DEVELOPMENT OF A PROGRAMMING LIBRARY FOR GENERAL BIOINFORMATICS

vom Fachbereich Biologie
der Technischen Universität Darmstadt

zur Erlangung des Grades
Doctor rerum naturalium (Dr. rer. nat.)

genehmigte Dissertation von

PATRICK KUNZMANN

Erstgutachter:	Prof. Dr. Kay Hamacher
Zweitgutachterin:	Prof. Dr. Beatrix Süß

DARMSTADT 2023

This document was typeset based on a template by Daniel Bauer published under the Apache 2.0 License. The document was compiled using XeTEX and Biblatex. Used fonts were Vollkorn (serif), Open Sans (sans serif) and PT Mono (monospace).

Patrick Kunzmann: Development of a programming library for general bioinformatics
Darmstadt, Technische Universität Darmstadt,
Jahr der Veröffentlichung der Dissertation auf TUprints: 2023
URN: urn:nbn:de:tuda-tuprints-247849
Tag der mündlichen Prüfung: 09.03.23

Abstract

Bioinformatics progresses at an unprecedented pace. At the same time the software implementing the essential algorithms is often incompatible with each other in terms of data input and output. In consequence it can require substantial effort to establish a workflow that combines different programs. Furthermore, the flexibility of such software is usually limited to a relatively small number of options. These circumstances hamper the adaption of these programs to new problems. An alternative approach to command line programs are programming libraries, that enable the user to apply already implemented algorithms and at the same time to harness the full feature spectrum of a programming language.

In this thesis the *Python* bioinformatics package *Biotite* is presented. It unifies popular algorithms from sequence and structure analysis into a flexible library, which is applicable to a wide range of biological questions. Furthermore, new algorithms are presented, enhancing the bioinformatician's toolkit with a novel sequence alignment visualization approach and universally applicable hydrogen prediction method. Finally, via the application of *Biotite* this thesis provides new insights into the molecular mechanism of cation channels and novel evaluation methods for sequencing data from SELEX experiments.

Zusammenfassung

Die Bioinformatik verzeichnet Fortschritte in einem noch nie dagewesenen Tempo. Gleichzeitig sind die Programme, die die zentralen Algorithmen implementieren, häufig in der Dateneingabe und -ausgabe miteinander inkompatibel. Infolgedessen kann es einen erheblichen Mehraufwand erfordern, einen Workflow zu etablieren, der verschiedene Programme kombiniert. Zudem ist die Flexibilität von Kommandozeilenprogrammen in der Regel auf eine übersichtliche Anzahl von Optionen beschränkt. Diese Umstände erschweren die Anpassung dieser Programme an neue Probleme. Eine Alternative zu Kommandozeilenprogrammen sind Programmbibliotheken, die es dem Benutzer ermöglichen, bereits implementierte Algorithmen anzuwenden und dabei gleichzeitig den vollen Funktionsumfang einer Programmiersprache bieten.

In dieser Dissertation wird das *Python*-Bioinformatik-Paket *Biotite* vorgestellt. Es vereint populäre Algorithmen aus der Sequenz- und Strukturanalyse in einer flexiblen Programmbibliothek, die sich auf eine Vielzahl von biologischen Fragestellungen anwenden lässt. Darüber hinaus werden neue Algorithmen vorgestellt, die das Instrumentarium des Bioinformatikers um einen neuartigen Ansatz zur Sequenzalignment-Visualisierung und um eine universelle Methode zur Wasserstoffvorhersage erweitert. Abschließend bietet diese Arbeit durch die Anwendung von *Biotite* neue Einblicke in den molekularen Mechanismus von Kationenkanälen und neue Ansätze zur Auswertung von Sequenzdaten aus SELEX-Experimenten.

Contents

List of Figures

List of Tables

Preface

In the history of bioinformatics many algorithms were invented to generate insight from the increasing wealth of available sequence and structure data. These algorithms have been casted into dedicated programs. However, combining multiple programs to solve a certain biological problem can be challenging as their compatibility with each other and their flexibility is limited.

Here the programming library *Biotite* written for the *Python* programming language is introduced as potential solution to such issues: It offers functionalities for analysis of sequence and molecular structure data. It unifies well-established algorithms in bioinformatics into a fast and consistent package, which allows its users flexible combination of different methods to achieve their desired result. In consequence this library allows developers to simply reuse commonly required functionalities and focus on unique aspects of their software. Furthermore, scientists can use the package to answer their biological questions using the full repertoire of a programming language without the need to write interfaces between different programs.

Thesis Outline

Part I of this thesis will introduce the reader into the current bioinformatics ecosystem for sequence and structure analysis and presents the design principles behind *Biotite*. Then Part II elaborates on the new methods and algorithms that were implemented in *Biotite* and its extensions during the creation of this work, combined with respective application examples. Beyond established methods, these include novel algorithms that enhance the analysis of sequence and structure data. This part first describes data visualization techniques, proceeds with sequence analysis methods and concludes with structure analysis methods. In Part III *Biotite* is used to provide insights into current biological questions in the domains of ion channel research and *in vitro* aptamer selection. Finally, the work is summarized in Part IV.

Reproduction

Figures in this thesis were created with *PyMOL*, *Matplotlib*[1] and *Inkscape*. To support reproducibility of the results shown in this thesis, the complete source code for generation of this dissertation is available as an archive[2]. Additional data required in Chapter 16 is available upon request.

[1] Schrödinger (2017); Hunter (2007).

[2] **Kunzmann** (2022).

Publications

Major parts of this work were previously published in or submitted to peer-reviewed journals. These articles include:

Kunzmann, P. and Hamacher, K. (2018)
Biotite: A Unifying Open Source Computational Biology Framework in Python. *BMC Bioinformatics* 19.1, p. 346. DOI: 10.1186/s12859-018-2367-z.

Kunzmann, P., Mayer, B. E., and Hamacher, K. (2020)
Substitution Matrix Based Color Schemes for Sequence Alignment Visualization. *BMC Bioinformatics* 21.1, p. 209. DOI: 10.1186/s12859-020-3526-6.

Kunzmann, P., Anter, J. M., and Hamacher, K. (2022)
Adding Hydrogen Atoms to Molecular Models via Fragment Superimposition. *Algorithms for Molecular Biology* 17.1, p. 7. DOI: 10.1186/s13015-022-00215-x.

Kunzmann, P., Müller, T., Greil, M., Krumbach, J. H., Anter, J. M., Islam, F., and Hamacher, K. (2022)
Biotite: New tools for a versatile Python bioinformatics library. *BMC Bioinformatics (submitted).*

Part I

Background

Chapter 1
The bioinformatics ecosystem

Bioinformatics is the discipline of applying "*computational techniques to understand and organize the information associated with biological macromolecules.*"[1]. Depending on definition of the term, bioinformatics also extends into the fields of gene expression analysis, systems biology and even microscopy image analysis. However, for the purpose of this thesis the term is restricted to the sequence and structure analysis of biomacromolecules, such as nucleic acids and proteins. In the past decades an ever increasing amount of sequence and structure data has been deposited in public databases. As example for this trend Figure 1.1 shows the annual releases of structures in the *Protein Data Bank* (PDB) [2]. In order to cope with these data and gain new insights from them, researchers have developed a large variety of algorithms and corresponding software - from sequence alignments to structure prediction. The entirety of available bioinformatics software and the interoperability between them will be termed herein the *bioinformatics ecosystem*.

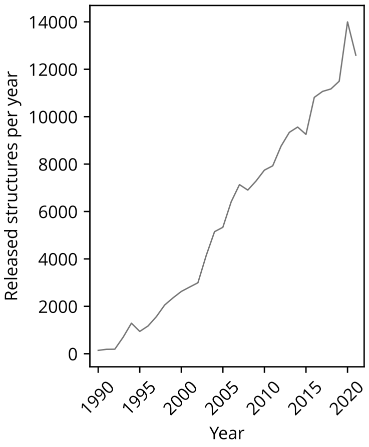


Figure 1.1: Annual releases of PDB structures. The plot was created based on metadata from the entirety of PDB entries.

1.1 Standalone software vs. programming libraries

Two types of software should be distinguished here. On the one hand, there is standalone software, called *tool* herein, made for end users to solve one or different problems based on given data. These are usually command line based[3]: They take given input files and other options and write the result or '*solution*' into the specified output file. An example would be a multiple sequence alignment (MSA) tool that takes a file containing input sequences and optionally a scoring scheme to write the aligned sequences into a new file. Second, there are programming *libraries*, that provide predefined functionalities for other programmers to facilitate writing their own tools in a certain programming language. For example, the library *SeqAn*[4] provides a set of functions for sequence analysis in the programming language *C++*. To use the library for actual problem solving, a tool needs to be written, that combines and applies these offered functionalities for some kind of input data.

The '*Small tools manifesto for bioinformatics*'[5] advocates a bioinformatics ecosystem which is made up of small tools, where each one is specialized for a single focused task. These tools should be "*modular and pluggable*" to allow building workflows upon a combination of them to solve the respective scientific question. However, the combination of pure tools has several shortcomings. Although nowadays the bioinformatics community has settled to a relatively small set of file formats for one type of data, there are still incompatibilities, especially if old

[1] Luscombe, Greenbaum, and Gerstein (2001).

[2] Berman et al. (2000).

[3] Programs that require user input via a graphical user interface are not discussed here, as they cannot be integrated into automatized analysis workflows.

[4] Reinert et al. (2017).

[5] Prins (2014).

tools are involved. For instance, while mmCIF is now the standard format for storing macromolecular structure data in the PDB[6], the deprecated PDB format prevails as primary input and output format in many even modern tools[7]. Furthermore, tools have limited flexibility, as the customization is restricted to a number of command line arguments. Therefore, output data from one tool often needs to be edited before it can be given as input to the following tool. This raises the need for a *script* that performs this editing and conversion (Figure 1.2A). Here the notion of *script* is used to describe a custom tool that is tailored to the specific use case and that can not be easily generalized. To facilitate programming such scripts, a library can be used, i.e. the provided general functionalities are combined and extended with new source code to fit the specific use case. A second application scenario of libraries are the tools itself: Many different tools have common subtasks that need to be performed. For example, independent from the actual structure analysis a tool might perform, all structure analysis tools need to parse structure data from files. Instead of rewriting such common functionality for every new tool, the same library can be used for this subtask by many tools (Figure 1.2B). Finally, the required analysis might be too custom, so no tools fits the purpose, or most tasks in a workflow are relatively simple, so concatenating multiple tools would significantly inflate the runtime due to repeated file input and output. The second point is especially true in structure analysis, as the file formats for structure data are relatively complex compared to the character string-based sequence file formats. In these cases a custom script for the entire analysis or part of it may be a viable option (Figure 1.2C).

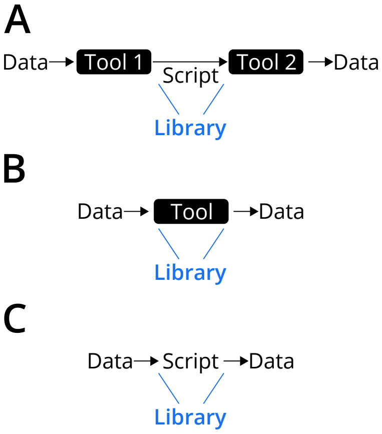


Figure 1.2: Use cases for programming libraries in data analysis workflows. A Usage in a script to prepare output from one tool as input of another tool. **B** Usage as basis for writing a tool itself. **C** Usage for complete data analysis.

1.2 The Python programming language and its scientific computing ecosystem

In recent years the general purpose programming language *Python* matured into a preferred language for scientific computing[8]. In contrast to compiled programming languages as *C/C++*, the source code is executed by the *Python* interpreter skipping the time required for compilation. In combination with dynamic typing[9] and omitted manual memory management, programming in *Python* allows the programmer to focus on the actual program logic. The standard library shipped with the *Python* interpreter already provides a number of functionalities for general purposes. Additional third-party libraries, that can be optionally installed, are called *packages*. To date a myriad of open-source packages for scientific computing have emerged. This section highlights the pillars of the scientific computing ecosystem in *Python* as well as some prominent programming libraries for tasks in bioinformatics.

1.2.1 Efficient vectorized computation with NumPy

The *NumPy* package[10] is at the core of scientific computing in *Python*, as it solves the arguably greatest disadvantage of *Python*: The simplicity of the language comes at the cost of a lot of internal overhead in the

[6] Westbrook et al. (2022).

[7] Das and Baker (2008); Li, Roy, and Zhang (2009); Lindahl et al. (2021).

[8] Pérez, Granger, and Hunter (2011).

[9] A variable may change its type at runtime.

[10] Harris et al. (2020).

Python interpreter, leading to orders of magnitude higher computation times compared to equivalent implementations in compiled programming languages.

Basically *NumPy* provides the `ndarray` data structure: a fixed size array for numerical data[II] that may span multiple dimensions. For example a (3×3) rotation matrix may be represented as `ndarray` containing floating point values with 2 dimensions and a size of 3 in each dimension. The `ndarray` will be called *NumPy array* or *array* throughout this thesis. *NumPy* overcomes the lacking performance of *Python* by vectorization: An arithmetic operation can be applied to the entire *NumPy* array using a single instruction in *Python*. This includes operations involving two arrays, e.g. addition (Figure 1.3), one array and a scalar value, e.g. scalar multiplication, or only a single array, e.g. the sum of its elements. The instruction is dispatched to efficient compiled routines that apply the respective operation to each element of the array(s). This way a performance close to compiled programming languages can be achieved, while the simplicity of *Python* is preserved. For example, the computation of the dot product between two arrays `a` and `b` representing vectors of equal length n could be written as '`sum(a*b)`'. Here only two functions are called from *Python*: the multiplication and the sum. The n arithmetic operations, required for each of these two functions, run in compiled routines. In addition to arithmetic, *NumPy* provides a comprehensive set of linear algebraic operations and other basic functionalities such as sorting or finding the maximum value of an array.

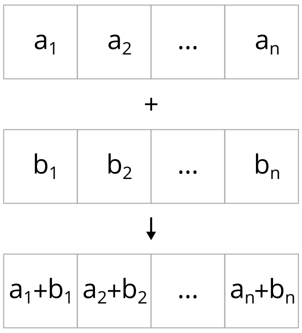
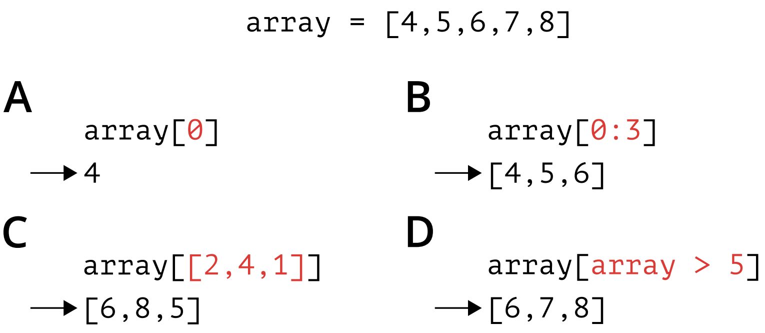


Figure 1.3: Vectorized addition of two arrays. The addition of two arrays *a* and *b* is schematically depicted.

Assessing an element in an array is called *indexing*, which is done by providing a position (the index) of the element in the array (Figure 1.4A). Alternatively, entire subarrays can be obtained by providing a range (Figure 1.4B) or array (Figure 1.4C) of indices. In addition an array can be filtered based on one or multiple conditions (Figure 1.4D): The condition is expressed as *boolean mask*, an array of boolean values with the same size as the array to be filtered. When the *boolean mask* is used for indexing, all elements where the mask is *true* are part of the resulting subarray. Since a *boolean mask* is a *NumPy* array itself, logical operators can be a applied to it, allowing for complex conditions.



```
array = [4,5,6,7,8]
```

A

```
array[0]
→ 4
```

B

```
array[0:3]
→ [4,5,6]
```

C

```
array[[2,4,1]]
→ [6,8,5]
```

D

```
array[array > 5]
→ [6,7,8]
```

Figure 1.4: Ways to index a *NumPy* array. The elements of the example array are shown at the top. The respective index is shown in red. The resulting value(s) are shown next to the arrow. Note that indexing is 0-based, i.e. the first element has the index 0. **A** Integer. **B** Slice. **C** Index array. **D** Boolean mask.

[II] This includes: booleans, integers, floating point numbers, complex numbers and fixed size character strings.

1.2.2 Cython allows writing easily maintainable C-extensions

Vectorized arithmetic operations are limited to scenarios, where many values can be processed independent of each other. Fortunately, the *Python* interpreter exposes an application programming interface (API), that allows writing functions in the compiled language *C*, which can be later used in *Python* code. Those functions are called *C-extensions*. This API is also used by *NumPy* to call the optimized routines mentioned above. However, writing such extensions requires much effort to integrate functions and objects written in *C* properly into *Python*. *Cython*[12] is a *Python* dialect for writing *C-extensions*, that conceals most of this complexity. Although the general syntax is similar or even equal to *Python* itself, a higher performance can be achieved by enforcing static typing or pure *C*-operations in performance-critical parts of the source code.

1.2.3 SciPy and its SciKits

While *NumPy* provides basic functionality for vectorized data handling, *SciPy* offers "*fundamental algorithms for scientific computing in Python*"[13]. These include methods that are commonly used in different fields of science, such as statistic methods or common optimization algorithms. *SciPy* can be supplemented by additional packages specialized for a certain scientific domain, so called *SciKits*. One of the most famous *SciKits* is *Scikit-learn*[14], which provides functionalities for machine learning. *Scikit-bio*, which offers analyses for sequence bioinformatics, is another example.

1.2.4 Data visualization with Matplotlib

Matplotlib[15] is the *de facto* standard library for creating 2D and, in a limited manner, 3D graphics in *Python*. Although many of its capabilities focus on plots, the high flexibility of the package allows scripted creation of vector graphics in general. To facilitate the production of publication-ready plots, the *Seaborn* library offers a "*high-level interface to Matplotlib*"[16], i.e. it provides convenience functions for common types of plots, hiding much of *Matplotlib*'s flexibility but also complexity.

1.2.5 Analysis of molecular molecular dynamics simulations

Molecular dynamics (MD) simulate the movement of atoms in a molecular system[17] over time. The resulting trajectories contain time series of atom coordinates. Popular packages for the analysis of these trajectories include *MDAnalysis* and *MDTraj*[18]. Both packages make use of *NumPy* to enable efficient analysis of the large number of models in a typical trajectory.

[12] Behnel et al. (2011).

[13] Virtanen et al. (2020).

[14] Pedregosa et al. (2011).

[15] Hunter (2007).

[16] Waskom (2021).

[17] such as a solvated protein

[18] Gowers et al. (2016); McGibbon et al. (2015).

1.2.6 Biopython as pioneer of bioinformatics in Python

Biopython is a bioinformatics library that focuses on analysis of sequence and structure data. Its advent dates back to 1999[19], when *NumPy* and most of the current scientific computing ecosystem was not established, yet. In consequence, the package rarely uses *NumPy* arrays for data representation. Hence, the implemented as well as custom analyses on these data have limited performance. Furthermore, the data representation is partly inconsistent: For example structure superimposition requires conversion of a `Structure` object to a list of `Atom` objects. Nevertheless, *Biopython* is established in the bioinformatics community and is primarily used as "*glue between the multiple different data formats*"[20]. Although packages with similar scope have also been developed for other programming languages as well[21], *Biopython* is by far the most popular one[22].

[19] Cock et al. (2009).

[20] taken from '*Podcast.__init__*' podcast episode 125 with three maintainers of the *Biopython* project (accessed 2022-10-06)

[21] Stajich et al. (2002); Grant et al. (2006); Goto et al. (2010); Lafita et al. (2019).

[22] based on number of citations and *GitHub 'stars'* on 2022-11-26

Chapter 2
Preliminary work

The high popularity of *Biopython* indicates the usefulness of a *Python* package for general sequence and structure analysis for the bioinformatics community. However, especially its low performance limits its potential. Therefore an endeavor was started to build the package *Biotite* as modern alternative to *Biopython*. It places the usage of *NumPy* arrays at its core to harness the performance of vectorized operations and to achieve interoperability with other packages in the *Python* scientific computing ecosystem. It aims to implement algorithms and support file formats that make up the '*backbone*' of sequence and structure bioinformatics. The *Biotite Python* package was initially created as part of a Master's thesis[1]. The general structure of the *Python* package is described and its previous scope of functionalities is outlined in this chapter.

2.1 Package organisation

The *Biotite* package is divided into four interconnected subpackages (Figure 2.1) that are able to handle a major part of the typical data analysis workflow in bioinformatics. Using the `biotite.database` subpackage data entries can be searched in and selected files can be downloaded from biological databases such as *NCBI Entrez* and *RCSB PDB*[2]. Sequence and structure files can be read for further analysis with the `biotite.sequence` and `biotite.structure` subpackage. These implemented functionalities include a variety of data analysis, editing and visualization capabilities which are further outlined in the following sections. Where the functionalities implemented into *Biotite* do not reach far enough, `biotite.application` provides interfaces to external applications, including locally installed software or web services. These interfaces are seamless: Invocations of the application and any necessary output evaluation are handled internally.

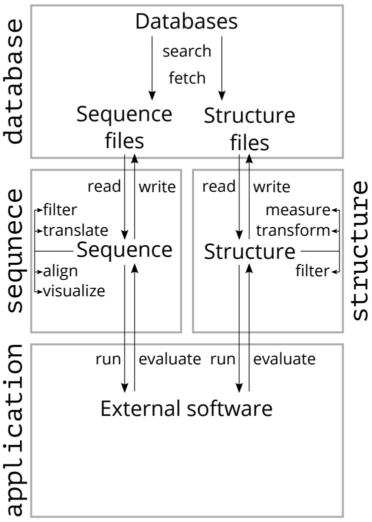


Figure 2.1: Overview of the Biotite package organization. The `database` subpackage allows searching in biological databases and fetching files from these online resources. The structure and sequence files can be loaded into the internal data representation using the `sequence` and `structure` subpackage, respectively. While these subpackages offer a multitude of data analysis and editing functionalities, the `application` subpackage provides seamless interfaces to commonly used software and web services for further analysis.

2.2 Data representation

A distinguishing point of *Biotite* to comparable packages like *Biopython* is its *NumPy* array based data representation whenever possible. This increases the performance of most operations to a level comparable to compiled programming languages and allows implementation of custom analyses in a simple way by using the already existing algorithms in *NumPy*, *SciPy* and other packages in the scientific computing ecosystem.

[1] **Kunzmann** (2018).

[2] Sayers et al. (2022); Burley et al. (2021).

2.2.1 Sequences

The two most common types of sequences in biology are nucleotide and amino acid sequences. While these are typically represented as character strings as for example in *FASTA* files, *Biotite* uses a different representation: When a `Sequence` object is created, it maps each symbol [3] in the sequence into an integer, the so called *symbol code*, using an associated `Alphabet` object, that defines the allowed symbols for that sequence type (Figure 2.2). The symbol code is simply the index of the respective symbol in the list of allowed symbols. The resulting symbol codes for the entire sequence are stored in a *NumPy* array as the so called *sequence code*.

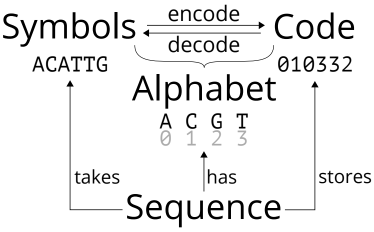


Figure 2.2: Sequence encoding in *Biotite*. When a `Sequence` is instantiated, the input symbols are converted into a sequence code using an associated `Alphabet` and stored inside the `Sequence`. Adapted from Kunzmann and Hamacher (2018) (CC BY 4.0).

Although this encoding requires additional computation time, it has multiple advantages compared to the string representation:

- More exotic sequence types like pharmacophores or structure embeddings can be represented, even if the number of symbols in the alphabet exceed the 95 printable ASCII characters.
- Most functionalities for sequence data in *Biotite* are agnostic with respect to the sequence type, since they operate completely on the *sequence code*.
- For alignment purposes the sequence code can be directly used as index to substitution matrices (see Section 2.4.2). This saves computation time since no additional conversion is necessary.
- The calculation of k-mers is fast (see Section 6.2.1). This has implications on the speed of nucleotide sequence translation and alignment searches.

2.2.2 Sequence annotations

Sequence annotations describe the functions of regions within a sequence, for example the location and product of a gene. *Biotite* represents these as `Annotation` objects, which are collections of `Feature` objects, that, in turn, describe the location and functionality of such a sequence region. `Annotation` objects are one of the few objects in *Biotite* that do not internally use a *NumPy* to store the data: As each sequence position may have an arbitrary number of associated features, no meaningful way to save the data in a vectorized manner was found. However, the amount of annotation data is usually relatively small compared to the sequence data itself. Thus, the performance impact of annotation data handling is minor in most cases.

2.2.3 Structure models

Molecular structure models require a combination of different data types. While most importantly each atom is described by its coordinates in the three spatial dimensions, other annotations like its containing peptide/nucleotide chain, the residue and the atom name are crucial to distinguish the atoms from each other. One possible data

[3] i.e. the character

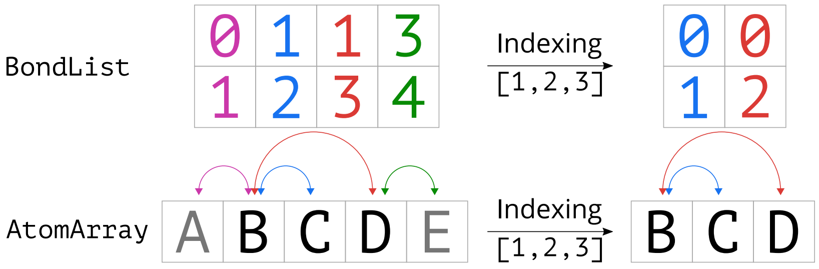


Figure 2.3: `BondList` **indexing behavior.** An example `BondList` with reference `AtomArray` including the atoms A-E is shown. The bond type is not displayed. The `BondList` contains 0-based indices pointing to bonded atoms in the reference `AtomArray`, signified by color coding. After applying an index to both, the `BondList` and the `AtomArray`, the indices in the `BondList` still correctly depict the bonds of the remaining atoms. The removed atoms are shown in gray.

representation for a structure model containing n atoms could have been a list of length n, where each list element describes the coordinates and further annotations of a single atom. However, this approach would prevent vectorized computing using *NumPy* and thus negatively impact the computation performance. Instead, a structure is represented by multiple *NumPy* arrays of length n, where each array contains an annotation type or the coordinates. The arrays are bundled into an `AtomArray` object for single-model structures or an `AtomArrayStack` for multi-model structures, such as NMR models or frames from a MD simulation. The key difference between an `AtomArray` and `AtomArrayStack` is the number of dimensions in the coordinate array: While an `AtomArray` uses an $(n{\times}3)$-dimensional array, an `AtomArrayStack` requires $(m \times n \times 3)$ dimensions to represent the coordinates in m different models. These two types of objects are the universal structure representations used throughout the `structure` subpackage. In contrast to *MDAnalysis* and *MDTraj*[4], which both use a string-based atom selection algebra, atom selections in *Biotite* use *NumPy*-based indexing (see Section 1.2.1). Thus, the user can take advantage of utilities from *NumPy* and other packages as means to customize the selection.

2.2.4 Covalent bonds

Covalent bonds between atoms in an `AtomArray`[5] are represented by `BondList` objects. A `BondList` stores indices that point to bonded atoms in a reference `AtomArray` as $(k \times 3)$-dimensional *NumPy* array. Each of the k elements represents a bond, specified by three values: the index to the first atom i involved in the bond, the index to the second atom j $(j > i)$, and an integer representing the type of the bond [6].

When indexing an `AtomArray`, i.e. selecting atoms, the index is simply propagated to the underlying arrays containing coordinates and annotations. However, when applying the same index to the corresponding `BondList`, the index is not directly forwarded to the internal array of bonds. Instead the internal array is adjusted, so that the atom indices still point to the same atoms in the filtered `AtomArray` (Figure 2.3): Atom indices are shifted accordingly and bonds involving atoms, that were filtered out, are removed.

[4] Gowers et al. (2016); McGibbon et al. (2015).

[5] Bonds work in the same way for `AtomArrayStack` objects and are omitted in this section for brevity.

[6] single, double, aromatic, etc.

2.3 File formats

The support for a wide range of different data file formats is key for bioinformatics libraries to enable handling of data from many different sources and to make edited data available to a wide range of tools for downstream analysis. Prior to the work on this thesis *Biotite* already supported several sequence and structure file formats. Sequences can be read from and written to *FASTA* files or, if sequence annotations are required, *GenBank* files. For structures *Biotite* supports the *PDBx/mmCIF* format, which is the standard format of the PDB[7]. However, as many programs still require the legacy *PDB* format, it is also supported. For fast file loading and transmission, the *MMTF* format[8] is available. For the purpose of MD simulation analysis various trajectory formats are also supported via the *MDTraj* package[9].

[7] Adams et al. (2019).

[8] Bradley et al. (2017).

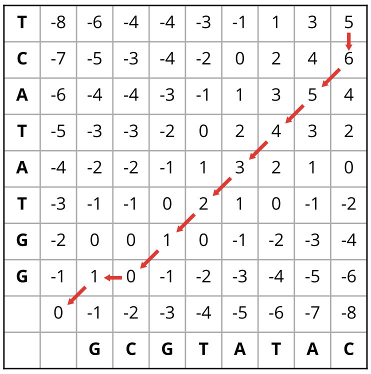
[9] McGibbon et al. (2015).

2.4 Sequence analysis

2.4.1 Sequence types

Inheriting from the `Sequence` superclass, `NucleotideSequence` and `ProteinSequence` represent the most common sequence types. `NucleotideSequence` objects can form their reverse complement counterparts or can be translated into a `ProteinSequence` using a given codon table. Note that `NucleotideSequence` represents both, DNA and RNA sequences, although the corresponding `Alphabet` contains the symbols 'A', 'C', 'G' and 'T'. The reason is, that DNA and RNA sequences convey the same information. Only the corresponding character string would be different [10], which can be easily substituted using standard *Python* functionality.

[10] 'T' replaced by 'U' for RNA



T	-8	-6	-4	-4	-3	-1	1	3	5
C	-7	-5	-3	-4	-2	0	2	4	6
A	-6	-4	-4	-3	-1	1	3	5	4
T	-5	-3	-3	-2	0	2	4	3	2
A	-4	-2	-2	-1	1	3	2	1	0
T	-3	-1	-1	0	2	1	0	-1	-2
G	-2	0	0	1	0	-1	-2	-3	-4
G	-1	1	0	-1	-2	-3	-4	-5	-6
	0	-1	-2	-3	-4	-5	-6	-7	-8
		G	C	G	T	A	T	A	C

Figure 2.4: Sequence alignment using dynamic programming. The figure shows the aligned sequences alongside the dynamic programming table containing the scores. The arrows depict the alignment trace, i.e. the path through the table that maximizes the score.

2.4.2 Pairwise alignment

Homology between two sequences can be investigated using pairwise sequence alignments. Optimal alignments can be achieved using dynamic programming[11] requiring two scoring schemes: the gap penalty relating to the evolutionary probability of insertions or deletions, and a substitution matrix providing the log-odds of substitutions for each pair of symbols[12,13]. The latter one is represented by a `SubstitutionMatrix` object, which stores log-odds scores in an $(m \times n)$-dimensional *NumPy* array and contains two `Alphabet` objects with m and n symbols, respectively. The `Alphabet` objects define the types of `Sequence` that can be aligned using this `SubstitutionMatrix`. Usually, both alphabets are identical, since typically nucleotide sequences are aligned with nucleotide sequences and amino acids sequences with amino acids sequences. However, the possibility to supply two different alphabets allows for more exotic applications, like the alignment of amino acid sequences with a structural alphabet. Since sequences are stored as sequence code, i.e. the indices

[11] Needleman and Wunsch (1970); Smith and Waterman (1981).

[12] e.g. amino acids or nucleobases

[13] Henikoff and Henikoff (1992); Keul et al. (2017).

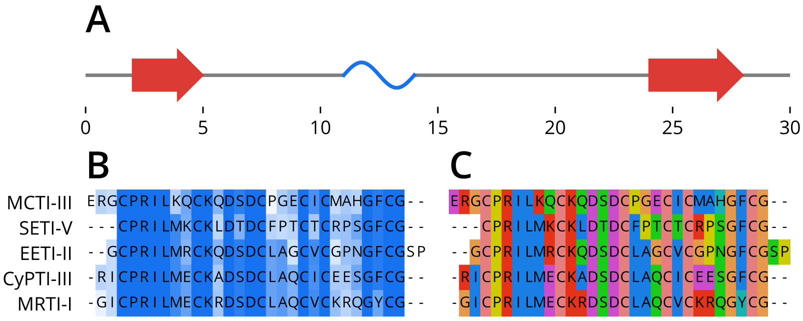

A
0
5
10
15
20
25
30
B
C
MCTI-III
SETI-V
EETI-II
CyPTI-III
MRTI-I
ERGCPRILKQCKQDSDCPGECICMAHGFCG--
---CPRILMKCKLDTDCFPTCTCRPSGFCG--
--GCPRILMRCKQDSDCLAGCVCGPNGFCGSP
-RICPRILMECKADSDCLAQCICEESGFCG--
-GICPRILMECKRDSDCLAQCVCKRQGYCG--

Figure 2.5: Sequence visualizations available in *Biotite*. A selection of trypsin inhibitor sequences was taken as example. The sequences including annotation data were downloaded from the UniProt database. **A** Feature map of the EETI-II secondary structure. α-helices are shown in blue, β-strands in red. **B+C** MSA of trypsin inhibitor sequences. The coloring is either based on sequence conservation in an alignment column (**B**) or based on the type of symbol (**C**).

of symbols in the alphabet, the $(m \times n)$ score matrix can be directly indexed with two symbol codes to obtain the corresponding substitution score for the respective two symbols. In the course of the sequence alignment, a dynamic programming table is filled (Figure 2.4). The aligned symbols correspond to the path through the table that maximizes the score. The result is one or multiple[14] `Alignment` objects that stores the aligned sequences as well as indices that point to their aligned symbols: the trace.

2.4.3 Visualization

For visual analysis *Biotite* provides different visualization functionalities using the *Matplotlib* plotting library. These include feature maps (Figure 2.5A) and sequence alignments (Figure 2.5B+C). Alignments can be colored either by similarity in the alignment columns (Figure 2.5B) or by type of the symbol (Figure 2.5C). For this purpose different color schemes are available[15].

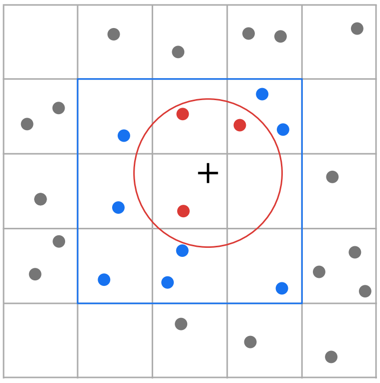


Figure 2.6: Finding atoms within cutoff distance using a cell list. For visualization purposes only a two-dimensional cell list is depicted. The atom positions are marked by dots, the position of interest is highlighted with a cross. First, all atoms in the cell of the position of interest and its adjacent cells are filtered (blue square). Second, atoms within cutoff distance (red circle) are identified by measuring the Euclidean distance to the filtered atoms only.

2.5 Structure analysis

2.5.1 Cell list

Finding atoms in vicinity of a given position, defined by a cutoff distance, is a common task for various structure analysis methods such as partial charge estimation or surface area measurement[16]. The most simple way to achieve this aim is checking the distance of each atom to the position of interest. However, this procedure requires $\mathcal{O}(n)$ run time for an `AtomArray` with n atoms. If this needs to be performed for many positions, for example to find all pairs of nearby atoms, this approach quickly becomes too expensive for larger structures. *Biotite* provides a cell list as alternative (Figure 2.6): It divides the three-dimensional space into cells, where the atoms are sorted in according to their positions. To filter atoms that are guaranteed within the cutoff distance of a position of interest only adjacent cells[17] need to be considered, reducing the time complexity to $\mathcal{O}(1)$. Then only the distances to the filtered atoms need to be calculated. The `CellList` implementation is a versatile functionality used throughout the *Biotite* `structure`

[14] Multiple solutions that maximize the score may exist. To avoid long run times the maximum number of computed paths can be limited.

[15] Larkin et al. (2007); Waterhouse et al. (2009).

[16] Rappe and Goddard (1991); Shrake and Rupley (1973).

[17] This is only true, if the cutoff distance is equal to or lower than the cell size, otherwise more cells need to be checked.

subpackage for efficient computation of vicinity.

2.5.2 Measurements

Based on atom coordinates, a variety of measurements can be performed for an `AtomArray` or `AtomArrayStack`. These range from measurements of distances, angles and dihedrals to more complex measures such as the solvent accessible surface area[18] and secondary structure elements[19]. Specifically for the multiple models of an `AtomArrayStack` the root mean square deviation (RMSD) and root mean square fluctuation (RMSF) with respect to a reference model can be calculated. These are defined as

$$\begin{aligned} \mathrm{RMSD}(i) &= \sqrt{\frac{1}{n}\sum_{j=1}^{n}(\vec{x}_{ij} - \vec{x}_{\mathrm{ref},ij})^2}, \\ \mathrm{RMSF}(j) &= \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\vec{x}_{ij} - \vec{x}_{\mathrm{ref},ij})^2}. \end{aligned} \tag{2.1}$$

$\vec{x}$ are the atom coordinates, where the index i specifies the model and j values from the flattened atom coordinates [20] within a model. $\vec{x}_{\mathrm{ref}}$ are the corresponding coordinates of the reference model. For RMSF calculation the time average of $\vec{x}$ over all models is usually taken as $\vec{x}_{\mathrm{ref}}$.

2.5.3 Transformations

Biotite offers functionalities for structure transformations including simple rotations and translations as well as superimposition of one structure model onto another model minimizing their RMSD[21]. Besides, custom structure editing is also possible, as the coordinates in `AtomArray` and `AtomArrayStack` objects are accessible to the user.

2.6 Interfaces to external applications

As at the time of the preliminary work only pairwise sequence alignments were directly implemented in *Biotite*, interfaces to widespread software were created to enable the user to also perform MSAs[22]. Other application interfaces include an interface to the *BLAST*[23] web service, allowing automation of alignment searches, and *DSSP*[24] to measure secondary structure elements in molecular models. These interfaces are implemented as subclasses of the *Biotite* `Application` superclass. They feature seamless interaction with the respective software, as the complexity of communicating with the software is handled internally: For local tools, data input/output uses temporary files and the respective tool is called as separate process. In the case of *BLAST*, the REST API of the web service is used.

[18] Shrake and Rupley (1973).

[19] Labesse et al. (1997).

[20] $[x_1, y_1, z_1, x_2, y_2, z_2, ...]$

[21] Kabsch (1976); Kabsch (1978).

[22] Edgar (2004); Katoh et al. (2002); Sievers et al. (2011).

[23] Altschul et al. (1990).

[24] Kabsch and Sander (1983).

2.7 Software accessibility

General accessibility of software including its source code is paramount for reproducibility of scientific results[25] and widespread usage of a software. Therefore, *Biotite* is devised as open source project. The source code is hosted at *GitHub*[26] under the 3-Clause BSD License. To facilitate usage and reproduction, binaries of the *Biotite* library are available via both, the standard Python package manager *pip* as well as the general scientific software package manager *Conda* for *Windows*, *Linux* and *MacOS*. The API of the library is extensively documented including tutorials and examples to increase the accessibility for novices [27].

[25] Ivie and Thain (2018).

[26] `https://github.com/biotite-dev/biotite`

[27] `https://www.biotite-python.org/`

Part II

Implemented methods

This part of the thesis covers the continuation of the development of *Biotite* after the initial publication outlined in Chapter 2. Instead of describing incremental improvements of existing functionalities introduced in the course of this work, this part of the thesis illuminates major novel functionalities and their underlying algorithms. For a more detailed listing of all changes, the reader can review the changelog of the library starting from version `0.8.0`. While most of these new components were directly incorporated into the *Biotite* package, some of them were too specialized for a specific task or use novel algorithms. Adding these to *Biotite* directly would arguably clutter the package. Therefore, these functionalities were packaged into separate *extension packages* that build upon *Biotite*. This will be explicitly stated in the respective chapter, if this is the case. The URL of the code repositories and documentation for *Biotite* and its extension packages is listed in Table A.1.

To illustrate the potential applications of the new features, each of the following chapters is accompanied by an '*Application example*' section. These depict the use of the respective functionality to solve a distinct biological problem. Although the presented specific problems are purely artificial, the functionalities can be easily adapted and combined with further analyses to answer actual current biological questions. While these sections roughly describe the analysis method and present its results, the detailed source code to reproduce the results is included in the `examples` directory of the published companion workflow[28]. The specific script file name for an application example is given in the beginning of the respective section as shown on the left.

Companion source code:
`<script_name>.py`

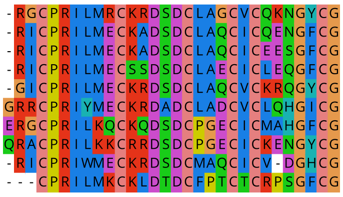
[28] **Kunzmann** (2022).

Chapter 3
Alignment color scheme generation

This chapter builds upon ideas and figures from a published journal article[1]. The content is licensed under the CC BY 4.0 license. Changes were made on size and arrangement of figure elements.

3.1 Introduction

As outlined in Section 2.4.3, *Biotite* is able to color symbols in sequence alignment visualizations according to the type of the symbols. This is especially useful for multiple sequence alignments of amino acid sequences, where the color may depict physicochemical properties such as size, charge and hydrophobicity of the respective amino acid. This representation of alignments is commonly used to allow scientists visual inspection of protein homologies and differences. Beside *Biotite* popular programs such as *MSAViewer*, *JalView* or *ClustalX*[2] are able to create these alignment illustrations, often featuring their own color schemes. Figure 3.1 shows an example alignment using the color scheme from *ClustalX*.



```
-RGCPRILMRCKRDSDCLAGCVCQKNGYCG
-RICPRILMECKADSDCLAQCICQENGFCG
-RICPRILMECKADSDCLAQCICEESGFCG
-RICPRILMECSSDSDCLAECICLEQGFCG
-GICPRILMECKRDSDCLAQCVCKRQGYCG
GRRCPRIYMECKRDADCLADCVCLQHGICG
ERGCPRILKQCKQDSDCPGECICMAHGFCG
QRACPRILKKCRRDSDCPGECICKENGYCG
-RICPRIWMECKRDSDCMAQCICV-DGHCG
---CPRILMKCKLDTDCFPTCTCRPSGFCG
```

Figure 3.1: Example alignment visualized using default color scheme from *ClustalX*. The underlying alignment comprises a selection of trypsin inhibitors. The alignment was created using *MAFFT*. Gaps are colorless, since they appear in absence of symbols. Adapted from Kunzmann, Mayer and Hamacher (2020) (CC BY 4.0).

These color schemes are typically handcrafted by scientists considering multiple physicochemical properties of the proteinogenic amino acids in the design. However, this approach is highly subjective: How is the size of an amino acid defined and how is it weighed compared to for example the physiological charge of the amino acid? Even if some properties could be quantified, the problem of mapping this multitude of properties into the three-dimensional color space would still persist.

Therefore, an impartial source to highlight similarities and differences between amino acids, and symbols in general, is sought: the evolutionary substitution probabilities in the form of a substitution matrix. This chapter describes an algorithm to compile the substitution scores into a color scheme, where similar symbols have similar colors. The method is implemented in the *Biotite* extension package *Gecos*, which provides a simple command-line interface (CLI) as well as a *Python* interface intended for integration with *Biotite*.

3.1.1 Color spaces

Technically, colors are described using coordinates in a color space: a three-dimensional vector space that comprises all colors that can be depicted within that space. The arguably most popular color space is *standard RGB* (*sRGB*)[3] which is generally used to define colors for display devices and describes colors as addition of red, green and blue light. How-

[1] **Kunzmann**, Mayer, and Hamacher (2020).

[2] Yachdav et al. (2016); Waterhouse et al. (2009); Larkin et al. (2007).

[3] CIE (1999).

ever, *sRGB* is not perceptual uniform[4], i.e. the Euclidean distance between two points in the color space is not proportional to the perceived color change.

For this reason, color spaces like *L*a*b** were conceived[5] to achieve good perceptual uniformity (Figure 3.2). The *L*a*b** spaces comprise the three dimensions

- the lightness L^* ranging from 0 (black) to 100 (white),
- the red-green component a^* ranging from unlimited negative values (green) to unlimited positive values (red) and
- the blue-yellow component b^* ranging from unlimited negative values (blue) to unlimited positive values (yellow).

Note that while the a^* and b^* dimensions are not limited, only a finite subspace can be represented in *sRGB* and thus on display devices. These conversion limits on a^* and b^* become more narrow towards the extreme ends of the L^* value.

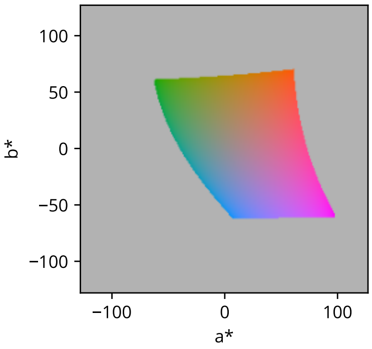


Figure 3.2: Excerpt of the *L*a*b color space.** The figure shows a cross section of the three-dimensional color space as L^* = 60. The displayed color is the color of the respective point in the space. The gray area represents colors that cannot be represented in *sRGB*. Adapted from Kunzmann, Mayer and Hamacher (2020) (CC BY 4.0).

3.2 Methods

In principle the algorithm of *Gecos* defines a score function based on a given substitution matrix that assesses the colors[6] for an alphabet of symbols: The score is better for a set of colors, where the relative perceptual differences between the colors is closer to the relative evolutionary distances between corresponding symbols. Using this score function, an optimizer is employed to create a set of colors that fulfills this criterion as good as possible.

3.2.1 Score function

For a given set of *L*a*b** colors for an alphabet of interest[7], called *color conformation*, the score function computes a score: More negative values indicate a better color conformation. For the calculation it additionally requires a symmetric substitution matrix M[8], and a contrast factor, which is explained later.

Since, M is symmetric, only the lower triangle of the matrix is considered. Initially, M is converted into a triangular distance matrix

$$D'_{ij} = \frac{(M_{ii} + M_{jj})}{2} - M_{ij}. \tag{3.1}$$

Since the similarity of a symbol to itself is maximal, $M_{ii} \geq M_{ij}$ for $i \neq j$. Hence, all entries of D' are non-negative. Typically, substitution matrices are multiplied by an arbitrary factor to obtain integer values that retain sufficient precision. Consequently, a direct usage of D' in the score function would also result in an arbitrary scale of the score function, that would interfere with the later optimization process. Therefore, D' is scaled into $D = D' / \langle D' \rangle$, so that the average distance is 1. In this section the operator $\langle X \rangle$ gives the arithmetic mean of the entries in non-zero diagonals of a matrix.

[4] Bernard et al. (2015).

[5] CIE (2019).

[6] as they would be used in an alignment visualization

[7] e.g. the amino acid alphabet

[8] Matrices with two different alphabets, as outlined in Section 2.4.2, cannot be used here.

While D_{ij} gives the ideal distances between two symbols i and j, the triangular matrix C gives the actual perceptual distances for the given color conformation. By default, *Gecos* uses the *CIEDE2000* formula[9] for calculation of perceptual differences, which is omitted for the sake of brevity, but can be approximated by the Euclidean distances between the colors[10]. Thus,

$$C'_{ij} \approx \sqrt{(L^*_i - L^*_j)^2 + (a^*_i - a^*_j)^2 + (b^*_i - b^*_j)^2}. \tag{3.2}$$

Analogous to D', C' is scaled to obtain $C = C' / \langle C' \rangle$.

As the entries of both matrices D and C average 1, the matrices can be set into direct relation. This is contrasted with D' and C', which use different scales to quantify their distances and thus cannot be directly compared. The score function penalizes deviations of C from the ideal distances D using a harmonic potential. Therefore,

$$S_H = \sum_{ij} \left(C_{ij} - D_{ij}\right)^2. \tag{3.3}$$

While the score S_H would be sufficient to achieve color conformations that represent evolutionary distances, the colors might be barely distinguishable with a large portion of the color space unused, as only relative distances are considered but not the absolute color distances. However, for a usable color scheme distinguishable colors are required. Hence, a second term is added to the score function that penalizes color conformations with low contrast, i.e. with low average perceptual differences:

$$S_C = \frac{f_C}{\langle C' \rangle}. \tag{3.4}$$

The contrast factor f_C is a parameter supplied by the user[11], that balances contrast and conformity with the substitution matrix: At high values a high contrast is achieved, by driving symbols to the edges of the color space, at the cost of worse representation of symbol similarity. Both terms are combined resulting in the score function

$$S = S_H + S_C. \tag{3.5}$$

3.2.2 Optimization

To find a color conformation that minimizes the score function S the *Metropolis-Monte-Carlo* algorithm[12] is used (Figure 3.3). Starting from an initially randomized color conformation, the vector representing the current color conformation is updated by adding a random value between $-\alpha$ and α to each component of the vector. Hence, α describes the step size. If S for the new color conformation is lower than S for the prior conformation, the new conformation is accepted. Otherwise, it is accepted with a probability that exponentially decreases with increasing score difference weighted with the factor β. This process is repeated a predefined number of times n, where the color conformation

[9] CIE (2001).

[10] CIE (2019).

[11] *Gecos* provides a default value of $f_C = 700$. However this default value is based on subjective visual appeal.

[12] Metropolis et al. (1953).

```
procedure OPTIMIZE($S, X, n, \alpha_0, \alpha_1, \beta_0, \beta_1$)
    $\vec{x} \leftarrow$ draw random color conformation from $X$
    $s \leftarrow S(\vec{x})$
    for $i \leftarrow 1, n$ do
        $\alpha \leftarrow \alpha_0 \cdot (\alpha_1/\alpha_0)^{i/n}$
        $\beta \leftarrow \beta_0 \cdot (\beta_1/\beta_0)^{i/n}$
        repeat
            $\vec{\xi} \leftarrow$ draw uniform random values $\in [-1, 1]$
            $\vec{x}_{\text{new}} \leftarrow \vec{x} + \alpha\vec{\xi}$
        until $\vec{x}_{\text{new}}$ is within $X$
        $s_{\text{new}} \leftarrow S(\vec{x}_{\text{new}})$
        $p \leftarrow$ draw uniform random value $\in [0, 1]$
        if $p < e^{-\beta(s_{\text{new}} - s)}$ then
            $s \leftarrow s_{\text{new}}$
            $\vec{x} \leftarrow \vec{x}_{\text{new}}$
        end if
    end for
    return $\vec{x}$
end procedure
```

Figure 3.3: Pseudocode for simulated annealing algorithm. The figure shows the algorithm that optimizes a color conformation with respect to the score function S. X gives the allowed color space which is restricted to the *sRGB* convertible colors and may be subject to further constraints, e.g. a limited lightness range. n denotes the number of iterations. α_0, α_1 and β_0, β_1 give the start and end values for α and β, respectively. In the actual implementation, arrays containing $\vec{x}$ and s of each iteration are also returned.

is replaced with color conformation from the previous step, if it was accepted. While at low values of β, the algorithm is able to '*jump over*' local minima in the score landscape, high values steer the color conformation into a minimum. For a successful optimization both properties are desireable: In the beginning, the color conformation should be able to leave local score minima and approach the global optimum, where it should settle in the end of the optimization. Hence, the *simulated annealing* variant of the algorithm described above is used[13]. Rather than keeping α and β constant over the course of optimization, α is slowly decreased and β increased each step in an exponential manner. In consequence, the optimization method depends on four parameters: α and β at the start and end of the optimization, respectively.

3.2.3 Meta-optimization

To find optimal values for these parameters, a meta-optimization was performed. As benchmark a color scheme generation for the popular *BLOSUM62* matrix in unconstrained *L*a*b** space using 10000 optimization steps[14] was chosen. Now those parameters were sought, that achieve the best score S at the end of a corresponding optimization run. Because the optimization algorithm is stochastic in nature, the generated color conformations and the corresponding scores are subject to noise. Therefore, not the score of a single optimization run was taken as objective for meta-optimization, but the median from 100 runs.

Note that sampling the optimization parameters is costly, since 100 entire optimization runs are required for each sampled data point. Therefore, a *tree-structured Parzen estimator*[15] from the hyperparameter opti-

[13] Kirkpatrick, Gelatt, and Vecchi (1983).

[14] An optimization with 10000 steps can be run on modern commodity hardware within approximately one minute.

[15] Bergstra et al. (2011).

mization library *Optuna*[16] is used to obtain a well-performing set of parameters with less sampling compared to simple random sampling. In short, the algorithm builds a probabilistic model of the score landscape from previous samples, to select new samples that are likely to score better than those previous samples.

Note that this meta-optimization is not part of the *Gecos* package. However, the found optimal values are used as default values for both the CLI and application programming interface (API).

3.2.4 Implementation details

Gecos implements the score function and optimizer as separate objects. This allows the replacement of the score function via the *Python* API to include other aspects of an appealing color scheme not envisioned in this chapter. Besides a score function, the optimizer requires a color space as input, that allows the user to mask certain regions of the *L*a*b** space. For example its recommended to restrict *L** to a range of approximately size 15 to obtain a visually appealing result. If *L** is unconstrained, the generated color schemes will inevitably contain very bright and dark colors that poorly contrast with the background and symbols, respectively, in an alignment visualization. Furthermore, the color itself can be optionally constrained for each symbol, to set a color according to personal flavor or to reuse some colors from a previous optimization.

The data representation in *Gecos* integrates into the one used by *Biotite*: Substitution matrices are represented by `SubstitutionMatrix` objects and the optimized color conformations are *NumPy* arrays of *RGB* values, that can be directly used in *Biotite*'s functions for alignment visualization. Multiple pre-generated schemes with different properties are also available via keyword in *Biotite*, eliminating the need to install and run *Gecos* in many cases. Moreover, *Gecos* provides also a CLI. In this case the substitution matrices are given as text file in *NCBI* format. By default, the CLI runs 16 optimizations in parallel and outputs the created color scheme in *JSON* format.

3.3 Results and discussion

3.3.1 Optimization

As explained above, the simulated annealing procedure depends on the α and β parameters at start and end of the optimization. These parameters were sampled to find a parameter set that gives optimal results in the benchmark. The samples and associated scores are shown in Figure 3.4. The results indicate a broad region for each parameter, where good results can be achieved. The sample with the best results is listed in Table 3.1. These values were rounded and used henceforth as default arguments for the optimizer.

[16] Akiba et al. (2019).

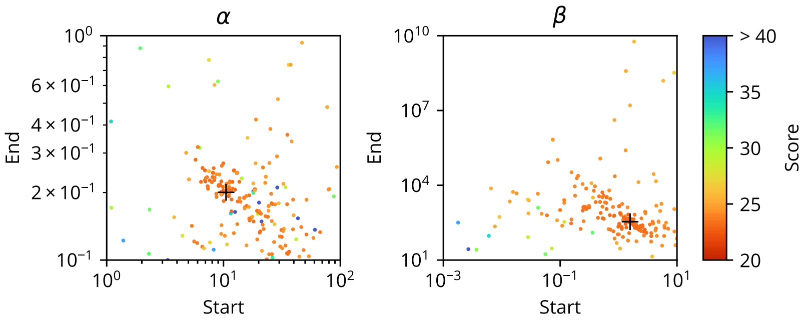


Figure 3.4: Optimization of α and β range for simulated annealing. The figure shows the sampled parameters during meta-optimization and the associated score of the color conformation achieved after optimization using these parameters. The sample with the lowest score is indicated with a '+'. The density of the points is higher in areas of lower score due to the working principle of the sampler. Note that α and β are not sampled independently, i.e. each point in one subplot corresponds to a point in the respective other subplot.

Table 3.1: Optimal parameters for simulated annealing. The derived default arguments for the optimizer are shown in parentheses.

	α	β
Start	10.5 (10)	1.57 (1)
End	0.20 (0.2)	345 (500)

Based on the determined parameters the effect of the number of optimization steps and the applicability of these parameters on different substitution matrices was tested (Figure 3.5). The optimization algorithm achieves a clear decrease in score for the popular *BLOSUM62* and *PAM250*. The fact that the performance for the identity matrix seems inferior can be attributed to the fact, that this matrix has equal optimal distances for each pair of amino acids which cannot be properly depicted in a three-dimensional color space. Increasing the number of steps beyond $\sim$ 20000 has no considerable effect on the score. Therefore this value is used as the default number of steps in *Gecos*. Using this number. With this number of steps the generation of a new color scheme typically takes less than two minutes.

The course of an optimization with the found parameters is illustrated for the *BLOSUM62* matrix in Figure 3.6. Although no chemical information is directly used by the algorithm, some chemical properties are implicitly included in the substitution matrix. For example the small nonpolar amino acids methionine, valine, leucine and isoleucine have similar assigned colors. However, the scheme also depicts similarities that may be counterintuitive: Alanine is for instance closer to the polar amino acids serine and threonine than to the other small nonpolar amino acids.

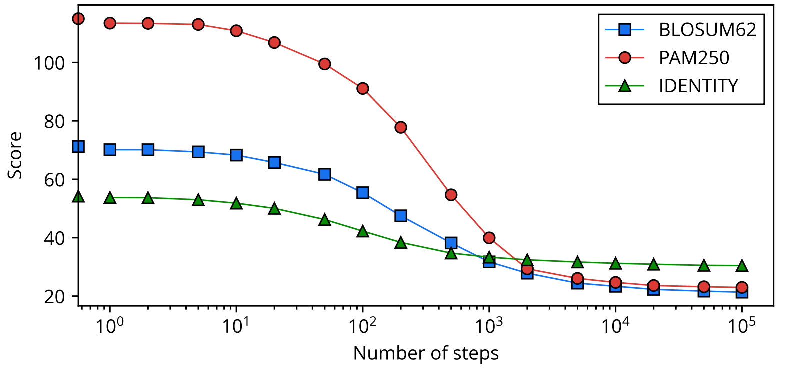


Figure 3.5: Effect of optimization step number on score of color conformation. An optimization of the color conformation was performed using default α and β for *BLOSUM62*, *PAM250* and the identity substitution matrix. For each number of steps 100 optimizations were run and the median of the final scores were taken. The marker on the y-axis gives the median score for the initial unoptimized color conformation.

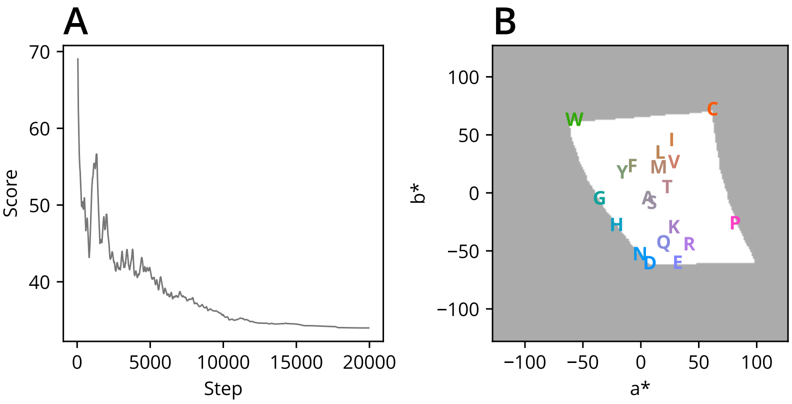


Figure 3.6: Insight into the optimization process. For the depicted optimization of the color conformation the *BLOSUM62* matrix was chosen. The color spaced was limited to $L^* = 60$ for the sake of presentability. **A** Course of the score during the optimization process. For clarity a moving average over 100 steps was applied. **B** The color conformation after optimization. The white area represents the allowed space at $L^* = 60$. The symbols are colored according to their position in color space. Adapted from Kunzmann, Mayer and Hamacher (2020) (CC BY 4.0).

3.3.2 Use cases

The presented algorithm has multiple potential use cases. In the most basic case it allows automatic generation of color schemes that may be simply more pleasant than existing ones due to individual flavor. Furthermore, the color schemes are able to depict the particularities of different amino acids. Figure 3.7A and 3.7B show an alignment visualized with a scheme generated for *BLOSUM62* and *PAM250*, respectively. Compared to *BLOSUM62*, *PAM250* has considerably lower substitution probabilities for cysteine and tryptophane to other amino acids. This circumstance is reflected by the *PAM250*-based color scheme (Figure 3.7B) that emphasizes the color of cysteine and tryptophane. Therefore, it is beneficial to visualize an alignment with a color scheme based on the corresponding substitution matrix, to give the observer a better understanding of the alignment.

The ability to constrain the color space allows the user to create a color scheme with an individual lightness or hue. This option can be especially useful to create a color scheme that is more friendly to people with color vision deficiencies. The most frequent deficiency is deuteranopia[17] that affects $\sim$0.5 % of female and $\sim$8 % of male humans[18]. By removing the green part of the color space (Figure 3.7C) alignments can be made visually better accessible for affected people in comparison to existing color schemes.

Eventually, schemes can be created for more exotic alphabets, where no color scheme exists, yet, provided there is a substitution matrix. This includes structural alphabets, such as *protein blocks* (PB)[19], which encodes three-dimensional protein structures into a sequence based on backbone dihedral angles. Figure 3.7D shows an alignment of PB sequences for lysozyme structures from different organisms. In this case, this allows not only the visual assessment of structural similarities within an alignment column, but also the identification of recurring structural motifs [20].

[17] disability in the red-green color differentiation

[18] Al-Aqtum and Al-Qawasmeh (2001); Kovalev (2004); NIH (2019).

[19] de Brevern, Etchebest, and Hazout (2000); Barnoud et al. (2017).

[20] e.g. the recurring ‘`fklm`’ sequence

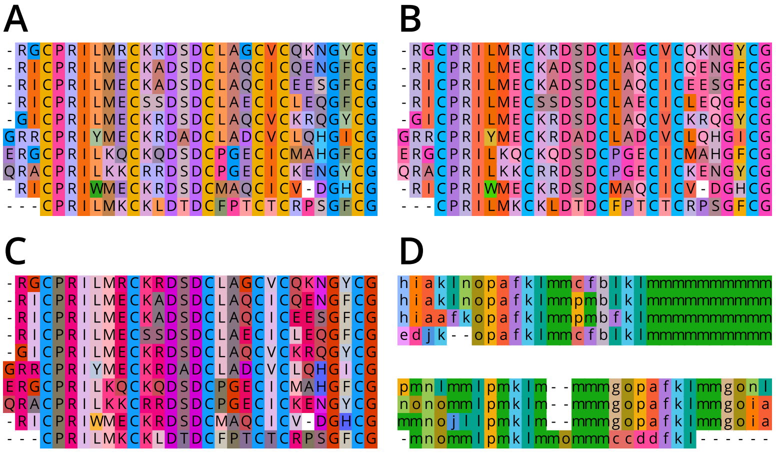


Figure 3.7: Different applications of *Gecos*. The displayed alignments are visualized using color schemes generated by *Gecos*. If not stated otherwise, the schemes were produced using default parameters based on *BLOSUM62* and a L^* range between 60 and 75. **A** Color scheme using all parameters described above. **B** Color scheme based on PAM250. **C** Color scheme adapted for red-green color vision deficiency. This is achieved by removing the green portion of the color space, i.e. where $a^* < 0$. **D** Color scheme based on a substitution matrix for the PB structural alphabet. The PB sequences are computed from structures of different lysozyme variants. Adapted from Kunzmann, Mayer and Hamacher (2020) (CC BY 4.0).

Companion source code:
`pfasum_scheme.py`

3.3.3 Application example

For the purpose of an example a color scheme was generated for the *PFASUM60* amino acid substitution matrix: The *PFASUM* series of matrices[21] is analogous to the *BLOSUM* matrices, but is computed from a larger dataset based on structural alignments. In addition to the proteinogenic amino acids, the matrix includes substitution scores for symbols indicating amino acid ambiguity. Since these symbols do not represent actual amino acids, they should not influence the color conformation of the symbols representing real amino acids. Hence, the color scheme generation was separated into two steps. First, a scheme was generated using default parameters for a truncated substitution matrix, that contains only scores for substitutions for unambiguous amino acids. Second, another color scheme was generated for the complete substitution matrix including ambiguity symbols, where the unambiguous amino acids were constrained to the colors of the previously generated scheme. The resulting scheme is shown in Figure 3.8A. Furthermore, a dendrogram was created based on a tree computed using the UPGMA hierarchical clustering method (see Section 5.2.2) (Figure 3.8B). The pairwise distances between the symbols were calculated from the substitution matrix as described above. As expected, the colors are generally more similar the closer two symbols are located in the tree.

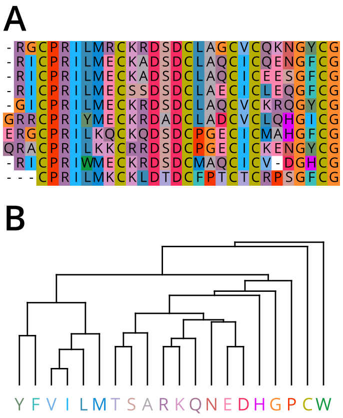


Figure 3.8: Color scheme for *PFASUM60*. **A** Alignment visualization using the color scheme. **B** Dendrogram of *PFASUM60*. The symbols are colored according to the generated scheme.

3.4 Conclusion

The presented method offers a mechanism to generate alignment color schemes for a variety of purposes that are not covered by conventional schemes, including different hues, evolutionary models and even underlying alphabets. While the most common application of such color schemes is alignment visualization, their use case can be extended to related figure types such as sequence logos.

[21] Keul et al. (2017).

Although substitution matrices are used as foundation for color scheme generation, the resulting colors implicitly also depict chemical properties of amino acids, like it is intended by conventional color schemes: Only amino acids with similar chemical characteristics have a high substitution probability, as substitutions by an amino acid with significantly different properties often lead to a loss of protein function and thus vanish in the evolutionary selection process. However, the presented method substantiates chemical properties with evolutionary similarity in terms of substitution probabilities.

As a final remark it needs to be noted that colors are neither able to perfectly depict evolutionary distances or a comprehensive set of chemical properties in most cases: It is not possible to map for example the 20 × 19 different amino acid substitution probabilities to the mere three dimensions of a color space. Nevertheless, the optimization method described in this chapter aims to provide a mapping that is as good as possible.

Chapter 4
Molecular visualization

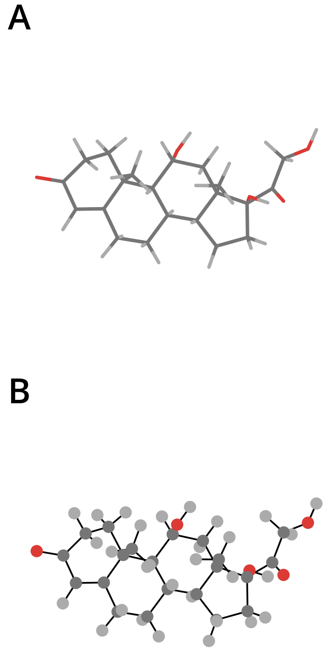


Figure 4.1: Molecular sketches of cortisol. Carbon atoms are shown in dark gray, oxygen atoms in red and hydrogen atoms in light gray. **A** Line model of the molecule. **B** Ball-and-stick model of the molecule.

4.1 Introduction

Images of molecules are commonly one of the final products of studies in structural biology as well as structural bioinformatics. These pictures are able to give impressions about the shape of macromolecules and their complexes and allow visual analysis of details such as the potential binding and reaction mechanism with a substrate. This chapter presents convenient ways to visualize structural models in *Biotite*, i.e. an `AtomArray` or `AtomArrayStack`.

4.2 Implementation

4.2.1 Visualization using Matplotlib

The *Python* scientific computing ecosystem already provides mature libraries for data visualization. The arguably most popular plotting tool in the *Python* community is *Matplotlib*[1]. Its versatility is harnessed by *Biotite* to create simple molecular sketches. Based on the coordinates and the `BondList` of an *AtomArray*, connected atoms can be either depicted as lines or as ball-and-stick model in interactive 3D plots (Figure 4.1). Recent improvements on the 3D plotting functionality of *Matplotlib* further increase the quality of the molecular visualizations by ensuring the correct render order, i.e. objects that are in the front occlude objects that are in the background.

This functionality allows simple and fast sketching of molecules for testing purposes. *Matplotlib* is usually already part of the computational scientist's repertoire, so no additional dependency is required. However, publication-quality visualizations cannot be produced this way: The plots do not support lighting, which makes visualizations for large molecular systems incomprehensible. Furthermore, common elements like secondary structures or molecule surfaces cannot be visualized.

4.2.2 Visualization using PyMOL

PyMOL[2] is a popular software suite dedicated for creating molecular visualizations. While in the base version its source code is freely available, the binary of the software is also distributed as commercial version. The *Biotite* extension package *Ammolite* utilizes its *Python* API to transfer an `AtomArray` or `AtomArrayStack` to the *PyMOL* workspace without the need of intermediate files. In addition to saving

[1] Hunter (2007).

[2] Schrödinger (2017).

the time for reading and writing a structure file, this approach ensures that no information is lost in the transfer process[3]. For applying commands such as coloration to a subset of atoms, *PyMOL* normally uses a custom string-based atom selection syntax. To provide an experience consistent with the *NumPy*-based selection syntax of *Biotite*, *Ammolite* allows calling these commands using *NumPy* indices. Internally, the *NumPy*-based selection is converted into a *PyMOL* selection string before the command is executed. Finally, *Ammolite* wraps the usage of *PyMOL*'s compiled graphics objects (CGOs) to enable the user to draw custom shapes in the molecular visualization, including spheres, cylinders and cones. This allows for example the addition of three-dimensional arrows to depict atom displacements.

To simplify the installation process a compiled *PyMOL* binary, based on the open-source variant is provided via the *Conda* package manager. This package is automatically downloaded, when *Ammolite* is installed via *Conda*.

4.3 Application example

Companion source code: `hcn4_pore.py`

Although all molecular visualizations in this thesis are created with *Ammolite*, this section presents the visualization of the tetrameric HCN4 potassium channel as a dedicated example. More specifically, the transmembrane domain including the pore diameter should be displayed. For this purpose the structural model of the presumably open conformation elucidated via electron microscopy[4] was fetched from the PDB[5,6] and loaded as `AtomArray`. The structure was filtered to the helices S4-S6 which make up the central part of the transmembrane domain.

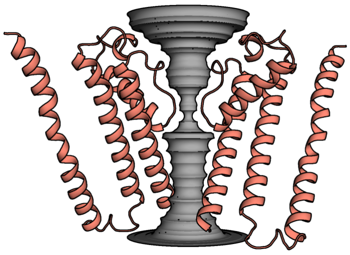


Figure 4.2: Pore radii of the HCN4 channel visualized using *Ammolite*. The S4-S6 helices are shown in red. The pore diameter along the pore is shown in gray.

The diameter of the pore is not constant, but varies along the pore axis. Therefore, the diameter was measured at multiple positions along the axis. At each position a plane perpendicular to the pore axis is spanned, whereas the atoms of the protein are represented by spheres with radius equal to the Van der Waals radius. Since the structure contains no hydrogen atoms, coarse grained radii[7] are used here. The pore radius is the minimum distance to a point where the plane intersects any of these spheres.

For the visualization (Figure 4.2) only two opposing chains of the tetramer are shown. The diameters at neighboring height positions are shown using CGOs encoding a cone shape. Concatenating these cones creates the appearance of a continuous shape depicting the pore.

[3] For example *mmCIF* files do not save bond information.

[4] Saponaro et al. (2021).

[5] PDB: 7NP3

[6] Saponaro et al. (2021).

[7] Tsai et al. (1999).

Chapter 5
Guide trees and multiple sequence alignments

5.1 Introduction

Multiple sequence alignments (MSAs) are an invaluable tool in bioinformatics: From identification of conserved protein or nucleotide regions over reconstruction of phylogeny[1] to *de novo* prediction of protein structures[2], MSAs constitute the data foundation. Although the optimal[3] alignment of k sequences can be solved using dynamic programming[4] the time complexity scales with $\mathcal{O}(\prod_{i=1}^{k} n_i)$, where n gives the length of the respective sequence. This procedure quickly becomes unfeasible for even a few sequences.

5.1.1 Progressive Alignment

Hence a technique called ***progressive alignment***[5] is commonly employed: It begins with two sequences A and B that are aligned pairwise into a *sub-MSA* C. Then, another sequence or sub-MSA D is added to the MSA: C is aligned pairwise with D, treating previously introduced gaps in C[6] as neutral symbols, i.e. they score zero with each other symbol. In consequence, gap positions are not revaluated: Once a gap is introduced in a sub-MSA, it will persist in higher level sub-MSAs, even if the similarity score could be improved by gap rearrangement[7]. Thereafter, further sub-MSAs are progressively aggregated into the alignment using this principle until all sequences are added, resulting in the final MSA. How a pairwise alignment of two sub-MSAs is scored in detail, varies in the different implementations of the progressive alignment method. Commonly, the following scheme is used[8]: Given that a and b are alignment columns of two sub-MSAs, containing m and n sequences, respectively, their score

$$S(a,b) = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} S(a_i, b_j). \tag{5.1}$$

In prose, $S(a,b)$ is the mean of the cartesian product of all pairwise symbol scores between the sub-MSA columns.

5.1.2 Guide trees

The order in which the sequences or sub-MSAs are aligned is determined by a binary tree[9], the ***guide tree***. The leaf nodes in the tree represent single sequences, intermediate nodes sub-MSAs and the root the full MSA. Beginning from the leaf nodes, the progressive alignment follows the branches of the tree until the root is reached (Figure 5.1). Based

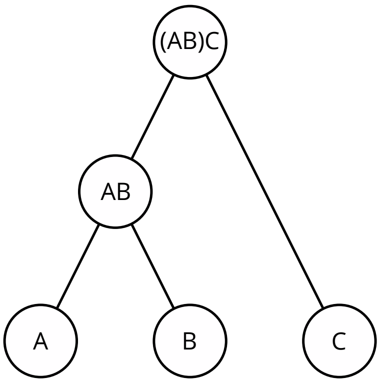


Figure 5.1: Example guide tree. The figure shows a simple guide tree: First sequences A and B would be pairwise aligned and then C would be aligned to the sub-MSA AB.

[1] Kapli, Yang, and Telford (2020).

[2] Jumper et al. (2021).

[3] according to a given substitution matrix and gap penalty

[4] Needleman and Wunsch (1970).

[5] Feng and Doolittle (1987).

[6] and also D, if D is a sub-MSA

[7] This principle is often titled '*Once a gap, always a gap*'.

[8] Thompson, Higgins, and Gibson (1994).

[9] a noncyclic graph, where each non-leaf node has exactly two child nodes

on a distance measure for each pair of sequences, the guide tree can be calculated using hierarchical clustering. Two common algorithms used for this task are the *unweighted pair group method with arithmetic mean* (UPGMA) and *neighbor-joining*, which should be summarized in the following paragraphs.

Let $d_{i,j}$ denote the distance between sequences i and j. Starting from a list of leaf nodes to be connected, UPGMA combines the two nodes with the minimum $d_{i,j}$, say A and B, into a new node AB. A and B are removed from the list and AB is added to the list as new node. The distances of AB to another node X in the list is

$$d_{AB,X} = \frac{|A| d_{A,X} + |B| d_{B,X}}{|A| + |B|}, \tag{5.2}$$

where $|A|$ and $|B|$ is the total number of leaf nodes that are combined in A and B, respectively[10]. This procedure is repeated until only a single node remains in the list, which becomes the root node. UPGMA computes the branch lengths, so that all leaf nodes have the same distance to the root. Hence, this method implicitly assumes a *molecular clock*, meaning that mutation rate would be constant over all species. However, more distantly related species show variations in the mutation rate[11].

Neighbor-joining[12] overcomes the molecular clock assumption. Instead of combining nodes with minimum $d_{i,j}$, those nodes are joined, where the resulting tree has a minimum total branch length, i.e. where $S_{i,j}$ is at minimum:

$$S_{i,j} = (N-2) d_{i,j} - \sum_{k=1}^{N} d_{i,k} - \sum_{k=1}^{N} d_{j,k}, \tag{5.3}$$

where N gives the number of remaining nodes to be joined. The distance of the new node to other nodes is

$$d_{AB,X} = \frac{1}{2} \left(d_{A,X} + d_{B,X} - d_{A,B} \right). \tag{5.4}$$

Neighbor-joining ensures that the traversed distance between two leaf nodes in the tree is equal to the corresponding $d_{i,j}$ value. In contrast to UPGMA, there is no dedicated root node, any position on a branch could represent the common ancestor.

Although modern phylogeny reconstruction is superseded by more accurate methods[13], neighbor-joining can still be used to create phylogenetic trees for well-conserved sequences, given a proper distance measure.

5.2 Implementation

Today numerous programs are available that use different variations of the guide tree creation and progressive alignment algorithm to improve on the computational performance of MSAs. However, often

[10] in other words: 1 if a single sequence, > 1 if a sub-MSA

[11] Bromham and Penny (2003).

[12] Saitou and Nei (1987); Studier and Keppler (1988).

[13] Kapli, Yang, and Telford (2020).

the flexibility of such software is limited and their availability is constrained to *Unix*-based operating systems. *Biotite* allows users to perform simple[14] MSAs on all of its supported platforms and for all types of sequences (see Section 2.2.1). For this purpose it implements both presented hierarchical clustering algorithms to create guide trees and functionality to use guide trees for progressive alignment.

5.2.1 Tree representation

A tree, such as a guide tree or phylogenetic tree, is represented by `Tree` objects in *Biotite*. A `Tree` contains a root `TreeNode`, that defines the tree recursively: Each `TreeNode` has a parent tree `TreeNode` and a distance to it, unless it is the root, and may have a number of child `TreeNode` objects. If a `TreeNode` has no children, it is a leaf node. Leaf nodes must have a reference index (an integer), that points to elements in a list of corresponding objects. For example, if a `Tree` represents a guide tree, such a list would contain `Sequence` objects. If the leaf `TreeNode` objects with reference index 0 and 1 have a common parent `TreeNode`, the first and second sequence[15] from the list should be pairwise aligned. Still, a `Tree` might refer to any list of objects, enabling a broad applicability for the user. The usage of a reference index allows unambiguous mapping of a leaf node to an object, while keeping the `Tree` separate from these objects.

In addition to basic traverse through the tree structure, a `Tree` provides functionalities for identification of the lowest common ancestor and distance between two nodes. Furthermore, a `Tree` can be read from and written to the *Newick* format enabling interoperability with other software such as MSA or phylogeny tools[16].

5.2.2 Hierarchical clustering

With the `upgma()` and `neighbor_joining()` functions objects can be hierarchically clustered using the aforementioned algorithms. Both functions require a symmetric pairwise distance matrix. They return a `Tree` object, where the reference indices of the n leaf nodes correspond to the $n \times n$ shape of the distance matrix.

5.2.3 Tree display

Using *Matplotlib*, `Tree` objects can be displayed as dendrogram, with `plot_dendrogram()` (Figure 5.2A). A `Tree` can also be converted into a directed graph of the *NetworkX Python* library[17]. As *NetworkX* provides graph plotting capabilities on its own, this permits the depiction of unrooted trees (Figure 5.2B).

[14] The MSAs from dedicated software are more accurate for less conserved sequences and faster for a large number of sequences.

[15] 0-based indexing

[16] Baum (1989); Edgar (2004); Katoh et al. (2002); Sievers et al. (2011).

[17] Hagberg, Schult, and Swart (2008).

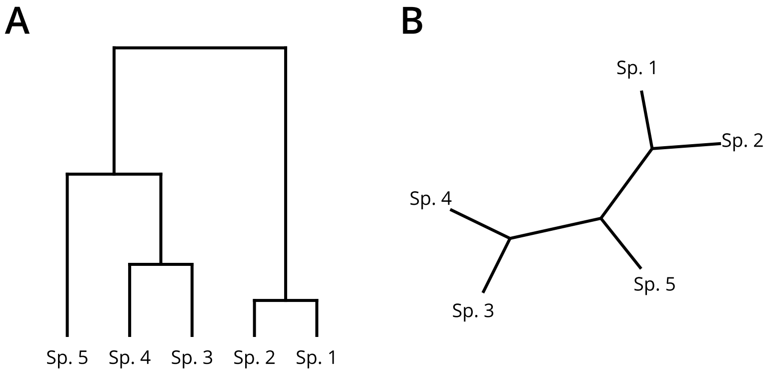


Figure 5.2: Tree visualizations. The underlying distance matrix is fictive. **A** Dendrogram created with `plot_dendrogram()`. The tree was clustered with UPGMA. **B** Unrooted tree visualized with *NetworkX*. The tree was clustered with neighbor joining.

5.2.4 Multiple sequence alignments

The function `align_multiple()` uses progressive alignment to create MSAs from a list of `Sequence` objects. By default, the pairwise distances are inferred from pairwise global alignments (see Section 2.4.2) of all sequence pairs using the same scoring scheme as given for the later multiple alignment. The distance $D(i,j)$ is computed from the similarity score $S(i,j)$ between the two sequences i and j as[18]

$$D(i,j) = \frac{S(i,j) - S_r(i,j)}{\frac{S(i,i)+S(j,j)}{2} - S_r(i,j)}. \tag{5.5}$$

S_r gives the background noise for two aligned random sequences with same length and symbol composition, estimated with

$$S_r(i,j) = \frac{1}{L(i,j)} \sum_a \sum_b M_{ab} N_a(i) N_b(j) - N_g g. \tag{5.6}$$

$L(i,j)$ is the length of the alignment, M_{ab} the similarity score of the symbols a and b from the substitution matrix, g is the gap penalty, N_a, N_b and N_g are the number of symbols a and b and gaps, respectively. Based on these distances the guide tree is computed with UPGMA. Alternatively, a custom distance matrix or a guide tree can be provided by the user.

The progressive alignment proceeds according to the order given by the guide tree. To align two sub-MSAs (or a sub-MSAs and a single sequence), their two sequences with the lowest distance are aligned. The gap positions introduced in this alignment are adopted by the sub-MSAs and converted into a neutral symbol. This way the already existing pairwise alignment functionality from *Biotite* can be fully reused. To implement the neutral symbol a new `Alphabet`, that contains a unique neutral symbol, is created from the original `Alphabet` of the sequences to be aligned. In addition a new `SubstitutionMatrix` is instantiated using the modified `Alphabet` that contains scores of 0 in the row and column of the neutral symbol. After the full MSA is computed, the neutral symbols in the alignment are replaced by actual gaps.

[18] Feng and Doolittle (1996).

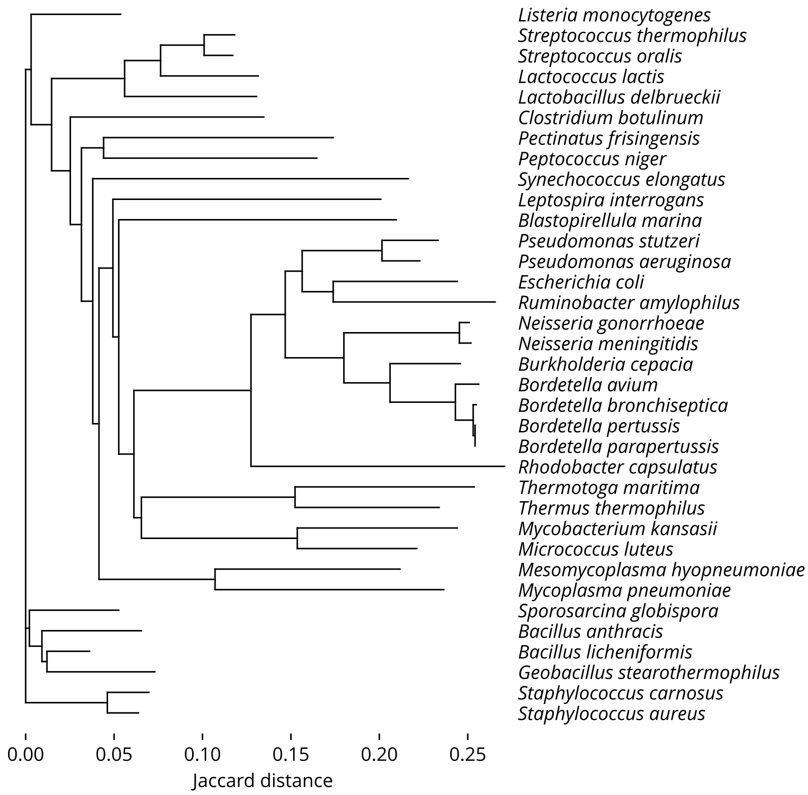


Figure 5.3: Phylogenetic tree of selected bacterial species. Horizontal branch lengths depict the Jaccard distance, i.e. the percentage of mutated positions between the 23s rRNA sequences. Clustering was performed using neighbor joining.

5.3 Application example

Companion source code: `bacterial_phylogeny.py`

16s and 23s ribosomal RNA (rRNA) sequences are popular markers for building phylogenetic trees due to their large size and high conservation[19]. Here, the phylogeny of a selection of different bacterial species was inferred from 23s rRNA sequences. The sequences were downloaded from the *NCBI Entrez* database. The MSA was created with `align_multiple()` using nucleotide similarity scores and gap penalty taken from *MMseqs2*[20]. The MSA served as foundation for phylogeny reconstruction: For each pair of sequences in the MSA the *Jaccard distance*[21] was calculated, or more precisely the percentage of positions in the shorter sequence, that were substituted or deleted. The distances were input to `neighbor_joining()` to obtain the phylogenetic tree. Finally, the tree was graphically displayed using `plot_dendrogram()` (Figure 5.3).

[19] Ludwig and Schleifer (1994).

[20] Steinegger and Söding (2017).

[21] Wolf et al. (2002).

Chapter 6 Heuristic alignment searches

This chapter builds upon ideas and figures from a submitted journal article[1]. The content will be licensed under the CC BY 4.0 license. Changes were made on size and arrangement of figure elements. The application example of this chapter was adapted from the example gallery at the *Biotite* documentation website (Table A.1).

6.1 Introduction

The progressive alignment algorithm illustrated in the previous chapter introduces a heuristic variation of the original dynamic programming algorithm to solve MSAs in reasonable time. Analogously, other heuristics can be introduced to massively increase the performance for pairwise alignments of large sequences such as genomes or huge sequence datasets. This type of method has obtained different names from the community, here it will be termed *alignment search*. Originally such a method was implemented in the *BLAST* software[2], which is still popular to date. Since then numerous variations have been published, with *MMseqs2*[3] being a modern representative.

The underlying concepts are usually quite similar: The alignment search is separated into multiple consecutive stages. In the initial stage similar regions between the two given sequences, so called *hits*, are identified very roughly. Each following stage narrows down the hits from the respective previous stage using a more sensitive[4] sequence comparison. Hits that do not meet a certain threshold similarity are filtered out. Although the increasing accuracy with each stage comes at the cost of longer computations for each hit, the overall time is kept moderate, as the number of considered hits decreases with each stage. Since at each each of the stages an heuristic method is used to filter sequences, some homologous regions might potentially be missed in this process. In the following sections the typical stages are elaborated in more detail.

6.1.1 Stage 0: Removing low complexity regions

Low-complexity regions, such as tandem repeats[5], make up a large part of genomic sequences. Such repeats are thought to arise primarily from replication slippage and gene conversion[6]. Aligning genomes containing such regions lead to hits that represent spurious homologies. Therefore, *alignment search* methods generally filter out these regions prior to the first stage. Dedicated programs for this purpose include *DUST* and *tantan*[7].

[1] **Kunzmann** et al. (2022).

[2] Altschul et al. (1990).

[3] Steinegger and Söding (2017).

[4] also weaker sequence similarities can be detected

[5] directly adjacent sequence repetitions, eg. `catcatcatcat...`

[6] Richard, Kerrest, and Dujon (2008).

[7] Morgulis et al. (2006); Frith (2011).

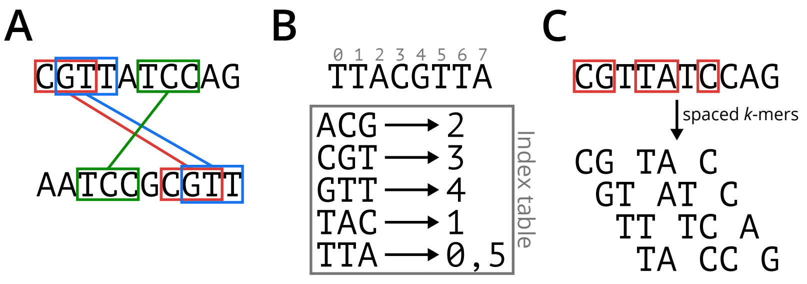


Figure 6.1: k-mer matching. A Matching identical 3-mers between two sequences. **B** Indexing 3-mer positions in a sequence. The positions are zero-based. **C** Identifying spaced 5-mers in a sequence. The spacing pattern is indicated by the red boxes.

6.1.2 Stage 1: K-mer matching

Initial hits between two sequences A and B are found by matching identical short subsequences of length k, so called *k-mers*, in both sequences (Figure 6.1A). In a naive approach all overlapping k-mers of A could be iterated and for each k-mer in A, the k-mers of B would be iterated to find matching positions. However, the computation time of this procedure would scale linearly with the length of A and B. To improve the performance the overlapping k-mers of A are indexed into a lookup table that maps a k-mer to the positions in A where this k-mer appears (Figure 6.1B)[8]. Hence, only one sequence needs to be iterated: For each position in B, the matching positions in A can be simply looked up in the table using the k-mer at the respective position as key.

Instead of matching only identical k-mers, some variability can be optionally allowed. If a substitution matrix S is available for the underlying alphabet of the sequences, a threshold score T can be used: A k-mer a matches a k-mer b if $\sum_i^k S_{a_i,b_i} \geq T$. This allows for example that k-mers with evolutionary similar amino acids can be matched with each other in protein sequences. Alternatively, a certain number of mismatches can be allowed if no substitution matrix is available. In both cases all similar k-mers are enumerated for a k-mer and matching positions are obtained by combining the looked up positions for each of these k-mers.

As alternative to contiguous subsequences as k-mers, *spaced k-mers*[9] have gained increasing popularity due to their higher sensitivity. Here, a k-mer omits symbols at constant positions relative to the start of the k-mer, as shown in Figure 6.1C. Which positions are skipped is defined by the spacing pattern. Several approaches to find patterns for different k that optimize sensitivity have been published[10]. The high sensitivity of spaced k-mers can be attributed to a higher probability to find at least one hit in a large homologous but not necessarily identical region[11].

6.1.3 Stage 2: Ungapped hit extension

In the following stage each of the obtained hits are extended in both directions of the respective hit without introducing gaps. Let A_i and

[8] Some programs index A and B, but this does not give an improvement in time complexity in comparison to indexing only A though.

[9] Ma, Tromp, and Li (2002).

[10] Ma, Tromp, and Li (2002); Choi, Zeng, and Zhang (2004); Hahn et al. (2016); Noé (2017).

[11] Zhang (2007).

B_j be the sequence positions of the hit, the so called *seed*. In the extension A_{i+1} is aligned with B_{j+1}, A_{i+2} with B_{j+2}, etc. Likewise, A_{i-1} is aligned with B_{j-1}, A_{i-2} with B_{j-2}, etc. In each direction the extension ends when the total alignment score falls a given value X below the maximum score found. Therefore, this criterion is called *X-drop*. The resulting alignment is truncated to the range, that gave the maximum score, i.e. the range that decreased the score is removed from each end. Only those alignments that exceed a certain threshold score are passed on to the next stage.

The concept behind the hit extension is that it is relatively fast, since no dynamic programming table needs to be computed, while it is still more sensitive than k-mer matching, because longer segments can be aligned, even if some mismatches appear in the aligned segments. Often a so called *double-match* strategy is employed[12]: Not a single k-mer match is sufficient to trigger hit extension, but two matches on the same diagonal[13] within a short distance are required. This reduces the number of hit extensions and hence also the computation time.

6.1.4 Stage 3: Gapped alignments

Due to insertions and deletions the introduction of gaps into an alignment is usually required to detect long homologous sequence segments such as entire genes. In the context of alignment searches the original approach for optimal alignments[14] is not applicable: The algorithm searches for the optimal alignment using a dynamic programming table spanning the length of both aligned sequences (see Section 2.4.2) as shown in Figure 6.2A. This would defeat the purpose of the hit-based approach so far.

Hence, different strategies must be employed to decrease the explored area[15] of the dynamic programming table using the filtered hits from the previous ungapped alignment stage. The reduction is based on the assumption that the trace of the optimal local alignment does not leave the explored area. If this assumption does not hold true, the optimal alignment cannot be found by the algorithm and homologous sequences may be missed in consequence.

Similar to ungapped alignments, the X-drop strategy can also be adapted to gapped local alignments[16]. It is assumed that the score of an optimal alignment does not intermediately drop X below the best alignment already seen and recovers after that. Consequently, the dynamic programming table is not explored beyond the area where the score falls X below the maximum score seen so far (Figure 6.2B). Rather than iterating over rows and columns of the dynamic programming table, the iteration is performed over antidiagonals[17] and diagonals.

The *banded alignment* strategy[18] makes the assumption, that it is improbable that in any point in the optimal alignment W more gaps are in-

[12] Altschul et al. (1997).

[13] The diagonal refers to the dynamic programming table, i.e. for two sequence positions i and j the diagonal can be defined as $j - i$.

[14] Needleman and Wunsch (1970); Smith and Waterman (1981).

[15] The explored area comprises the cells of the table that are actually computed. The computation time roughly scales linearly with the number of explored cells.

[16] Altschul et al. (1997); Zhang et al. (2000).

[17] Antidiagonals are perpendicular to the diagonals. Analogous to the diagonals, they can be defined as $j + i$.

[18] Chao, Pearson, and Miller (1992).

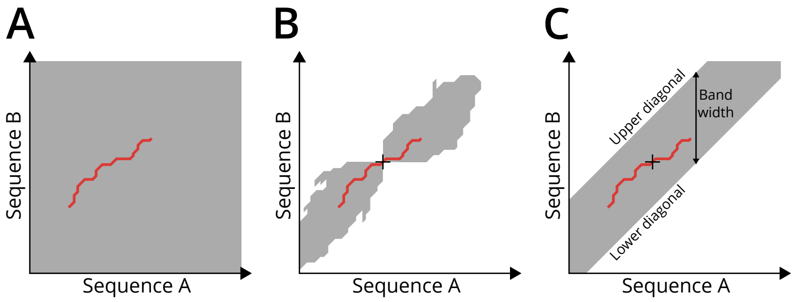


Figure 6.2: Comparison of algorithms to find gapped local sequence alignments. Each plot depicts a schematic dynamic programming table, with the axes indicating the sequence positions. The gray area shows the portion of the table explored by the respective algorithm. The '+' shows the seed position. The red line shows the trace of the best alignment found. **A** Original dynamic programming algorithm. **B** *X*-drop alignment algorithm. **C** Banded alignment algorithm. Adapted from Kunzmann *et al.* (2022) (CC BY 4.0).

serted in sequence A than in sequence B. Therefore the explored area of the table is reduced to a diagonal band (Figure 6.2C). The center diagonal of the band is the respective hit position. The band width is a parameter of the search method and can be, for example, determined using statistical considerations[19]. An advantage of the banded alignment is that it can also be used to perform global alignments.

6.1.5 Significance evaluation

In the process explained above the large number of initial hits from the k-mer matching stage are reduced to a relatively low number of gapped alignments. Although each alignment has an associated similarity score, this absolute value does directly measure the significance of the alignment - the probability to obtain such sequence similarity by mere chance. *BLAST* introduced the notion of the *expect value* (E-value). Let s be the score of an alignment of interest. The E-value gives the number of alignments with a score equal to or higher than s, that would be expected from alignment of randomized sequences with equal length and composition. An E-value close to 1 or even higher indicates that the alignment of interest is not significant.

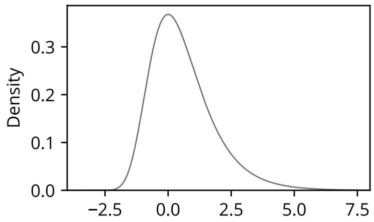


Figure 6.3: Extreme value distribution. The figure shows the probability density function of the distribution for $\lambda = 1$ and $u = 1$.

For a gapped local alignment of two random sequences with length m and n, respectively, the score s is approximated by an extreme value distribution (Figure 6.3), if m and n are sufficiently large[20]. Its probability density function

$$\begin{aligned} f(s) &= \lambda t(s) e^{-t(s)}, \\ t(s) &= e^{-\lambda(s-u)} \end{aligned} \tag{6.1}$$

is governed by the parameters λ and u which need to be determined from a number of sample alignments for a given combination of sequence length, symbol composition, substitution matrix and gap penalty. The dependency on sequence length suggests that resampling is necessary for each alignment search, since the length of sequence B may vary, even if A is part of a constant sequence database. This task would be time consuming and detrimental for the performance of alignment searches.

Luckily, u scales logarithmically with the length of both sequences.

[19] Gibrat (2018).

[20] Altschul and Gish (1996).

Hence, it can be substituted with

$$u = \frac{\ln(Kmn)}{\lambda}. \tag{6.2}$$

λ is approximately constant for sufficiently large subsequences[21] (see Section 6.3.1). Therefore, the obtained λ and K from a sampling process for a single set of sequence lengths can be generalized to evaluate alignments using different sequence lengths.

Using 6.2 and the cumulative distribution function of Equation 6.1, the probability of finding an alignment with score equal to or larger than s using random sequences is

$$P(S \geq s) = 1 - \exp(-Kmne^{-\lambda s}). \tag{6.3}$$

The corresponding E-value is calculated as[22]

$$E = Kmne^{-\lambda s}. \tag{6.4}$$

6.2 Implementation

Since the different alignment search methods mostly differ in nuances, such as the replacement of one of the presented stages, *Biotite* implements alignment searches as modular toolkit: For each stage, *Biotite* provides functionalities that covers common methods for that stage. These functionalities can be freely combined and parametrized to obtain a custom alignment search method for the problem at hand. Since the functionalities presented in this section cannot be computed in fully vectorized manner, the components of the alignment search toolkit are implemented in *Cython*[23] to obtain a substantial performance increase compared to a pure *Python* implementation. This enables the usage of this toolkit for alignment searches in feasible computation time in the first place.

6.2.1 K-mer encoding and matching

In *Biotite* the list of consecutive k-mers of a sequence is considered itself a sequence, whose symbols are governed by a `KmerAlphabet`. A `KmerAlphabet` requires a base `Alphabet`[24], the length k and an optional spacing pattern. The number of symbols in this alphabet is q^k for a base `Alphabet` with q symbols. Just like a regular `Alphabet`, a `KmerAlphabet` encodes symbols into a symbol code and vice versa (see Section 2.2.1), with the difference that the symbols are k-mers in this case. Let c be the sequence of symbol codes representing a k-mer in the base `Alphabet`[25]. The corresponding symbol code in the `KmerAlphabet` (k-mer code in short) is $\sum_{i=0}^{k-1} q^i c_i$. A central functionality of the `KmerAlphabet` is its ability to translate a sequence in the base alphabet, e.g. a protein sequence, into a sequence of k-mer codes.

Based on this k-mer code sequence, or multiple ones, a `KmerTable` is created, inspired by the *MMseqs2* design. The k-mers are indexed in two

[21] approximately larger than 100

[22] Altschul (1991).

[23] Behnel et al. (2011).

[24] e.g. the 20 proteinogenic amino acids

[25] In consequence the length of c is k.

passes. In the first pass the number of occurrences of each k-mer code in the sequence(s) is counted. Using these counts a position array that fits the number counts is created for each unique k-mer. Such an array contains *(sequence ID, position)*-tuples as elements. The *sequence ID* identifies in which sequence the k-mer appears and the *position* indicates where the k-mer is located in that sequence. The array of counts is converted into an array of pointers to these position arrays. Compared to allocating space for an entirely new array, this conversion can save a lot of memory if k is large, as the size of the pointer array is $8q^k$ bytes. Finally, finding the positions of a given k-mer is simply accomplished by accessing the pointer array with the corresponding k-mer code. Since each k-mer is unambiguously associated with a k-mer code, the `KmerTable` cannot encounter *hash collisions*[26].

Low-complexity regions can be omitted from indexing by giving the `KmerTable` a boolean mask containing ignored sequence positions. This mask can be created using the *Biotite* interface to *tantan* or can be provided by any custom source.

k-mer indexing is a time consuming process. Through support of *Python*'s standard serialization protocol *pickle*, a `KmerTable` for a sequence database can be saved to hard drive and reloaded as required. Thus, repetitive indexing of the same sequence database is not necessary. Furthermore, multiple `KmerTable` objects can be combined into a single `KmerTable`. This allows parallelized indexing of portions of the sequence database in multiple processes and the final assembly of this table into a complete table.

For k-mer matching, the k-mer code sequence of a query sequence is created. For each k-mer in the query sequence the matching positions are accessed as described above and the position in the query as well as the matching positions are appended to a list of match positions. To allow matching of similar instead of identical k-mers a `SimilarityRule` can be given for the matching process. A `SimilarityRule` is an object that enumerates all similar k-mers for a given k-mer. The matching is then performed for all of these similar k-mers. Which k-mers are actually considered '*similar*' is subject to the actual implementation of the `SimilarityRule`, allowing for a high degree of customization. The built-in `ScoreThresholdRule`, for example, outputs all k-mers that have at least a given threshold similarity score with the input k-mer, based on a `SubstitutionMatrix`. This is achieved using a *branch-and-bound* algorithm[27]. The `ScoreThresholdRule` class can also be used to allow a given number of mismatches in a k-mer, if an identity matrix is used as substitution matrix.

[26] Hash collisions appear, if different values translate into the same table key.

[27] Hauser, Mayer, and Söding (2013).

6.2.2 Gapped and ungapped alignments

The obtained match positions can be used as seed for a local ungapped or gapped alignment based on X-drop criterion with the functions `align_local_ungapped()` and `align_local_gapped()`, respectively. A challenge for `align_local_gapped()` in comparison to an optimal alignment is the initially unknown size of the explored area of the dynamic programming table. Hence, the table is initialized with a rather small size of 100×100 cells and its size in the respective dimension is doubled, when a margin of the table is reached.

Alternatively, banded local or global alignments can be performed using `align_banded()`. Instead of a seed it requires the lower and upper diagonal, that define the boundaries of the dynamic programing table. Here the challenge of a partially explored dynamic programming table is solved by indentation of the table memory layout to obtain a substantially smaller table size in most scenarios (Figure 6.4).

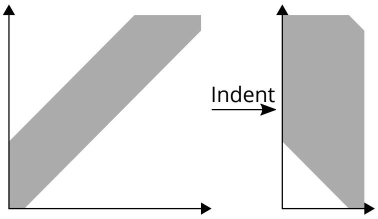


Figure 6.4: Indented memory layout for banded sequence alignments. The gray area shows the explored portion of the table. Adapted from Kunzmann *et al.* (2022) (CC BY 4.0).

6.2.3 E-value estimation

The E-value of an `Alignment` can be estimated using the `EValueEstimator` class. Instances of this class are specific for a combination of λ and K: The parameters can either be sampled, by supplying a scoring scheme and symbol frequencies, or given directly upon instantiation[28]. By default, the parameters are sampled using 1000 randomized sequence pairs, with each sequence having a length of 1000. For each pair the optimal local alignment is computed using the given scoring scheme (see Section 2.4.2). The parameters λ and u of the extreme value distribution are estimated using the method of moments[29], with

$$\begin{aligned} \lambda &= \frac{\pi}{\sqrt{6V}}, \\ u &= \mu - \frac{\gamma}{\lambda}. \end{aligned} \tag{6.5}$$

Here γ is Euler's constant and μ and V are arithmetic mean and variance of the sampled alignment scores. K is computed via Equation 6.2. To compute an E-value, the `EValueEstimator` then requires only the similarity score of the alignment of interest and the lengths of the aligned sequences.

6.3 Results and discussion

6.3.1 Parameter sampling

Using the `EValueEstimator` the generalizability of sampled λ and u parameters to arbitrary sequence lengths was tested. For this purpose multiple sequence lengths were evaluated. For each length 1000 pairwise alignments were created from two randomized sequences having that length. For scoring the *BLOSUM62* matrix with -12 gap opening and -1 gap extension penalty was used[30]. Amino acid background fre-

[28] This is useful, if the parameters are already known, either from tabulated values or previous sampling results.

[29] Altschul and Erickson (1986).

[30] Gap penalties were taken from the *MMseqs2* default values.

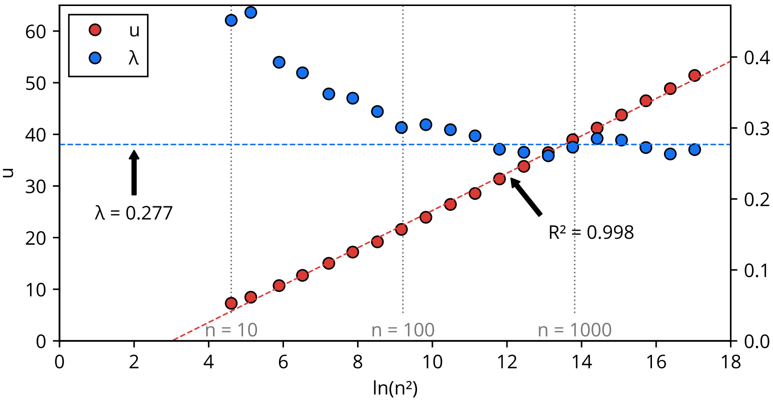


Figure 6.5: Sampled distribution parameters from different sequence lengths. The sampled λ and u parameters of the extreme value distribution is shown for different sequence lengths n. The dashed red line shows a linear regression of u. The dashed blue line depicts λ determined from this regression.

quencies were taken from Robinson *et al.*[31]. The results are shown in Figure 6.5.

As equation 6.2 can be rewritten as

$$u = \frac{1}{\lambda} \ln K + \frac{1}{\lambda} \ln n^2, \tag{6.6}$$

if $m = n$, a linear relation between $\ln n^2$ and u is expected. Hence, a linear regression was performed on u to test this expected behavior. Furthermore, λ can be read from the slope of the fitted function, to compare it with the points for individual sequence lengths. With $R^2 = 0.998$ it is clear that u from any sequence length can be used to determine u for other lengths as well. However, for sequence lengths $n \lessapprox 100$, λ is considerably higher than estimated from higher sequence lengths or the linear regression. Note, that the extreme value distribution only approximates accurately scores of random alignments for sufficiently large sequence lengths[32].

6.3.2 Computation time

The main advantage of the illustrated heuristics compared to the simple dynamic programming approach is the orders of magnitude faster identification of homologous sequences. The speed makes its application feasible for certain applications such as searches in entire genomes. Therefore, the computation time was evaluated in comparison to the original dynamic programming method and *MMseqs2*[33].

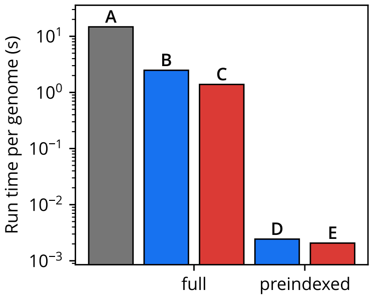


Figure 6.6: Run time of different alignment approaches. The run time of the alignment search per *Synechocystis* genome copy is shown. **A** Alignment using original dynamic programming method. **B** Alignment search using *Biotite* alignment search toolkit. **C** Alignment search using *MMseqs2*. **D+E** Equal to **B+C**, but time for k-mer indexing was not considered.

For the alignment search using the presented toolkit, the same stages as in Section 6.3.3 were used, with the exception that repeat masking was omitted[34]. In the benchmark scenario, the tRNA$^{\text{Ala}}$ sequence from *Escherichia coli* was aligned against 100 copies of the genome of the cyanobacterium *Synechocystis* sp. PCC 6803[35]. For comparison *MMseqs2* was executed on a single thread using default parameters, only the repeat masking was omitted and the k-mer spacing pattern was adapted. For the dynamic programming approach, a local optimal alignment of the tRNA$^{\text{Ala}}$ sequence to the genome was computed 10 times.

[31] Robinson and Robinson (1991).

[32] Altschul and Gish (1996).

[33] Steinegger and Söding (2017).

[34] running *tantan* would have biased the computation time

[35] *GenBank* IDs `CP001509` and `NC_000911`, respectively

The benchmark results are shown in Figure 6.6. The alignment search using *Biotite* (Figure 6.6B) is $\sim 6\times$ faster than pure dynamic programming (Figure 6.6A). However, in most scenarios a sequence database is only indexed once and thereafter reused for alignment with different query sequences. Hence, the k-mer indexing time is usually negligible. However, for a single run indexing takes the major part of the computation time: If the indexing step is not included in the run time, the alignment search is approximately three orders of magnitude faster (Figure 6.6D).

Biotite does not reach the performance of *MMseqs2* in this benchmark. This can be expected, as the program is heavily optimized: It uses an efficient implementation for double hit identification and is able to compute multiple gapped alignments in parallel using *single-instruction-multiple-data* (SIMD) instructions[36]. Furthermore, it uses multiple threads for k-mer indexing by default, which was not tested in this benchmark. In certain scenarios, this could have significant performance advantages compared to a multi-processed setup of *Biotite*. The run time gap could vary in other scenarios, since the run time of each stage is dependent on parameter setting, total sequence database size and sequence composition.

However, the alignment search implementation in *Biotite* still has a performance in a similar order of magnitude as *MMseqs2*, a representative for modern software dedicated for alignment searches. In applications where the computation speed is not a bottleneck, *Biotite* offers a flexible toolkit that can be relatively easily used to rapidly implement a solution for a problem where no existing software fits the purpose. In addition it allows rapid prototyping of new methods for alignment searches.

6.3.3 Application example

Companion source code:
`genome_comparison.py`

One of the arguably more famous relationships in biology is the one between chloroplasts and cyanobacteria: The chloroplasts of todays plants and algae are believed to originate from ancient endosymbiosis with cyanobacteria. Hence, chloroplasts and cyanobacteria share many homologous genes. To find these homologies, the genomes of the cyanobacterium *Synechocystis* sp. PCC 6803 and the *A. thaliana* chloroplasts were compared using the presented alignment search toolkit.

The sequences of both genomes were fetched from the *NCBI Entrez* database[37]. For k-mer matching the bacterial genome and its reverse complement were indexed into a `KmerTable`. Low complexity regions were masked using *tantan* in this process. To increase search sensitivity spaced k-mers with the spacing pattern `111*1*11*1**11*111`[38] were used instead of contiguous ones. To significantly reduce the number of hits for the downstream stages, only matches with at least another match on the same diagonal were filtered[39]. For each diago-

[36] Steinegger and Söding (2017).

[37] *GenBank* accessions `NC_000932` and `NC_000911`

[38] Choi, Zeng, and Zhang (2004).

[39] This is the application the double-hit strategy.

nal then only the first hit is considered, since multiple hits on the same diagonal presumably correspond to the same homologous region. In the following each hit was extended with X-drop criterion using `align_local_ungapped()`. As actual alignments were computed later, only similarity scores were calculated here to decrease the runtime. Based on a fixed score threshold well-performing hits were filtered. For each remaining hit a local gapped alignment was performed using `align_local_gapped()`. Again, only the score was computed, since most resulting alignments were expected to be insignificant. The substitution matrix and gap penalty were taken from the *MMseqs2* default. Afterwards each score was tested on significance using an `EValueEstimator`. For the estimation of the E-value the length of the bacterial genome was given as 2-fold as alignment search was performed against both, the forward and reverse complement sequence. Alignments with $E \leq 0.01$ were considered significant. For the significant hits the alignment was repeated with the alignment trace included. Multiple hit positions could result in the same alignment, since insertions and deletions in a homologous region result in hits on different diagonals. Hence, duplicate alignments were filtered out and the remaining alignments were reported. Figure 6.7 shows how the number of hits decreased with each stage.

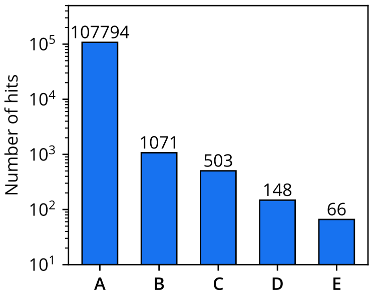
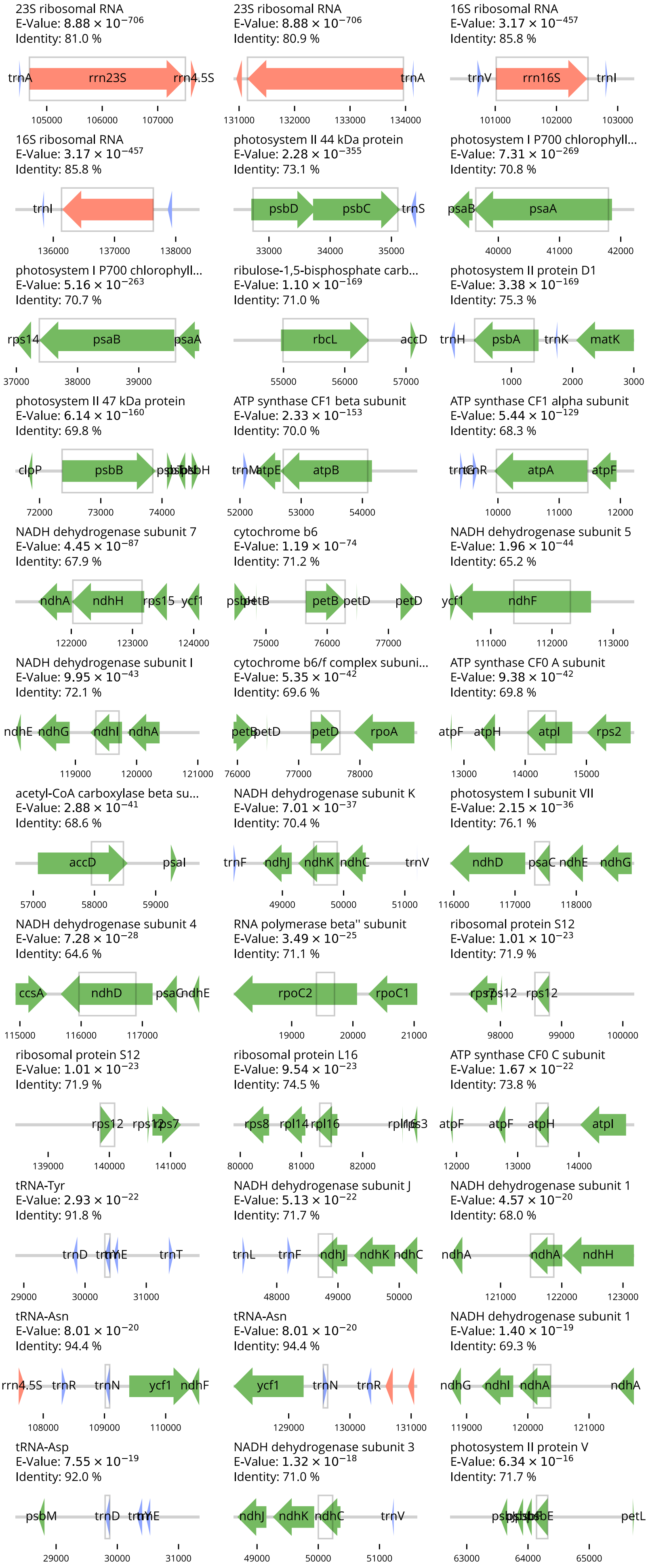


Figure 6.7: Number of remaining hits after each alignment search stage. A Initial k-mer matching. **B** Double hit filtering and aggregation of hits on same diagonal. **C** Ungapped hit extension. **D** Gapped alignment. **E** Aggregation of duplicate alignments.

For deeper analysis of the found sequence similarities, the genes found in the homologous regions of the chloroplast genome were identified. For this purpose the sequence feature table for the chloroplast genome was downloaded from *NCBI Entrez* and the gene positions were extracted. For each homologous region including some margin the genes in that region were visualized using *Matplotlib* (Figure 6.8). The most significant sequence conservation is found in the 23s and 16s ribosomal RNA sequences. This is not surprising: Ribosomal RNAs are well conserved in the evolutionary process. This is one of the reasons the 16s rRNA and 23s rRNA sequence similarity is a popular method for reconstructing prokaryotic phylogeny[40]. The following hits include mainly proteins with central functionality for photosynthetic activity, such as ribulose-1,5-bisphosphate carboxylase-oxygenase and proteins involved in the photosystem complexes. With decreasing sequence similarity, the local alignments often cover only a part of a protein. This issue could be mitigated by performing an alignment search based on translated nucleotide sequences, since protein sequences are known to be better conserved than nucleic acid sequences. Strikingly, the homologous regions rarely extend into non-coding regions. This observation signifies the strong evolutionary conservation of genes.

[40] Ludwig and Schleifer (1994).



23S ribosomal RNA
E-Value: 8.88×10^{-706}
Identity: 81.0 %

23S ribosomal RNA
E-Value: 8.88×10^{-706}
Identity: 80.9 %

16S ribosomal RNA
E-Value: 3.17×10^{-457}
Identity: 85.8 %

16S ribosomal RNA
E-Value: 3.17×10^{-457}
Identity: 85.8 %

photosystem II 44 kDa protein
E-Value: 2.28×10^{-355}
Identity: 73.1 %

photosystem I P700 chlorophyll...
E-Value: 7.31×10^{-269}
Identity: 70.8 %

photosystem I P700 chlorophyll...
E-Value: 5.16×10^{-263}
Identity: 70.7 %

ribulose-1,5-bisphosphate carb...
E-Value: 1.10×10^{-169}
Identity: 71.0 %

photosystem II protein D1
E-Value: 3.38×10^{-169}
Identity: 75.3 %

photosystem II 47 kDa protein
E-Value: 6.14×10^{-160}
Identity: 69.8 %

ATP synthase CF1 beta subunit
E-Value: 2.33×10^{-153}
Identity: 70.0 %

ATP synthase CF1 alpha subunit
E-Value: 5.44×10^{-129}
Identity: 68.3 %

NADH dehydrogenase subunit 7
E-Value: 4.45×10^{-87}
Identity: 67.9 %

cytochrome b6
E-Value: 1.19×10^{-74}
Identity: 71.2 %

NADH dehydrogenase subunit 5
E-Value: 1.96×10^{-44}
Identity: 65.2 %

NADH dehydrogenase subunit I
E-Value: 9.95×10^{-43}
Identity: 72.1 %

cytochrome b6/f complex subuni...
E-Value: 5.35×10^{-42}
Identity: 69.6 %

ATP synthase CF0 A subunit
E-Value: 9.38×10^{-42}
Identity: 69.8 %

acetyl-CoA carboxylase beta su...
E-Value: 2.88×10^{-41}
Identity: 68.6 %

NADH dehydrogenase subunit K
E-Value: 7.01×10^{-37}
Identity: 70.4 %

photosystem I subunit VII
E-Value: 2.15×10^{-36}
Identity: 76.1 %

NADH dehydrogenase subunit 4
E-Value: 7.28×10^{-28}
Identity: 64.6 %

RNA polymerase beta'' subunit
E-Value: 3.49×10^{-25}
Identity: 71.1 %

ribosomal protein S12
E-Value: 1.01×10^{-23}
Identity: 71.9 %

ribosomal protein S12
E-Value: 1.01×10^{-23}
Identity: 71.9 %

ribosomal protein L16
E-Value: 9.54×10^{-23}
Identity: 74.5 %

ATP synthase CF0 C subunit
E-Value: 1.67×10^{-22}
Identity: 73.8 %

tRNA-Tyr
E-Value: 2.93×10^{-22}
Identity: 91.8 %

NADH dehydrogenase subunit J
E-Value: 5.13×10^{-22}
Identity: 71.7 %

NADH dehydrogenase subunit 1
E-Value: 4.57×10^{-20}
Identity: 68.0 %

tRNA-Asn
E-Value: 8.01×10^{-20}
Identity: 94.4 %

tRNA-Asn
E-Value: 8.01×10^{-20}
Identity: 94.4 %

NADH dehydrogenase subunit 1
E-Value: 1.40×10^{-19}
Identity: 69.3 %

tRNA-Asp
E-Value: 7.55×10^{-19}
Identity: 92.0 %

NADH dehydrogenase subunit 3
E-Value: 1.32×10^{-18}
Identity: 71.0 %

photosystem II protein V
E-Value: 6.34×10^{-16}
Identity: 71.7 %

Figure 6.8: Identified homologies between cyanobacteria and chloroplast genomes. The figure displays the 36 most significant homologous regions between the genomes of *Synechocystis* sp. PCC 6803 and *A. thaliana* chloroplasts. For each homologous region the position in the chloroplast genome is shown (gray box). The arrows depict position and coding strand of protein coding sequences (green), rRNAs (red) and tRNAs (blue). If available, the name of the gene is shown in the arrow.

Chapter 7
General structure information

The application example of this chapter was adapted from the example gallery at the *Biotite* documentation website (Table A.1).

7.1 Introduction

The PDB is a large archive of experimentally determined macromolecular structure models. Detailed interpretation of these structures sometimes requires additional information: For example the PDB format does only contain 3-letter abbreviations of residues[1], while the *mmCIF* format omits information about atom connectivity within a residue. Hence, the *Chemical Component Dictionary* (CCD)[2] has been made available as companion catalog for structure analysis. For each residue, that is part of any PDB structure it lists comprehensive information about the compound in general and its comprised atoms. *Biotite* harnesses this information for a variety of its functionalities, e.g. the identification of bonds in structure models without bond information, the assembly of structures from scratch or the prediction of hydrogen positions (see Chapter 14).

7.2 Implementation

7.2.1 Parsing the CCD

The CCD is provided by the PDB as single *mmCIF* file. Each residue is described by a data block in this file. This format can be parsed with the respective parser from *Biotite*. However, parsing the CCD at demand of the user would significantly increase run time, as the CCD contains more information about each residue as requested and parsing the *mmCIF* file is relatively slow. Therefore, the *Biotite* package contains necessary data from the CCD as preprocessed files, separated by data type[3]. The data is stored in the binary *MessagePack* format which is more compact than comparable text formats and can be read relatively fast. If certain information is demanded by the user, the respective data file is parsed and the data is cached in memory.

7.2.2 Bond identification

Biotite stores bonds between atoms within a residue in a dictionary that maps the name of a residue and the name of two atoms to a bond type (see Section 2.2.4). If two atoms are not bonded with each other, no mapping is available for the combination. The dictionary is accessible by the user via the `bond_dataset()` function, but mainly it is used in-

[1] Monomers inside larger macromolecules as well as small molecules are termed '*residues*' here.

[2] Westbrook et al. (2015).

[3] e.g. one file for full residue names, one file for atom connectivity, etc.

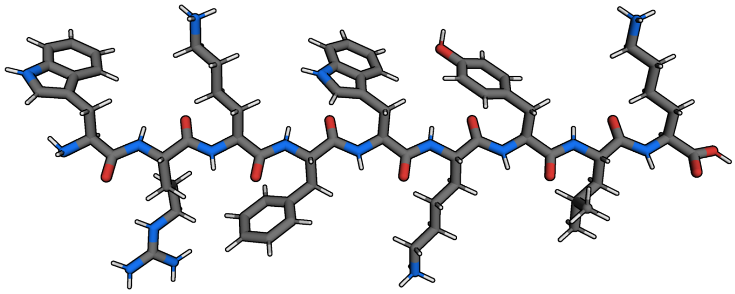


Figure 7.1: Linearly assembled antimicrobial peptide. The sequence of the peptide is `WRKFWKYLK`.

ternally by `connect_via_residue_names()` to create a `BondList` for an `AtomArray` or `AtomArrayStack`. This allows accurate computation of bonds for structure models even from other sources than the PDB, as the CCD comprises a wide range of residues, including protein and nucleic acid monomers, small molecules, saccharides and lipids.

7.2.3 Building residues

The CCD does not only provide name, element and charge for each atom, but also coordinates, either from an experimentally determined structure or from computation results. Together with the atom connectivity, *Biotite* can use this information to build an `AtomArray` from residue name with the `residue()` function.

7.3 Application example

Companion source code: `peptide_assembly.py`

A linear peptide can be used as starting conformation for *de novo* protein structure prediction[4]. Since no folding is attempted in this simple approach, knowing the geometric course of the protein backbone and the atoms of the amino acid side chains is sufficient to build such linear conformation from sequence only. An antimicrobial peptide against *Mycobacterium tuberculosis*[5] should serve as example here.

Peptide assembly began with building a backbone structure for all residues in the sequence. This chain contained the N, C_α and C for each residue, using backbone dihedral angles of 180° resulting in a '*zigzag*' chain lying in the z-plane. Then, for each amino acid the atoms and their positions and bonds were obtained via `residue()`. Each residue was superimposed onto to the respective positions of the backbone and the atoms that are lost in the peptide condensation[6] were removed. A bond was formed between C and N atoms of subsequent residues. Finally the geometries of the hydrogen and oxygen atom of the peptide bond were adjusted. This was achieved by superimposition of known peptide bond coordinates onto each peptide bond of the assembled peptide. The structure of the finished peptide chain is shown in Figure 7.1.

[4] Le et al. (2020).

[5] Ramón-García et al. (2013).

[6] namely a hydroxy group of the C-terminus and a hydrogen atom from the N-terminus

Chapter 8
Unit cells, simulation boxes and macromolecular assemblies

8.1 Introduction

Molecular structures seldomly exist in isolation. In crystals used for X-ray structure elucidation, macromolecules are densely packed in periodic organization, with one or more copies in each unit cell. Similarly, molecules in an molecular dynamics (MD) simulation are placed in a simulation box, which is repeated periodically.

In nature biomacromolecules do not necessarily assemble according to the conformation suggested by the crystallographic unit cell. Hence, the actual *macromolecular assembly*[I] is more important: It describes how one or multiple copies of the molecules in the structure are spatially arranged to form a putative functional unit.

In order to analyze structure models in context of the unit cell, simulation box or assembly, *Biotite* provides functions to handle them, which are covered in this chapter. This enables the application of *Biotite* to the domains of MD simulations and large macromolecular complexes.

8.2 Implementation

8.2.1 Periodic boxes

Unit cells and simulation boxes can be handled equivalently, as they describe the same circumstance: For each atom one copy exists one cell or box length away in each dimension. Therefore, for the purpose of the implementation in *Biotite* and the description in this chapter both are aggregated into the term ***box***. The `box` attribute of an `AtomArray` is a (3×3)-dimensional array, where each row is one of the three box vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ (Figure 8.1). In an `AtomArrayStack` each of the m models can have an individual box, resulting in a $(m \times 3 \times 3)$-dimensional array. While the `box` can be manually set, it is most commonly read from a structure file. The popular *PDB* and *mmCIF* formats store box vectors using the vectors' lengths $|\vec{a}|$, $|\vec{b}|$ and $|\vec{c}|$ and the angles α, β and γ between them (Figure 8.1). These values can be converted to box vectors using `vectors_from_unitcell()` and vice versa with `unitcell_from_vectors()`.

Based on the `box` attribute, a number of manipulations can be performed. Most prominently, structure models, that are fragmented due

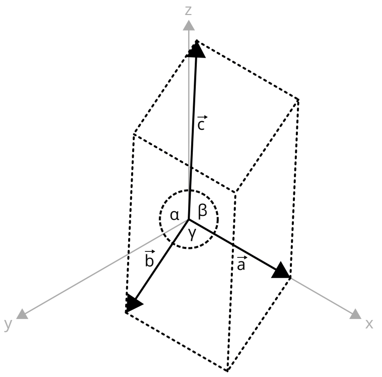


Figure 8.1: Unit cell or simulation box vectors. The axes of the coordinate system are shown in gray. In the case of unit cells $\vec{a}$ lies on the x-axis and $\vec{b}$ on the xy-plane.

[I] also called *biological assembly* or *biological unit* to the periodic boundary, can be reattached with `remove_pbc()` or periodic copies of a structure can be added with `repeat_box()` to study their interaction with each other. Furthermore geometric properties, such as distances and angles between atoms, can be measured taking periodic boundary conditions into account. Geometric measurements rely on one or multiple displacement vectors between atoms. For example, for a distance the displacement between two atoms is required and for the angle between atoms ABC, the displacement AB and BC is required. Let $\vec{x}$ be such a displacement vector that is simply calculated using position vector difference. However, in the periodic case $\vec{x}$ may not represent the shortest line, between the atoms, since a periodic copy might be closer. The aim is to find a vector $\vec{y}$ from $\vec{x}$ that minimizes the displacement vector length by adding a multiple of each vector in the box matrix B. Thus,

$$\mathbf{B} = \begin{pmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{pmatrix},$$

$$\vec{y} = \text{arg min}\, |\vec{u}|$$

$$\text{s.t. } \vec{u} = \left(\vec{x} + \mathbf{B}^\mathsf{T} \cdot \vec{p}\right),\ \vec{p} \in \mathbb{Z}^3. \tag{8.1}$$

To find $\vec{y}$ the approach from the *MDTraj* package[2] was adapted. $\vec{x}'$ is computed, which describes $\vec{x}$ as fraction of the box vectors:

$$\vec{x} = \mathbf{B}^\mathsf{T} \cdot \vec{x}'$$

$$\vec{x}' = \mathbf{B}^{\mathsf{T}^{-1}} \cdot \vec{x}. \tag{8.2}$$

Substituting Equation 8.2 in Equation 8.1,

$$\vec{u} = \mathbf{B}^\mathsf{T} \cdot (\vec{x}' + \vec{p})\,. \tag{8.3}$$

In orthorhombic boxes the box vectors are orthogonal to each other. In this case determining $\vec{y}$ becomes the problem of finding the minimum absolute value for each vector component of $(\vec{x}' + \vec{p})$. This is simply

$$(\vec{x}' + \vec{p})_i = \begin{cases} \vec{x}_i' \,\%\, 1, & \text{if } \vec{x}_i' \,\%\, 1 \leq 0.5 \\ (\vec{x}_i' \,\%\, 1) - 1, & \text{if } \vec{x}_i' \,\%\, 1 > 0.5 \end{cases}, \tag{8.4}$$

where % denotes the modulo operator. Intuitively, this means that for each dimension the displacement vector is moved within one box length and then the direction with the minimum length is chosen (Figure 8.2).

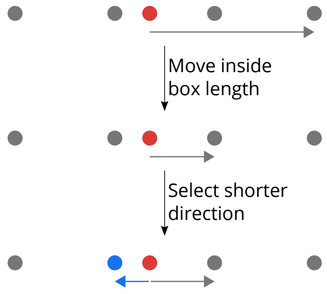


Figure 8.2: Displacement vector calculation in a orthorhombic box. The figure shows the calculation of $\vec{y}$ (blue arrow) from $\vec{x}$ (upper gray arrow) according to Equation 8.4. The two points, whose displacement is measured are shown in red and in gray, including the periodic copies of the latter one.

For triclinic boxes a more time consuming approach is required, since the problem cannot be simplified to Equation 8.4. Instead, the displacement vector is moved inside the box ($\vec{x}' \,\%\, 1$) and the directly adjacent periodic copy with the lowest distance is chosen, by checking all $3^3 = 27$ possibilities in a vectorized manner:

$$\vec{u} = \mathbf{B}^\mathsf{T} \cdot ((\vec{x}' \,\%\, 1) + \vec{p})\,,\ \vec{p}_i \in \{-1, 0, 1\}. \tag{8.5}$$

[2] McGibbon et al. (2015).

To handle periodic boundaries in a `CellList`, adjacent periodic copies of the structure are also added to it. With a total of 27 copies in the `CellList`, its creation is clearly slowed down. However, to find atoms in vicinity of given coordinates is now simply a lookup in the relevant cells, as it is done for the non-periodic case. Note that for heavily skewed boxes the periodic copy with the lowest distance may not be in the directly adjacent box and the correct solution might not be found in this rare case (Figure 8.3).

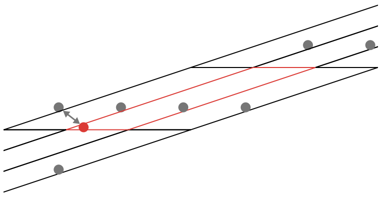


Figure 8.3: Minimum distance in a heavily skewed box. The minimum distance between the red and the gray point is sought. As the box is heavily skewed, the periodic copy with the minimum distance is found not within the same (red) or directly adjacent (black) boxes, but outside of the searched boxes (arrow).

These functionalities are especially important for analyses on MD simulation trajectories, as macromolecular chains usually move beyond the periodic boundary and reenter the box from the other side during the course of the simulation.

8.2.2 Macromolecular assemblies

The coordinates given by a structure file from the PDB refer to the method, which was used to resolve the structure. For example, in X-ray crystal structures they represent the asymmetric unit. The *PDBx/mmCIF* format provides instructions on how to obtain a certain macromolecular assembly from these coordinates. The file can define multiple assemblies, each one identified by a unique *assembly ID*. The `pdbx_struct_assembly` category lists these IDs together with a short description. The `pdbx_struct_assembly_gen` category describes which combination of transformations need to be applied to which chains[3] to obtain the assembly. Finally, `pdbx_struct_oper_list` gives the rotation matrix and translation vector for each operation. In *Biotite* the `get_assembly()` function parses these fields to obtain the necessary transformations to obtain the assembly for a given assembly ID. These instructions are applied to the `AtomArray` parsed from the asymmetric unit to build and return an `AtomArray` that represents the chosen assembly. This procedure can be likewise used for multi-model structures to obtain an `AtomArrayStack`.

8.3 Application example

Companion source code: `phage_capsid.py`

In this example the macromolecular assembly for the capsid of the λ-phage[4] was extracted from the PDB entry 7VII and visualized. The structure file was downloaded in *PDBx/mmCIF* format. The structure model contains two different proteins, identified by unique *entity IDs* (Table 8.1 top). Furthermore, multiple assemblies are available (Table 8.1 bottom), from which the '*complete icosahedral assembly*' was chosen. The assembly listed first[5] typically corresponds to the most relevant one, which is the reason why the first assembly ID is used by default in `get_assembly()`, if no explicit ID is given. `get_assembly()` was used to create an `AtomArray` for the chosen assembly ID. The entity ID was included in the creation as additional atom annotation, to distinguish between the two types of protein in the visualization, as shown in Figure 8.4.

[3] A chain comprises atoms with the respective *chain ID*.

[4] Wang, Zeng, and Wang (2022).

[5] usually assembly with ID '1'

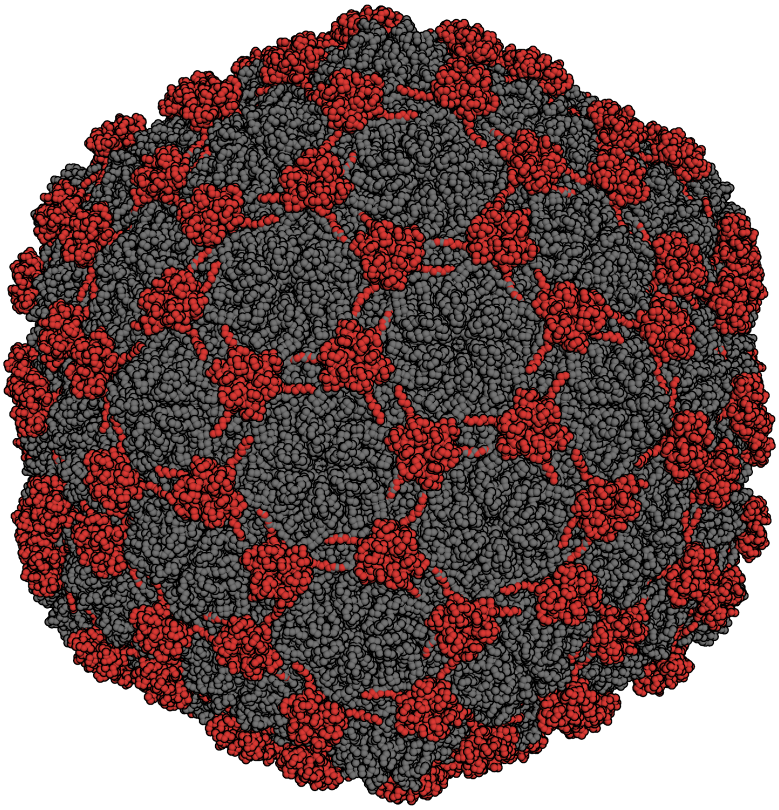


Figure 8.4: Structure of the bacteriophage λ capsid. The major capsid protein and capsid decoration protein are shown in gray and red, respectively.

Table 8.1: Entities and assemblies.

Entities	
ID	Description
1	Major capsid protein
2	Capsid decoration protein

Assemblies	
ID	Description
1	complete icosahedral assembly
2	icosahedral asymmetric unit
3	icosahedral pentamer
4	icosahedral 23 hexamer
5	icosahedral asymmetric unit, std point frame

Chapter 9
Partial charge estimation

The implementation of the functionality presented in this chapter was produced in collaboration in the course of a Bachelor's thesis[1].

9.1 Introduction

In the classical mechanical view of a molecule partial charges arise when the location of an electron is shared among multiple atoms. Although quantum mechanics rejects the underlying concept of point charges and substitutes it with a continuous electron density distribution, the partial charge is still a useful quantity to derive a number of physical properties of a molecule, such as acidity[2]. Furthermore, molecular modelling often resorts to a classical mechanical system to describe a molecular system. This allows computations in a reasonable time scale, that is not achievable with the rigorous incorporation of quantum mechanics. In consequence, techniques like MD simulations or molecular docking[3] need to assign partial charges to the point particles, the atoms, in order to calculate electrostatic interactions between them.

Although the partial charges of atoms in a molecular system can be accurately calculated using the mathematics provided by quantum mechanics, this is again computationally too costly for applications to biomacromolecules. An alternative is the tabulation of partial charges for a range of single residues calculated this way[4], and picking the charges for the required residues. However, this approach is still not completely accurate, as the partial charges shift in the context of the larger molecular system. More importantly this approach limits the applicable systems to ones that contains only the tabulated residues. As the *Biotite* package strives to be generally applicable, irrespective of the molecular model at hand, a more universal method to compute partial charges is desired.

The *partial equalization of orbital electronegativity* (PEOE) method[5] distributes charges between directly bonded atoms solely based on the chemical elements and the number of bonds of each atom: In a bond between two atoms the more electronegative atom attracts the binding electrons attaining a negative partial charge in this process, while the other atom obtains a positive partial charge with the same magnitude. The electronegativity can be defined as

$$\chi = \frac{E_I + E_A}{2}, \tag{9.1}$$

where E_I is the ionization energy and E_A is the electron affinity of the

[1] Anter (2021).

[2] Gasteiger and Marsili (1980).

[3] Oostenbrink et al. (2004); Trott and Olson (2010).

[4] Cieplak et al. (1995).

[5] Gasteiger and Marsili (1980).

valence state of the atom[6]. The electronegativity is furthermore dependent on the charge Q of the atom, for instance already negatively charged atoms exert less attraction to electrons due to electrostatic repulsion. The PEOE method approximates the electronegativity of an atom i with

$$\chi_i(Q) = a_i + b_i Q + c_i Q^2, \tag{9.2}$$

where a, b and c are tabulated values dependent on element and number of bonded atoms. The partial charges are now distributed within a network of bonded atoms in an iterative manner. Starting from the formal atom charges Q^0, the charge of each atom i is updated in each iteration step n with

$$Q_i^{n+1} = Q_i^n + \left(\frac{1}{2}\right)^{n+1} \cdot \sum_j \frac{\chi_j(Q_j^n) - \chi_i(Q_i^n)}{\chi^+},$$
$$\chi^+ = \begin{cases} \chi_i(1), \text{ if } \chi_j(Q_j^n) > \chi_i(Q_i^n) \\ \chi_j(1), \text{ otherwise} \end{cases} . \tag{9.3}$$

Here j iterates over the atoms directly bonded to atom i. In the studies conducted for the original PEOE description $n = 6$ iterations were sufficient to achieve convergence in any case[7].

Another popular approach to the problem of estimating partial charges are the charge equilibration[8] (*Qeq*) and its derived methods. However, *Qeq* requires pairwise atom distances, which change dependent on the molecular conformation. Hence, a proper application of this method would require periodic recalculation during a molecular simulation. Furthermore, usage of PEOE is consistent with the data preparation procedure for molecular docking tools of the *AutoDock* software family[9].

9.2 Implementation

The PEOE algorithm was implemented in the `partial_charges()` function written in *Cython*. As required input it takes an `AtomArray`, representing the molecular system to calculate the partial charges for. The chemical elements and formal charges are taken from the annotations of the `AtomArray`. Optionally, the initial charges Q^0 can be given separately, for example to handle the delocalized electrons of deprotonated carboxy groups. The bonds are taken from a `BondList` associated to the `AtomArray`. By default the function uses 6 iterations as proposed in the original paper.

Although the described algorithm is basically straightforward to implement, some special cases need to be considered. First, hydrogen atoms only have one electron. In consequence a hydrogen atom with a charge of 1 cannot be ionized, which poses a problem for calculating χ^+ in Equation 9.3. For hydrogen atoms $\chi^+ = 20.02\,\text{eV}$ was employed here in accordance with the original article [10]. The second problem are ele-

[6] Mulliken (1934).

[7] Gasteiger and Marsili (1980).

[8] Rappe and Goddard (1991).

[9] Goodsell and Olson (1990).

[10] Gasteiger and Marsili (1980).

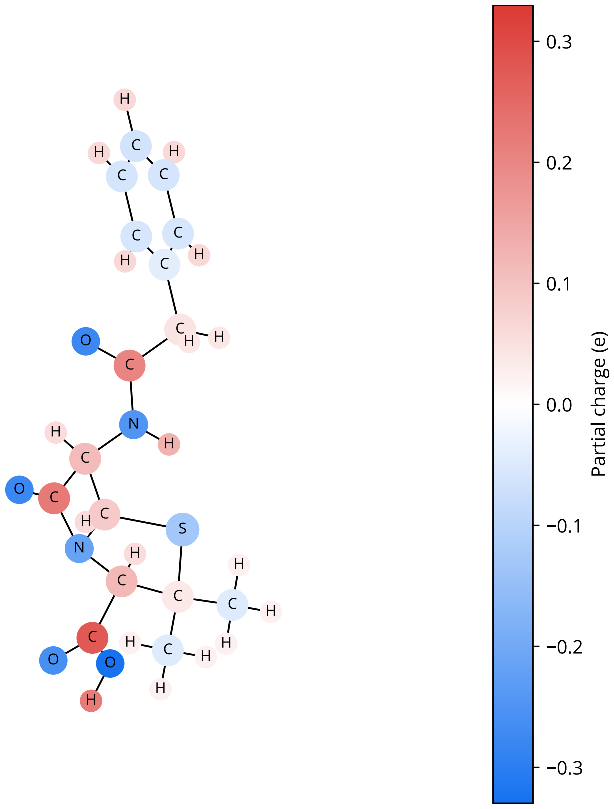


Figure 9.1: Partial charge distribution of penicillin G according to PEOE method. `plot_ball_and_stick_model()` was used for visualization.

ments for which the parameters a, b and c are not tabulated. Since most elements occurring in organic molecules are present, it would be unreasonable to reject partial charge calculation for large molecular systems, if parameters were missing for only a few atoms. Instead the implementation completely ignores such atoms, i.e. no charge is transferred from or to an atom with missing parameters. In the return value their partial charge is given as *not-a-number*. This way partial charges can be calculated for all organic molecules at the cost of inaccuracies in vicinity of non parametrized atoms.

9.3 Application example

Companion source code: `penicillin_charges.py`

To demonstrate the implemented algorithm, PEOE is applied to the β-lactam antibiotic penicillin G. The calculated charges for each atom after the default number of iterations are shown in Figure 9.1. To support the decision for 6 iterations, the partial charges were calculated with different number of iterations and compared to 10 iterations as the '*correct*' reference. As measure of error the

$$\mathrm{RMSD}_{\mathrm{Charge}}(n) = \sqrt{\langle (Q_i^n - Q_i^{10})^2 \rangle} \tag{9.4}$$

was calculated (Figure 9.2). After 6 iterations the $\mathrm{RMSD}_{\mathrm{Charge}}$ was merely 0.0007 e with respect to the result after 10 iterations.

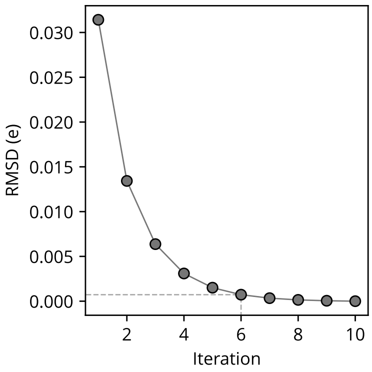


Figure 9.2: Partial charge RMSD in dependence of iteration step. The default number of iterations is marked with the dashed line.

Chapter 10
Molecular docking with AutoDock

The application example of this chapter was adapted from the example gallery at the *Biotite* documentation website (Table A.1).

10.1 Introduction

Molecular docking is the process of finding the conformation that a receptor and ligand molecule have in complex with each other, the so called *binding mode*. In computer-aided drug design, docking of small molecules to biomacromolecules is of particular interest. Docking methods can be separated into three categories[1]:

- *rigid docking*, where the individual conformation of receptor and ligand is fixed and only the relative positioning is searched,
- *flexible docking*, where conformation of receptor and ligand can adapt in the binding process, and
- *flexible-rigid docking*, where only the conformation of the small molecule can adapt.

While flexible docking gives most accurate results, especially since it is able to simulate an induced fit, rigid docking allows fast computation. Flexible-rigid docking offers a tradeoff between accuracy and performance.

The *AutoDock* software family[2] is a representative of the latter category, though selected receptor side chains can be optionally handled as flexible. Although MD simulations and deep learning[3] have become increasingly promising approaches for drug design, the force-field based *AutoDock* has still a high popularity due to its relatively fast computation. The most modern iteration of this family is *AutoDock Vina*[4], which uses a Metropolis-Monte-Carlo algorithm in combination with local optimization to find conformations that minimize the energy function. *Vina* proposes multiple such candidates for the binding mode, so called *poses*, and rates each one with the estimated free energy of the binding process.

Since *Vina* achieves reasonable docking results but requires a complex setup, *Biotite* provides a simple interface to this software, that hides most of the complexity and does not require additional dependencies, to allow fast and simple docking.

10.2 Implementation

Docking with *Vina* in principle requires only a single command line invocation, which is encapsulated by the `VinaApp` class (see Section 2.6)

[1] Fan, Fu, and Zhang (2019).

[2] Goodsell and Olson (1990).

[3] Amaro et al. (2018); Gawehn, Hiss, and Schneider (2016).

[4] Trott and Olson (2010).

in *Biotite*. The `VinaApp` governs the creation of input files for docking and parsing of the resulting output files. Furthermore, it provides options to customize other parameters of the docking procedure such as its exhaustiveness. However, the main challenge of interfacing *Vina* is the data preparation: The program uses the custom *PDBQT* format for structure input and output. The format makes alterations to the *PDB* format, which are discussed in the following subsections. The authors of the *AutoDock* software family provide the *MGLTools* software package for *PDBQT* file preparation. However, the license of the software restricts is usage to academic applications and the package requires *Python* 2.7, which is no longer supported. Therefore, *Biotite* provides the `PDBQTFile` class for reading and writing structure data in this format as alternative.

10.2.1 Partial charges

The charge column of *PDBQT* files contains partial charges instead of formal charges. By default they are calculated using the PEOE method implemented in `partial_charges()` (see Chapter 9).

10.2.2 Atom types

Instead of the element of an atom, the *AutoDock atom type* is given, that incorporates information about the direct chemical environment of the atom. For example aliphatic and aromatic carbon atoms are distinguished. `PDBQTFile` uses the `BondList` associated to the input `AtomArray` to determine the environment and select the correct atom type. Non-polar hydrogen atoms are removed entirely in *PDBQT* files.

10.2.3 Rotatable bonds

To define which bonds in the ligand molecule are rotatable, the *PDBQT* format represents a molecule in a tree-like manner: A group of connected atoms, that are rotated collectively when the dihedral angle of a rotatable bond is altered, is a node in this tree. Because such a group of atoms may contain rotatable bonds itself, a node may have child nodes. In the *PDBQT* format each group of atoms is wrapped by a '`BRANCH`' and an '`ENDBRANCH`' record, which also comprise the index of the atoms connected by the rotatable bond. A child node is described by a '`BRANCH`'/'`ENDBRANCH`' block nested within a group. The '`ROOT`' record defines the root of this tree. The resulting number of torsional degrees of freedom is written into a '`TORSDOF`' record. *Biotite* again uses the `BondList` to identify rotatable bonds and groups of atoms connected by them. A bond is considered rotatable, if it is a single bond, it does not lead to a terminal atom[5] and the two connected atoms are not within the same ring. If two atoms are still connected via a 'path' through the `BondList` ignoring the rotatable bonds, they belong to the same node. By default, the node, where the first atom in the structure belongs to, is taken as root of the tree. Child nodes of the root are

[5] Rotation about a terminal bond would have no effect on the coordinates of the affected atom.

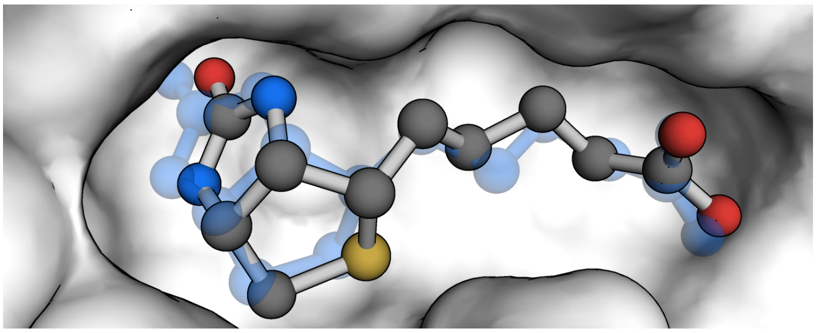


Figure 10.2: Putative and reference binding mode of the strepatividin-biotin complex. The surface of the streptavidin binding pocket with the docked biotin molecule is shown. The reference binding mode from the original structure is shown in transparent blue.

wrapped by a 'BRANCH' and 'ENDBRANCH' record. This procedure is repeated recursively for each child node until all nodes were added to the *PDBQT* file.

10.3 Application example

Companion source code: `biotin_docking.py`

As example docking application the ligand biotin was docked to a structure model of the receptor streptavidin (PDB: 2RTG)[6]. This structure was chosen, as hydrogen atoms are resolved in the model, which is required for the docking setup. Furthermore, it already includes bound biotin. However, for the purpose of this example it was assumed that the binding mode is unknown.

In the first step a single streptavidin monomer is extracted from the dimeric asymmetric unit. The existing (reference) biotin molecule is removed and a new biotin model is taken from the CCD (see Chapter 7) as ligand. Docking between receptor and ligand was performed in a 20 Å radius of the reference ligand centroid using `VinaApp`. The returned number of models and the maximum free-energy was not restricted to obtain a larger number of poses. Finally, the structure models and corresponding predicted free energy values were obtained from the docking results.

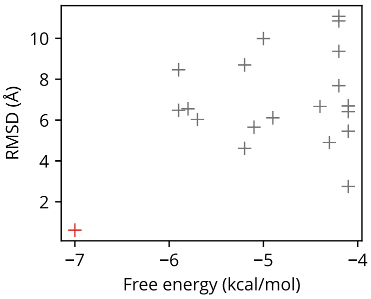


Figure 10.1: Accuracy of predicted binding poses. For each binding pose predicted by *Vina*, the predicted free energy and the RMSD to the reference binding mode is shown. The minimum free energy pose is marked in red.

When measuring the docking accuracy in terms of the RMSD between each pose and the reference ligand, the pose with the lowest free energy was also the closest to the reference (Figure 10.1). This indicates a good accuracy, as the binding mode with the lowest energy is theoretically also the most prevalent one and is therefore found in experimental data. The minimum free energy pose is shown in Figure 10.2, which demonstrates again how close the predicted binding mode is compared to the reference mode.

[6] Katz (1997).

Chapter 11
Elastic network models

The implementation of the functionality presented in this chapter was produced in collaboration with coworkers[1].

11.1 Introduction

While experimentally determined structures only give static images of a given macromolecular system, in-depth insights often require knowledge about motions of atoms, residues and entire domains. An MD simulation generates atomistic trajectories[2] that allow analyses of such motions. Based on an interatomic potential $V(\vec{r})$, where $\vec{r}$ are the atom coordinates, it calculates the interatomic forces $F(\vec{r}) = -\nabla_p V(\vec{r})$ and integrates them over small time steps. However, bond vibrations occur in the range of femtoseconds and dictate the integration step size, while global conformational changes may take milliseconds or longer[3]. In consequence the MD simulation of time scales for observation of global movements is computationally unfeasible due to the large number of required steps.

Elastic network models (ENMs) offer a fast but coarse-grained alternative to assess global dynamics of a molecular system. First, consider the potential function V as dependent on pairwise atom distances $s_{ij} = |\vec{r}_i - \vec{r}_j|$ instead of $\vec{r}$ directly. By expanding $V(r)$ into a *Taylor* series around the input atom coordinates $\vec{r}^0$ and the corresponding distances s^0, it becomes

$$\begin{aligned} V(r) = \sum_{ij} [V(s^0_{ij}) \\ &+ \left.\frac{\partial V}{\partial s_{ij}}\right|_{s^0_{ij}} \left(s_{ij} - s^0_{ij}\right) \\ &+ \frac{1}{2} \left.\frac{\partial^2 V}{\left(\partial s_{ij}\right)^2}\right|_{s^0_{ij}} \left(s_{ij} - s^0_{ij}\right)^2 \\ &+ \dots]. \end{aligned} \tag{11.1}$$

The model assumes that the input structure[4] is at energy minimum for all pairs of atoms[5]. The first term of the series is constant and represents the potential at energy minimum. Hence it can be set w.l.o.g. to zero. $\partial V / \partial s_{ij}$ vanishes at energy minimum and in consequence the entire second term. Terms after the third one are omitted, leading to

$$V(r) = \sum_{ij} \left[\frac{1}{2} \left.\frac{\partial^2 V}{\left(\partial s_{ij}\right)^2}\right|_{s^0_{ij}} \left(s_{ij} - s^0_{ij}\right)^2 \right]. \tag{11.2}$$

Consequently, ENMs describe the energy landscape in vicinity of the energy minimum. Equation 11.2 simplifies the ENM to a system of

[1] Krumbach, J. H. and Islam, F., TU Darmstadt

[2] a time series of atom coordinates

[3] Bahar et al. (2010).

[4] usually determined from an experiment

[5] Bahar et al. (2010).

Hookean springs (Figure 11.1),

$$V(r) = \sum_{ij} \frac{1}{2}\gamma_{ij}\left(s_{ij} - s_{ij}^{0}\right)^{2}, \tag{11.3}$$

where γ_{ij} is the force constant of the virtual '*spring*' between atom i and j.

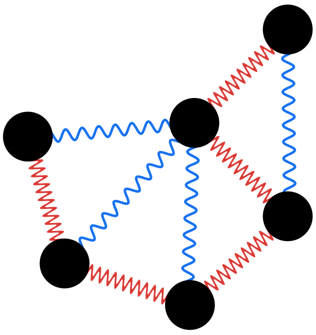


Figure 11.1: Schematic depiction of an ENM. The atoms are connected by springs with two different force constants.

Typically, not all atoms of a molecular system are included in an ENM. For protein structures generally only C_α atoms are used. Different approaches can be used to calculate γ_{ij} for each pair of residues based on the input structure. In the most basic case a simple cutoff distance is defined, within which two residues are considered to interact with each other: If two C_α atoms have distance below this cutoff, their γ is set to a constant value, otherwise to 0. More modern models offer a continuous distance-dependent behavior[6], consider the type of interacting residues[7] or include both of these information[8]. However, the choice of the model has often only little impact on the insights obtained from the ENM, as global movements mostly result from the overall 3D shape of the system[9].

Due to the representation as spring network, an ENM can be subjected to normal mode analysis (NMA). Each normal mode is a collective oscillation of all atoms at the same frequency and phase. The superimposition of all normal modes gives the entirety of movements in the molecular system within the assumptions of the ENM. If the direction of these movements is irrelevant for the use case, a *Gaussian* network model (GNM) [10] can be employed, defined by the Kirchhoff matrix

$$\Gamma_{ij} = \begin{cases} -\gamma_{ij}, & \text{if } i \neq j \\ \sum_{i,i\neq j}\gamma_{ij}, & \text{if } i = j \end{cases}. \tag{11.4}$$

Each eigenvalue-eigenvector pair of Γ corresponds to one normal mode, where the eigenvalue $\lambda_k \propto \omega_k^2$, the frequency of the oscillation[11]. The corresponding eigenvector quantifies the contribution of each atom in this oscillation. Modes with $\lambda_k = 0$ represent rotations and translations of the entire molecular system[12]. Otherwise, the amplitude of a mode k is proportional to $1/\lambda_k$, as on average each mode has the same vibrational energy, which is proportional to[13] ω_k^2. In consequence, the slow modes with small λ_k represent the global movements sought after. The pseudoinverse of Γ, Γ^{-1}, is the covariance matrix which gives the correlation between the atomic movements.

In contrast to GNMs, anisotropic network models (ANMs)[14] take the direction of oscillations into account. Its analog of the Kirchhoff matrix is the Hessian matrix **H**. Each element of the matrix is a submatrix

$$\mathbf{H}_{ij} = \begin{cases} -\frac{\gamma_{ij}}{s_{ij}^2}\begin{pmatrix} x_{ij}^2 & x_{ij}y_{ij} & x_{ij}z_{ij} \\ y_{ij}x_{ij} & y_{ij}^2 & y_{ij}z_{ij} \\ z_{ij}x_{ij} & z_{ij}y_{ij} & z_{ij}^2 \end{pmatrix}, & \text{if } i \neq j \\ \sum_{i,i\neq j} H_{ij}, & \text{if } i = j \end{cases}, \tag{11.5}$$

[6] Hinsen et al. (2000); Yang, Song, and Jernigan (2009).

[7] Miyazawa and Jernigan (1996); Keskin et al. (1998); Hamacher and McCammon (2006).

[8] Dehouck and Mikhailov (2013).

[9] Bahar et al. (2010).

[10] Bahar, Atilgan, and Erman (1997).

[11] Bahar et al. (2010).

[12] a GNM has only a single zero-mode

[13] Bahar et al. (2010).

[14] Atilgan et al. (2001).

where $(xyz)_{ij}$ is the displacement vector between the atoms i and j[15]. The eigenvectors of **H** not only comprise the contribution of an atom to the oscillation, but also resolve the direction. This allows the spatial analysis and visualization of normal modes. Furthermore, it allows the application of linear response theory (LRT)[16]: Within the framework of an ANM, if a force vector $\vec{F}$ is applied on the network model, the equilibrium atom positions are displaced by

$$\Delta\vec{r} = \mathbf{H}^{-1}\vec{F}. \tag{11.6}$$

Note that a '*flattened*' version of **H**, $\vec{F}$ and $\vec{r}$ is used in Equation 11.6, i.e. **H** is a matrix of shape $(3N \times 3N)$ and $\vec{F}$ and $\vec{r}$ are vectors in the form $(x_1, y_1, z_1, x_2, ...)$.

The functionalities available in *Biotite* along with its design upon *NumPy* and its optimized linear algebra operations make the package predestined for application to ENMs. Hence, the presented methodology was implemented in the extension package *Springcraft* to assess global dynamics of a given molecular structure.

11.2 Implementation

The basis for an ENM is the force field that defines the force constants γ for each pair of atoms. In *Springcraft* γ calculation is governed by the abstract `ForceField` superclass. Subclasses of `ForceField` implement specific force fields, including

- `InvariantForceField`, that assigns the same γ to all pairs of atoms within a given cutoff distance,
- `ParameterFreeForceField`, where γ scales continuously with the inverse of the squared distance instead of using a cutoff[17],
- `HinsenForceField`, which uses a higher order polynomial function of the distance to calculate[18] γ,
- `TabulatedForceField`, that selects tabulated γ values based on amino acid type and/or distance[19], and
- `PatchedForceField`, that allows enforcing, severing and overriding chosen contacts of another `ForceField`.

Note that while a `ForceField` may define a cutoff distance, the cutoff is not considered yet at this point of the ENM calculation: The `ForceField` computes γ for given pairs of atoms and their respective distances. How the pairs of atoms within a cutoff distance are identified, is explained in the section below. The separation of `ForceField` from the rest of the ENM logic[20] allows the user to provide a custom implementation for calculating γ by the means of choice.

The `GNM` and `ANM` classes are used to compute GNMs and ANMs, respectively and include basic functionalities for analysis of these. Given the atom coordinates and a chosen `ForceField`, Γ or **H** is calculated, respectively. For this purpose, a cell list is created for the coordinates and

[15] in other words, using the notation from the start of the chapter, $\vec{r}_j^{\,0} - \vec{r}_i^{\,0}$

[16] Ikeguchi et al. (2005).

[17] Yang, Song, and Jernigan (2009).

[18] Hinsen et al. (2000).

[19] Hamacher and McCammon (2006); Dehouck and Mikhailov (2013).

[20] Hessian/Kirchhoff calculation, NMA, etc.

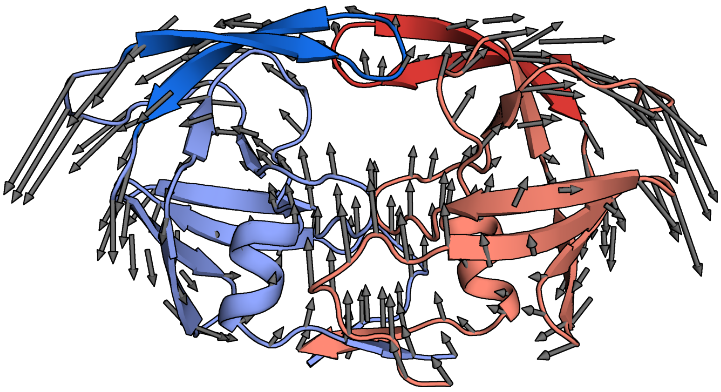


Figure 11.3: *Flap* oscillation of HIV-1 protease. The two HIV-1 protease monomers are shown in red and blue, respectively. The *flaps* are highlighted in darker colors. The normal mode from the ANM showing the experimentally found *flap* movement is depicted by arrows.

an adjacency matrix **A** using the cutoff distance s_c is calculated:

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if } s_{ij} \leq s_c \\ 0 & \text{if } s_{ij} > s_c \end{cases}. \tag{11.7}$$

Those pairs of atoms where $\mathbf{A}_{ij} = 1$ are given to the chosen `ForceField` to calculate γ_{ij} for them. Where $\mathbf{A}_{ij} = 0$, γ_{ij} is 0. If instead no cutoff is given, all possible pairs are considered. Using all γ_{ij}, Γ or **H** are computed according to Equation 11.4 or 11.5, respectively. In more detail, the n pairs of adjacent atoms are represented by indices to the underlying `AtomArray`. The `ForceField` receives the pairs and distances as *NumPy* arrays of shape $(n, 2)$ and (n) respectively. This allows fully vectorized computation of the entire EMM creation.

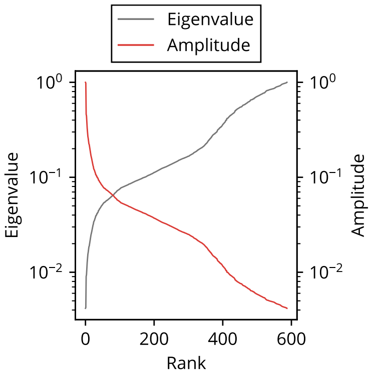


Figure 11.2: Normal modes of HIV-1 protease. Modes corresponding to translation and rotation are omitted. The eigenvalues and oscillation amplitudes were normalized.

For further analysis based on Γ or **H**, respectively, such as covariance computation and NMA, *Springcraft* makes intensive use of the linear algebra operations implemented in *NumPy*. Furthermore, the *NumPy* arrays representing Γ and **H** can be obtained and edited directly, encouraging the user to apply custom analyses on GNM and ANM objects.

Companion source code:
`normal_modes.py`

11.3 Application example

It has been experimentally shown that the HIV-1 protease dimer displays a functionally important movement[21] of its β-hairpins termed '*flaps*'. This movement is also visible as normal mode of an ANM[22]. To reproduce these findings, an ANM with the *extended ANM* force field (`TabulatedForceField.e_anm()`) was computed based on a crystal structure of the dimer[23,24]. The eigenvalues and corresponding oscillation amplitudes are shown in Figure 11.2. The first ten normal modes were visually scanned for a mode that resembles the experimental results, by adding eigenvectors as arrows to the 3D visualization. The third mode showed the expected *flap* oscillation (Figure 11.3).

[21] Hornak et al. (2006).

[22] Hamacher and McCammon (2006).

[23] Hodge et al. (1996).

[24] PDB: `1DMP`

Chapter 12
Hydrogen bond measurement

The implementation of the functionality presented in this chapter was produced in collaboration with a coworker[1].

12.1 Introduction

Hydrogen bonds are non-covalent electrostatic interactions involving three atoms. The hydrogen bond donor D is a relatively strong electronegative atom compared to the hydrogen atom H covalently bound to it[2]. Thus the DH pair forms a dipole with a partially positive charge on H, which attracts a lone electron pair of the acceptor atom A, forming the hydrogen bond (Figure 12.1). Hydrogen bonds play a crucial role in folding and stabilizing the native conformations of biomacromolecules: In proteins they particularly drive the formation of secondary structure elements[3], in nucleic acids they partake in base pairing. Furthermore, they determine selectivity in ligand binding[4].

To detect hydrogen bonds in structure models, geometric criteria can be applied[5]: A hydrogen bond is assumed, if the angle between DHA $\theta \geq 120°$ and HA distance $d \leq 2.5$Å. Furthermore, atoms D and A must be of appropriate chemical elements.

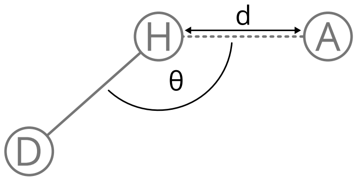


Figure 12.1: Geometry of a hydrogen bond. The involved atoms are shown as circles. The dashed line represents the hydrogen bond.

12.2 Implementation

These criteria are implemented the `hbond()` function. It takes an `AtomArray` or `AtomArrayStack` as input structure model(s) to find hydrogen bonds in. Furthermore, the search can be optionally restricted to a given part of the structure or to certain elements for D and A. By default, oxygen, nitrogen and sulfur are applicable elements.

For all potential donor atoms D selected this way, bound H are sought. If bond information is available, DH pairs are found by creating a dictionary from the `BondList`, that maps each atom to its bound atoms[6] and simply accessing this dictionary for D. If bonds are not available, all H within 1.5 Å distance of D in the same residue are assumed to be bound.

Then all potential A for each DH pair are searched. For this purpose a cell list with cell size d is created containing the coordinates for all identified H from the previous step. For each potential acceptor atom A, all H in the same and all directly adjacent cells are listed. Note that at this point the distance for such an HA pair might still be greater than d (see Section 2.5.1). Finally, the corresponding DH and HA pairs are

[1] Bauer, D., TU Darmstadt

[2] D might be bound to multiple H.

[3] Hubbard and Kamran Haider (2010).

[4] Hubbard and Kamran Haider (2010).

[5] Baker and Hubbard (1984).

[6] Each atom is represented by the index in the `AtomArray` (or `AtomArrayStack`).

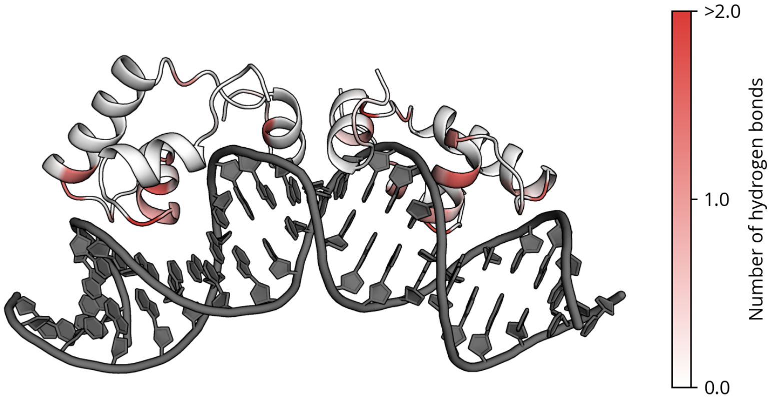


Figure 12.2: Hydrogen bonds between *lac* repressor and O_I operator. The number of hydrogen bonds between a amino acid residue and the bound DNA is shown by color intensity.

combined to DHA triplets in form of a $(k, 3)$-dimensional *NumPy* array, where k is the number of found triplets. For each triplet θ and d is measured and triplets that do not meet the geometric criteria for hydrogen bonds are filtered out. Finally, the triplets are returned. This whole procedure also supports periodic boundary conditions (see Chapter 8).

If an `AtomArrayStack`, i.e. multiple models m, are used as input, the cell list creation and geometric condition check is performed for each model. In addition to the triplets a boolean mask with shape (m, k) is returned, that indicates in which model a given triplet forms a hydrogen bond.

12.3 Application example

Companion source code: `repressor_hbonds.py`

The *lac* operon is one of the most prominent examples for gene regulation in procaryotes. It is a DNA region comprising genes for lactose catabolism, as well as promoter and repressor binding regions upstream of these genes. The *lac* repressor binding regions are called *operators*. The *lac* operon contains three operators in vicinity of the promoter: O_I, O_2 and O_3, showing small sequence variations to each other[7].

Based on the solution NMR structures[8] of the complex between each of the three operators to the DNA binding domain dimer of the *lac* repressor[9], the hydrogen bonds between DNA and protein were analyzedfor this chapter. After fetching and loading the structure models, the hydrogen bonds were measured with `hbond()`. The hydrogen bond search was restricted to bonds between DNA and protein, bonds within the operator or the repressor were ignored. The NMR structures contain multiple models and some hydrogen bonds exist only in part of the models. Hence, the relative frequency of each hydrogen bond was calculated. Cases, where one hydrogen atom has multiple hydrogen bond acceptors according to the geometric conditions, were counted as separate hydrogen bonds. The frequencies for hydrogen bonds belonging to the same amino acid residue were summed.

[7] Lewis (2005).

[8] PDB: `2KEI`, `2KEJ` and `2KEK`

[9] Romanuka et al. (2009).

The location of amino acid residues in the repressor forming hydrogen bonds with the O_1 DNA is shown in Figure 12.2. The majority of hydrogen bonds is located at the second helix of a helix-turn-helix motif, situated in the major groove of the bound DNA. The number of hydrogen bonds between repressor and operator decreases from O_1 to O_3. This trend fits the declining affinity between repressor and operator[10] well (Table 12.1), though presumably other factors, such as differences in contact area, also have an effect.

Table 12.1: Total number of hydrogen bonds between *lac* repressor and operator. The measured K_d from Romanuka *et al.* is listed for comparison.

Operator	# H-bonds	K_d (nM)
O_1	32.4	0.05
O_2	28.6	0.1
O_3	22.9	100

[10] Romanuka et al. (2009).

Chapter 13
Nucleic acid structure analysis

The implementation of the functionality presented in this chapter was produced in collaboration in the course of a Bachelor's thesis[1].

13.1 Introduction

The secondary structure of a nucleic acid molecule can be defined as the entirety of its base pairs. Typically base pairs form between purines (G and A) and pyrimidines (C, T and U). Preferably the *Watson-Crick* pairs AT, AU and GC are observed. These canonical base pairs are composed of hydrogen bonds between atoms at the *Watson-Crick edge* (Figure 13.1). In addition to these widely known base pairing geometries, complex nucleic acid structures allow hydrogen bonds to form also between other atoms in nucleotide residues as well. This gives rise to so called non-canonical base paring geometries that may include the *Hoogsteen* and *Sugar* edge as well[2] (Figure 13.1).

While base pairs determine the secondary structure, the dominating factor for structure stability is π-π stacking between the aromatic rings of consecutive bases. The impact of stacking interactions becomes more clear with a look at the free energy of DNA double helix formation from separate DNA strands: At 15 mM Na^+ and 37° C the contribution of a single stacking interaction between two bases amounts to $\Delta G \approx -4$ kJ/mol to -6 kJ/mol in comparison to the $\Delta G \approx 2.6$ kJ/mol and 0.04 kJ/mol of a canonical AT and GC base pair, respectively[3]. These numbers shows that due to the decreasing entropy of a structured nucleic acid, energy contributions from base pairing alone would not be sufficient to form a stable conformation.

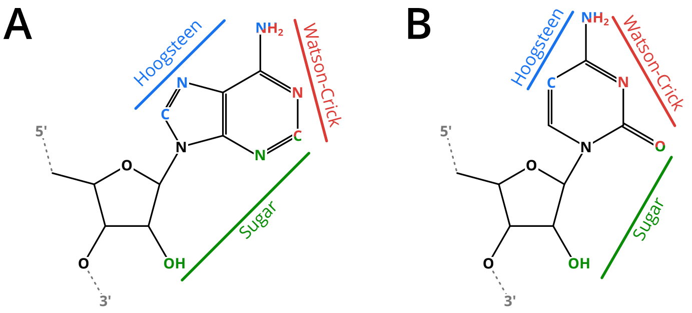


Figure 13.1: Base pair edges. The interaction edges for purines, exemplified by adenosine (**A**), and pyrimidines, exemplified by cytosine (**B**), are outlined. The atoms that are included in an edge are shown in the respective color. Non-polar hydrogen atoms are omitted.

[1] Müller (2021).

[2] Leontis and Westhof (2001).

[3] Yakovchuk, Protozanova, and Frank-Kamenetskii (2006).

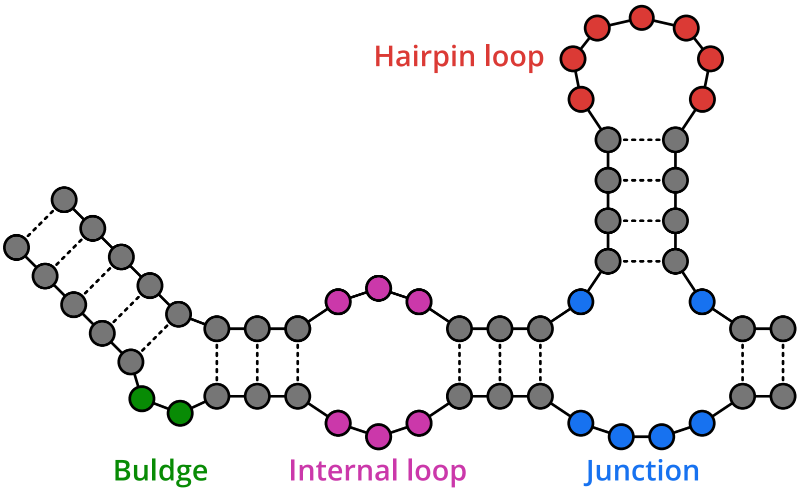


Figure 13.2: Secondary structure elements in nucleic acids. The secondary structure elements are schematically depicted by color. Each circle represents a single nucleotide. The solid lines depict the nucleic acid backbone, dashed lines show base pairs.

13.1.1 Elements in nucleic acid secondary structure

Especially RNA is able to form complex secondary and consequently tertiary structures due to intramolecular base pairs. In consequence, a variety of structure elements emerge[4] (Figure 13.2). Sections of consecutive base parings give rise to a *double helix*, comparable to the typical DNA helix with the difference that RNA creates a helix in *A*- instead of *B*-form. If one strand in such a helix is continuously base-paired while the other strand contains a span of one or multiple unpaired bases, a *bulge* arises in the helix on the side of the additional base(s)[5]. If instead the helix is interrupted by a region where both strands are not paired, the structure element is called an *internal loop*. A loop at the tip of a helix is a *hairpin loop*. An internal loop, where three or more helices meet is termed *junction*.

Usually the majority of base pairs in an RNA structure is nested. To understand what this means, let the *span* of a base pair be the sequence position range from the first to the second base in the pair. The RNA secondary structure is completely nested, if for any combination of two base pairs, one span lies within the other span or the spans do not overlap at all. Base pairs that do not suffice this condition are called pseudoknots (Figure 13.3). Note that the distinction between nested base pairs and pseudoknots is purely arbitrary in terms of physical properties. Often, the largest set of base pairs that is nested within itself is taken as the regular structure, while the remaining base pairs are defined as pseudoknots. The set of pseudoknotted base pairs itself may also not be completely nested, resulting in pseudoknots within the pseudoknots. As such description would quickly become cumbersome, the term of *pseudoknot order* was introduced[6]: The original pseudoknots would have an order of 1, the pseudoknots within pseudoknots would have an order of 2, etc.

[4] Tinoco and Bustamante (1999).

[5] Hermann and Patel (2000).

[6] Antczak et al. (2014).

13.1.2 Geometric measurement

If a structure model of the nucleic acid of interest is available, the task of finding the secondary structure is simplified to measuring whether two proximate bases form a base pair. For this purpose a number of conditions can be stated that two bases need to suffice in order to be recognized as base pair. Depending on base pair definition, these criteria may include the distance between the bases, the relative orientation of the base planes and the presence of hydrogen bonds[7]. Analogously, geometric criteria can also be defined to identify stacking interactions between bases[8].

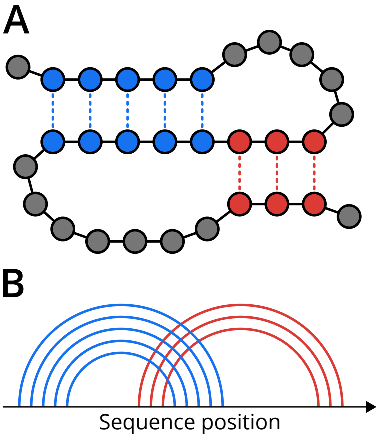


Figure 13.3: Schematic depiction of a pseudoknot. **A** The shown secondary structure comprises two helical regions that are pseudoknotted with each other, highlighted in blue and red, respectively. Usually, the section containing more base pairs (blue) would be defined as the regular structure, while the other region (red) would be termed pseudoknots. **B** The spans show that the helical regions are pseudoknotted. Each arc represents a base pair. While each helical region is nested within itself, the spans between both regions overlap.

13.1.3 Secondary structure prediction

If no tertiary structure information exists, one needs to resort to secondary structure prediction based on sequence. If a larger number of homologous sequences are available, co-evolving sequence positions in an alignment can be used as hint to infer base pairs[9]: Especially if two positions, that could potentially form a canonical base pair, generally mutated in concert into another canonical pair, such positions often form a pair also in reality.

In nucleic acids the major part of folding energy is contributed by the sum of base pairs[10]. This contrasts with proteins where each amino acid may interact with multiple residues and geometric arrangement is a determining factor. This fact is exploited by algorithms that evaluate the free energy merely based on a base pairing graph, without the need to build a 3D structural model of the molecule. Finding the minimum free energy (MFE) structure is reduced to the task of finding a graph where the sum of free energy contributions of base pairs, stacking interactions and loops is minimal[11]. Under the assumption that potential pseudoknots have no impact on the remaining secondary structure, this can be achieved by means of dynamic programming[12], simplifying the hard combinatorial task into a problem solvable in polynomial time.

ViennaRNA[13] is a popular collection of command-line tools that use the dynamic programming principle to predict the secondary structure from RNA sequence. Furthermore, it includes tools for secondary structure comparison and visualization.

13.2 Implementation

13.2.1 Base pair measurement

Biotite provides the `base_pairs()` function to identify base pairs in an `AtomArray`. First all atoms that belong to nucleotides are filtered. To take also non-canonical bases[14] into account, all residues with names that are marked with 'DNA/RNA-linking' in the CCD (see Chap-

[7] Lu, Bussemaker, and Olson (2015).

[8] Gabb et al. (1996).

[9] Gesteland (1998).

[10] Mathews (2006).

[11] Tinoco et al. (1973); Pipas and McMahon (1975).

[12] Zuker and Stiegler (1981).

[13] Lorenz et al. (2011).

[14] other than A, C, G, T, U

ter 7) are included. Pairs of nucleotides that are in vicinity of each other are efficiently retrieved using a cell list. A distance of 3.6 Å was taken as cutoff that must be satisfied by at least one pair of atoms, so that the corresponding nucleotides are considered near to each other. The cutoff was chosen, as at higher distances hydrogen bonds between nucleotides are implausible[15,16].

For all potential base pairs filtered this way, the base pair identification algorithm from the *DSSR* software[17] was adapted. First, it is decided which canonical nucleotide the nucleotide at hand matches best. This is done based on the number of matching atom names and congruence between each canonical nucleotide and the present nucleotide. Second, a standard reference frame of the chosen nucleotide[18] is assigned to each nucleotide in the potential pair: The reference frame defines for each canonical base atom positions lying in the xy-plane that is used later for checking base pairing geometry. The assignment is performed by superimposing the atoms of the reference base onto the nucleotide and applying the same transformation to the principal axes of the corresponding reference frame. Finally, if all of the following criteria are fulfilled, the pair of nucleotides is considered to be an actual base pair:

1. The distance between the origins of the transformed reference frames must be $\leq 15\,\text{Å}$.
2. The vertical separation between the base planes must be $\leq\ 2.5\,\text{Å}$. This is the distance between the reference frame origins projected along the averaged z-axes[19] of both reference frames
3. The angle between the z-axes of both reference frames must be $\leq 65°$.
4. There must not be base stacking between both bases. The details are explained below.
5. At least one hydrogen bond between both nucleotides must be present. If hydrogen atoms are present, hydrogen bonds are identified using the algorithm described in Chapter 12. Otherwise the identification falls back to a simple and potentially inaccurate distance measurement between potential hydrogen bond donor-acceptor pairs.

The functions returns the base pairs as $(n \times 2)$-dimensional *NumPy* array of indices that point to the respective first atom of the paired residues.

If additional information about the interaction edges of the base pairs is desired, the array of base pairs can be fed into `base_pairs_edge()`. For each base pairs and edge it counts the number of hydrogen bonds that includes atoms corresponding to that edge. If an atom corresponds to two edges, the hydrogen bond is counted for both of them. For each

[15] maximum hydrogen bond distance (2.5 Å) + donor-hydrogen distance (1.0 Å) + buffer (0.1 Å)

[16] Baker and Hubbard (1984); Allen et al. (1987).

[17] Lu, Bussemaker, and Olson (2015).

[18] Olson et al. (2001).

[19] The z-axis is equal to the normal vector of the base plane.

base pair the edge with the most hydrogen bonds is selected. Again, an $(n \times 2)$-dimensional array of integers is returned. These integers are members of the `Edge` enumeration type[20].

13.2.2 Base stacking measurement

Finding stacking interactions is similar to finding base pairs. Merely the geometric criteria change[21]:

1. The distance between the aromatic ring centers must be ≤ 15 Å.
2. The angle between the z-axes of both reference frames must be $\leq 23°$.
3. Let $\vec{d}$ denote the displacement vector between any two aromatic ring centers and $\vec{n}$ the z-axis of any reference frame. The angle between any pair of $\vec{d}$ and $\vec{n}$ must not be $> 40°$

As base stacking does not involve hydrogen bonds, the cutoff for the cell list to find nearby nucleotides is substituted with 15 Å. Nucleotides with distances larger than this values cannot meet the first criterion.

13.2.3 Conversion into dot-bracket notation

A popular notation for storing the secondary structure is the so called *dot-bracket* string. Each character in the string corresponds to the base at the respective sequence position. An unpaired base is represented as '`.`'. A base pair is depicted by '`(`' and '`)`' at the position of the first and second base, respectively[22]. In *Biotite* the function `dot_bracket()` generates a dot-bracket string from a $(n \times 2)$-dimensional array of base pairing positions. The major drawback of this notation is that it cannot represent pseudoknots, as they would create situations, where opening brackets cannot be unambiguously assigned to closing brackets. This issue is remedied with the *dot-bracket-letter* notation[23], which uses a unique pair of characters for each pseudoknot order (Table 13.1).

Table 13.1: Dot-bracket-letter notation.

Pseudoknot order	character
0	`()`
1	`[]`
2	`{}`
3	`<>`
4	`Aa`
5	`Bb`
...	

In order for `dot_bracket()` to support this notation, its needs a way to identify pseudoknots. As mentioned earlier, there is no intrinsic property of a base pair that makes it a pseudoknot, but its rather dependent on convention which nested set of base pairs depicts the regular[24] pairs. A typical definition defines the set of regular pairs as the completely nested set, that contains the maximum number of base pairs possible. The function `pseudoknots()` solves this optimization problem via a dynamic programming algorithm[25]. A pseudoknot order of 0 is assigned to the regular base pairs and the method is executed again on the remaining pseudoknotted pairs. This process is repeated until no pseudoknots remain. With each iteration the pseudoknot order is incremented. After the process has finished, `pseudoknots()` returns the pseudoknot order for each base pair. By default, `pseudoknots()`

[20] For example, 1 represents the Watson crick edge.

[21] Gabb et al. (1996).

[22] For example, the string '`(((.(((((....))))))))`' would represent a helix with a bulge and a hairpin loop.

[23] Antczak et al. (2018).

[24] as opposed to 'pseudoknotted'

[25] Smit et al. (2008).

assigns a score of 1 to each base pair, which mirrors the stated goal of maximizing the number of regular base pairs. However, an individual score can be given for each base pair to customize the optimization objective. For example, GC pairs can be weighted more strongly than AT pairs to take the strength of the interaction into account.

Finally, `dot_bracket_from_structure()` takes pairs of atom indices, e.g. output from `base_pairs()`, to map them to sequence positions and creating the dot-bracket-letter string. The conversion from atom indices to dot-bracket notation bridges the gap between nucleic acid tertiary structure analysis in *Biotite* and external secondary structure prediction tools, which typically output dot-bracket strings.

13.2.4 Interfacing ViennaRNA for base pair prediction

For convenience the tools *RNAfold*, *RNAalifold* and *RNAplot* from *ViennaRNA* are directly interfaced as `Application` classes (see Section 2.6) in *Biotite*. *RNAfold* accomplishes the most basic task in RNA secondary structure prediction: the prediction of the MFE secondary structure and its associated free energy from sequence. In contrast, *RNAalifold* informs the structure prediction with an additional artificial energy term based on co-evolving sequence positions from an alignment[26]. The added value of the `RNAfoldApp` and `RNAalifoldApp` interfaces in *Biotite* is the handling of file input/output and the conversion of the resulting dot-bracket string into positions in the input sequence.

13.2.5 Secondary structure visualization

The interface to *RNAplot* (`RNAplotApp`) converts the output coordinates of each base in the 2D secondary structure plots into a *NumPy* array. Although *RNAplot* itself is able to directly output secondary structure plots as image file, there are little customization capabilities. Therefore, the interface class `RNAplotApp` does not use *RNAplot* to create a secondary structure image, but instead to output the coordinates of each base in the hypothetical 2D plot instead. The coordinates are read and converted into a *NumPy* array. The obtained coordinates from `RNAplotApp` are used in in the function `plot_nucleotide_secondary_structure()` to create a highly adjustable secondary structure plot with *Matplotlib*. For instance, in contrast to the image output of *RNAplot*, custom symbols for a base and coloring for these symbols as well as for the lines depicting pairing can be chosen. In addition, the sequence position of bases can be annotated and pseudoknots can be depicted, which is not possible in the image output of *RNAplot*.

13.3 Application example

Companion source code: `trna_structure.py`

The prediction accuracy of the native free energy minimization algorithm of *RNAfold* was compared to the alignment-assisted method of *RNAalifold*. As test case the structure of the *E. coli* Asp-tRNA[27] was

[26] Hofacker, Fekete, and Stadler (2002).

[27] PDB: 6UGG

analyzed. The base pairs in the molecular model were taken as reference for the later comparison. The base pairs were identified in the structure with `base_pairs()`. Only canonical base pairing geometries should be considered here, so the interacting edges were analyzed with `base_pairs_edge()` and only bases with Watson-Crick/Watson-Crick edge interaction were filtered. Since the dynamic programming algorithm employed by *RNAfold* and *RNAalifold* is not able to identify pseudoknots, such base pairs were also removed to ensure fair comparison. The pseudoknot order for each base in the sequence was computed with `pseudoknots()` and base pairs with pseudoknot order greater than 0 were filtered out.

To generate the input MSA for *RNAalifold* a dataset of 1110 tRNA sequences were downloaded from the *T-psi-C* tRNA sequence database[28]. The sequences of tRNAs carrying aspartate were filtered, resulting in a final dataset of 32 sequences. The sequences were aligned with *Muscle*[29]. *RNAfold* and *RNAalifold* were run using the `RNAfoldApp` and `RNAalifoldApp` interfaces.

Table 13.2: Prediction accuracy.

	Sensitivity	PPV
RNAfold	30.4 %	28.0 %
RNAalifold	91.3 %	100.0 %

To assess the prediction accuracy, the sensitivity and positive predicted value (PPV) were calculated based on the set of known base pairs measured in the X-ray structure (K) and the set of predicted base pairs (P)[30]:

$$\begin{aligned}\text{Sensitivity} &= \frac{|P \cap K|}{|K|}, \\ \text{PPV} &= \frac{|P \cap K|}{|P|}.\end{aligned} \tag{13.1}$$

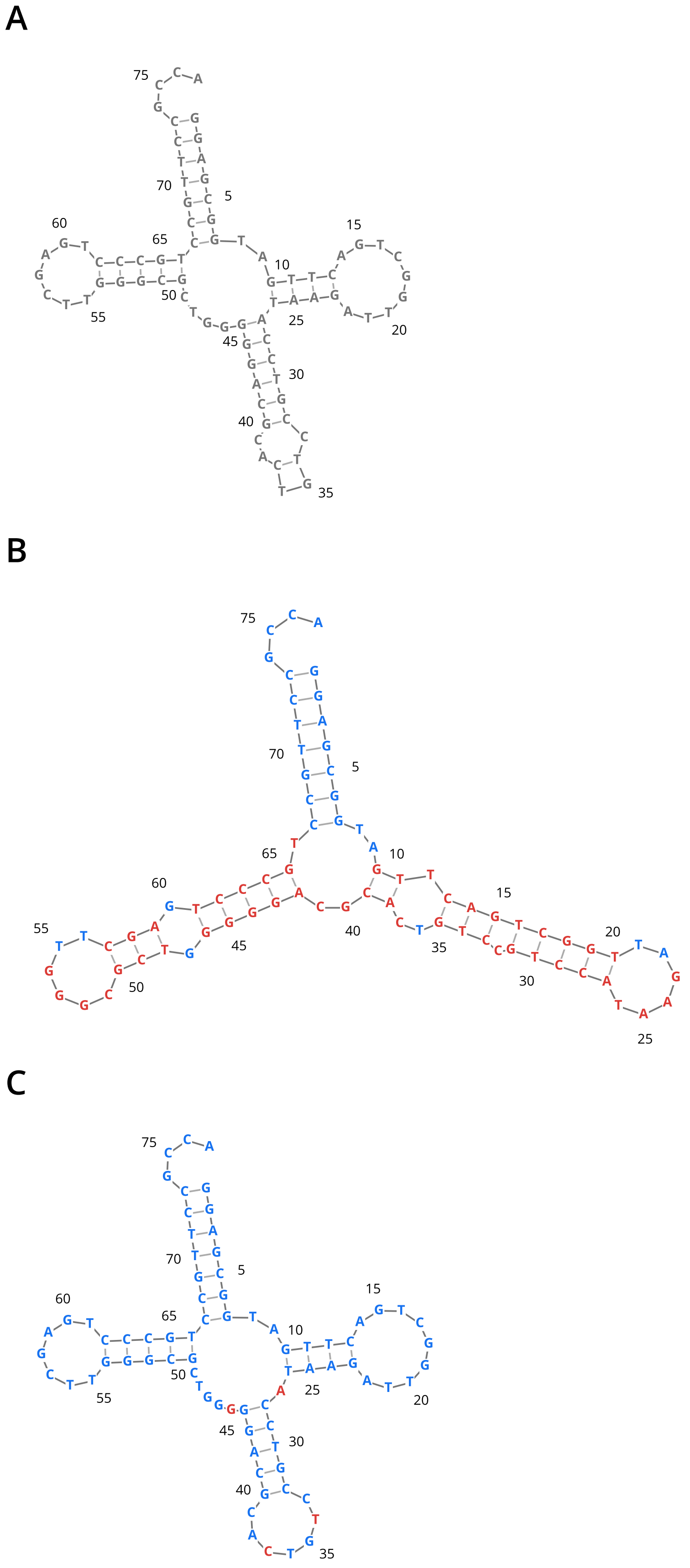
The results are shown in Table 13.2. Both the sensitivity and PPV are considerably higher, when the coevolution information from the alignment is added to the structure prediction. The disparity is more clearly localized with a closer look at the secondary structure created with `plot_nucleotide_secondary_structure()` and shown in Figure 13.4. Without the evolutionary information *RNAfold* does not find the typical cloverleaf conformation of the tRNA, but instead combines three of the known hairpins into two hairpins (Figure 13.4B). *RNAalifold* identified almost all base pairs correctly, with the exception of one pair at the central junction and one pair at one of the hairpin loops (Figure 13.4C).

[28] Sajek et al. (2020).

[29] Edgar (2004).

[30] Mathews (2006).

Figure 13.4: Predicted *E. coli* Asp-tRNA secondary structure. The predicted base pairs are compared with the reference base pairs measured in the corresponding X-ray crystal structure. Correctly predicted pairs are shown in blue, incorrect ones in red. **A** Reference secondary structure measured in crystal structure. **B** Prediction from sequence with *RNAfold*. **C** Prediction from alignment of multiple Asp-tRNA sequences with *RNAalifold*.



Chapter 14
Hydrogen addition

This chapter builds upon ideas and figures from a published journal article[1]. The content is licensed under the CC BY 4.0 license. Changes were made on size and arrangement of figure elements.

14.1 Introduction

Some of the aforementioned methods, like the measurement of hydrogen bonds or the accurate identification of base pairs, and many other analysis and simulation applications require complete structural models including hydrogen atoms. However, due to their small size and electron density especially hydrogen atoms elude the structural investigation via crystallographic methods or electron microscopy. In consequence, only $\sim 16\,\%$ of structure models deposited in the PDB have annotated hydrogen atoms. Furthermore, even some MD simulation and docking tools do not include hydrogen atoms in their output files[2].

To mitigate this problem most MD simulation packages contain programs that add hydrogen atoms to structures as preparation for the downstream simulations. These include *pdb2gmx* from *Gromacs* and *HBUILD* from *CHARMM*[3]. Furthermore, also standalone programs like *REDUCE*, *OpenBabel* or HAAD[4] exist for solving this problem. However, most of these programs use molecule-specific force fields to predict hydrogen positions. This characteristic limits the application of these programs to structures containing exclusively these parametrized molecules. An exception is *OpenBabel*[5], though it focuses on small molecules.

Therefore an alternative approach is presented in this work: Instead of requiring complex force fields for each type of molecule, the positions of hydrogen atoms are inferred from a library of molecular fragments compiled from a large catalog of reference molecules. Hence, this algorithm is able to accurately predict hydrogen positions for most organic compounds including small molecules and large biomacromolecules. This algorithm is bundled in the *Biotite* extension package *Hydride*.

14.2 Methods

The concept of the algorithm is that hydrogen positions from complete[6] molecular models, called *reference molecules*, are used to predict the hydrogen positions for a *target molecule*, where the model misses hydrogen atoms. Nevertheless, the algorithm expects that the heavy atoms[7] are accurately positioned in the target molecule. The definition of target

[1] **Kunzmann**, Anter, and Hamacher (2022).

[2] Trott and Olson (2010); Brooks et al. (1983); Webb and Sali (2016).

[3] Lindahl et al. (2021); Brooks et al. (2009).

[4] Word et al. (1999); O'Boyle et al. (2011); Li, Roy, and Zhang (2009).

[5] O'Boyle et al. (2011).

[6] containing all atoms, including hydrogen

[7] all atoms except hydrogen

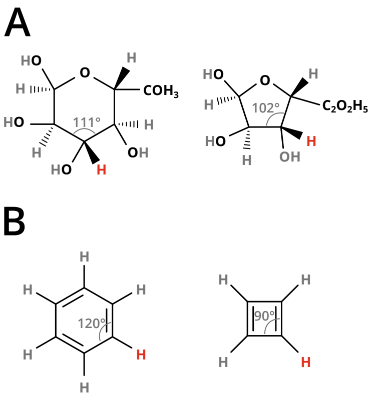


Figure 14.1: Pairs of cyclic molecules containing a fragments with different geometry but equal library keys. Each pair has the same library key for the fragment containing the hydrogen atom depicted in red. However, the shown bond angles are different. **A** α-D-Glucopyranose and α-D-Glucofuranose. **B** Benzene and Cyclobutadiene. Adapted from Kunzmann, Anter and Hamacher (2022) (CC BY 4.0).

molecule is not limited to a single molecule, but extends to structural models containing multiple molecules.

14.2.1 Hydrogen addition

First, each molecular model from a catalog of *reference molecules* is fragmented (Figure 14.2A): One *fragment* is created for each heavy atom in the molecule. A fragment contains information about

- the position, element and formal charge of the respective heavy atom,
- the position of covalently bonded hydrogen atoms,
- the position and connecting bond order of covalently bonded heavy atoms and
- the enantiomer, if the respective heavy atom is a stereocenter.

The fragments of the reference molecules are stored in a *fragment library*, which maps the combined element, charge, bond order and stereocenter information of a fragment (the *library key*) to its position of bonded heavy and hydrogen atoms. The fragments stored in the library are referred to as *library fragments*. Note that the position of the central heavy atom is not stored in the fragment library, as the positions of the atoms in a library fragment are translated to place its central heavy atom in the coordinate origin. Library keys are unambiguous: If multiple fragments have the same key, only one of these fragments is added to the library, since both fragments ideally convey sufficiently similar geometric information. A case where two fragments with equal library keys display different geometries are cyclic compounds (Figure 14.1). However, the geometric differences do not considerably affect the accuracy of hydrogen positioning (see Section 14.3.2) to justify an additional computationally expensive step to select the library fragment with the optimal heavy atom geometry. Nitrogen as the central atom poses an additional challenge: Using its lone electron pair nitrogen is able to form a partial double bond to atoms, that are formally bonded with a single bond. However, the partial double bond induces a planar instead of a tetragonal conformation. To distinguish fragments with tetragonal geometry from the ones with planar geometry, partial double bonds are depicted by an additional bond order. If nitrogen forms a single bond to a heavy atom, that itself forms a double bond to another heavy atom, that single bond is considered as partial double bond.

The target molecule is fragmented in the same way, with the exception that the resulting fragments do not contain hydrogen atoms (Figure 14.2B). Then, for each of these fragments (called *target fragments*) the library key can be computed, as library keys do not require information about hydrogen atoms (see above). The fragment library is accessed with these library keys to obtain the matching library fragment for each of the target fragments. If no entry is found the library for a given key, the hydrogen positions cannot be computed for the respective heavy atom. Hence, this algorithm relies on a comprehensive frag-

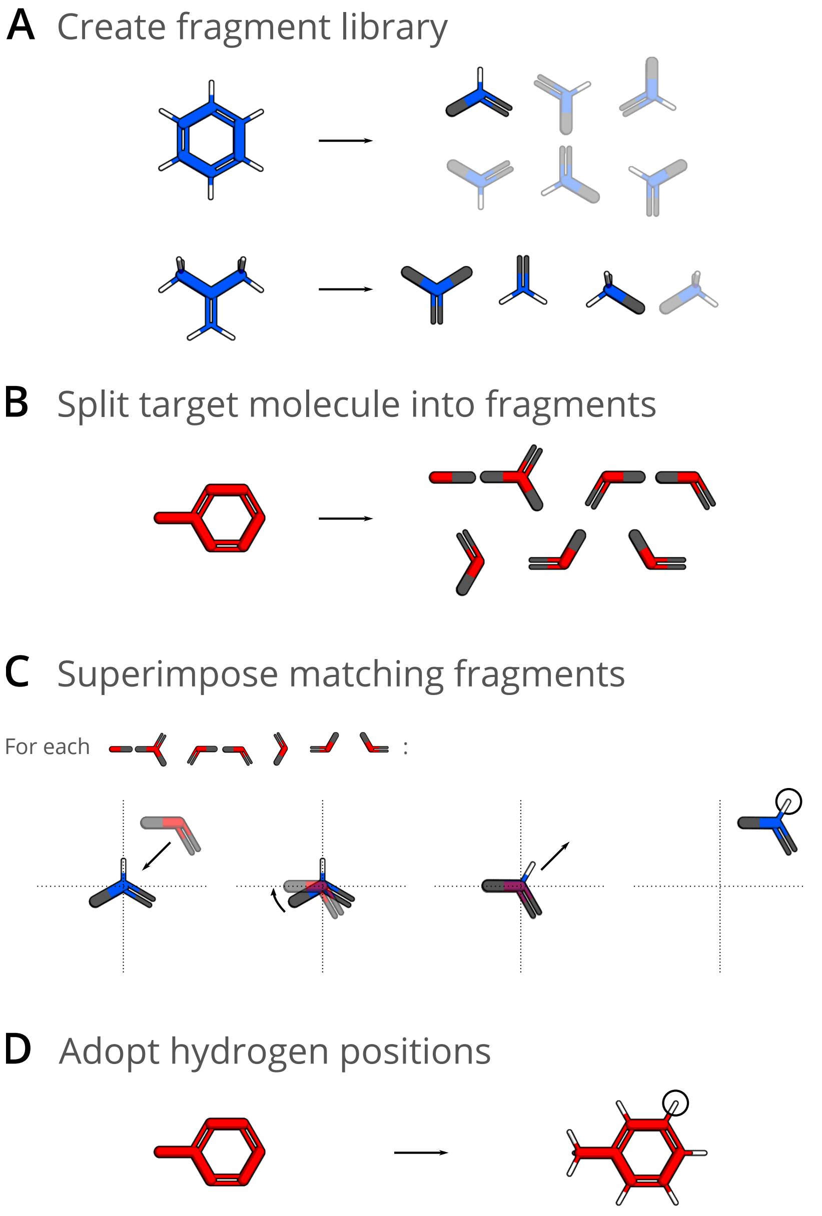

A Create fragment library
B Split target molecule into fragments
C Superimpose matching fragments
For each
:
D Adopt hydrogen positions

Figure 14.2: Summary of the hydrogen addition algorithm. The figure exemplarily depicts how hydrogen atoms are added to the target toluene (red) from geometric information of the reference molecules benzene and isobutylene (blue). **A** The reference molecules are fragmented. Each fragment comprises the respective central atom (blue), bonded heavy atoms (gray) and bonded hydrogen atoms (white). The fragments are stored in the fragment library. Fragments with duplicate library keys are ignored (transparent). **B** The target molecule is fragmented. The central atom is shown in red, hydrogen atoms are missing. **C** The matching library fragment is selected for each target fragment and superimposed onto it. The position of the target fragment's central heavy atom is translated into the coordinate origin, the library fragment is rotated to achieve maximum congruence and the library fragment is translated to the original position of the target fragment. The resulting hydrogen coordinates of the transformed library fragment (encircled) are the desired hydrogen coordinates for the target fragment. **D** The calculated hydrogen positions for each target fragment are adopted for the target molecule. Adapted from Kunzmann, Anter and Hamacher (2022) (CC BY 4.0).

ment library that is created from a wide range of reference molecules.

For each pair of target and library fragment, the library fragment is superimposed onto the target fragment (Figure 14.2C): The target fragment is translated to move its central heavy atom into the coordinate origin[8] and then the library fragment is rotated to achieve optimum congruence with the target fragment, i.e. the RMSD of the corresponding atoms is minimized[9]. Finally, the library fragment is translated to the original position of the target fragment by subtracting the same translation vector from the position of the library fragment. The resulting hydrogen position(s) of the transformed library fragment are the required hydrogen coordinates for the target molecule (Figure 14.2D).

[8] The library fragment is already located in the origin.

[9] Kabsch (1976); Kabsch (1978).

14.2.2 Hydrogen relaxation

The described algorithm should accurately place hydrogen atoms with respect to bond lengths and angles. For most heavy atoms no rotational freedom exists, since they are constrained by bonded heavy atoms, whose positions are considered constant in the scope of this algorithm. However, terminal groups that are connected to the rest of the molecule with a single bond[10] still have rotational freedom. Thus, their hydrogen positions are ambiguous and need to be optimized, with respect to non-bonded interactions with nearby atoms.

The non-bonded interactions, i.e. the *Lennard-Jones* (V_{LJ}) and electrostatic (V_{el}) potentials, are modeled using the *Universal force field* (UFF)[11]. The energy function for the interaction of two atoms i and j is

$$\begin{aligned} V_(\vec{r}_i, \vec{r}_j) &= V_{\text{el}}(\vec{r}_i, \vec{r}_j) + V_{\text{LJ}}(\vec{r}_i, \vec{r}_j) \\ V_{\text{el}}(\vec{r}_i, \vec{r}_j) &= 332.064 \frac{q_i q_j}{|\vec{r}_j - \vec{r}_i|} \\ V_{\text{LJ}}(\vec{r}_i, \vec{r}_j) &= \epsilon_{ij} \left(\left(\frac{\delta_{ij}}{|\vec{r}_j - \vec{r}_i|} \right)^{12} - 2 \left(\frac{\delta_{ij}}{|\vec{r}_j - \vec{r}_i|} \right)^{6} \right), \end{aligned} \tag{14.1}$$

where $\vec{r}_i$ gives the position vector and q_i gives the partial charge of the respective atom i [12]. ϵ_{ij} is the well depth and δ_{ij} gives the optimal distance. They are computed as the geometric and arithmetic mean from the parameters ϵ_i and δ_j, taken from the UFF:

$$\begin{aligned} \epsilon_{ij} &= \sqrt{\epsilon_i \epsilon_j}, \\ \delta_{ij} &= \frac{\delta_i + \delta_j}{2}. \end{aligned} \tag{14.2}$$

If i and j are a potential hydrogen bond donor-acceptor pair, δ_j is multiplied with 0.79 to better reproduce physical distances in hydrogen bonds[13]. To reduce computation time, interactions are only calculated for pairs of atoms within a predefined cutoff distance[14]. Furthermore, only interactions that involve at least one hydrogen atom are considered, as distances between heavy atoms are constant during the course of the energy minimization.

Using this energy function new conformations of the molecular model can be assessed. Let ϕ_k be the torsion angle of the rotatable bond connected to the terminal heavy atom k. ϕ_k determines the positions $\vec{r}_p...\vec{r}_q$ of hydrogen atoms bonded to k. In each iteration of the relaxation algorithm each ϕ is altered by an angle increment to obtain the updated angle $\phi^*_k = \phi + \Delta\phi$ and hydrogen positions $\vec{r}^*_p...\vec{r}^*_q$. Special considerations need to be taken for imine groups, since two diastereomers exist with respect to the hydrogen position, but free rotation is not possible. Hence $\Delta\phi = 180°$ for imine groups and alternately $\Delta\phi = \pm 10°$ for other heavy atoms[15]. For each k, the energy difference with respect to ϕ_k, ΔV^*, is calculated with

$$\Delta V^*(k) = \sum_{a=p}^{q} \sum_{b}^{\text{all}} \left[V(\vec{r}^*_a, \vec{r}_b) - V(\vec{r}_a, \vec{r}_b) \right]. \tag{14.3}$$

[10] for example methyl or hydroxy groups

[11] Rappe et al. (1992).

[12] The factor 332.064 includes $\frac{N_{\text{A}}}{e^2/4\pi\,\epsilon_0}$ and the conversion factors from J→kcal and m→Å

[13] Ogawa and Nakano (2010).

[14] 10 Å by default

[15] This is the default value. It can be decreased to get more precise results at the cost of computation speed.

In prose, the angle update of a single heavy atom k is treated as isolated, i.e. only the position update for hydrogen atoms bonded to k is applied. Then the energy difference is calculated for all interactions, where these hydrogen atoms are involved. If the new conformation for hydrogen atoms bonded to k is more energetically preferable, i.e. $\Delta V^*(k) < 0$, ϕ_k^* is accepted for the next iteration ($\phi_k \leftarrow \phi_k^*$). Otherwise ϕ_k^* is rejected and the original ϕ_k is used again. If $\Delta V^*(k) \geq 0$ for each k in two subsequent iterations, the relaxation terminates - a local minimum has been reached.

14.2.3 Charge calculation

The partial charges required for V_{el} are calculated using the PEOE method[16] (see Chapter 9). Both, the partial charge calculation and selection of library fragments, require physiological formal charges for all heavy atoms in the molecular model. Otherwise, the conformation will not be correctly relaxed and, even worse, wrong protonation states will be assumed. Unfortunately, a large number of structure models in the PDB provides the formal charge for acidic and basic groups as neutral. To circumvent this issue, *Hydride* optionally calculates the expected formal charge for heavy atoms in amino acids based on tabulated pK_a values[17] and a user-provided pH value.

14.2.4 Atom order and naming

Although the number and position of hydrogen atoms is a paramount information, a structural model in context of an `AtomArray` or a structure file requires also names for the added hydrogen atoms and a reasonable atom ordering. *Hydride* places hydrogen atoms behind the heavy atoms of the respective residue in the order of the heavy atoms within the residue. For atom naming *Hydride* uses an *atom name library*: If a residue is part of the library, it provides canonical names for added hydrogen atoms. By default, the library contains atom names for all proteinogenic amino acids and canonical nucleotides. If a residue is not part of this library, unambiguous hydrogen atom names are generated based on the name of the bonded heavy atom.

14.2.5 Implementation details

In accordance with *Biotite*, easy usage, flexibility and performance were key points in the design of the implementation. Thus, *Hydride* provides both, a simple CLI for standard use cases and a more flexible Python API that is embedded into the *Biotite* environment - molecular structures are represented by `AtomArray` objects and *NumPy* arrays are used as container for numeric values. Where possible, functionalities from *Biotite* where reused, ranging from file input and output to geometric measurements and partial charge calculations.

To optimize the performance of the algorithm, most computation time

[16] Gasteiger and Marsili (1980).

[17] Lide (2003).

bottlenecks are either computed vectorized or implemented in *Cython*. To find atom pairs within cutoff distance for the hydrogen relaxation, a cell list is employed.

By default, *Hydride* uses all molecules from the CCD[18] (See Chapter 7) to compile the fragment library, though the user can decide to use another catalog of molecules, such as *PubChem*[19]. This ensures that hydrogen addition is supported for at least all structures deposited in the PDB. However, since the molecules in the CCD comprise a wide range of different organic moieties, hydrogen positions can be predicted for almost any organic target molecule. In edge cases, where the fragment library does not contain a fitting fragment for a given heavy atom, a complete structural model of the respective molecule can be added to the fragment library to solve this issue. As mentioned in Section 14.2.1, duplicate library keys are ignored. Practically, this means that only the last added fragment for the same key is stored in the library.

In the Python API, hydrogen addition and relaxation are bundled by the separate functions `add_hydrogen()` and `relax_hydrogen()`: If energy-minimized conformations are not necessary for the use case, relaxation can be omitted to notably save computation time. Alternatively, if a complete structural model already exists, only a relaxation needs to be conducted. Furthermore, individual heavy atoms, whose hydrogen positions should be predicted, can be selected. This enables for example the application of *Hydride* to partially complete structures.

Since, also the atom charges can be supplied separately, other methods to calculate these can be integrated: For example charges considering the spatial arrangement of amino acids could be computed using *PROPKA*[20]. The built-in formal charge calculation is called via `estimate_amino_acid_charges()`.

14.3 Results and discussion

14.3.1 Prediction accuracy

The presented hydrogen addition algorithm was evaluated on three structure datasets for proteins, nucleic acids and small molecules. The structure models in the datasets are complete and contain hydrogen atoms. For the validation the hydrogen atoms in the structures in each dataset were removed. Then hydrogen atoms were assigned to these structures with the *Hydride* CLI using default values[21]. The distances between the assigned and original hydrogen positions were measured and classified based on whether the hydrogen atom was fixed[22] or part of rotatable polar or nonpolar groups (Figure 14.3). The RMSD of these distances was taken as indicator for the prediction accuracy (Table 14.1).

For the protein dataset, the same structures were selected from the PDB as in the *HAAD* study[23] to enable direct comparison with *HBUILD*, *RE-*

[18] Westbrook et al. (2015).

[19] Kim et al. (2021).

[20] Olsson et al. (2011).

[21] including the default fragment library based on reference molecules from the CCD

[22] no relaxation required due to constrained rotational freedom

[23] Li, Roy, and Zhang (2009).

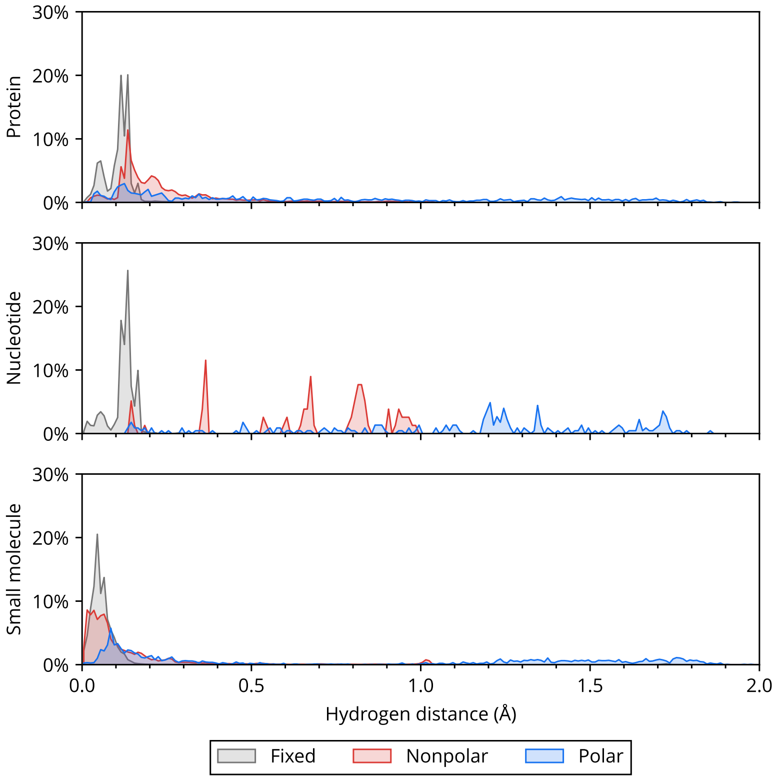


Figure 14.3: Accuracy of predicted hydrogen positions. The figure shows a histogram of distances between the prediction and reference hydrogen atom position for each dataset and class. All histograms are normed and do not reflect relative frequencies when compared to each other. Adapted from Kunzmann, Anter and Hamacher (2022) (CC BY 4.0).

DUCE and *HAAD*. *Hydride* achieved an average RMSD = 0.25 Å, similar to RMSD = 0.28 Å, 0.23 Å, 0.21 Å in the compared programs, respectively. The slightly lower accuracy compared to *REDUCE* and *HAAD* may be caused by the decision for a element-specific in contrast to a molecule-specific force field. However, the usage of UFF is paramount to generalize the hydrogen assignment to all organic molecules.

To test *Hydride* on nucleic acids, all pure nucleic acid X-ray structures with a resolution $\leq$ 1.0 Å were fetched from the PDB, resulting in 29 structures. While for fixed hydrogen atoms the accuracy is close to the value for protein structures (RMSD = 0.13 Å), the distances are noticeably increased for freely rotatable groups: For nonpolar and polar hydrogen atoms an RMSD = 0.71 Å and RMSD = 1.20 Å was measured, respectively. In nucleic acids, the nonpolar hydrogen atoms are located in the C7-methyl groups of thymine. Here two distinctive rotamers are possible: One hydrogen atom in the methyl group orients either toward the C4 or the C6 atom. In the evaluated crystal structures one of the hydrogen atoms in each group orients towards the C4 atom, whereas

	fixed	nonpolar	polar	total
protein	0.12	0.30	0.90	0.25
nucleotide	0.13	0.71	1.20	0.31
small molecule	0.13	0.24	1.03	0.21

Table 14.1: RMSD of predicted hydrogen positions. The RMSD in Å between the predicted position and the position in the reference structure was measured for all hydrogen atoms in the respective dataset and class.

quantum mechanics calculations favor the orientation toward the C6 atom with an energy difference of[24] 1.1 kcal / mol. However, due to the rather low barrier both rotamers are physically plausible. The relaxation with *Hydride* creates different conformations based on the environment in vicinity. Polar rotatable hydrogen atoms are nearly exclusively bonded to O2′ atoms. In crystallographic studies it commonly orients towards the O3′ or O4′ atom of the same residue or the O4′ atom of the successive nucleotide, while the relaxation only yields the latter orientation. Overall, the freely rotatable groups make up merely 6.7 % of hydrogen atoms in the evaluated nucleic acid molecules. Hence, the total RMSD = 0.31 Å.

To test the ability of *Hydride* to assign hydrogen atoms to molecules that are not part of the fragment library, a randomized dataset of 5000 small molecules was downloaded from the *PubChem* database[25]. Matching fragments were found for 99.98 % of hydrogen atoms in this dataset. Consequently, *Hydride* was able to successfully assign all hydrogen atoms to 99.8 % of the tested molecules. For hydrogen atoms in the 'fixed' (RMSD = 0.13 Å) and 'nonpolar' (RMSD = 0.24 Å) class the accuracy is similar to the measured values from the protein dataset. However polar hydrogen atoms show a considerably higher deviation (RMSD = 1.03 Å). A possible explanation for this discrepancy are the missing ambient atoms in the small molecule dataset: In the protein dataset polar groups have electrostatic interactions with nearby polar atoms from other residues, restraining the rotatable bond to a favorable conformation. In order to substantiate this hypothesis we compared the hydrogen assignment accuracy for polar hydrogen atoms in free α-D-glucopyranose and in a α-cyclodextrin-receptor complex[26] (PDB: 5MTU). In the bound form the accuracy increased to RMSD = 1.15 Å from RMSD = 1.35 Å in the free state, verifying the assumption.

14.3.2 Fragment compatibility

As mentioned in Section 14.2.1, the fragment library does not distinguish between different molecular geometries for the same library key. However, especially cyclic compounds may exhibit different geometries based on the ring size. To assess whether this circumstance poses an issue for accurate hydrogen positioning, two pairs of cyclic molecules were tested:

- α-D-glucopyranose and α-D-glucofuranose and
- benzene and cyclobutadiene

(Figure 14.1). The molecular models were taken from *PubChem*. For each pair, a fragment library from one molecule was compiled and used to predict the hydrogen positions for the respective other molecule and vice versa. This way a deviation of 0.025 Å and 0.070 Å to the original position was achieved for the C3 atom in the glucose isomers. For the planar benzene and cyclobutadiene molecules the distance was 0.006 Å in both predictions. Since the mean amplitude of molecular vibration

[24] Mastryukov, Fan, and Boggs (1995).

[25] Kim et al. (2021).

[26] α-cyclodextrin comprises six α-D-glucopyranose monomers.

in a C-H bond[27] (≈ 0.08 Å) is larger than the observed deviations, these inaccuracies can be neglected.

14.3.3 Application example

Companion source code:
`cytochrome_hbonds.py`

Usage of *Hydride* is demonstrated on the electron transporter in the mitochondrial intermembrane space cytochrome C (PDB: `1HRC`)[28] shown in Figure 14.4. Cytochrome C contains heme C as prosthetic group which is connected to the apoprotein via two thioester bonds to cysteine residues. This section examines the hydrogen bonds between the apoprotein and the heme group in the structural model. Since the structure was resolved at 1.90 Å, no hydrogen atoms are annotated in the model. Hence, *Hydride*'s *Python* API is used first to add hydrogen atoms to the structure to enable subsequent analysis of hydrogen bonds.

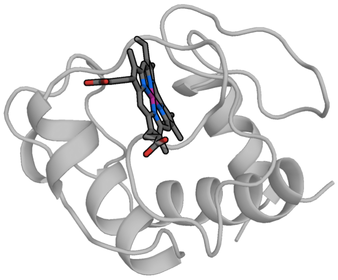


Figure 14.4: Overview of the cytochrome C structure. The apoprotein is shown in transparent. Heme is shown in dark gray.

First, the structure was fetched from PDB and loaded in MMTF format. Then the formal charges were recalculated, as the structural model does not provide physiological charges. The charges of the protein residues were computed with `estimate_amino_acid_charges()` at pH = 6.9 of the intermembrane space[29], with special handling of His18, that is involved in Fe^{2+} complexation. Furthermore, the charge of the heme carboxy groups was adjusted to represent a deprotonated state. Then hydrogen atoms were added with `add_hydrogen()` and their positions were relaxed with `relax_hydrogen()`, using default parameters. `hbond()` (see Chapter 12) was used to measure the hydrogen bonds given in Table 14.2. The positions of these hydrogen bonds is shown in Figure 14.5.

Table 14.2: Predicted hydrogen bonds between cytochrome C apoprotein and heme. The table gives the amino acid and heavy atom involved in each measured hydrogen bond. 'N' refers to the backbone nitrogen atom.

residue	atom
G41	N
Y48	O_η
T49	N, O_γ
N52	N_δ
W59	N_ε
K79	N

14.4 Conclusion

Complete molecular models are required for a wide range of structural analyses. For some functionalities in *Biotite* complete structural models are mandatory, which limits the analysis capabilities for a high percentage of structures. *Hydride* closes this gap: Now any `AtomArray` without hydrogen atoms can be converted into an `AtomArray` with hydrogen atoms in a simple way. On the other side *Hydride* exemplifies how *Biotite* can be used as programming library to simplify the development of more specialized software: A large portion of required functionalities where not specifically implemented for *Hydride* but simply reused from *Biotite*. As *Biotite* supports multiple structure file formats, this flexibility is passed on to *Hydride*, which allows structure input and output in the CLI in most commonly used file formats. This is a significant advantage compared to most of the other presented hydrogen addition programs[30], which only support either the deprecated *PDB* format[31] or software specific formats. The specialization of specific types of molecules is another important limitation of these programs. Even though for most structures simply the appropriate program can be chosen for hydrogen addition[32], the procedure becomes nontrivial for certain mixed molecular models, e.g. a protein-ligand

[27] Bartell, Kuchitsu, and deNeui (1961); Cyvin et al. (1979); Tanimoto, Kuchitsu, and Morino (1971).

[28] Bushnell, Louie, and Brayer (1990).

[29] Porcelli et al. (2005).

[30] except *OpenBabel*, which also supports a wide range of structure file formats

[31] Adams et al. (2019).

[32] i.e. the program whose force fields supports all residues in said structure

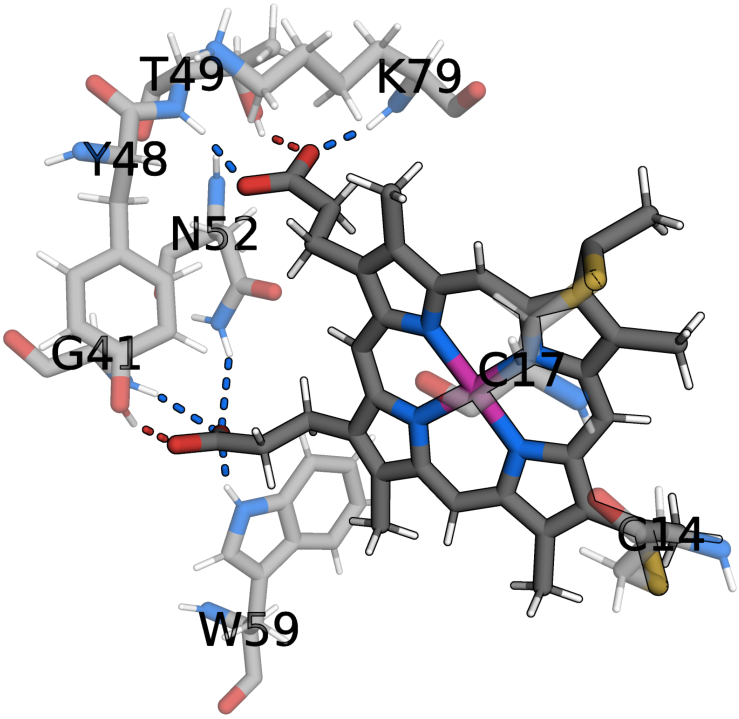


Figure 14.5: Predicted hydrogens bonds between cytochrome C apoprotein and heme. The atoms are colored as following: Carbon in gray, nitrogen in blue, oxygen in red, iron in violet, hydrogen in blue. The relevant amino acid residues of the apoprotein are shown in transparent. The measured hydrogen bonds are shown as dashed lines in the color of the hydrogen bond donor. The labels give the amino acids that either form hydrogen or thioether bonds to the prosthetic group.

complex, since none of these programs is able to assign hydrogen atoms to both, the protein and the ligand, and to take the interaction between each other into account. As a tradeoff, *Hydride* achieves a slightly lower accuracy, but offers a straightforward usage - also for cases where other software is not applicable.

Part III

Application on biological questions

In the following chapters, *Biotite* and its extension packages are used to tackle current biological questions. In contrast to the application examples of the previous chapters, which demonstrated the capabilities of *Biotite*, but otherwise showed well-known results, here two topics are thoroughly analyzed and novel insights are presented.

The source code for the data analysis workflow used for these chapters is available as an archive[33]. The relevant source code resides in the `networks` and `aptamers` directory, respectively.

[33] **Kunzmann** (2022).

Chapter 15
Ligand induced conformational changes in the HCN4 channel

15.1 Introduction

15.1.1 Hyperpolarization-activated cyclic nucleotide-gated channels

Hyperpolarization-activated cyclic nucleotide-gated (HCN) channels are a family of tetrameric transmembrane protein complexes that allow passive flow of cations through the plasma membrane. The channels are selective for monovalent cations, primarily K^+ and Na^+ at a ratio of approximately 4:1[1]. In contrast to most other cation channels, that open upon depolarization of the membrane potential, HCN channels open at hyperpolarization voltage at about[2] -70 mV to -100 mV. Although HCN channels are weakly selective for K^+ ions, under physiological conditions Na^+ influx dominates, depolarizing the membrane. The consequence are periodically firing action potentials as required by a number of rhythmic processes from sleep-wake cycle to the heart beat[3].

HCN channels are allosterically regulated by cyclic adenosine monophosphate (cAMP). Upon cAMP binding half activation voltage is shifted by about[4] 10 mV to 25 mV, increasing the open probability of the channel. By regulating the activity of adenylyl cyclase responsible for cAMP formation, the frequency of action potential firing can be decreased (lowered cAMP concentrations) or increased (raised cAMP concentration)[5]. Patch-clamp fluorometry experiments suggest that cAMP binding not only allosterically regulates channel opening, but also influences the affinity of the remaining cAMP binding sites in the tetramer, resulting in a cooperative binding behavior[6]. However, whether ligand binding cooperativity truly exists in HCN channels has been challenged[7].

Mammalians have four homologous HCN variants, HCN1-4, that are differentially expressed in a wide range of tissues[8]. The central functionality is equal for all variants, namely the hyperpolarization activated and cAMP modulated cation conductance. However, they vary in a number of parameters, including the hyperpolarization voltage as well as affinity, specificity and response to cAMP binding[9]. The most prominent process involving HCN channels is probably the '*pacemaker current*' of the heart run by sinoatrial nodal cells. This periodic current is primarily driven by HCN4[10].

[1] Wahl-Schott and Biel (2008).

[2] Wahl-Schott and Biel (2008).

[3] McCormick and Bal (1997); DiFrancesco (1993).

[4] Wahl-Schott and Biel (2008).

[5] Biel, Schneider, and Wahl (2002); Behar et al. (2016).

[6] Kusch et al. (2010); Kusch et al. (2012).

[7] White et al. (2021).

[8] Benzoni et al. (2021).

[9] Wahl-Schott and Biel (2008).

[10] DiFrancesco (2020).

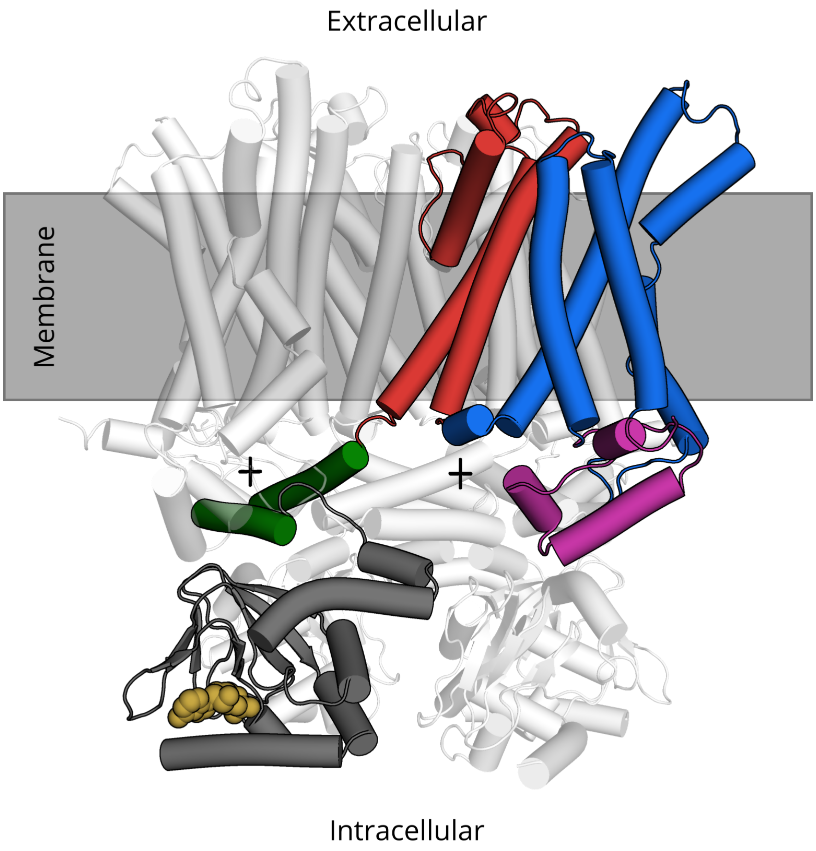


Figure 15.1: Overview of HCN4 tetramer structure. The domains of a single monomer are highlighted. From N- to C-terminus: HCN domain (violet), voltage sensor domain (blue), pore domain (red), C-linker (green), CNBD (gray). The bound ligand cAMP is shown in yellow, the putative Mg^{2+}-binding sites of the tetrads are marked with a '+'.

15.1.2 Structure of the HCN4 channel

The structure of the rabbit HCN4 channel tetramer was only recently elucidated by cryogenic electron microscopy (cryo-EM)[11] at a resolution of 3.3 Å (Figure 15.1). Both a structure model without ligand (*apo*) and with bound ligand (*holo*) were obtained.

The general domain organization is conserved within the HCN family[12]. At the cytosolic C-terminus of HCN channels the cyclic nucleotide binding domain (CNBD) is located, containing the cAMP binding pocket. Upon ligand binding, conformation changes are communicated via the C-linker to the transmembrane region. This region contains six transmembrane helices (S1-6) per subunit (Figure 15.2). Here the pore domain is situated, controlling the selective flow of ions via a loop between the S5 and S6 helix. This pore domain is flanked by the voltage sensor domain, whose S4 helix responds to voltage via nine positively charged and evenly spaced amino acids[13]. The N-terminal HCN domain is crucial for correct channel assembly in the plasma membrane, voltage dependence in channel opening and coupling the C-linker to the voltage sensor domain[14].

In contrast to the likewise structurally resolved HCN1 channel, HCN4 contains a tetrameric arrangement of four amino acids[15] termed *tetrad* between the S4-5 linker and the C-linker of the proximate subunit, that

[11] Saponaro et al. (2021).

[12] Jackson, Marshall, and Accili (2007).

[13] Wahl-Schott and Biel (2008).

[14] Porro et al. (2019).

[15] H407, D411, H553 and E557

putatively binds[16] Mg^{2+} as marked in Figure 15.1. Depletion of Mg^{2+} or removal of the tetrad leads to decreased stability and halved response to cAMP binding in terms of of half activation voltage.

15.1.3 Objective

Using the tool of the anisotropic network model (ANM) (see Chapter 11) and with structure models for both, *apo* and *holo* HCN4, at hand the linear response induced by cAMP binding is investigated in this chapter. First, the signal transduction from the CNBD to the C-linker and transmembrane domains as well as the role of the tetrad is analyzed. Second, potential cooperative behavior in the cAMP binding process is studied.

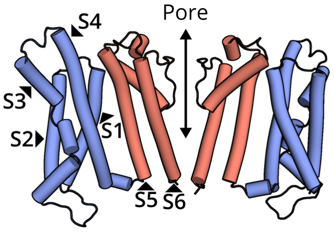


Figure 15.2: View on HCN4 transmembrane region. The voltage sensor domain (blue) and pore domain (red) are depicted. Only two opposite monomers are shown for clarity.

15.2 Methods

15.2.1 Network model of HCN4 channel

An ANM of the HCN4 channel was created based on the cryo-EM of *apo* HCN4[17] using *Springcraft*. Force constants between residues were obtained from the *sdENM* force field[18]. To investigate the effects of Mg^{2+}-binding in the tetrad, a variant of this ANM was created where in each tetrad the four involved residues received force constants equivalent to a covalent bond. This alteration should depict a strong stabilization of the tetrad due to the bound ion.

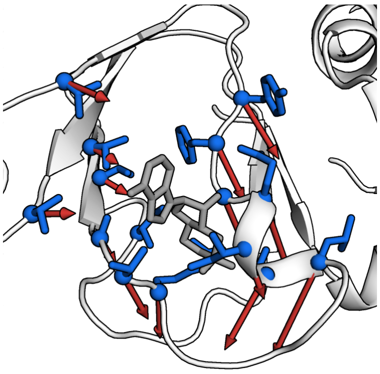


Figure 15.3: Forces used for linear response calculation. An *apo* CNBD and the *holo* cAMP position (gray) are shown. The residues in vicinity of cAMP are colored blue. The red arrows depict the displacement these residues perform in transition from *apo* to *holo* conformation. The displacement is equivalent to the force vector used in LRT. The arrow lengths are magnified by a factor of two for better visibility.

15.2.2 Linear response induced by cAMP binding

cAMP binding is accompanied by a conformation change in the CNBD that corresponds to the transition from the *apo* to the *holo*[19] structure model, assuming that the experimental structure models display the behavior under physiological conditions. In this work the displacement of the cAMP-binding residues in the *apo*-*holo* transition is interpreted as force vector in the framework of linear response theory (LRT). The relevant residues were chosen in the *holo* model based on a cutoff distance using a cell list: A residue is in vicinity of the ligand, if any of its heavy atoms is within 5 Å of any cAMP atom[20]. For each HCN4 monomer the CNBD[21] of the *holo* structure is superimposed upon the *apo* CNBD. The force vector $\vec{F}$ was defined as the displacement vector $\Delta\vec{r}_{\text{bind}}$ between the *apo* and *holo* positions of the C_α atoms in the chosen residues (Figure 15.3). The linear response of the ANM, i.e. atom displacement, induced by $\vec{F}$ was computed. For analysis of effects on the transmembrane region, simultaneous binding of cAMP to all four binding sites was simulated this way. For cooperativity analysis $\vec{F}$ was split into four force vectors $\vec{f}_i$, each one containing the forces on atoms in a single CNBD monomer. For each $\vec{f}_i$ the induced displacement $\Delta\vec{r}_{\text{ind},i}$ was obtained via LRT calculations. Binding of cAMP to any combination of the four binding sites can be simulated by summing the respec-

[16] Saponaro et al. (2021).

[17] PDB: 7NP3

[18] Dehouck and Mikhailov (2013).

[19] PDB: 7NP4

[20] The identifed residues are I624, V643, T645, T651, Y658, F659, G660, E661, I662, C663, L664, R670, T671, A672 and V674.

[21] residue 593 – 715

tive $\Delta\vec{r}_{\mathrm{ind},i}$, since

$$\begin{aligned}\vec{F} &= \sum_i \vec{f}_i,\\ \Delta\vec{r}_{\mathrm{ind}} &= \mathbf{H}^{-1}\vec{F}\\ \Delta\vec{r}_{\mathrm{ind}} &= \mathbf{H}^{-1}\sum_i \vec{f}_i\\ \Delta\vec{r}_{\mathrm{ind}} &= \sum_i \left(\mathbf{H}^{-1}\vec{f}_i\right)\\ \Delta\vec{r}_{\mathrm{ind}} &= \sum_i \Delta\vec{r}_{\mathrm{ind},i}. \end{aligned} \tag{15.1}$$

15.2.3 Displacement overlap measurement

To compare the induced displacements $\Delta\vec{r}_{\mathrm{ind}}$ according to LRT with the experimental displacements $\Delta\vec{r}_{\mathrm{bind}}$ between *apo* and *holo* conformations, the displacement overlap[22]

$$O = \frac{\Delta\vec{r}_{\mathrm{ind}} \cdot \Delta\vec{r}_{\mathrm{bind}}}{|\Delta\vec{r}_{\mathrm{ind}}| \cdot |\Delta\vec{r}_{\mathrm{bind}}|} \tag{15.2}$$

was calculated (Figure 15.4). O has a maximum value of 1, if the two vectors point in the same direction, and a minimum value of -1 if they point in opposite directions. $\vec{r}_{\mathrm{ind}}$ and $\vec{r}_{\mathrm{bind}}$ do not necessarily represent a single position but comprise all atoms of interest as a '*flattened*' vector. Note that in Equation 15.2 in contrast to Tama and Sanejouand the absolute of the numerator is not taken: While in that work normal modes were discussed, which by nature describe displacements in forward and opposite direction, the LRT model used here gives actual displacements in a single direction.

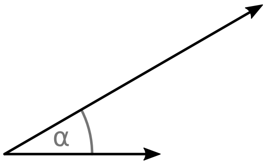


Figure 15.4: Overlap in two dimensions. The overlap O is equal to the cos α between two vectors. Since for the actual application n 3D vectors are combined into a flattened $3n$-dimensional vector, the term overlap is used for distinction.

15.3 Results and Discussion

15.3.1 Signal transduction to transmembrane region

cAMP binding is known to lower the hyperpolarization voltage required for opening and increases the open probability of HCN channels[23]. The effect is enhanced when $\mathrm{Mg}^{2+}$is bound to the tetrad[24]. As cAMP is bound in the CNBD, the resulting conformational change needs to be communicated through the C-linker to the pore or voltage sensor domain in order to enable channel opening. The ANM in combination with LRT gives an estimate how the conformation change in the CNBD may affect the conformation in those domains.

The induced displacement in the C-linker upon full ligation of all four CNBDs is shown in Figure 15.5. It can be observed that induced displacement does not resemble the displacement from *apo* to *holo* conformation observed in the experimentally determined structure. As quantitative measure the overlap O between the induced and experimental displacement vector, i.e. the congruence of vector directions, was calculated. With a value of $O = 0.073$ there is no considerable connection be-

[22] Tama and Sanejouand (2001).

[23] Wahl-Schott and Biel (2008).

[24] Saponaro et al. (2021).

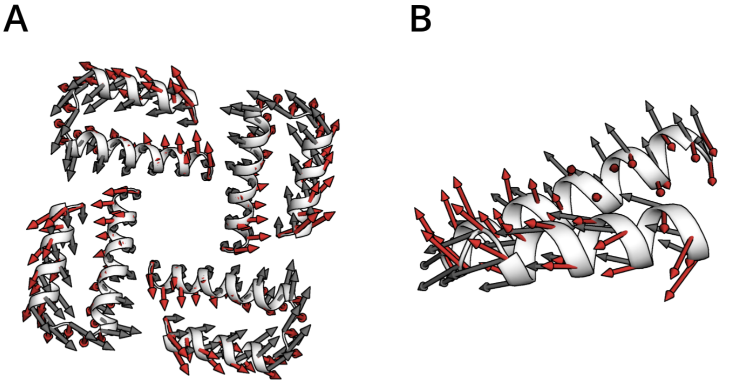


Figure 15.5: Computed and experimental C-linker displacement upon cAMP binding. The HCN4 C-linker domain in *apo* conformation is shown together with the displacement between the experimental *apo* and *holo* structure (red arrows) and the cAMP-induced displacement according to LRT (gray arrows). The arrows are magnified for the sake of clarity. **A** View into channel axis on C-linker tetramer. **B** Rotated view on a single C-linker domain. The bottom points toward the intracellular side, the top points toward the extracellular side.

tween both vectors. However, the induced displacement indicates a rotational movement of the C-linker disk, as seen in HCN1[25], though the direction of the rotation is opposite to the expected one. Whether the discrepancy between experiments and the simulation originates from artifacts in structure determination or the inaccuracies of the ANM, remains eluded.

In the next step the linear response of cAMP binding in the transmembrane region was analyzed. Similar to the C-linker, the induced movement in the transmembrane region does not fit the shift of S5 seen in the structure (Figure 15.6). The displacement overlap for the C_α atoms in helices S4-6 amounts merely to $O = 0.003$. Furthermore, the differences between the computed displacements with or without tetrad stabilization are small ($O = -0.045$). This supports observations that ENMs largely depend on global 3D shape[26] and in consequence are not suitable to resolve differences in only few residue contacts.

15.3.2 Ligand binding cooperativity

Though HCN2 cooperativity in binding cAMP was experimentally shown[27], this finding has also been challenged[28]. Furthermore, it is unknown to date whether such behavior also applies to HCN4 and in which order cAMP binds to the four CNBD monomers. In an endeavor to answer these two questions, binding of the ligand to a single HCN4 CNBD was simulated via LRT. The effect on the remaining three binding sites was evaluated in terms of overlap between the required displacement for cAMP binding[29] and the induced displacement according to LRT. A positive overlap would indicate positive cooperativity: The equilibrium position of the relevant residues are forced into the direction that is necessary for binding cAMP, facilitating ligand binding according to the '*lock-and-key*' principle. In consequence a negative overlap suggests negative cooperativity.

[25] Lee and MacKinnon (2017).

[26] Bahar et al. (2010).

[27] Kusch et al. (2012).

[28] White et al. (2021).

[29] displacement of *apo-holo* transition

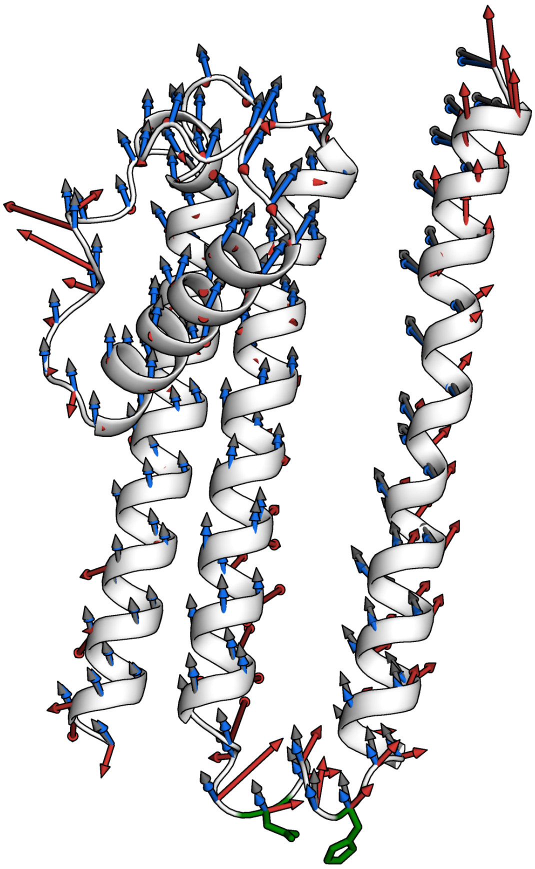


Figure 15.6: Computed and experimental transmembrane displacement upon cAMP binding. The helices S4-6 of a HCN4 monomer in *apo* conformation are shown. Red arrows depict the displacement between the experimental *apo* and *holo* structure. Gray and blue arrows show cAMP-induced displacement according to LRT without and with tetrad stabilization, respectively. The arrows are magnified for the sake of clarity. The residues involved in the tetrad are shown in stick representation (green).

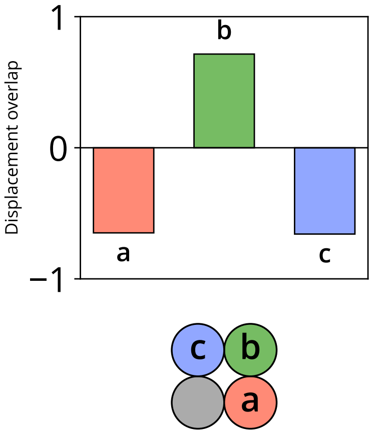


Figure 15.7: Overlap between induced and required displacement for cAMP binding. The overlap for the binding sites proximate (red, blue) and opposite (green) to the first cAMP binding site (gray) is shown. The view on the domains is from intracellular to extracellular side.

The calculated overlap indicates that the CNBD on the opposite of the first cAMP binding position shows positive cooperativity while the two proximate CNBDs show negative cooperativity (Figure 15.7). The induced and required displacements in the proximate and opposite CNBDs are visualized as arrows in Figure 15.8A and 15.8B, respectively. As the tetramer has almost rotational symmetry, the observed cooperativity should be independent of which CNBD is populated first. This circumstance is reflected by the overlap, which was found nearly identical in all four cases. In order to test whether the findings are an artifact of the ANM parameters, the computations were repeated using different force fields. All tested force fields showed qualitatively similar results (Table A.2). The results suggest, that on average the opposite position is populated second, as binding affinity is increased due to the cooperative behavior. This is followed by population of the proximate binding sites for the third and fourth binding.

Following this proposed binding order, the cooperativity pattern as shown in Figure 15.9 arises: As already pointed out the second binding

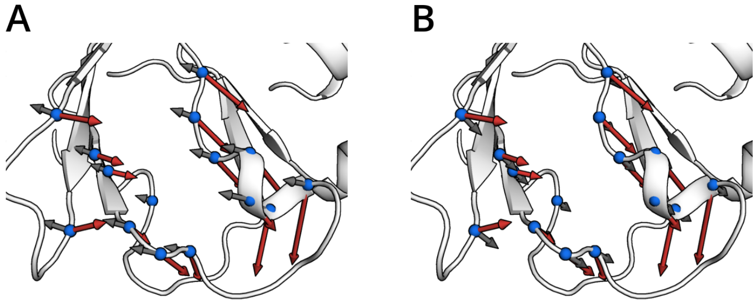


Figure 15.8: Induced and required displacement for cAMP binding. The residues in vicinity of cAMP are colored blue. The required (red) and induced (gray) displacements are shown as arrows. The arrow lengths are magnified by a factor of two for better visibility. **A** Proximate CNBD. **B** Opposite CNBD.

shows cooperative behavior. The two populated binding sites on the opposite of each other both induce negative cooperativity to the two empty proximate binding sites. Conversely, the third binding facilitates binding of cAMP to the remaining binding site on the opposite of it again. The overlap curve is consistent with to the experimentally measured binding affinity of cAMP to HCN2 (Figure 15.9), that also showed a positive-negative-positive cooperativity pattern[30].

The data shown in this work suggest that the binding affinity for the fourth cAMP binding event is lower than for the first binding, as the corresponding overlap is smaller than 0. This observation resembles the measured affinity for the open HCN2 channel more closely (Figure 15.9 red line), which shows a higher cAMP affinity for the first than for the fourth binding event. However, this interpretation has to be taken with caution, as with increasing number of bound cAMP molecules the actual HCN4 structure deviates more from the *apo* structure, which itself also has only limited accuracy. However, other experiments[31] also indicate that the *apo* structure of HCN4, which was taken as ANM input, is in an open state.

In the past mutational studies on HCN2 CNBD have shown that cAMP binding to subunits on opposite positions has a greater impact on a positive shift of the activation voltage than binding on proximate positions[32]. Assuming that the opposite CNBD is also populated second as proposed by LRT in this thesis, this observation fits the finding that the equilibrium constant between open and closed state is increased more by the second (presumably opposite) binding, than by the third (presumably proximate) binding[33].

15.4 Conclusion

The presented analyses demonstrate limitations and possibilities of ENMs. The simple '*ball-and-spring*' model of the protein complex was unable to distinguish between the Mg^{2+}-bound and unbound tetrad, probably due to the very local change. Furthermore, the simulated movement in the C-linker and transmembrane region did not match the *apo-holo* transition observed in the cryo-EM structures of HCN4. On

[30] Kusch et al. (2012).

[31] Saponaro et al. (2021).

[32] Ulens and Siegelbaum (2003).

[33] Kusch et al. (2012).

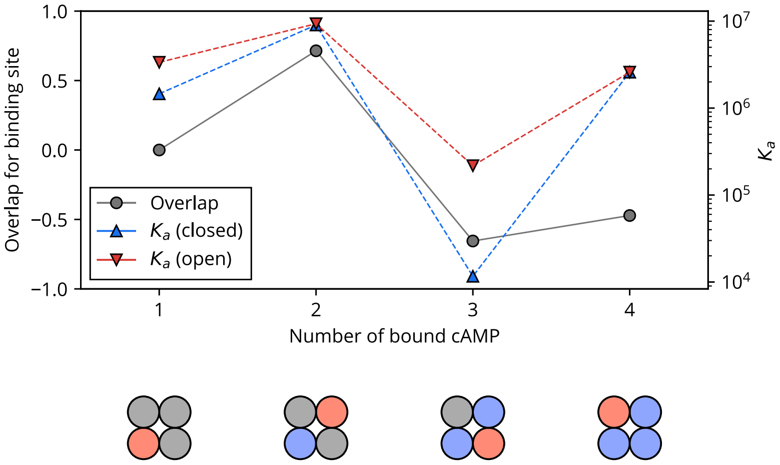


Figure 15.9: Overlap for successive cAMP binding. The figure shows the displacement overlap for the nth binding of cAMP to HCN4, based on the combined induced displacement from the previous $n - 1$ binding steps. The pictograms below the x-axis depict the position of the acute cAMP binding site (red) as well as already populated binding sites (blue). The experimentally measured HCN2 affinity for cAMP at each binding step is shown for reference.

the other hand, LRT revealed a plausible mechanism for cooperativity in HCN channels. It shows that the second cAMP binding likely occurs in the CNBD on the opposite of the first binding site. This method could not be applied to HCN2 for direct comparison with the experimental results, as neither an *apo* nor a *holo* structure is available for HCN2. Furthermore, experimental validation of cooperative behavior in HCN4, that would also give information about magnitude of cooperativity, is still missing. However, due to the strong conservation within the HCN family[34], a similar behavior as in HCN2 is expectable.

Cooperative behavior may have biological relevance: As shown for HCN2[35], cooperativity alters the channel's open probability as function of ligand concentration from a hyperbolic to a sigmoidal curve. In the popular cooperativity example of hemoglobin[36], the sigmoidal behavior leads to a stronger differentiation between low and high oxygen environments and in consequence to better oxygen loading at high oxygen concentrations and unloading at low oxygen concentrations. In context of HCN channels, cooperativity could improve switching behavior. Instead of the open probability slowly increasing with cAMP concentration, the response would become closer to binary when binding is cooperative: Above a threshold cAMP concentration the open probability would quickly reach the maximum value. Hence, cooperativity could result in a clearer response of HCN channels to signal transduction cascades altering cAMP concentration.

[34] Jackson, Marshall, and Accili (2007).

[35] Kusch et al. (2010).

[36] Ackers and Holt (2006).

Chapter 16
Analysis of HT-SELEX experiments

16.1 Introduction

Aptamers receive increasing attention for their sensoric and therapeutic capabilities and their opportunities in synthetic biology. Novel high-throughput sequencing (HTS) technologies allow researches to look into the process of *in vitro* selection of aptamers. In this chapter *Biotite* is used for thorough analysis of sequencing data from those selections. The results demonstrate how HTS can be harnessed to accelerate the time consuming selection process. Furthermore, the insights from sequencing data reveal issues in the SELEX protocol, that lead to deviations from an established mathematical description based on chemical equilibria. However, this work also proposes how these issues can be mitigated and elaborates how the mathematical foundation could be used to estimate the binding affinity for individual nucleic acid (NA) species from these data. Furthermore, this chapter suggests how the computed dissociation constants can be used to predict optimal selection rounds and high affinity aptamer sequences.

16.1.1 Aptamers bind target molecules with high affinity

Aptamers are single-stranded RNA or DNA molecules that bind a respective target compound. They can be seen as an NA analog to the protein-based antibodies, as they typically bind their target molecule with both, high affinity[1] and specificity[2]. Similar to their protein counterparts, aptamers are often structurally complex, frequently forming stem-loops and pseudoknots. A significant advantage of NAs over proteins from a technological point of view are the easy possibilities to synthesize, manipulate and sequence NAs. These properties allow aptamers to be artificially manufactured, in contrast to antibodies which need to be produced *in vivo*[3], and facilitate chemical modification. Furthermore, this enables the development of aptamers via high-throughput *in vitro* selection techniques, explained later in Section 16.1.3. This makes aptamers very versatile with respect to their molecular targets.

Aptamers for a multitude of target molecules have been published, ranging from large molecules like human activated protein C (APC)[4], to small targets like adenosine[5]. Figure 16.1 exemplarily shows the structure of a flavin mononucleotide (FMN) RNA aptamer[6]. Similar to targets of other aptamers[7], FMN binding is stabilized via hydrogen bonds to the opposing nucleobase as well as π-π stacking interactions with adjacent nucleotides[8].

[1] often in nanomolar range

[2] Li et al. (2021).

[3] Li et al. (2021).

[4] Gal et al. (1998).

[5] Huizenga and Szostak (1995).

[6] Fan et al. (1996).

[7] Peselis and Serganov (2014).

[8] Fan et al. (1996).

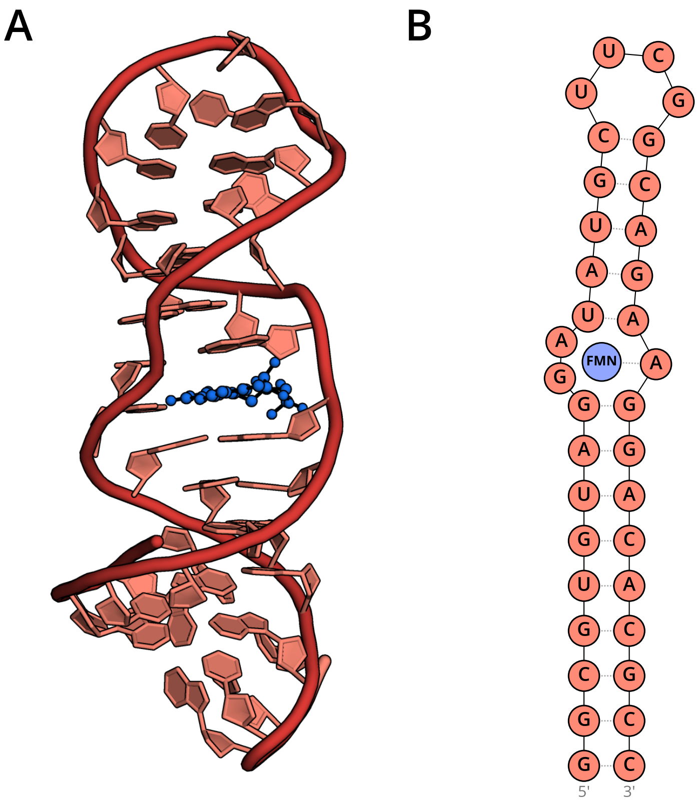


Figure 16.1: Structure of a flavin mononucleotide aptamer. The figure shows the structure of an artificial RNA aptamer that binds FMN (PDB: 1FMN). **A** Tertiary structure of the aptamer. FMN is shown in blue. **B** Secondary structure of the aptamer. Base pairs and the hydrogen bonds between FMN and the RNA are shown as dotted lines.

The high binding affinity of aptamers to their target molecule can be harnessed for different kinds of applications. These include fluorescent RNA aptamers acting as RNA counterpart to fluorescent proteins. In contrast to those proteins with inherent fluorescence, fluorescent RNA aptamers require an additional fluorescent dye, which the aptamer binds with high affinity and whose fluorescence is enhanced upon binding. Such aptamers can be used to localize RNAs in living cells by fusion of the RNA of interest and the fluorescent aptamer on the genetic level. This methodology can be extended to detect *in vivo* RNA-RNA interaction, by introduction of a second fluorescent aptamer and measurement of Förster resonance energy transfer (FRET)[9].

Aptamers are also used in sensor devices to detect specific small molecules in the environment. After binding of the aptamer to the target molecule, the bound conformation can be measured

- colorimetrically, e.g. via gold nanoparticles,
- electrochemically, by attaching a redox reporter to the aptamer or
- via fluorescence of an attached dye.

However, these types of reporters typically require conformational change of the aptamer upon binding. Colorimetric measurement via nanoparticles requires assembly or disassembly of nanoparticle networks, electrochemical detection requires that the distance of the reporter moiety to the electrode changes and fluorescence detection requires quenching of the fluorophore. An exception are for example ap-

[9] Neubacher and Hennig (2019).

tamers that work via the principle of fluorophore displacement, where the target molecule replaces the fluorophore in the binding pocket[10]. To enforce conformational changes, special *in vitro* selection methods can be employed (see Section 16.1.3) or the aptamer sequences can be engineered after selection by truncating stabilizing stem regions or even split the aptamer into two separate molecules[11]. Aptamer-based sensors allow diagnostic applications including diagnosis of infectious diseases and early detection of cancer cells[12].

Beside their sensing functionality, the binding properties of aptamers can also be exploited for therapeutic uses. It can be used for targeted delivery of a conjugated drug or for blocking protein function[13]. To date the only FDA and EMA approved aptamer-based drug is *Macugen* for treatment of age-related macular degeneration[14]. This aptamer-based drug specifically binds to anti-vascular endothelial growth factor (VEGF) inhibiting its signal transduction. However, *Macugen* was superseded by an antibody-based drug[15]. Nevertheless, the increasing number of clinical trials for aptamer-based drugs[16] signifies their relevance in therapeutic applications.

16.1.2 Riboswitches regulate gene expression

Riboswitches are specialized RNA aptamers that are able to regulate synthesis of gene products *in vivo*: In presence of the respective target molecule, the gene expression is either '*switched*' on or off. Although riboswitches have also been found via *in vitro* selection, similar to pure aptamers[17], numerous natural riboswitches appear in a range of species, that utilize these molecular devices to adapt to different environment conditions.

Figure 16.2 exemplarily shows the work principle of the guanine riboswitch from *Bacillus subtilis* that regulates the gene expression of the *xpt* gene, which is involved in purine metabolism[18]. A usual riboswitch consists of two structural domains: an aptamer and an expression platform. The aptamer is responsible for binding the target

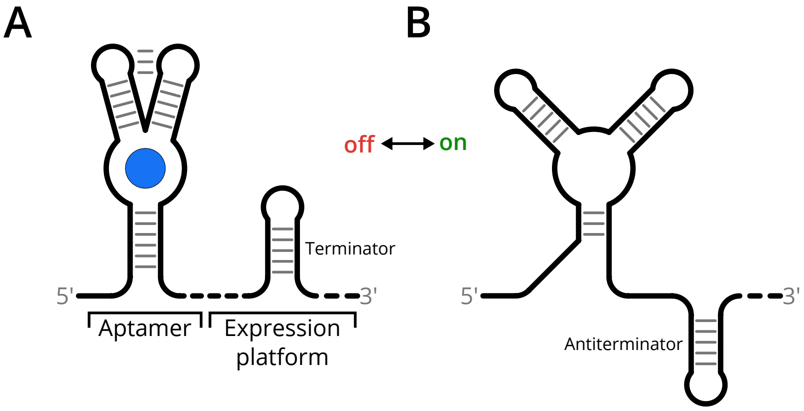


Figure 16.2: Operating principle of the guanine riboswitch. The figure schematically shows the secondary structure of the natural guanine-binding riboswitch upstream of the *xpt* gene from *Bacillus subtilis*. The ribsowitch regulates gene expression on the transcriptional level. **A** target-bound conformation of the riboswitch. The target guanine is shown in blue. **B** Unbound conformation of the riboswitch.

[10] Yu et al. (2021).

[11] Yu et al. (2021).

[12] Kumar Kulabhusan, Hussain, and Yüce (2020).

[13] Kumar Kulabhusan, Hussain, and Yüce (2020).

[14] Li et al. (2021).

[15] Zhou and Rossi (2017).

[16] Kumar Kulabhusan, Hussain, and Yüce (2020).

[17] Etzel and Mörl (2017).

[18] Serganov et al. (2004); Peselis and Serganov (2014).

molecule, which is guanine in this case. The expression platform translates the binding by the adjacent aptamer into a signal for the gene expression machinery to turn the expression of the downstream gene on or off. In detail, the bound molecule stabilizes a certain conformation in the aptamer. The conformation influences the folding of the proximate expression platform in consequence. In case of the guanine riboswitch this means, that the expression platform refolds from a conformation sequestering an antiterminator signal into a terminator stem-loop upon binding guanine. In consequence the transcription of the downstream *xpt* gene is prematurely terminated: the gene expression is turned off.

While this guanine riboswitch regulates expression on a transcriptional level, the switching mechanisms observed in nature are diverse. Overall, these include[19]

- ρ-factor dependent or independent premature transcription termination,
- alternative splicing by binding to splicing sites of pre-mRNA,
- inhibition of translation by concealing the ribosome binding site,
- self-degradation by forming an RNA cleaving ribozyme with bound target as cofactor,
- exposition of an RNase cleavage site and
- hybridization with an antisense mRNA to terminate transcription.

The ability to detect molecules with high affinity without the need for an additional protein[20], that would increase complexity of the regulation, makes riboswitches interesting building blocks for synthetic biological circuits[21].

16.1.3 In vitro selection of aptamers and riboswitches

Aptamers or synthetic riboswitches that bind a given target molecule are usually obtained *in vitro* via SELEX[22]. The individual steps of a SELEX experiment are shown in Figure 16.3A. The process begins with an initial library of $\sim 10^{15}$ different NA species[23]. Whether DNA or RNA molecules are used, depends on the type of aptamer that should be obtained[24]. The corresponding sequences are randomized in the middle, flanked by constant primer binding regions on both ends. Optionally, the central region may include an additional constant sequence that is prone to form a stem-loop structure to increase the chance for a successful aptamer selection[25]. This pool of different species is incubated on a column with substrate that is coated with the target molecule. NA molecules bind to the target molecules according to their affinity to that molecule. Afterwards the column is washed to remove unbound NA molecules. Then the bound NA is eluted from the column and amplified via polymerase chain reaction (PCR). For RNA additionally reverse transcription prior to PCR and transcription after PCR is required. Since non-binding molecules were removed in the washing

[19] Peselis and Serganov (2014).

[20] e.g. a repressor

[21] Etzel and Mörl (2017).

[22] Ellington and Szostak (1990); Tuerk and Gold (1990).

[23] 'Species' is used in a chemical sense here: Two NA molecules with equal sequence are considered as the same species.

[24] Riboswitches always require an RNA library.

[25] Davis and Szostak (2002).

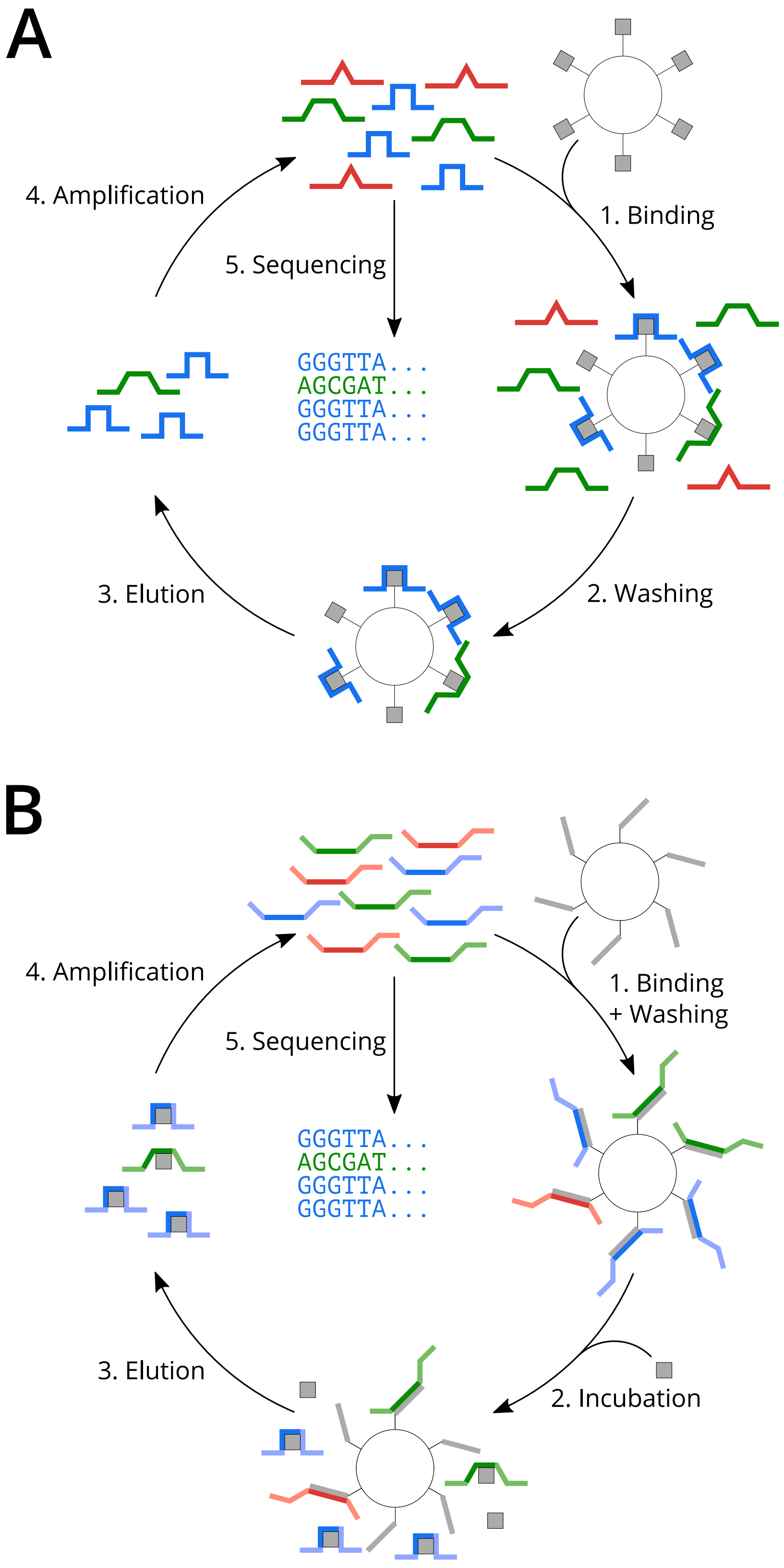


Figure 16.3: HT-SELEX experiment in a nutshell. **A** Traditional SELEX. The different NA species are depicted by different colors. The target molecules are shown as squares. Starting from the second iteration, the cyclic process begins with the amplified pool from the previous iteration. **B** Capture-SELEX. The common capture region in the NA species is highlighted by color intensity and the complementary capture oligonucleotide is shown in gray.

step(s), the newly obtained pool is enriched with NA species that exhibit at least some affinity to the target. To further enrich the pool with affine species, the selection round is repeated with the new pool, optionally with a lower target concentration to increase selection stringency. In order to avoid that NA species with affinity for the substrate also enrich, a negative selection round with pure substrate can be conducted prior to the actual SELEX.

The increasing collective affinity of the NA pool to the target can be tracked over the selection rounds by measuring the ratio between the eluted and input amount of NA. After a sufficient number of selection rounds, the enriched pool can be processed in further experiments: For example the pool can be cloned into the target organism for *in vivo* studies or it can be sequenced to obtain the aptamer sequence.

Since its invention, many variants of the SELEX protocol emerged. One of these developments is the capture-SELEX, in which not the target, but the nucleotide library is immobilized on the substrate (Figure 16.3B)[26]. This is achieved by placing a constant so called *capture region* inside the randomized sequence of the library. The substrate is coated with an oligonucleotide complementary to the capture region via a biotin-linker. Upon incubation of the substrate with the library, a part of the NA molecules hybridizes with the immobilized oligonucleotide molecules. Unbound NA are washed off. Thereafter, the target is added to the mixture. To be released from the substrate in this step not only target binding is required, but also a conformation change upon binding, that occludes the capture region. Similar to the protocol described above, these NA molecules are eluted and amplified, forming the new pool for following selection rounds. The circumstance that the capture-SELEX selects also for conformational changes is a major advantage of the method compared to the traditional SELEX, as such refolding is essential for the operating principle of riboswitches and sensoric aptamers. In addition, it does not require the target to be chemically modified for immobilization which can be technically complex and influences the physical interaction of the molecule with the NAs: An aptamer with affinity to the immobilized target may not bind the free target very well in the actual application scenario. Furthermore, counter selection against another, often similar compound, which should not be bound by the aptamer, is facilitated: Prior to elution with the target molecule, NA species with unspecific binding properties are washed off using the counter target.

With the use of HTS, deep insights into the SELEX experiment can be gained. In high-throughput SELEX (HT-SELEX) the NA pool obtained after each selection round is sequenced[27]. In this process ideally the sequence for each NA molecule in the pool sample is obtained. Hence, the NA species composition after each selection round can be observed in a detailed manner allowing the observation of sequence enrichment in earlier selection rounds.

[26] Stoltenburg, Nikolaus, and Strehlitz (2012).

[27] Zimmermann et al. (2010); Cho et al. (2010).

16.1.4 Objective

How sequencing data from HT-SELEX experiments can be used to obtain useful information about the progress of the SELEX experiment is laid out in this chapter. The analysis proceeds from quality control over data preparation to analysis of the overall sequence enrichment. This is exemplified for multiple previously generated HT-SELEX datasets. In contrast to using existing software for this analysis such as tools from the *Galaxy* project[28], *Biotite* is used here to gain additional flexibility for the analysis and to demonstrate fields of application for this library. Then this work applies an established mathematical foundation[29] to these datasets to reveal issues in the application of this theory to the actual SELEX experiment. Furthermore, this work presents an approach to estimate the dissociation constant of individual NA species from sequence counts. Based on the computed affinities methods are proposed to select ideal NA pools for further experiments and to improve aptamer affinity via predictions from a neural network.

16.2 Methods

16.2.1 HT-SELEX datasets

For the following studies HT-SELEX datasets from the working group for *Synthetic RNA biology* were analyzed. These data comprise experiments for RNA aptamer and riboswitch selection against different small molecules as target molecule.

For selection against ciprofloxacin (CX) a classic SELEX approach was used[30]. The initial RNA library comprises a mixture of two designs: First, a completely randomized sequence, and second, a short fixed sequence flanked by two randomized regions. In both designs, the sequence is surrounded by 5' and 3' primer binding regions. Prior to sequencing, DNA barcodes of length four were attached to the 3'-end to assign sequences to their corresponding selection round after multiplexed sequencing. SELEX was conducted for ten rounds, where the selection pressure was increased in round 5, 6 and 10 by decreasing the concentration of the column bound CX, switching to elution with the target molecule and increasing the number of washes.

For SELEX against paramomycin (PM)[31] and all following targets a capture-SELEX was conducted instead. The RNA library consists of a fixed capture oligonucleotide flanked by randomized regions and, again, by primer binding regions. Barcodes of length seven were attached to the 5'-end for sequencing. The first seven selection rounds elution was conducted with 1 mM PM. In selection rounds 8 and 9 the PM concentration was reduced to 0.1 mM. In rounds 10 and 11 an additional counter elution with 0.1 mM neomycin (NM) was performed prior to elution.

[28] Thiel and Giangrande (2016).

[29] Levine and Nilsen-Hamilton (2007).

[30] Groher et al. (2018).

[31] Boussebayle et al. (2019).

While in the previously presented SELEX experiments the aim was achievement of a riboswitch, the purpose of the following selections was identification of RNA aptamers for the respective targets. For increased stability the RNA contained 2'-deoxy-2'-fluorocytidine and 2'-deoxy-2'-fluorouridine as nucleosides.

The SELEX experiments for selection against levofloxacin (LX) and chloroquine (CQ)[32] were sequenced together using multiplexing. The library and barcode design is identical to the selection against PM. For each of these experiments eleven selection rounds were run. While in the first nine rounds elution was performed with 1 mM target, five different conditions were tested for round 10, as discussed in Section 16.3.3. In round 11 a counter selection with 1 mM CX or amodiaquine (AQ) was performed, respectively, followed by an elution with 1 mM target.

For selection against cortisol (CL)[33] 27 SELEX rounds were run. Up to round 22 elution was performed with 1 mM target. In the remaining five rounds two conditions were tested: In one approach elution was performed with 0.1 mM target, in the second approach counter-elution was performed with 1 mM cortisone (CN) prior to elution with 1 mM CL. Since sequencing was not multiplexed, no barcodes were used.

The sequences of the library designs are shown in Figure 16.4. The sequences of the PM riboswitch candidate and the CL aptamer candidate are shown in Figure A.1. In the following, datasets are usually abbreviated by their respective target name for the sake of brevity. For a clear distinction between the dataset and the target itself, the datasets are highlighted in italics.

16.2.2 Sequencing quality assessment

It is good practice to assess the quality of HTS data before using the data for further analysis. In this initial quality control the researcher can verify that the sequencing output fulfills the expected parameters and has an appropriate confidence of the base calls[34].

The confidence for each base call is usually expressed as Phred quality score

$$Q = -10 \log_{10} p_{\text{err}}, \tag{16.1}$$

that is based on the error probability p_{err}[35,36]. Tools like *FastQC*[37] offer

CX partially randomized:

GGGAGACGCAACUGAAUGAA N26 CUGCUUCGGCAG N26 UCCGUAACUAGUCGCGUCAC

CX fully randomized:

GGGAGACGCAACUGAAUGAA N64 UCCGUAACUAGUCGCGUCAC

PM, *LX*, *CQ* and *CL*:

GGGCAACUCCAAGCUAGAUCUACCGGU N40 CUACUGGCUUCUA N10 AAAAUGGCUAGCAAAGGAGAAGAACUUUUCACU

Figure 16.4: Library designs for SELEX experiments. RNA sequences are given from 5'-end to 3'-end. '*Nxx*' indicates a randomized region with the given length. Barcodes are excluded.

[32] Suess *et al.* (unpublished data).

[33] Suess *et al.* (unpublished data).

[34] Base calls are the assignment of nucleobases to raw signals from the sequencing device

[35] the probability of an incorrect base call

[36] Ewing and Green (1998).

[37] Andrews (2010).

a user-friendly way to visualize the sequencing quality by providing a multitude of different plots. In this work customizable scripts using *Biotite* were used for quality analysis instead.

A challenge in the analysis of HTS data is its usual size: A single FASTQ file may reach a file size of multiple gigabytes. Hence, especially commodity hardware is not able to load an entire file into memory. This circumstance prevents usage of vectorized operations to the full extent. Instead, the employed scripts read the sequences consecutively and only the relevant data, for example its average Phred score, was extracted from each read. This condensed data was then used for further analysis.

For analysis of the sequence composition reverse complement sequences were corrected. If the first six nucleotides of a read were equal to the reverse complement of the final six nucleotides at the 3'-end of the designed library (see Section 16.2.1), the sequence was considered a reverse complement.

16.2.3 Demultiplexing and aggregation

For the underlying SELEX datasets sequencing was multiplexed[38]: All selection rounds from one SELEX experiment were pooled and sequenced simultaneously. To keep the output reads unambiguously assignable to the selection round they originate from, each read has a short[39] barcode sequence at the 5' or 3' end, that was attached to the DNA prior to pooling. The barcodes for each selection round were selected in a way that at least two substitutions are necessary to transform the barcode of one selection round to the barcode of any other selection round. This measure ensures correct round assignment in the presence of sequencing errors.

For further data analysis the sequencing datasets were demultiplexed by comparing the barcode region of each read to the used barcodes: The round where the corresponding barcode has the lowest *Hamming distance*[40] to the barcode region of a read was chosen as the round of that read. If multiple barcodes had the same distance, no unambiguous assignment was possible and the sequence was ignored in consequence. Reverse complement sequences were corrected prior to round assignment.

In the next step, equal sequences from the same selection round were aggregated into a single entry and the number of their occurrences was counted. Special measures to mitigate sequence errors were not employed due to the low sequencing error probability that was observed in the datasets. The unique sequences were sorted by their frequency in descending order. Their position in the sorted list is called *rank*.

[38] with the exception of the *CL* dataset

[39] 4 to 7 bases, depending on the dataset

[40] i.e. the lowest number of base substitutions

16.2.4 Analysis of sequence enrichment

The aim of a SELEX experiment is the enrichment of NA species with good binding properties to a defined target molecule from an initially randomized pool. HT-SELEX allows the condensation of this pool enrichment into a single value for each selection round, though no universal standard has been established, yet. Instead, multiple measures have emerged[41], from which the following are used in this work.

For calculation of the metrics, a list of all unique NA species $(1, ..., N)$ observed in a selection round is required. The list is sorted by relative frequency p_i or absolute frequency n_i of the unique sequences in descending order. One metric is the percentage of orphans

$$p_{\text{Orphans}} = \sum_{i=1}^{N} \begin{cases} p_i, & \text{if } n_i = 1 \\ 0, & \text{otherwise} \end{cases}, \tag{16.2}$$

that declines with increasing enrichment. On the other side the percentage of the 100 most abundant sequences

$$p_{\Sigma\,\text{Top 100}} = \sum_{i=1}^{100} p_i \tag{16.3}$$

is expected to increase with the number of selection rounds. The advantage of these two percentages is their value range between 0 and 1. However, both measures only take the least or most enriched part of species into account, respectively.

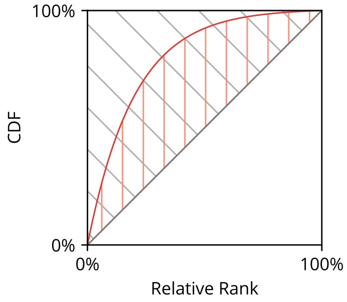


Figure 16.5: Visualization of the Gini index. The red line represents the cumulative distribution function (CDF) of interest, while the gray line is the cumulative uniform distribution. The Gini index is the ratio between the area hatched in red and the area hatched in gray.

One quantity that remedies this shortcoming is the entropy

$$H = -\sum_{i=1}^{N} p_i \log_2 p_i, \tag{16.4}$$

that originates from information theory[42], though it lacks an upper limit.

Instead the Gini index

$$G = 2\left(\left(\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{i} p_j\right) - \frac{1}{2}\right), \tag{16.5}$$

provides both, a value range between 0 and 1 and a consideration of all sequences in the selection round. Originating from economics, it describes the area difference between a cumulative distribution function of interest and a uniform distribution[43] (Figure 16.5).

16.2.5 Estimation of dissociation constant

The knowledge about the relative frequency of NA species after each round of selection creates the possibility to estimate their dissociation constants, as elaborated in the following section.

[41] Thiel et al. (2012); Groher et al. (2018); Boussebayle et al. (2019).

[42] Shannon (1948).

[43] Farris (2010).

Previously, Levine and Nilsen-Hamilton described a mathematical model for the SELEX process based on chemical equilibria[44]. The equilibrium of a binding process can be expressed via its dissociation constant

$$K_d = \frac{[R][T]}{[C]}, \tag{16.6}$$

where $[R]$, $[T]$ and $[C]$ are the concentrations of free receptor (the NA), free target and their complex in the solution, respectively. In context of a conventional SELEX this equilibrium describes the incubation of an NA pool with the target-coated substrate. For a capture-SELEX it means the elution of NA from the substrate with the target. Analogously, $[R]_{\text{tot}}$ and $[T]_{\text{tot}}$ describe total concentrations of receptor and target in the solution, i.e. the sum of both free receptor/target and complex. Hence, one can also write

$$K_d = \frac{([R]_{\text{tot}} - [C])[T]}{[C]}. \tag{16.7}$$

In the context of a SELEX, there is no single receptor species but a mixture of different NA species. Hence, K_d refers to the collective affinity of the NA pool and is called $K_{d,\text{coll}}$ from now on. Furthermore, one can define $\theta = [C]/[R]_{\text{tot}}$. In the context of a SELEX round θ describes the percentage of NA, that is eluted from the column relative to the amount of input NA initially given onto the column. This quantity is usually tracked in SELEX experiments. Thus,

$$K_{d,\text{coll}} = \frac{(1-\theta)[T]}{\theta}. \tag{16.8}$$

Solving Equation 16.7 for the formed complex gives

$$\begin{aligned} K_{d,\text{coll}} &= \frac{([R]_{\text{tot}} - [C])[T]}{[C]} \\ K_{d,\text{coll}} &= \left(\frac{[R]_{\text{tot}}}{[C]} - 1\right)[T] \\ K_{d,\text{coll}} &= \frac{[R]_{\text{tot}}[T]}{[C]} - [T] \\ [C] &= \frac{[R]_{\text{tot}}[T]}{K_{d,\text{coll}} + [T]}. \end{aligned} \tag{16.9}$$

One can also define the individual dissociation constant $K_{d,i}$ for a single NA species with the free and total concentration $[R_i]$ and $[R_i]_{\text{tot}}$ respectively. Analogous to Equation 16.9, the Equation 16.6 can be solved for the concentration of the complex $[C_i]$, that this species forms with the target in equilibrium:

$$\begin{aligned} K_{d,i} &= \frac{[R_i][T]}{[C_i]} \\ [C_i] &= \frac{[R_i]_{\text{tot}}[T]}{K_{d,i} + [T]}. \end{aligned} \tag{16.10}$$

Using the relative frequency of an NA species within a pool $p_i = [R_i]_{\text{tot}}/[R]_{\text{tot}}$, one obtains

$$[C_i] = \frac{p_i[R]_{\text{tot}}\,[T]}{K_{d,i} + [T]}, \tag{16.11}$$

[44] Levine and Nilsen-Hamilton (2007).

where

$$\sum_i p_i = 1. \tag{16.12}$$

Thus, using Equations 16.9, 16.11 and 16.12

$$\begin{aligned}
\frac{[R]_{\text{tot}}[T]}{K_{d,\text{coll}} + [T]} &= [C] \\
\frac{[R]_{\text{tot}}[T]}{K_{d,\text{coll}} + [T]} &= \sum_i [C_i] \\
\frac{[R]_{\text{tot}}[T]}{K_{d,\text{coll}} + [T]} &= [R]_{\text{tot}}[T] \sum_i \frac{p_i}{K_{d,i} + [T]} \\
\frac{1}{K_{d,\text{coll}} + [T]} &= \sum_i \frac{p_i}{K_{d,i} + [T]}.
\end{aligned} \tag{16.13}$$

Figuratively speaking, Equation 16.13 shows that at low target concentrations $[T] \ll K_{d,i}$ the association constant $K_a = 1/K_d$ of the pool is the average of the association constants of the individual NA molecules.

The eluted NA $[C_i]$ of one selection round is the basis of the input NA pool $[R_i']_{\text{tot}}$ of the next round. The amplification process between the rounds may change the absolute NA concentrations, but in theory it does not alter the relative frequencies of the NA species. Thus, $[C_i]$ and $[R_i']_{\text{tot}}$ have a proportional relation with a constant factor a:

$$[R_i']_{\text{tot}} = a[C_i]. \tag{16.14}$$

With Equation 16.11 this equation can be rewritten as

$$p_i'[R']_{\text{tot}} = a\frac{p_i[R]_{\text{tot}}\,[T]}{K_{d,i} + [T]}. \tag{16.15}$$

By setting the relative frequencies of two different NA species into relation, most parameters, including the unknown a, are removed from this equation:

$$\begin{aligned}
\frac{p_i'}{p_j'}\frac{[R']_{\text{tot}}}{[R']_{\text{tot}}} &= \frac{p_i[R]_{\text{tot}}\,[T]}{K_{d,i} + [T]}\,\frac{K_{d,j} + [T]}{p_j[R]_{\text{tot}}\,[T]} \\
\frac{p_i'}{p_j'} &= \frac{p_i}{p_j}\,\frac{K_{d,j} + [T]}{K_{d,i} + [T]}.
\end{aligned} \tag{16.16}$$

Based on the knowledge about the relative frequencies of each NA species in the pool after each selection round, this work uses this model by Levine and Nilsen-Hamilton to calculate the individual K_{d_i} value for each NA species. Based on Equations 16.8, 16.13 and 16.16 the relation

$$\begin{aligned}
\frac{1}{K_{d,\text{coll}} + [T]} &= \sum_i \frac{p_i}{K_{d,i} + [T]} \\
\frac{1}{K_{d,\text{coll}} + [T]} &= \sum_i \frac{p_j}{K_{d,j} + [T]} \frac{p'_i}{p'_j} \\
\frac{1}{K_{d,\text{coll}} + [T]} &= \frac{p_j}{p'_j} \frac{1}{K_{d,j} + [T]} \sum_i p'_i \\
K_{d,j} + [T] &= \frac{p_j}{p'_j} \left(K_{d,\text{coll}} + [T]\right) \\
K_{d,j} &= \frac{p_j}{p'_j} \left(\frac{(1-\theta)[T]}{\theta} + [T]\right) - [T] \\
K_{d,j} &= \frac{\frac{p_j}{p'_j} - \theta}{\theta}[T]
\end{aligned} \tag{16.17}$$

can be given. Hence, only the relative frequencies of the respective NA species before and after a round of selection, θ and the concentration of the free target are required. In case of a capture-SELEX $[T]$ can be easily substituted by $[T]_{\text{tot}}$ as approximation, if an excess amount of target is used for elution. Furthermore, a precise value of $[T]$ is not important for the relative comparison of K_d for different NA species, as it merely represents a proportionality factor. In contrast, a precise measurement of θ is essential to obtain sensible K_d values, even if the species are only compared relatively.

If θ is larger than p_j/p'_j, Equation 16.17 gives nonphysical negative values for $K_{d,j}$. In the context of the model employed here, this cannot happen as illustrated by the following extreme case: Let the NA species j be a perfect binder that represents only a small proportion of the entire NA pool, i.e. $K_{d,j} \rightarrow 0$ and $p_j \rightarrow 0$. After elution, all molecules from species j are still present in the pool due to its high affinity, but only θ of all other species remain. Consequently, in this case the relative frequency of species j in the new pool is increased by the factor $1/\theta$, i.e. $p_j/p'_j = \theta$. In all less extreme cases[45], this amplification factor is smaller, i.e. $p_j/p'_j > \theta$. Hence, negative K_d values may not appear, if the assumptions of this model uphold.

These assumptions include the following:

1. **The incubation times are long enough for binding processes to reach equilibrium.** If not stated otherwise, the NA molecules were incubated with the target solution for at least 5 minutes for the SELEX datasets used in this work.
2. **Each NA molecule can only bind to a single target molecule.** Finding an aptamer with two binding sites for the same target from a pool of randomized NA molecules is improbable. Furthermore, calorimetric measurements performed on aptamers obtained from the SELEX experiments discussed in this chapters show, that these aptamers have indeed only one binding site.

[45] larger $K_{d,j}$, larger p_j

3. **There is neither unspecific binding nor loss of the NA molecules to the support.** Unspecific binding is usually impeded by an initial counter selection against the support alone.

4. **The reverse transcription, amplification and transcription do not change the composition of the NA pool, i.e. the relative frequency of each NA species does not change in this process.** Since the sequence regions responsible for these steps are constant across all NA species, there is ideally no bias, that would favor certain species. However, laboratory experiments suggest that such biases exist nevertheless[46].

5. **There are no stochastic effects, i.e. for the entire selection and amplification process always the expected value is assumed for each NA species.** At least for the later rounds, the basic idea behind SELEX is that the relevant NA species[47] appear a large number of times in the NA pool. Due to the law of large numbers, a convergence of the concentrations to the expected value is assumed. However, at low species number, mathematical modelling has shown dramatic stochastic effects[48].

16.2.6 Calculation of library score

In the SELEX projects whose datasets were used in this work, a central purpose of sequencing the NA pools was the identification of an appropriate pool for *in vivo* studies[49]. First, a high enrichment of well performing aptamers in the NA pool is desired to gain a high probability that these aptamers are picked by chance, when the NA pool is cloned into the respective expression system. Second, a reasonable sequence diversity is also essential, to be able to test multiple different aptamers in the *in vivo* assays. This is especially necessary for riboswitches, since an aptamer with a high affinity to the target does not necessarily have good switching capabilities in the context of an mRNA. However, these objectives are conflicting: The highest enrichment is obtained, when only a single NA species remains in the pool, i.e. the diversity is minimal. The established metrics, including the ones introduced in Chapter 16.2.4[50] are optimal[51] when the sequence enrichment is at maximum. To date, there is no indicator published that also takes the sequence diversity into account as well.

Hence, a novel pool score for a NA pool q is introduced in this work: The score takes the probability P_i^q, that a NA species i is randomly picked at least once as sample for an e.g. *in vivo* assay, and weights this probability with the association constant $K_a = 1/K_d$, representing the *'importance'* of the species. Therefore the pool score S_{pool} is defined as

$$\begin{aligned} S_{\text{pool}}(q) &= \sum_i \frac{1}{K_{d,i}} P_i^q(X \geq 1) \\ S_{\text{pool}}(q) &= \sum_i \frac{1}{K_{d,i}} (1 - P_i^q(X = 0)). \end{aligned} \tag{16.18}$$

[46] Thiel et al. (2011); Tsuji et al. (2009); Takahashi et al. (2016).

[47] the ones with a low K_d

[48] Spill et al. (2016).

[49] Groher et al. (2018); Boussebayle et al. (2019).

[50] entropy, Gini index, Σ Top 100, orphans

[51] either at minimum or at maximum

The K_d for an NA species can be estimated via equation 16.17. Drawing the same NA species more than once is considered redundant, so it would not positively contribute to the score. Quite the contrary, having a high probability for a single NA species to be drawn multiple times would consequently lead to a lower probability for every other species to be drawn even once. Hence, a high enrichment of a single species would be penalized.

When an NA molecule is picked from the pool as a sample, this molecule is then missing in the pool. Therefore, a hypergeometric distribution would accurately model this behavior. However, this distribution requires knowledge about the absolute number of molecules for each NA species in the pool to be sampled from. Since the sample used for sequencing comprises only a small fraction of the original pool and at the same time also only a fraction of the pool is used for picking samples, using a hypergeometric distribution for P_i^q would require a lot of experimental details. However, for a high number of the respective NA species in the pool and a relatively small number of samples, the hypergeometric distribution can be approximated by a binomial distribution[52] $B(n, p, k)$, which only requires the relative frequency p_i^q of each NA species i instead of the total number. Thus,

$$S_{\text{pool}} = \sum_i \frac{1}{K_{d,i}} \left(1 - B(\lambda, p_i^q, 0)\right), \tag{16.19}$$

where λ is the number of samples picked. For example, if downstream *in vivo* assays are performed using a 96-well plate, λ would be 96 in this case. Eventually, using the S_{pool} as metric, the optimal NA pool q_{opt} for such assays would be the one with maximum S_{pool}. Hence,

$$q_{\text{opt}} = \arg\max_q S_{\text{pool}}. \tag{16.20}$$

16.2.7 Machine learning

Machine learning algorithms are able to automatically detect patterns in given data. Supervised learning is the subfield in machine learning, where the algorithm learns patterns on curated training data to apply the '*knowledge*' on unknown data of the same type. More specifically in the training phase the algorithm obtains pairs of input data (e.g. the pixels of an image) and output data (e.g. the label `'cat'`). Since the output data is known, the data is called *labeled*. Thereafter, the algorithm is tasked with predicting the corresponding output data for comparable *unlabeled* input data [53]. Two types of output data can be distinguished here: If the possible output is one of multiple qualitative classes (e.g. `'cat'` or `'dog'`), the problem is called a *classification* problem. If the output is a quantity (e.g. the number of whiskers), the problem is called a *regression* problem.

The *perceptron*[54] is such a supervised learning algorithm: The percep-

[52] College of William & Mary (2010).

[53] data with unkown output

[54] Rosenblatt (1958).

tron represents a *node* that accepts input data and returns a linear function of these data. The weights of the linear function are adapted in the training phase to match the labels as optimally as possible. For prediction of unlabeled data the linear function is applied to the input with the weights obtained from training phase. A neural network (NN) is an extension of this concept to multiple layers and multiple nodes per layer: The output of one node is the input to the nodes of the following layer. If more than one layer is used, the model design is often called deep NN, hence the name *deep learning*. In order to be able to learn non-linear relations between input and output data, usually a non-linear *activation function* is applied to the output of each node. Multiple variants of NN emerged to increase their performance for certain classes of learning tasks. One prominent variation is the convolutional neural network (CNN): While originally all features in the input data are treated as independent and no vicinity can be represented between them, a CNN convolutes *filters*[55] with overlapping frames of an input sequence of values. Since adjacent data points in the input sequence are used to compute an output value, the vicinity of data points can be recognized with this method. CNNs have been proven very successful in image analysis and have also shown promising results in the application on biological sequence data[56].

In recent years, numerous deep learning approaches have been established to predict NA binding specificities from experimental data, such as chromatin immunoprecipitation and HT-SELEX[57]. The underlying idea is that well-binding sequences have common motifs that can be learned by an NN. The sequence of a respective NA species is the input to the NN, while an experimentally determined value, e.g. whether the sequence is enriched after a number of rounds[58], serves as output value for training the NN. Including RNA secondary structure information from minimum free energy (MFE) computations in the NN input was also explored in recent years but did not lead to a considerable improvement of prediction accuracy[59].

Previous work has focused on predicting an experimental value from sequence, for example, whether an NA species is enriched in the final SELEX round[60]. The predicted classification itself is usually not of interest, since it is specific to the experiment and no intrinsic property of the NA. Consequently, such classification is mainly used to identify sequence motifs or predict mutations that optimize the respective experimental value. However, the dissociation constant of a binding process between an aptamer and its target, as computed according to Section 16.2.5, is a fundamental quantity.

Hence, in the context of this work the established practices for predicting binding specificities from sequence were applied to estimate the dissociation constant of an NA species to the target, even if that species did not appear during the SELEX experiment. Consequently, the objective was shifted from a classification problem to a regression problem.

[55] Filters are adapted in the training process, similar to the weights of regular NNs.

[56] Alipanahi et al. (2015).

[57] Trabelsi, Chaabane, and Ben-Hur (2019).

[58] Alipanahi et al. (2015); Asif and Orenstein (2020).

[59] Trabelsi, Chaabane, and Ben-Hur (2019).

[60] Alipanahi et al. (2015); Yuan et al. (2019); Asif and Orenstein (2020).

These predictions should be used to propose beneficial mutations for already performant aptamers.

Note that if the absolute K_d was used as NN output, the prediction accuracy would not match the scientist's intuition: NA species with poor binding properties typically have a K_d orders of magnitude higher than affine aptamers. Hence, NN training would be biased to minimize the prediction error for those poorly binding species with a high absolute K_d, while any value close to zero would already have a small difference to the K_d of well binding species. In contrast to that, a high prediction accuracy for species with particularly low K_d is important for *in silico* mutation scanning. Consequently, the correct order of magnitude of the dissociation constant is the more relevant metric here. Hence,

$$pK_d = -\log_{10} K_d, \tag{16.21}$$

is defined here, analogous to e.g. pH and pK_a, and used as output of the NN.

A challenge for the training task was the low number of NA species for which a pK_d can be calculated: Although thousands of different sequences appear in the dataset of a single selection round, only a small fraction of them appear in sufficient numbers as explained in the following Section 16.2.8. Filtering reduces the number of training samples to only a few hundred. Furthermore, the aptamer candidate used for *in silico* mutation studies was excluded from the training set to avoid overfitting to this specific sequence of high interest. The sequences used for training exhibit at least some affinity to the target, since they still appear after selection. Hence, an equal number of *decoy sequences* was added to the training set: For each decoy a randomized sequence was generated with the same constant regions as the initial pool used for SELEX. A $pK_d = 0$ was arbitrarily assigned to each decoy, representing a low affinity of $K_d = 1\,\text{M}$.

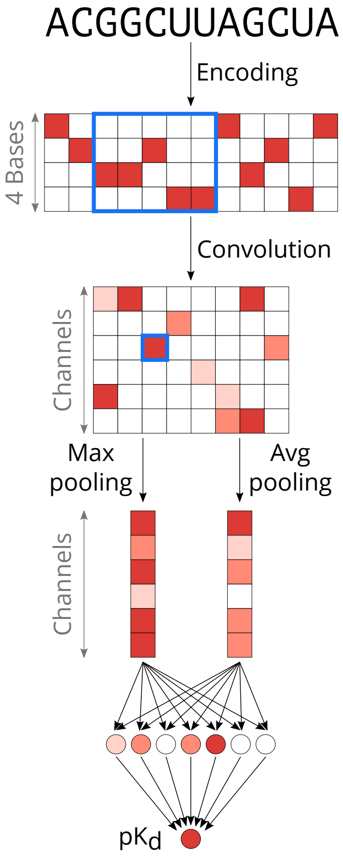


Figure 16.6: Layers of the NN for $\mathbf{pK_d}$ prediction. The figure depicts the data flow through the network based on a hypothetical RNA sequence. An exemplary convolution kernel and its output value is depicted in blue.

Finally, each sequence was *one-hot* encoded[61] into a feature vector of shape $(k, 4)$, where k is the sequence length of the initial SELEX pool, excluding the constant 5' and 3' primer binding sites. Due to insertions and deletions some sequences are slightly longer or shorter. To keep the feature vector at constant size, those sequences were truncated or extended into the constant region, respectively. The feature vector was used as input for the NN, while the calculated[62] pK_d was the expected output.

Due to the small number of training samples, a relatively simple NN design based on *DeepBind*[63] was chosen to avoid overfitting. It contains

1. a one dimensional convolutional layer,
2. a global maximum and average pooling layer,
3. a fully connected layer and
4. an output node

[61] In *One-hot* encoding an array is created for each symbol with the same length as the underlying alphabet. Then the position corresponding to the symbol is set to 1, while the other elements are set to 0. For example the encoding for 'G' would be '`0010`'.

[62] constant value for decoys

[63] Alipanahi et al. (2015).

(Figure 16.6). A *rectified linear unit* was used as activation function after each layer except the pooling layer. As the NN performs a regression, the mean squared error (MSE) between the predicted and expected output was used as loss function. The NN was implemented using the *PyTorch* library[64] and the *Adam* optimizer[65] was used for weight optimization in the training process.

In order to increase the prediction accuracy, the hyperparameters

- batch size,
- learning rate,
- weight decay,
- convolution kernel size,
- number of convolution channels and
- number of fully connected nodes

were optimized using random sampling implemented via the *Optuna* library[66].

For training the *CL* data was randomly separated into three sets: 80 % of the data (694 sequences) was used for training, i.e. for optimization of the weights in the NN, 10 % (86 sequences) were used as validation set for hyperparameter optimization and 10 % (86 sequences) were used as test set for the resulting NN.

16.2.8 Mitigating experimental deviations from theory

Equation 16.19 sums over all NA species that exist in the respective pool. However, as discussed later in Section 16.3.4, especially NA species that appear in small numbers tend to have erratic amplification factors[67]. Hence, the implementation of S_{pool} in this work deviates from Equation 16.19 by adding a relative frequency threshold of 0.01 %. The K_d was not calculated for NA species that fall below this threshold[68] and hence were excluded from S_{pool} calculation. In theory, NA species that appear with a low frequency have a low K_a and low $P_i^q(X \geq 1)$. Thus, they would only minimally contribute to S_{pool}. In addition, NA species with apparent nonphysical[69] K_d values got the minimum positive K_d from the pool assigned instead.

This measure also influenced the NN training, since the number of training samples was drastically reduced this way. This lack of data limits the complexity the model is able to learn. However, the aim was to still achieve a higher prediction accuracy, as the NA species filtered out would add substantial noise to the training data.

Furthermore, the K_d values were calculated from a single NA pool, i.e. the S_{pool} for each SELEX round would rely on K_d values from a common pool, preferably one obtained after multiple selection rounds, to reduce stochastic effects.

[64] Paszke et al. (2019).

[65] Kingma and Ba (2017).

[66] Akiba et al. (2019).

[67] defined as $\frac{p'_j}{p_j}$

[68] either before or after the respective round of selection

[69] negative

16.3 Results and discussion

16.3.1 Quality control

The quality of the sequencing data from the HT-SELEX experiments was controlled prior to further analysis. Only the quality control results for the *PM* dataset is exemplarily presented here. However, the observations are similar to the other datasets used in this work (see Section 16.2.1). Hence, the following discussion can be applied to these ones as well.

Although the SELEX employs no explicit sequence mutation step, 28.5 % of sequence reads deviates from the expected length[70] of 130 (Figure 16.7A). This observation is probably not caused by sequencing errors since overall sequencing quality is very high: The mean Phred score is consistently above 30 over all sequence positions (Figure 16.7B). A Phred score of 30 corresponds to a base call error probability of 0.1 %.

[70] length of designed NA including barcode

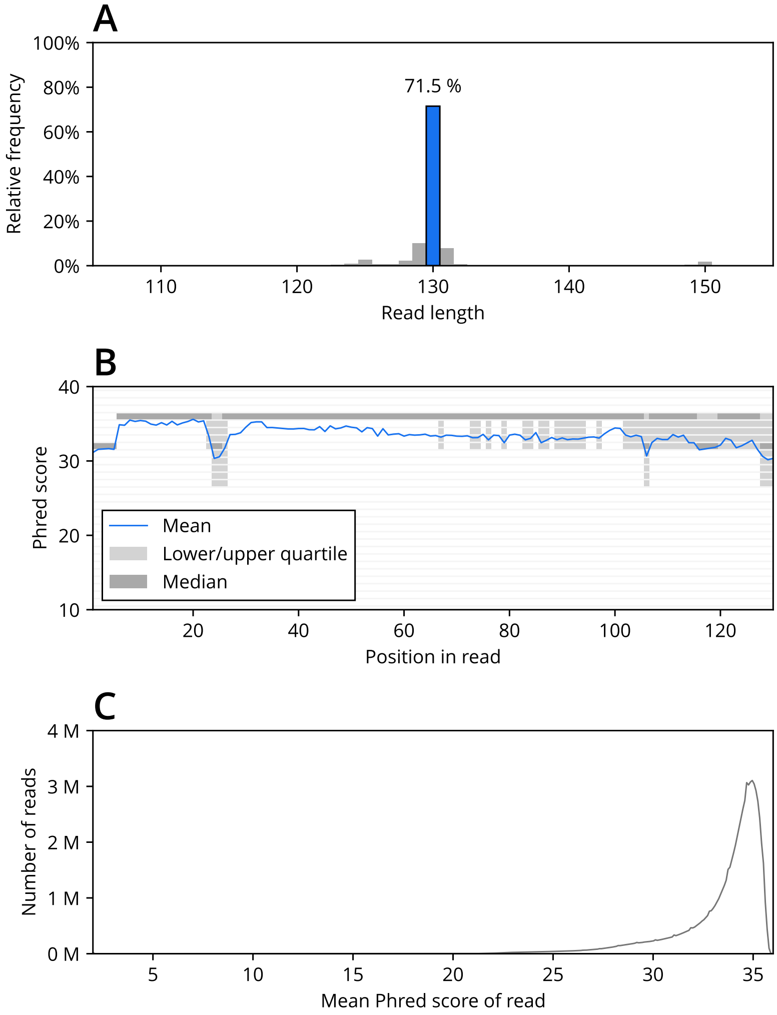


Figure 16.7: Quality control of sequence reads from *PM* dataset. A Histogram of read lengths. The expected length is marked in blue. **B** Positional Phred score distribution. For each read position the mean, median and quartiles are calculated over all reads. The displayed positions include the range of the expected sequence length. **C** Histogram of mean Phred scores. The Phred scores are averaged for each read.

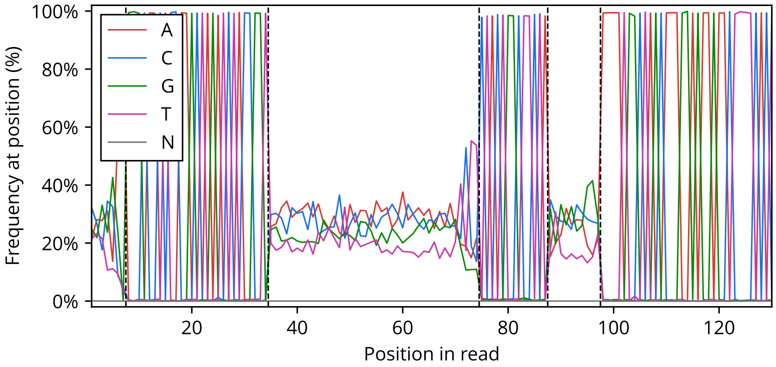


Figure 16.8: Nucleotide composition in sequence reads from *PM* dataset. For each nucleotide the relative frequency at the respective read position is shown. The vertical dashed lines depict transitions from a constant to a variable region and vice versa. Non-resolved nucleotides are depicted as 'N'.

The distribution of the average Phred score for each read (Figure 16.7C) is also regular and shows no additional peaks that would indicate erroneous sequencing. Hence, the significant number of deviations from the ideal sequence lengths are most likely caused by the SELEX itself during sequence amplification.

To evaluate the integrity of the NA library the sequence composition was analyzed (Figure 16.8). Since the sequencing data also includes reverse complement sequences, these were corrected first. The data contains almost no non-resolved nucleotides ('N'). The sequence composition clearly highlights the different sequence regions of the NA library: While most sequence reads are identical at positions corresponding to the constant parts, the variable regions show only a slight preference for certain nucleotides. The distribution in these randomized regions is still not uniform, as all selection rounds are incorporated in the dataset. Consequently, the nucleotides of enriched sequences appear more frequently.

Despite the low overall base call error probability, on average every 2200th nucleotide is called wrong. Consequently, this leads to appearance of virtual NA species, that do not reflect actually existing NA species in the pool, on approximately every 17th read. The reason for this is that any difference between two sequences means that these are treated as separate species. These virtual species cannot be distinguished from actual NA species: Although an erroneous read only has a small sequence difference to an existing species, such high similarity between two NA species might also be caused by mutation in the amplification process or by chance. However, for the sequence analysis discussed in the following sections this effect should be negligible. For computation of the pool score and pK_d prediction such spurious sequences are filtered out using the frequency threshold introduced in Section 16.2.8. For the calculation of the Gini index and the entropy these sequences only have a small effect due to their low frequency. Merely the percentage of orphans could be heavily biased by sequencing errors, because it is likely that one combination of sequencing errors only occurs once for a species.

16.3.2 Monitoring enrichment in a SELEX experiment

Based on the metrics discussed in Section 16.2.4, the enrichment of certain NA species[71] over the course of the selection rounds can be monitored (Figure 16.9). The four enrichment metrics show a clear trend during the course of the experiment: While the entropy and percentage of orphans decrease, there is an increase in Σ Top 100 and the Gini index. Strikingly, the trend is more clear in *PM* compared to *CX*. All metrics indicate a substantially higher enrichment in the SELEX experiment conducted with *PM*, despite a similar number of selection rounds. Furthermore, *CX* shows a declining enrichment in certain rounds according to some of the metrics, compared to the predominantly monotonic and consistent curve shape of *PM*. Apart from a different target, the two SELEX experiments have a significant difference in the selection protocol: *CX* uses a traditional SELEX, while *PM* employs a capture-SELEX. The presumption, that the difference in the selection protocol is paramount, is underpinned by the *CQ*, *LX* and *CL* datasets (Figure 16.10): All three datasets were obtained using capture-SELEX experi-

[71] and consequently depletion of most other species

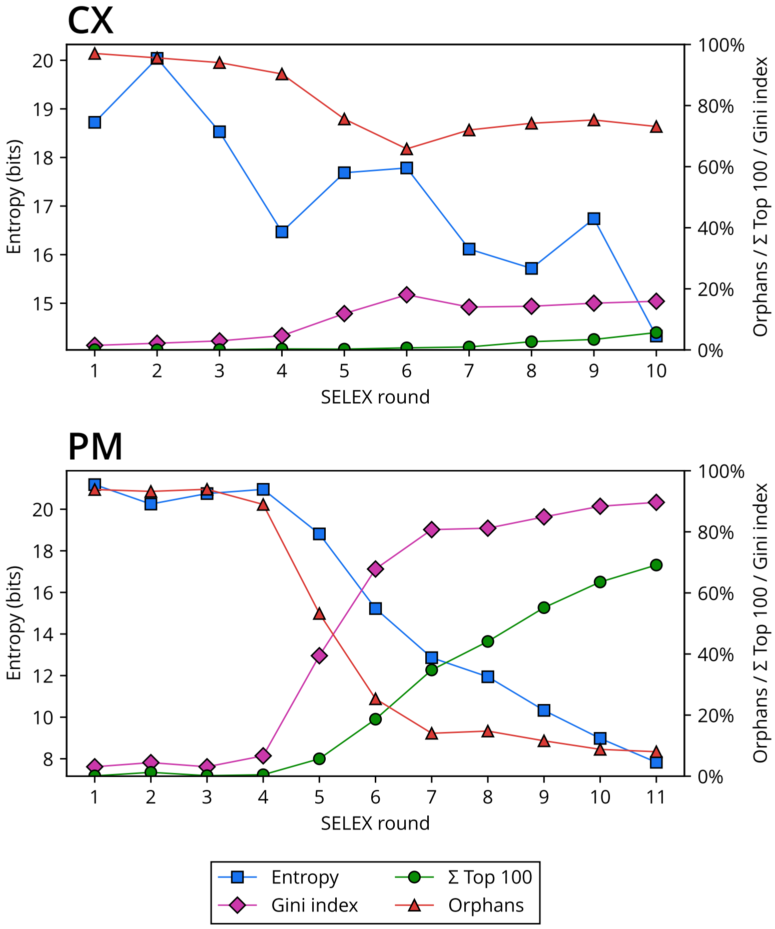


Figure 16.9: Sequence enrichment in *PM* and *CX* datasets using different metrics. The shown metrics were calculated from sequenced NA pools obtained after the displayed selection round.

ments, and although the targets are different and the experimenter is not the same as in *PM*, the degree of enrichment and the shape of the curve is similar to *PM*. This suggests that capture-SELEX provides more consistent results, with less required rounds on average to find an appropriate aptamer. However, for the traditional SELEX only the *CX* dataset is available. More such datasets would be necessary to confidently attribute the poor enrichment to the SELEX protocol and to exclude that other experimental parameters are the cause.

An advantage of observing the enrichment directly using HTS additionally to measuring the percentage of input NA eluted[72], is that this method reveals the enrichment of the NA in earlier selection rounds[73] and more precisely. Hence, the enrichment metrics could be used to decide after how many rounds the SELEX experiment should be terminated, if incorporated into a laboratory workflow.

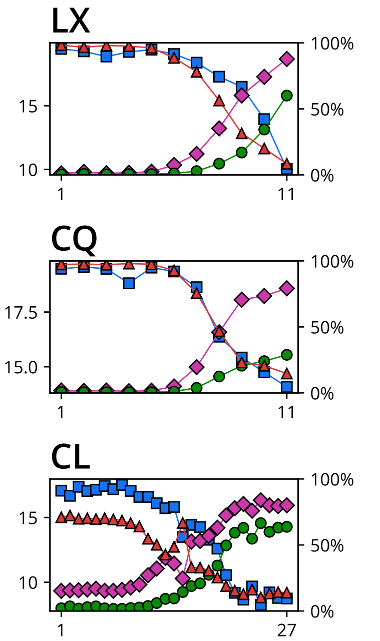


Figure 16.10: Sequence enrichment in *LX*, *CQ* and *CL* dataset. The labelling is analogous to Figure 16.9. For rounds with multiple tested elution conditions, only the result of the regular elution is displayed.

16.3.3 Comparison of different selection parameters

For *CQ* and *LX* different selection protocols were applied in round 10 to test their influence on sequence enrichment as displayed in Figure 16.11. Curiously, an upstream counter elution (Figure 16.11A) does not clearly increase the enrichment compared to a run omitting the counter elution (Figure 16.11B). This result implies that the upstream counter elution does not apply significant selection pressure, i.e. the NA species with sufficient affinity for the target do not bind well to the respective counter target. Consequently, this leads to the assumption that an elution with the counter target[74] (Figure 16.11D) would enforce a strong selection pressure. However, this is not the case: The enrichment is in a similar order as the enrichment from the elution with the target. Probably the selection conditions need to be sustained for multiple rounds of selection to be able to see clear effects of a counter elution.

The enrichment metrics also indicate that elution with a lowered concentration of target (Figure 16.11C) and elution over a short amount of time (Figure 16.11E) lead to a stronger enrichment. The former result is consistent with the mathematical model of the SELEX[75] as seen in Equation 16.16. The latter observation indicates that k_{on}, the rate constant of the aptamer-target association, is a discriminating factor for aptamer affinity in this case. This finding is interesting, since kinetic studies on aptamers for other targets indicate in contrast, that aptamers with differing affinity for the same target mostly vary in k_{off}, while k_{on} is similar between these aptamers[76]. However, in this case, not only affine aptamers compete with each other in the SELEX experiment, but the RNA pool includes also a large percentage of non- or weakly-binding RNA species. Therefore, k_{on} might substantially differ between affine and non-affine RNA species, while this value is similar among the more affine species. Since no sufficiently affine aptamers from these experiments were obtained and kinetically characterized, the exact explanation of this finding remains to be answered.

[72] equivalent to θ defined in section 16.2.5

[73] Groher et al. (2018); Boussebayle et al. (2019).

[74] Note that in case of elution the eluate is collected instead of discarded.

[75] Levine and Nilsen-Hamilton (2007).

[76] Bao et al. (2011); Chang et al. (2014).

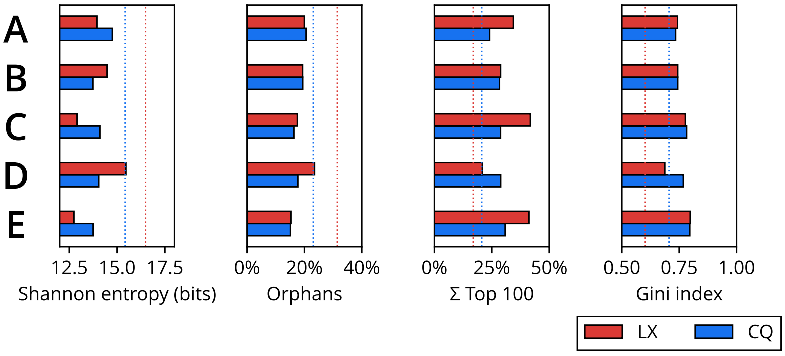


Figure 16.11: Effect of selection parameters on sequence enrichment for *LX* and *CQ*. Different selection protocols were applied in round 10 and the sequence enrichment was measured from the resulting RNA pool. The dashed line depicts the value for the respective enrichment metric from the previous round. If not stated otherwise, elutions were conducted over 5 min. **A** Counter elution with 1 mM CX/AQ, elution with with 1 mM LX/CQ. **B** Elution with with 1 mM LX/CQ. **C** Elution with with 100 µM LX/CQ. **D** Elution with 1 mM CX/AQ. **E** Elution with with 1 mM LX/CQ over 10 s.

Nevertheless, it needs to be noted that all tested selection protocols enriched the pool in a measurable manner. However, also note that the enrichment itself does not tell whether the enriched RNA species have the desired binding behavior. Since the *CQ* and *LX* SELEX projects did not conclude in a satisfactory aptamer, it cannot be investigated under which selection condition such an aptamer accumulated the most.

16.3.4 Applicability of mathematical model

The mathematical model of the SELEX process[77] gives a foundation to relate the relative frequencies of NA species to their respective dissociation constants. Based on an initial pool of species with uniform distribution, the first selection round would enrich species according to their individual $K_{d,i}$ (Equation 16.16). Here we denote the relative frequencies after the nth round as p_i^n. Thus,

$$\frac{p_i^1}{p_j^1} = \frac{K_{d,j} + [T]}{K_{d,i} + [T]}. \tag{16.22}$$

The relative frequencies in the following round are calculated as

$$\begin{aligned} \frac{p_i^2}{p_j^2} &= \frac{p_i^1}{p_j^1}\frac{K_{d,j} + [T]}{K_{d,i} + [T]} \\ \frac{p_i^2}{p_j^2} &= \left(\frac{K_{d,j} + [T]}{K_{d,i} + [T]}\right)^2. \end{aligned} \tag{16.23}$$

Thus, in general

$$\frac{p_i^n}{p_j^n} = \left(\frac{K_{d,j} + [T]}{K_{d,i} + [T]}\right)^n. \tag{16.24}$$

Furthermore, one can define the amplification of an NA species $A_i = p_i^{n+1}/p_i^n$. Again, using Equation 16.16

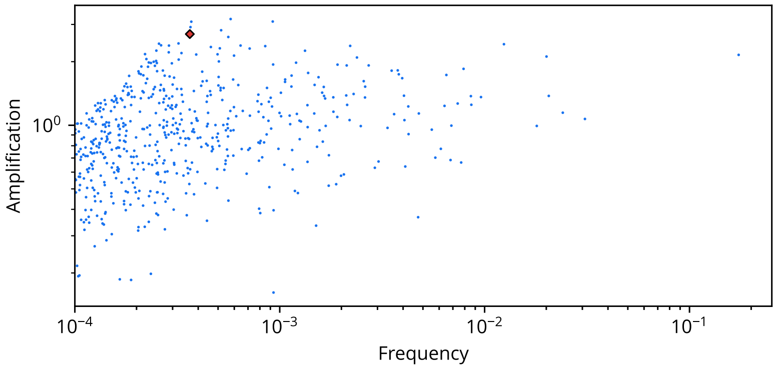
$$\frac{A_i}{A_j} = \frac{K_{d,j} + [T]}{K_{d,i} + [T]}, \tag{16.25}$$

which can be substituted in Equation 16.24 to obtain

$$\frac{p_i^n}{p_j^n} = \left(\frac{A_i}{A_j}\right)^n. \tag{16.26}$$

[77] Levine and Nilsen-Hamilton (2007).

Figure 16.12: Amplification and frequency of RNA species in *PM* round 9. For each species i with a relative frequency of at least 0.01 %, this figure shows its amplification A_i in round 9 and its relative frequency after the selection round. This selection round was the last round prior to counter selection. The riboswitch candidate is marked in red.

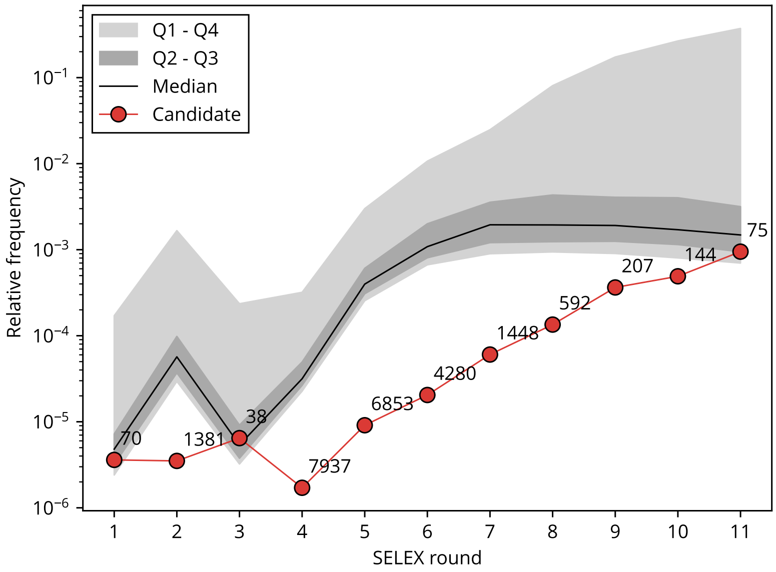


Hence, the amplification A_i of an NA species during a selection round is in theory a monotonically increasing function of its relative frequency p_i. This trend is further enhanced by an increased number of selection rounds n. A change in target concentration for the elution would merely intensify or weaken the relative amplification A_i/A_j, the monotonicity would still uphold.

However, the sequencing data from the *PM* SELEX shows substantial noise in this theoretically monotonic relation as displayed in Figure 16.12. After nine rounds of selection the correlation between A_i and p_i is weak (0.35) despite the mitigating measures explained in Section 16.2.8. Since Equation 16.26 does not indicate a linear relation, the *Spearman* rank correlation coefficient was employed here.

When taking a closer look at the relative frequency of the riboswitch candidate for *PM* over the selection rounds (Figure 16.13), this RNA

Figure 16.13: Enrichment of the *PM* riboswitch candidate over the course of selection rounds. The figure shows the relative frequency of the riboswitch candidate in the respective selection round in red. The corresponding rank is given as label. For comparison the median and quartiles (Q1 - Q4) of the Top 100 sequences in the respective round are also displayed.



species does not accumulate up to round 4. Only after round 5 the exponential increase suggested by Equation 16.24 can be assumed. This is remarkable, since the selection conditions[78] do not change until round 7. Hence, the experimental data contradicts the theory: Under constant target concentration one would expect an initial exponential increase until the relative frequency levels out due to only similarly affine aptamers remaining in the RNA pool.

The large amount of noise in the experiment is not limited to *PM* but also appears in all other datasets discussed in this work as seen in Figure 16.14, with correlations of 0.71, 0.66, 0.56 and 0.17 for *CX*, *LX*, *CQ* and *CL*, respectively. The effect extends to other experimental setups: A large study to determine DNA binding specificities for transcription factors (TFs) via HT-SELEX [79] yielded 520 HTS datasets[80]. Based on the respective final selection round for each TF the correlation between A_i and p_i was calculated. Datasets, where the correlation p-value was higher than 5 % or where the number of data points was less than 200, were filtered out. The correlations for the remaining 108 HTS datasets are displayed as histogram in Figure 16.15. From this data three exemplary amplification plots, analogous to Figure 16.12, are shown in Figure A.2. Although with an average correlation of 0.73 the results are closer to the mathematical description than the other SELEX experiments discussed above, they still clearly deviate from a monotonic relation between A_i and p_i. Note that since DNA binding studies were performed here, no reverse transcription and transcription steps were employed.

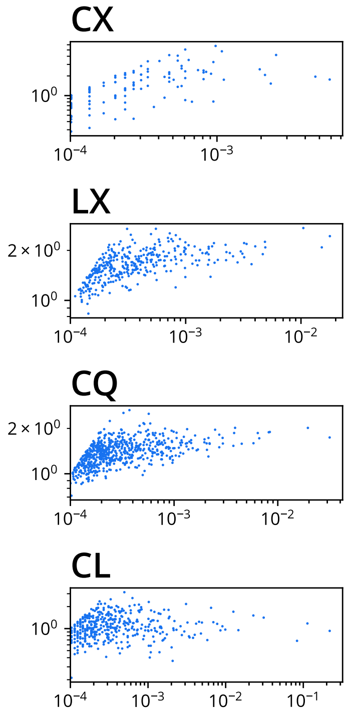


Figure 16.14: Amplification and frequency of RNA species. The labelling is analogous to Figure 16.12. The following rounds were chosen for visualization: **CX** round 10, **LX** round 9, **CQ** round 9, **CQ** round 26.

These observations suggest, that the large amount of noise is not due to experimental deficiencies but is inherent to most SELEX experiments. A probable reason is that at least one of the assumptions listed in Section 16.2.5 is not realistic. One explanation could be stochastic effects especially in the amplification via PCR: Due to the exponential nature of this amplification, small stochastic deviations could have a large effect[81]. Furthermore, it was previously suggested that different NA species might not be equally amplified[82] and that also the initial pool might be biased[83]. By employing SELEX-T or ddPCR[84] the bias introduced in the amplification step can potentially be reduced.

A notable consequence of the deviation from the theory are apparent negative K_d values. As described in Section 16.2.5, those nonphysical values may not appear in the ideal model, since there are certain limits to θ and A_i. However, if θ is not measured with perfect accuracy or A_i deviates due to stochastic effects, such negative K_d values appear as shown in Table 16.1. Nonphysical values appear most prominently for sequences with low frequency. Presumably, stochastic effects are more dominant for NA species at low numbers. This finding emphasizes the importance of the frequency threshold described in Section 16.2.8, that is used to calculate S_{pool}.

[78] including the target concentration

[79] Jolma et al. (2013).

[80] NCBI BioProject ID: PRJEB3289

[81] Spill et al. (2016).

[82] Thiel et al. (2011); Tsuji et al. (2009); Takahashi et al. (2016).

[83] Takahashi et al. (2016).

[84] Tsuji et al. (2009); Takahashi et al. (2016).

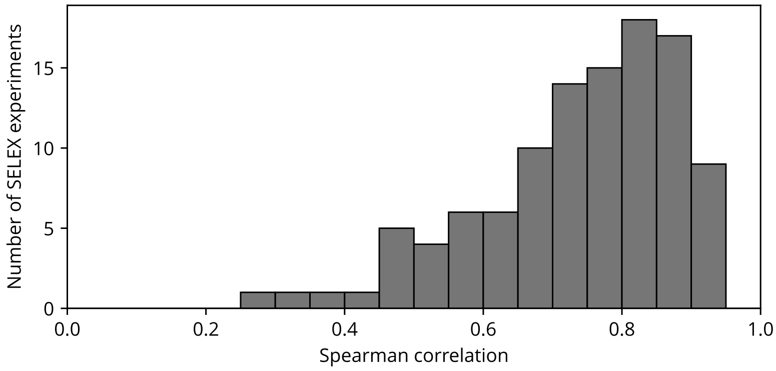


Figure 16.15: Correlation between amplification and frequency of DNA species in TF SELEX. The histogram shows the correlation between A_i and p_i for the last respective selection round in 108 SELEX experiments with TFs as target molecule.

Another striking aspect of the SELEX experiments discussed here are the dissociation constants of the obtained aptamers in comparison with the target concentration used for elution: As example, the *PM* project resulted in an aptamer with $K_d = 21\,\text{nM}$, while the minimum target concentration used for elution was 100 µM, which was applied for 4 rounds. However, based on the presented theory an enrichment of such an affine aptamer should not be possible under the applied selection pressure: All RNA species with a $K_d \ll 100\,\mu\text{M}$ should be almost completely eluted, so no discrimination between different well-binding species is gained. As sample calculation, let the aptamer i with $K_d = 21\,\text{nM}$ be in a pool together with an RNA species j with $K_d = 10\,\mu\text{M}$ at equal frequency. According to Equation 16.24 the ratio p_i/p_j after four rounds of selection with 100 µM target would only be 1.46. In a realistic scenario, where such a affine aptamer does not appear at equal frequency to worse-binding species, but is more like a *'needle in a haystack'*, an enrichment by a factor of 1.46 is negligible. In reality, a strong enrichment of the discussed aptamer against other species is clearly visible in the *PM* dataset. For this reason the estimated dissociation constant for this aptamer based on selection round 9 using Equation 16.17 is 78 µM, which is several orders of magnitude higher than 21 nM. It is improbable that inaccuracies in the measurement of θ alone cause this difference, though a more precise determination of θ could give insights into this issue.

Table 16.1: Estimated nonphysical K_d values in *PM*. For each selection round the K_d for each RNA species was estimated based on the sequencing data and the measured eluted percentage. Only species whose sequences also appeared in the previous round were considered. The number of species where the calculation yields a nonphysical (negative) K_d value is given, compared to the total number of species and split by the threshold value.

Round	$\geq 0.01\,\%$		$< 0.01\,\%$	
	nonphysical	total	nonphysical	total
2	0	2	12	563
3	0	3	0	1170
4	0	2	171	7023
5	5	8	3494	81142
6	0	409	896	112519
7	0	794	368	80748
8	0	625	292	62859
9	0	557	1203	48672

Despite the disagreements with the mathematical foundation, it is clear that the SELEX method provides a reasonable enrichment of affine aptamers, since the *PM* and *CX* projects yielded aptamers with dissociation constants in the nanomolar range[85]. Hence, this work assumes that metrics that are drawn from the entirety of a pool, including the enrichment metrics and S_{pool}, are still roughly valid at least for qualitative assessment. Nevertheless, the investigation of the cause for the presented significant discrepancies between theory and experiment could be worthwhile to better understand and optimize SELEX processes.

16.3.5 Selection of an appropriate SELEX round for in vivo studies

This work leverages estimated individual K_d values for NA species to calculate S_{pool}, a metric that should provide decision guidance for the selection of an appropriate NA pool for further screening, especially for *in vivo* studies. S_{pool} depends on a number of parameters that need to be properly adjusted: Most importantly, the selection round and frequency threshold for K_d calculation need to be chosen. If the experiment would perfectly adhere to the theory elaborated in Section 16.2.5, the round would not have an impact and a frequency threshold would not be necessary. However, as Section 16.3.4 showed that there is a strong deviation between theory and experiment, these two parameters have an impact. The final parameter is the number of samples λ that are taken for further assays. This value solely depends on the experimental setup, e.g. if *in vivo* assays are conducted on a 96-well plate, $\lambda = 96$. To investigate the effect of these parameters on S_{pool} for the *PM* dataset, each parameter was scanned across a reasonable range, while the other parameters were kept constant.

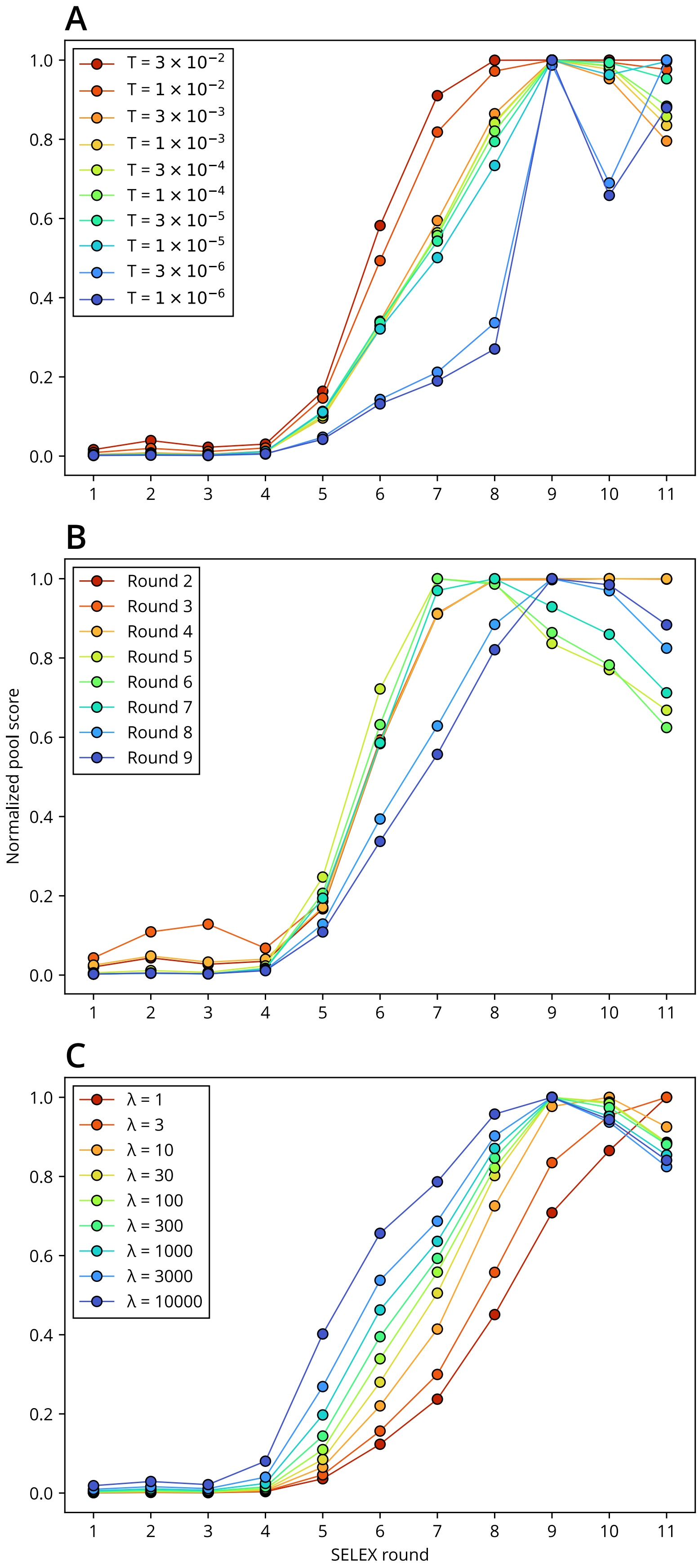
At a high threshold only the most frequent species are considered. Hence, it is expectable that S_{pool} maximizes at maximum enrichment[86] at a 10 % threshold (Figure 16.16A). At lower threshold values S_{pool} penalizes the high enrichment of few species, since consequently other less frequent but still promising species have a low probability of being sampled. Hence, earlier selection rounds that are already sufficiently enriched become more favorable. This trend continues until a threshold of approximately 0.01 %. Below this threshold the S_{pool} curve becomes more erratic. This is expected, since NA species with low numbers are included, resulting in stochastic effects.

Next, the different rounds for K_d calculation were tested (Figure 16.16B): All rounds starting from round 2 to 9 were included in the range. Using round 1 was not possible, since no sequencing was performed for the original pool. Round 10 and 11 included a counter selection, thus they cannot be soundly described by the theory from Section 16.2.5. S_{pool} shows a tendency to overestimate the score for the round, where the K_d values are taken from. The reason for this effect is that species that were strongly amplified in a round are overall more abundant in this

[85] *CX*: 60 nM (after optimization), *PM*: 21 nM

[86] according to the established metrics from Section 16.2.4

Figure 16.16: Effect of parameters on S_{pool} calculation in *PM*. Each plot shows the effect of one parameter for S_{pool} calculation by scanning the parameter over a range of values while keeping all other parameters constant. For improved comparability, S_{pool} is normalized to the maximum of the respective curve. If not stated otherwise, K_d values from *PM* round 9 with a frequency threshold of 0.01 % and $\lambda = 96$ samples were taken for S_{pool} calculation. **A** The threshold for the minimum relative frequency is scanned, i.e. NA species with a relative frequency smaller than the threshold value are not considered for S_{pool} calculation. **B** The round from which the K_d values are calculated is scanned. **C** The number of samples λ is scanned.



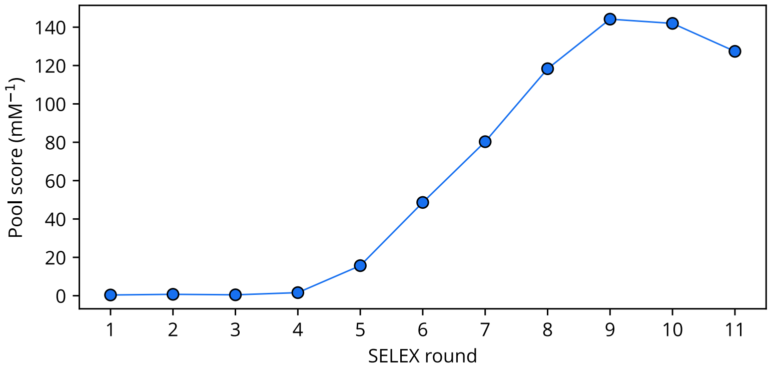


Figure 16.17: S_{pool} for each *PM* round. S_{pool} was calculated for each selection round in the *PM* dataset, based on K_d values from round 9 and $\lambda = 96$ samples.

round. Still, round 8 or 9 have the highest S_{pool} regardless of the round, that was taken as basis for calculation. An exception is round 5 that maximizes S_{pool} for round 7, though Table 16.1 shows that this round has a high number of species with apparent nonphysical K_d, so their respective S_{pool} are doubtful. Since stochastic effects on K_d are less significant for species with larger numbers in a pool, it is suggested to use a late round for K_d calculation, since in these rounds, generally speaking, species with relevant K_d appear in high abundance.

Finally, the impact of λ is investigated (Figure 16.16C). The displayed array of curves show a clear trend: At an increasing number of samples, selection rounds with lower enrichment become more favorable. However, even for an unrealistically high number of samples ($\lambda = 10000$) round 9 still clearly provides the optimal pool according to this metric.

As result of these studies, round 9 with a threshold of 0.01 % was chosen for K_d calculation. Consistent with the downstream *in vivo* studies for *PM*, λ was set to 96. Based on this parametrization round 9 provides the most suitable RNA pool according to S_{pool} as displayed in Figure 16.17. Interestingly, this result is close to round 8, which Boussebayle *et al.* intuitively considered as sufficiently enriched based on the established enrichment metrics[87].

With the library score a mathematical foundation is at hand that balances the two objectives *'high enrichment'* and *'still enough diversity'* required for successful *in vivo* selection of well performing riboswitches. However, whether this score is a reasonable decision tool for SELEX round selection still needs to be evaluated in future laboratory experiments.

16.3.6 Prediction of aptamer sequences

Since a dissociation constant can be calculated for any sufficiently enriched NA species in the HT-SELEX experiment, these sequence-affinity pairs can be used as training set for supervised learning to pre-

[87] Boussebayle et al. (2019).

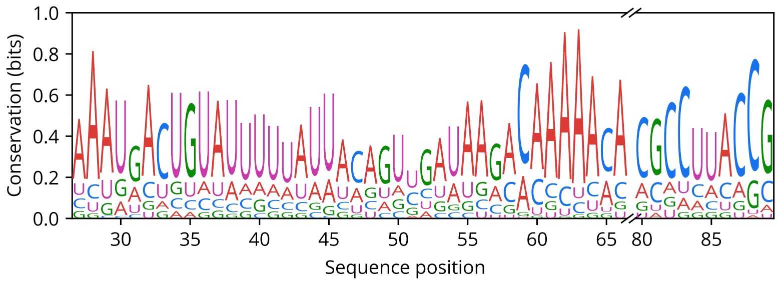


Figure 16.18: Conservation of the Top 100 sequences after selection. The sequence logo shows the sequence conservation in the variable regions of the most frequent RNA species in the *CL* dataset after the final selection round. The conservation is limited to 2 bits at complete sequence identity. Note that occurrences of nucleobases were not weighted by the frequency of the RNA species.

dict a dissociation constant for unknown sequences.

This approach was used to conduct *in silico* mutation scanning on the *CL* aptamer candidate with a previously *in vitro* measured[88] $K_d \approx 13\,\mu\text{M}$ to improve its binding affinity to the target. Such a prediction necessarily requires recurring sequence motifs across the affine NA species. The sequence conservation of the Top 100 NA species after the final selection round (Figure 16.18) shows that certain nucleobases are positionally conserved. In addition, an NN is potentially able to also detect patterns that are hidden from simple positional analysis.

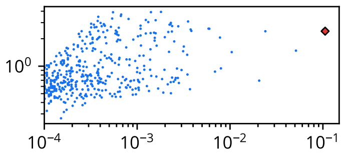


Figure 16.19: Amplification and frequency of RNA species in *CL* round 20. The labelling is analogous to Figure 16.12.

For the purpose of K_d prediction, the RNA species frequencies from selection round 20 were chosen, as a large enrichment change was still observable in this round according to the presented enrichment metrics (Figure 16.10). This ensures the presence of RNA species in sufficient frequency, while also the binding affinities of the present RNA species, determined from the species amplification in this round, was expected to be diverse (Figure 16.19). For each of these RNA species the pK_d was calculated and used as the expected output for NN training.

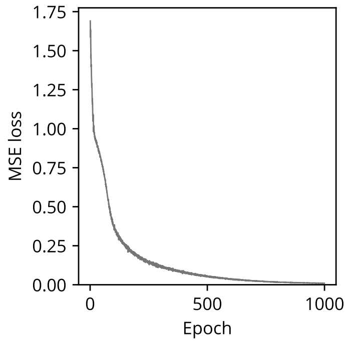


Figure 16.20: Course of the MSE loss during NN training. A moving average with a frame of ten batches was applied for clarity.

The NN trained with the optimum hyperparameters[89] converged to an MSE of 0.013 (Figure 16.20). The used hyperparameters are listed in Table A.3. This high accuracy is not surprising due to the low number of training samples. Indeed, a slight overfit on the training data is observed, since the MSE is 0.218 on the validation set and even 0.442 on the final test set. However, the MSE on the test set is still low enough to assume an accurate prediction of the dissociation constant within the correct order of magnitude. Despite the candidate aptamer sequence not being part of the training set, the NN predicted a $K_d = 4.22\,\text{mM}$ for this sequence very close to the calculated $K_d = 4.17\,\text{mM}$.

Based on the trained NN, the effect of all possible single and double mutations[90] of the aptamer candidate sequence on its binding affinity were tested. The ΔpK_d gives the difference between the predicted pK_d of the mutant and the original sequence, where positive values denotes an improvement of binding affinity. Figure 16.21 shows the results for all single mutations. A ranked list of the double mutants with the high-

[88] Suess *et al.* (unpublished data).

[89] according to the MSE of the validation set

[90] substitutions and deletions

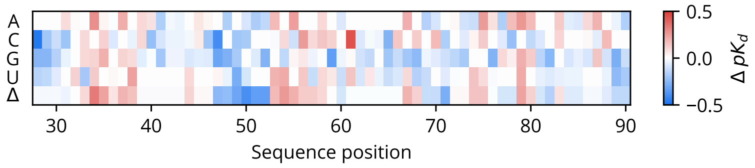


Figure 16.21: Effects of single mutations on predicted pK_d. The figure shows the effect of deletion (Δ) and substitutions on predicted affinity for each sequence position. As primer binding regions were not part of the NN input, they are omitted. Beneficial mutations are highlighted in red, detrimental ones in blue.

est ΔpK_d is given in Table A.4.

Two promising single and double mutants were selected for *in vitro* analysis of binding affinity: A61C, Δ54, Δ54-A61C and G53A-A61C. For this purpose a selection round was conducted with a single RNA species, for each of the mutants in duplicate. To measure the elution percentage θ the RNA was radiolabelled [91]. Figure 16.22 shows the results of this measurement. Via equation 16.8 θ can be related to the dissociation constant of the RNA species. Only the A61C mutant shows affinity to cortisol comparable to the original riboswitch candidate. This variant was the single mutant with the highest predicted $\Delta pK_d = 0.44$. The Δ54 variant shows no measurable affinity to the target anymore. Consequently, the double mutant Δ54-A61C also lost its binding behavior. The lost binding affinity of the G53A-A61C variant indicates that G53A is also a loss-of-function mutation, though this single mutation was not tested.

Although no improvement on the affinity of the aptamer candidate was achieved, the single mutant with the best predicted affinity performed comparable to the original candidate in the laboratory experiment. Multiple factors may impede the prediction accuracy. First, the small number of training samples of merely a few hundred NA species leads to at least slight model overfitting and prevents the employment of more complex NN designs. Second, the labels of the training dataset itself contain substantial noise as explained in Section 16.3.4: Even if the prediction is accurate within the described framework for K_d computation, these computed K_d values may not be underpinned by lab-

[91] Suess *et al.* (unpublished data).

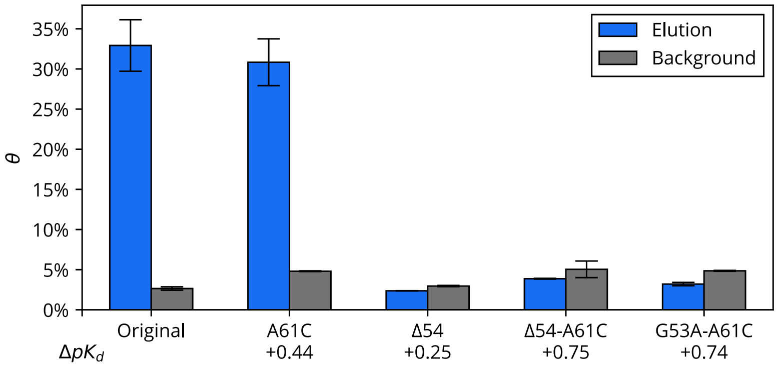


Figure 16.22: Elution percentage of *CL* aptamer candidate and tested mutants. For each variant the predicted ΔpK_d is given. The bar shows the average of the duplicate measurements, while the error bar depicts the deviation to the individual measurements. Elution was performed with 1 mM cortisol. The background is the signal from the third washing step prior to elution with the target.

oratory experiments. Hence, the prediction would benefit from an increased measurement accuracy for θ and a reduced bias in the enrichment during the SELEX experiment. With decreased noise the frequency threshold could also be lowered, potentially expanding the size of the training dataset.

16.4 Conclusion

A sound estimation of affinity to the target for all relevant NA species after selection rounds allows deeper insights into the progress of a SELEX experiment. The dissociation constant is an intrinsic property of an aptamer-target-complex in contrast to raw sequence counts or the frequency ratio of an NA species between two selection rounds which are specific to the SELEX experiment. This work presented a method to calculate such affinities from HTS data and elution percentage, without the need for costly assays such as isothermal titration calorimetry or fluorescence titration. This method can be possibly extended to binding studies of protein-DNA complexes[92] via HT-SELEX to not only identify sequence specificities but also to quantify the dissociation constant. In addition, this data opens potential opportunities for selection of reasonably enriched NA pools and aptamer optimization, as this chapter outlined. The prediction of affinities for aptamer mutants can be potentially improved in the future, since the dissociation constants can be simply integrated into existing deep-learning approaches, which are rapidly evolving in the present.

Note that a reliable application of the presented methods ideally requires an accurate data foundation. However, the analyzed HT-SELEX datasets indicate that typical SELEX experiments contain a substantial amount of noise in the sequence data and that the measured NA elution percentage does not match the magnitude expected from the NA affinities determined by other experiments. Therefore, a thorough investigation and elimination of the discrepancies between theory and experiment could enhance the knowledge gained from HT-SELEX experiments and increase the probability of finding a high-performance aptamer.

[92] for example TFs

Part IV

Conclusion

The central quest of the *Biotite* project is to build a easy-to-use library that can be integrated into tools, scripts and other libraries to solve actual biological questions. This is supposed to simplify the implementation of recurrent tasks in bioinformatics and in consequence shorten the time between a researchers' idea and the scientific insight. For this purpose *Biotite* offers a broad range of functions working on the same consistent data representation based on the widely used *NumPy* arrays. This allows an almost infinite number of combinations of these functions to tackle a wide range of problems in sequence and structure analysis. This thesis and by now also other works have demonstrated how the original vision of *Biotite* is realized.

Hydride (Chapter 14) extensively uses *Biotite* in underlying routines, but itself is a focused command-line tool - in line with the idea of the '*Small tools manifesto for bioinformatics*'[93]. *Springcraft* (Chapter 11) shows how functionality from *Biotite* in combination with *NumPy* can be used to implement the logic of elastic network models (ENMs) in a straightforward way, beating even the performance of established software in this field[94]. And finally, the analyses presented in Part III show how the flexibility of *Biotite* and its extensions can be used in scripts tailored to specific biological questions. Since the initial release also other projects started to use *Biotite* in the intended way. The *TeachOpenCADD* platform[95] teaches computer-aided drug design using *Python* scripts and incorporates *Biotite* for part of the analyses. The projects *Foldingdiff* and *ESM* use *Biotite* for underlying structure representation and analysis with the aim to generate biologically plausible protein backbone structures or to solve the inverse protein folding problem, respectively[96].

In contrast to tools and libraries that offer a focused set of functions to solve a certain class of problems, a broad library such as *Biotite* is able to utilize synergies between the individual functionalities: A tool based on *Biotite* is able to support a wide range of file formats out of the box, only the unique program logic needs to be implemented. A library that calculates hydrogen bonds using *Biotite* (Chapter 12) can enhance the accuracy by predicting hydrogen positions first (Chapter 14). A script that calculates normal modes using an ENM (Chapter 11) can visualize them without much further effort with *Ammolite* (Chapter 4).

In addition to offering the bioinformatics community a versatile programming library to address concrete questions, this thesis itself also presented novel scientific findings obtained by the application of *Biotite* and its extension packages. As a use case for structural analyses ENMs implemented with *Springcraft* were combined with general structure-related functionalities from *Biotite* to study conformational changes in HCN4 upon ligand binding (Chapter 15). The computational investigations confirmed experimental observations that ligand binding in the tetrameric HCN channels is cooperative and proposed a putative order in which the ligand binds to the subunits. As a future prospect the abil-

[93] Prins (2014).

[94] **Kunzmann** et al. (2022).

[95] Sydow et al. (2019).

[96] Wu et al. (2022); Hsu et al. (2022).

ity to study binding cooperativity using ENMs may reveal the essential regions within the protein that mechanically transmits the conformation change induced by ligand binding between binding sites.

The sequence analyses of HT-SELEX data (Chapter 16) harnessed the file interfaces and the *NumPy*-based sequence representation from *Biotite*. It was shown how sequencing data from a SELEX experiment can be potentially used to estimate binding affinities for each species present in the nucleic acid library. Furthermore, approaches were presented how this knowledge may allow better selection of SELEX rounds for *in vivo* experiments or even the prediction of aptamers with improved binding affinities. However, it was also shown how in reality SELEX experiments deviate from the idealized theoretical foundation, hampering those applications. Dedicated laboratory experiments implementing remedies for these deviations would be needed to investigate whether the outlined problems can be overcome to unleash the unexploited potential of HT-SELEX data.

Biotite follows good practices in software development. Most of its components are comprehensively tested to ensure correctness of its functions. Every public function is documented to make the library not only accessible to its active developers, but to a larger base of users. A tutorial and application examples on the official documentation website provide an introduction for newcomers. The *Biotite* package is distributed for all common operating systems, including *Windows*, *MacOS* and *Linux*, via both major package managers *Conda* and *pip*. Furthermore, it is developed under an open-source license. First, this makes algorithms, whose initial software implementation is not easily or freely available, accessible to the public. Second, it encourages external developers to contribute to the future progression of the project. With these presented measures *Biotite* commits to a sustainable bioinformatics ecosystem leading to reproducible scientific results.

References

Ackers, G. K. and Holt, J. M. (2006)
Asymmetric Cooperativity in a Symmetric Tetramer: Human Hemoglobin *. *Journal of Biological Chemistry* 281.17, pp. 11441–11443. DOI: 10.1074/jbc.R500019200 (cited on p. 100).

Adams, P. D., Afonine, P. V., Baskaran, K., Berman, H. M., Berrisford, J., Bricogne, G., Brown, D. G., Burley, S. K., et al. (2019)
Announcing Mandatory Submission of PDBx/mmCIF Format Files for Crystallographic Depositions to the Protein Data Bank (PDB). *Acta Crystallographica Section D: Structural Biology* 75.4, pp. 451–454. DOI: 10.1107/S2059798319004522 (cited on pp. 14, 89).

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019)
Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631. DOI: 10.1145/3292500.3330701 (cited on pp. 25, 118).

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. (2015)
Predicting the Sequence Specificities of DNA- and RNA-binding Proteins by Deep Learning. *Nature Biotechnology* 33.8, pp. 831–838. DOI: 10.1038/nbt.3300 (cited on pp. 116, 117).

Allen, F. H., Kennard, O., Watson, D. G., Brammer, L., Orpen, A. G., and Taylor, R. (1987)
Tables of Bond Lengths Determined by X-ray and Neutron Diffraction. Part 1. Bond Lengths in Organic Compounds. *Journal of the Chemical Society, Perkin Transactions 2* 12, S1–S19. DOI: 10.1039/P298700000S1 (cited on p. 76).

Altschul, S. F. (1991)
Amino Acid Substitution Matrices from an Information Theoretic Perspective. *Journal of Molecular Biology* 219.3, pp. 555–565. DOI: 10.1016/0022-2836(91)90193-A (cited on p. 43).

Altschul, S. F. and Erickson, B. W. (1986)
A Nonlinear Measure of Subalignment Similarity and Its Significance Levels. *Bulletin of Mathematical Biology* 48.5, pp. 617–632. DOI: 10.1007/BF02462327 (cited on p. 45).

Altschul, S. F. and Gish, W. (1996)
Local Alignment Statistics. *Methods in Enzymology*. Computer Methods for Macromolecular Sequence Analysis 266, pp. 460–480. DOI: 10.1016/S0076-6879(96)66029-7 (cited on pp. 42, 46).

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990)
Basic Local Alignment Search Tool. *Journal of Molecular Biology* 215.3, pp. 403–410. DOI: 10.1016/S0022-2836(05)80360-2 (cited on pp. 16, 39).

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997)
Gapped BLAST and PSI-BLAST: A New Generation of Protein Database Search Programs. *Nucleic Acids Research* 25.17, pp. 3389–3402. DOI: 10.1093/nar/25.17.3389 (cited on p. 41).

Amaro, R. E., Baudry, J., Chodera, J., Demir, Ö., McCammon, J. A., Miao, Y., and Smith, J. C. (2018)
Ensemble Docking in Drug Discovery. *Biophysical Journal* 114.10, pp. 2271–2278. DOI: 10.1016/j.bpj.2018.02.038 (cited on p. 61).

Andrews, S. (2010)
FastQC: A Quality Control Tool for High Throughput Sequence Data. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ (cited on p. 108).

Antczak, M., Popenda, M., Zok, T., Zurkowski, M., Adamiak, R. W., and Szachniuk, M. (2018)
New Algorithms to Represent Complex Pseudoknotted RNA Structures in Dot-Bracket Notation. *Bioinformatics* 34.8, pp. 1304–1312. DOI: 10.1093/bioinformatics/btx783 (cited on p. 77).

Antczak, M., Zok, T., Popenda, M., Lukasiak, P., Adamiak, R. W., Blazewicz, J., and Szachniuk, M. (2014)
RNApdbee—a Webserver to Derive Secondary Structures from Pdb Files of Knotted and Unknotted RNAs. *Nucleic Acids Research* 42.W1, W368–W372. DOI: 10.1093/nar/gku330 (cited on p. 74).

Anter, J. M. (2021)
Biology Department Computational Biology and Simulation Secondary structure analysis of nucleic acid structures and sequences (cited on p. 57).

Al-Aqtum, M. T. and Al-Qawasmeh, M. H. (2001)
Prevalence of Colour Blindness in Young Jordanians. *Ophthalmologica* 215.1, pp. 39–42. DOI: 10.1159/000050824 (cited on p. 27).

Asif, M. and Orenstein, Y. (2020)
DeepSELEX: Inferring DNA-binding Preferences from HT-SELEX Data Using Multi-Class CNNs. *Bioinformatics* 36.Supplement_2, pp. i634–i642. DOI: 10.1093/bioinformatics/btaa789 (cited on p. 116).

Atilgan, A. R., Durell, S. R., Jernigan, R. L., Demirel, M. C., Keskin, O., and Bahar, I. (2001)
Anisotropy of Fluctuation Dynamics of Proteins with an Elastic Network Model. *Biophysical Journal* 80.1, pp. 505–515. DOI: 10.1016/S0006-3495(01)76033-X (cited on p. 66).

Bahar, I., Atilgan, A. R., and Erman, B. (1997)
Direct Evaluation of Thermal Fluctuations in Proteins Using a Single-Parameter Harmonic Potential. *Folding and Design* 2.3, pp. 173–181. DOI: 10.1016/S1359-0278(97)00024-2 (cited on p. 66).

Bahar, I., Lezon, T. R., Bakan, A., and Shrivastava, I. H. (2010)
Normal Mode Analysis of Biomolecular Structures: Functional Mechanisms of Membrane Proteins. *Chemical reviews* 110.3, pp. 1463–1497. DOI: 10.1021/cr900095e (cited on pp. 65, 66, 97).

Baker, E. N. and Hubbard, R. E. (1984)
Hydrogen Bonding in Globular Proteins. *Progress in Biophysics and Molecular Biology* 44.2, pp. 97–179. DOI: 10.1016/0079-6107(84)90007-5 (cited on pp. 69, 76).

Bao, J., Krylova, S. M., Reinstein, O., Johnson, P. E., and Krylov, S. N. (2011)
Label-Free Solution-Based Kinetic Study of Aptamer–Small Molecule Interactions by Kinetic Capillary Electrophoresis with UV Detection Revealing How Kinetics Control Equilibrium. *Analytical Chemistry* 83.22, pp. 8387–8390. DOI: 10.1021/ac2026699 (cited on p. 122).

Barnoud, J., Santuz, H., Craveur, P., Joseph, A. P., Jallu, V., Brevern, A. G. de, and Poulain, P. (2017)
PBxplore: A Tool to Analyze Local Protein Structure and Deformability with Protein Blocks. *PeerJ* 5, e4013. DOI: 10.7717/peerj.4013 (cited on p. 27).

Bartell, L. S., Kuchitsu, K., and deNeui, R. J. (1961)
Mean and Equilibrium Molecular Structures of Methane and Deuteromethane as Determined by Electron Diffraction. *The Journal of Chemical Physics* 35.4, pp. 1211–1218. DOI: 10.1063/1.1732025 (cited on p. 89).

Baum, B. R. (1989)
PHYLIP: Phylogeny Inference Package. Version 3.2. Joel Felsenstein. *The Quarterly Review of Biology* 64.4, pp. 539–541. DOI: 10.1086/416571 (cited on p. 35).

Behar, J., Ganesan, A., Zhang, J., and Yaniv, Y. (2016)
The Autonomic Nervous System Regulates the Heart Rate through cAMP-PKA Dependent and Independent Coupled-Clock Pacemaker Cell Mechanisms. *Frontiers in Physiology* 7 (cited on p. 93).

Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D. S., and Smith, K. (2011)
Cython: The Best of Both Worlds. *Computing in Science and Engineering* 13.2, pp. 31–39. DOI: 10.1109/MCSE.2010.118 (cited on pp. 8, 43).

Benzoni, P., Bertoli, G., Giannetti, F., Piantoni, C., Milanesi, R., Pecchiari, M., Barbuti, A., Baruscotti, M., et al. (2021)
The Funny Current: Even Funnier than 40 Years Ago. Uncanonical Expression and Roles of HCN/f Channels All over the Body. *Progress in Biophysics and Molecular Biology*. 1979-2019: A "Funny" Journey Lasting 40 Years, the Multiple Roles of f/HCN Channels 166, pp. 189–204. DOI: 10.1016/j.pbiomolbio.2021.08.007 (cited on p. 93).

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011)
"Algorithms for Hyper-Parameter Optimization". *Advances in Neural Information Processing Systems*. Vol. 24. Curran Associates, Inc. (cited on p. 24).

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. (2000)
The Protein Data Bank. *Nucleic Acids Research* 28.1, pp. 235–242. DOI: 10.1093/nar/28.1.235 (cited on p. 5).

Bernard, J., Steiger, M., Mittelstädt, S., Thum, S., Keim, D., and Kohlhammer, J. (2015)
"A Survey and Task-Based Quality Assessment of Static 2D Colormaps". *Visualization and Data Analysis 2015*. Vol. 9397. SPIE, pp. 247–262. DOI: 10.1117/12.2079841 (cited on p. 22).

Biel, M., Schneider, A., and Wahl, C. (2002)
Cardiac HCN Channels: Structure, Function, and Modulation. *Trends in Cardiovascular Medicine* 12.5, pp. 206–213. DOI: 10.1016/S1050-1738(02)00162-7 (cited on p. 93).

Boussebayle, A., Torka, D., Ollivaud, S., Braun, J., Bofill-Bosch, C., Dombrowski, M., Groher, F., Hamacher, K., et al. (2019)
Next-Level Riboswitch Development—Implementation of Capture-SELEX Facilitates Identification of a New Synthetic Riboswitch. *Nucleic Acids Research* 47.9, pp. 4883–4895. DOI: 10.1093/nar/gkz216 (cited on pp. 107, 110, 114, 122, 129).

Bradley, A. R., Rose, A. S., Pavelka, A., Valasatava, Y., Duarte, J. M., Prlić, A., and Rose, P. W. (2017)
MMTF—An Efficient File Format for the Transmission, Visualization, and Analysis of Macromolecular Structures. *PLOS Computational Biology* 13.6, e1005575. DOI: 10.1371/journal.pcbi.1005575 (cited on p. 14).

Bromham, L. and Penny, D. (2003)
The Modern Molecular Clock. *Nature Reviews Genetics* 4.3, pp. 216–224. DOI: 10.1038/nrg1020 (cited on p. 34).

Brooks, B., Brooks, C., MacKerell, A., Nilsson, L., Petrella, R., Roux, B., Won, Y., Archontis, G., et al. (2009)

CHARMM: The Biomolecular Simulation Program. *Journal of Computational Chemistry* 30.10, pp. 1545–1614. DOI: 10.1002/jcc.21287 (cited on p. 81).

Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S., and Karplus, M. (1983) CHARMM: A Program for Macromolecular Energy, Minimization, and Dynamics Calculations. *Journal of Computational Chemistry* 4.2, pp. 187–217. DOI: 10.1002/jcc.540040211 (cited on p. 81).

Burley, S. K., Bhikadiya, C., Bi, C., Bittrich, S., Chen, L., Crichlow, G. V., Christie, C. H., Dalenberg, K., et al. (2021) RCSB Protein Data Bank: Powerful New Tools for Exploring 3D Structures of Biological Macromolecules for Basic and Applied Research and Education in Fundamental Biology, Biomedicine, Biotechnology, Bioengineering and Energy Sciences. *Nucleic Acids Research* 49.D1, pp. D437–D451. DOI: 10.1093/nar/gkaa1038 (cited on p. 11).

Bushnell, G. W., Louie, G. V., and Brayer, G. D. (1990) High-Resolution Three-Dimensional Structure of Horse Heart Cytochrome c. *Journal of Molecular Biology* 214.2, pp. 585–595. DOI: 10.1016/0022-2836(90)90200-6 (cited on p. 89).

Chang, A. L., McKeague, M., Liang, J. C., and Smolke, C. D. (2014) Kinetic and Equilibrium Binding Characterization of Aptamers to Small Molecules Using a Label-Free, Sensitive, and Scalable Platform. *Analytical Chemistry* 86.7, pp. 3273–3278. DOI: 10.1021/ac5001527 (cited on p. 122).

Chao, K.-M., Pearson, W. R., and Miller, W. (1992) Aligning Two Sequences within a Specified Diagonal Band. *Bioinformatics* 8.5, pp. 481–487. DOI: 10.1093/bioinformatics/8.5.481 (cited on p. 41).

Cho, M., Xiao, Y., Nie, J., Stewart, R., Csordas, A. T., Oh, S. S., Thomson, J. A., and Soh, H. T. (2010) Quantitative Selection of DNA Aptamers through Microfluidic Selection and High-Throughput Sequencing. *Proceedings of the National Academy of Sciences* 107.35, pp. 15373–15378. DOI: 10.1073/pnas.1009331107 (cited on p. 106).

Choi, K. P., Zeng, F., and Zhang, L. (2004) Good Spaced Seeds for Homology Search. *Bioinformatics* 20.7, pp. 1053–1059. DOI: 10.1093/bioinformatics/bth037 (cited on pp. 40, 47).

CIE (1999) IEC 61966-2-1:1999. Multimedia Systems and Equipment - Colour Measurement and Management - Part 2-1: Colour Management - Default RGB Colour Space - sRGB (cited on p. 21).

CIE (2001) CIE 142:2001. Improvement to Industrial Colour-Difference Evaluation (cited on p. 23).

CIE (2019) ISO/CIE 11664-4:2019. Colorimetry — Part 4: CIE 1976 L*a*b* Colour Space (cited on pp. 22, 23).

Cieplak, P., Cornell, W. D., Bayly, C., and Kollman, P. A. (1995) Application of the multimolecule and multiconformational RESP methodology to biopolymers: Charge derivation for DNA, RNA, and proteins. *Journal of Computational Chemistry* 16.11, pp. 1357–1377. DOI: 10.1002/jcc.540161106 (cited on p. 57).

Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., et al. (2009) Biopython: Freely Available Python Tools for Computational Molecular Biology and Bioinformatics. *Bioinformatics* 25.11, pp. 1422–1423. DOI: 10.1093/bioinformatics/btp163 (cited on p. 9).

College of William & Mary (2010)
The Binomial Distribution Is the Limit of the Hypergeometric Distribution. http://www.math.wm.edu/~leemis/chart/UDR/PDFs/HypergeometricBinomial.pdf (cited on p. 115).

Cyvin, S. J., Cyvin, B. N., Brunvoll, J., Whitmer, J. C., Klaeboe, P., and Gustavsen, J. E. (1979)
Condensed Aromatics. Part III. In-Plane Molecular Vibrations of Pyrene. *Zeitschrift für Naturforschung A* 34.7, pp. 876–886. DOI: 10.1515/zna-1979-0713 (cited on p. 89).

Das, R. and Baker, D. (2008)
Macromolecular Modeling with Rosetta. *Annual Review of Biochemistry* 77, pp. 363–382. DOI: 10.1146/annurev.biochem.77.062906.171838 (cited on p. 6).

Davis, J. H. and Szostak, J. W. (2002)
Isolation of High-Affinity GTP Aptamers from Partially Structured RNA Libraries. *Proceedings of the National Academy of Sciences* 99.18, pp. 11616–11621. DOI: 10.1073/pnas.182095699 (cited on p. 104).

de Brevern, A. G., Etchebest, C., and Hazout, S. (2000)
Bayesian Probabilistic Approach for Predicting Backbone Structures in Terms of Protein Blocks. *Proteins* 41.3, pp. 271–287. DOI: 10.1002/1097-0134(20001115)41:3<271::aid-prot10>3.0.co;2-z (cited on p. 27).

Dehouck, Y. and Mikhailov, A. S. (2013)
Effective Harmonic Potentials: Insights into the Internal Cooperativity and Sequence-Specificity of Protein Dynamics. *PLOS Computational Biology* 9.8, e1003209. DOI: 10.1371/journal.pcbi.1003209 (cited on pp. 66, 67, 95).

DiFrancesco, D. (1993)
Pacemaker Mechanisms in Cardiac Tissue. *Annual Review of Physiology* 55.1, pp. 455–472. DOI: 10.1146/annurev.ph.55.030193.002323 (cited on p. 93).

DiFrancesco, D. (2020)
A Brief History of Pacemaking. *Frontiers in Physiology* 10 (cited on p. 93).

Edgar, R. C. (2004)
MUSCLE: Multiple Sequence Alignment with High Accuracy and High Throughput. *Nucleic Acids Research* 32.5, pp. 1792–1797. DOI: 10.1093/nar/gkh340 (cited on pp. 16, 35, 79).

Ellington, A. D. and Szostak, J. W. (1990)
In Vitro Selection of RNA Molecules That Bind Specific Ligands. *Nature* 346.6287, pp. 818–822. DOI: 10.1038/346818a0 (cited on p. 104).

Etzel, M. and Mörl, M. (2017)
Synthetic Riboswitches: From Plug and Pray toward Plug and Play. *Biochemistry* 56.9, pp. 1181–1198. DOI: 10.1021/acs.biochem.6b01218 (cited on pp. 103, 104).

Ewing, B. and Green, P. (1998)
Base-Calling of Automated Sequencer Traces Using Phred. II. Error Probabilities. *Genome Research* 8.3, pp. 186–194. DOI: 10.1101/gr.8.3.186 (cited on p. 108).

Fan, J., Fu, A., and Zhang, L. (2019)
Progress in Molecular Docking. *Quantitative Biology* 7.2, pp. 83–89. DOI: 10.1007/s40484-019-0172-y (cited on p. 61).

Fan, P., Suri, A. K., Fiala, R., Live, D., and Patel, D. J. (1996)
Molecular Recognition in the FMN – RNA Aptamer Complex. *Journal of Molecular Biology* 258.3, pp. 480–500. DOI: 10.1006/jmbi.1996.0263 (cited on p. 101).

Farris, F. A. (2010)
The Gini Index and Measures of Inequality. *The American Mathematical Monthly* 117.10, pp. 851–864. DOI: 10.4169/000298910X523344 (cited on p. 110).

Feng, D.-F. and Doolittle, R. F. (1987)
Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees. *Journal of Molecular Evolution* 25.4, pp. 351–360. DOI: 10.1007/BF02603120 (cited on p. 33).

Feng, D.-F. and Doolittle, R. F. (1996)
Progressive Alignment of Amino Acid Sequences and Construction of Phylogenetic Trees from Them. *Methods in Enzymology*. Computer Methods for Macromolecular Sequence Analysis 266, pp. 368–382. DOI: 10.1016/S0076-6879(96)66023-6 (cited on p. 36).

Frith, M. C. (2011)
A New Repeat-Masking Method Enables Specific Detection of Homologous Sequences. *Nucleic Acids Research* 39.4, e23–e23. DOI: 10.1093/nar/gkq1212 (cited on p. 39).

Gabb, H. A., Sanghani, S. R., Robert, C. H., and Prévost, C. (1996)
Finding and Visualizing Nucleic Acid Base Stacking. *Journal of Molecular Graphics* 14.1, pp. 6–11. DOI: 10.1016/0263-7855(95)00086-0 (cited on pp. 75, 77).

Gal, S. W., Amontov, S., Urvil, P. T., Vishnuvardhan, D., Nishikawa, F., Kumar, P. K. R., and Nishikawa, S. (1998)
Selection of a RNA Aptamer That Binds to Human Activated Protein C and Inhibits Its Protease Function. *European Journal of Biochemistry* 252.3, pp. 553–562. DOI: 10.1046/j.1432-1327.1998.2520553.x (cited on p. 101).

Gasteiger, J. and Marsili, M. (1980)
Iterative Partial Equalization of Orbital Electronegativity—a Rapid Access to Atomic Charges. *Tetrahedron* 36.22, pp. 3219–3228. DOI: 10.1016/0040-4020(80)80168-2 (cited on pp. 57, 58, 85).

Gawehn, E., Hiss, J. A., and Schneider, G. (2016)
Deep Learning in Drug Discovery. *Molecular Informatics* 35.1, pp. 3–14. DOI: 10.1002/minf.201501008 (cited on p. 61).

Gesteland, R. F. (1998)
The RNA World, Second Edition. Second Edition. Cold Spring Harbor Laboratory (cited on p. 75).

Gibrat, J.-F. (2018)
A Short Note on Dynamic Programming in a Band. *BMC Bioinformatics* 19.1, p. 226. DOI: 10.1186/s12859-018-2228-9 (cited on p. 42).

Goodsell, D. S. and Olson, A. J. (1990)
Automated Docking of Substrates to Proteins by Simulated Annealing. *Proteins: Structure, Function, and Bioinformatics* 8.3, pp. 195–202. DOI: 10.1002/prot.340080302 (cited on pp. 58, 61).

Goto, N., Prins, P., Nakao, M., Bonnal, R., Aerts, J., and Katayama, T. (2010)
BioRuby: Bioinformatics Software for the Ruby Programming Language. *Bioinformatics* 26.20, pp. 2617–2619. DOI: 10.1093/bioinformatics/btq475 (cited on p. 9).

Gowers, R. J., Linke, M., Barnoud, J., Reddy, T. J. E., Melo, M. N., Seyler, S. L., Domański, J., Dotson, D. L., et al. (2016)
MDAnalysis: A Python Package for the Rapid Analysis of Molecular Dynamics Simulations. *Proceedings of the 15th Python in Science Conference*, pp. 98–105. DOI: 10.25080/Majora-629e541a-00e (cited on pp. 8, 13).

Grant, B. J., Rodrigues, A. P. C., ElSawy, K. M., McCammon, J. A., and Caves, L. S. D. (2006) Bio3d: An R Package for the Comparative Analysis of Protein Structures. *Bioinformatics* 22.21, pp. 2695–2696. DOI: 10.1093/bioinformatics/btl461 (cited on p. 9).

Groher, F., Bofill-Bosch, C., Schneider, C., Braun, J., Jager, S., Geißler, K., Hamacher, K., and Suess, B. (2018) Riboswitching with Ciprofloxacin—Development and Characterization of a Novel RNA Regulator. *Nucleic Acids Research* 46.4, pp. 2121–2132. DOI: 10.1093/nar/gkx1319 (cited on pp. 107, 110, 114, 122).

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008) “Exploring Network Structure, Dynamics, and Function Using NetworkX”. *Proceedings of the 7th Python in Science Conference*. Pasadena, CA USA, pp. 11–15 (cited on p. 35).

Hahn, L., Leimeister, C.-A., Ounit, R., Lonardi, S., and Morgenstern, B. (2016) Rasbhari: Optimizing Spaced Seeds for Database Searching, Read Mapping and Alignment-Free Sequence Comparison. *PLOS Computational Biology* 12.10, e1005107. DOI: 10.1371/journal.pcbi.1005107 (cited on p. 40).

Hamacher, K. and McCammon, J. A. (2006) Computing the Amino Acid Specificity of Fluctuations in Biomolecular Systems. *Journal of Chemical Theory and Computation* 2.3, pp. 873–878. DOI: 10.1021/ct050247s (cited on pp. 66–68).

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., et al. (2020) Array Programming with NumPy. *Nature* 585.7825, pp. 357–362. DOI: 10.1038/s41586-020-2649-2 (cited on p. 6).

Hauser, M., Mayer, C. E., and Söding, J. (2013) kClust: Fast and Sensitive Clustering of Large Protein Sequence Databases. *BMC Bioinformatics* 14.1, p. 248. DOI: 10.1186/1471-2105-14-248 (cited on p. 44).

Henikoff, S. and Henikoff, J. G. (1992) Amino Acid Substitution Matrices from Protein Blocks. *Proceedings of the National Academy of Sciences* 89.22, pp. 10915–10919. DOI: 10.1073/pnas.89.22.10915 (cited on p. 14).

Hermann, T. and Patel, D. J. (2000) RNA Bulges as Architectural and Recognition Motifs. *Structure* 8.3, R47–R54. DOI: 10.1016/S0969-2126(00)00110-6 (cited on p. 74).

Hinsen, K., Petrescu, A.-J., Dellerue, S., Bellissent-Funel, M.-C., and Kneller, G. R. (2000) Harmonicity in Slow Protein Dynamics. *Chemical Physics* 261.1, pp. 25–37. DOI: 10.1016/S0301-0104(00)00222-6 (cited on pp. 66, 67).

Hodge, C. N., Aldrich, P. E., Bacheler, L. T., Chang, C. H., Eyermann, C. J., Garber, S., Grubb, M., Jackson, D. A., et al. (1996) Improved Cyclic Urea Inhibitors of the HIV-1 Protease: Synthesis, Potency, Resistance Profile, Human Pharmacokinetics and X-ray Crystal Structure of DMP 450. *Chemistry & Biology* 3.4, pp. 301–314. DOI: 10.1016/s1074-5521(96)90110-6 (cited on p. 68).

Hofacker, I. L., Fekete, M., and Stadler, P. F. (2002) Secondary Structure Prediction for Aligned RNA Sequences. *Journal of Molecular Biology* 319.5, pp. 1059–1066. DOI: 10.1016/S0022-2836(02)00308-X (cited on p. 78).

Hornak, V., Okur, A., Rizzo, R. C., and Simmerling, C. (2006) HIV-1 Protease Flaps Spontaneously Open and Reclose in Molecular Dynamics Simulations. *Proceed-*

ings of the National Academy of Sciences 103.4, pp. 915–920. DOI: 10.1073/pnas.0508452103 (cited on p. 68).

Hsu, C., Verkuil, R., Liu, J., Lin, Z., Hie, B., Sercu, T., Lerer, A., and Rives, A. (2022) Learning Inverse Folding from Millions of Predicted Structures. *BioRxiv*, p. 2022.04.10.487779. DOI: 10.1101/2022.04.10.487779 (cited on p. 135).

Hubbard, R. E. and Kamran Haider, M. (2010) Hydrogen Bonds in Proteins: Role and Strength. *Encyclopedia of Life Sciences*. DOI: 10.1002/9780470015902.a0003011.pub2 (cited on p. 69).

Huizenga, D. E. and Szostak, J. W. (1995) A DNA Aptamer That Binds Adenosine and ATP. *Biochemistry* 34.2, pp. 656–665. DOI: 10.1021/bi00002a033 (cited on p. 101).

Hunter, J. D. (2007) Matplotlib: A 2D Graphics Environment. *Computing in Science Engineering* 9.3, pp. 90–95. DOI: 10.1109/MCSE.2007.55 (cited on pp. 1, 8, 31).

Ikeguchi, M., Ueno, J., Sato, M., and Kidera, A. (2005) Protein Structural Change Upon Ligand Binding: Linear Response Theory. *Physical Review Letters* 94.7, p. 078102. DOI: 10.1103/PhysRevLett.94.078102 (cited on p. 67).

Ivie, P. and Thain, D. (2018) Reproducibility in Scientific Computing. *ACM Computing Surveys* 51.3, 63:1–63:36. DOI: 10.1145/3186266 (cited on p. 17).

Jackson, H. A., Marshall, C. R., and Accili, E. A. (2007) Evolution and Structural Diversification of Hyperpolarization-Activated Cyclic Nucleotide-Gated Channel Genes. *Physiological Genomics* 29.3, pp. 231–245. DOI: 10.1152/physiolgenomics.00142.2006 (cited on pp. 94, 100).

Jolma, A., Yan, J., Whitington, T., Toivonen, J., Nitta, K. R., Rastas, P., Morgunova, E., Enge, M., et al. (2013) DNA-Binding Specificities of Human Transcription Factors. *Cell* 152.1, pp. 327–339. DOI: 10.1016/j.cell.2012.12.009 (cited on p. 125).

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., et al. (2021) Highly Accurate Protein Structure Prediction with AlphaFold. *Nature* 596.7873, pp. 583–589. DOI: 10.1038/s41586-021-03819-2 (cited on p. 33).

Kabsch, W. (1976) A Solution for the Best Rotation to Relate Two Sets of Vectors. *Acta Crystallographica Section A* 32.5, pp. 922–923. DOI: 10.1107/S0567739476001873 (cited on pp. 16, 83).

Kabsch, W. (1978) A Discussion of the Solution for the Best Rotation to Relate Two Sets of Vectors. *Acta Crystallographica Section A* 34.5, pp. 827–828. DOI: 10.1107/S0567739478001680 (cited on pp. 16, 83).

Kabsch, W. and Sander, C. (1983) Dictionary of Protein Secondary Structure: Pattern Recognition of Hydrogen-Bonded and Geometrical Features. *Biopolymers* 22.12, pp. 2577–2637. DOI: 10.1002/bip.360221211 (cited on p. 16).

Kapli, P., Yang, Z., and Telford, M. J. (2020)
Phylogenetic Tree Building in the Genomic Age. *Nature Reviews Genetics* 21.7, pp. 428–444. DOI: 10.1038/s41576-020-0233-0 (cited on pp. 33, 34).

Katoh, K., Misawa, K., Kuma, K.-i., and Miyata, T. (2002)
MAFFT: A Novel Method for Rapid Multiple Sequence Alignment Based on Fast Fourier Transform. *Nucleic Acids Research* 30.14, pp. 3059–3066. DOI: 10.1093/nar/gkf436 (cited on pp. 16, 35).

Katz, B. A. (1997)
Binding of Biotin to Streptavidin Stabilizes Intersubunit Salt Bridges between Asp61 and His87 at Low pH11Edited by I. A. Wilson. *Journal of Molecular Biology* 274.5, pp. 776–800. DOI: 10.1006/jmbi.1997.1444 (cited on p. 63).

Keskin, O., Bahar, I., Jernigan, R. L., Badretdinov, A. Y., and Ptitsyn, O. B. (1998)
Empirical Solvent-Mediated Potentials Hold for Both Intra-Molecular and Inter-Molecular Inter-Residue Interactions. *Protein Science* 7.12, pp. 2578–2586. DOI: 10.1002/pro.5560071211 (cited on p. 66).

Keul, F., Hess, M., Goesele, M., and Hamacher, K. (2017)
PFASUM: A Substitution Matrix from Pfam Structural Alignments. *BMC Bioinformatics* 18.1, p. 293. DOI: 10.1186/s12859-017-1703-z (cited on pp. 14, 28).

Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B. A., et al. (2021)
PubChem in 2021: New Data Content and Improved Web Interfaces. *Nucleic Acids Research* 49.D1, pp. D1388–D1395. DOI: 10.1093/nar/gkaa971 (cited on pp. 86, 88).

Kingma, D. P. and Ba, J. (2017)
Adam: A Method for Stochastic Optimization. *ArXiv*. DOI: 10.48550/arXiv.1412.6980 (cited on p. 118).

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983)
Optimization by Simulated Annealing. *Science* 220.4598, pp. 671–680. DOI: 10.1126/science.220.4598.671 (cited on p. 24).

Kovalev, V. (2004)
"Towards Image Retrieval for Eight Percent of Color-Blind Men". *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 2, 943–946 Vol.2. DOI: 10.1109/ICPR.2004.1334414 (cited on p. 27).

Kumar Kulabhusan, P., Hussain, B., and Yüce, M. (2020)
Current Perspectives on Aptamers as Diagnostic Tools and Therapeutic Agents. *Pharmaceutics* 12.7, p. 646. DOI: 10.3390/pharmaceutics12070646 (cited on p. 103).

Kunzmann, **P.** (2018)
A Convenient and Efficient Computational Molecular Biology Framework in Python (cited on p. 11).

Kunzmann, **P.** (2022)
Workflow for producing thesis results. https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/3628 (cited on pp. 1, 20, 92).

Kunzmann, **P.**, Anter, J. M., and Hamacher, K. (2022)
Adding Hydrogen Atoms to Molecular Models via Fragment Superimposition. *Algorithms for Molecular Biology* 17.1, p. 7. DOI: 10.1186/s13015-022-00215-x (cited on p. 81).

Kunzmann, P., Mayer, B. E., and Hamacher, K. (2020)
Substitution Matrix Based Color Schemes for Sequence Alignment Visualization. *BMC Bioinformatics* 21.1, p. 209. DOI: 10.1186/s12859-020-3526-6 (cited on p. 21).

Kunzmann, P., Müller, T., Greil, M., Krumbach, J. H., Anter, J. M., Islam, F., and Hamacher, K. (2022)
Biotite: New tools for a versatile Python bioinformatics library. *BMC Bioinformatics (submitted)* (cited on pp. 39, 135).

Kusch, J., Biskup, C., Thon, S., Schulz, E., Nache, V., Zimmer, T., Schwede, F., and Benndorf, K. (2010)
Interdependence of Receptor Activation and Ligand Binding in HCN2 Pacemaker Channels. *Neuron* 67.1, pp. 75–85. DOI: 10.1016/j.neuron.2010.05.022 (cited on pp. 93, 100).

Kusch, J., Thon, S., Schulz, E., Biskup, C., Nache, V., Zimmer, T., Seifert, R., Schwede, F., et al. (2012)
How Subunits Cooperate in cAMP-induced Activation of Homotetrameric HCN2 Channels. *Nature Chemical Biology* 8.2, pp. 162–169. DOI: 10.1038/nchembio.747 (cited on pp. 93, 97, 99).

Labesse, G., Colloc'h, N., Pothier, J., and Mornon, J.-P. (1997)
P-SEA: A New Efficient Assignment of Secondary Structure from Cα Trace of Proteins. *Bioinformatics* 13.3, pp. 291–295. DOI: 10.1093/bioinformatics/13.3.291 (cited on p. 16).

Lafita, A., Bliven, S., Prlić, A., Guzenko, D., Rose, P. W., Bradley, A., Pavan, P., Myers-Turnbull, D., et al. (2019)
BioJava 5: A Community Driven Open-Source Bioinformatics Library. *PLOS Computational Biology* 15.2, e1006791. DOI: 10.1371/journal.pcbi.1006791 (cited on p. 9).

Larkin, M., Blackshields, G., Brown, N., Chenna, R., McGettigan, P., McWilliam, H., Valentin, F., Wallace, I., et al. (2007)
Clustal W and Clustal X Version 2.0. *Bioinformatics* 23.21, pp. 2947–2948. DOI: 10.1093/bioinformatics/btm404 (cited on pp. 15, 21).

Le, K., Adolf-Bryfogle, J., Klima, J., Lyskov, S., Labonte, J., Bertolani, S., Burman, S. R., Leaver-Fay, A., et al. (2020)
PyRosetta Jupyter Notebooks Teach Biomolecular Structure Prediction and Design. *Preprints*. DOI: 10.20944/preprints202002.0097.v1 (cited on p. 52).

Lee, C.-H. and MacKinnon, R. (2017)
Structures of the Human HCN1 Hyperpolarization-Activated Channel. *Cell* 168.1, 111–120.e11. DOI: 10.1016/j.cell.2016.12.023 (cited on p. 97).

Leontis, N. B. and Westhof, E. (2001)
Geometric Nomenclature and Classification of RNA Base Pairs. *RNA* 7.4, pp. 499–512. DOI: 10.1017/s1355838201002515 (cited on p. 73).

Levine, H. A. and Nilsen-Hamilton, M. (2007)
A Mathematical Analysis of SELEX. *Computational Biology and Chemistry* 31.1, pp. 11–35. DOI: 10.1016/j.compbiolchem.2006.10.002 (cited on pp. 107, 111, 122, 123).

Lewis, M. (2005)
The Lac Repressor. *Comptes Rendus Biologies*. Retour Sur l'operon Lac 328.6, pp. 521–548. DOI: 10.1016/j.crvi.2005.04.004 (cited on p. 70).

Li, L., Xu, S., Yan, H., Li, X., Yazd, H. S., Li, X., Huang, T., Cui, C., et al. (2021)
Nucleic Acid Aptamers for Molecular Diagnostics and Therapeutics: Advances and Perspectives. *Angewandte Chemie International Edition* 60.5, pp. 2221–2231. DOI: 10.1002/anie.202003563 (cited on pp. 101, 103).

Li, Y., Roy, A., and Zhang, Y. (2009)
HAAD: A Quick Algorithm for Accurate Prediction of Hydrogen Atoms in Protein Structures. *PLOS ONE* 4.8, e6701. DOI: 10.1371/journal.pone.0006701 (cited on pp. 6, 81, 86).

Lide, D. R. (2003)
CRC Handbook of Chemistry and Physics. Boca Raton: CRC Press LLC (cited on p. 85).

Lindahl, Abraham, Hess, and van der Spoel (2021)
GROMACS 2021.2 Manual. DOI: 10.5281/zenodo.4723561 (cited on pp. 6, 81).

Lorenz, R., Bernhart, S. H., Höner zu Siederdissen, C., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. (2011)
ViennaRNA Package 2.0. *Algorithms for Molecular Biology* 6.1, p. 26. DOI: 10.1186/1748-7188-6-26 (cited on p. 75).

Lu, X.-J., Bussemaker, H. J., and Olson, W. K. (2015)
DSSR: An Integrated Software Tool for Dissecting the Spatial Structure of RNA. *Nucleic Acids Research* 43.21, e142–e142. DOI: 10.1093/nar/gkv716 (cited on pp. 75, 76).

Ludwig, W. and Schleifer, K. (1994)
Bacterial Phylogeny Based on 16S and 23S rRNA Sequence Analysis. *FEMS Microbiology Reviews* 15.2-3, pp. 155–173. DOI: 10.1111/j.1574-6976.1994.tb00132.x (cited on pp. 37, 48).

Luscombe, N. M., Greenbaum, D., and Gerstein, M. (2001)
What Is Bioinformatics? An Introduction and Overview. *Yearbook of Medical Informatics* 10.1, pp. 83–100. DOI: 10.1055/s-0038-1638103 (cited on p. 5).

Ma, B., Tromp, J., and Li, M. (2002)
PatternHunter: Faster and More Sensitive Homology Search. *Bioinformatics* 18.3, pp. 440–445. DOI: 10.1093/bioinformatics/18.3.440 (cited on p. 40).

Mastryukov, V. S., Fan, K., and Boggs, J. E. (1995)
The Effect of Methylation on the Structure of Uracil. *Journal of Molecular Structure*. Electron Diffraction and Structural Chemistry 346, pp. 173–186. DOI: 10.1016/0022-2860(94)09006-B (cited on p. 88).

Mathews, D. H. (2006a)
Predicting RNA Secondary Structure by Free Energy Minimization. *Theoretical Chemistry Accounts* 116.1, pp. 160–168. DOI: 10.1007/s00214-005-0027-7 (cited on p. 79).

Mathews, D. H. (2006b)
Revolutions in RNA Secondary Structure Prediction. *Journal of Molecular Biology* 359.3, pp. 526–532. DOI: 10.1016/j.jmb.2006.01.067 (cited on p. 75).

McCormick, D. A. and Bal, T. (1997)
Sleep and Arousal: Thalamocortical Mechanisms. *Annual Review of Neuroscience* 20, pp. 185–215. DOI: 10.1146/annurev.neuro.20.1.185 (cited on p. 93).

McGibbon, R. T., Beauchamp, K. A., Harrigan, M. P., Klein, C., Swails, J. M., Hernández, C. X., Schwantes, C. R., Wang, L.-P., et al. (2015)

MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. *Biophysical Journal* 109.8, pp. 1528–1532. DOI: 10.1016/j.bpj.2015.08.015 (cited on pp. 8, 13, 14, 54).

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953)
Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* 21.6, pp. 1087–1092. DOI: 10.1063/1.1699114 (cited on p. 23).

Miyazawa, S. and Jernigan, R. L. (1996)
Residue – Residue Potentials with a Favorable Contact Pair Term and an Unfavorable High Packing Density Term, for Simulation and Threading. *Journal of Molecular Biology* 256.3, pp. 623–644. DOI: 10.1006/jmbi.1996.0114 (cited on p. 66).

Morgulis, A., Gertz, E. M., Schäffer, A. A., and Agarwala, R. (2006)
A Fast and Symmetric DUST Implementation to Mask Low-Complexity DNA Sequences. *https://home.liebertpub.com/cmb* 13.5. DOI: 10.1089/cmb.2006.13.1028 (cited on p. 39).

Müller, T. (2021)
Biology Department Computational Biology and Simulation Secondary structure analysis of nucleic acid structures and sequences (cited on p. 73).

Mulliken, R. S. (1934)
A New Electroaffinity Scale; Together with Data on Valence States and on Valence Ionization Potentials and Electron Affinities. *The Journal of Chemical Physics* 2.11, pp. 782–793. DOI: 10.1063/1.1749394 (cited on p. 58).

Needleman, S. B. and Wunsch, C. D. (1970)
A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *Journal of Molecular Biology* 48.3, pp. 443–453. DOI: 10.1016/0022-2836(70)90057-4 (cited on pp. 14, 33, 41).

Neubacher, S. and Hennig, S. (2019)
RNA Structure and Cellular Applications of Fluorescent Light-Up Aptamers. *Angewandte Chemie International Edition* 58.5, pp. 1266–1279. DOI: 10.1002/anie.201806482 (cited on p. 102).

NIH (2019)
Color Blindness. https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/color-blindness (cited on p. 27).

Noé, L. (2017)
Best Hits of 11110110111: Model-Free Selection and Parameter-Free Sensitivity Calculation of Spaced Seeds. *Algorithms for Molecular Biology* 12.1, p. 1. DOI: 10.1186/s13015-017-0092-1 (cited on p. 40).

O'Boyle, N. M., Banck, M., James, C. A., Morley, C., Vandermeersch, T., and Hutchison, G. R. (2011)
Open Babel: An Open Chemical Toolbox. *Journal of Cheminformatics* 3.1, p. 33. DOI: 10.1186/1758-2946-3-33 (cited on p. 81).

Ogawa, T. and Nakano, T. (2010)
The Extended Universal Force Field (XUFF):Theory and Applications. *Chem-Bio Informatics Journal* 10, pp. 111–133. DOI: 10.1273/cbij.10.111 (cited on p. 84).

Olson, W. K., Bansal, M., Burley, S. K., Dickerson, R. E., Gerstein, M., Harvey, S. C., Heinemann, U., Lu, X.-J., et al. (2001)
A Standard Reference Frame for the Description of Nucleic Acid Base-Pair Geometry. *Journal of Molecular Biology* 313.1, pp. 229–237. DOI: 10.1006/jmbi.2001.4987 (cited on p. 76).

Olsson, M. H. M., Søndergaard, C. R., Rostkowski, M., and Jensen, J. H. (2011)
PROPKA3: Consistent Treatment of Internal and Surface Residues in Empirical pKa Predictions. *Journal of Chemical Theory and Computation* 7.2, pp. 525–537. DOI: 10.1021/ct100578z (cited on p. 86).

Oostenbrink, C., Villa, A., Mark, A. E., and Van Gunsteren, W. F. (2004)
A Biomolecular Force Field Based on the Free Enthalpy of Hydration and Solvation: The GROMOS Force-Field Parameter Sets 53A5 and 53A6. *Journal of Computational Chemistry* 25.13, pp. 1656–1676. DOI: 10.1002/jcc.20090 (cited on p. 57).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., et al. (2019)
PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems* 32 (cited on p. 118).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., et al. (2011)
Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research* 12.85, pp. 2825–2830 (cited on p. 8).

Pérez, F., Granger, B. E., and Hunter, J. D. (2011)
Python: An Ecosystem for Scientific Computing. *Computing in Science & Engineering* 13.2, pp. 13–21. DOI: 10.1109/MCSE.2010.119 (cited on p. 6).

Peselis, A. and Serganov, A. (2014)
Themes and Variations in Riboswitch Structure and Function. *Biochimica et Biophysica Acta (BBA) - Gene Regulatory Mechanisms*. Riboswitches 1839.10, pp. 908–918. DOI: 10.1016/j.bbagrm.2014.02.012 (cited on pp. 101, 103, 104).

Pipas, J. M. and McMahon, J. E. (1975)
Method for Predicting RNA Secondary Structure. *Proceedings of the National Academy of Sciences* 72.6, pp. 2017–2021. DOI: 10.1073/pnas.72.6.2017 (cited on p. 75).

Porcelli, A. M., Ghelli, A., Zanna, C., Pinton, P., Rizzuto, R., and Rugolo, M. (2005)
pH Difference across the Outer Mitochondrial Membrane Measured with a Green Fluorescent Protein Mutant. *Biochemical and Biophysical Research Communications* 326.4, pp. 799–804. DOI: 10.1016/j.bbrc.2004.11.105 (cited on p. 89).

Porro, A., Saponaro, A., Gasparri, F., Bauer, D., Gross, C., Pisoni, M., Abbandonato, G., Hamacher, K., et al. (2019)
The HCN Domain Couples Voltage Gating and cAMP Response in Hyperpolarization-Activated Cyclic Nucleotide-Gated Channels. *eLife* 8, e49672. DOI: 10.7554/eLife.49672 (cited on p. 94).

Prins, P. (2014)
Small Tools Manifesto for Bioinformatics. DOI: 10.5281/zenodo.11321 (cited on pp. 5, 135).

Ramón-García, S., Mikut, R., Ng, C., Ruden, S., Volkmer, R., Reischl, M., Hilpert, K., and Thompson, C. J. (2013)
Targeting Mycobacterium Tuberculosis and Other Microbial Pathogens Using Improved Synthetic Antibacterial Peptides. *Antimicrobial Agents and Chemotherapy* 57.5, pp. 2295–2303. DOI: 10.1128/AAC.00175-13 (cited on p. 52).

Rappe, A. K., Casewit, C. J., Colwell, K. S., Goddard, W. A., and Skiff, W. M. (1992)
UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations. *Journal of the American Chemical Society* 114.25, pp. 10024–10035. DOI: 10.1021/ja00051a040 (cited on p. 84).

Rappe, A. K. and Goddard, W. A. I. (1991)
Charge Equilibration for Molecular Dynamics Simulations. *The Journal of Physical Chemistry* 95.8, pp. 3358–3363. DOI: 10.1021/j100161a070 (cited on pp. 15, 58).

Reinert, K., Dadi, T. H., Ehrhardt, M., Hauswedell, H., Mehringer, S., Rahn, R., Kim, J., Pockrandt, C., et al. (2017)
The SeqAn C++ Template Library for Efficient Sequence Analysis: A Resource for Programmers. *Journal of Biotechnology*. Bioinformatics Solutions for Big Data Analysis in Life Sciences Presented by the German Network for Bioinformatics Infrastructure 261, pp. 157–168. DOI: 10.1016/j.jbiotec.2017.07.017 (cited on p. 5).

Richard, G.-F., Kerrest, A., and Dujon, B. (2008)
Comparative Genomics and Molecular Dynamics of DNA Repeats in Eukaryotes. *Microbiology and Molecular Biology Reviews : MMBR* 72.4, pp. 686–727. DOI: 10.1128/MMBR.00011-08 (cited on p. 39).

Robinson, A. B. and Robinson, L. R. (1991)
Distribution of Glutamine and Asparagine Residues and Their near Neighbors in Peptides and Proteins. *Proceedings of the National Academy of Sciences* 88.20, pp. 8880–8884. DOI: 10.1073/pnas.88.20.8880 (cited on p. 46).

Romanuka, J., Folkers, G. E., Biris, N., Tishchenko, E., Wienk, H., Bonvin, A. M. J. J., Kaptein, R., and Boelens, R. (2009)
Specificity and Affinity of Lac Repressor for the Auxiliary Operators O2 and O3 Are Explained by the Structures of Their Protein–DNA Complexes. *Journal of Molecular Biology* 390.3, pp. 478–489. DOI: 10.1016/j.jmb.2009.05.022 (cited on pp. 70, 71).

Rosenblatt, F. (1958)
The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review* 65.6, pp. 386–408. DOI: 10.1037/h0042519 (cited on p. 115).

Saitou, N. and Nei, M. (1987)
The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees. *Molecular Biology and Evolution* 4.4, pp. 406–425. DOI: 10.1093/oxfordjournals.molbev.a040454 (cited on p. 34).

Sajek, M. P., Woźniak, T., Sprinzl, M., Jaruzelska, J., and Barciszewski, J. (2020)
T-Psi-C: User Friendly Database of tRNA Sequences and Structures. *Nucleic Acids Research* 48.D1, pp. D256–D260. DOI: 10.1093/nar/gkz922 (cited on p. 79).

Saponaro, A., Bauer, D., Giese, M. H., Swuec, P., Porro, A., Gasparri, F., Sharifzadeh, A. S., Chaves-Sanjuan, A., et al. (2021)
Gating Movements and Ion Permeation in HCN4 Pacemaker Channels. *Molecular Cell* 81.14, 2929–2943.e6. DOI: 10.1016/j.molcel.2021.05.033 (cited on pp. 32, 94–96, 99).

Sayers, E. W., Bolton, E. E., Brister, J. R., Canese, K., Chan, J., Comeau, D. C., Connor, R., Funk, K., et al. (2022)
Database Resources of the National Center for Biotechnology Information. *Nucleic Acids Research* 50.D1, pp. D20–D26. DOI: 10.1093/nar/gkab1112 (cited on p. 11).

Schrödinger (2017)
The PyMOL Molecular Graphics System, Version 2.0. https://pymol.org (cited on pp. 1, 31).

Serganov, A., Yuan, Y.-R., Pikovskaya, O., Polonskaia, A., Malinina, L., Phan, A. T., Hobartner, C., Micura, R., et al. (2004)

Structural Basis for Discriminative Regulation of Gene Expression by Adenine- and Guanine-Sensing mRNAs. *Chemistry & Biology* 11.12, pp. 1729–1741. DOI: 10.1016/j.chembiol.2004.11.018 (cited on p. 103).

Shannon, C. E. (1948)
A Mathematical Theory of Communication. *The Bell System Technical Journal* 27.3, pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x (cited on p. 110).

Shrake, A. and Rupley, J. A. (1973)
Environment and Exposure to Solvent of Protein Atoms. Lysozyme and Insulin. *Journal of Molecular Biology* 79.2, pp. 351–371. DOI: 10.1016/0022-2836(73)90011-9 (cited on pp. 15, 16).

Sievers, F., Wilm, A., Dineen, D., Gibson, T. J., Karplus, K., Li, W., Lopez, R., McWilliam, H., et al. (2011)
Fast, Scalable Generation of High-Quality Protein Multiple Sequence Alignments Using Clustal Omega. *Molecular Systems Biology* 7, p. 539. DOI: 10.1038/msb.2011.75 (cited on pp. 16, 35).

Smit, S., Rother, K., Heringa, J., and Knight, R. (2008)
From Knotted to Nested RNA Structures: A Variety of Computational Methods for Pseudoknot Removal. *RNA* 14.3, pp. 410–416. DOI: 10.1261/rna.881308 (cited on p. 77).

Smith, T. F. and Waterman, M. S. (1981)
Identification of Common Molecular Subsequences. *Journal of Molecular Biology* 147.1, pp. 195–197. DOI: 10.1016/0022-2836(81)90087-5 (cited on pp. 14, 41).

Spill, F., Weinstein, Z. B., Irani Shemirani, A., Ho, N., Desai, D., and Zaman, M. H. (2016)
Controlling Uncertainty in Aptamer Selection. *Proceedings of the National Academy of Sciences* 113.43, pp. 12076–12081. DOI: 10.1073/pnas.1605086113 (cited on pp. 114, 125).

Stajich, J. E., Block, D., Boulez, K., Brenner, S. E., Chervitz, S. A., Dagdigian, C., Fuellen, G., Gilbert, J. G. R., et al. (2002)
The Bioperl Toolkit: Perl Modules for the Life Sciences. *Genome Research* 12.10, pp. 1611–1618. DOI: 10.1101/gr.361602 (cited on p. 9).

Steinegger, M. and Söding, J. (2017)
MMseqs2 Enables Sensitive Protein Sequence Searching for the Analysis of Massive Data Sets. *Nature Biotechnology* 35.11, pp. 1026–1028. DOI: 10.1038/nbt.3988 (cited on pp. 37, 39, 46, 47).

Stoltenburg, R., Nikolaus, N., and Strehlitz, B. (2012)
Capture-SELEX: Selection of DNA Aptamers for Aminoglycoside Antibiotics. *Journal of Analytical Methods in Chemistry* 2012, e415697. DOI: 10.1155/2012/415697 (cited on p. 106).

Studier, J. A. and Keppler, K. J. (1988)
A Note on the Neighbor-Joining Algorithm of Saitou and Nei. *Molecular Biology and Evolution* 5.6, pp. 729–731. DOI: 10.1093/oxfordjournals.molbev.a040527 (cited on p. 34).

Sydow, D., Morger, A., Driller, M., and Volkamer, A. (2019)
TeachOpenCADD: A Teaching Platform for Computer-Aided Drug Design Using Open Source Packages and Data. *Journal of Cheminformatics* 11.1, p. 29. DOI: 10.1186/s13321-019-0351-x (cited on p. 135).

Takahashi, M., Wu, X., Ho, M., Chomchan, P., Rossi, J. J., Burnett, J. C., and Zhou, J. (2016)
High Throughput Sequencing Analysis of RNA Libraries Reveals the Influences of Initial Library and PCR Methods on SELEX Efficiency. *Scientific Reports* 6.1, p. 33697. DOI: 10.1038/srep33697 (cited on pp. 114, 125).

Tama, F. and Sanejouand, Y.-H. (2001)
Conformational Change of Proteins Arising from Normal Mode Calculations. *Protein Engineering, Design and Selection* 14.1, pp. 1–6. DOI: 10.1093/protein/14.1.1 (cited on p. 96).

Tanimoto, M., Kuchitsu, K., and Morino, Y. (1971)
Molecular Structure of Diacetylene as Studied by Gas Electron Diffraction. *Bulletin of the Chemical Society of Japan* 44.2, pp. 386–391. DOI: 10.1246/bcsj.44.386 (cited on p. 89).

Thiel, W. H., Bair, T., Peek, A. S., Liu, X., Dassie, J., Stockdale, K. R., Behlke, M. A., Miller, F. J., et al. (2012)
Rapid Identification of Cell-Specific, Internalizing RNA Aptamers with Bioinformatics Analyses of a Cell-Based Aptamer Selection. *PLoS ONE* 7.9, e43836. DOI: 10.1371/journal.pone.0043836 (cited on p. 110).

Thiel, W. H., Bair, T., Wyatt Thiel, K., Dassie, J. P., Rockey, W. M., Howell, C. A., Liu, X. Y., Dupuy, A. J., et al. (2011)
Nucleotide Bias Observed with a Short SELEX RNA Aptamer Library. *Nucleic Acid Therapeutics* 21.4, pp. 253–263. DOI: 10.1089/nat.2011.0288 (cited on pp. 114, 125).

Thiel, W. H. and Giangrande, P. H. (2016)
Analyzing HT-SELEX Data with the Galaxy Project Tools – A Web Based Bioinformatics Platform for Biomedical Research. *Methods*. Nucleic Acid Aptamers 97, pp. 3–10. DOI: 10.1016/j.ymeth.2015.10.008 (cited on p. 107).

Thompson, J. D., Higgins, D. G., and Gibson, T. J. (1994)
CLUSTAL W: Improving the Sensitivity of Progressive Multiple Sequence Alignment through Sequence Weighting, Position-Specific Gap Penalties and Weight Matrix Choice. *Nucleic Acids Research* 22.22, pp. 4673–4680. DOI: 10.1093/nar/22.22.4673 (cited on p. 33).

Tinoco, I., Borer, P. N., Dengler, B., Levine, M. D., Uhlenbeck, O. C., Crothers, D. M., and Gralla, J. (1973)
Improved Estimation of Secondary Structure in Ribonucleic Acids. *Nature New Biology* 246.150, pp. 40–41. DOI: 10.1038/newbio246040a0 (cited on p. 75).

Tinoco, I. and Bustamante, C. (1999)
How RNA Folds. *Journal of Molecular Biology* 293.2, pp. 271–281. DOI: 10.1006/jmbi.1999.3001 (cited on p. 74).

Trabelsi, A., Chaabane, M., and Ben-Hur, A. (2019)
Comprehensive Evaluation of Deep Learning Architectures for Prediction of DNA/RNA Sequence Binding Specificities. *Bioinformatics* 35.14, pp. i269–i277. DOI: 10.1093/bioinformatics/btz339 (cited on p. 116).

Trott, O. and Olson, A. J. (2010)
AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization and Multithreading. *Journal of Computational Chemistry* 31.2, pp. 455–461. DOI: 10.1002/jcc.21334 (cited on pp. 57, 61, 81).

Tsai, J., Taylor, R., Chothia, C., and Gerstein, M. (1999)
The Packing Density in Proteins: Standard Radii and Volumes. *Journal of Molecular Biology* 290.1, pp. 253–266. DOI: 10.1006/jmbi.1999.2829 (cited on p. 32).

Tsuji, S., Hirabayashi, N., Kato, S., Akitomi, J., Egashira, H., Tanaka, T., Waga, I., and Ohtsu, T. (2009)

Effective Isolation of RNA Aptamer through Suppression of PCR Bias. *Biochemical and Biophysical Research Communications* 386.1, pp. 223–226. DOI: 10.1016/j.bbrc.2009.06.013 (cited on pp. 114, 125).

Tuerk, C. and Gold, L. (1990)
Systematic Evolution of Ligands by Exponential Enrichment: RNA Ligands to Bacteriophage T4 DNA Polymerase. *Science*. DOI: 10.1126/science.2200121 (cited on p. 104).

Ulens, C. and Siegelbaum, S. A. (2003)
Regulation of Hyperpolarization-Activated HCN Channels by cAMP through a Gating Switch in Binding Domain Symmetry. *Neuron* 40.5, pp. 959–970. DOI: 10.1016/S0896-6273(03)00753-0 (cited on p. 99).

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., et al. (2020)
SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17.3, pp. 261–272. DOI: 10.1038/s41592-019-0686-2 (cited on p. 8).

Wahl-Schott, C. and Biel, M. (2008)
HCN Channels: Structure, Cellular Regulation and Physiological Function. *Cellular and Molecular Life Sciences* 66.3, p. 470. DOI: 10.1007/s00018-008-8525-0 (cited on pp. 93, 94, 96).

Wang, C., Zeng, J., and Wang, J. (2022)
Structural Basis of Bacteriophage Lambda Capsid Maturation. *Structure* 30.4, 637–645.e3. DOI: 10.1016/j.str.2021.12.009 (cited on p. 55).

Waskom, M. L. (2021)
Seaborn: Statistical Data Visualization. *Journal of Open Source Software* 6.60, p. 3021. DOI: 10.21105/joss.03021 (cited on p. 8).

Waterhouse, A. M., Procter, J. B., Martin, D. M. A., Clamp, M., and Barton, G. J. (2009)
Jalview Version 2—a Multiple Sequence Alignment Editor and Analysis Workbench. *Bioinformatics* 25.9, pp. 1189–1191. DOI: 10.1093/bioinformatics/btp033 (cited on pp. 15, 21).

Webb, B. and Sali, A. (2016)
Comparative Protein Structure Modeling Using MODELLER. *Current Protocols in Bioinformatics* 54, pp. 5.6.1–5.6.37. DOI: 10.1002/cpbi.3 (cited on p. 81).

Westbrook, J. D., Shao, C., Feng, Z., Zhuravleva, M., Velankar, S., and Young, J. (2015)
The Chemical Component Dictionary: Complete Descriptions of Constituent Molecules in Experimentally Determined 3D Macromolecules in the Protein Data Bank. *Bioinformatics (Oxford, England)* 31.8, pp. 1274–1278. DOI: 10.1093/bioinformatics/btu789 (cited on pp. 51, 86).

Westbrook, J. D., Young, J. Y., Shao, C., Feng, Z., Guranovic, V., Lawson, C. L., Vallat, B., Adams, P. D., et al. (2022)
PDBx/mmCIF Ecosystem: Foundational Semantic Tools for Structural Biology. *Journal of Molecular Biology*. Computation Resources for Molecular Biology 434.11, p. 167599. DOI: 10.1016/j.jmb.2022.167599 (cited on p. 6).

White, D. S., Chowdhury, S., Idikuda, V., Zhang, R., Retterer, S. T., Goldsmith, R. H., and Chanda, B. (2021)
cAMP Binding to Closed Pacemaker Ion Channels Is Non-Cooperative. *Nature* 595.7868, pp. 606–610. DOI: 10.1038/s41586-021-03686-x (cited on pp. 93, 97).

Wolf, Y. I., Rogozin, I. B., Grishin, N. V., and Koonin, E. V. (2002)
Genome Trees and the Tree of Life. *Trends in Genetics* 18.9, pp. 472–479. DOI: 10.1016/S0168-9525(02)02744-0 (cited on p. 37).

Word, J. M., Lovell, S. C., Richardson, J. S., and Richardson, D. C. (1999)
Asparagine and Glutamine: Using Hydrogen Atom Contacts in the Choice of Side-Chain Amide orientation11Edited by J. Thornton. *Journal of Molecular Biology* 285.4, pp. 1735–1747. DOI: 10.1006/jmbi.1998.2401 (cited on p. 81).

Wu, K. E., Yang, K. K., Berg, R. van den, Zou, J. Y., Lu, A. X., and Amini, A. P. (2022)
Protein Structure Generation via Folding Diffusion. *ArXiv*. DOI: 10.48550/arXiv.2209.15611 (cited on p. 135).

Yachdav, G., Wilzbach, S., Rauscher, B., Sheridan, R., Sillitoe, I., Procter, J., Lewis, S. E., Rost, B., et al. (2016)
MSAViewer: Interactive JavaScript Visualization of Multiple Sequence Alignments. *Bioinformatics* 32.22, pp. 3501–3503. DOI: 10.1093/bioinformatics/btw474 (cited on p. 21).

Yakovchuk, P., Protozanova, E., and Frank-Kamenetskii, M. D. (2006)
Base-Stacking and Base-Pairing Contributions into Thermal Stability of the DNA Double Helix. *Nucleic Acids Research* 34.2, pp. 564–574. DOI: 10.1093/nar/gkj454 (cited on p. 73).

Yang, L., Song, G., and Jernigan, R. L. (2009)
Protein Elastic Network Models and the Ranges of Cooperativity. *Proceedings of the National Academy of Sciences* 106.30, pp. 12347–12352. DOI: 10.1073/pnas.0902159106 (cited on pp. 66, 67).

Yu, H., Alkhamis, O., Canoura, J., Liu, Y., and Xiao, Y. (2021)
Advances and Challenges in Small-Molecule DNA Aptamer Isolation, Characterization, and Sensor Development. *Angewandte Chemie International Edition* 60.31, pp. 16800–16823. DOI: 10.1002/anie.202008663 (cited on p. 103).

Yuan, H., Kshirsagar, M., Zamparo, L., Lu, Y., and Leslie, C. S. (2019)
BindSpace Decodes Transcription Factor Binding Signals by Large-Scale Sequence Embedding. *Nature Methods* 16.9, pp. 858–861. DOI: 10.1038/s41592-019-0511-y (cited on p. 116).

Zhang, L. (2007)
Superiority of Spaced Seeds for Homology Search. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4.3, pp. 496–505. DOI: 10.1109/tcbb.2007.1013 (cited on p. 40).

Zhang, Z., Schwartz, S., Wagner, L., and Miller, W. (2000)
A Greedy Algorithm for Aligning DNA Sequences. *Journal of Computational Biology* 7.1-2, pp. 203–214. DOI: 10.1089/10665270050081478 (cited on p. 41).

Zhou, J. and Rossi, J. (2017)
Aptamers as Targeted Therapeutics: Current Potential and Challenges. *Nature reviews. Drug discovery* 16.3, pp. 181–202. DOI: 10.1038/nrd.2016.199 (cited on p. 103).

Zimmermann, B., Gesell, T., Chen, D., Lorenz, C., and Schroeder, R. (2010)
Monitoring Genomic Sequences during SELEX Using High-Throughput Sequencing: Neutral SELEX. *PLOS ONE* 5.2, e9169. DOI: 10.1371/journal.pone.0009169 (cited on p. 106).

Zuker, M. and Stiegler, P. (1981)
Optimal Computer Folding of Large RNA Sequences Using Thermodynamics and Auxiliary Information. *Nucleic Acids Research* 9.1, pp. 133–148. DOI: 10.1093/nar/9.1.133 (cited on p. 75).

Part V

Appendix

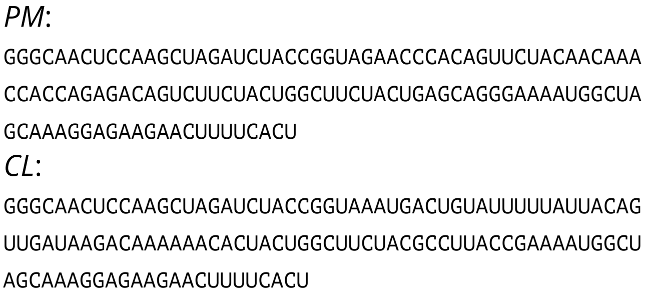
Chapter A
Supplementary figures and tables

Package	Code repository	Documentation
Biotite	github.com/biotite-dev/biotite	www.biotite-python.org
Ammolite	github.com/biotite-dev/ammolite	ammolite.biotite-python.org
Gecos	github.com/biotite-dev/gecos	gecos.biotite-python.org
fastpdb	github.com/biotite-dev/fastpdb	github.com/biotite-dev/fastpdb
Hydride	github.com/biotite-dev/hydride	hydride.biotite-python.org
Springcraft	github.com/biotite-dev/springcraft	springcraft.biotite-python.org

Table A.1: Websites for *Biotite* and its extension packages.

CNBD position	a	b	c
sdENM	-0.65	0.71	-0.66
eANM	-0.66	0.71	-0.69
constant γ	-0.64	0.71	-0.68

Table A.2: Overlap between induced and required displacement in HCN4 CNBD. The displacement overlap for combinations of force field and CNBD position relative to the first binding is tabulated.



PM:

GGGCAACUCCAAGCUAGAUCUACCGGUAGAACCCACAGUUCUACAACAAA
CCACCAGAGACAGUCUUCUACUGGCUUCUACUGAGCAGGGAAAAUGGCUA
GCAAAGGAGAAGAACUUUUCACU

CL:

GGGCAACUCCAAGCUAGAUCUACCGGUAAAUGACUGUAUUUUUAUUACAG
UUGAUAAGACAAAAAACACUACUGGCUUCUACGCCUUACCGAAAAUGGCU
AGCAAAGGAGAAGAACUUUUCACU

Figure A.1: Aptamer candidate sequences obtained from SELEX experiments. RNA sequences are given from 5'-end to 3'-end.

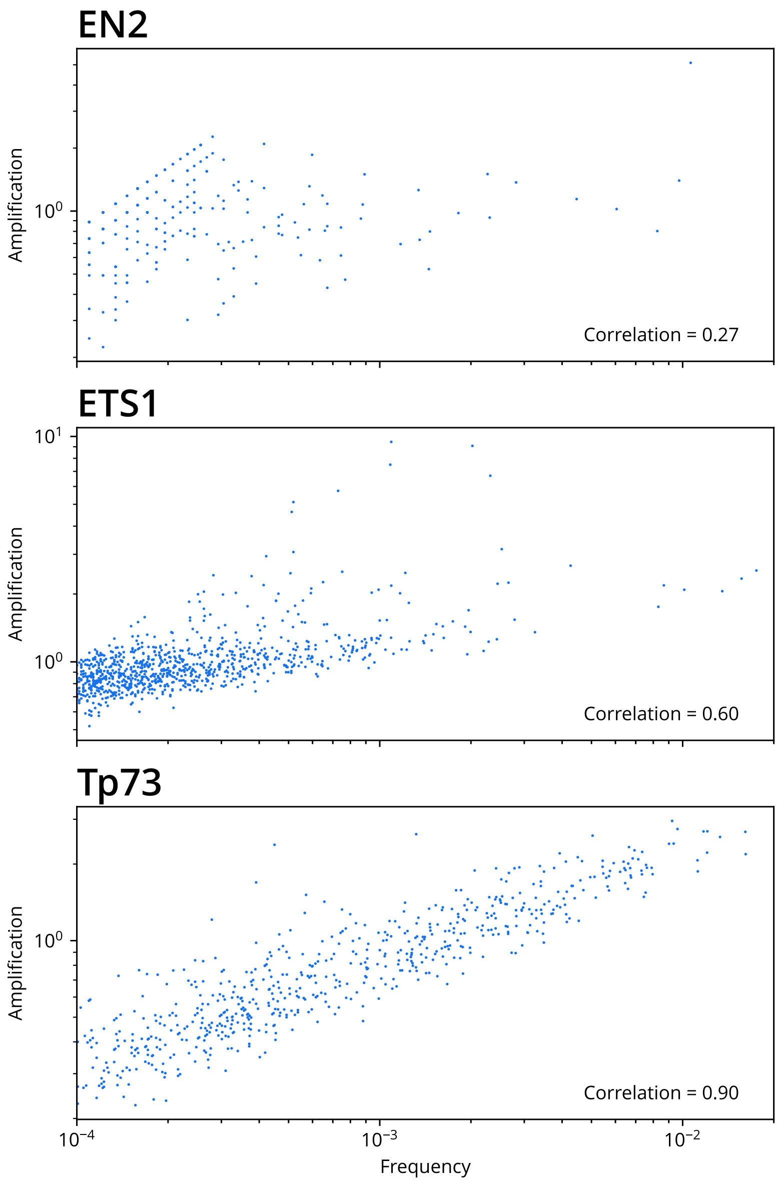


Figure A.2: Amplification and frequency of DNA species in TF SELEX experiments. For each of the exemplarily displayed TF the respective last round is displayed. The Spearman correlation for each experiment is shown in the bottom right.

Table A.3: Optimum hyperparameters for pK$_d$ prediction.

Parameter	Value
Batch size	100
Learning rate	0.0003
Weight decay	0.0001
Kernel size	7
Number of channels	30
Number of fully connected nodes	15

Mutations		ΔpK_d
Δ54	A61C	0.75
G53A	A61C	0.74
G58C	A61C	0.74
Δ34	A61C	0.74
T55A	A61C	0.73
A54T	A61C	0.72
A61C	C79A	0.70
C34A	A61C	0.70
G53T	A61C	0.68
A56T	A61C	0.68

Table A.4: Top 10 of *CL* aptamer candidate double mutants. The table depicts the double mutants from the *in silico* mutation scan with the highest predicted ΔpK_d value.

Chapter B
Abbreviations

ANM	Anisotropic network model
APC	Activated protein C
API	Application programming interface
AQ	Amodiaquine
cAMP	Cyclic adenosine monophosphate
CCD	Chemical Component Dictionary
CGO	Compiled graphics object
CL	Cortisol
CLI	Command-line interface
CN	Cortisone
CNBD	Cyclic nucleotide binding domain
CNN	Convolutional neural network
CQ	Chloroquine
Cryo-EM	Cryogenic electron microscopy
CX	Ciprofloxacin
DNA	Deoxyribonucleic acid
E-value	Expect value
ENM	Elastic network model
FMN	Flavin mononucleotide
FRET	Förster resonance energy transfer
GNM	Gaussian network model
GUI	Graphical user interface
HCN	Hyperpolarization-activated cyclic nucleotide-gated channel
HT-SELEX	High-throughput SELEX
HTS	High-throughput sequencing
LRT	Linear response theory
LX	Levofloxacin
MD	Molecular dynamics
MFE	Minimum free energy
mRNA	Messenger RNA
MSA	Multiple sequence alignment
MSE	Mean squared error
NA	Nucleic acid
NM	Neomycin
NMA	Normal mode analysis
NN	Neural network
PB	Protein blocks
PCR	Polymerase chain reaction
PDB	Protein Data Bank
PEOE	Partial equalization of orbital electronegativity
PM	Paramomycin
PPV	Positive predicted value

Qeq	Charge equilibration
RMSD	Root mean square deviation
RMSF	Root mean square fluctuation
RNA	Ribonucleic acid
rRNA	Ribosomal RNA
SELEX	Systematic evolution of ligands by exponential enrichment
sRGB	Standard RGB
TF	Transcription factor
tRNA	Transfer-RNA
UFF	Universal force field
UPGMA	Unweighted pair group method with arithmetic mean
VEGF	anti-vascular endothelial growth factor

Acknowledgements

This thesis would not have been possible without the funding from the LOEWE initiative and the support of many people. First of all I would like to thank Prof. Dr. Kay Hamacher for his guidance and especially for giving me the opportunity to work on my initially small passion project called '*Biotite*' during my Master's and Ph.D. thesis. Further I want to thank Prof. Dr. Beatrix Süß for supervising me and additionally Dr. Marc Vogel for providing me a wealth of data to perform my analyses on. I am grateful to Prof. Dr. Gerhard Thiel, for accepting to review my work and inspiring me with new project ideas, and Prof. Dr. Dominik Niopek who also kindly consented to examine my thesis. I also would like to thank my students Tom Müller and Jacob Anter, who helped me implementing essential parts of my project. I would like to express my gratitude to my family, friends and colleagues for the joy and support in the past years, foremost Daniel, who provided me comprehensive feedback, and Nora, who has been giving me so much encouragement. Last but not least, I want to mention all the other *Biotite* developers and users, who found bugs, wrote fixes, implemented features, and encouraged me in the meaningfulness of my work.

Ehrenwörtliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit entsprechend den Regeln guter wissenschaftlicher Praxis selbstständig und ohne unzulässige Hilfe Dritter angefertigt habe. Sämtliche aus fremden Quellen direkt oder indirekt übernommenen Gedanken sowie sämtliche von Anderen direkt oder indirekt übernommenen Daten, Techniken und Materialien sind als solche kenntlich gemacht. Die Arbeit wurde bisher bei keiner anderen Hochschule zu Prüfungszwecken eingereicht. Die eingereichte elektronische Version stimmt mit der schriftlichen Version überein.

Datum und Unterschrift