FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

> APPLIED LIFE SCIENCES

MASTERSTUDIUM:
> BIOINFORMATIK

In silico sequence evolution with site-specific interactions along phylogenetic trees under a thermodynamic viewpoint

MASTERARBEIT

ZUR ERLANGUNG DES AKADEMISCHEN GRADES

MASTER OF SCIENCE IN ENGINEERING

DER

FACHHOCHSCHULE FH CAMPUS WIEN

MASTER-STUDIENGANG BIOINFORMATIK

Vorgelegt von:
Lukas Huber
Personenkennzeichen: 1810542032

FH-Hauptbetreuer*in:
Dr. Tanja Gesell

Zweitprüfer*in:
DI Norbert Auer

Erklärung:

Ich erkläre, dass die vorliegende Masterarbeit von mir selbst verfasst wurde und ich keine anderen als die angeführten Behelfe verwendet bzw. mich auch sonst keiner unerlaubter Hilfe bedient habe. Ich versichere, dass ich dieses Masterarbeitsthema bisher weder im In- noch im Ausland (einer Beurteilerin/einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Weiters versichere ich, dass die von mir eingereichten Exemplare (ausgedruckt und elektronisch) identisch sind.

Datum

Unterschrift

Abstract

Diese Arbeit zeigt die Bedeutung der Effizienz des gewählten Randomisierungsalgorithmus bzw. bioinformatischen Tools innerhalb einer Forschungspipeline auf.

Am Beispiel der Detektion nicht kodierender RNAs (ncRNAs) wurde die Bedeutung randomisierter Hintergrundmodelle in den vergangenen Jahren im aktuellen Forschungsfeld diskutiert. Durch den statistischen Vergleich zufälliger Alignments mit biologischen genomweiten Alignments können ncRNAs gefunden werden. Insbesondere zur Strukturfindung muss der Algorithmus in der Lage sein Mono- und Dinukleotidgehalt der randomisierten Alignments nicht zu verändern. Für den Dinukleotidgehalt stehen derzeit zwei Herangehensweisen zur Verfügung. Zum einem über phylogenetische Simulationen mittels `SISSIz` und zum anderen über Mischen der Alignments z.B. von `multiperm`.

In dieser Studie verdeutlichen wir die Vor- und Nachteile beider Methoden unter Simulation künstlicher Alignments mit kompensatorischen Mutationen mittels `SISSI`. Zur Analyse werden verschiedene Programme zur Strukturfindung benutzt und kompensatorische als auch thermodynamische Perspektiven aufgezeigt.

Die Arbeit zeigt, dass Randomisierung mittels Mischalgorithmen zu einer höheren Rate von falsch positiven Ergebnissen im Vergleich zu Simulationsalgorithmen unter bestimmten, phylogenetischen Aspekten führen kann.

This work aims to demonstrate the importance of the efficiency of the chosen randomizing algorithm or bioinformatics tool within a research pipeline.

Using the detection of non-coding RNAs (ncRNAs) as an example, the importance of randomized background models has been discussed in recent years in the current research field. By statistically comparing random alignments with biological genome-wide alignments, ncRNAs can be found. In particular, for structure finding, the algorithm must not modify mono- and dinucleotide content of the randomized alignments. For the dinucleotide content, two approaches are currently available. One is via phylogenetic simulations using `SISSIz` and the other is via shuffling of the alignments i.e. by `multiperm`.

In this work, we illustrate the advantages and disadvantages of both methods by simulating artificial alignments with compensatory mutations using `SISSI`. Different structure finders are used for analysis and compensatory as well as thermodynamic perspectives are shown.

This work shows that randomization by shuffling algorithms can lead to a higher false positive rate compared to simulation algorithms under certain phylogenetic aspects.

Acknowledgements

This thesis would not have been without the love, caring and support of my partner and best friend Vanessa.

Thanks to Dr. Tanja Gesell, who dedicated countless hours patiently helping me and answering all questions.

Contents

1 Introduction

During evolution biological sequences commonly have multiple sites, depending on each other. However, this is generally not accounted for by regularly used models for sequence evolution. Considering this problem, a Markov model of nucleotide sequence evolution is included by Simulating Site-Specific Interaction (SISSI), in which the instantaneous substitution rate at a specific site depends on the states of other sites. Therefore, a neighbourhood system is introduced, representing a universal description of complex dependencies among sites. The resulting framework manages to simulate the evolution of Ribonucleic acid (RNA) or any other character-based sequences, taking into account secondary structure, pseudo knots and other tertiary interactions. [15] The advantage of simulated sequences lies within the possibility to create a large number of synthetic sequences, closely created to reality, including their evolution along a phylogenetic tree. The resulting framework or respectively sequences can be used afterwards for a multitude of auxiliary investigations.

Detection of evolutionarily conserved RNA structures suffer from high false-positive rates in genome-wide screens. By producing dinucleotide-controlled, simulated alignments, SISSIz manages to decrease this high false-positive rate via modelling of site-specific interactions depending on the branch length (evolutionary time d) from a phylogenetic tree estimated from the original alignment. SISSIz uses SISSI as a background model with a dinucleotide model. The resulting alignments uphold the sequence characteristics of the original alignment and therefore can be used for gene finders, like SISSIz, RNAz or EvoFold, as a background distribution. [48]

SISSI can also be used to enhance algorithms for analysis and detection of functional RNAs, with emphasis on detection of signatures of conserved RNA secondary structure. [9] As already discussed, these analyses are plagued by a high false-positive rate. To remedy this problem, Eddy devised a way to create negative controls containing no structural RNA, yet are matched controls for all other background properties of genomic alignments, such as sequence conservation, GC content and indel patterns, by simulating synthetic alignments according to a phylogenetic model. [9]

The framework can also be utilized to investigate the base pair covariance (an important indicator of evolutionary pressure on functional structures), by calculating the significance of small sliding windows, used to split up the genome, due to computational restrictions. [31]

1.1 Sequences

Sequences in a biological context can be differentiated in three types - Deoxyribonucleic acid (DNA), RNA and proteins. While DNA and RNA consist of only four bases each, proteins are coded by 20 different amino acids. The four bases for DNA and RNA are comprised of two purine bases (Adenine and Guanine) and two pyrimidine bases (Cytosine and Thymine respectively Uracil in RNA). Those are important for transversion and transition rates and probabilities in the models used for sequence generation. Transversion refers to a point mutation in sequences, in which a purine is replaced by a pyrimidine, or vice versa, while transition, is associated with a change of a purine nucleotide to another purine, or a pyrimidine nucleotide to another pyrimidine. [13] Amino acids, the building blocks of proteins, are derived from a triplet of DNA or RNA sequences. Permutating all possibilities of such triplets results in 64 possible triplets called condons. In nature there is a total of 64 codons, 61 of which encode amino acids and 3 specifying termination of translation.

1.1.1 Sequence Evolution and Markov models

In evolution, all substitutions and natural selection only act upon the molecules present in an organism, having no knowledge of their previous history, which can be explained as lack of memory. This can be statistically summarized in a Markov process, meaning that future evolution is only dependent on its current and not on its ancestral state. This enables us to compute the probability of a nucleotide *i* in its initial state to evolve into nucleotide *j* for every positive time span *t*, using a stationary, time homogeneous and reversible Markovian substitution process. [14] Following assumptions are made:

- Lack of memory: The substitution process is independent of pas events
- Independence: Every site of the sequence evolves independently
- Homogeneity: The process remains constant through time
- Continuity: Substitutions occur in continuous time
- Stationarity: The process starts at equilibrium

To summarize such a process, a instantaneous rate matrix Q is defined, in which $q_{\text{ij}} > 0$ stands for the rate of change from a specific nucleotide i to j during an infinitesimal

Table 1: Various different instantaneous rate matrices. Those instantaneous rate matrices are based on a set of assumptions mentioned in section 1.1.1. The first model developed by Jukes and Cantor in 1969 (JC69) is specified by a single free parameter π. [27] Subsequent models include more and more sophisticated parameters. Kimura included two parameter, to be able to distinguish between transitions and transversions (K80). [28] HKY [24], TN93 [50], F81 [12] and GTR [41] incorporates a more general single substitution model with different base compositions. Diagonal elements (*) are calculated as explained in equation 1.

	JC69				K80				HKY			
	A	C	G	T	A	C	G	T	A	C	G	T
A	*	α	α	α	*	β	α	β	*	$\beta\pi_C$	$\beta\pi_G$	$\beta\pi_T$
C	α	*	α	α	β	*	β	α	$\beta\pi_A$	*	$\beta\pi_G$	$\beta\pi_T$
G	α	α	*	α	α	β	*	β	$\beta\pi_A$	$\beta\pi_C$	*	$\beta\pi_T$
T	α	α	α	*	β	α	β	*	$\beta\pi_A$	$\beta\pi_C$	$\beta\pi_G$	*
	TN93				F81				GTR			
	A	C	G	T	A	C	G	T	A	C	G	T
A	*	$\beta\pi_C$	$\alpha_1\pi_G$	$\beta\pi_T$	*	π_C	π_G	π_T	*	$a\pi_C$	$b\pi_G$	$c\pi_T$
C	$\beta\pi_T$	*	$\beta\pi_G$	$\alpha_2\pi_T$	π_A	*	π_G	π_T	$a\pi_A$	*	$d\pi_G$	$e\pi_T$
G	$\alpha_1\pi_A$	$\beta\pi_C$	*	$\beta\pi_T$	π_A	π_C	*	π_T	$b\pi_A$	$d\pi_C$	*	$f\pi_T$
T	$\beta\pi_T$	$\alpha_2\pi_C$	$\beta\pi_G$	*	π_A	π_C	π_G	*	$c\pi_A$	$e\pi_C$	$f\pi_G$	*

period of time. The rate at which a change from this state occurs is the sum of all the rates of changes from this state, dictating the waiting time from this particular state before changing to another. [14] Consequently the diagonal elements of Q, which are given by equation 1, result in a sum of zero in all rows.

$$q_{ij} = -\sum_{j} q_{ij} \tag{1}$$

We do not observe the rate at which characters in molecular sequence data are evolving, but the actual characters at some given time. Because of its relation to the probability matrix $P(t)$ and the process being time-homogeneous, Q enables it to compute the probability $P_{\text{ij}}(t)$ to be in state j after a time t, assuming that the (initial) state i can be calculated using Q, via equation 2. [14]

$$P(t) = e^{Qt} \tag{2}$$

Introducing the identity matrix I, the matrix exponential can be defined by equation 3. [35]

$$\sum_{n=0}^{\infty} \frac{(Qt)^n}{n!} = I + Qt + \frac{(Qt)^2}{2!} + \frac{(Qt)^3}{3!} + ... \tag{3}$$

This equation represents a series, which can be calculated numerically using standard linear algebra techniques. One of the more popular methods to calculate such a series in molecular phylogeny uses eigendecomposition. It is found by diagonalization of Q, as shown in equation 4.

$$P(t) = U * diag\{e^{\lambda_1 t}, ..., e^{\lambda_{|A|} t}\} * U^{-1} \tag{4}$$

In equation 4 *diag{...}* represents a diagonal matrix containing the eigenvalues and U denotes the matrix of the corresponding eigenvectors. Additionally, standard models assume an $|A| * |A|$ matrix, where $|A|$ is the number of character states. As this thesis mainly considers RNA evolution, with A = {A,C,G,U}, hence $|A|$ = 4. This assumption can be extended for amino acids and condons, being alphabets of 20 and 64, respectively. It follows, that the distribution is the stationary distribution $\pi = (\pi_1, ..., \pi_{|A|})$, to which an initial distribution π^α converges independent of the starting state. Additionally, reversibility is given by the Q or P matrices for all $i, j \in \{1, ..., |A|\}$, as following:

$$\begin{aligned} \pi_i p_{ij} &= \pi_j p_{ji} \\ \pi_i q_{ij} &= \pi_j q_{ji} \end{aligned} \tag{5}$$

As a mathematical convenience, it can be concluded from equation 6, that a stationary distribution π exists, which can be found by solving equation 7 for any time *t* by solving equation 8.

$$\sum_i \pi_i p_{ij} = \pi_j \tag{6}$$

$$\pi P(t) = \pi \tag{7}$$

$$\pi Q = 0 \tag{8}$$

It is important to note that time is measured in expected numbers of substitutions, therefore it is scaled such that the expected rate of substitutions per site is shown in equation 9. The instantaneous rate matrix can be normalised using any factor, because time and rate are interchangeable. Furthermore their product can be derived without external information. [12]

$$-\sum_i \pi_i q_{ii} = 1 \tag{9}$$

Summarizing, the number of substitutions d per site can be computed by equation 10, which is not practical due to multiple substitutions per site. It is more feasible to observe the number of differences h, as calculated with equation 11

$$d = -\sum_{i \in A} \pi_i q_{ii} \tag{10}$$

$$h = 1 - \sum_{i \in A} \pi_i p_{ii}(t) \tag{11}$$

1.1.2 Compensatory substitution

Nucleotides within paired regions of RNA molecules obviously do not evolve independently from each other. If a substitution arises in such regions, compensatory substitutions can occur to remedy unstable situations of originally base-paired doublets within a given sequence, i.e. a hairpin (see figure 1). It is quite likely that a substitution at a non-base-paired doubled (see figure 1B) will lead to a base-paired doublet within a relatively short time interval, representing a compensatory substitution. [14] Assuming a fixed RNA secondary structure, all such regions are dividable into helical and loop regions. [14] Units of loop regions are considered as mononucleotides (see the conventional independent models in table 1). Units of helical regions are doublets or base pairs, resulting in the extension of the state space to all possible 16 pair combinations $i, j \in A * A = \{AA, AC, AG, AU, ..., UG, UU\}$. For this purpose Schöniger und von Haeseler [45] extended the F81 model (see table 1) with stationary frequencies

```
     ( ( ( ( ( . . . . . . . . . ) ) ) ) )
A :  A C U A G A U U G C C G A C U A G U
B :  A A U A G A U U G C C G A C U A G U
C :  A A U A G A U U G C C G A C U A U U
```

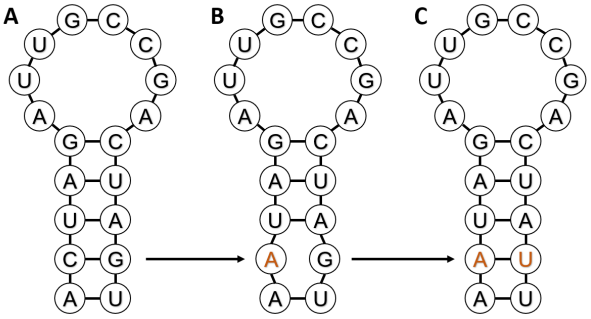

Figure 1: Compensatory mutations. The first line represents the dot-bracket notation of line A. A: Initial sequence without any mutations results in a classical hairpin structure. B: A mutation (highlighted in red) at the second position destroys the original base pair, resulting in a more fragile structure (hairpin). C: A compensatory mutation (highlighted in red) at the second last position allows the formation of a base pair, producing an more stable hairpin structure.

$\pi_\mu = \{\pi_{AA}, \pi_{AC}, \pi_{AG}, \pi_{AU}, ..., \pi_{UG}, \pi_{UU}\}$, resulting in the following 16 * 16 instantaneous rate matrix:

$$q_{ij} = -\sum_{\substack{k \in A*A: \\ k \neq i}} \begin{cases} \pi_j & \text{for } i \neq j \text{ and } h(i, j = 1) \\ q_{ik} & \text{for } i = j \\ 0 \end{cases} \tag{12}$$

with $i, j \in A^2$. Note that the Hamming distance h in equation 12 is restricted to only one substitution per unit time is possible. Subsequently, following equation can be used to calculate the expected number of substitutions:

$$d = -\frac{1}{2}\sum_{i=1}^{16} \pi_i q_{ii} \tag{13}$$

The number of observed differences can be calculated using:

$$h = \frac{1}{2}\sum_{i=1}^{16}\sum_{\substack{j=1 \\ H(i,j)=1}}^{16} \pi_i p_{ij}(t) + \sum_{i=1}^{16}\sum_{\substack{j=1 \\ H(i,j)=2}}^{16} \pi_i p_{ij}(t) \tag{14}$$

These equations suffice to describe the whole process analytically, making it possible

to predict the structure only using compensatory substitutions. Further attempts have been made by Renée [40], Muse [34], Rzhetsky [43], Tillier and Collins [52, 51] to model dependencies by base-pairing into Markov models of sequence evolution. Muse [34] presented the effect of forming or destroying a base pair by introduction of a new pairing parameter λ. It takes into consideration that the relative probability of a change from an unpaired state to a paired state within stem structures should be greater than the respective probability when sites are not paired. [14] Consequently, changes from a paired to an unpaired state should happen with lower frequencies. The corresponding 16 * 16 matrices are given by:

$$q_{ij} = \begin{cases} \pi_t & \text{for transversion, pairing unchanged} \\ \kappa\pi_t & \text{for transition, pairing unchanged} \\ \pi_t\lambda & \text{for transversion, unpaired to paired} \\ \kappa\pi_t\lambda & \text{for transition, unpaired to paired} \\ \pi_t\frac{1}{\lambda} & \text{for transversion, paired to unpaired} \\ \kappa\pi_t\frac{1}{\lambda} & \text{for transition, paired to unpaired} \\ 0 & i \text{ and } j \text{ differ at more than one site} \end{cases} \tag{15}$$

where π_t is the frequency of the nucleotide that differ in i and j.

Table 3: Top matrix: Exemplary process of base-pairs in RNA helical regions (doublets) as an extended F81 model with joint substitution events. Additionally, it is restricted to one substitution per unit time for one base of the doublet. [45] (see equation 12) Zero values are represented by '-'. Diagonal elements represented by '*' are defined by the mathematical requirement that the sum of each row has to be zero. **Bottom matrix:** Example of an instantaneous rate matrix Q_k of a site k wiht $n_k = 1$ while taking into account site i. In this matrix, only the current site k is allowed to mutate, resulting also in a rate matrix Q_k with 16 * 16 dimensions.

(k,i)	AA	AC	AG	AU	CA	CC	CG	CU	GA	GC	GG	GU	UA	UC	UG	UU
AA	*	π_{AC}	π_{AG}	π_{AU}	π_{CA}	-	-	-	π_{GA}	-	-	-	π_{UA}	-	-	-
AC	π_{AA}	*	π_{AG}	π_{AU}	-	π_{CC}	-	-	-	π_{GC}	-	-	-	π_{UC}	-	-
AG	π_{AA}	π_{AC}	*	π_{AU}	-	-	π_{CG}	-	-	-	π_{GG}	-	-	-	π_{UG}	-
AU	π_{AA}	π_{AC}	π_{AG}	*	-	-	-	π_{CU}	-	-	-	π_{GU}	-	-	-	π_{UU}
CA	π_{AA}	-	-	-	*	π_{CC}	π_{CG}	π_{CU}	π_{GA}	-	-	-	π_{UA}	-	-	-
CC	-	π_{AC}	-	-	π_{CA}	*	π_{CG}	π_{CU}	-	π_{GC}	-	-	-	π_{UC}	-	-
CG	-	-	π_{AG}	-	π_{CA}	π_{CC}	*	π_{CU}	-	-	π_{GG}	-	-	-	π_{UG}	-
CU	-	-	-	π_{AU}	π_{CA}	π_{CC}	π_{CG}	*	-	-	-	π_{GU}	-	-	-	π_{UU}
GA	π_{AA}	-	-	-	π_{CA}	-	-	-	*	π_{GC}	π_{GG}	π_{GU}	π_{UA}	-	-	-
GC	-	π_{AC}	-	-	-	π_{CC}	-	-	π_{GA}	*	π_{GG}	π_{GU}	-	π_{UC}	-	-
GG	-	-	π_{AG}	-	-	-	π_{CG}	-	π_{GA}	π_{GC}	*	π_{GU}	-	-	π_{UG}	-
GU	-	-	-	π_{AU}	-	-	-	π_{CU}	π_{GA}	π_{GC}	π_{GG}	*	-	-	-	π_{UU}
UA	π_{AA}	-	-	-	π_{CA}	-	-	-	π_{GA}	-	-	-	*	π_{UC}	π_{UG}	π_{UU}
UC	-	π_{AC}	-	-	-	π_{CC}	-	-	-	π_{GC}	-	-	π_{UA}	*	π_{UG}	π_{UU}
UG	-	-	π_{AG}	-	-	-	π_{CG}	-	-	-	π_{GG}	-	π_{UA}	π_{UC}	*	π_{UU}
UU	-	-	-	π_{AU}	-	-	-	π_{CU}	-	-	-	π_{GU}	π_{UA}	π_{UC}	π_{UG}	*

(k,i)	AA	AC	AG	AU	CA	CC	CG	CU	GA	GC	GG	GU	UA	UC	UG	UU
AA	*	-	-	-	π_{CA}	-	-	-	π_{GA}	-	-	-	π_{UA}	-	-	-
AC	-	*	-	-	-	π_{CC}	-	-	-	π_{GC}	-	-	-	π_{UC}	-	-
AG	-	-	*	-	-	-	π_{CG}	-	-	-	π_{GG}	-	-	-	π_{UG}	-
AU	-	-	-	*	-	-	-	π_{CU}	-	-	-	π_{GU}	-	-	-	π_{UU}
CA	π_{AA}	-	-	-	*	-	-	-	π_{GA}	-	-	-	π_{UA}	-	-	-
CC	-	π_{AC}	-	-	-	*	-	-	-	π_{GC}	-	-	-	π_{UC}	-	-
CG	-	-	π_{AG}	-	-	-	*	-	-	-	π_{GG}	-	-	-	π_{UG}	-
CU	-	-	-	π_{AU}	-	-	-	*	-	-	-	π_{GU}	-	-	-	π_{UU}
GA	π_{AA}	-	-	-	π_{CA}	-	-	-	*	-	-	-	π_{UA}	-	-	-
GC	-	π_{AC}	-	-	-	π_{CC}	-	-	-	*	-	-	-	π_{UC}	-	-
GG	-	-	π_{AG}	-	-	-	π_{CG}	-	-	-	*	-	-	-	π_{UG}	-
GU	-	-	-	π_{AU}	-	-	-	π_{CU}	-	-	-	*	-	-	-	π_{UU}
UA	π_{AA}	-	-	-	π_{CA}	-	-	-	π_{GA}	-	-	-	*	-	-	-
UC	-	π_{AC}	-	-	-	π_{CC}	-	-	-	π_{GC}	-	-	-	*	-	-
UG	-	-	π_{AG}	-	-	-	π_{CG}	-	-	-	π_{GG}	-	-	-	*	-
UU	-	-	-	π_{AU}	-	-	-	π_{CU}	-	-	-	π_{GU}	-	-	-	*

Table 6: Top matrix: Extension of the instantaneous rate matrix Q_k in table 3 rewritten as a block matrix, with $n_k = 1$. This rate matrix acts on site k taking site i into account. Zero values are represented by '-'. Diagonal elements represented by '*' are defined by the mathematical requirement that the sum of each row has to be zero. Only k (bold) is allowed to mutate. **Bottom matrix:** Same instantaneous rate matrix Q_k, where only site i (bold) is allowed to mutate.

*(**k**,i)*	**A**A	**C**A	**G**A	**U**A	**A**C	**C**C	**G**C	**U**C	**A**G	**C**G	**G**G	**U**G	**U**A	**C**A	**G**U	**U**U
AA	*	π_{CA}	π_{GA}	π_{UA}	-	-	-	-	-	-	-	-	-	-	-	-
CA	π_{AA}	*	π_{GA}	π_{UA}	-	-	-	-	-	-	-	-	-	-	-	-
GA	π_{AA}	π_{CA}	*	π_{UA}	-	-	-	-	-	-	-	-	-	-	-	-
UA	π_{AA}	π_{CA}	π_{GA}	*	-	-	-	-	-	-	-	-	-	-	-	-
AC	-	-	-	-	*	π_{CC}	π_{GC}	π_{UC}	-	-	-	-	-	-	-	-
CC	-	-	-	-	π_{AC}	*	π_{GC}	π_{UC}	-	-	-	-	-	-	-	-
GC	-	-	-	-	π_{AC}	π_{CC}	*	π_{UC}	-	-	-	-	-	-	-	-
UC	-	-	-	-	π_{AC}	π_{CC}	π_{GC}	*	-	-	-	-	-	-	-	-
AG	-	-	-	-	-	-	-	-	*	π_{CG}	π_{GG}	π_{UG}	-	-	-	-
CG	-	-	-	-	-	-	-	-	π_{AG}	*	π_{GG}	π_{UG}	-	-	-	-
GG	-	-	-	-	-	-	-	-	π_{AG}	π_{CG}	*	π_{UG}	-	-	-	-
UG	-	-	-	-	-	-	-	-	π_{AG}	π_{CG}	π_{GG}	*	-	-	-	-
AU	-	-	-	-	-	-	-	-	-	-	-	-	*	π_{CU}	π_{GU}	π_{UU}
CU	-	-	-	-	-	-	-	-	-	-	-	-	π_{AU}	*	π_{GU}	π_{UU}
GU	-	-	-	-	-	-	-	-	-	-	-	-	π_{AU}	π_{CU}	*	π_{UU}
UU	-	-	-	-	-	-	-	-	-	-	-	-	π_{AU}	π_{CU}	π_{GU}	*

*(k,**i**)*	A**A**	A**C**	A**G**	A**U**	C**A**	C**C**	C**G**	C**U**	G**A**	G**C**	G**G**	G**U**	U**A**	U**C**	U**G**	U**U**
A**A**	*	π_{AC}	π_{AG}	π_{AU}	-	-	-	-	-	-	-	-	-	-	-	-
A**C**	π_{AA}	*	π_{AG}	π_{AU}	-	-	-	-	-	-	-	-	-	-	-	-
A**G**	π_{AA}	π_{AC}	*	π_{AU}	-	-	-	-	-	-	-	-	-	-	-	-
A**U**	π_{AA}	π_{AC}	π_{AG}	*	-	-	-	-	-	-	-	-	-	-	-	-
C**A**	-	-	-	-	*	π_{CC}	π_{CG}	π_{CU}	-	-	-	-	-	-	-	-
C**C**	-	-	-	-	π_{CA}	*	π_{CG}	π_{CU}	-	-	-	-	-	-	-	-
C**G**	-	-	-	-	π_{CA}	π_{CC}	*	π_{CU}	-	-	-	-	-	-	-	-
C**U**	-	-	-	-	π_{CA}	π_{CC}	π_{CG}	*	-	-	-	-	-	-	-	-
G**A**	-	-	-	-	-	-	-	-	*	π_{GC}	π_{GG}	π_{GU}	-	-	-	-
G**C**	-	-	-	-	-	-	-	-	π_{GA}	*	π_{GG}	π_{GU}	-	-	-	-
G**G**	-	-	-	-	-	-	-	-	π_{GA}	π_{GC}	*	π_{GU}	-	-	-	-
G**U**	-	-	-	-	-	-	-	-	π_{GA}	π_{GC}	π_{GG}	*	-	-	-	-
U**A**	-	-	-	-	-	-	-	-	-	-	-	-	*	π_{UC}	π_{UG}	π_{UU}
U**C**	-	-	-	-	-	-	-	-	-	-	-	-	π_{UA}	*	π_{UG}	π_{UU}
U**G**	-	-	-	-	-	-	-	-	-	-	-	-	π_{UA}	π_{UC}	*	π_{UU}
U**U**	-	-	-	-	-	-	-	-	-	-	-	-	π_{UA}	π_{UC}	π_{UG}	*

1.2 RNA

Table 9: Types of RNA structure

Type	Definition
Primary structure	The specific sequence of the different nucleotides make up the primary structure of RNA.
Secondary structure	The succession of all base pairs enable the primary structure to fold in a two dimensional way, resulting in stem loops, hairpin loops, pseudo knots and multiple other structures. The 2D representation of this structure is defined as secondary structure.
Tertiary structure	Through arrangement of the secondary structure in a 3 dimensional way the tertiary structure can be obtained.
Quaternary structure	The quaternary structure is defined as the inter-molecular base pairing with other RNA molecules.

RNA is a polymer of nucleotides consisting of the four bases Adenine, Uracile, Guanine and Cytosine. Adenine forms a weaker base pair with Uracile using two hydrogen bonds in contrast to Guanine and Cytosine, which form a stronger base pair via three hydrogen bonds. The bases themselves consist of a ribose group, a phosphate group and one of the four specific bases mentioned above.

Generally, RNA is first transcribed from DNA resulting in messenger RNA (mRNA), followed by the translation into proteins. [6] However, there is also a large family of RNAs, which do not serve the purpose of protein synthesis, called non coding RNA (ncRNA).

Using its ability to form the above mentioned bonds, RNA is able to arrange in complex structures, described as stem loops, interior loops, multi loops, pseudo knots and hairpin loops in two dimensional space, representing the secondary structure (see table 9). By calculating the maximum number of matching base pairs, Nussinov et al. were among the first to develop an algorithm for the prediction of secondary structures in RNA. [38] This approach was improved by Zuker et al. through their consideration of energy minimization. [61] However, these first algorithms only took a single sequence into consideration, whereas more recent tools use alignment based approaches to predict a consensus structure and its Minimum Free Energy (MFE).

1.2.1 Mutual Information

An approach to ncRNA detection is to find pairs of columns, which show an increased degree of covariation than expected by chance. This results in the Mutual Information (MI) measure [47], which was subsequently adapted to RNA predictions [5, 23, 7], using the following expression:

$$MI_{i,j} = \sum_{(X_i,X_j)} f(X_i,X_j) log_2 \frac{f(X_i,X_j)}{f(X_i) * f(X_j)} \qquad \text{with} X_i, X_j \in A \tag{16}$$

The sum is over all possible pairs of (X_i, X_j), whereas the observed frequencies of nucleotide pairs are $f(X_i, X_j)$ and $f(X_i)$ and $f(X_j)$ are the nucleotide at the sites i and j. There is a number of programs which use the MI or variants hereof, e.g. `ILM` [42], `MatrixPlot` [19], `KNetFold` [4], `ConStruct` [32, 58] or `COVE` [10].

1.2.2 Structural stability and conservation

To reliably predict ncRNAs, an accurate measure for structural conservation is essential. Current approaches to measuring RNA structure conservation are divided into several types:

- Comparison of predicted MFE
- Comparison of single structures as well as sets of structures representing the entire folding space
- Some specialized methods

Some of those methods were recently tested by Gruber et al. [22]. According to their study, a plain base-pair distance metric, as well as the folding energy, are the most accurate measure of structure conservation. The base-pair distance metric (Structure Conservation Index (SCI)) is calculated using following formula:

$$SCI = \frac{E_{cons}}{\bar{E}_{single}} \tag{17}$$

E_{cons} being the consensus MFE, which results of the consensus structure finding programs described later and $\bar{E}_{single}$ representing the average MFE of each single se-

quence. This normalizes the index and renders it independent of nucleotide composition or length of the alignment.

1.2.3 Covariance and Thermodynamics

Some methods for structure prediction calculate the consensus structure from an alignment using thermodynamic method, which average the energy contribution over all sequences and integrate covariance methods. Two bioinformatical tools using these methods are `RNAalifold`[25] and `ConStruct` [32, 58], both of which leave an open question about the optimal weighting between thermodynamics and covariance. Steger [11] coined these methods as phylogenetic methods, although no phylogeny is taken into account. However, since then a small number of methods have started to take phylogeny into account, albeit not considering thermodynamics and instead mainly being based on comparative mutations in conjunction with an probabilistic approach. As experimental determination of RNA structures can be laborious and are therefore not feasible on a large scale, often computational predication based on thermodynamic methods are used in everyday analysis. The goal with thermodynamic methods for RNA folding is to find the structure with the most favourable folding energy. Usually, the free energy ΔG of a chosen folding relative to the unfolded structure is computed. Using this method, paired regions of the sequence add stabilizing energy, which is mathematically expressed as a negative value, to ΔG, whereas unpaired regions add destabilizing energy, expressed by positive values. First attempts in calculating the optimal secondary structures for simplified energy models were executed by Nussinov et al. [37, 36]. The simplest approach to this method is to assign each type of base-pair a negative, fixed energy value and subsequently find the structure with the maximum number of base pairs. A more sophisticated, but still unrealistic scenario, introduces a specific energy contribution β_{ij}, which is assigned to each type of base pair (*i,j*). Afterwards, the sum of all base pair energies result in the overall energy of the fold. Using this model, the minimal energy $F_{i,j}$ of a sequence $x_{i,j}$ is found. By adding one base at a time, it is either unpaired or forms a pair with some other base *k*. The minimum of these two cases constitutes the overall minimum. In case *i* forms a base-pair, the minimum energy has to be computed, by evaluating all possible base-pairs (*i,k*). Following recursion can be used to obtain the minimum free energy:

$$F_{ij} = min\left\{ F_{i+1,j}, \min_{\substack{i+1\leq k\leq j \\ \Pi_{ik}=1}} \left\{ F_{i+1,k-1} + F_{k+1,j} + \beta_{ik} \right\} \right\} \tag{18}$$

This example of dynamic programming algorithm, frequently encountered in bioinformatics, describes a matrix filled with the optimal solution for all possible subsequences. The entry $F_{1,n}$ subsequently contains the optimal solution for the whole sequence with a length of n. However, the complexity of this algorithm is $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ for memory and Central Processing Unit (CPU), respectively. Furthermore, equation 18 only gives the minimum free energy, but not the structure of the folded sequence. To compute the structure corresponding to the optimal energy, backtracking is used to get the list of base pairs. First, a temporary helper matrix K is filled during the recursion, followed by setting $K_{i,j} = k$, k being the base, giving the optimal secondary structure when paired with i for the subsequence from i to j. In case i is unpaired in the computed optimal secondary structure, $K_{i,j} = 0$. Utilizing recursive functions shown in algorithm 1, a list of base pairs of the optimal structure can be retrieved. Starting with input $(i, j) = (1, n)$, $K_{1,j}$ is considered to hold the pairing partner k of position 1 in the optimal structure of the whole sequence of length n. Finally, $K_{1,n} = k$ divides the sequence in two independent sub-sequences, which are subsequently evaluated again, by calling the same function recursively. Using this method, it is possible to compute RNA structure only using its sequence. [14]

Algorithm 1: Recursive backtracing procedure, presented as pseudocode. It returns the optimal structure's list of base-pairs.

```
Function Backtrace(i, j)
    if i > j then
        return
    if K_{i,j} = 0 then
        Backtrace(i + 1, j)
    else
        output(i, K_{i,j})
        Backtrace(i + 1, K_{i,j} − 1)
        Backtrace(K_{i,j} + 1, j)
end
```

Using the method above clearly results in the optimal structure in an algorithmic sense, but structure predictions obtained this way can not be considered biological realistic. Instead the energy model based on simple base pairing rules poorly reflects the biophysical properties of real RNA molecules, as stacking interactions of neighbouring base pairs in RNA secondary structures contribute the most to the stabilizing energy. To improve the model, in order to achieve a more realistic energy model, loops in the structure need to be considered. The nearest-neighbour model designates every loop

l in a given structure s a free energy ΔG:

$$\Delta G(S) = \sum_{l \in S} \Delta G(l) \tag{19}$$

As the underlying energy rules used in current state-of-the-art prediction programs are too complex [59, 33], and thus out of the scope of this thesis, they generalized here: The energy is depending on the type and size of the loop, the closing base pairs and the bases immediately interior to the closing base pair. The energy values, needed for incorporating stacking energies in an algorithm, have been empirically determined by melting experiments, which measure the required energy to open specific structural elements. As only stacks and some other small loops could be investigated, the energy rules for other types of loops were tabulated through extrapolations or other approximations. [55]

1.2.4 Pairwise vs. overlapping nucleotide content

Throughout this thesis, two different classifications of nucleotide contents are used. Overlapping nucleotide content (ON) represents the application of analysing the dinucleotide content of any given sequence. Starting at the first position of a sequence, it is calculated by taking its direct neighbour into account, without regards to the secondary structure of the sequence (see figure 2) The counting window starts at K_n, where n = 0, and includes K_{n+1}. It is then shifted step by step, by increasing n by one until the end of the sequence is reached.

Contrary to the ON, the Pairwise nucleotide content (PN) takes the secondary structure of the sequence into account. As seen in figure 2, the counting window is formed around a base pair K_nK_m. The window is then shifted to the next position in the sequence, possessing an neighbour according to the base pairs of the sequence.

It is important to consider that both ON and PN represent neighbourhood systems, although only PN takes the neighbourhood system N into account.

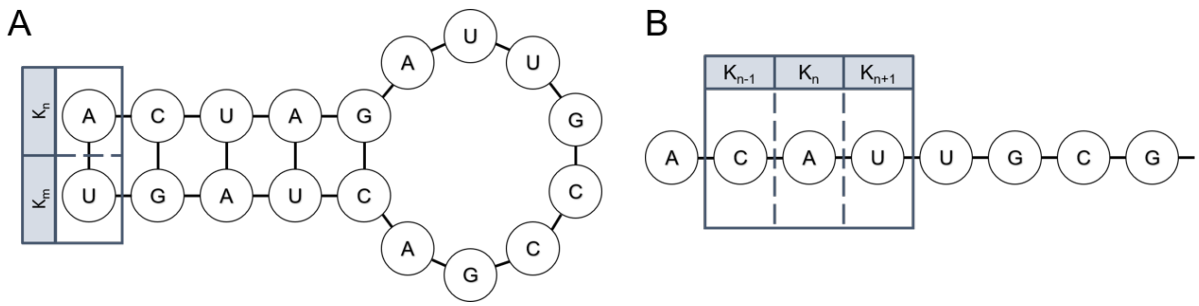

Figure 2: Pairwise vs. overlapping nucleotide content. **A**: Visualisation of pairwise nucleotide content (PN). The pair K_nK_m is counted towards PN. The counting window is then shifted to the next position, which possesses a neighbour according to the neighbourhood system. **B**: Overlapping nucleotide content (ON) does not take the secondary structure of the given sequence into account. Instead the pair K_nK_{n+1} is counted towards the ON. The counting window is then shifted by increasing n by one.

1.3 Complex Markov models

Commonly used Markov models are side independent. However, we are interested in the intertwined relationship between the structure and the substitution process. Therefore, `SISSI` was developed to include side specific interactions inside a Markov model.

The `SISSI` framework allows to define site dependencies in the form of a neighbourhood system N, regardless of their complexity and may also include overlapping dependencies. [15] Furthermore, utilizing stationary frequencies within `SISSI`'s model, enables it to calculate a dinucleotide frequency. The obtained frequency equals the alignment to be randomised (in equilibrium-state). To achieve this, the dinucleotide frequencies are counted as an average of all sequences in the original alignment. The corresponding trinucleotide frequencies needed for Q_k are calculated as a function of the single and dinucleotide frequencies using approximation of simple conditional probabilities (see equation 20). [2, 8]

$$\pi(\alpha\beta\gamma) = \frac{\pi(\alpha\beta)\pi(\beta\gamma)}{\pi(\beta)} \tag{20}$$

In equation 20 $\pi(\alpha\beta\gamma)$ represents the trinucleotide frequencies $\pi(\alpha\beta)$ and $\pi(\beta\gamma)$ represent the respective, counted dinucleotide frequencies and $\pi(\beta) = \sum_\alpha \pi(\alpha\beta) = \sum_\alpha \pi(\beta\gamma)$ with $\alpha, \beta, \gamma \in \{A, C, G, U\}$.

1.4 Random number generation

Generating random numbers may sound trivial, but are hard to generate on computers. In general, random number generators can be split in two categories: Hardware Random Number Generators (HRNGs), which generate true random numbers as a function of current value of some physical environment attribute, like atmospheric noise, thermal noise or other external electromagnetic and quantum phenomena, and Pseudo Random Number Generators (PRNGs), which generate numbers in a seemingly random, but actually deterministic way. The former bares the disadvantages of being comparatively slow and requiring specialized hardware, thus being unsuitable for most applications. The latter is much faster and suffices for most purposes. As HRNGs are rarely used in software, we only discuss PRNGs in this thesis.

1.4.1 Pseudo Random Number Generators (PRNG)

PRNGs, in essence, are algorithms which generate a sequence of random numbers, depending on the seed number, which the algorithm uses as a starting point. Simple PRNGs generate the same numbers, if the same seed is used repeatedly. The optimal Pseudo Random Number Generator (PRNG) strives for a perfect distribution of its random numbers over the whole range of its intended period, without correlating values or predictable distance between is successively drawn values. In practice, PRNGs exhibit artifacts, that weaken the randomness of their output:

- Some seeds produce shorter than expected periods
- Correlating successive values
- Distances between values are differently distributed as those in random sequences
- Distribution of large quantities of generated numbers lacks uniformity
- Dimensional distribution of the output sequence is poor

To investigate the effectiveness of PRNG algorithms several tests have been proposed to analyse their quality. `DIEHARD` [53] represents such a test suit, as well as its successor `Dieharder` [26]. Many such test suites have been shown to be defective utilizing these PRNGs, including the standards used in the programming languages `C` and `Java`. [30] Therefore, it is advisable to always use a separate PRNG.

Entropy, the reflection of disorder in a certain system, is an important topic in PRNGs, for obvious reasons. As a PRNG algorithm always returns the same result, depending on the used seed, it represents a system with no entropy. Thus, a seed incorporating as much entropy as possible is needed to approach true randomness. To accomplish this task, ideally the seed integrates multiple factors, which are affected by real randomness. Time seems like an obvious start in search of such randomness, as the user starts the program at a random time. However if a certain program is used multiple times in parallel, such as in a pipeline, time is not a suitable choice on its own, even if measured in the millisecond range. Therefore additional parameters have to be blended into the seed. Jones [26] suggests a seed consisting of a combination of date/time, process number and system information that changes frequently (for example the MD5Sum of frequently written files). Furthermore, the length of the produced seed is important: By generating 16-bit or 32-bit seeds (a typical size of integers in older programming languages) a total of 2^{16} = 6.5536 or 2^{32} = 4.294.967.296 values are possible. Considering large, complex pipelines with millions of generated pseudo random numbers, the possibility of a multitude of processes started with the same seeds arises. This problem can be eliminated by using seeds with a length of at least 64 bits (2^{64} = 1,845*10^{19}) or more. Furthermore, running the same software in parallel on multiple Graphics Processing Units (GPUs) or on a cluster posses an additional difficulty. Two instances of the same software will use the same seed, if started at the exact same time. Therefore, the properties unique to the computing environment have to be included as well. To achieve a seed with the most entropy, as many properties as possible should be included, without losing efficiency in generation of such seeds. A list of such properties would be too exhaustive, as very creative sources of randomness have been discovered. Those include:

- Timings of interrupts and disk accesses
- Timings of keystrokes and/or other input devices
- The number of pixels in a certain color
- The movement of the mouse

1.5 Tools / Software

1.5.1 SISSI as alignment generator

SISSI constitutes an entire framework [16], culminating in a software tool which generates any number of data sets of related sequences with site-specific interactions for a given phylogeny, user defined systems of neighbourhoods and instantaneous rate matrices. However, other programs have also been designed to simulate nucleotide sequences along a tree. [46, 39, 20, 60, 49, 44, 54] SISSI incorporating site-specific interactions complements the available simulation approach. The basic idea is to use separate models for each site defined by the interactions with other sites in the sequence. It follows that the rates of substitution may be not only variable in time (rate heterogeneity), but also correlated. With different null models, specific annotations and adequate parameters for the neighbourhoods, there are a lot of models imaginable for this simulation procedure. [18]

The output of SISSI can be written in Clustal (CLU), Mutation Annotation Format (MAF) or Phylip (default) format. For the purpose of this thesis the CLU format is chosen, as tools further downstream of the pipeline require said format. Furthermore, the sequence length (flag '-l') has to agree with the length of the neighbourhood system. Note that there is no space between the '-l' flag and the chosen length.

Following listing visualizes the most simple use of SISSI:

Listing 1: SISSI example.

```
~$ sissi099 [options] -fs <mono_frequencies> -fd <double_frequencies> /
  -nn <neighbourhood> -l<sequence length> <tree>
```

In this thesis version 0.99 of SISSI is used.

1.5.2 SISSIz as alignment simulator and structure finder

SISSIz randomizes multiple sequence alignments while preserving the average ON by using an adapted version of SISSI. Its algorithm is guided by a phylogenetic tree and can be used in combination with the RNAalifold algorithm. [3, 25]. This cooperation of tools allows for the detection of functional RNA structures. To this end, SISSIz calculates the consensus folding energy of the original data and compares it to the consensus folding energy distribution of the random alignments. The significance of

the prediction is measured as a z-score:

$$z = \frac{native_{MFE} - \mu_{MFE}}{\sigma_{MFE}} \tag{21}$$

Negative z-scores indicate secondary structures that are more conserved/stable than expected by chance. Thus, `SISSIz` can be used in two ways. Namely, the simulation of alignments and the prediction of RNA genes.

If `SISSIz` is used in simulation mode, alignments are simulated with on average the same mono- and dinucleotide content, Mean Pairwise Identity (MPI), local conservation and gap patterns. All of the mentioned statistics affect the MFE of the alignment's consensus secondary structure. [17] `SISSIz`'s simulated alignments may be useful for creating negative control datasets for benchmarking [21], or to estimate the False Discovery Rate (FDR) in ncRNA screens.[57]

When `SISSIz` is employed as a gene finder, the simulated alignments serve as a null model for the prediction as shown above. In genome wide screens the genomic alignments are split into alignments with a certain length and overlap (usually 200 and 100, respectively).

Usage of this tool is relatively easy, using following command:

Listing 2: `SISSIz` example.

```
~$ SISSIz [options] <input> > <output>
```

During this study, both use cases described above are applied. First, the flag '-s' is incorporated, which triggers the tool to skip folding, resulting in a drastically faster runtime. Additionally, no structure prediction is performed further decreasing runtime. As `SISSIz` offers the option to shuffle using a mono- or dinucleotide model (default) the option '-i' is used to switch between those. Both models are investigated in this thesis. The second use case in this thesis is structure prediction. Both flags ('-s' and '-i') are omitted in this case. Instead '–sci' is used to predict the structural content of the input sequence and additionally calculate the SCI value. If no option is used at startup of the tool, a structure prediction is performed.

In this thesis version 3.0 of `SISSIz` is used.

1.5.3 `shuffle-aln` as mononucleotide shuffler

`shuffle-aln` (shuffle alignment) is a simple Perl script which is applied in the bioinformatical tool `ALIFOLDz`, which - likewise to `SISSIz` - is used to calculate z-scores for alignments and single sequences. For the purpose of just shuffling sequences, the distinct script `shuffle-aln` is available, which randomizes a multiple sequence alignment.

Its shuffling algorithm is comparable to `multiperm`, as it only shuffles the columns of the sequences in an alignment. Furthermore, it takes an conservative approach to this as default. Only columns with the same gap-pattern and conservation pattern are shuffled. It provides an option to shuffle all columns regardless of gap- or conservation pattern, but for the purpose of this thesis, only the default options are investigated.

To execute the script, a multiple sequence alignment has to be provided by Standard In (STDIN) and a shuffled version is subsequently written to Standard Out (STDOUT). Supported formats are FASTA and CLU. This script can be executed with following command:

Listing 3: `shuffle-aln` example.

```
~$ ./shuffle-aln.pl < <input> > <output>
```

As this script does not provide a definite version number, it cannot be specified in this thesis, for the purpose of reproducibility. Nonetheless, there is a timestamp inside the source code available. This thesis uses the `shuffle-aln` script with following timestamp: <06/07/31 13:32:45>

1.5.4 `multiperm` as dinucleotide shuffler

`multiperm` permutates each column of an alignment preserving the dinucleotide frequencies of the alignment's consensus sequence, hence approximately in each respective sequence. It does so while maintaining gap contrains and local conservation patterns. Based on gap positions and an estimation of nucleotide conservations pattern in each colum, this tool subdivides columns in the alignment into conservation classes. This means the algorithm shuffles the alignment only in a one dimensional way, meaning that only columns but no rows are shuffled. However, `multiperm` does not modify the GC content, but aims to destroy any structural elements in the sequence. This approach is not able to destroy ancestral correlations. As this thesis mainly uses alignment files in CLU format, the flag '-w' has to be supplied. Usage of this tool is

rather simple, considering following command:

Listing 4: `multiperm` example.

```
~$ multiperm -w <input>
```

No custom, user-specified name for the output can be supplied. Instead, the prefix 'perm_xxx_' is added to the filename of the input ('xxx' being a numerical iterator). As this does not suffice to appropriately name files, an additional step of renaming the output has to be considered. In case of this study, following command was used:

Listing 5: `multiperm` example.

```
~$ multiperm -w <input> && mv perm_001_<input> <output>
```

In this thesis version 0.94 of `multiperm` is used.

1.5.5 `RNAz` as structure finder

`RNAz` calculates various folding characteristics to classify an alignment. Those characteristics are displayed in the header section of the output.

As already covered in section 1.5.2, the mean single MFE is compared to the consensus MFE. The corresponding formular can be seen at equation 17. The MFE depends on the mean pairwise identity and the number of sequences in the alignment. [56] Therefore, it is not possible to interpret the significance of a MFE-value in absolute terms. As a rule of thumb, a MFE near or even above the mean pairwise identity is considered as good and might indicate structural conservation.

`RNAz` is very helpful in terms of estimating the probability that the RNA under investigation contains a conserved structure. It provides an overall RNA-class probability, or p-value. Note, that this is not a p-value in a strict statistical sense, simply because there is no underlying statistical model. Instead, `RNAz` uses a rather ad hoc machine learning technique to calculate this value. It supports vector machines for prediction. High-confidence predictions equal $p \geq 0.9$, while lower-confidence predictions generally are considered when $p \geq 0.5$. The false positive rate at this threshold was observed to be at approximately 4%, i.e. 4 positive hits in 100 random alignments are expected. Most applications require a more useful cutoff at p = 0.9, which entails a false positive rate of roughly 1%. It was found, that the two levels of significance at p = 0.5 and p = 0.9 are the most useful. A more sophisticated interpretation of the p-value without

considering the other values is generally not useful. In most cases one cannot say that, for example, a hit with p = 0.97 is more reliable than a result with p = 0.95. [56]

The lower part of the RNAz output constitutes the predicted structures for the given sequences.The predicted structures are given below the sequences in the 'dot-bracket' notation. Next to the structure one can see the MFE in kcal/Mol.

Originally (as in version 1) RNAz assumed the random background for the z-score calculations to have a given mononucleotide content. As of version 2, RNAz uses a different background model that also considers the dinucleotide content. Obviously this new model is superior and the default. However, with the option "–mononcleotide" the original mononucleotide model of RNAz 1 still can be used. In some cases, the sequences in the input data is not suitable to calculate a z-score (e.g. sequence composition is outside an acceptable range of values). [56] prior to version 2 RNAz showed a warning that the z-score calculation might not be accurate. As of version 2 the z-score can be calculated empirically by shuffling the sequences.

In case of this thesis no shuffling was done by RNAz (flag "-n"), as this would considerably increase the runtime of the entire pipeline and would have falsified the results for evaluating the performance of the investigated shuffling algorithms. RNAz can be invoked with following command:

Listing 6: RNAz example.

```
~$ RNAz -n <input> > <output>
```

In this thesis version 2.1.1 of RNAz is used.

1.5.6 EvoFold as structure finder

EvoFold is a bioinformatical tool to predict and score secondary structures in multiple sequence alignments. It relies on a probabilistic model construction, called Stochastic Context-Free Grammar (phylo-SCFG), and uses the characteristic differences of the substitution process in stem-pairing and unpaired regions to construct its predictions. Each of these predictions consists of a specific secondary structure and a folding potential score. For usage an alignment and the corresponding phylogenetic tree is required. As the input tree from SISSI does not represent exactly the alignment, the tree has to be recalculated for each alignment. Logarithmic-odds ratios are used as predictors for this software, meaning higher scores represent higher confidence in structure

Table 11: Explanation of output from `EvoFold`

Attribute	Description
seqID	Sequence identifier
beginPos	Structure begin position (0-based)
endPos	Structure end position (0-based and not included)
basePairCount	Count of base pairs in structure prediction
strCykProb	Probability of (most)probable) predicted structure under the structure model
bgCykProb	Probability of most probable prediction under the background model
strProb	Likelihood of the region under the structure model
bgProb	Likelihood of the region under the background model
cykScore	log-odds ration of strCykProb and bgCykProb [log(strCykProb) / log(bgCykProb)]
score	log-odds ration of strProb and bgProb [log(strProb / bgProb)]
strPostProb	Posterior probability of the predicted structure
fold	Predicted structure in parenthesis format
posScore	Position specific posterior probabilities along the region

predictions. For analysis purposes the predictive value 'cykScore' from the output is used. A detailed description of the full output can be found at table 11.

Usage of `EvoFold` is more complex than comparable tools, as a corresponding phylogenetic tree has to be supplied in addition to the structure. Furthermore, the input file for the structure requires the unusual Annotated Multiple Alignment (AMA) format, often involving an additional step to convert the file. In our case, a conversion from CLU to AMA is necessary. However, once the tree and the structure are available in the correct format, the tool can be invoked as following:

Listing 7: EvoFold example.

```
~$ EvoFoldV2 [options] <sequence-file> <tree-file>
```

In this thesis version 2 of `EvoFold` is used.

1.5.7 `snakemake` as pipeline framework

`snakemake` poses a workflow management system to create reproducible and scalable data pipelines. Its syntax is closely related to Python and thus is easily human readable. It offers the ability to be scaled to server, cluster, grid and cloud environments, without the need of modifications to the workflow definitions. It is a highly popular tool, with more than 5 new citations per week. [29] Its syntax closly resembles the GNU Make paradigm. Workflows consist of rules, which define the process of creating output files from input files using a predefined set of commands. The usage is greatly simplified as `snakemake` automatically determines dependencies between the

rules, creating a Directed Acyclic Graph (DAG) of jobs that can be automatically parallelized. In this thesis we regularly make use of it feature to support multiple named wildcards (or variables) in the filenames of either input or output files. Those wildcards are automatically inferred.

As described above, `snakemake` infers the actual workflow from a set of rules with input and output files, while inter-operating with any installed tool or available web service with well-defined input and output formats. This enables this tool to be most flexible, although this approach lacks type checking of intermediate files. Furthermore, this renders `snakemake` fully portable, only requiring a Python installation.

A valid rule entails a name, any number of input and output files and either a shell command or Python code, which creates the output from the input. The advantage of `snakemake` enables its users to incorporate one or multiple wilcards in the input and output filenames. These wildcards are automatically inferred from the files desired by the user.

Listing 8 shows an example snakefile for creating shuffled negative samples from generated positive samples, which is a typical task in this thesis. There are two variable definitions (POS_SAMPLES and NEG_SAMPLES) included for demonstration purposes. To keep listing 8 short, other variable declarations are not shown. Furthermore, there are five rules followed by a name and the definitions of input and output files and shell commands. Besides Python code, any available tool or service my be invoked in a shell- or run-block.

Rule 'generatePosSamples' (listing 8 lines 12-18) describes how to generate positive sequence samples, taking a previously defined neighbourhood and tree file. Here, `SISSI` is used for sequence generation, which produces positive samples with a length of 401 base pairs. It uses one named wildard (pos_index), which results in the generation of an arbitrary number of positive sequence samples. The range of 'pos_index' is defined in line 9 of listing 8, taking another previously defined variable as upper limit of said range.

The three rules 'generateNegativeSamplesSISSIz', 'generateNegativeSamplesMultiperm' and 'generateNegativeSamplesShuffle-aln' are used to shuffle the already generated positive samples, using the respective shuffling algorithm. The output of each rule incorporates two named wildards 'neg_index' and 'pos_index'. The latter is the same, as already used in rule 'generatePosSamples'. The range of 'neg_index' is defined in the same way as already described above (listing 8 line 10). We assume that

both indices are set to a range of one to ten. The created DAG therefore results in the creation of ten negative samples per positve sample . To summarize, if rule 'all' is invoked once, this pipeline will create ten positive samples and 100 negative samples and store them in respectively named directories. The first rule (here rule 'all') is executed, if `snakemake` is invoked without a specific target. It ensures that the coverage plot and consequently all needed intermediate files are created for each sample.

It is important to note, that `snakemake` has troubles with highly complex pipelines, as the creation and processing of the DAG becomes an issue. The runtime for file checking and determining the correct order of rules does not increase linearly with the complexity of the DAG, which requires to either split the pipeline in multiple smaller pipelines or decreasing the range of used wildcards and subsequently invoking the pipeline, only changing the range of the wildcards, until the full range is covered. Alternatively a Bourne-again shell (`BASH`) script can be used to execute the latter automatically.

In this thesis version 5.27.4 of `snakemake` is used.

1.6 `SISSIz` vs. `multiperm`

The randomization algorithms of `SISSIz` and `multiperm` differentiate fundamentally from each other. This study considers the differences between simulated alignments, using `SISSIz` and shuffled alignments, using multiperm. `SISSIz` uses exactly this approach, by estimating a phylogenetic tree based on the input alignment and simulating new alignments, taking the counted frequencies from its input as a basis. Contrary, `multiperm` basically shuffles the columns of an alignment (one dimensional), resulting in the shuffling of sequences exclusively, although without alteration of the GC content. Theoretically, structural elements should also be destroyed, however less effectively. Both tools are explained in more detail in section 1.5.

Listing 8: Snakemake example.

```
POS_SAMPLES = ["pos_sample"]
NEG_SAMPLES = ["neg_sample_SISSIz", "neg_sample_multiperm",/
               "neg_sample_aln-shuffle"]

rule all:
    input:
        expand("neg_samples/{sample}_{pos_index}_{neg_index}.txt",
                sample = NEG_SAMPLES,
                pos_index = range(0, ITER_POS),
                neg_index = range(0, ITER_NEG)),

rule generatePosSamples:
    input:
        nn = NEIGHBOURHOOD,
        tree = TREE
    output: "pos_samples/pos_sample_{pos_index}.clu"
    shell: "sissi099 -fs {FREQS_SINGLE} -fd {FREQS_DOUBLE} -nn /
            {input.nn} -l401 {input.tree} -oc -d > {output}"

rule generateNegativeSamplesSISSIz:
    input: "pos_samples/pos_sample_{pos_index}.clu"
    output: "neg_samples/neg_sample_SISSIz_{pos_index}_{neg_index}.clu"
    shell: "SISSIz -s {input} > {output}"

rule generateNegativeSamplesMultiperm:
    input: "pos_samples/pos_sample_{pos_index}.clu"
    output: "neg_samples/neg_sample_multiperm_{pos_index}_{neg_index}.clu"
    shell: "multiperm -w {input} && mv perm_001_pos_sample_*.clu {output}"

rule generateNegativeSamplesAln-shuffle:
    input: "pos_samples/pos_sample_{pos_index}.clu"
    output: "neg_samples/neg_sample_aln-shuffle_{pos_index}_{neg_index}.clu
       "
    shell: "shuffle-aln.pl < {input} > {output}"
```

2 Methods

2.1 Pipeline

To efficiently compare the randomization algorithms of simulations via `SISSIz` and shuffling via `aln-shuffle` and `multiperm`, the need for a flexible, fully automatic, reliable and reproducible pipeline arose. For this purpose, `snakemake` was used, as it fulfills all criteria. As a first step 100 positive samples were generated using `SISSI`, providing it with a neighbourhood system N and a phylogenetic tree T as input. Additionally, nucleotide frequencies F were supplied, resulting in an alignment with the sequence lengths predetermined by the neighbourhood system. The number of different sequences of the alignment is determined by the number of taxa in the supplied phylogenetic tree. To yield 100 positive samples, each call of this step was supplied with a numeric wildcard with a range of 0 to 99, which additionally served as suffix to the sample file received from each individual run. This wildcard was maintained in the filename of each subsequent randomization step, allowing to backtrack each randomized file to the original positive sample.

Each positive sample was then randomized by all three randomization algorithms 1,000 times utilizing the same wildcard principle as before. Additionally to the first wildcard representing the Identification (ID) of the positive sample, another wildcard with a range from 0 to 999 was implemented. Both wildcards were used to generate the filename of each shuffled sample, allowing to backtrack which positive sample was used for a specific randomized sample. This step results in a total of 100,000 randomized samples per randomizing algorithm.

As a last step each sample (100 positive samples and 300.000 negative samples) were analyzed utilizing all three structure finders. Each tool produced one summary file per sample, which were subsequently parsed by a custom python script resulting in one file per detection tool and sample type. The condensed results were afterwards analyzed using R.

For a schematic overview of the resulting pipeline see figure 3.

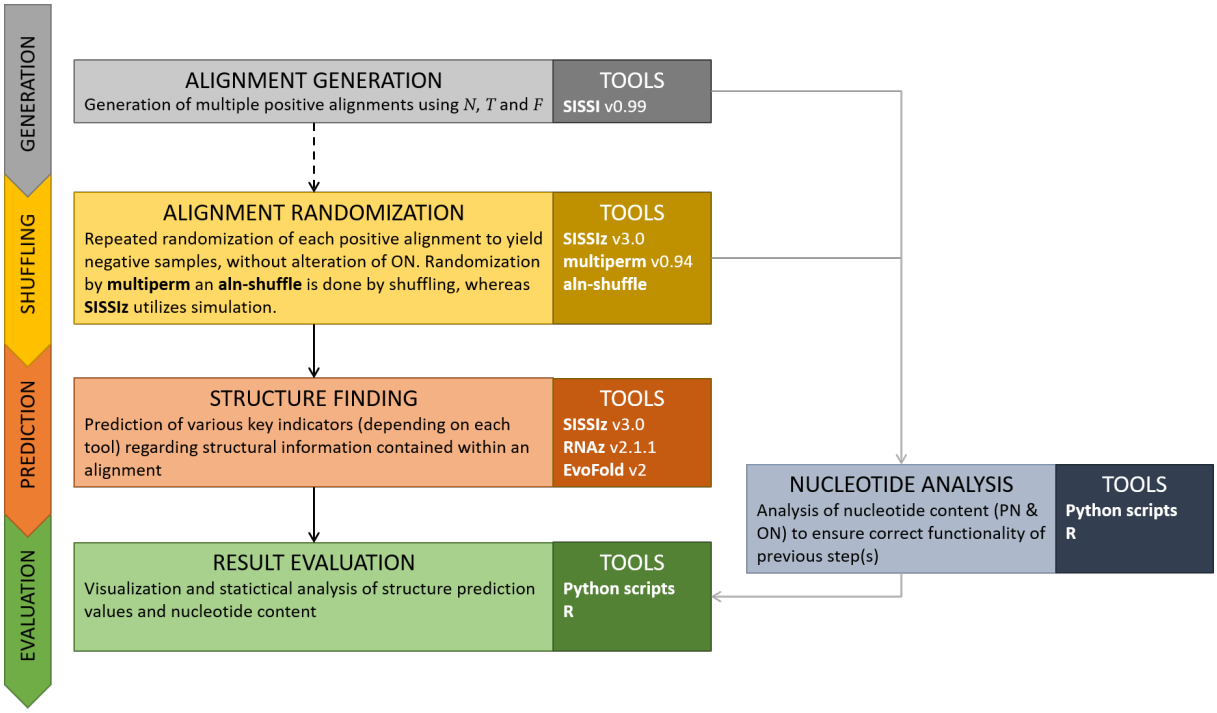

Figure 3: Workflow of the implemented pipeline. Alignment generation is done, by providing `SISSI` with N, T and F. This step is omitted if existing alignments (e.g. from RFAM database) are used, as indicated by the dashed arrow. Subsequently all positive alignments are repeatedly randomized by `SISSIz`, `multiperm` and `shuffle-aln`, followed by structure finding using `SISSIz`, `RNAz` and `EvoFold`. To facilitate easier file management for `R`, various custom Python scripts are employed, which aggregate key performance indications of all three structure finder into single files. Graphs are then created using `R`. Outside of this workflow an additional nucleotide analysis is conducted after alignment generation and randomization, respectively. This ensures correct creation of alignments by maintaining nucleotide frequencies and mono- and dinucleotide content.

Table 12: Nucleotide frequencies. These mono- and dinucleotide frequencies are used as input at each call of `SISSI` to produce one random positive sample of a fixed length. The individual values are supplied by work of Gesell. [14]

	Dinucleotide frequencies				Mononucleotide frequencies
	A	C	G	T	
A	0.000423	0.004228	0.012685	0.169133	0.422360
C	0.004228	0.000423	0.262156	0.000423	0.105590
G	0.012685	0.262156	0.000423	0.042283	0.236025
T	0.169133	0.000423	0.042283	0.016915	0.236025

2.2 Positive sample generation

Generation of positive samples using SISSI needs a neighbourhood system N, a phylogenetic tree T and mono- and dinucleotide frequencies F as input. For this purpose the neighbourhood system 'bsubtilis401.nei' was used. A visual representation of this structure can be seen in figure 4. Its length of 401 positions also predetermines the length of the resulting sequence. 'alina.tree' serves as the phylogenetic tree in this setup. The used mono- and dinucleotide sequences were extracted from Gesell's doctoral thesis [14], as can be seen in table 12.

In this step SISSI was used to generate alignments CLUSTAL W format. The implementation in snakemake included a forced waiting time of two seconds, as the random number generator of SISSI only takes the current time down to seconds as its seed. Furthermore, a forced exclusion of parallel executions had to be implemented. The code below was used for creation of all positive samples, by calling it 100 times including a wildcard ('pos_wc') with a range of 0 to 99.

```
rule generatePosSample:
    input:
        nh_file = <neighbourhood file>
        tree_file = <tree file>
        mono_freq = <mononucleotide frequencies>
        di_freq = <dinucleotide frequencies>
    output: "/samples/pos_sample_{pos_wc}.clu"
    resources: sissi = 1
    shell: "sissi0.99 -fs {mono_freq} -fd {di_freq} -nn {fn_file} \
           -l401 {tree_file} -oc -d > {output} && sleep 2"
```

2.3 Negative sample generation

Generation of negative samples utilizes the same wildcard principle as the generation of positive samples for all three randomization algorithms. A second wildcard ('neg_wc') identifying each sample was additionally employed. Both wildcards are stored in the filename, allowing the tracing of a negative sample to its original positive sample. Furthermore the name of the randomization algorithm was added, to allow the identification of the used tool. In summary, the filename of each individual negative sample was composed as follows:

negative_sample_{shuffling algorithm}_{pos_wc}_{neg_wc}

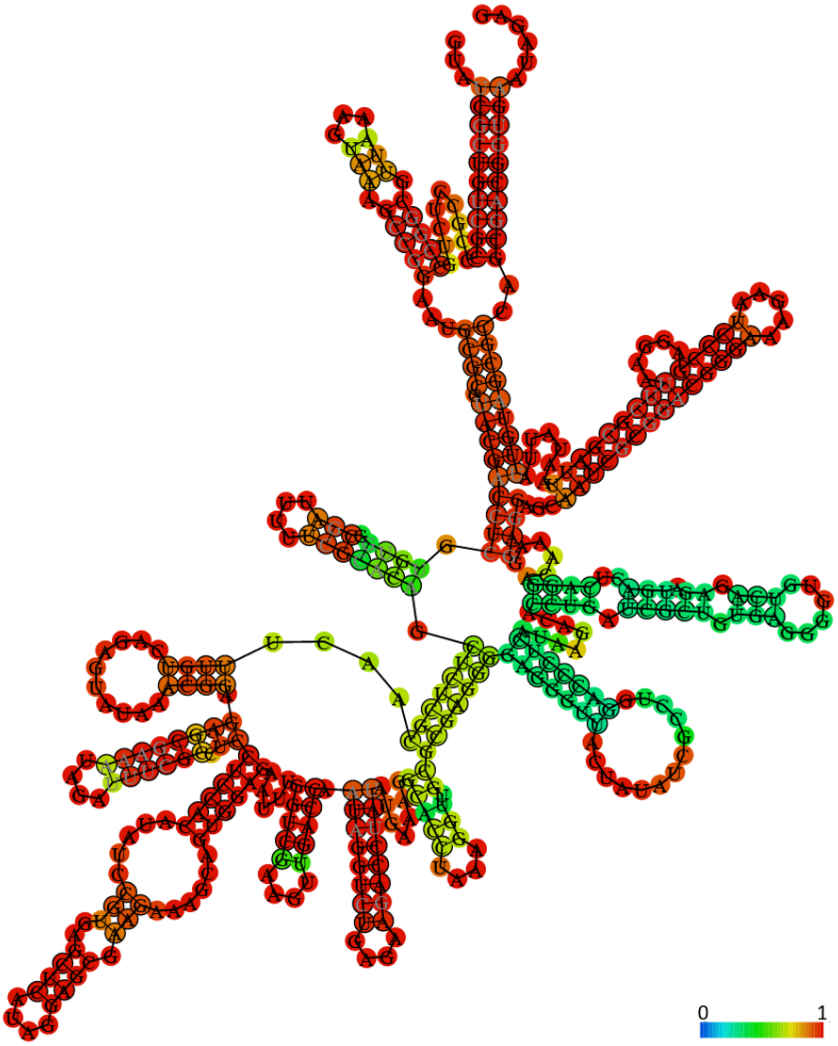

Figure 4: Visual representation of the consensus structure from a generated alignment of b.subtilis. The color of each base pair of a simulated positive alignment represents the base pair probability of the respective site.

2.3.1 Simulation by `SISSIz`

`SISSIz` offers the option to either use a mononucleotide or dinucleotide model to use during randomization. To use both options, two separate rules were implemented in `snakemake`, incorporating the responsible flag "-i" depending on the desired model. By default the dinucleotide model is used, which can be changed by the aforementioned flag. Also, the flag "-s" was used, to disable the folding during simulation of the supplied sequence, which drastically decreases the runtime per sample file. Following implementation in `snakemake` was used ("-i" was omitted depending on desired nucleotide model):

```
rule generateNegSampleSISSIz:
    input: "/samples/pos_sample_{pos_wc}"
    output "/samples/neg_sample_SISSIz_{pos_wc}_{neg_wc}"
    shell: "SISSIz -s -i {input} > {output}"
```

2.3.2 Shuffling by `shuffle-aln`

For the generation of negative samples using `shuffle-aln`, the script "shuffle-aln.pl" can be used. It represents the extracted shuffling algorithm from `shuffle-aln`, without the need to install the whole software. The script itself is easy to use, as no additional arguments have to be supplied for successful shuffling of input sequences. Following code was employed in the pipeline:

```
rule generateNegSampleShuffle-aln:
    input: "/samples/pos_sample_{pos_wc}"
    output "/samples/neg_sample_shuffle-aln_{pos_wc}_{neg_wc}"
    shell: "shuffle-aln.pl < {input} > {output}"
```

2.3.3 Shuffling by `multiperm`

Usage of `multiperm` required an additional step to retain the ability to backtrack each resulting output file to its original input file. As `multiperm` offers no possibility to name its output file, an additional step to rename its output file was needed. Each output file follows the convention 'perm_{iterator}_{input}.clu'. However, `multiperm` offers the ability to supply it with multiple input files, which consequently are numbered using the iterator mentioned before, but the way `snakemake` was set up in this thesis, multiple files

could not be supplied in a single step. In consequence an additional step was used to rename the negative sample to our liking.

Output files use MAF by default, which is changed to CLU format by usage of the "-w" flag. Following code was used to shuffle each positive sample and rename it:

```
rule generateNegSampleMultiperm:
    input: "/samples/pos_sample_{pos_wc}"
    output "/samples/neg_sample_multiperm_{pos_wc}_{neg_wc}"
    shell: "multiperm –w {input} && mv perm_001_{input}.clu {output}"
```

2.4 Analysis

The positive samples, as well as the negative samples, are investigated for their amount of detectable structure. In an optimal case, no structure can be detected by either, `SISSIz`, `RNAz` or `Evofold`.

An additional wildcard (’sample_wc’) containing the names of the randomization algorithms and whether or not it is a positive or negative sample, has to be incorporated, in order to use the same rule for analysis of the samples of all randomization algorithms. To be able to discern between the analysis results of each structure finder, an additional wildcard would be necessary if all files would be stored in the same directory. However, as a high number of wildcards can negatively impact the runtime of `snakemake`, an approach to store the files in disparate directories without this additional wildcard is advisable and was also conducted during this studies.

2.4.1 Analysis by `SISSIz`

`SISSIz` not only provides the capability of randomizing an sequence by simulation, but also analyses the sequence for detectable structure. Normally, it would be possible to merge the randomization and structure finding into one rule, but as additional negative samples from `shuffle-aln` and `multiperm` had to be analysed, it was more feasible to outsource the command for it to an additional rule, which can be utilized by means of the wildcards explained in section 2.3. Furthermore, the flag "–sci" is incorporated to additionally write the SCI-value to the usual output. Using this approach leads to following implementation in `snakemake`:

```
rule analyseWithSISSIz:
    input: "/samples/{sample_wc}_{pos_wc}_{neg_wc}"
    output: "/analysis_SISSIz/{sample_wc}_{pos_wc}_{neg_wc}"
    shell: "SISSIz --sci {input} >> {output}"
```

2.4.2 Analysis by `RNAz`

Usage of `RNAz` is very similar to `SISSIz`, as it also provides randomization capabilities. The flag "-n" does prevent this structure finder from randomizing its input prior to analysis. `RNAz` was implemented to the pipeline by following rule:

```
rule analyseWithSISSIz:
    input: "/samples/{sample_wc}_{pos_wc}_{neg_wc}"
    output: "/analysis_RNAz/{sample_wc}_{pos_wc}_{neg_wc}"
    shell: "RNAz -n {input} >> {output}"
```

2.4.3 Analysis by `EvoFold`

Usage of `EvoFold` requires to supply it with a tree, specific to the investigated sample, as well as the input sequence in the format of .ama file. The former is easily achieved by usage of `SISSIz`. However, as `SISSIz` always names the resulting tree file 'aln.tree', an additional step had to be incorporated to rename the file according to our liking. Further efforts had to be ensured, to prevent this step to run in parallel, as the resulting tree file could otherwise be inadvertently overwritten. The latter was accomplished by a custom Python script, written for the sole purpose of converting a .clu file to an .ama file. This script uses the 'AlignIO' package of the Biopython library. Note that the main rule - the analysis step - implements the flag "-f" to write the complete structure finding for each input element in addition to the sub-structures.

```
rule convertCLUtoAMA:
        input: "/samples/{sample_wc}_{pos_wc}_{neg_wc}.clu"
        output: "/samples/{sample_wc}_{pos_wc}_{neg_wc}.ama"
        shell: "{cluTOama_converter.py} -i {input} -o {output}"

rule buildTreeFromSample:
        input: "/samples/{sample_wc}_{pos_wc}_{neg_wc}.clu"
        output: "/samples/{sample_wc}_{pos_wc}_{neg_wc}.tree"
        resources: trees=1
        shell: "SISSIz -b -s {input} > /dev/null && mv aln.tree {output}"
```

```
rule analyseWithEvoFold:
    input:
        ama = "/samples/{sample_wc}_{pos_wc}_{neg_wc}.ama",
        tree = "/samples/{sample_wc}_{pos_wc}_{neg_wc}.tree"
    output: "/analysis_EvoFold/neg_sample_{shuffle_wc}_{pos_wc}_{neg_wc}"
    shell: "EvoFoldV2 -f {output} {input.ama} {input.tree}"
```

2.5 Result aggregation using R

Initial investigations regarding PN and ON are conducted to ensure that further examinations can be relied upon. If abnormalities in nucleotide counts would be found, an inspection of the pipeline setup would be due. The analysis of various different nucleotide contents are discussed in section 3.2.

All calculations are conducted using R in addition to custom Python scripts, which aggregates either mono- or dinucleotide content (PN or ON), as well as the key performance indicators of all used analysis tools (refer to section 1.5) in single files. The purpose of outsourcing the data aggregation from R code is to keep the R code as simple as possible, as it requires more complex code to read several different file format in R compared to Python code. Furthermore, it was simpler to write the required code for checking various attributes (e.g. completeness) in Python. In summary the Pythons scripts handled following tasks:

- Check number of files generated by pipeline
- Check completeness of files [1]
- Aggregate attributes in single files respective to attribute and source
- Check if aggregated number of observations corresponds to pipeline setup

In theory all deployed Pythons scripts, as well as the final investigation using R, could be incorporated to the pipeline easily to avoid having to execute them manually. However, this would have increased the complexity of the pipeline unnecessarily. As complexity of the pipeline was already becoming an issue for `snakemake` in regard to performance, it was decided that all steps after sample analysis (see figure 3) were conducted manually.

[1] If pipeline execution is halted abruptly, e.g. due to power loss, currently opened files were created, but not fully written. If the pipeline is continued, `snakemake` would view these files as completed, leaving incomplete data behind.

Once all desired values were aggregated, all further examinations were conducted using R. The first step was reading all aggregation files, format the data in order to be able to create graphs and conduct statistical tests upon it. Note, that no checks regarding completeness of data were done in R, as those were already considered in the custom Python scripts. Afterwards, the graphs were created using the package `ggplot2`, because of its simple handling and powerful customization options. As a last step following statistical tests were carried out, in order to draw conclusions of the generated data. The data was investigated in regards to its homogeneity of variance and normality. Obviously continuous, nominal data was observed, which can be classified as categorical (2 groups). Also the data was of quantitative type and both groups came from the same population. Because of its non parametric nature, all statistical tests were conducted using the Wilcoxon signed-rank test.

Besides result aggregation, R was also used to create custom phylogenetic trees, by usage of the libraries 'geiger' and 'phytools'. A custom R script then took the original 'alina.tree' file as input and converted it to a star-shaped phylogenetic tree with defineable branch lengths. These custom phylogenetic trees are hereinafter called 'star-phylogeny' for formating purposes in various graphs. The respective branch lenghts are abbreviated on figures by the acronym d.

2.6 Runtime benchmarking

For benchmarking and comparing the used tools a modified version of the already incorporated pipeline was used. In this modified version, only single rules were processed, to achieve runtime measurements of single tools. The generation, randomization and structure finding of samples was set to 1,000 iterations and the standard linux command `time` was added to obtain the runtime of each individual pipeline run. To automate the benchmarks and to minimize human error, a simple bash script was created:

```
#!/bin/bash

for i in {1..10}
do
    rm -rf files/
        rm -rf .snakemake/
        time snakemake --cores 62 -q > OUTlog$i.txt 2> ERRlog$i.txt
done
```

This script was then executed, with its STDERR and STDOUT piped to a separate log file in order to obtain the runtimes. Note, that for benchmarking only 62 core were used ("–cores 62"), to allow the remaining 2 cores (64 in total) to process the overhead produced from `snakemake`. The resulting runtimes were then multiplied by 62 to normalize the results, enabling the comparison to other server architectures.

Usage of `time` outputs three different time measurements:

- 'real' is real time - time from the moment of the call to the end of it. This includes all elapsed time including time slices used by other processes and time the process spends blocked (for example if it is waiting for I/O to complete).
- 'user' is the amount of CPU time spent in user-mode code (outside the kernel) within the process. This constitutes actual CPU timespent on exclusively this process. Other processes and time the process spends blocked do not count towards this figure.
- 'sys' is the amount of CPU time spent in the kernel within the process. It consists of CPU-time spent executing system calls within the kernel, as opposed to library code still running in user space. As with 'user', this is simply CPU time spent exclusively on that process.

For the purpose of this benchmarking, the figure 'user' and 'sys' is summarized, as this represents the total amount of CPU time spent within the process.

The main command (`snakemake`) was deliberately executed without the otherwise usual command `nice` to prevent the used cores from sharing their processing time with possible other users of the server, the pipeline was running on.

Note, that the pipeline was customized to allow parallel execution of multiple `multiperm` and `shuffle-aln` instances. In case of `multiperm`, this removed the traceability of files created by it, whereas `shuffle-aln` required at least one second pause between each execution in order to sample different seeds for its pseudorandom number generator. Both attributes (traceability and correct seed management) were disregarded in favor of comparability of benchmark results.

Also, the results are specific to the server (and its hardware) the benchmark was run on. The absolute values cannot be compared to any other machine, as runtimes are characteristic to their system. However, running all benchmarks on the same machine, with the same amount of cores, the tools can be compared to each other, provided that

no other, unrelated CPU intense processes were run during the tests.

Additionally the pipeline was modified to dump all created files in one single directory, which could then easily be deleted after each successful run. Also the hidden directory '.snakemake', which is used by `snakemake` to save logs and Directed Acyclic Graphs (DAGs) amongst other temporary files, was deleted to guarantee equal conditions for each runthrough. Furthermore, it is important to note, that the `time` command was only used for the actual pipeline, to not add the time needed for resetting the benchmarking environment.

3 Results

3.1 Runtime benchmarks

3.1.1 Sequence generation

Due to the lack of comparable tools incorporated into the pipeline, the runtime of SISSI could not be compared. Incorporating other alignment generation tools would have necessitated to run the entire subsequent pipeline twice, which was omitted because of time management reasons. Nevertheless, measuring the runtime of SISSI allowed us to calculate the relative time expenditure of the entire pipeline.

Taking 152,04 seconds on average to generate 1,000 alignments containing 78 sequences with a length of 401 base pairs, SISSI is well capable of producing a high number of files in reasonable time. This enables SISSI to be incorporated into any arbitrary pipeline, without unreasonably increasing its entire runtime.

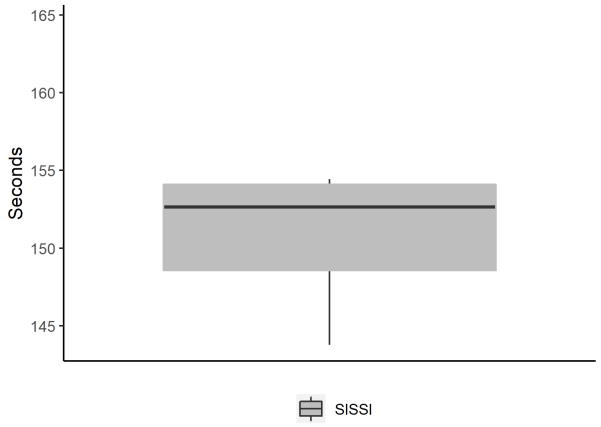

Figure 5: Runtime benchmark of sequence generation by SISSI. As no other tools were used for sequence generation, no comparison could be conducted. For benchmarking, the time needed to generate 1,000 sequences with a length of 401 base pairs was measured. SISSI took an average of 152,04 seconds out of 10 consecutive repeats.

3.1.2 Sequence randomization

Due to SISSIz's approach to randomization, which is actually closer to simulating new alignments by estimating the phylogenetic tree and counting the nucleotide frequencies of its input data, it has the highest runtime of all randomization algorithms, regardless if the mononucleotide or dinucleotide model is employed. Using its mononucleotide model, it needed on average 208.42 seconds, whereas the dinucleotide model needed on average 302.7 seconds.

multiperm is the fastest of all compared shuffling algorithms. It took 53.21 seconds on average, to shuffle 1.000 alignments. Its speed almost certainly can be attributed to its simple approach to shuffling. (see section 1.5.4).

The runtime of shuffle-aln is comparable to SISSIz's mononucleotide mode, taking 164.16 seconds on average and is therefore much slower than multiperm, although their shuffling performance is similar. This renders shuffle-aln as not being recommendable for large pipeline, considering multiperm being a better choice.

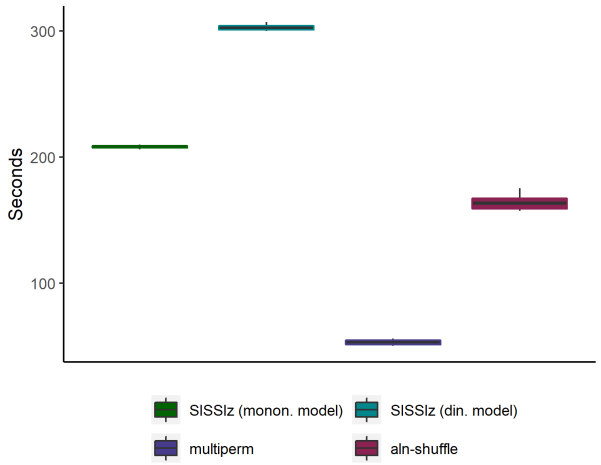

Figure 6: Runtime benchmarks of sequence shuffling by SISSIz, multiperm and shuffle-aln. For benchmarking, the time needed to shuffle the same positive sample 1,000 times by each shuffling tool was measured from 10 consecutive runs. On average SISSIz needed 208.42 seconds using its mononucleotide model and 302.70 seconds utilizing its dinucleotide model. multiperm and shuffle-aln took 53.21 and 164.16 seconds respectively.

3.1.3 Structure prediction

The runtimes of SISSIz (with and without SCI calculation) and RNAz are very similar, as seen in figure 7. Note, that only 1.000 alignments were analyzed during the benchmark. Analyzing a larger dataset, like in the main run of this thesis, their differences in runtimes are more obvious.

EvoFold is the fastest structure prediction tool out of the four compared algorithms. It took only 1,011.70 seconds to analyze 1.000 alignments. In contrast, SISSIz needs 1,641.61 seconds, when not calculating the SCI and 1,726.11 seconds when calculating it. RNAz is the slowest algorithm, taking 1,740.41 seconds on average.

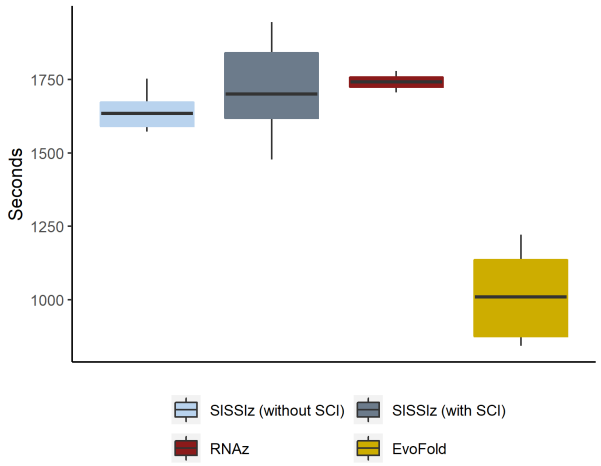

Figure 7: Runtime benchmark of structure prediction by SISSIz, RNAz and EvoFold. As likewise in the other benchmarks, 1,000 samples were analysed for its structural content in 10 consecutive runs. The time needed for these runs was then compared. SISSIz provides the ability to calculate the SCI in desired. Structure prediction without calculating SCI took SISSIz an average of 1,641.61 seconds, whereas runs with calculation of SCI took on average 1,726.11 seconds. RNAz and EvoFold on average needed 1,740.41 seconds and 1,011.70 seconds for the same task, respectively.

3.2 Nucleotide content

To ensure reliable data, it had to be secured, that ON was not altered through any of the steps of the pipeline. For dependable results in ncRNA detection, only the structure of a given sequence may be altered, without any change to the underlying ON. Hence, we investigated each step of the established pipeline in regards to the nucleotide content before and after said step (see section 3.2.1).

3.2.1 Nucleotide content of positive samples

Before setting up the entire pipeline, the PN and ON of the generated positive samples in section 2.2 had to be verified using a simplified pipeline, to ensure sequences according to the nucleotide frequencies defined in table 12, were generated. For this purpose 10,000 positive samples were generated, as described in section 2.2 and subsequently analyzed for their mono- and dinucleotide frequencies, also considering the neighbourhood system. To this end, a custom python script was created, which takes the neighbourhood system into account, by counting it towards the dinucleotide portion of PN, if a specific position in a sequence has a neighbour. Positions not having a neighbour according to the neighbourhood system were counted towards the mononucleotide portion of PN. Additionally to these 10,000 positive samples, another 10,000 positive samples with equally distributed mononucleotide frequencies (indicated as 'b.subtilis*' throughout this work) were generated and analyzed for their nucleotide counts, to visualize the impact of varying nucleotide frequencies.

As visible in graphs A and B of figure 8 all predefined nucleotide frequencies (blue points) are met by the generated positive samples by both the unaltered frequencies as well as equally distributed mononucleotide frequencies. This verifies the correct functionality of `SISSI`, to generate simulated sequences with any chosen frequencies.

The graphs C and D of figure 8 show differences in the ON. Most distinctly, the mononucleotide content counted over the entire alignment is different between $F_{\text{b.subtilis}}$ and $F_{\text{b.subtilis*}}$. This is attributed to the change of frequencies during sequence simulation. Although only frequencies for mononucleotides were altered, this had an profound effect on the entire ON. This effect was expected, as the ON was counted taking the whole sequence length into consideration and therefore does not indicate any problem with sequence simulation using `SISSI`. Theoretically this also proves the work of Gesell and von Haeseler. [15]

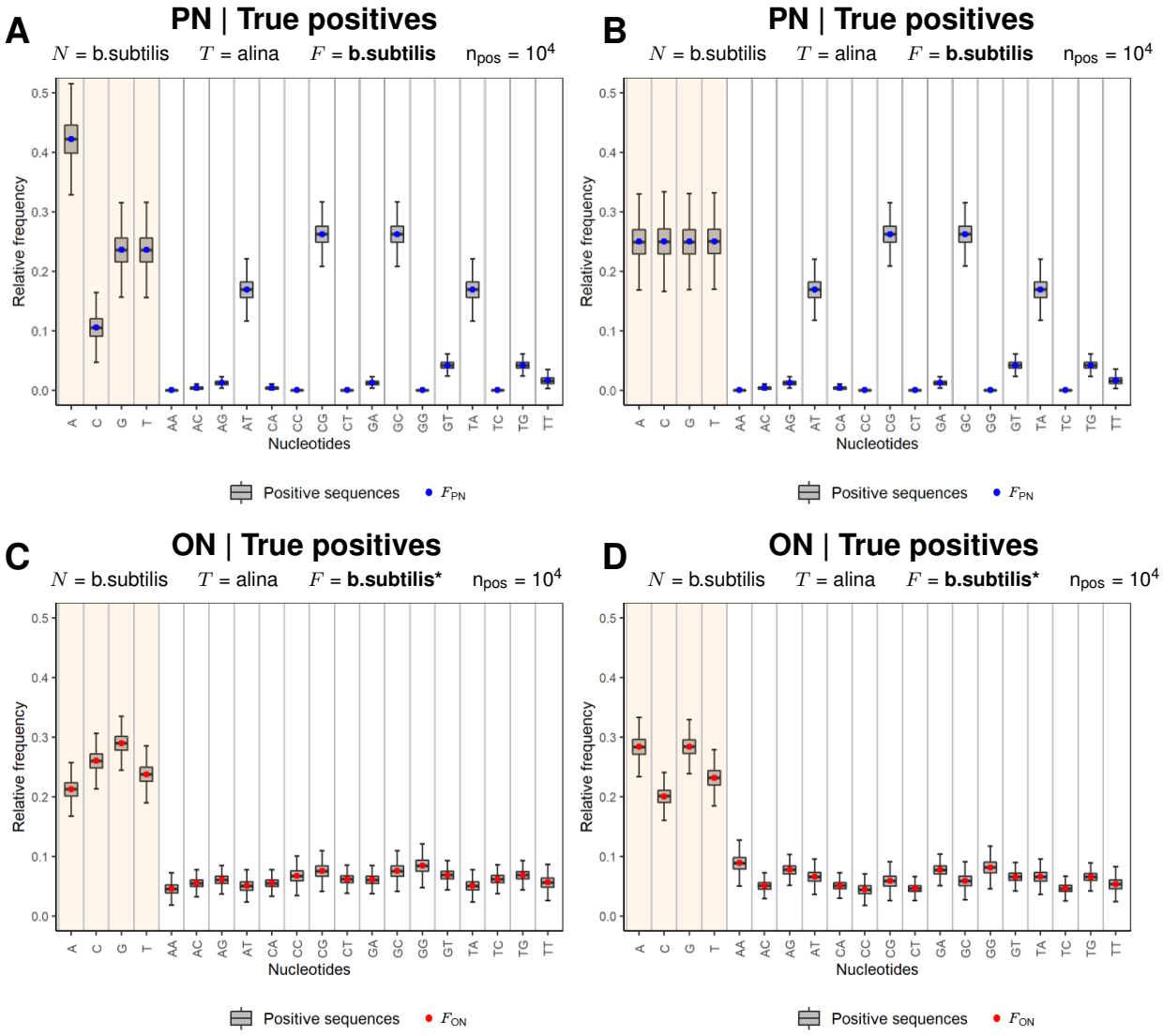

Figure 8: Nucleotide content of positive samples. A&B: This graph depicts the relative PN of 10,000 generated positive samples as described in section 2.2, taking N, T and F into account. Figure A&B visualizes data created using the exact frequencies from table 12, whereas data in figure C&D was created using custom frequencies of b.subtilis (b.subtilis*; mononucleotide frequencies were equally distributed, whereas dinucleotide frequencies were left original.). **C&D:** ON of the same sequences also shown in figure A and B, respectively.
The blue points depict the individual PN frequencies used for sequence generation. Red points represent ON frequencies.

3.2.2 Nucleotide content of randomized/negative samples

To ensure the successful alteration of PN within the positive samples, the mono- and dinucleotide content of all resulting negative samples were investigated. To this end, each positive sample was randomized 1.000 times by each randomization algorithm and subsequently analysed for its PN and ON. During this analysis, the PN is of special interest, as we aimed to alter the nucleotide counts in the neighbourhood as much as possible while randomizing. Additionally, the ON was investigated, to ensure no change thereof in any of the applied algorithms.

Figure 9 depicts an approach of the PN of shuffled alignments to equal distribution of mono- and dinucleotide content, resulting in the successful randomization of true positive sample alignments. Usage of the algorithm of `SISSIz` shows similar PN after simulation, regardless if the mononucleotide or dinucleotide model was used.

Although PN was successfully randomized, the absence of differences between sequences randomized using the mononucleotide and dinucleotide model became apparent. Theoretically a disparity between the two models should be visible in figure 9B. This phenomenon is explained if the sequence generation of the positive sequences is considered. `SISSI` uses preset frequencies for sequence generation (as presented in section 3.2.1 and visible as blue dots in each figure). However, the chosen frequencies for mononucleotide and dinucleotide content are dependent on each other in this case. The dinucleotide frequencies were calculated using the mononucleotide frequencies as a basis. Thus, no differences between mononucleotide and dinucleotide model can be observed, when comparing the results from these `SISSI`-generated positive samples.

To show, that the two different simulation models of `SISSIz` in fact produce different PN, an additional pipeline run was conducted. Instead of generated positive samples, the alignment 'rRNA', which is provided by the `SISSIz`-package, was used. To achieve comparable results, this single positive alignment was randomized 10,000 times by each randomization algorithm, respectively. Afterwards the same structure prediction tools and analysis methods as during the main pipeline run were conducted. The resulting boxplots (see figure 9D) show a difference of ON between the mononucleotide and dinucleotide model of `SISSIz`'s simulation algorithm, confirming our hypothesis explained above.

Additionally the nucleotide content of negative samples shuffled by `multiperm` and `shuffle-aln` were investigated (see figure 10). There is a clear difference between the simulation approach of `SISSIz` and the conventional shuffling of `multiperm` and

`shuffle-aln`. The individual PN of alignments shuffled by `shuffle-aln` are not nearly as approached to the calculated frequencies (displayed as red dots), as the other two randomization algorithms. Unfortunately the shuffling technique of `shuffle-aln` is only poorly described in its documentation. [1] Nonetheless, the difference to its two opponents can be explained by its conservative approach to shuffling, as described in section 1.5.3.

Comparing of the destruction of PN between `SISSIz` and `multiperm` results in rather similar findings. Both randomization algorithms manage to satisfactory alter PN, recognizable as the approach of PN towards ON frequencies (depicted as red dots). However, successfull alteration of PN does not result in less structural information, as later observations will show that `SISSIz` can clearly destroy structures more efficiently than `multiperm`.

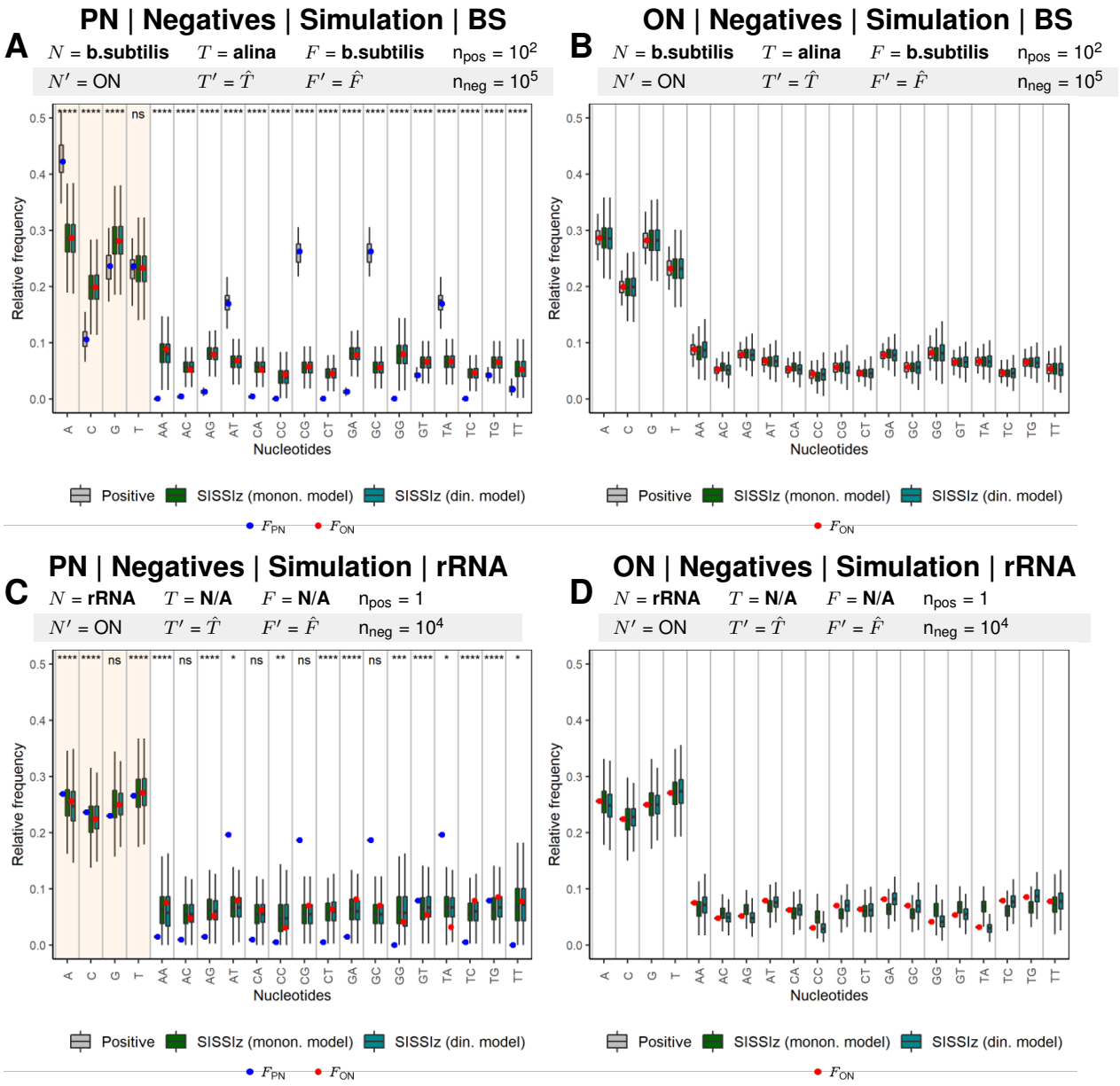

Figure 9: Nucleotide content for negative samples of the main run using simulation by `SISSIz`. **A**: Analysis of PN of negative samples randomized by `SISSIz` taking the N into account, shows successful alteration of PN of for both simulation models. **B**: Investigation of ON shows no modification, which demonstrates the effective randomization of positive samples, without any change of ON. **C&D**: Same analysis as A and B, but using a real life rRNA alignment as starting point. Comparing A to C, no differences between the mononucleotide model and the dinucleotide model can be seen. However, a distinction between the two models is recognizable in D, which cannot be observed in B. The blue points depict the respective nucleotide frequency of each PN used during positive sample generation, whereas the red dots show the respective relative, median ON of the positive samples.

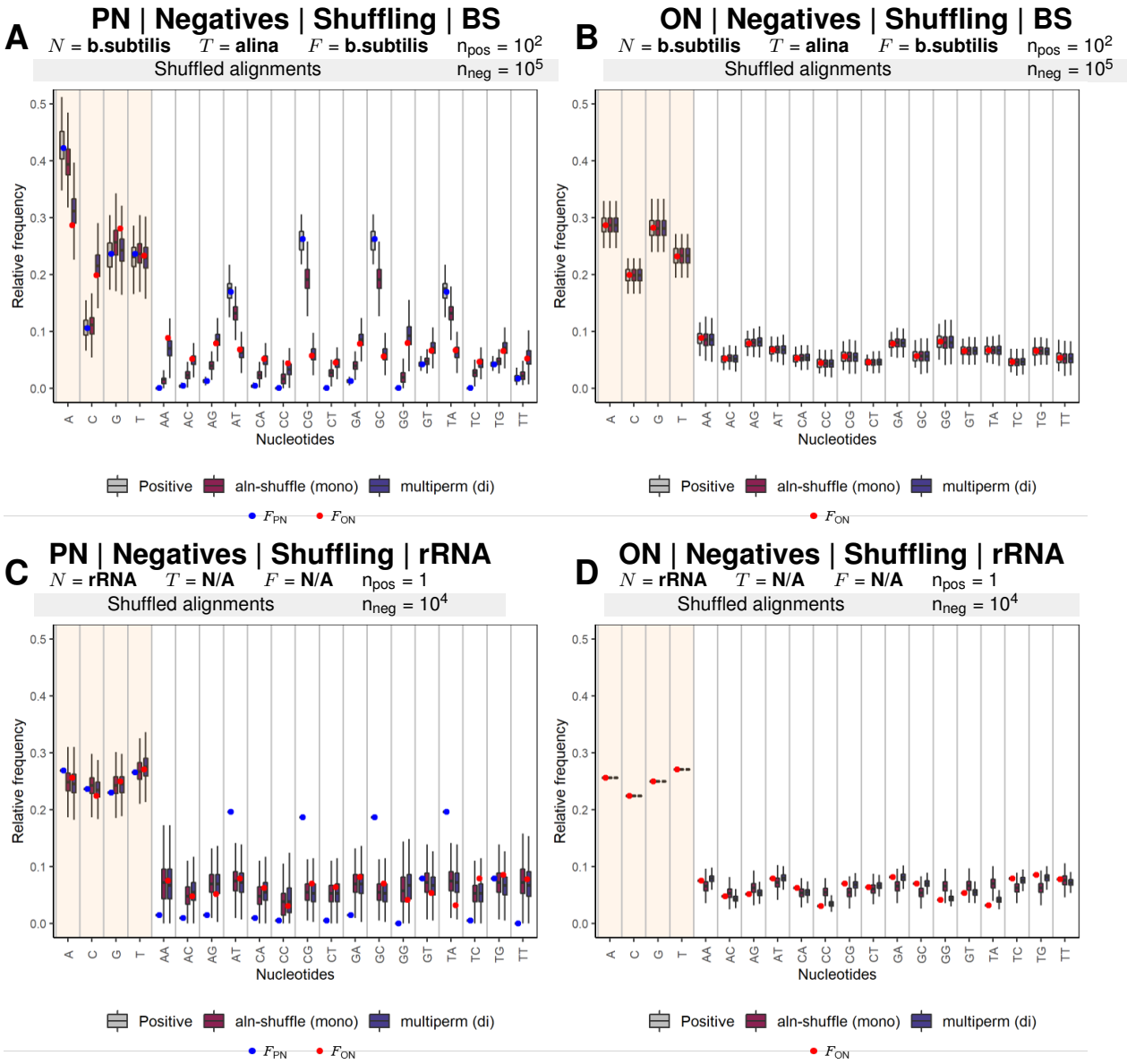

Figure 10: Comparison between shuffling of generated alignments and real life rRNA alignments. **A**: Investigating PN of negative samples shuffled by `shuffle-aln`, taking N into account, shows worse shuffling of nucleotides compared to `SISSIz` and `multiperm` (see figure 9A). B: Again, the analysis of ON in the shuffled samples of `shuffle-aln` and `multiperm` compared to the positive samples shows no difference, depicting functioning shuffling algorithms. **C&D**: Replication of A&B using a real life rRNA alignment as a starting point. Similar to the graph D of figure 9 a difference in ON between the two shuffling algorithms can be observed, due to the contrasting mononucleotide and dinucleotide shuffling models. The blue points depict the respective nucleotide frequencies of each PN used during positive sample generation, whereas the red dots show the respective relative, median ON of the positive samples.

3.3 Structure finding

The performances of all three randomization algorithms were evaluated using three structure finder, namely SISSIz, RNAz and EvoFold. All of them utilize different approaches to detect possible structures within sequences, as already discussed in section 1.5.

Looking at the key indicators for structured sequences in the main run, obvious differences are visible. Both SISSIz and RNAz provide a SCI as one of their structure indicators, which in fact is equally calculated. Therefore, the SCI of RNAz was not visualized outside the appendix, to prevent duplicate graphs. Note that lower values of SCI generally mean lesser structural information within a given sequence or alignment.

Differences in the randomization performance of all three algorithms started to become evident. Sequences simulated by SISSIz contain significantly less structural information according to all key indicators of all three structure prediction tools. This applies also to the z-score of SISSIz, p-value of RNAz and cykScore of EvoFold (see figures 11). The most structural information was destroyed using randomization of SISSIz, followed by multiperm and shuffle-aln being the least efficient randomization algorithm. Looking at the z-score calculated by SISSIz, sequences shuffled by multiperm and shuffle-aln even are below the high-confidence prediction thresholds of -4 (represented by the red line), indicating incomplete destruction of secondary structure.

3.3.1 Influence of tree topology and evolutionary time differences

To observe the influence of tree topologies and the evolutionary time d, represented as branch length in phylogenetic trees, a custom phylogenetic tree was constructed (see figure 12). The aim was to emphasize the differences in topologies and d, without influence of other factors. To this end, star-shaped phylogenetic trees with branch lengths of 0.1, 0.5, 1 and 2 were created, to remove compensatory effects and ancestral correlations of the phylogenetic tree upon sequence evolution.

The effect of d on sequence generation is shown by figure 13 and 14. For figure 14, the two extremes of d, namely 0.1 and 2.0, were deliberately chosen to exaggerate the differences. All figures regarding star-phylogenies with 0.5 and 1 branch length can be explored section 7.

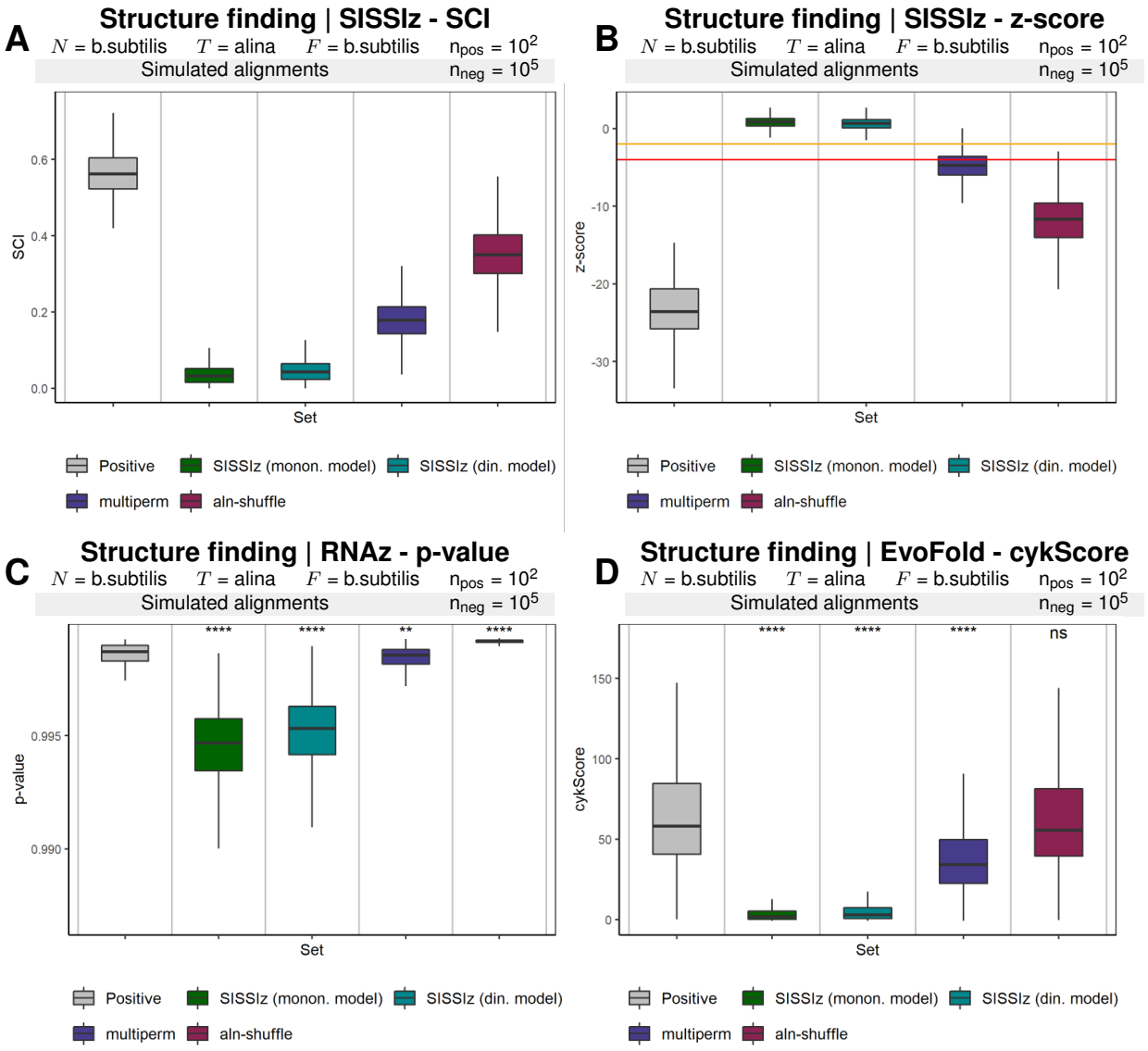

Figure 11: Structure prediction of the main run by all three structure finder. A: Predicted structure conservation index using `SISSIz`. A higher value indicates more conserved structures. **B**: Predicted z-score using `SISSIz`. A lower value indicates more conserved structures. Orange lines represent lower-confidence predictions(z-score $\leq$ -2), while red lines represent high-confidence predictions (z-score $\leq$ -4). A high number of false positives generated by the shuffling algorithms can be observed. **C**: Predicted p-value by `RNAz`. A higher p-values indicates that more structural information is embedded in the observed samples. **D**: Predicted cykScore by `EvoFold`. A higher cykScore hints towards more conserved structures withing the alignments.

Comparing the graphs A and B of figure 14 a large differences in SCI and z-score can be observed. Using a branch length of 0.1 in sequence simulation results in very few mutations within the sequences of an alignment, as not enough evolutionary time has passed for any mutations to occur. This is the reason why almost no difference of negative samples exists compared to positive samples. However, considering the z-score of graph B, all shuffling algorithms managed to increase this value above the lower and higher-confidence thresholds, indicating complete destruction of secondary structure. Increasing d to 2.0 allows enough evolutionary time to pass, that no ancestral correlations are embedded within the positive samples. This is can be observed as a higher difference between positive and negative samples. Figure 13 shows this effect in more detail. The differences of positive samples to negative samples increases with d. The highest change of disparity can be observed between d = 0.1 and d = 0.5, whereas almost no change in differences can be seen when d = 1.0 is changed to d = 2.0.

It must be highlighted, that almost no difference in randomizing performance of all randomizing algorithms can be seen in figure 13 and 14 (for star-shaped phylogenies). As a star-shaped phylogenetic tree was used during sequence simulation, not compensatory effects or ancestral correlations are simulated, due to the absence of nodes (see figure 12). Especially ancestral correlations within bushes of phylogenetic trees (highlighted in red in figure 12) are lost, due to the lack thereof. As `SISSIz` outperforms both other shuffling algorithms in terms of destruction of compensatory mutations and ancestral correlation within an alignment, no noteworthy difference can be stated, as it cannot play to its strengths. This allows `multiperm` and `shuffle-aln` to catch up to `SISSIz` in terms of randomization performance, where the latter managed to outperform the first two thus far. This phenomenon can be best seen in figure 14, when contrast between randomizing performance of 'alina' is compared to the randomizing performance of any run using a star-shaped phylogenetic tree.

An interesting effect can be observed by examining differences in SCI and z-scores over multiple different d. Looking at figure 13A, it becomes apparent, that more ancestral correlations can be destroyed if higher values of d are used during alignment generation, because more of them are deposited during said step. Nonetheless, the graph of figure 13B shows no differences between negative samples, regardless of d. Yet it becomes apparent, that more ancestral correlations are deposited within positive alignments, when d is increased, as the z-score of positive alignments decreases with higher values of d. The same observation is visible in graph C and D of figure 13.

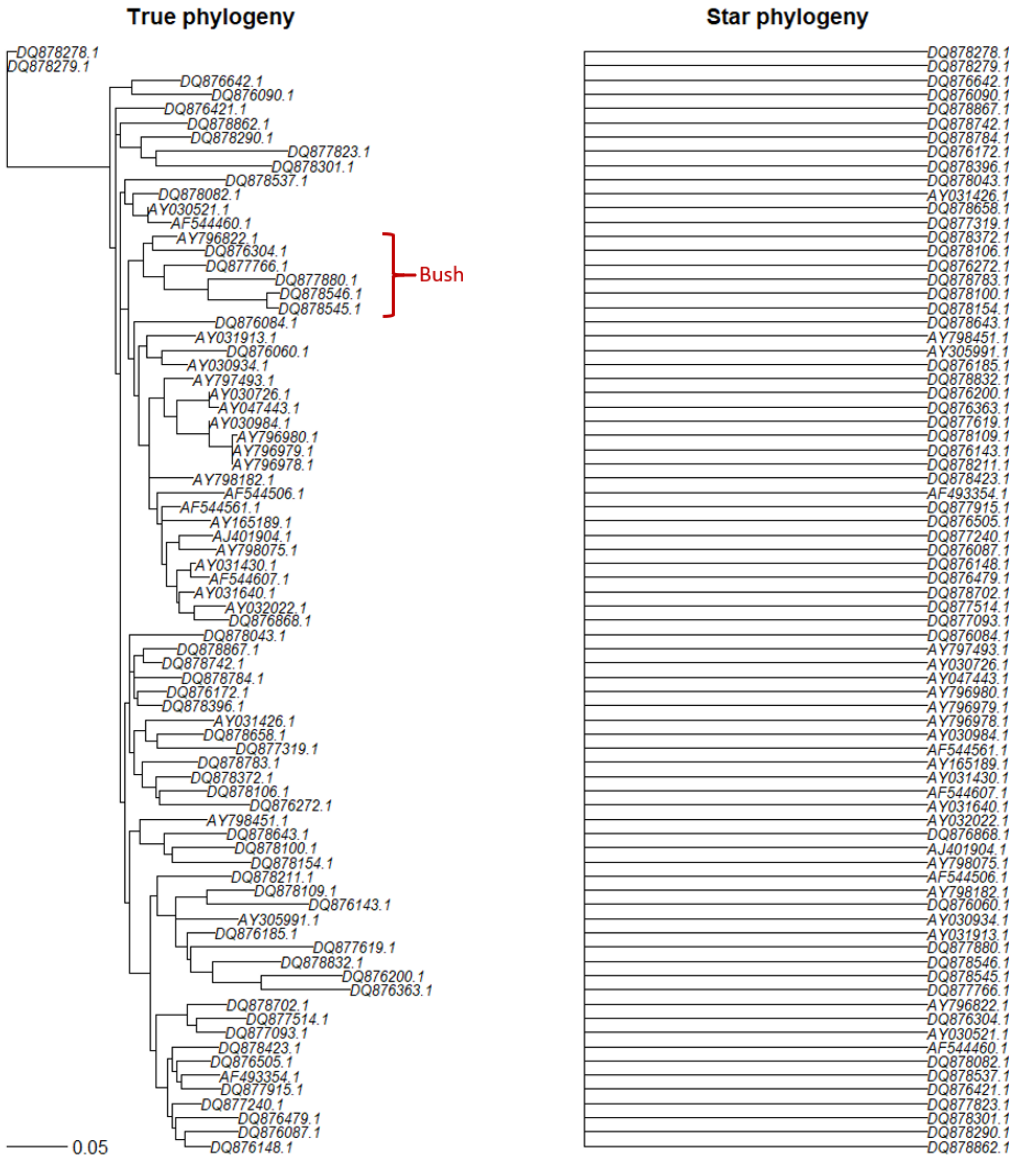

Figure 12: True phylogeny vs star shaped phylogeny. The branch length of the star shaped phylogenetic tree were adjusted, as explained in section 3.3.1. The branch length represents d. As an example of a bush, one has been highlighted in red. They can be recognized by the many temporally and relationally close species. These bushes contain particularly many ancestral correlations, which are consequently lost by the transformation into a star-shaped phylogenetic tree after enough evolutionary time. Note that d of the star phylogeny was adjusted to 0.1, 0.5, 1 or 2 during this thesis.

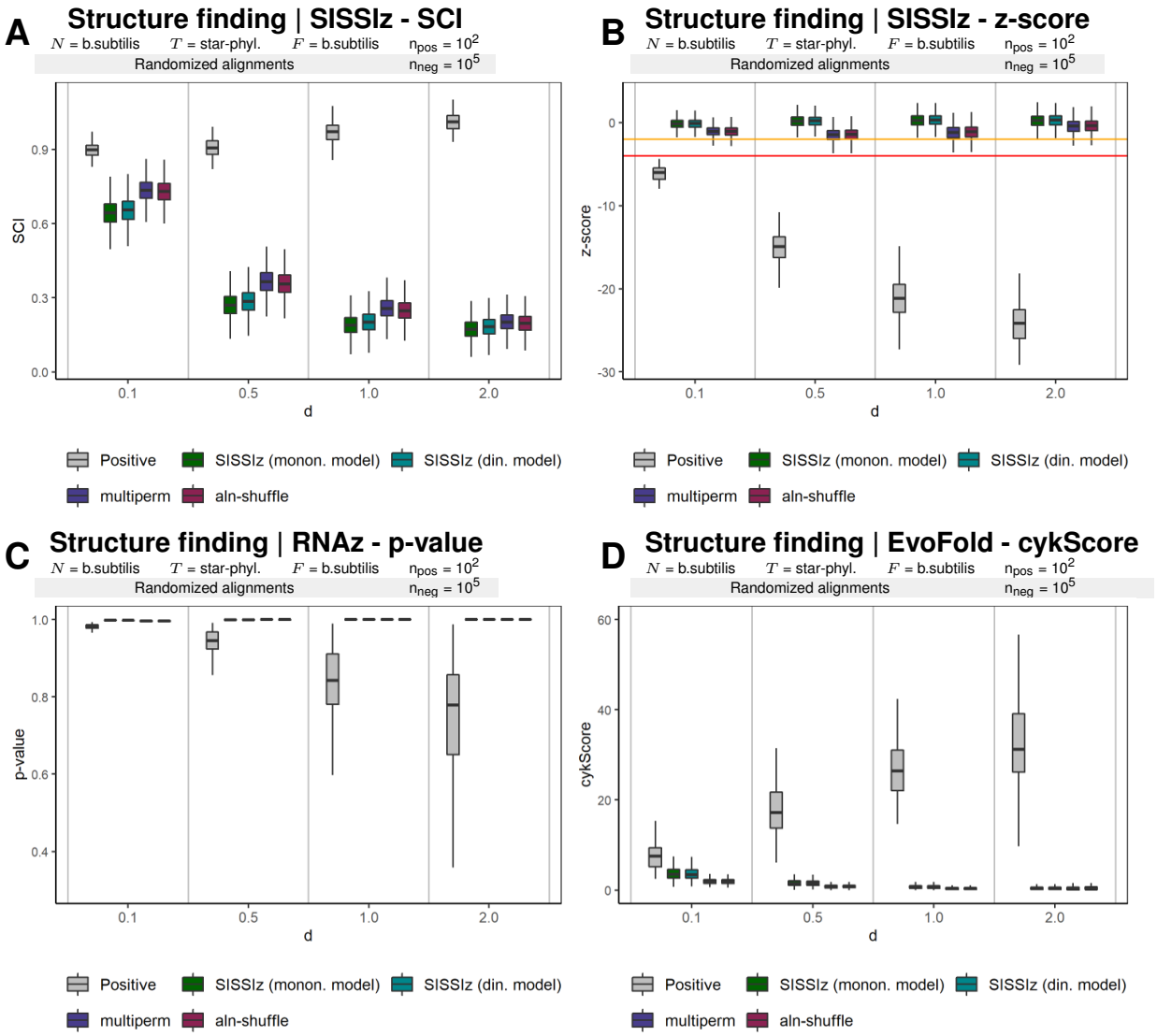

Figure 13: Structure finding of alignments generated using star-shape phylogenies with different evolutionary time d. **A**: This graph depicts increasing differences between positive and negative alignments if d is increased, as structure evolved more differently as more evolutionary time passed. If only a short time passes, more ancestral correlations remain between the sequences. The upward trend of positive samples is caused by the increase of mutual information content. **B**: Looking at the z-score no differences of negative samples can be seen between different d. This is cause by the change in statistical power, caused by the way how the z-score is calculated (see equation 21). **C**: Likewise to B, no change of negative samples can be observed, whereas the p-value for positive samples decreases when d is increased. **D**: Similar to the observed SCI in A, the cykScore shows opposing trends of positive and negative samples.

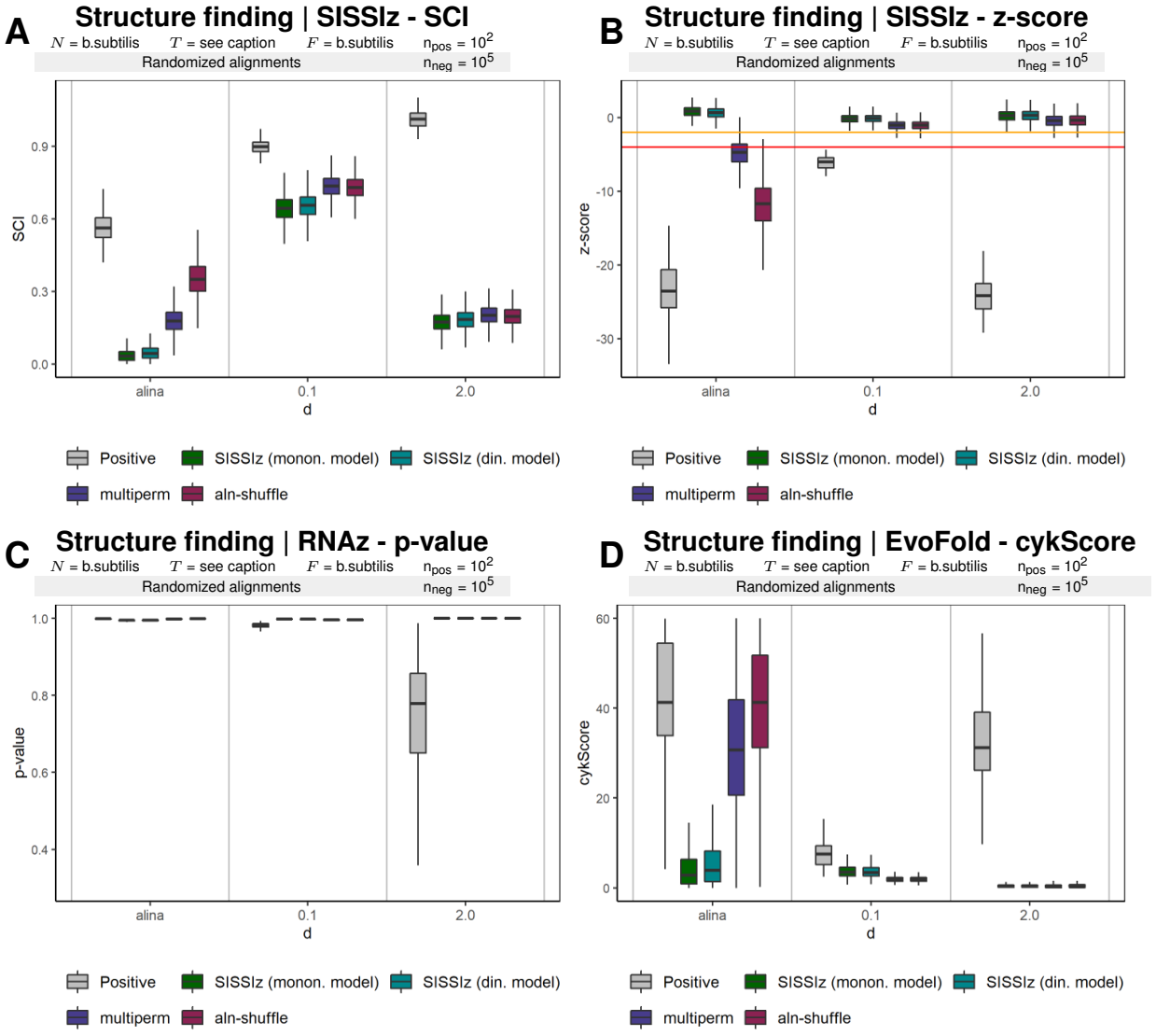

Figure 14: Structure prediction of sequences generated using a star-shaped phylogeny with d = 0.1 and d = 2.0. All figures include the results of the main pipeline run, which are labeled as 'alina'. This was done, to emphasize the differences between tree topologies (T_{alina} = b.subtilis, $T_{0.1}$ = star-phyl. d = 0.1, $T_{2.0}$ = star-phyl. d = 2.0). **A**: Clearly the high content in ancestral correlations in case of d = 0.1, results in a high false positive rate, regardless of randomization algorithm. Using a tree with either a more realistic topology or higher values of d, lowers the SCI to similar levels. **B**: Similar to figure 13, this graph shows the increase in statistical power, when d is increased. In contrast to the main pipeline run, no false positives could be detected using the z-score. **C**: The p-value of the main pipeline run and d = 0.1 is comparable. Due to the same reasons as in graph B, the p-value of the positive samples change dramastically. **D**: This graph is comparable to A. Note, that according to the cykScore, a high number of false positives are observed in the main pipeline run.

4 Summary

4.1 Testing SISSI's capabilities

SISSI can be used to generate synthetic alignment in conjunction with real life data. Extracting the neighbourhood system and phylogentic tree, as well as counting the PN from the original alignment of a RFAM dataset, artificial alignments can be generated.

In Gesell [14] a phylogenetic definition of structure is provided. A phylogenetic structure is defined as follows:

- The neighbourhood system N with $N = (N_k)_{k=1,2,...,l}$ for each site $k = 1, ..., l$ in a sequence, respectively each alignment site in an alignment **A**. In other words, the annotation of dependencies along sites.
- A substitution model constituting a collection of possible different substitution processes acting on the sequence.
- The phylogenetic tree T, presenting the evolutionary history. In this case both rooted and unrooted trees are allowed.

The sum of these three aspects can be seen as phylogenetic structure being an abstract object. It is defined by N, T and a substitution model.

This raises the question, how far this model explains a structure from the view of compensatory mutations. By comparing real life data to alignments generated under this model, a better assessment could be made.

4.2 SISSI evaluation

We present the difference of PN and ON in section 1.2.4, as distinct ways to count nucleotide frequencies, depending on whether N is included or disregarded during the count. Furthermore, those nucleotide counts are heavily tributary on the chosen nucleotide frequencies, such as those presented in 12. One has to consider if the chosen mononucleotide and dinucleotide frequencies are dependent on each other, as this influences the functionality of different simulation models, as discussed in section 3.2.2.

These dependencies have been shown in figure 8 of chapter 3.2.2. In this example, the mononucleotide portion of the same nucleotide frequencies were left original for the first pipeline run and then equally distributed during the second pipeline run. Note, that the dinucleotide portion was left original in both cases. In this case, mononucleotide and dinucleotide frequencies were dependent on each other, resulting in the alteration of dinucleotide counts of ON, although only mononucleotide frequencies were changed for alignment generation.

We successfully demonstrated, that `SISSI` is able to generate alignments which fully comply to the predetermined nucleotide frequencies. The frequencies have to be chosen carefully, as the characteristics of dependent versus independent frequencies can have a profound effect downwards the pipeline.

4.3 Simulation vs. shuffling

The aim of this thesis is to present the contrast between shuffling and simulating algorithms. Looking at figure 11 the diverging capabilities of both types of algorithms can be seen best. The data presented in this figure utilizes an unaltered phylogenetic tree named 'alina', and the frequencies and neighbourhood system counted from an alignment of b.subtilis. As such, the generated alignments contain compensatory mutations, which shuffling algorithms like `multiperm` and `shuffle-aln` cannot destroy completely. Contrary to shuffling algorithms, `SISSIz` uses `SISSI` simulation as randomization algorithm. It estimates a phylogenetic tree depending on its input data and simulates new alignments resulting in output data void of compensatory mutations, thus effectively destroying all structural information contained in its input data. Evidence of this can be seen in figure 11. It shows huge consequences for gene finding and consequently the false positive rate. All three structure finders detected lower structural content in samples randomized by simulation algorithms, than in samples randomized by shuffling algorithms. Especially the z-score shows a not to be neglected amount of false positive samples produced by shuffling algorithms. Summing up, shuffling algorithms are not always capable of destroying compensatory mutations or simulate the same correlation patterns independent of the functional structure, such as ancestral correlations.

4.4 Investigating tree topologies

Further investigations of compensatory mutations and ancestral correlations were conducted, by transforming the original 'alina' tree into a star shaped phylogenetic tree (see figure 12). By removing all nodes within the tree and adjusting all branches to the same length, different patterns of ancestral correlations are generated. The resulting positive samples generated with a star-shaped tree are void of any ancestral correlations after enough evolutionary time. Especially closely related species (unidentifiable by the formation of bushes inside the tree), would otherwise contain disproportionate amounts thereof.

By additionally using different branch lengths, the evolutionary time d can be adjusted, which affects the number of mutations within an alignment. Our testing shows, that simulation and shuffling algorithms perform similarly, due to the lack of compensatory mutations. Without these mutations, simulation algorithms cannot play their strength, allowing shuffling algorithms to keep up regarding randomization performance. These findings are shown in figure 13 and 14, both of which show similar destruction of structural information regardless of the type of randomization algorithm.

Although no difference in randomization performance can be observed, figure 14 visualizes different amounts of mutual information deposited in positive samples depending on d. This is expected as the amount of passed evolutionary time affects the number of mutations.

In a nutshell, the thesis clearly shows the benefit of background simulation programs versus shuffling algorithms for both mono- and dinucleotide content. Usage of shuffling algorithms can lead to a higher rate of false positives. In particular, if we have ancestral correlation in the tree. We assume that using simulations, ancestral correlations will be mimicked and result in a similar z-score. This has to be further investigated.

5 Outlook

5.1 Improving SISSI's framework

Gene prediction algorithms may be based on thermodynamic data, in order to more accurately predict conserved regions in alignments. Gesell [14] suggested a model, that takes thermodynamic energies into account, represented by stacking energies within a sequence. This would result in more life like alignments, as stacking energies are known to be the most stabilizing factors in a sequence structure. However, thermodynamic data is not yet implemented in SISSI's current version.

Another limitation is raised by only supporting DNA or RNA sequences. This limitation mainly arose due to the availability of substitution matrices only for those two alphabets. By including suitable matrices for codon, protein and binary alphabets, SISSI could be extended to also include the capability to simulate such sequences.

5.2 Utilizing the pipeline

Using data sets from the RFAM database was not in scope of this thesis. However, the pipeline proposed in this work can also utilize alignments, F, N and T found in those data sets.

It would be possible to further investigate SISSI's capabilities, by comparing the results of pipeline runs using the original RFAM data set versus generated positive samples. All data needed for SISSI can be easily obtained from each RFAM entry. Furthermore, it would be possible to adjust the pipeline to automatically extract needed data from an RFAM entry and running the original and the generated samples in parallel.

5.3 Optimizing the pipeline

The implemented pipeline was created to compare multiple different tools with each other, as well as to assess the potent framework of SISSI. Such an approach results in an unrealistic workflow for real life application of such tools. Realistically, one would choose an appropriate tool for the intended use, instead of incorporating multiple, vastly different tools, like we did in this thesis.

Thus, our pipeline produces a high amount of files, which was difficult for `snakemake` to handle. This became especially apparent at the beginning of this thesis. The overhead produces from `snakemake` required to thoroughly optimize the workflow, by simplification and merging similar steps into one single rule, in order to achieve acceptable runtimes. This helped to reduce the size of the resulting DAG `snakemake` had to deal with. Additionally, the entire process of accumulating the key performance indicators, counting PN and ON, as well as performing result evaluation and creating graphs, was excluded from the pipeline, since the runtime of the entire pipeline already suffered from the amount of created files. Due to the size of the DAG, it was decided to exclude these steps, instead performing them manually after completion of a pipeline run.

If `snakemake` gets updated regarding performance, the pipeline described in this thesis could be improved by fully automating each step from alignment generation to creating graphs.

5.4 Minor Points

During the beginning of this thesis, cumbersome aspects of the integrated tools arose. As discussed in 1.4.1 a PRNG relies on the ability to generate unique seeds. Unfortunately, `SISSI` and `shuffle-aln` implemented this aspect poorly by only using the current date and time to a resolution of seconds as seed. Obviously a highly optimized pipeline aims to start multiple instances within seconds, resulting in identical seeds. To circumvent this problem, a limitation to single instances, in conjunction with hard-coded waiting times (`sleep`) had to be included. This limitation of `SISSI` did not pose a problem for the run time of the entire pipeline, as a single positive alignment undergoes multiple other steps during the subsequent pipeline. The shuffling of `shuffle-aln` however would have needed to be run in parallel to guarantee an optimal workflow.

Another problem was the poor implementation of `multiperm`'s naming convention of output files. IT does not provide the option to select either the directory for storing output files, nor to define filenames. Instead both are (partly) predefined, resulting in the loss of back-traceability. To circumvent this an additional step to move and rename each output file was included, losing the ability to run `multiperm` in parallel. In retrospect, it would have been possible to outsource `multiperm`'s shuffling step into a separate rule and running it in its own directory. However, it is unclear if the ability to run this tool in parallel compensates the increase in pipeline complexity and thus its created DAG.

6 Glossary

`BASH` Bourne-again shell

`SISSI` Simulating Site-Specific Interaction

AMA Annotated Multiple Alignment

CLU Clustal

CPU Central Processing Unit

DAG Directed Acyclic Graph

DAGs Directed Acyclic Graphs

DNA Deoxyribonucleic acid

FDR False Discovery Rate

GPUs Graphics Processing Units

HRNGs Hardware Random Number Generators

ID Identification

MAF Mutation Annotation Format

MFE Minimum Free Energy

MI Mutual Information

MPI Mean Pairwise Identity

mRNA messenger RNA

ncRNA non coding RNA

ON Overlapping nucleotide content

phylo-SCFG Stochastic Context-Free Grammar

PN Pairwise nucleotide content

PRNG Pseudo Random Number Generator

PRNGs Pseudo Random Number Generators

RNA Ribonucleic acid

SCI Structure Conservation Index

STDIN Standard In

STDOUT Standard Out

References

[1] *aln-shuffle Manual*. https://www.tbi.univie.ac.at/newpapers/SUPPLEMENTS/04-04-001/shuffle-aln.html. Accessed: 2021-06-11.

[2] P. F. Arndt, C. B. Burge, and T. Hwa. "DNA sequence evolution with neighbor-dependent mutation". In: *J Comput Biol* 10.3-4 (2003). [DOI:10.1089/10665270360688039] [PubMed:12935330], pp. 313–322.

[3] S. H. Bernhart et al. "RNAalifold: improved consensus structure prediction for RNA alignments". In: *BMC Bioinformatics* 9 (Nov. 2008). [PubMed Central:PMC2621365] [DOI:10.1186/1471-2105-9-474] [PubMed:15665081], p. 474.

[4] E. Bindewald and B. A. Shapiro. "RNA secondary structure prediction from sequence alignments using a network of k-nearest neighbor classifiers". In: *RNA* 12.3 (Mar. 2006). [PubMed Central:PMC1383574] [DOI:10.1261/rna.2164906] [PubMed:10791023], pp. 342–352.

[5] D. K. Chiu and T. Kolodziejczak. "Inferring consensus structure from nucleic acid sequences". In: *Comput Appl Biosci* 7.3 (July 1991). [DOI:10.1093/bioinformatics/7.3.347] [PubMed:1913217], pp. 347–352.

[6] F. H. CRICK. "On protein synthesis". In: *Symp. Soc. Exp. Biol.* 12 (1958). [PubMed:13580867], pp. 138–163.

[7] Richard Durbin et al. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[8] L. Duret and N. Galtier. "The covariation between TpA deficiency, CpG deficiency, and G+C content of human isochores is due to a mathematical artifact". In: *Mol Biol Evol* 17.11 (Nov. 2000). [DOI:10.1093/oxfordjournals.molbev.a026261] [PubMed:11070050], pp. 1620–1625.

[9] S. R. Eddy. "Computational analysis of conserved RNA secondary structure in transcriptomes and genomes". In: *Annu Rev Biophys* 43 (2014). [PubMed Central:PMC5541781] [DOI:10.1146/annurev-biophys-051013-022950] [PubMed:19668382], pp. 433–456.

[10] S. R. Eddy and R. Durbin. "RNA sequence analysis using covariance models". In: *Nucleic Acids Res* 22.11 (June 1994). [PubMed Central:PMC308124] [DOI:10.1093/nar/22.11.2079] [PubMed:1608946], pp. 2079–2088.

[11] R. Ettrich. "Steger, G.: Bioinformatik. Methoden zur Vorhersage von RNA- und Proteinstrukturen." In: *Photosynthetica* 41.2 (2003), pp. 266–266. ISSN: 03003604.

[12] J. Felsenstein. "Evolutionary trees from DNA sequences: a maximum likelihood approach". In: *J Mol Evol* 17.6 (1981). [DOI:10.1007/BF01734359] [PubMed:117114], pp. 368–376.

[13] D. Futuyma. *Evolution*. Sinauer, 2013. ISBN: 9781605351155.

[14] T. Gesell. "A phylogenetic definition of structure". PhD thesis. University of Vienna, Aug. 2009.

[15] T. Gesell and A. von Haeseler. "In silico sequence evolution with site-specific interactions along phylogenetic trees". In: *Bioinformatics* 22.6 (Mar. 2006). [DOI:10.1093/bioinformatics/bti812] [PubMed:16332711], pp. 716–722.

[16] T. Gesell and A. von Haeseler. “In silico sequence evolution with site-specific interactions along phylogenetic trees”. In: *Bioinformatics* 22.6 (Mar. 2006). [DOI:10.1093/bioinformatics/bti812] [PubMed:16332711], pp. 716–722.

[17] T. Gesell and S. Washietl. “Dinucleotide controlled null models for comparative RNA gene prediction”. In: *BMC Bioinformatics* 9 (May 2008). [PubMed Central:PMC2453142] [DOI:10.1186/1471-2105-9-248] [PubMed:12079347], p. 248.

[18] Tanja Gesell and Arndt von Haeseler. *SISSI: Simulating Sequence Evolution withSite-Specific Interactions*. English. 16 pp.

[19] J. Gorodkin et al. “MatrixPlot: visualizing sequence constraints”. In: *Bioinformatics* 15.9 (Sept. 1999). [DOI:10.1093/bioinformatics/15.9.769] [PubMed:10498780], pp. 769–770.

[20] N. C. Grassly, J. Adachi, and A. Rambaut. “PSeq-Gen: an application for the Monte Carlo simulation of protein sequence evolution along phylogenetic trees”. In: *Comput Appl Biosci* 13.5 (Oct. 1997). [DOI:10.1093/bioinformatics/13.5.559] [PubMed:9367131], pp. 559–560.

[21] A. R. Gruber et al. “RNAz 2.0: improved noncoding RNA detection”. In: *Pac Symp Biocomput* (2010). [PubMed:19908359], pp. 69–79.

[22] A. R. Gruber et al. “Strategies for measuring evolutionary conservation of RNA secondary structures”. In: *BMC Bioinformatics* 9 (Feb. 2008), p. 122.

[23] R. R. Gutell et al. “Identifying constraints on the higher-order structure of RNA: continued development and application of comparative sequence analysis methods”. In: *Nucleic Acids Res* 20.21 (Nov. 1992). [PubMed Central:PMC334417] [DOI:10.1093/nar/20.21.5785] [PubMed:1689306], pp. 5785–5795.

[24] M. Hasegawa, H. Kishino, and T. Yano. “Dating of the human-ape splitting by a molecular clock of mitochondrial DNA”. In: *J Mol Evol* 22.2 (1985), pp. 160–174.

[25] I. L. Hofacker, M. Fekete, and P. F. Stadler. “Secondary structure prediction for aligned RNA sequences”. In: *J Mol Biol* 319.5 (June 2002). [DOI:10.1016/S0022-2836(02)00308-X] [PubMed:12079347], pp. 1059–1066.

[26] David Jones and UCL Unit. “Good practice in (pseudo) random number generation for bioinformatics applications”. In: *URL http://www. cs. ucl. ac. uk/staff/d. jones/GoodPracticeRNG. pdf* (Jan. 2010).

[27] THOMAS H. JUKES and CHARLES R. CANTOR. “CHAPTER 24 - Evolution of Protein Molecules”. In: *Mammalian Protein Metabolism*. Ed. by H.N. MUNRO. Academic Press, 1969, pp. 21–132. ISBN: 978-1-4832-3211-9.

[28] M. Kimura. “A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences”. In: *J Mol Evol* 16.2 (Dec. 1980). [DOI:10.1007/BF01731581] [PubMed:865622], pp. 111–120.

[29] Johannes Köster and Sven Rahmann. *Snakemake*. 2014.

[30] Pierre L’Ecuyer and Richard Simard. “TestU01: A C library for empirical testing of random number generators”. In: *ACM Trans. Math. Softw.* 33 (Jan. 2007).

[31] Z. Lu and H. Y. Chang. “The RNA Base-Pairing Problem and Base-Pairing Solutions”. In: *Cold Spring Harb Perspect Biol* 10.12 (Dec. 2018). [PubMed Central:PMC6280703] [DOI:10.1101/cshperspect.a034926] [PubMed:8455623].

[32] R. Lück, G. Steger, and D. Riesner. “Thermodynamic prediction of conserved secondary structure: application to the RRE element of HIV, the tRNA-like element of CMV and the mRNA of prion protein”. In: *J Mol Biol* 258.5 (May 1996). [DOI:10.1006/jmbi.1996.0289] [PubMed:8637012], pp. 813–826.

[33] D. H. Mathews et al. “Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure”. In: *J Mol Biol* 288.5 (May 1999). [DOI:10.1006/jmbi.1999.2700] [PubMed:10329189], pp. 911–940.

[34] S. V. Muse. “Evolutionary analyses of DNA sequences subject to constraints of secondary structure”. In: *Genetics* 139.3 (Mar. 1995). [PubMed Central:PMC1206468] [PubMed:8336541], pp. 1429–1439.

[35] J. R. Norris. *Markov Chains*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.

[36] R. Nussinov and A. B. Jacobson. “Fast algorithm for predicting the secondary structure of single-stranded RNA”. In: *Proc Natl Acad Sci U S A* 77.11 (Nov. 1980). [PubMed Central:PMC350273] [DOI:10.1073/pnas.77.11.6309] [PubMed:100768], pp. 6309–6313.

[37] Ruth Nussinov et al. “Algorithms for Loop Matching”. In: *Siam Journal on Applied Mathematics - SIAMAM* 35 (July 1978).

[38] Ruth Nussinov et al. “Algorithms for Loop Matchings”. In: *SIAM Journal on Applied Mathematics* 35.1 (1978), pp. 68–82.

[39] A. Rambaut and N. C. Grassly. “Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees”. In: *Comput Appl Biosci* 13.3 (June 1997). [DOI:10.1093/bioinformatics/13.3.235] [PubMed:9183526], pp. 235–238.

[40] Elisabeth Renée and Marie Tillier. “Maximum likelihood with multiparameter models of substitution”. In: *Journal of Molecular Evolution* 39 (2004), pp. 409–417.

[41] F. Rodriguez et al. “The general stochastic model of nucleotide substitution”. In: *J Theor Biol* 142.4 (Feb. 1990). [DOI:10.1016/s0022-5193(05)80104-3] [PubMed:2338834], pp. 485–501.

[42] J. Ruan, G. D. Stormo, and W. Zhang. “An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots”. In: *Bioinformatics* 20.1 (Jan. 2004). [DOI:10.1093/bioinformatics/btg373] [PubMed:14693809], pp. 58–66.

[43] A. Rzhetsky. “Estimating substitution rates in ribosomal RNA genes”. In: *Genetics* 141.2 (Oct. 1995). [PubMed Central:PMC1206772] [PubMed:6587395], pp. 771–783.

[44] N. J. Savill, D. C. Hoyle, and P. G. Higgs. “RNA sequence evolution with secondary structure constraints: comparison of substitution rate models using maximum-likelihood methods”. In: *Genetics* 157.1 (Jan. 2001). [PubMed Central:PMC1461489] [PubMed:8325490], pp. 399–411.

[45] M. Schöniger and A. von Haeseler. “A stochastic model for the evolution of autocorrelated DNA sequences”. In: *Mol Phylogenet Evol* 3.3 (Sept. 1994). [DOI:10.1006/mpev.1994.1026] [PubMed:7529616], pp. 240–247.

[46] M. Schöniger and A. von Haeseler. “Simulating efficiently the evolution of DNA sequences”. In: *Comput Appl Biosci* 11.1 (Feb. 1995). [DOI:10.1093/bioinformatics/11.1.111] [PubMed:7796269], pp. 111–115.

[47] C. E. Shannon. “A Mathematical Theory of Communication”. In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423.

[48] M. A. Smith et al. “Widespread purifying selection on RNA structure in mammals”. In: *Nucleic Acids Res* 41.17 (Sept. 2013). [PubMed Central:PMC3783177] [DOI:10.1093/nar/gkt596] [PubMed:12970751], pp. 8220–8236.

[49] J. Stoye, D. Evers, and F. Meyer. “Rose: generating sequence families”. In: *Bioinformatics* 14.2 (1998). [DOI:10.1093/bioinformatics/14.2.157] [PubMed:9545448], pp. 157–163.

[50] K. Tamura and M. Nei. “Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees”. In: *Mol Biol Evol* 10.3 (May 1993). [DOI:10.1093/oxfordjournals.molbev.a040023] [PubMed:8336541], pp. 512–526.

[51] E. Tillier and R. Collins. “High apparent rate of simultaneous compensatory base-pair substitutions in ribosomal RNA.” In: *Genetics* 148 4 (1998), pp. 1993–2002.

[52] E. Tillier and R. Collins. “Neighbor Joining and Maximum Likelihood with RNA Sequences: Addressing the Interdependence of Sites”. In: *Molecular Biology and Evolution* 12 (1995), pp. 7–7.

[53] Wai Tsang and George Marsaglia. “Some Difficult-to-Pass Tests of Randomness”. In: *Journal of Statistical Software* 07 (Jan. 2002).

[54] P. Tufféry. “CS-PSeq-Gen: simulating the evolution of protein sequence under constraints”. In: *Bioinformatics* 18.7 (July 2002). [DOI:10.1093/bioinformatics/18.7.1015] [PubMed:12117802], pp. 1015–1016.

[55] D. H. Turner, N. Sugimoto, and S. M. Freier. “RNA structure prediction”. In: *Annu Rev Biophys Biophys Chem* 17 (1988). [DOI:10.1146/annurev.bb.17.060188.001123] [PubMed:2456074], pp. 167–192.

[56] Stefan Washietl. *RNAz 2.1*. English. Version 2.1. 61 pp.

[57] S. Will, M. Yu, and B. Berger. “Structure-based whole-genome realignment reveals many novel noncoding RNAs”. In: *Genome Res* 23.6 (June 2013). [PubMed Central:PMC3668356] [DOI:10.1101/gr.137091.111] [PubMed:15304649], pp. 1018–1027.

[58] A. Wilm, K. Linnenbrink, and G. Steger. “ConStruct: Improved construction of RNA consensus structures”. In: *BMC Bioinformatics* 9 (Apr. 2008). [PubMed Central:PMC2408607] [DOI:10.1186/1471-2105-9-219] [PubMed:16368769], p. 219.

[59] T. Xia et al. “Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs”. In: *Biochemistry* 37.42 (Oct. 1998). [DOI:10.1021/bi9809425] [PubMed:9778347], pp. 14719–14735.

[60] Z. Yang. “PAML: a program package for phylogenetic analysis by maximum likelihood”. In: *Comput Appl Biosci* 13.5 (Oct. 1997). [DOI:10.1093/bioinformatics/13.5.555] [PubMed:9367129], pp. 555–556.

[61] Michael Zuker and David Sankoff. “RNA secondary structures and their prediction”. In: *Bulletin of Mathematical Biology* 46.4 (July 1984), pp. 591–621. ISSN: 1522-9602.

7 Appendix

7.1 SISSI references

Table 13: List of references to SISSI paper [16]

Author	Title	Year	Month	Topic	Summary
Gouveia-Oliveira R., Pedersen AG.	Finding coevolving amino acid residues using row and column weighting of mutual information and multi-dimensional amino acid representation	2007	10	Coevolution of amino acid residues	Presents six new methods for detecting coevolving residues. Also suggests measures that are variant of Mutual Information and measures that use a multidimensional representation of each residue in order to capture the physicochemical similarities between amino acids.
Gesell T., Washietl S.	Dinucleotide controlled null models for comparative RNA gene prediction	2008	5	Alignment simulation	Introduction of SISSIz, which simulates multiple alignments of a given average dinucleotide content, meeting additional requirements of an accurate null model, achieving randomized alignments that are on average of the same sequence diversity and also preserve local conservation and gap patterns.
Kelle A., Förster F., Müller T., Dandekar T., Schultz J., Wolf M.	Including RNA secondary structures improves accuracy and robustness in reconstruction of phylogenetic trees	2010	1	Simulation of phylogenetic trees	Paper shows, that secondary structures of ribosomal genes improve the quality of phylogenetic reconstructions, by investigating the accuracy and robustness of phylogenetics with individual secondary structures by simulation experiments for artificial tree topologies.
Seemann SE., Richter AS., Gesell T., Backofen R., Gorodkin J.	PETcofold: predicting conserved interactions and structures of two multiple alignments of RNA sequences	2011	1	Prediction of RNA-RNA interactions	Presents a method, PETcofold, that can take covariance information in intra-molecular and intermolecular base pairs into account to predict interactions and secondary structures of two multiple alignments of RNA sequences.
Nebel ME., Scheid A., Weinberg F.	Random generation of RNA secondary structures according to native distributions	2011	10	Random generation method for secondary structures	Presents a general framework for deriving algorithms for the non-uniform random generation of combinatorial objects. This framework is then used to derive an algorithm for the generation of RNA secondary structures of a given fixed size.

Table 13 continued from previous page

Author	Title	Year	Month	Topic	Summary
Thi Nguyen MA., Gesel T., von Haeseler A.	ImOSM: intermittent evolution and robustness of phylogenetic methods	2012	2	Sequence evolution simulation	Introduces ImOSM, a tool to imbed intermittent evolution as model violation into an alignment. Intermittent evolution refers to extra substitutions occurring randomly on branches of a tree, thus cchanign alignment site patterns.
Dalquen DA., Anisimova M., Gonnet GH., Dessimoz C.	ALF—a simulation framework for genome evolution	2012	4	Simulation of genome evolution	Postulates that current simulation packages tend to focus either on gene-level aspects of genome evolution such as character substitutions and indels or on genome-level aspects such as genome rearrangements and speciation events. Introduces Artificial Life Framework (ALF) aimed at simulating the entire range of evolutionary forces.
Arenas M.	Simulation of molecular data under diverse evolutionary scenarios	2012	5	Simulation of molecular evolution	Brief background on simulation approaches and description of some of the most important simulators developed until 2012. Also shows several practical examples for simulating particular scenarios.
Ragan C., Mowry BJ., Bauer DC.	Hybridization-based reconstruction of small non-coding RNA transcripts from deep sequencing data	2012	9	ncRNA detection	Presents NorahDesk, an unbiased and universally applicable method for small ncRNAs detection rom RNA sequencing data. It utilizes the coverage-distribution of small RNA sequence data as well as thermodynamic assessments of secondary structure.
Smith MA., Gesell T., Stadler PF., Mattick JS.	Widespread purifying selection on RNA structure in mammals	2013	9	RNA structure prediction	Presents a novel benchmarking pipeline aimed at calibrating the precision of genome-wide scans for consensus RNA structure prediction. Furthermore, shows efficiency through identifying >4 mio. evolutionarily constrained RNA structures.
Caballero J., Smith AF., Hood L., Glusman G.	Realistic artificial DNA sequences as negative controls for computational genomics	2014	7	Generation of artificial (negative) sequences	Postulates that shuffling real sequences underestimats the false-positive rates when used as negative samples. Introduces method to generate artificial sequences that are modelled after real intergenic sequences in terms of composition, complexity and interspersed repeat content.

Table 13 continued from previous page

Author	Title	Year	Month	Topic	Summary
Eddy SR.	Computational analysis of conserved RNA secondary structure in transcriptomes and genomes	2014	-	Improvements for detection of conserved RNA secondary structure	Proposes approaches for incorporating structure probing data into computational predictions of RNA secondary structure. (SHAPE data)
Bayegan AH., Garcia-Martin JA., Clote P.	New tools to analyze overlapping coding regions	2016	12	RNA sequence analysis tool	Introduces a new tool, RNAsampleCDS, designed to compute the number of RNA sequences that code two (or more) peptides in overlapping reading frames.
Levy Karin E., Shkedy D., Ashkenazy H., Cartwright RA., Pupko T.	Inferring Rates and Length-Distributions of Indels Using Approximate Bayesian Computation	2017	5	Sequence simulation	Same as paper above, but a standalone application instead of a web server tool.
Ashkenazy H., Levy Karin E., Mertens Z., Cartwright RA., Pupko T.	SpartaABC: a web server to simulate sequences with indel parameters inferred using an approximate Bayesian computation algorithm	2017	7	Sequence simulation	Solving the problem of users in simulation studies of not knowing which insertion and deletion (indel) parameters to choose. SpartaABC = web server tool to infer indel parameters from sequence input data.
Lu Z., Carter AC., Chang HY.	Mechanistic insights in X-chromosome inactivation	2017	11	X-chromosome inactivation (XCI) via XIST	Summary on the specific long non-coding RNA (lncRNA) XIST and its associated proteins in their function of X-chromosome inactivation.
Lu Z., Chang HY.	The RNA Base-Pairing Problem and Base-Pairing Solutions	2018	12	RNA structure analysis	Comparison of commonly used methods in structure determination. Brief historical account of psoralen crosslinking studies. Highlight of the important features of crosslinking methods in RNA structure determination.
Lu Z., Gong J., Zhang QC.	PARIS: Psoralen Analysis of RNA Interactions and Structures with High Throughput and Resolution	2018	-	RNA structure analysis	Description of a new method to determine transcriptome-wide base pairing interactions. PARIS combines in vivo cross-linking, 2D gel purification, proximity ligation and high-throughput sequencing to determine RNA structurome and interactome in living cells.
Lu Z., Zhang QC, Lee B., Flynn RA., Smith MA., Robinson JT., Davidovich C., Gooding AR., Goodrich KJ., Mattick JS., Mesirov JP., Cech TR., Chang HY.	RNA Duplex Map in Living Cells Reveals Higher-Order Transcriptome Structure	2019	5	RNA structure analysis	Follow up paper on the paper “PARIS: Psoralen Analysis of RNA Interactions . . . ”. PARIS analysis in three human and mouse cell types revealed frequent long-range structures of higher- order architectures and RNA-RNA interaction in trans across the transcriptome.

7.2 SISSIz references

Table 14: List of references to SISSIz paper [17]

Author	Title	Year	Month	Topic	Summary
Varadarajan A, Bradley RK, Holmes IH.	Tools for simulating evolution of aligned genomic regions with integrated parameter estimation	2008	10	Sequence simulation	Introduces three programs: GSIMULATOR, SIMGRAM and SIMGENOME. Each offers algorithm for parameter measurements and reconstruction of ancestral sequence.
Bernhart SH, Hofacker IL, Will S, Gruber AR, Stadler PF.	RNAalifold: improved consensus structure prediction for RNA alignments	2008	11	RNA alignment structure prediction	Shows that the accuracy of RNAalifold predictions can be improved substantially by introducing a different, more rational handling of alignment gaps and by replacing the rather simplistic model of covariance scoring with more sophisticated RIBOSUM-like scoring matrices.
Bradley RK, Uzilov AV, Skinner ME, Bendaña YR, Barquist L, Holmes I.	Evolutionary modeling and prediction of non-coding RNAs in Drosophila	2009	8	ncRNA gene prediction	Performed benchmarks of phylogenetic grammar-based ncRNA gene prediction, experimenting with eight different models of structural evolution and two different programs for genome alignment.
Gorodkin J, Hofacker IL, Torarinsson E, Yao Z, Havgaard JH, Ruzzo WL.	De novo prediction of structured RNAs from genomic sequences	2010	1	RNA structure prediction	States that comparative genomics is a powerful tool predict structured RNAs from genomic sequences and is arguably preferable to any high-throughput experimental technology then available, because evolutionary conservation highlights functionally important regions.
Zimmermann B, Gesell T, Chen D, Lorenz C, Schroeder R.	Monitoring genomic sequences during SELEX using high-throughput sequencing: neutral SELEX	2010	2	Genomic aptamer identification	Indicates that positive selection in SELEX acts independently of the neutral selective requirements imposed on the sequences. This paper concludes that Genomic SELEX, when combined with high-throughput sequencing of positively and neutrally selected pools, as well as the genomic library, is a powerful method to identify genomic aptamers
Nygaard S, Braunstein A, Malsen G, Van Dongen S, Gardner PP, Krogh A, Otto TD, Pain A, Berriman M, McAuliffe J, Dermitzakis ET, Jeffares DC.	Long- and short-term selective forces on malaria parasite genomes	2010	9	Genome analysis	Uses evolutionary methods to describe selective processes in both the coding and non-coding regions of plasmodium parasites and a malaria control. Shows that protein-coding, intergenic and intronic regions are all subject to purifying selection.

Table 14 continued from previous page

Author	Title	Year	Month	Topic	Summary
Saito Y, Sato K, Sakakibara Y.	Robust and accurate prediction of noncoding RNAs from aligned sequences	2010	10	ncRNA prediction	Describes a new method, called Profile BPLA kernel, which predicts ncRNAs from alignment data in combination with support vector machines (SVMs). Profile BPLA kernel is an extension fo base-pairing profile local alignment (BPLA) kernel which was previously developed for the prediction from single sequences.
Washietl S, Findeiss S, Müller SA, Kalkhof S, von Bergen M, Hofacker IL, Stadler PF, Goldman N.	RNAcode: robust discrimination of coding and noncoding regions in comparative sequence data	2011	4	Coding region detection	Presents RNAcode, a program to detect coding regions in multiple sequence alignments that is optimized for emerging applications not covered by current protein gene-finding software.
Okada Y, Saito Y, Sato K, Sakakibara Y.	Improved measurements of RNA structure conservation with generalized centroid estimators	2011	8	RNA structure conservation	Proposes improved measurements based on SCI (SCI) and BPD (base pair distance), applying generalized centroid estimatiors to incorporate the robustness against low quality multiple alignments.
Mercer TR, Neph S, Dinger ME, Crawford J, Smith MA, Shearwood AM, Haugen E, Bracken CP, Rackham O, Stamatoyannopoulos JA, Filipovska A, Mattick JS.	The human mitochondrial transcriptome	2011	8	Transcriptome analysis	Provides a comprehensive analysis of the human mitochondrial transcriptome across multiple cell lines and tissues. Using directional deep sequencing and parallel analysis of RNA ends, this paper demonstrates wide variation in mitochondrial transcript abundance and precisely resolve transcript processing and maturation events.
Gorodkin J, Hofacker IL.	From structure prediction to genomic screens for novel non-coding RNAs	2011	8	Review on structure prediction	Covers the basic principles of RNA folding and touches upon some of the concepts in current methods that have been applied in genomic screens for de novo RNA structures in searches for novel ncRNA genes and regulatory RNA structure on mRNA.
Pervouchine DD, Khrameeva EE, Pichugina MY, Nikolaienko OV, Gelfand MS, Rubtsov PM, Mironov AA.	Evidence for widespread association of mammalian splicing and conserved long-range RNA structures	2012	1	RNA structure prediction/	Developed an efficient method for detecting conserved RNA structures on the genome-wide scale, one that does not require multiple sequence alignments and works equally well for the detection of local and long-range base pairings.

Table 14 continued from previous page

Author	Title	Year	Month	Topic	Summary
Shore AN, Kabotyanski EB, Roarty K, Smith MA, Zhang Y, Creighton CJ, Dinger ME, Rosen JM.	Pregnancy-induced noncoding RNA (PINC) associates with polycomb repressive complex 2 and regulates mammary epithelial differentiation	2012	-	Protein investigation	Shows that, in the post-poubertal mouse mammary gland, mPINC (Pregnancy-induced non-coding RNA) is enrichted in luminal and alveolar progenitors.
Will S, Siebauer MF, Heyne S, Engelhardt J, Stadler PF, Reiche, Backofen R.	LocARNAscan: Incorporating thermodynamic stability in sequence and structure-based RNA homology search	2013	4	RNA structure prediction	Introduces LocARNAscan as a tool for high throughput methods to determine RNA secondary structure, incorporating thermodynamic stability in sequence and structure.
Sabarinathan R, Tafer H, Seemann SE, Hofacker IL, Stadler PF, Gorodkin J.	RNAsnp: efficient detection of local RNA secondary structure changes induced by SNPs	2013	4	RNA secondary structure detection	Presents a strategy that focuses on the local regions of maximal structural change between mutant and wild-type. RNAsnp achieves both a noise reduction and speed-up of several orders of magnitude over shuffling-based approaches.
Will S, Yu M, Berger B.	Structure-based whole-genome realignment reveals many novel noncoding RNAs	2013	6	RNA secondary structure prediction	Presents the pipeline REAPR (RE-Alignment for Predicction of structural ncRNA), which efficiently realigns whole genomes based on RNA sequence and structure, thus allowing to boost the performance of de novo ncRNA predictors, such as RNAz
Smith MA., Gesell T., Stadler PF., Mattick JS.	Widespread purifying selection on RNA structure in mammals	2013	9	RNA structure prediction	Presents a novel benchmarking pipeline aimed at calibrating the precision of genome-wide scans for consensus RNA structure prediction. Furthermore, shows efficiency through identifying >4 mio. evolutionarily constrained RNA structures.
Ritz J, Martin JS, Laederach A.	Evolutionary evidence for alternative structure in RNA sequence co-variation	2013	-	RNA structure prediction	Sets out to characterize the evolutionary co-variation supporting alternative conformations in riboswitches to determine the extend to which alternative secondary structures are conserved.
Basu S, Müller F, Sanges R.	Examples of sequence conservation analyses capture a subset of mouse long non-coding RNAs sharing homology with fish conserved genomic elements	2013	-	Conservation of lncRNA	Predicts that between 4 and 11% of ∼2.800 investigated mouse lncRNASs present conserved sequence fragments in fish genomes. Believes this study will result useful as a reference to analyze the conservation of lncRNAs in newly sequenced genomes and transcriptomes

Table 14 continued from previous page

Author	Title	Year	Month	Topic	Summary
Hackermüller J, Reiche K, Otto C, Hösler N, Blumert C, Brocke-Heidrich K, Böhlig L, Nitsche A, Kasack K, Ahnert P, Krupp W, Engeland K, Stadler PF, Horn F	Cell cycle, oncogenic and tumor suppressor pathways regulate numerous long and macro non-protein-coding RNAs	2014	5	lncRNA identification	Finds that up to 80% of the pathway-triggered transcriptional responses are non-coding. Among these they identified very large macroRNAs with pathway-specific expression patterns and demonstrated that these are likely continuous transcripts
Reiche K, Kasack K, Schreiber S, Lüders T, Due EU, Naume B, Riis M, Kristensen VN, Horn F, Børresen-Dale AL, Hackermüller J, Baumbusch LO.	Long non-coding RNAs differentially expressed between normal versus primary breast tumor tissues disclose converse changes to breast cancer-related protein-coding genes	2014	9	Breast cancer prediction	Investigates the expression features and molecular characteristics of long non-coding RNAs in breast cancer, by investigating 26 breast tumour and 5 normal tissue samples utilizing a custom expression microarray enclosing probes for mRNAs as well as novel and previously identified lncRNAs.
Eddy SR.	Computational analysis of conserved RNA secondary structure in transcriptomes and genomes	2014	-	Improvements for detection of conserved RNA secondary structure	Proposes approaches for incorporating structure probing data into computational predictions of RNA secondary structure. (SHAPE data)
Pei S, Anthony JS, Meyer MM.	Sampled ensemble neutrality as a feature to classify potential structured RNAs	2015	2	RNA structure prediction	Hypothesizes that alignments corresponding to structured RNAs should consist of neutral sequences. Evaluates several measures of neutrality for their ability to distinguish between alignments of structured RNA sequences drawn from Rfam and various decoy alignments.
Theis C, Zirbel CL, Zu Siederdissen CH, Anthon C, Hofacker IL, Nielsen H, Gorodkin J.	RNA 3D Modules in Genome-Wide Predictions of RNA 2D Structure	2015	10	RNA secondary structure prediction	Uses RNAz in combination with 3D module prediction tools and apply them on a 13-way vertebrate sequence-based alignment. Finds that RNA 3D modules predicted by metaRNAmodulas and JAR3D are significantly enriched in the screened windows compared to their shuffled counterparts.
Seemann SE, Mirza AH, Hansen C, Bang-Berthelsen CH, Garde C, Christensen-Dalsgaard M, Torarinsson E, Yao Z, Workman CT, Pociot F, Nielsen H, Tommerup N, Ruzzo WL, Gorodkin J.	The identification and functional annotation of RNA structures conserved in vertebrates	2017	8	RNA secondary structure prediction	Computationally screened vertebrate genomes for conserved RNA structures, leveraging structure-based, rather than sequence-based, alignments. Predicts ∼516.000 human genomic regions containing conserved RNA structures, after careful correction for sequence identity and GC content.

Table 14 continued from previous page

Author	Title	Year	Month	Topic	Summary
Andrews RJ, Roche J, Moss WN.	ScanFold: an approach for genome-wide discovery of local RNA structural elements-applications to Zika virus and HIV	2018	12	RNA structure simulation	Presents a new approach where all base pairs from analysis windows are considered and weighted by favourable folding.
Kirsch R, Seemann SE, Ruzzo WL, Cohen SM, Stadler PF, Gorodkin J.	Identification and characterization of novel conserved RNA structures in Drosophila	2018	12	RNA secondary structure prediction	Provides a more comprehensive view of the ncRNA transcriptome in fly as well as evidence for differential expression of conserved RNA structures during development and in cell lines.
Turner AW, Wong D, Khan MD, Dreisbach CN, Palmore M, Miller CL.	Multi-Omics Approaches to Study Long Noncoding RNA Function in Atherosclerosis	2019	2	lncRNA function in atherosclerosis	Highlights several lncRNAs whose functional role in atherosclerosis is well-documented through traditional biochemical approaches as well as those identified through RNA-sequencing and other high-throughput assays.
Walter Costa MB, Höner Zu Siederdissen C, Dunjić M, Stadler PF, Nowick K.	SSS-test: a novel test for detecting positive selection on RNA secondary structure	2019	5	ncRNA and evolution	Introduces the SSS-Test (test for Selection on Secondary Structure) to identify positive selection and thus adaptive evolution in long non-coding RNAs.
Nowick K, Walter Costa MB, Höner Zu Siederdissen C, Stadler PF.	Selection Pressures on RNA Sequences and Structures	2019	8	ncRNA and evolution	Applies the SSS-Test (test for Selection on Secondary Structure) to explore the evolution of ncRNAs in primates and identified more than 100 long-noncoding RNAs that might evolve under positive selection in humans.

7.3 Figures of nucleotide contents

7.3.1 Nucleotide content of shuffled/negative samples by SISSIz

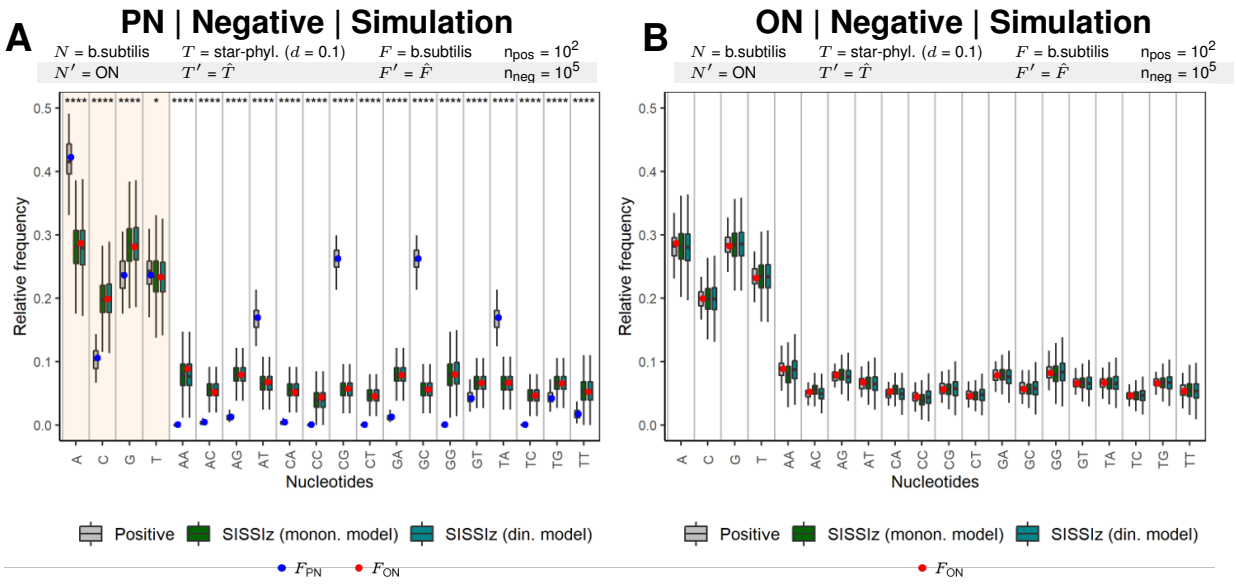

Figure 15: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 0.1 simulated by SISSIz.

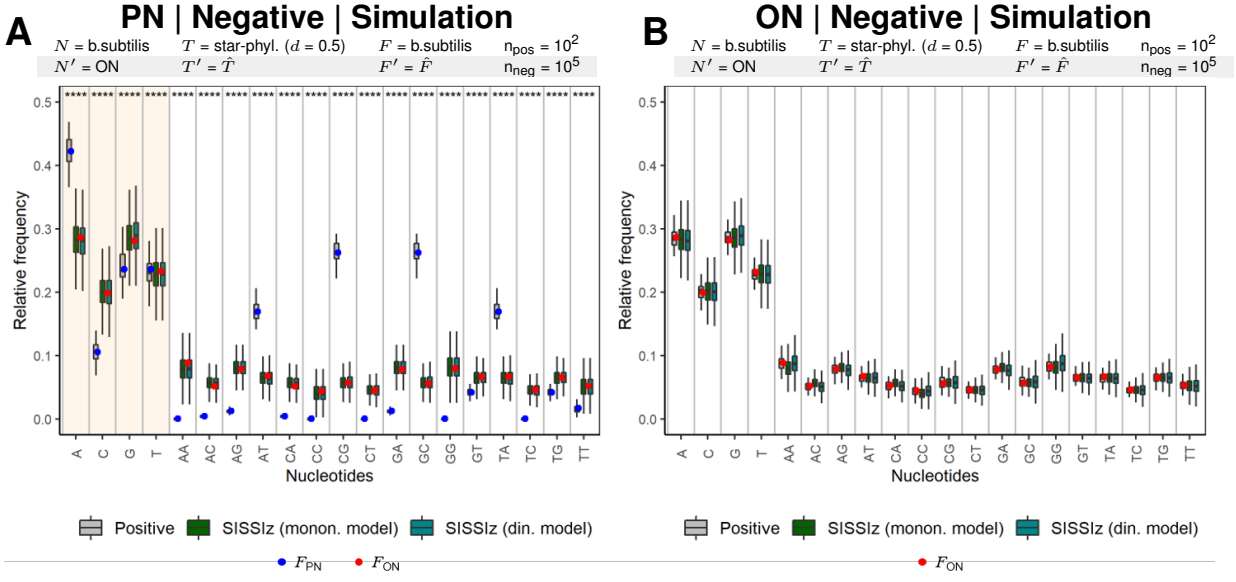

Figure 16: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 0.5 simulated by SISSIz.

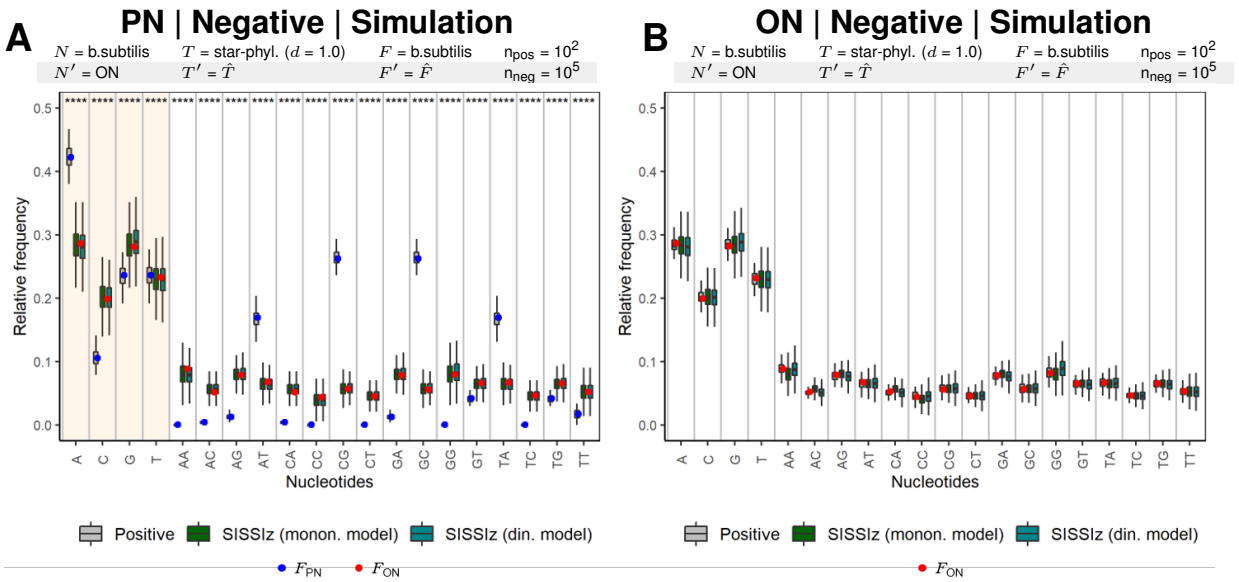

Figure 17: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 1.0 simulated by `SISSIz`.

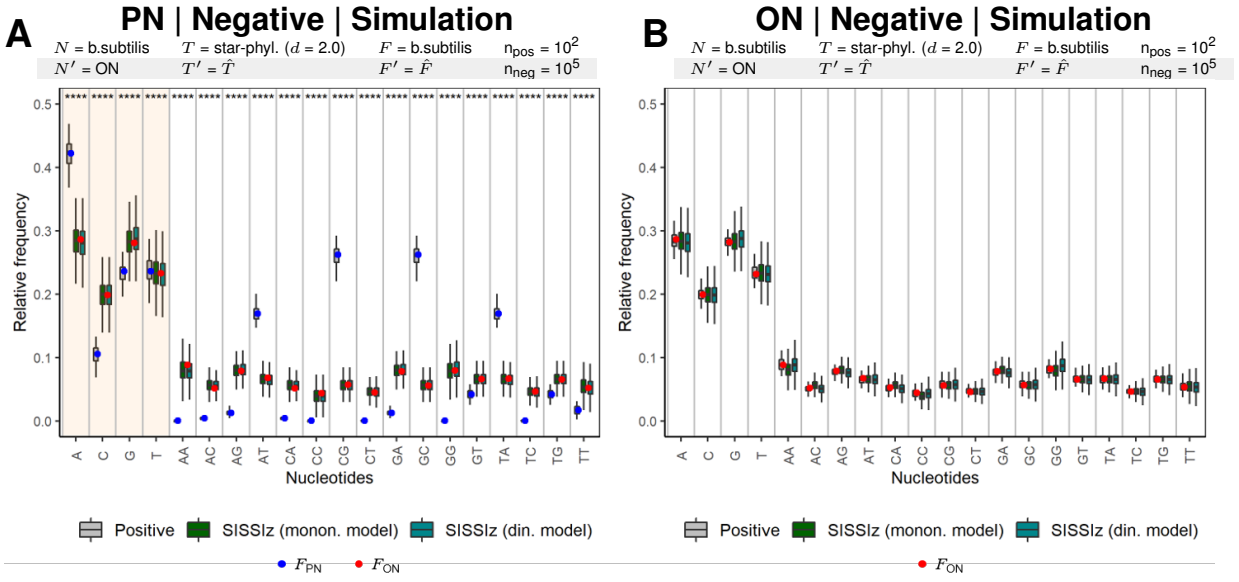

Figure 18: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 2.0 simulated by `SISSIz`.

7.3.2 Nucleotide content of shuffled/negative samples by `multiperm`

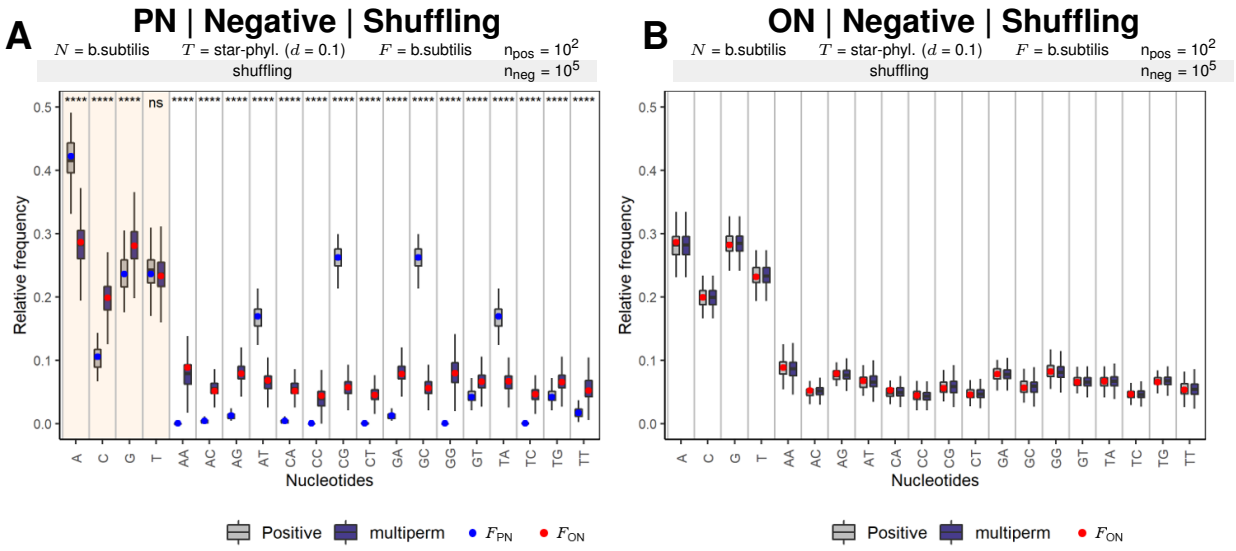

Figure 19: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 0.1 shuffled by `multiperm`.

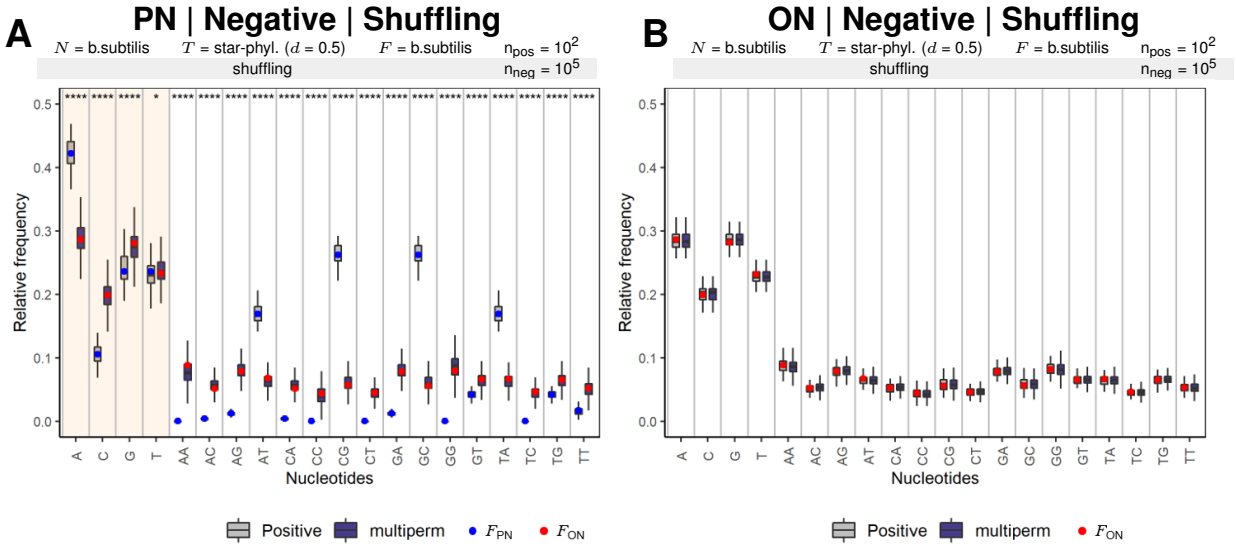

Figure 20: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 0.5 shuffled by `multiperm`.

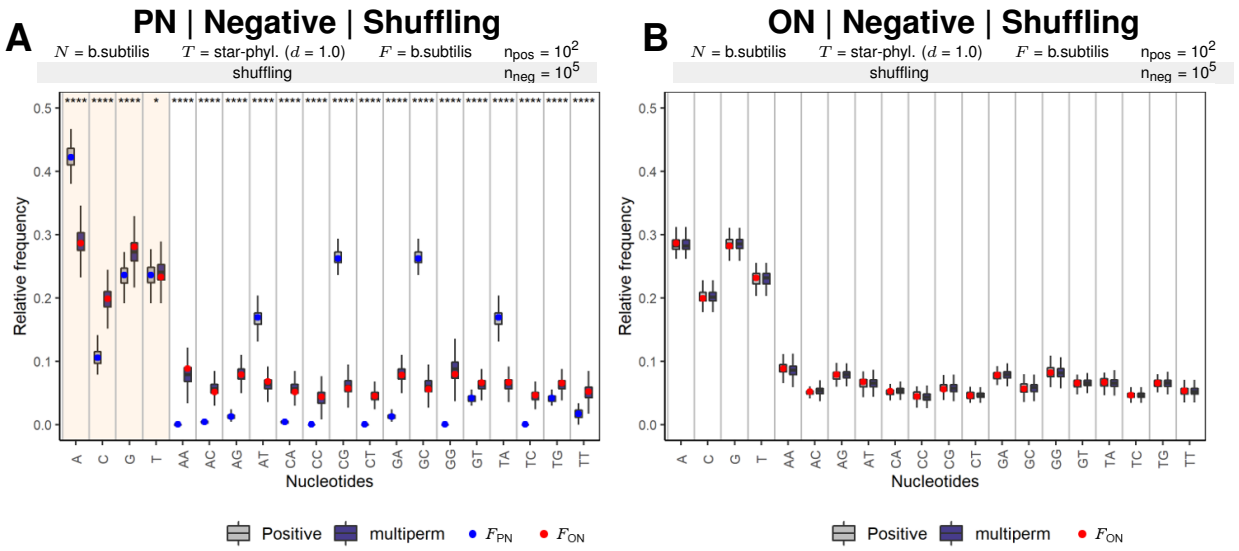

Figure 21: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 1.0 shuffled by `multiperm`.

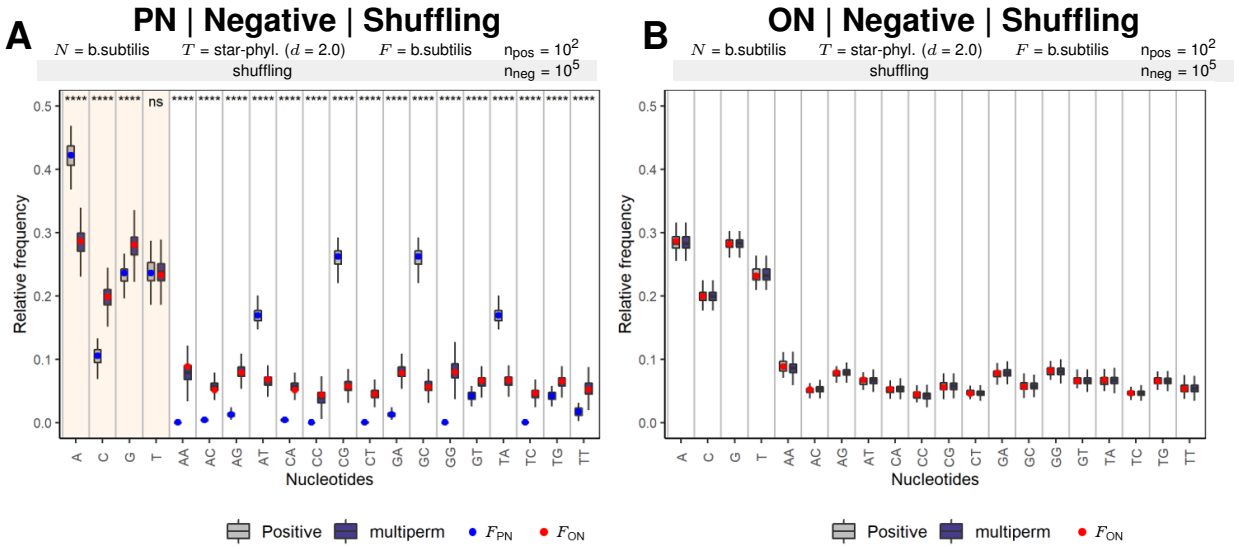

Figure 22: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 2.0 shuffled by `multiperm`.

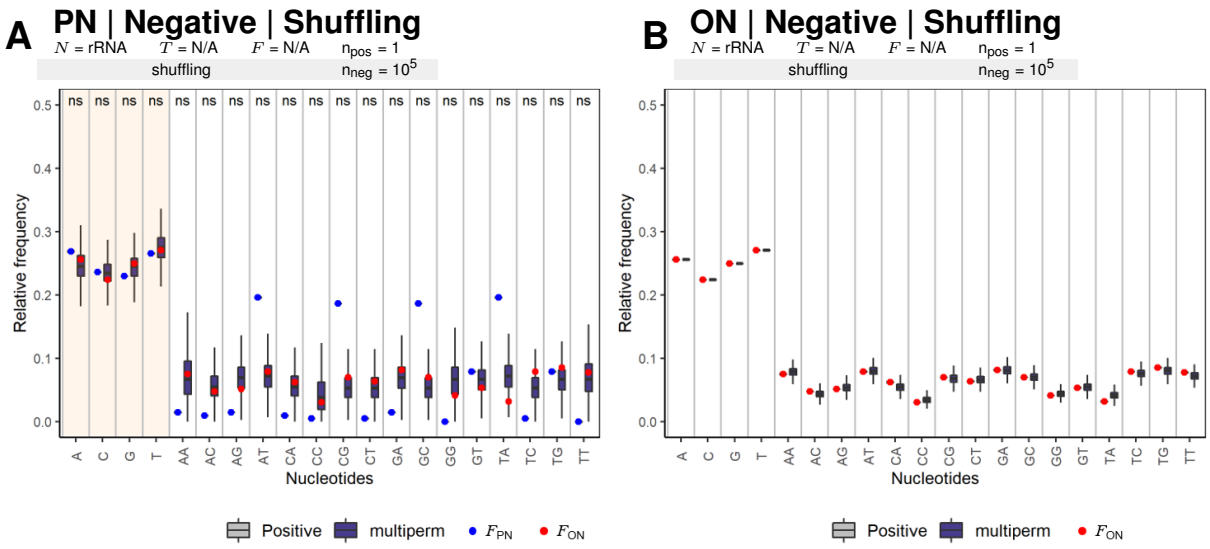

Figure 23: Nucleotide content for negative samples of the sequences generated using rRNA (provided by `SISSIz`) shuffled by `multiperm`.

7.3.3 Nucleotide content of shuffled/negative samples by `shuffle-aln`

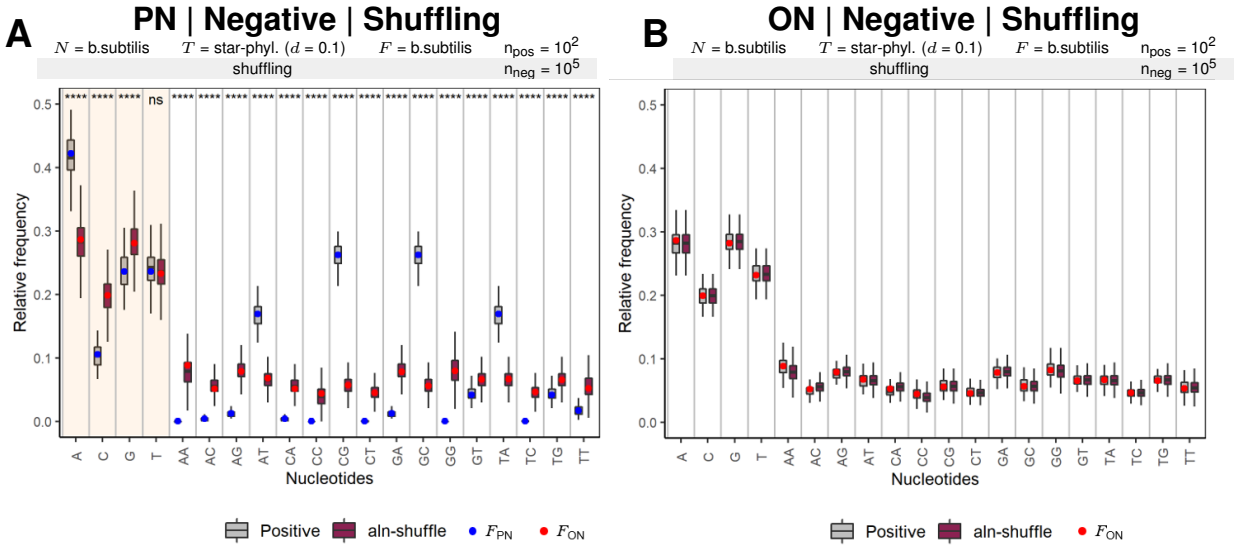

Figure 24: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 0.1 shuffled by `shuffle-aln`.

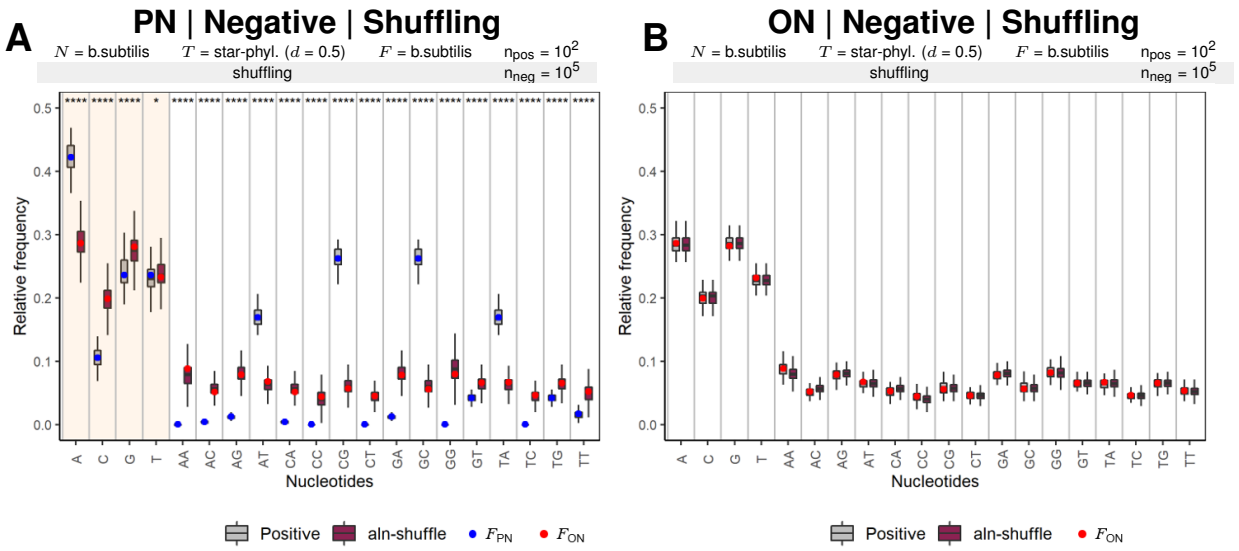

Figure 25: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 0.5 shuffled by `shuffle-aln`.

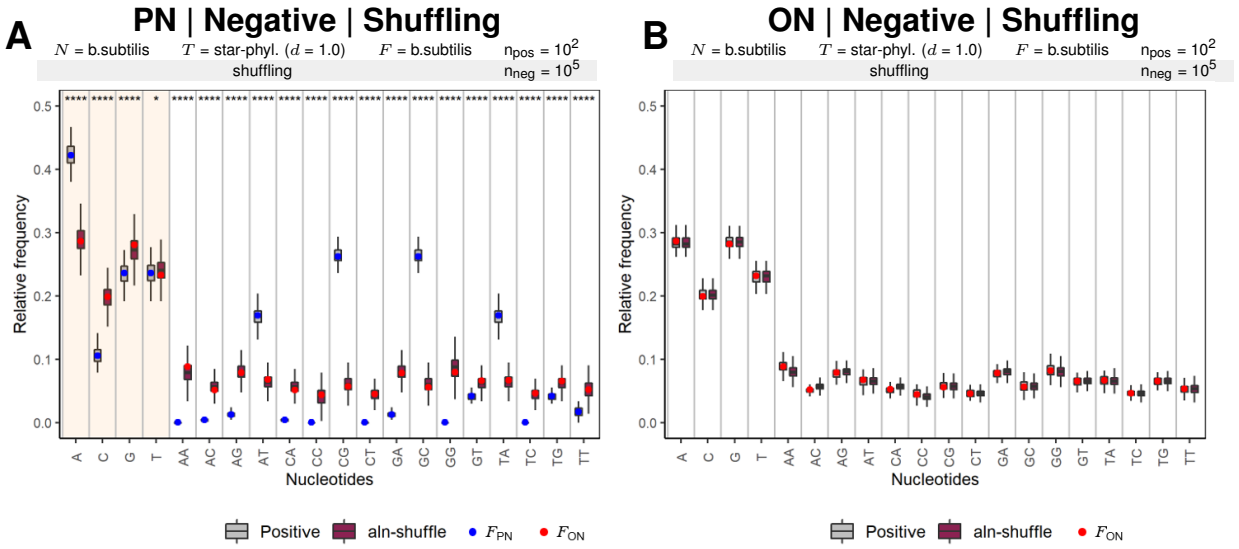

Figure 26: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 1.0 shuffled by `shuffle-aln`.

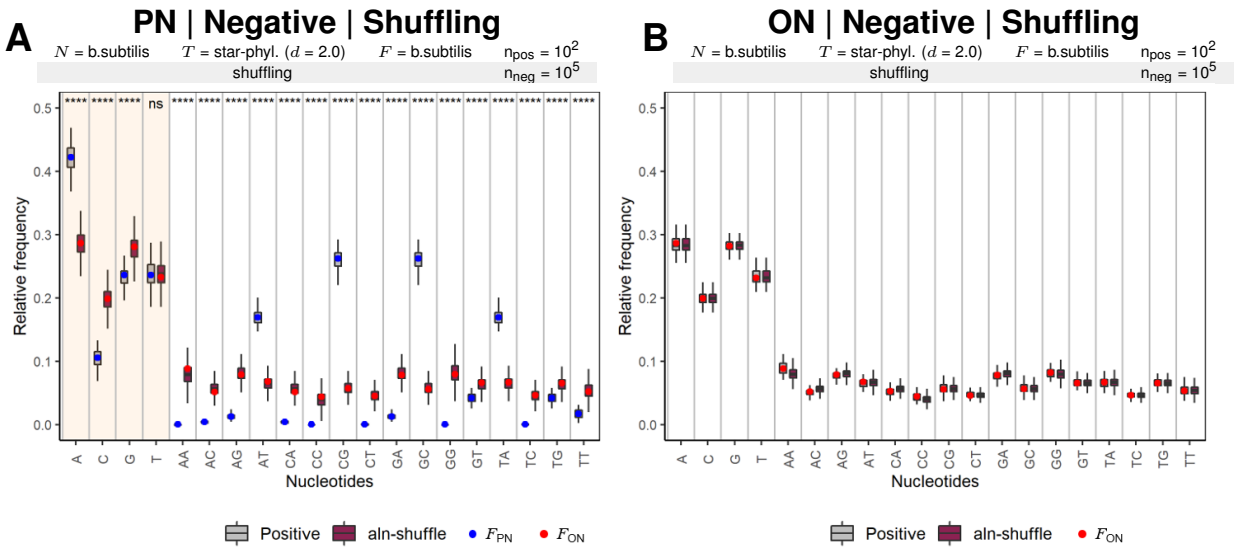

Figure 27: Nucleotide content for negative samples of the sequences generated using a star-shaped phylogeny with d = 2.0 shuffled by `shuffle-aln`.

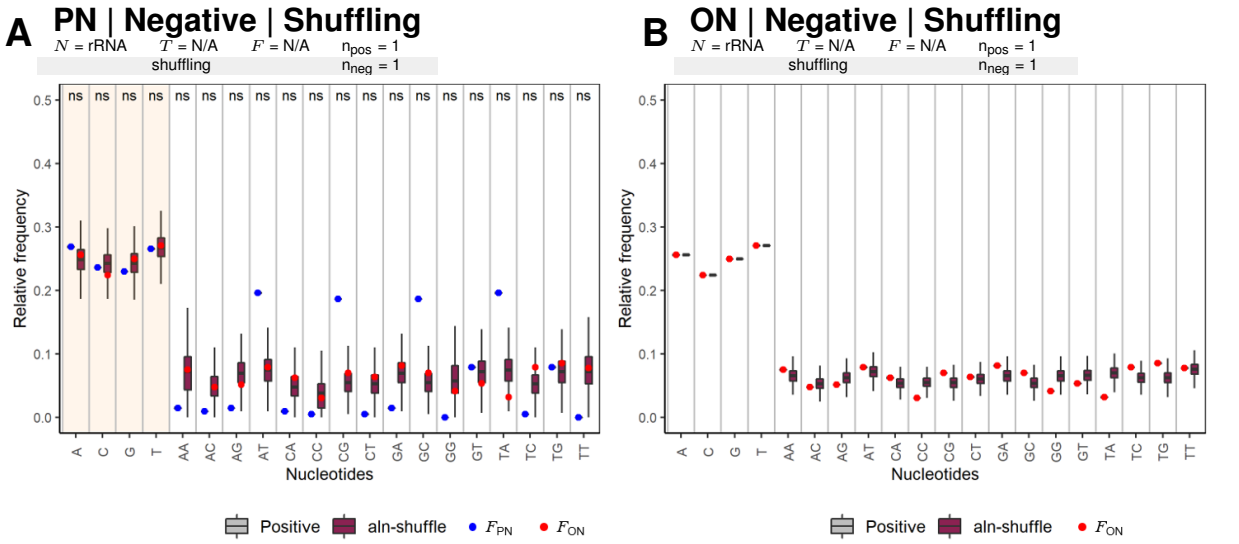

Figure 28: Nucleotide content for negative samples of the sequences generated using rRNA (provided by `SISSIz`) shuffled by `shuffle-aln`.

7.4 Figures of structure prediction indicators

7.4.1 Analysis by SISSIz

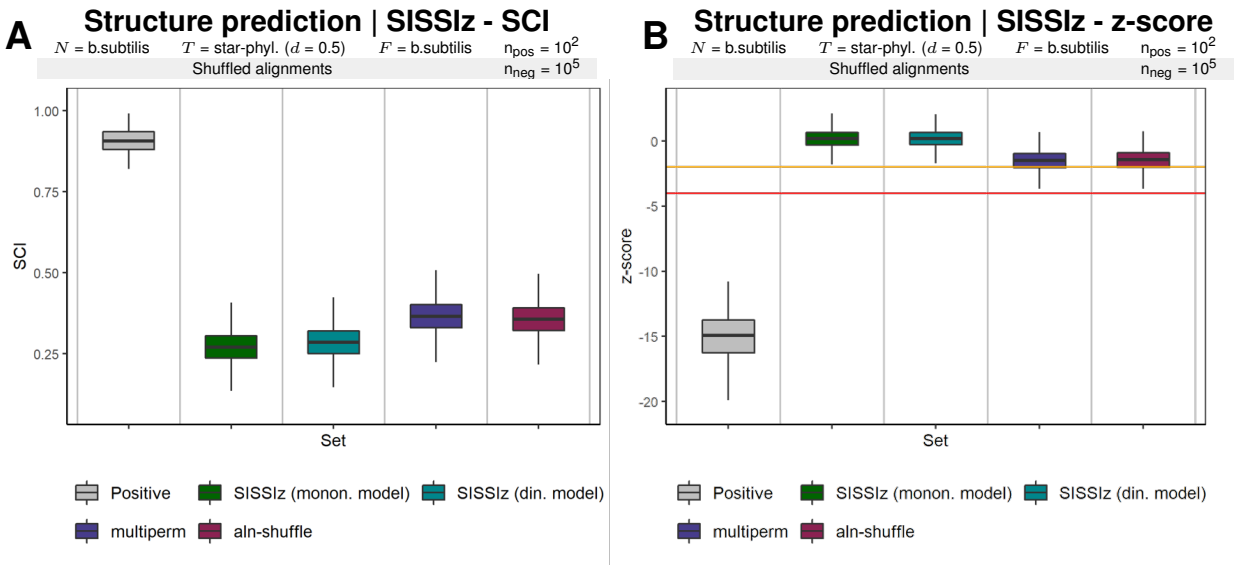

Figure 29: Structure prediction of sequences generated using a star-shaped phylogeny with d = 0.5 by SISSIz Orange lines represent lower-confidence predictions(z-score $\leq$ -2), while red lines represent high-confidence predictions (z-score $\leq$ -4) **(A)** Predicted z-score using SISSIz A lower z-score indicates more conserved structures. **(B)** Predicted SCI using SISSIz. A higher value indicates more conserved structures.

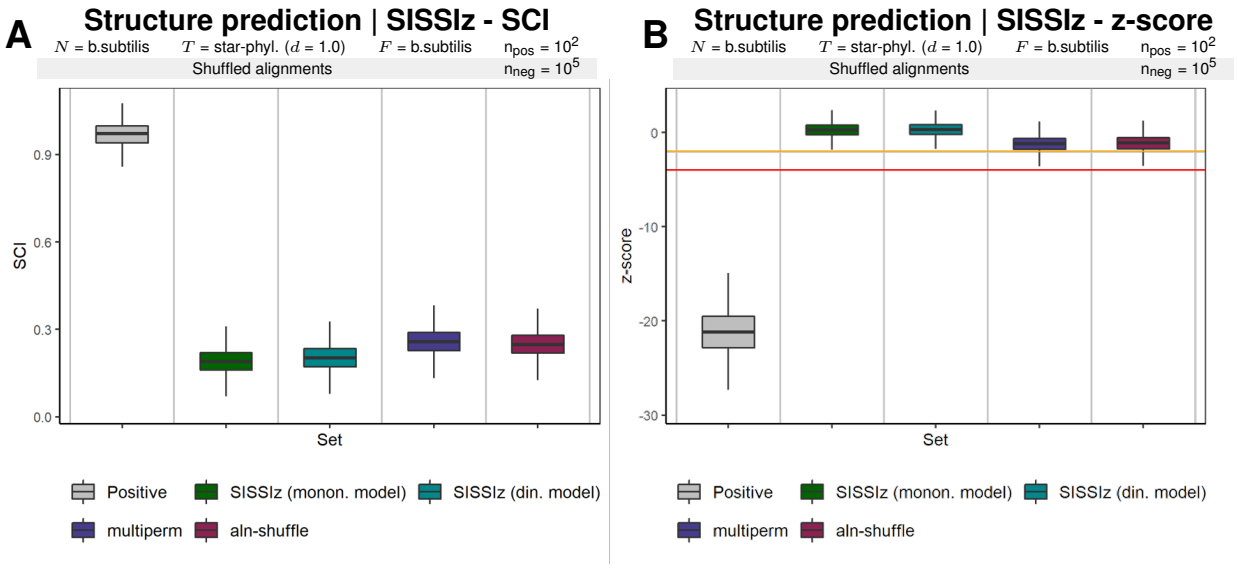

Figure 30: Structure prediction of sequences generated using a star-shaped phylogeny with d = 1.0 by SISSIz Orange lines represent lower-confidence predictions(z-score $\leq$ -2), while red lines represent high-confidence predictions (z-score $\leq$ -4) **(A)** Predicted z-score using SISSIz A lower z-score indicates more conserved structures. **(B)** Predicted SCI using SISSIz. A higher value indicates more conserved structures.

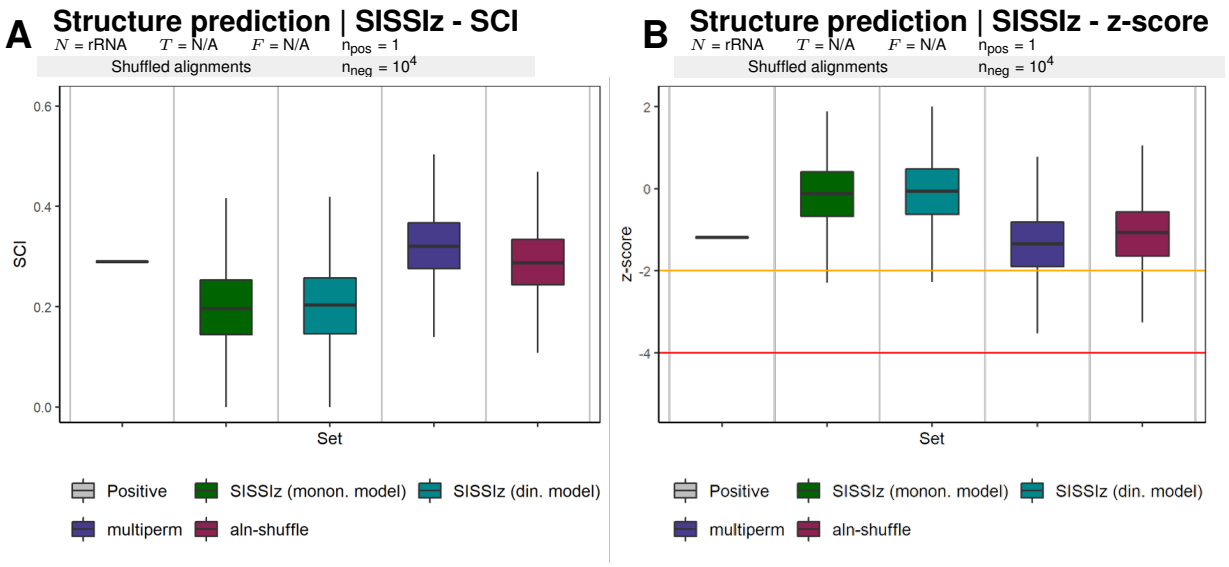

Figure 31: Structure prediction of sequences generated using rRNA (provided by `SISSIz`) by `SISSIz`. Orange lines represent lower-confidence predictions(z-score $\leq$ -2), while red lines represent high-confidence predictions (z-score $\leq$ -4) **(A)** Predicted z-score using `SISSIz`. A lower z-score indicates more conserved structures. **(B)** Predicted SCI using `SISSIz`. A higher value indicates more conserved structures.

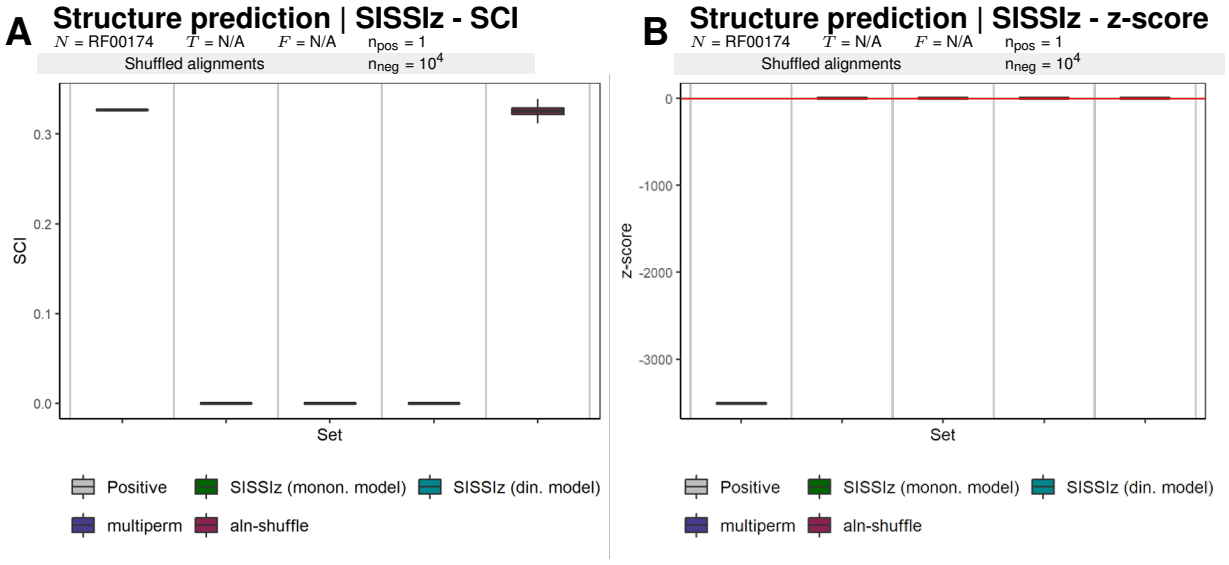

Figure 32: Structure prediction of sequences generated using RF00174 as starting point by `SISSIz`. Orange lines represent lower-confidence predictions(z-score $\leq$ -2), while red lines represent high-confidence predictions (z-score $\leq$ -4) **(A)** Predicted z-score using `SISSIz`. A lower z-score indicates more conserved structures. **(B)** Predicted SCI using `SISSIz`. A higher value indicates more conserved structures.

7.4.2 Analysis by RNAz

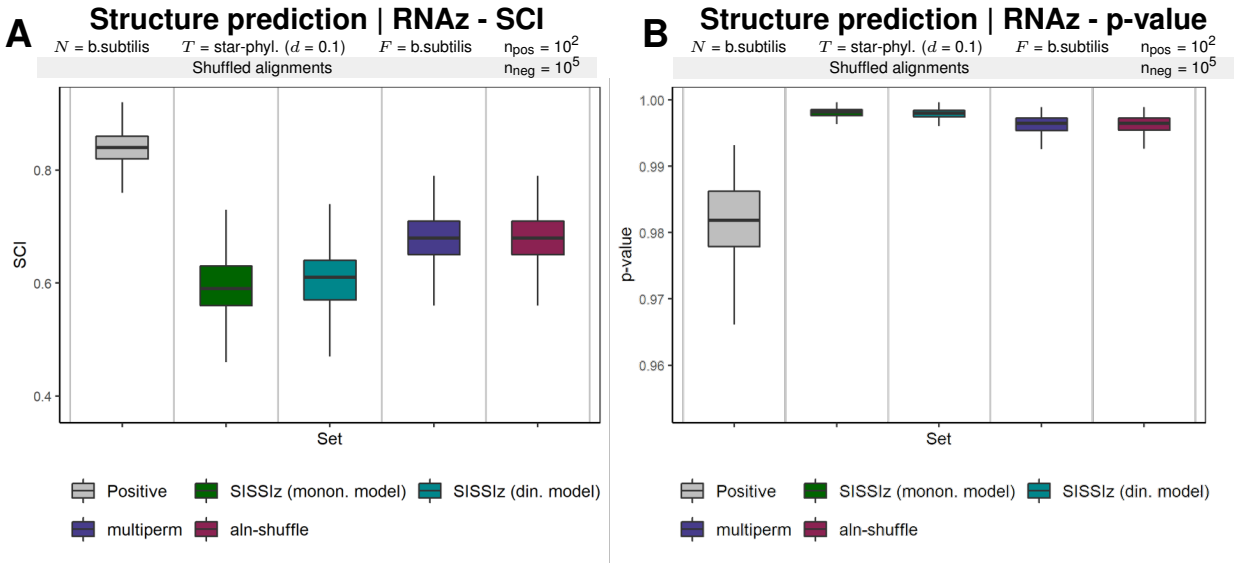

Figure 33: Structure prediction of sequences generated using a star-shaped phylogeny with d = 0.1 by RNAz. (A) Predicted z-score using RNAz. A lower z-score indicates more conserved structures. **(B)** Predicted SCI using RNAz. A higher value indicates more conserved structures.

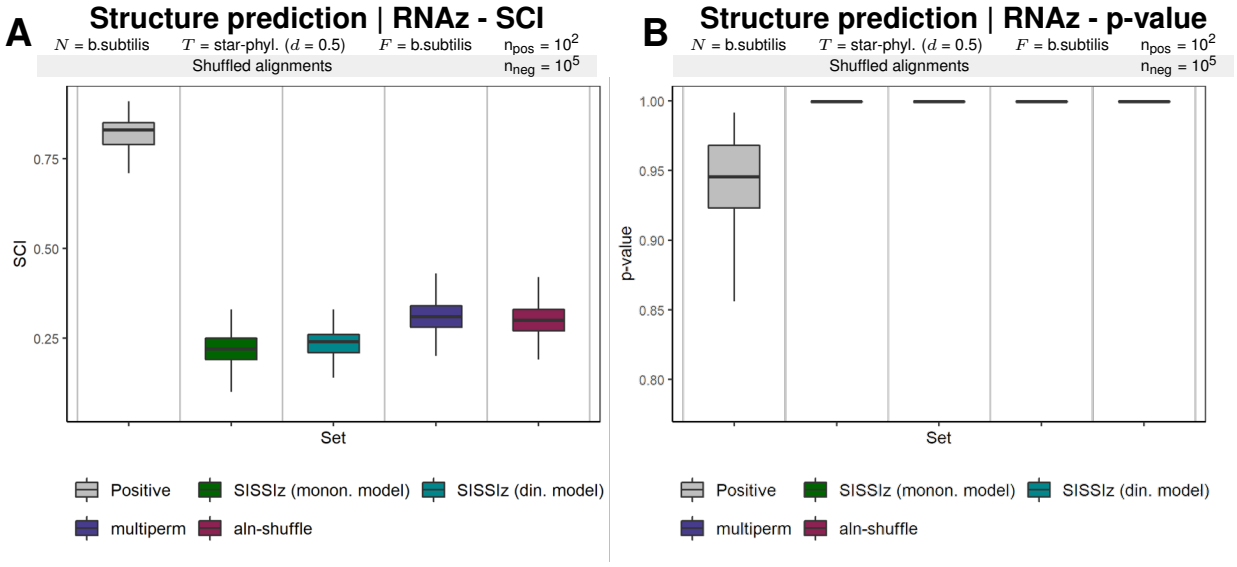

Figure 34: Structure prediction of sequences generated using a star-shaped phylogeny with d = 0.5 by RNAz. (A) Predicted z-score using RNAz. A lower z-score indicates more conserved structures. **(B)** Predicted SCI using RNAz. A higher value indicates more conserved structures.

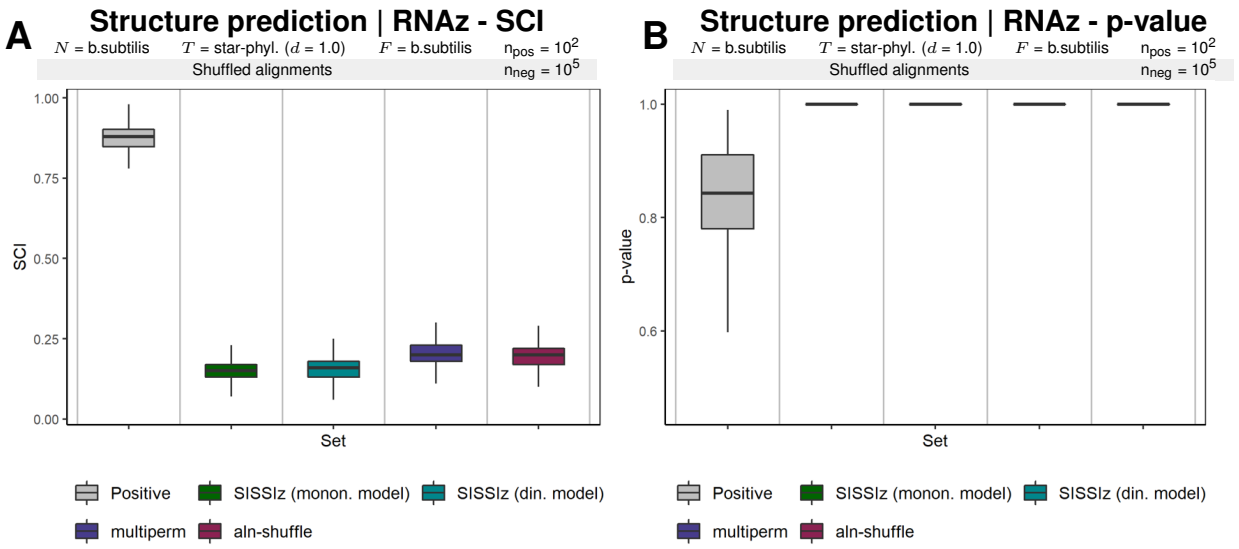

Figure 35: Structure prediction of sequences generated using a star-shaped phylogeny with d = 1.0 by `RNAz`. (A) Predicted z-score using `RNAz`. A lower z-score indicates more conserved structures. **(B)** Predicted SCI using `RNAz`. A higher value indicates more conserved structures.

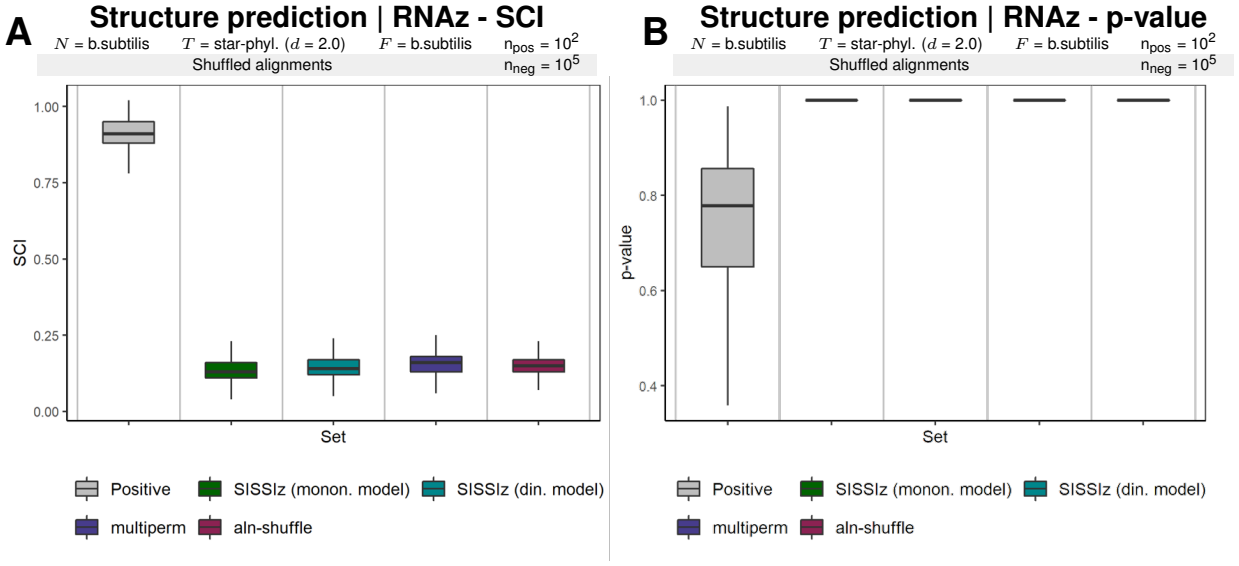

Figure 36: Structure prediction of sequences generated using a star-shaped phylogeny with d = 2.0 by `RNAz`. (A) Predicted z-score using `RNAz`. A lower z-score indicates more conserved structures. **(B)** Predicted SCI using `RNAz`. A higher value indicates more conserved structures.

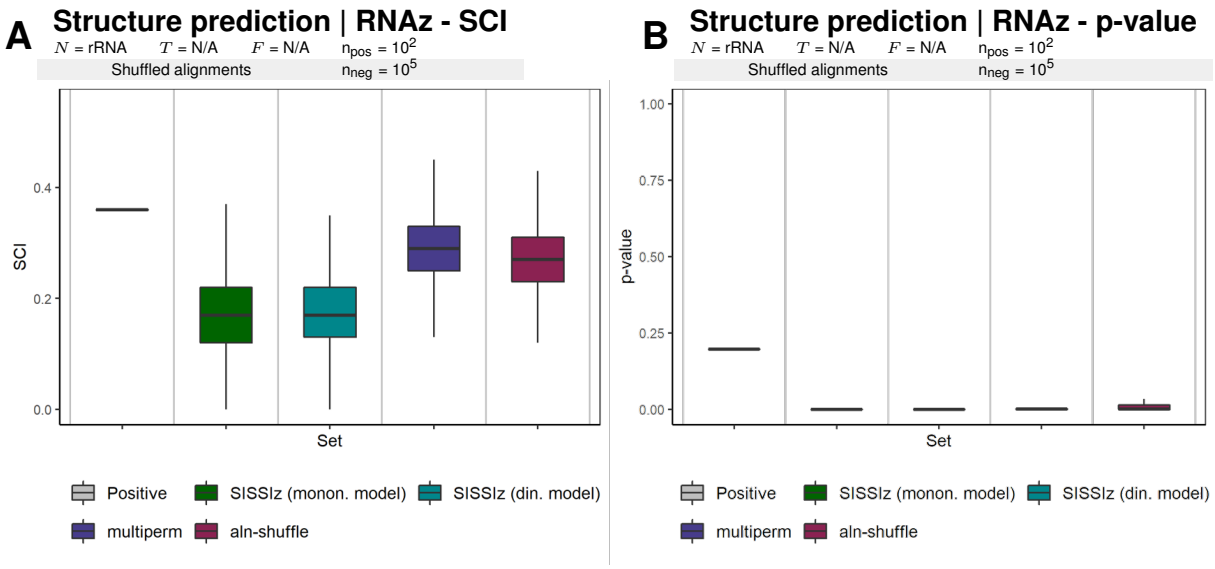

Figure 37: Structure prediction of sequences generated using rRNA (provided by `SISSIz`) by `RNAz`. **(A)** Predicted z-score using `RNAz`. A lower z-score indicates more conserved structures. **(B)** Predicted SCI using `RNAz`. A higher value indicates more conserved structures.

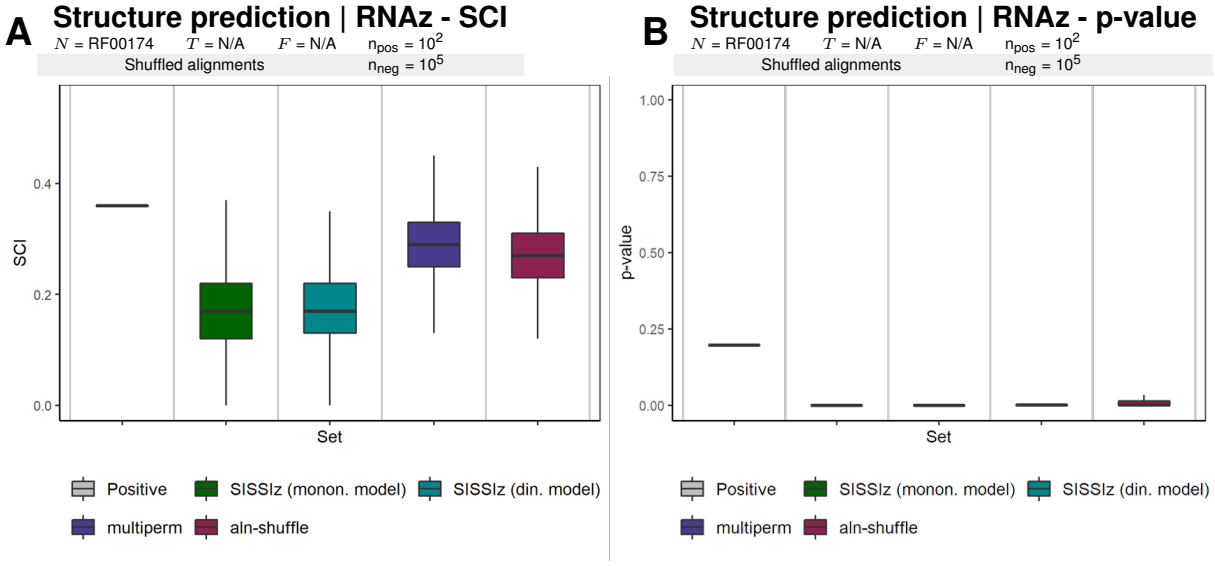

Figure 38: Structure prediction of sequences generated using RF00174 as starting point by `RNAz`. **(A)** Predicted z-score using `RNAz`. A lower z-score indicates more conserved structures. **(B)** Predicted SCI using `RNAz`. A higher value indicates more conserved structures.

7.4.3 Analysis by `EvoFold`

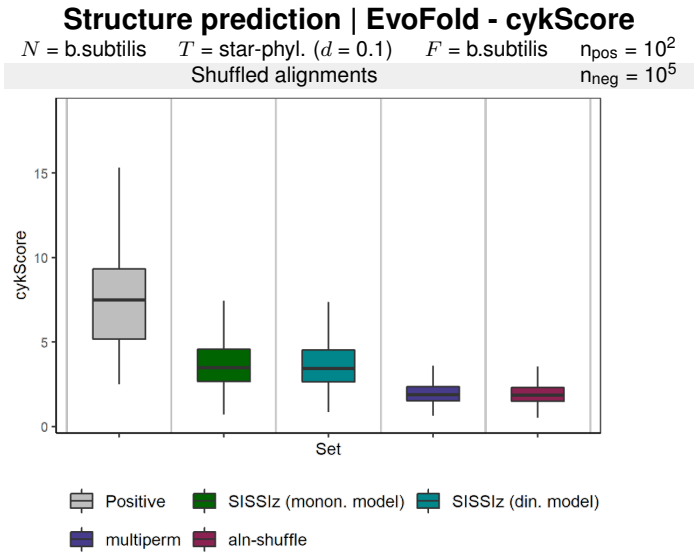

Figure 39: Structure prediction of sequences generated using a star-shaped phylogeny with d = 0.1 by `EvoFold`. Predicted cykScore using `EvoFold`. A higher cykScore indicates more conserved structures.

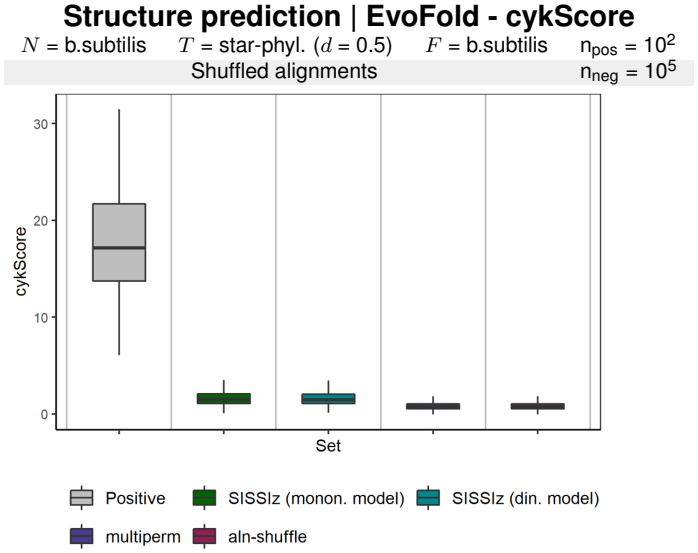

Figure 40: Structure prediction of sequences generated using a star-shaped phylogeny with d = 0.5 by `EvoFold`. Predicted cykScore using `EvoFold`. A higher cykScore indicates more conserved structures.

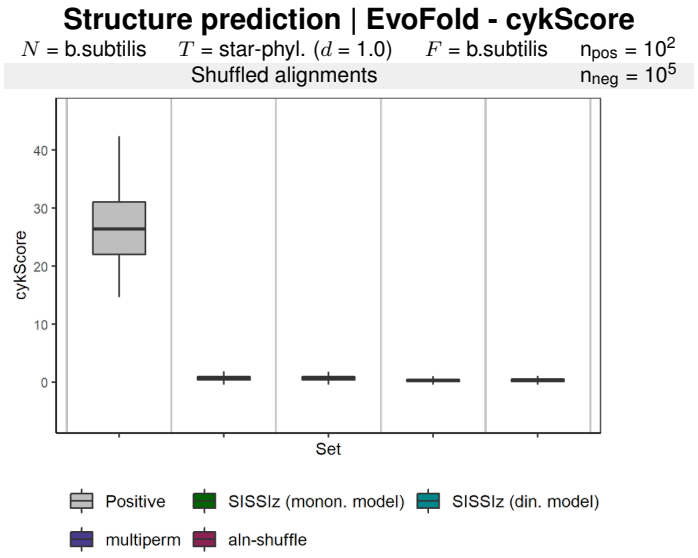

Figure 41: Structure prediction of sequences generated using a star-shaped phylogeny with d = 1.0 by `EvoFold`. Predicted cykScore using `EvoFold`. A higher cykScore indicates more conserved structures.

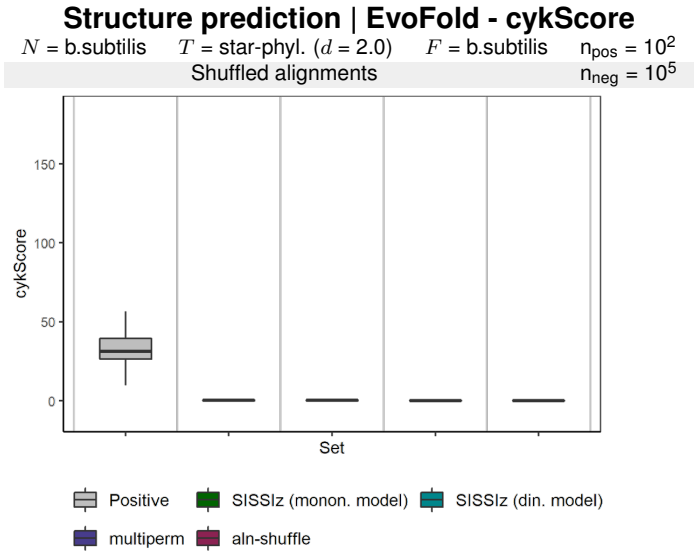

Figure 42: Structure prediction of sequences generated using a star-shaped phylogeny with d = 2.0 by `EvoFold`. Predicted cykScore using `EvoFold`. A higher cykScore indicates more conserved structures.

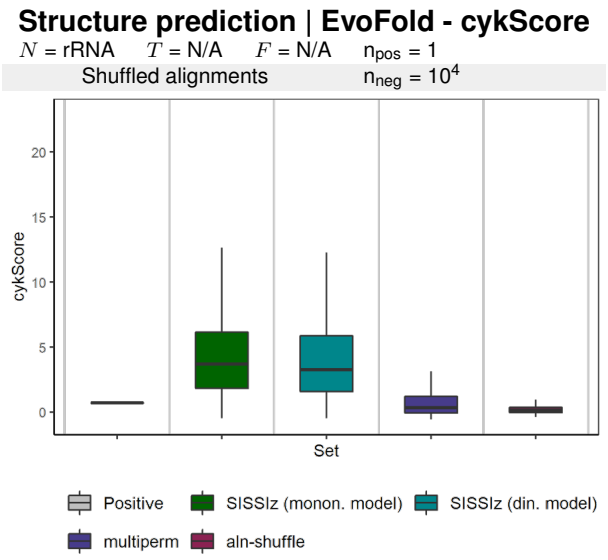

Figure 43: Structure prediction of sequences generated using rRNA (provided by `SISSIz`) by `EvoFold`. Predicted cykScore using `EvoFold`. A higher cykScore indicates more conserved structures.

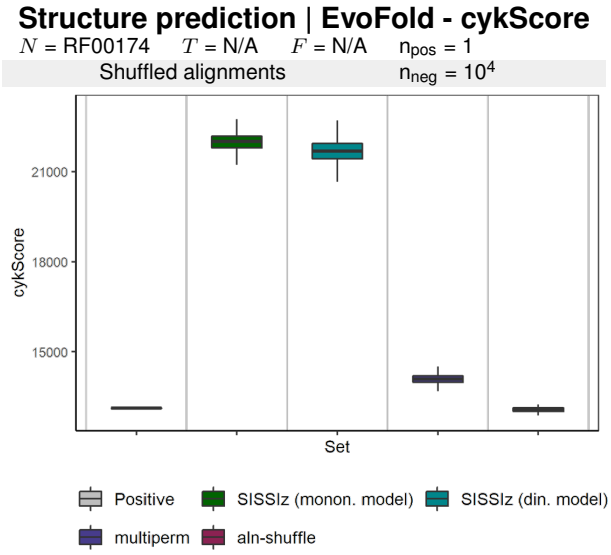

Figure 44: Structure prediction of sequences generated using RF00174 as starting point by `EvoFold`. Predicted cykScore using `EvoFold`. A higher cykScore indicates more conserved structures.