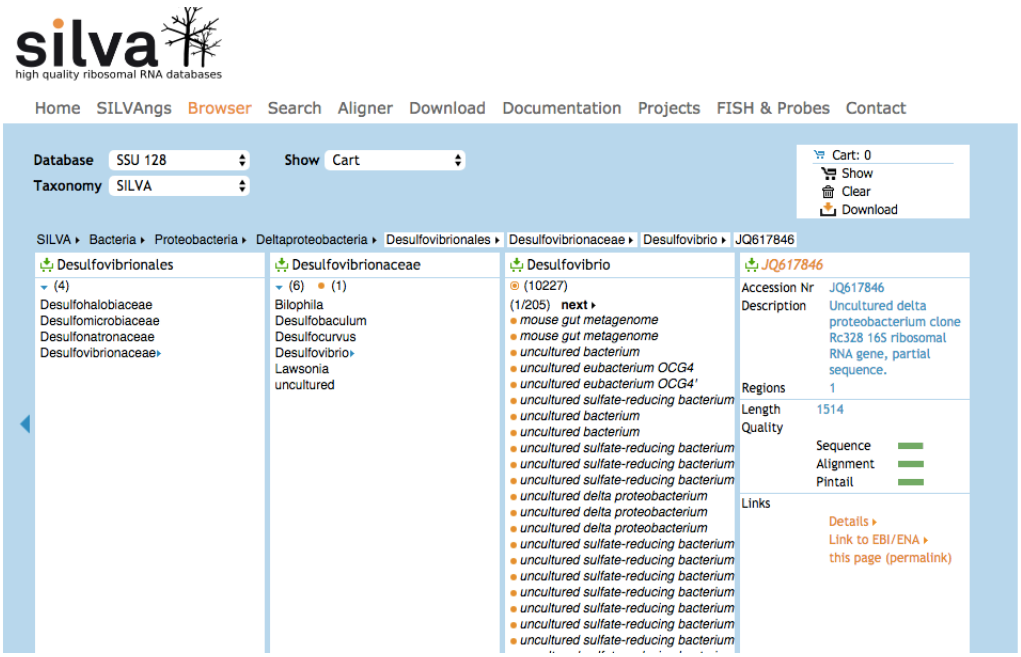


Figure 4.18: Proxy details in SILVA database project page.

Each proxy entry also contains the SILVA taxonomy and the number of mismatches between the proxy and the found probe. The last line of each entry contains the proxy ID and the matching sequence pattern, as well as the position in this proxy sequence.

4.4 Predefined lists

In the "lists section" of the web resource, a user can select from a wide range of options to get various predefined data tables. For example lists of probes tested for FISH, or lists of oligonucleotides sharing certain commonalities (e.g. sulfate-reducing microbes), etc.

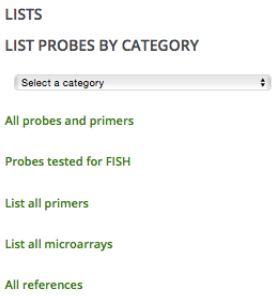


Figure 4.19: Predefined data lists section in probeBase.

Goal: *Bob wants to get a list of all probes in the probeBase targeting "phototrophic bacteria".*

By clicking on "Lists" in the main menu of the probeBase, Bob is redirected to the predefined lists area of the webpage. In the drop-down menu under "LIST PROBES BY CATEGORY", he can choose from a wide range of criteria probes must fulfil. In this use-case, Bob wants to get a list of all probes in the database which target "phototrophic bacteria". By clicking on this term, a list of all those oligonucleotides is displayed (Figure 4.20).

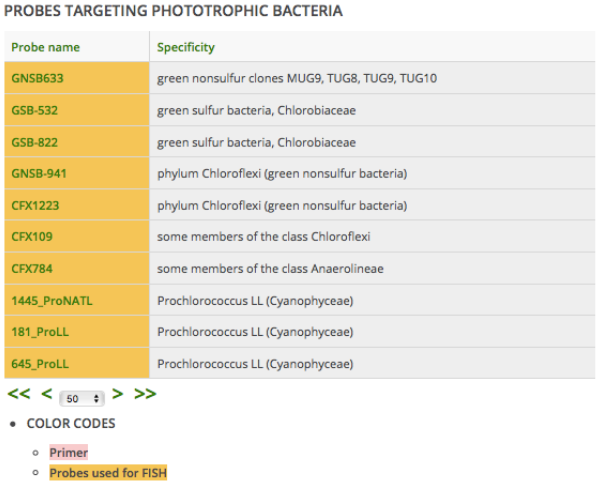


Figure 4.20: Results data list.

4.5 Submission

4.5.1 Single probe submission

Use-case: *User Bob has an oligonucleotide he wants to contribute to the project. He decides to upload it via the new probe submission form.*

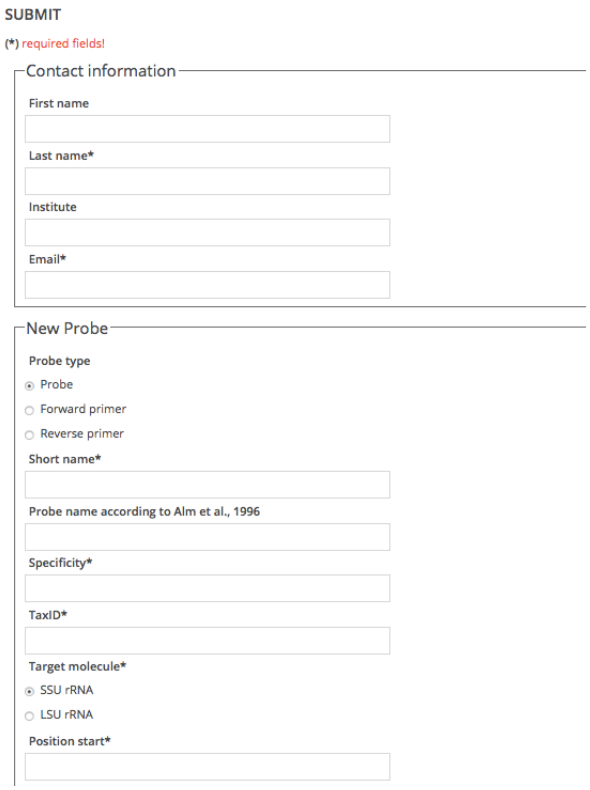


Figure 4.21: Single probe submission form.

All fields which are required for a successful submission are marked with an asterisk within the form. If Bob misses any of these required fields, the submission won't be accepted until he sorts out the problems. The section at the top contains all input fields for the contact information of the probe author. It continues with the input fields for the core information of the probe. These include the probe type, the specificity, the sequence or the starting position in the RNA. The submission page

also contains three hidden forms to specify the categories, possible microarrays and competitors of the oligonucleotide. By clicking on the names of each section, the user can reveal the corresponding subform.

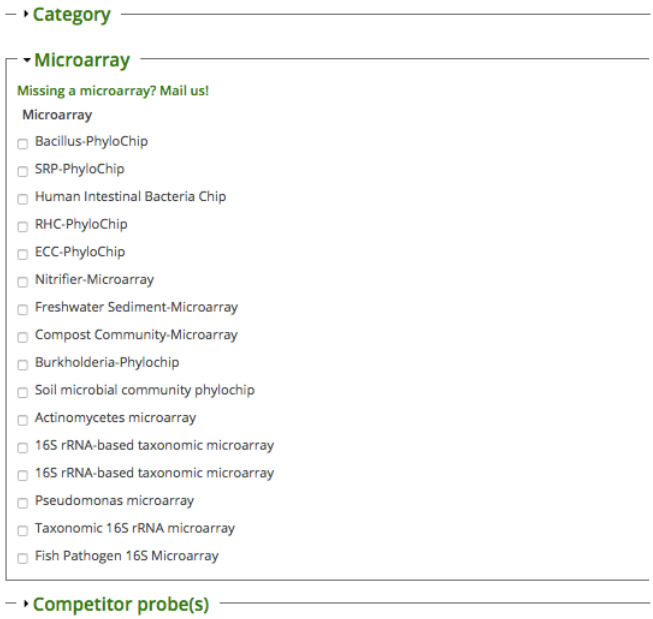


Figure 4.22: Extended categories subform.

The text field in the "Competitor probe(s)" subform is reserved for competitor sequences. One line contains one possible competitor sequence. The reference section of the submission form holds a custom input field, which can be used to provide information on the referring literature and an additional input field for PubMed IDs. If a paper has a PubMed ID, a submitter can enter this ID in the corresponding field. By pressing "Get PubMed entry", all required information, such as authors or title, are requested from the PubMed database and automatically filled out in the form. The probe submission is completed by clicking the "Submit" button at the end. The contribution is then sent to the database. Before the oligonucleotide can be accessed by other users, it needs to be activated by the probeBase team.

4.5.2 Batch submission

To upload several probes at once researchers can use the batch submission section.

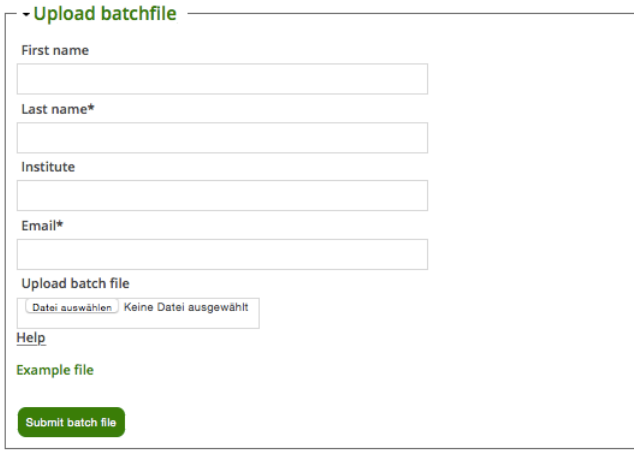


Figure 4.23: Batch submission form.

Use-case: *Alice and her colleagues have designed new probes which could be beneficial to the probeBase project. She decides to submit them via the batch submission feature.*

Alice first needs to examine the example file. By clicking the right mouse button on "Example file" and selecting the "save to" option from the context menu, she can download the example file. The file contains twelve columns. To use the batch-upload for her data, she has to sustain this file structure. After her data file is added to the form, she just needs to press the "Submit batch file" button.

```
SUP;SUP-6-16;p;Parachlamydia;1234;LSU;GTGCATTGGCT;5;our new probe;260832;GTGCATTTGC,GTAGCTAAGC;-
```

Code 4.1: Example of a probe entry in upload file.

Column	Description
Probe name	The name of the probe.
Name alm	The name alm of the probe.
Probe type	Probe type. 'r' for reverse primer, 'f' for forward primer and 'p' for probe.
Specificity	Specificity text field.
Taxonomy ID	Taxonomy ID of the targetorganism.
Target RNA	The target RNA. 16, 18 or 23.
Sequence	Probe sequence.
Formamide	Percentage formamide.
Remarks	Optional remarks.
PubMed ID	PubMed ID of the reference paper.
Competitor sequence(s)	Comma separated list of probe competitor sequences.
Custom reference	Custom reference information.

Table 4.1: Description of all CSV-fields.

CHAPTER 5

Conclusion and Outlook

The intention of this thesis is the reimplementation of the former probeBase. This goal was accomplished after approximately twelve months. Not only have the results been successfully published in the annual Database Issue of the journal "Nucleic Acids Research" but also the overall user impressions of the new web resource are positive and show that the updated service is well accepted. Many users notice the higher page speed, the improved user interface and the enhanced search tools. In spring 2017 the average number of users accessing the new probeBase service was approximately 800 unique requests per week.

Aside from favourable user impressions and good page access numbers, the thesis also improved the data quality of the original database through the introduction of NCBI taxonomy IDs. These identification numbers have been assigned to each oligonucleotide and precisely determine their target specificity. The former database held only a manually curated list of possible target organism names. Taxonomy IDs render these records obsolete and lower the chance for potential errors within the database since just a single unique number needs to be maintained. Most of the probes have been assigned automatically to their corresponding taxonomy ID via a custom python script. Due to ambiguities, about 401 probes have been mapped manually to a proper

taxonomy ID. With the help of the probeBase team, it was possible to sort out uncertainties and associate these oligonucleotides to an adequate taxonomy ID. Since the original probe specificity information is also available in the new database, it is still possible to find useful information or at least hints regarding the intended specificity of those oligonucleotides.

The old probeBase only offered an exact string matching tool. It was, therefore, necessary to add a new and dedicated sequence aligner to the database. This has been accomplished and offers users approximate string matching capabilities. Initially, it was planned to use only one tool, VMATCH, for all sequence related tasks, such as the search for oligonucleotides in the database or matching probes with submitted sequences. During development, various benchmarks revealed, that using VMATCH for huge sequences is not applicable because of the low performance and therefore an additional tool was needed. After testing multiple sequence aligner, LAST was chosen to fulfil this requirement and so it is used for the new proxy matching feature. This supports microbiologists in their research by suggesting proxy sequences for submitted reads which are too short for a "simple" probe match. Found proxies are used to query the database for suitable oligonucleotides as placeholders for the original ones. The first feedback concerning this feature is excellent. Some users already suggested enhancements like a more detailed alignment view. This shows that this tool may be one of those areas which will be further extended in the future.

During the development, I received a request from staff members of the SILVA database, asking me to the support one of their input fields, which allows their users to query the probeBase for probe sequences or probe names. It turned out that the modular software design of the probeBase was a good design choice since the implementation for this external input field was done without any difficulties.

The new technology stack behind probeBase ensures that this project can grow further over the coming years. Its maintenance is shared among the staff of the Division of Computational Systems Biology, which deals with all the technical aspects, like the software updates or the administration of the web server. Members of the Department of Microbiology and Ecosystem Science (DOME) are responsible for answering user questions and maintaining the data within the database.

One technical challenge this service will face in the next years will be the transition

from Drupal 7 to Drupal 8. Version 8 of this Content Management System is entirely redesigned from scratch and uses an object-oriented programming approach, introducing classes and objects, while Drupal 7 follows a procedural programming paradigm. Therefore most of the code needs to be refactored and ported to this style of programming. Since the complexity of the developed Drupal modules is somewhat manageable, it should not be too hard to transfer the codebase to the new version of the CMS. It may also be necessary to abandon some of the used third-party modules, which have been created by external developers if they lack Drupal 8 support [45].

Appendix A: Paper

D586–D589 *Nucleic Acids Research, 2016, Vol. 44, Database issue* *Published online 19 November 2015*
doi: 10.1093/nar/gkv1232

probeBase—an online resource for rRNA-targeted oligonucleotide probes and primers: new features 2016

Daniel Greuter[1], Alexander Loy[2,*], Matthias Horn[2,*] and Thomas Rattei[1]

[1]Division of Computational Systems Biology, Department of Microbiology and Ecosystem Science, Research Network Chemistry meets Microbiology, University of Vienna, A-1090 Wien, Austria and [2]Division of Microbial Ecology, Department of Microbiology and Ecosystem Science, Research Network Chemistry meets Microbiology, University of Vienna, A-1090 Wien, Austria

Received October 2, 2015; Revised October 27, 2015; Accepted October 30, 2015

ABSTRACT

probeBase http://www.probebase.net is a manually maintained and curated database of rRNA-targeted oligonucleotide probes and primers. Contextual information and multiple options for evaluating *in silico* hybridization performance against the most recent rRNA sequence databases are provided for each oligonucleotide entry, which makes probeBase an important and frequently used resource for microbiology research and diagnostics. Here we present a major update of probeBase, which was last featured in the NAR Database Issue 2007. This update describes a complete remodeling of the database architecture and environment to accommodate computationally efficient access. Improved search functions, sequence match tools and data output now extend the opportunities for finding suitable hierarchical probe sets that target an organism or taxon at different taxonomic levels. To facilitate the identification of complementary probe sets for organisms represented by short rRNA sequence reads generated by amplicon sequencing or metagenomic analysis with next generation sequencing technologies such as Illumina and IonTorrent, we introduce a novel tool that recovers surrogate near full-length rRNA sequences for short query sequences and finds matching oligonucleotides in probeBase.

INTRODUCTION

Our understanding of the diversity and role of microorganisms on our planet is to a great extent based on exploiting the ribosomal RNA as phylogenetic marker molecule in diagnostic molecular biology and microscopy assays. While rRNA-targeted oligonucleotides have been applied in different sorts of diagnostic formats such as DNA microarrays (PhyloChips) (1,2) and denaturing gradient gel electrophoresis (3), they are now most widely used for amplicon sequencing and fluorescence *in situ* hybridization (FISH) (4). Today, highly multiplexed amplicon sequencing with rRNA-targeted primers enables surveying microbial diversity across numerous samples (5,6), which provides unprecedented insights into the spatial distribution and temporal dynamics of the diverse microbial communities that thrive in the environment (7) or are associated with eukaryotic hosts (8,9). Furthermore, FISH with rRNA-targeted probes and quantitative microscopy is a standard tool for revealing the identity, abundance and spatial localization of microbial cells in complex samples. More than two decades of development of FISH probes and techniques for microbial diagnostics have established a variety of methods, such as DOPE-FISH (10), CARD-FISH (11), CLASI-FISH (12) and HCR-FISH (13), and a wealth of tested probes that target diverse phylogenetic and/or taxonomic groups of microorganisms. probeBase was originally established in 2002 (14) to provide a common, freely accessible repository for rRNA-targeted oligonucleotide sequences, including contextual information and multiple options for testing *in silico* specificity and coverage (15) against up-to-date rRNA sequence databases such as RDP-II (16) and SILVA (17). To date (September 2015), probeBase contains 2788 probes, 175 domain-specific PCR primers (18) and 16 microarrays from 499 publications and is an online resource that is frequently used by the scientific community (180 000 average page views per year).

Finding appropriate oligonucleotides with a suite of 'Search' and sequence 'Match' tools provides convenient access to the information in the database. Probes, primers, microarray layouts or references can further be retrieved through the 'Lists' service, including dynamic lists of all probes, all primers, all references or oligonucleotides that

*To whom correspondence should be addressed. Tel: +43 1 4277 76605; Fax: +43 1 4277 876605; Email: loy@microbial-ecology.net
Correspondence may also be addressed to Matthias Horn. Tel: +43 1 4277 76608; Fax: +43 1 4277 876608; Email: horn@microbial-ecology.net

target microorganisms from specific environments (e.g. intestinal microbiota) or with specific functions (e.g. sulfate-reducing microorganisms).

This update describes recent improvements and new features added since the last update in 2007 (19), including (i) extended 'Search' and 'Match' options, (ii) a new 'Proxy' tool that finds probe sets for short query sequences based on corresponding near full-length rRNA sequences and (iii) suggestion of taxonomically informed hierarchical probe sets for applications using multiple probes such as multi-color FISH and DNA microarrays.

NEW DATABASE BACKEND, SEARCH ENVIRONMENT AND WEBSITE FOR probeBase

The probeBase database has been moved to a new, more scalable database backend. This dramatically reduces the retrieval times when the database is queried and it can also handle a much larger number of requests simultaneously. A new database scheme was developed, which links probeBase entries with the NCBI taxonomy database (20). Database procedures were implemented to retrieve the taxonomic lineages (up and down) of each specificity term instantly from the NCBI taxonomy database. Thereby all changes in the reference taxonomy will be automatically adopted by probeBase.

The continuously growing number of sequences in probeBase and in rRNA sequence databases (16,17) made it necessary to refine the search environment of probeBase. The core of the new search and match tools are sequence indexes based on enhanced suffix array data structures. These suffix arrays allow very short retrieval times by rapid exact string matching. They are used by VMATCH (21) for probe/primer search ('Search') and sequence match ('Match'), and by LAST (22) for the proxy sequence match ('Proxy') functionality (see below for a description of the 'Search', 'Match' and 'Proxy' tools). The exact string matching in VMATCH is not aware of DNA ambiguity characters, such as R, Y, W, etc. probeBase therefore refines the alignments calculated by VMATCH for the sequence match ('Match') function and considers such ambiguity positions to determine the correct number of mismatches also for these positions.

The probeBase web page has been moved to a new content management system to facilitate maintenance and more rapid adaptations of the web page. Another advantage of the content management system is the responsible layout, which considers the size of the browsing device. Hence, the page will be optimized for smaller displays if users access probeBase via their mobile phone or tablet computer.

Result lists are now fully sortable by just clicking on the header of the respective column. In addition, longer tables are being split into multi-page tables to give the user a convenient overview even if a certain database query returns many results. This feature is particularly important due to the increased number of supported sequences per user query. The multi-page table views are accompanied by an export function. Users are able to export results in .xls and .tsv format, which allows performing further analysis in any other suitable software, such as Excel or OpenOffice.

A TARGET TAXONOMY FOR EACH OLIGONUCLEOTIDE

Detailed information is provided for each oligonucleotide (14,19), including its specificity, which indicates the intended target organism(s) of the respective probe/primer as described in the original publication or during user submissions to probeBase. Based on the information in the specificity field, we have automatically mapped each oligonucleotide to the NCBI taxonomy (20). Where necessary, assigned taxonomic names (i.e. NCBI taxonomy IDs) were manually corrected and curated to contain one taxonomic assignment per oligonucleotide. For probes targeting multiple taxa (e.g. two different species), we chose either the taxon that is predominantly covered by the probe or the next higher taxonomic rank that included all taxa (e.g. the genus, family, order, phylum or domain). Probe entries for which the specificity field did not contain any (e.g. 'clone XYZ') or only limited taxonomic information (e.g. 'deltaproteobacterial symbiont of…') were assigned to the root in the NCBI taxonomy or to the lowest meaningful taxonomic rank. The new taxonomy field in the probe details view shows the entire taxonomic hierarchy—from the assigned taxon to its highest taxonomic rank. It is noteworthy that this taxonomic assignment does not necessarily mean that an oligonucleotide is highly specific for a given taxon. Instead, it represents a systematic classification for all oligonucleotides in probeBase and allows for more advanced searches.

SEARCH probeBase

The 'Search' tool comprises multiple options for finding appropriate oligonucleotides and further information (Figure 1). Oligonucleotides can be recovered by the name of the target organism or taxon, by their specific target sites on the rRNA molecule, by the reference that originally described the oligonucleotide or simply by the name or sequence of the oligonucleotide itself. The output list of oligonucleotides that matched the search criteria can be restricted to primers, probes used successfully for FISH, and probes used on microarrays.

Because each probe is now assigned to a taxon in the NCBI taxonomy (20), a search for a specific target organism or taxon systematically returns all available oligonucleotides. The search target organism option is facilitated by an auto-complete function for taxonomic names to minimize the probability of typing errors. The search target organism option can be further adjusted by including oligonucleotides that target higher and/or lower taxonomic ranks than the query taxon. The corresponding output list of probes that target the query organism at different taxonomic levels helps researchers in identifying sets of hierarchically nested probes for application in multiple-probe hybridization formats such as multi-color FISH (10,12) and DNA microarrays (23).

The search for a probe/primer sequence not only yields perfectly complementary hits, but also oligonucleotides in probeBase with up to two mismatches to the query oligonucleotide. This allows, for example, to identify if a newly designed rRNA-targeted oligonucleotide and/or closely re-

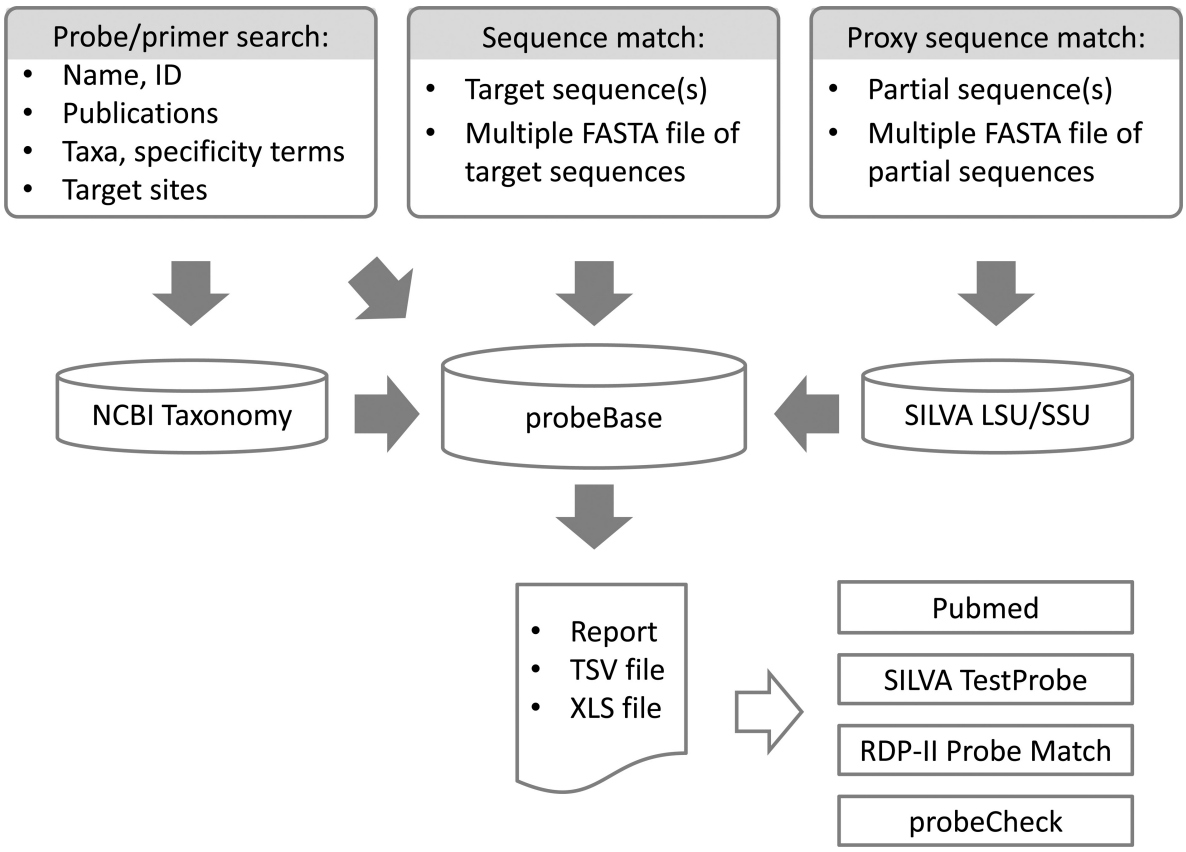


Figure 1. Structure and main tools of the probeBase database.

lated variants of it have already been developed, tested for FISH and published before.

MATCH rRNA QUERY SEQUENCES AGAINST OLIGONUCLEOTIDES IN probeBase

The original sequence match tool was developed to find all oligonucleotides in probeBase that perfectly match to up to 150 query rRNA sequences (14). However, due to the increased database size this tool was not usable and thus disabled for the past years. We have reimplemented and redesigned the sequence match tool (Figure 1), which is now based on a similarity search and able to process up to 1000 query sequences. Users can paste a (multi) fasta file into the text field, upload a (multi) fasta file or combine both options to query probeBase for complementary probes or primers with up to two mismatches. Results are either grouped by oligonucleotide, showing all matching sequences from the query per oligonucleotide, or by query sequence, showing all matching oligonucleotides per query sequence. A typical application of this tool is to quickly retrieve a set of available FISH probes that target rRNA sequences determined in an environmental microbial diversity survey without the need for extensive comparative sequence analysis. This probe set can then be readily applied for FISH to determine the abundance and spatial organization of the target organism in the sample (24).

FIND FULL-LENGTH PROXY SEQUENCES FOR SHORT rRNA READS AND MATCH AGAINST OLIGONUCLEOTIDES IN probeBase

Traditional surveys of microbial diversity by PCR amplification of near full-length rRNA gene sequences from environmental DNA, cloning and Sanger sequencing of clone inserts have been almost completely replaced by highly parallel next generation sequencing of rRNA gene amplicons because of higher sample throughput and sequencing depth (i.e. number of sequences per sample). However, common technologies for multiplexed amplicon sequencing, such as Illumina MiSeq, produce only short (paired-end) reads that are typically less than 500 nucleotides in length. The short rRNA reads limit the selection of complementary oligonucleotides (e.g. by the 'Match' tool) for follow-up hybridization applications (25). Here, we provide a new 'Proxy' tool that finds corresponding near full-length rRNA sequences for short query sequences (Figure 1). These long proxy sequences are retrieved from the small or large subunit rRNA reference database of SILVA (17) and matched against the oligonucleotides in probeBase analogous to the 'Match' tool. The output includes the identified proxy sequences for each short query sequence and shows the probes or primers that have up to two mismatches to the proxy sequences.

SUBMISSION OF MISSING OR NEWLY DEVELOPED OLIGONUCLEOTIDES

New or missing oligonucleotides can be submitted using an online form. The reference details (e.g. journal, authors, title, abstract, year) will now be automatically filled in by entering the PubMed-ID (PMID) (26) of a publication that contains new or missing probes/primers.

AVAILABILITY

probeBase is maintained by the Department of Microbiology and Ecosystem Science, University of Vienna, Wien, Austria and available at http://www.probebase.net. We welcome comments concerning probeBase and highly appreciate reports of bugs, errors or missing probes. You may contact us by email to probebase@microbial-ecology.net.

ACKNOWLEDGEMENT

We thank Julia Ramesmayer and Albert Müller for database maintenance and Florian Goldenberg for technical support.

FUNDING

Austrian Science Fund [FWF, P25111-B22, I2320-B22 to A.L., I1628-B22 to M.H.]; Vienna Science and Technology Fund [WWTF, LS12-001 to A.L.]; European Research Council ERC [StG EVOCHLAMY, 281633 to M.H.]. Funding for open access charge: Austrian Science Fund (FWF).
Conflict of interest statement. None declared.

REFERENCES

1. Loy,A., Lehner,A., Lee,N., Adamczyk,J., Meier,H., Ernst,J., Schleifer,K.-H. and Wagner,M. (2002) Oligonucleotide microarray for 16S rRNA gene-based detection of all recognized lineages of sulfate-reducing prokaryotes in the environment. *Appl. Environ. Microbiol.*, **68**, 5064–5081.
2. DeAngelis,K.M., Wu,C.H., Beller,H.R., Brodie,E.L., Chakraborty,R., DeSantis,T.Z., Fortney,J.L., Hazen,T.C., Osman,S.R., Singer,M.E. *et al.* (2011) PCR amplification-independent methods for detection of microbial communities by the high-density microarray PhyloChip. *Appl. Environ. Microbiol.*, **77**, 6313–6322.
3. Muyzer,G., de Waal,E.C. and Uitterlinden,A.G. (1993) Profiling of complex microbial populations by denaturing gradient gel electrophoresis analysis of polymerase chain reaction-amplified genes coding for 16S rRNA. *Appl. Environ. Microbiol.*, **59**, 695–700.
4. Amann,R. and Fuchs,B.M. (2008) Single-cell identification in microbial communities by improved fluorescence in situ hybridization techniques. *Nat. Rev. Microbiol.*, **6**, 339–348.
5. Herbold,C.W., Pelikan,C., Kuzyk,O., Hausmann,B., Angel,R., Berry,D. and Loy,A. (2015) A flexible and economical barcoding approach for highly multiplexed amplicon sequencing of diverse target genes. *Front. Microbiol.*, **6**, 731.
6. Wu,L., Wen,C., Qin,Y., Yin,H., Tu,Q., Nostrand,J.D., Yuan,T., Yuan,M., Deng,Y. and Zhou,J. (2015) Phasing amplicon sequencing on Illumina Miseq for robust environmental microbial community analysis. *BMC Microbiol.*, **15**, 125.
7. Nemergut,D.R., Costello,E.K., Hamady,M., Lozupone,C., Jiang,L., Schmidt,S.K., Fierer,N., Townsend,A.R., Cleveland,C.C., Stanish,L. *et al.* (2011) Global patterns in the biogeography of bacterial taxa. *Environ. Microbiol.*, **13**, 135–144.
8. Dethlefsen,L. and Relman,D.A. (2011) Incomplete recovery and individualized responses of the human distal gut microbiota to repeated antibiotic perturbation. *Proc. Natl. Acad. Sci. USA*, **108**(Suppl. 1), 4554–4561.
9. Seedorf,H., Griffin,N.W., Ridaura,V.K., Reyes,A., Cheng,J., Rey,F.E., Smith,M.I., Simon,G.M., Scheffrahn,R.H., Woebken,D. *et al.* (2014) Bacteria from diverse habitats colonize and compete in the mouse gut. *Cell*, **159**, 253–266.
10. Behnam,F., Vilcinskas,A., Wagner,M. and Stoecker,K. (2012) A straightforward DOPE (double labeling of oligonucleotide probes)-FISH (fluorescence in situ hybridization) method for simultaneous multicolor detection of six microbial populations. *Appl. Environ. Microbiol.*, **78**, 5138–5142.
11. Pernthaler,A., Pernthaler,J. and Amann,R. (2002) Fluorescence in situ hybridization and catalyzed reporter deposition for the identification of marine bacteria. *Appl. Environ. Microbiol.*, **68**, 3094–3101.
12. Valm,A.M., Welch,J.L.M., Rieken,C.W., Hasegawa,Y., Sogin,M.L., Oldenbourg,R., Dewhirst,F.E. and Borisy,G.G. (2011) Systems-level analysis of microbial community organization through combinatorial labeling and spectral imaging. *Proc. Natl. Acad. Sci. USA*, **108**, 4152–4157.
13. Nikolakakis,K., Lehnert,E., McFall-Ngai,M.J. and Ruby,E.G. (2015) Use of hybridization chain reaction-fluorescent in situ hybridization to track gene expression by both partners during initiation of symbiosis. *Appl. Environ. Microbiol.*, **81**, 4728–4735.
14. Loy,A., Horn,M. and Wagner,M. (2003) probeBase: an online resource for rRNA-targeted oligonucleotide probes. *Nucleic Acids Res.*, **31**, 514–516.
15. Loy,A., Arnold,R., Tischler,P., Rattei,T., Wagner,M. and Horn,M. (2008) probeCheck - a central resource for evaluating oligonucleotide probe coverage and specificity. *Environ. Microbiol.*, **10**, 2894–2896.
16. Cole,J.R., Wang,Q., Fish,J.A., Chai,B., McGarrell,D.M., Sun,Y., Brown,C.T., Porras-Alfaro,A., Kuske,C.R. and Tiedje,J.M. (2014) Ribosomal Database Project: data and tools for high throughput rRNA analysis. *Nucleic Acids Res.*, **42**, D633–D642.
17. Quast,C., Pruesse,E., Yilmaz,P., Gerken,J., Schweer,T., Yarza,P., Peplies,J. and Glockner,F.O. (2013) The SILVA ribosomal RNA gene database project: improved data processing and web-based tools. *Nucleic Acids Res.*, **41**, D590–D596.
18. Klindworth,A., Pruesse,E., Schweer,T., Peplies,J., Quast,C., Horn,M. and Glockner,F.O. (2013) Evaluation of general 16S ribosomal RNA gene PCR primers for classical and next-generation sequencing-based diversity studies. *Nucleic Acids Res.*, **41**, e1.
19. Loy,A., Maixner,F., Wagner,M. and Horn,M. (2007) probeBase–an online resource for rRNA-targeted oligonucleotide probes: new features 2007. *Nucleic Acids Res.*, **35**, D800–D804.
20. Federhen,S. (2015) Type material in the NCBI Taxonomy Database. *Nucleic Acids Res.*, **43**, D1086–D1098.
21. Abouelhoda,M.I., Kurtz,S. and Ohlebusch,E. (2004) Replacing suffix trees with enhanced suffix arrays. *J. Discrete Algorithms*, **2**, 53–86.
22. Kielbasa,S.M., Wan,R., Sato,K., Horton,P. and Frith,M.C. (2011) Adaptive seeds tame genomic sequence comparison. *Genome Res.*, **21**, 487–493.
23. Schönmann,S., Loy,A., Wimmersberger,C., Sobek,J., Aquino,C., Vandamme,P., Frey,B., Rehrauer,H. and Eberl,L. (2009) 16S rRNA gene-based phylogenetic microarray for simultaneous identification of members of the genus Burkholderia. *Environ. Microbiol.*, **11**, 779–800.
24. Daims,H., Lücker,S. and Wagner,M. (2006) DAIME, a novel image analysis program for microbial ecology and biofilm research. *Environ. Microbiol.*, **8**, 200–213.
25. Berry,D., Schwab,C., Milinovich,G., Reichert,J., Ben Mahfoudh,K., Decker,T., Engel,M., Hai,B., Hainzl,E., Heider,S. *et al.* (2012) Phylotype-level 16S rRNA analysis reveals new bacterial indicators of health state in acute murine colitis. *ISME J.*, **6**, 2091–2106.
26. Agarwala,R., Barrett,T., Beck,J., Benson,D.A., Bollin,C., Bolton,E., Bourexis,D., Brister,J., Bryant,S.H., Canese,K. *et al.* (2015) Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res.*, **43**, D6–D17.

Bibliography

[1] Gurdeep Rastogi and Rajesh K. Sani. Molecular techniques to assess microbial community structure, function, and dynamics in the environment. *Microbes and Microbial Technology*, page 29–57, 2011.

[2] Jane B. Reece and Neil A. Campbell. *Campbell biology*. Pearson, 2011.

[3] Francis Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, Aug 1970.

[4] D. Baltimore. Viral rna-dependent dna polymerase: Rna-dependent dna polymerase in virions of rna tumour viruses. *Nature*, 1970.

[5] Scott O. Rogers. *Integrated molecular evolution*. CRC Press, 2012.

[6] Theodor Svedberg. *Sedimentation constants, molecular weights, and isoelectric points of the respiratory proteins*. Biological Laboratory, 1933.

[7] Ralph Stadhouders, Suzan D. Pas, Jeer Anber, Jolanda Voermans, Ted H.m. Mes, and Martin Schutten. The effect of primer-template mismatches on the detection and quantification of nucleic acids using the 5' nuclease assay. *The Journal of Molecular Diagnostics*, 12(1):109–117, 2010.

[8] Zhang T Piao X, Sun L. Effects of mismatches and insertions on discrimination accuracy of nucleic acid probes. *Acta Biochimica Polonica*, page 289, 2008.

[9] John M. Walker and Ralph Rapley. *Medical biomethods handbook*. Humana Press, 2005.

[10] Corinne Whitby and Torben Lund. Skovhus. *Applied Microbiology and Molecular Biology in Oilfield Systems*. Springer, 2011.

[11] Elisa Garimberti and Sabrina Tosi. Fluorescence in situ hybridization (fish), basic principles and methodology. *Methods in Molecular Biology Fluorescence in situ Hybridization (FISH)*, page 3–20, 2010.

[12] Arthur M. Lesk. *Database annotation in molecular biology: principles and practice*. Wiley, 2005.

[13] Gary J. O Larsen N. The ribosomal database project. *Nucleic Acids Research*, 1993.

[14] James R. Cole and James M. Tiedje. History and impact of rdp. *RNA Biology*, 2014.

[15] J. R Cole. The ribosomal database project (rdp-ii): sequences and tools for high-throughput rrna analysis. *Nucleic Acids Research*, 2004.

[16] Silva background information. https://www.arb-silva.de/documentation/. Accessed: 2017-03-01.

[17] gg. Silva ribosomal rna database. `https://en.wikipedia.org/wiki/SILVA_ribosomal_RNA_database`. Accessed: 2017-03-01.

[18] C. Quast, E. Pruesse, P. Yilmaz, J. Gerken, T. Schweer, P. Yarza, J. Peplies, and F. O. Glöckner. The silva ribosomal rna gene database project: improved data processing and web-based tools. *Nucleic Acids Research*, 2012.

[19] ff. Silva. high quality ribosomal rna databases. https://www.arb-silva.de. Accessed: 2017-03-01.

[20] A. Loy, F. Maixner, M. Wagner, and M. Horn. probebase–an online resource for rrna-targeted oligonucleotide probes: new features 2007. *Nucleic Acids Research*, 35(Database), Mar 2007.

[21] Maxime Crochemore, Christophe Hancart, and Thierry Lecroq. *Algorithms on strings*. Cambridge University Press, 2014.

[22] Dan Gusfield. *Algorithms on strings, trees, and sequences: computer science and computational biology*. Cambridge Univ. Press, 2009.

[23] Robert S. Boyer and J. Strother Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, Jan 1977.

[24] Why gnu grep is fast. https://lists.freebsd.org/pipermail/freebsd-current/2010-August/019310.html. Accessed: 2017-03-01.

[25] Graham A. Stephen. *String searching algorithms*. World Scientific, 2000.

[26] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[27] Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86, 2004.

[28] Enhanced suffix arrays. from http://www.mi.fu-berlin.de/wiki/pub/ABI/RnaSeqP4/enhanced-suffix-array.pdf. Accessed: 2017-03-01.

[29] Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. The enhanced suffix array and its applications to genome analysis. *Lecture Notes in Computer Science Algorithms in Bioinformatics*, page 449–463, 2002.

[30] T.f. Smith and M.s. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[31] W. J. Kent. Blat—the blast-like alignment tool. *Genome Research*, 12(4):656–664, 2002.

[32] Blast and blat. https://ab.inf.uni-tuebingen.de/teaching/ws09/bioinformatics-i/04-blast-blat-fasta.pdf. Accessed: 2017-03-01.

[33] Wikiomics:blast. http://www.openwetware.org/wiki/Wikiomics:BLAST. Accessed: 2017-03-01.

[34] Medha Bhagwat, Lynn Young, and Rex R. Robison. Using blat to find sequence similarity in closely related genomes. *Current Protocols in Bioinformatics*, 2012.

[35] Seqmapabout. https://seqmap.compbio.iupui.edu/tutorial/SeqMapAbout.html. Accessed: 2017-03-01.

[36] Stefan Kurtz. *The Vmatch large scale sequence analysis software. A manual.* University of Hamburg, 2016.

[37] S. Ossowski, K. Schneeberger, R. M. Clark, C. Lanz, N. Warthmann, and D. Weigel. Sequencing of natural strains of arabidopsis thaliana with short reads. *Genome Research*, 18(12):2024–2033, Mar 2008.

[38] Mark A. Batzer and Prescott L. Deininger. Alu repeats and human genomic diversity. *Nature Reviews Genetics*, 3(5):370–379, Jan 2002.

[39] Szymon M. Kiełbasa, Raymond Wan, Kengo Sato, Paul Horton, and Martin C. Frith. Adaptive seeds tame genomic sequence comparison. *Genome Research*, 21(3):487–493, 3 2011.

[40] Last. http://last.cbrc.jp/. Accessed: 2017-03-01.

[41] B. M. Fuchs, K. Syutsubo, W. Ludwig, and R. Amann. In situ accessibility of escherichia coli 23s rrna to fluorescently labeled oligonucleotide probes. *Applied and Environmental Microbiology*, 67(2):961–968, Jan 2001.

[42] S. Behrens, C. Ruhland, J. Inacio, H. Huber, A. Fonseca, I. Spencer-Martins, B. M. Fuchs, and R. Amann. In situ accessibility of small-subunit rrna of members of the domains bacteria, archaea, and eucarya to cy3-labeled oligonucleotide probes. *Applied and Environmental Microbiology*, 69(3):1748–1758, Jan 2003.

[43] Module developer's guide. https://www.drupal.org/developing/modules. Accessed: 2017-07-01.

[44] Compare db_query() and db_select() performance. https://www.drupal.org/node/1067802. Accessed: 2017-06-01.

[45] Daniel Greuter, Alexander Horn, Matthias Horn, and Thomas Rattei. probebase—an online resource for rrna-targeted oligonucleotide probes and primers: new features 2016. *Nucleic Acids Research - Database Issue*, page D586–D589, Nov 2015.