Universität Bielefeld

Forschungsbericht der
Technischen Fakultät
Abteilung Informationstechnik

Rose: Generating Sequence Families

Jens Stoye, Dirk Evers and Folker Meyer

Report 97-04

Universität Bielefeld • Postfach 10 01 31 • 33501 Bielefeld • FRG

Impressum: Herausgeber:
Robert Giegerich, Alois Knoll, Peter Ladkin,
Helge Ritter, Gerhard Sagerer, Ipke Wachsmuth

Technische Fakultät der Universität Bielefeld,
Abteilung Informationstechnik, Postfach 10 01 31,
33501 Bielefeld, FRG

ISSN 0946-7831

Contents

Chapter 1

Abstract

Motivation:
We present a new probabilistic model of evolution of RNA-, DNA-, or protein-like sequences and a tool *Rose* that implements this model. Guided by an evolutionary tree, a family of related sequences is created from a common ancestor sequence by insertion, deletion and substitution of characters. During this artificial evolutionary process, the "true" history is logged and the "correct" multiple sequence alignment is created simultaneously. The model also allows for varying rates of mutation within the sequences making it possible to establish so-called sequence motifs.
Results:
The data created by *Rose* is suitable for the evaluation of methods in multiple sequence alignment computation and the prediction of phylogenetic relationships. It can also be useful when teaching courses in or developing models of sequence evolution and in the study of evolutionary processes.
Availability:
The software *Rose* is available on the Bielefeld Bioinformatics WebServer under the following URL:
http://bibiserv.TechFak.Uni-Bielefeld.DE/rose/
The sourcecode is available upon request.
Contact:
E-mail: folker@TechFak.Uni-Bielefeld.DE

Chapter 2

Introduction

It is useful for many reasons to have a family of sequences with well-known evolutionary history. This kind of data is used in the study of evolutionary processes, in the evaluation of multiple sequence alignments methods, and in the reconstruction of phylogenetic trees. Other applications in computational molecular biology may also benefit from its availability. Unfortunately, nature does not provide "benchmark" problems well suited for all these applications since there is no way to learn the exact phylogeny of the sequences involved. Therefore it is common practice to artificially create sequence data trying to be as close to the real world as possible.

The simulation of evolutionary processes on the molecular sequence level has a long tradition. Starting with the model of Jukes and Cantor [10], several generalizations and alterations have been presented, e.g. [11, 4, 9, 14]. These models were designed for the study of molecular evolution on the sequence level, focusing on a well-founded statistical basis rather than on producing sequence families most similar to those usually considered in molecular biology. The early models even ignored the well-known fact of insertions and deletions (*indels*) during evolution. Some models [18, 19] consider indels but still have some other restrictions.

To create most realistic sequence families, we have added indels and "sequence motifs" (patterns in a family of related sequences [20]) to the so-called HKY-model [9] which only allows the description of arbitrary-rate substitutions in DNA sequences. We also extended the underlying alphabet to cover amino acid sequences. An evolutionary process is simulated by iterated mutation of a "common ancestor sequence" following the edges of a given "mutation guide tree". This way, the topology of the tree induces the relationship of the sequences. The mutations are performed by insertion, deletion, and substitution of single characters or whole subsequences. Figure 2.1 sketches the creation process of a family of four sequences. In addition to knowing the exact evolutionary *distance* of the sequences, our approach provides us with their whole *evolutionary history* and the *true alignment*. Therefore, in contrast to biological applications, it is easily pos-

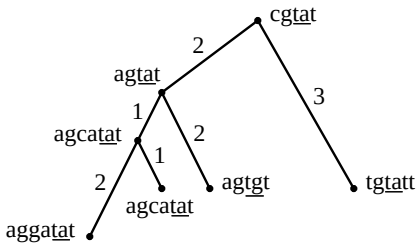

Figure 2.1: Example of a creation process of four sequences from a common ancestor `cgtat`. The underlined part denotes a sequence motif with smaller substitution probability.

sible to verify predictions about alignments and phylogenetic relationships drawn from the sequences simply by comparing the predicted phylogeny to the tree that was used in the creation process.

In fact we can go one step further and evaluate the adequacy of mathematical models such as maximum parsimony or sum-of-pairs multiple alignment. Given a program that calculates the best solution according to the model on a data set generated by rose, we may contrast these results to the "true" phylogeny or alignment.

The data created by our tool *Rose* (random-model of sequence evolution) has been extensively tested with the *Divide-and-Conquer Alignment* [17, 16] and *GeneFisher* [5, 13] software packages.

Chapter 3

Systems and Methods

For reasons of speed, efficiency and portability *Rose* was developed under UNIX using the ANSI C programming language. The software has been tested on various UNIX platforms e.g. DEC, HP, LINUX-PC, SGI, Sun. The actual program development was done on a Sun Sparcstation using gcc and Sun SPro C compilers, as well as bison and lex to build the input parser. The publicly available version runs on a Sun Enterprise 3000 server.

Chapter 4

Algorithm

4.1 The Model

Our procedure requires the following input:

an **Alphabet** $\mathcal{A}$
of size l, e.g. the DNA-alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ or the 20 character amino acid alphabet,

a **Root Sequence** s or an **Average Sequence Length** n
(if no root sequence is specified, a random sequence of length $n \geq 1$ is generated),

Character Frequencies $f = (f_1, \ldots, f_l)$
satisfying $\sum_{i=1}^{l} f_i = 1$ used for insertions and the creation of the root sequence (if not specified),

a **Mutation Guide Tree** T or a **Sequence Distance** d_{av}
the tree may be supplied with edge lengths (otherwise all edges are assumed to have uniform length 1), if no tree is entered, a binary mutation guide tree of user defined average pairwise sequence distance d_{av} (see 4.3.1) is created,

a **Mutation Matrix** M
of size $l \times l$ representing pairwise mutation frequencies used for substitutions,

Insertion and **Deletion Probability Functions**
representing the probability of an indel event p_{ins} or p_{del}, combined with indel length functions l_{ins} and l_{del}, respectively, and

a **Mutation Probability Vector** v
of length n allowing to specify regions of different mutation rate, e.g. to specify sequence motifs.

Given these parameters, *Rose* generates

a **Family of Sequences $s_1, \ldots, s_m$**
containing sequences with average length n and average pairwise evolutionary distance d_{av},

a **Multiple Sequence Alignment A**
of the sequences $s_1, \ldots, s_m$ that is correct with respect to the creation process i.e. it reflects the "true" evolutionary history of $s_1, \ldots, s_m$, and finally

a **Relatedness Tree T'**
showing the phylogenetic relationship of the created sequences. T' is the smallest subtree of T which contains all the nodes corresponding to the generated sequences (and possibly some additional inner nodes which can be seen as extinct ancestors).

An outline of the algorithm is given here:

$Rose(\mathcal{A}, s, n, f, T, d_{av})$
begin
  if $undefined(s)$
    $s := create\_root\_sequence(\mathcal{A}, n, f)$;
  fi
  if $undefined(T)$
    $T := create\_guide\_tree(d_{av})$;
  fi
  $T.seq := s$; //copy root sequence to root of tree
  $traverse(T)$; //recursively mutate sequences along tree
  $print\_sequences(T)$; //generate output
  $print\_alignment(T)$;
  $print\_tree(T)$;
end

where sub-function $traverse$ is implemented as follows:

$traverse(T)$
foreach subtree T' of T **do**
  $T'.seq := evolve(T.seq)$
  $traverse(T')$
od

In the following subsections, we take a closer look at the different steps of *Rose*.

4.2 The Root Sequence

The implementation of function $create\_root\_sequence$ is straightforward: If no pre-given root sequence is specified, each of the n positions in the root sequence is independently filled by a random process that returns letter $\mathcal{A}_i$ ($1 \leq i \leq l$) with probability f_i.

Rose works with arbitrary alphabets and any matching list of frequencies. For amino acid sequences we implemented as default values the *normalized frequencies* of the amino acids given in [3], and for nucleotides we use the frequencies given in [1].

4.3 The Mutation Guide Tree

The general behavior of $create\_guide\_tree$ is similar to that of $create\_root\_sequence$: If no tree T is specified, *Rose* computes a uniform binary tree with $k = 1023$ nodes whose edge labels are adjusted such that the average sequence distance (i.e. the expected length of a shortest path between two randomly chosen nodes in the tree) meets the user-defined value d_{av} (see below). After the tree is created, either only from the leaves or from the leaves and inner nodes (chosen by the user), the required number of sequences is selected uniformly. So, in the latter case it can happen that, at the same time, an inner node sequence and a sequence from the corresponding subtree is chosen.

Obviously it is possible to save space and computation time by pruning the unnecessary edges in the tree before performing the evolutionary process if not all of the sequences are contained in the final sequence family.

4.3.1 Adjusting the Edge Lengths

Assume a binary uniform tree of depth k with $K = 2^{k+1} - 1$ nodes and constant length b of every edge (see Figure 4.1). For the moment, let $b = 1$. Then, the average sequence distance d_{av} is the sum of all pairwise distances in the tree divided by $K(K-1)$, the number of pairs of distinct nodes. Consider therefore a node in level κ of the uniform binary tree, $0 \leq \kappa \leq k$. In the example of Figure 4.1, we have chosen $\kappa = 2$. The corresponding node is indicated by a circle.

In each level i, $0 \leq i \leq k - \kappa$ of the subtree "below" the observed node, there are 2^i nodes with distance i. The sum of distances to all these nodes is

$$\begin{aligned} D_\kappa &:= \sum_{i=0}^{k-\kappa} (2^i \cdot i) \\ &= 2^{k-\kappa+1}(k-\kappa-1) + 2. \end{aligned}$$

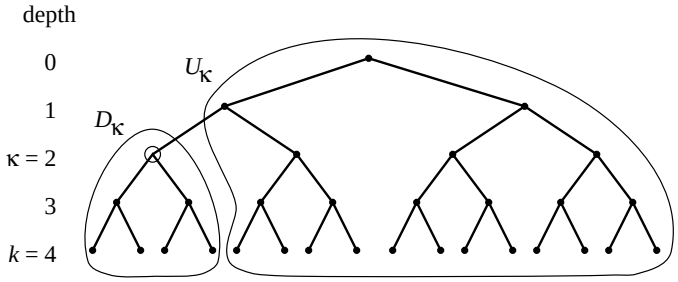

Figure 4.1: Uniform binary tree of depth $k = 4$ with $K = 2^5 - 1 = 31$ nodes. For a node in depth $\kappa = 2$ (marked by the circle), those nodes contributing to D_κ and U_κ, respectively, are shown.

Additionally, there are κ nodes "above" the observed node, each being the starting point of a subtree. Summing the distances to all these nodes gives

$$\begin{aligned} U_\kappa &:= \sum_{i=1}^{\kappa} \left(i + \sum_{j=1}^{k-(\kappa-i)} 2^{j-1} \cdot (i+j) \right) \\ &= 2^{k-\kappa+1}(\kappa - k + 3) \\ &\quad + 2^{k+1}(\kappa + k - 3) + \kappa. \end{aligned}$$

Thus, the total sum of distances from a node in level κ to all other nodes is

$$\begin{aligned} N_\kappa &:= D_\kappa + U_\kappa \\ &= 2^{k-\kappa+2} + 2^{k+1}(\kappa + k - 3) + \kappa + 2. \end{aligned}$$

Averaging this value over all pairs of distinct nodes, we obtain

$$\begin{aligned} d_{av} &:= \frac{\sum_{\kappa=0}^{k}(2^\kappa \cdot N_\kappa)}{K(K-1)} \\ &= 2 \cdot 2^{k+1} \, \frac{4 + k(1 + 2^{k+1}) - 2^{k+2}}{(2^{k+1}-1)(2^{k+1}-2)} \end{aligned}$$

which approximates

$$2 \, \frac{k \cdot 2^{k+1} - 2^{k+2}}{2^{k+1}} \quad = \quad 2 \, (k-2)$$

for sufficiently large k.

Similarly, if all edges have length b, we get

$$d_{av} \approx 2b \, (k-2).$$

Hence, to obtain sequences of a pre-given relatedness, we simply have to alter the edge length b:

$$b \approx \frac{d_{av}}{2\ (k-2)}.$$

For example, to obtain sequences of an average distance $d_{av} \approx 250$ PAM, the edge length of our default tree with $1023 = 2^{9+1} - 1$ nodes has to be set to $b \approx \frac{250}{2\ (9-2)} \approx 18$.

Note that in the above calculation we assumed that the sequences are selected from both the internal nodes and the leaves of the mutation guide tree. In case sequences are selected only from the leaves, a similar calculation leads to the formula

$$b \approx \frac{d_{av}}{2\ (k-1)}.$$

4.4 Creation of Child Sequences

We now take a closer look at the implementation of function $evolve$, the core of *Rose*. The following steps are used to create a new "descendant" sequence s_{new} from a given ancestor sequence s_{old}:
$evolve(s_{old})$

1. The mutation function $mutate$ for the given alphabet is applied to every position i in s_{old}:

   $$s_{new}[i] = mutate(s_{old}[i], b)$$

   where b is the length of the branch leading to the new node. The mutation matrix is selected with respect to b as described below.

2. One or more subsequences are deleted from s_{new} taking into account the deletion probability p_{del} and the deletion length function l_{del}:

   $$perform\_deletions(p_{del}, l_{del})$$

3. One or more sequences are inserted at arbitrary positions in s_{new}:

   $$perform\_insertions(p_{ins}, l_{ins})$$

Function $mutate$ makes use of the mutation probability matrix M. An entry $M[i,j]$ is interpreted as the probability for the jth letter of the alphabet $\mathcal{A}$ being substituted by the ith letter. Hence, the sum of each column of M should be $\sum_{i=1}^{l} M[i,j] = 1$ for all $j = 1, \ldots, l$. The diagonal values $M[i,i]$ determine the degree of stability: For example, a value of $M[i,i] = .99$ for all $i = 1, \ldots, l$

will result in an average mutability of one percent accepted mutations per unit of branch length.

In case the mutation matrix M is the probability matrix of one *accepted amino acid substitution per hundred sites* (1 PAM) given in [3] – which is our default for proteins – we denote this new unit of measure for the distance of a child sequence from its ancestor including insertions and deletions by 1 PAM* where the parameters for insertions and deletions have to be specified additionally.

Evolutionary rates of more than 1 PAM* are obtained by applying the creation procedure repeatedly. As Schöniger and v. Haeseler [14] have shown, the use of a custom matrix (such as PAM 10) helps to save time when the number of substitutions exceeds an upper bound. At each step along an edge of the guide tree, depending on the mutation rate the decision is made either to use precomputed PAM* matrices repeatedly or to compute a new custom matrix.

4.5 Sequence Motifs

Up to this point, we have assumed a constant rate of mutation over the whole length of the sequences. This is not very realistic: The mutation rate of genomic sequences found in nature is not constant for all positions in the genome. Mutations in regions with strong functional and/or structural importance are less often observed than elsewhere.

Therefore we have generalized the function $evolve$: We allow the use of different rates of mutation for different regions of the sequence by a vector v of length n with values $v_i \geq 0$ which linearly increase/decrease the degree of variability at position i of the root sequence. A value $v_i = 1$ yields exactly the variability given by the edge length. Values $v_i < 1$ suppress mutations ($v_i = 0$: no mutation) and higher values $v_i > 1$ allow to specify regions of particular high mutation rate, for example so-called *hot spots*. The vector v is inherited by child sequences. Indels are forbidden in regions with $v_i < 1$, thus establishing conserved sequence motifs. Inserted regions have a variability of 1.

4.6 Creation of Indels

It is obvious that the exact mechanism of insertion and deletion is crucial for the simulation of evolution. Unfortunately there is neither a well established model (like HKY for nucleotide substitution) nor consensus as to the number of indels that corresponds to a certain evolutionary distance. We therefore chose to accomodate a wide range of possibilities with a function that we call *inverted gap function*. The following pseudocode shows the selection and creation of inser-

tions; deletions are handled analogous:

$perform\_insertions(p_{ins}, l_{ins})$
begin
  do $T.dist$ **times**
    if $random\_number\_between\_zero\_and\_one() < p_{ins}$
      $pos := choose\_random\_position(length(s_{old}))$;
      $len := compute\_insertion\_length(l_{ins})$;
      if $v_i >= 1, \forall i \in \{pos, \ldots, pos + len\}$
        $do\_insertion(pos, len)$;
      fi
    fi
  od
end

The starting position for the insertion in $choose\_random\_position$ is selected uniformly among the positions $1, \ldots, length(s_{old})$. To allow a high degree of variability, *Rose* accepts any quantized length function $l_{ins} = l_{ins}^{(1)}, \ldots, l_{ins}^{(q_{ins})}$ with $\sum_{i=1}^{q_{ins}} l_{ins}^{(i)} = 1$. Then, length $len \in \{1, \ldots, q_{ins}\}$ is selected with probability $l_{ins}^{(len)}$.

Note that the average sequence length remains n if $p_{ins} = p_{del}$ and $l_{ins} = l_{del}$.

Function $do\_insertion$ finally is similar to the creation of the root sequence; the characters inserted maintain the initial character distribution.

Chapter 5

Implementation

5.1 Input/Output formats

The user input is done via an HTML forms interface, the user can also choose to feed a file with all the input information into *Rose* using a simple tag value format. The format and the parameters are further described in our online manual http://bibiserv.TechFak.Uni-Bielefeld.DE/rose/manual.html.

5.2 Resource Requirements

On a Sun Ultra I 167 MHz CPU, *Rose* used the following resources:

protein			DNA		
#seqs	sec	MB	#seqs	sec	MB
10	1.8	1.1	10	3.3	1.4
100	9.8	1.9	100	18.6	3.8
500	24.9	4.2	500	49.5	9.6

Here, the created protein sequences have an average length of 250 letters and an average relatedness of 250 PAM*; the DNA sequences have an average length of 1000 letters and an average relatedness of 50.

5.3 Examples

The following examples show some of the features and demonstrate the versatility of *Rose*.

(a)
```
FSAEAALVSPGKGDDEQVPNKDKCVYHGHKDGKRMNVKTPPTGPLVVGVHQ
YEGANEVGATCEESSYCYVKEQAIQVKESQECTDFARHEVKSFRGVPGKLTEVIPVPL
YGAAHPVGDPIKLGSLFLNHYESKGHTAAMCLLGMKTELIEPIEVQA
SGVTEPVPNPVPATGIKLDKYTREENCLGMCLMGMGPPMVTIGEVGI
```

(b)
```
FSAEAALVSP--------GKGDDEQVPNKDKCVYHGHKDGKRMNVKTPPTGPLVVGVHQ
YEGANEVGATCEESSYCYVKEQAIQVKESQECTDFARHEVKSFRGVPGKLTEV-IPVPL
YGAAHPVGDP--------IKLGSLFLNH---YESKGHTAAMCLLGMKTELIEP-IEVQA
SGVTEPVPNP--------VPATGIKLDK---YTREENCLGMCLMGMGPPMVTI-GEVGI
```

(c)
```
FSAEAALVSP--------GKGDDEQVPNKDKCVYHGHKDGKRMNVKTPPTGPLVVGVHQ
YEGANEVGATCEESSYCYVKEQAIQVKESQECTDFARHEVKSFRGVPGKLTE-VIPVPL
YGAAHPVGDP--------IKLGSLFL---NHYESKGHTAAMCLLGMKTELIE-PIEVQA
SGVTEPVPNP--------VPATGIKL---DKYTREENCLGMCLMGMGPPMVT-IGEVGI
```

Figure 5.1: (a) Sample family of random sequences obtained with *Rose* for n = 50 and m = 4; (b) "true" alignment of these sequences; (c) a score-optimal alignment according to PAM 250 substitution matrix and gap function $g(l)$ = 8 + 12l computed with the program MSA. While the overall optimal alignment is correct, the exact location of the gaps does not coincide in all cases.

5.3.1 A Protein Sequence Family

In Figure 5.1 (a), a sample family with $m = 4$ sequences of average length $n = 50$ is shown. This family is created with the default settings of *Rose*: A uniform binary mutation guide tree of depth $k = 9$ and uniform edge length $b = 18$ PAM*. The probability for insertions and deletions is set to $p_{ins} = p_{del} = 0.3\%$, and the insertion and deletion length functions are exponentially decreasing with a maximal length value of 10.

The alignment given in Figure 5.1 (b) is the "true" alignment corresponding to the creation process of the sequences. Figure 5.1 (c) shows an optimal alignment according to the PAM 250 substitution matrix [3] (in distance form with values between 0 and 24) and gap function $g(l) = 8 + 12\,l$ computed with the program MSA [12, 8]. While the overall optimal alignment is correct, the exact location of the gaps does not coincide in all cases. This suboptimality of true alignments regarding the standard alignment score functions is also shown by the (distance) scores for both alignments: The "true" alignment has an alignment score of 5184, while the optimal alignment a has a "better" score 5166.

5.3.2 A Simple DNA Sequence Family with Motif

The use of motifs in sequence families created by *Rose* is demonstrated in Figure 5.2. The upper part shows the "true" alignment of a family of 5 DNA sequences which contains a conserved TATAAT motif obtained with *Rose* using a mutation

(a)
```
AGTG------ACTATAAT---CG---GAGGACAG--
ATTCTGT---CCTATAAT---CG---GAGAAAAGCC
AGTCTGT---ACTATAATGTTGG---GAGGAAAAGC
AGTCCGTTGC--TATAAT---GG---GAGGAAAACC
AATCTGT---AGTATAAT---GGTGTGAGGAAAGCC
```

(b)
```
AGT----GACTATAAT---CGGAGGACAG--
ATTCTG-TCCTATAAT---CGGAGAAAAGCC
AGTCTG-TACTATAATGTTGGGAGGAAAAGC
AGTCCGTTGCTATAAT---GGGAGGAAAACC
AATCTG-TAGTATAATGGTGTGAGGAAAGCC
```

Figure 5.2: DNA example with TATAAT motif: (a) the "true" and (b) an optimal alignment.

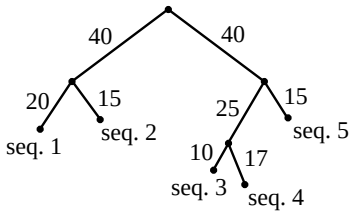

Figure 5.3: Relatedness tree for the sequences shown in Figure 5.4.

vector disallowing mutations within the motif while outside the mutability remains normal. Figure 5.2 (b) shows a score-optimal alignment of these sequences computed with MSA (unit substitution cost with gap function $g(l) = 2 + l$). It is considerably shorter than the "true" alignment. The parsimony objective underlying the sum-of-pairs scoring of MSA fails here.

5.3.3 A Protein Sequence Family with Varying Mutation Rate

Finally we present a protein example where we fixed the root sequence and the mutation guide tree. We also varied the mutability along the sequence.

As root sequence we took the human hemoglobin alpha sequence. The mutation guide tree is shown in Figure 5.3. The true alignment of our "artificial globins" is shown in Figure 5.4. The histogram above the alignment shows the mutation probability along the sequence allowing a higher mutation rate between the α helices than within.

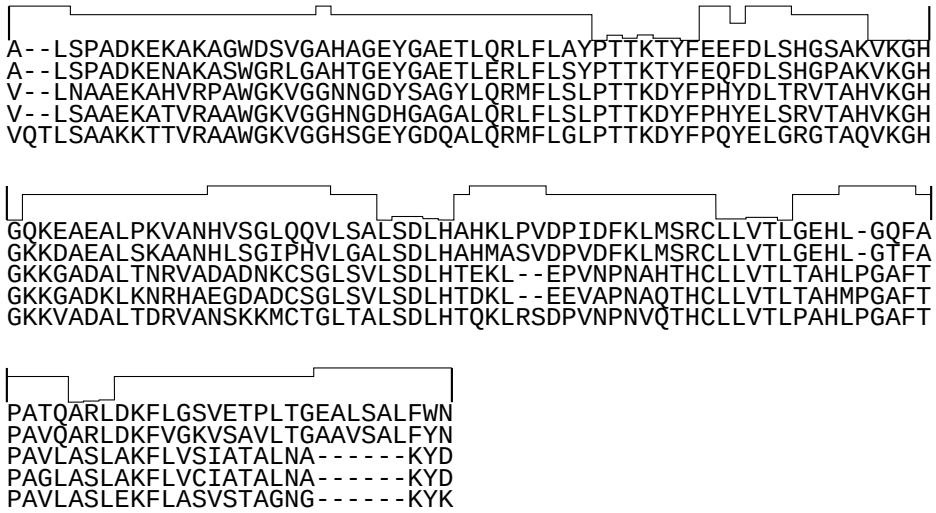

Figure 5.4: Family of “globins” created by using human hemoglobin alpha as root sequence. The mutability vector is shown in form of a histogram above the alignment.

Chapter 6

Discussion and Conclusion

The data sets created by *Rose* are artificial sequence families that contain both *indels* and *motifs*. The evaluation of multiple sequence alignment tools and phylogenetic reconstruction tools is possible with these benchmarks.

Previous models were mainly designed to better understand evolutionary processes rather than create nature-like sequence families. While such studies need a rigorous probabilistic foundation, they are quite far from a realistic simulation of the biological truth. Even the most sophisticated model [19] including indels of complete blocks cannot describe overlapping insertions and deletions as two evolutionary events since fragments cannot vary over time. Our "fragments" (i.e. inserted or deleted regions) can vary over time and hence overlap. Our model is based on empirically verified parameters. It is not *a priori* clear by which parameters the most natural results can be obtained and there does not seem to exist a single set of evolutionary parameters describing the whole variety one finds in nature. Therefore with *Rose* the user is free to set whatever parameters seem reasonable for the actual purpose.

While we have removed a number of limits that existed so far, there are still some limitations: while we do not assume that the characters of the sequences evolve independently and with the same rate in the whole family, we have not yet included a feature that simulates different rates of evolutionary pressure in different branches of the tree, enabling different lineages to evolve independently within our tree. This has been observed by a number of biologists [6, 7, 15, 2]. While we are planning to include this feature in a future release of *Rose* and extend the scope of our model even further, it is important to note that all results have to regard the adequacy of the chosen evolutionary parameters, and that simulations can only aid the evaluation of algorithms. What matters in the end is the success on *real* biological sequences.

Chapter 7

Acknowledgments

The authors wish to thank Robert Giegerich for encouraging us to work on this approach and Marcie McClure for making a good point. We would also like to thank the referees for valuable hints.

The work was partially supported by the German Ministry for Education and Sciences (BMBF), the Ministry of Science of North Rhine Westfalia (MWF-NRW), the German Research Council graduate program (DFG-GK) Strukturbildungsprozesse, and the German Academic Exchange Service (DAAD).

Bibliography

[1] P. Agarwal and D. J. States. A Bayesian Evolutionary Distance for Parametrically Aligned Sequences. *J. Comp. Biol.*, 3(1):1–17, 1996.

[2] S. A. Benner, M. A. Cohen, and G. H. Gonnet. Amino Acid Substitution during Functionally Constrained Divergent Evolution of Protein Sequences. *Protein Engng.*, 7(11):1323–1332, 1994.

[3] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A Model of Evolutionary Change in Proteins. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, volume 5, suppl. 3, pages 345–352. National Biomedical Research Foundation, Washington, D.C., 1979.

[4] J. Felsenstein. Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach. *J. Mol. Evol.*, 17:368–376, 1981.

[5] R. Giegerich, F. Meyer, and C. Schleiermacher. GeneFisher-Software support for the detection of postulated genes. In *Proc. of the Fourth Conference on Intelligent Systems for Molecular Biology, ISMB 96*, pages 68–78. AAAI Press, Menlo Park, CA, USA, 1996.

[6] J. Greer. Comparative Model-Building of the Mammalian Serine Proteases. *J. Mol. Biol.*, 153:1027–1042, 1981.

[7] J. Greer. Comparative Modeling Methods: Applications to the Family of the Mammalian Serine Proteases. *PROTEINS: Structure, Function, and Genetics*, 7:317–334, 1990.

[8] S. K. Gupta, J. D. Kececioglu, and A. A. Schäffer. Improving the Practical Space and Time Efficiency of the Shortest-Paths Approach to Sum-of-Pairs Multiple Sequence Alignment. *J. Comp. Biol.*, 2(3):459–472, 1995.

[9] M. Hasegawa, H. Kishino, and T. Yano. Dating of the Human-Ape Splitting by a Molecular Clock of Mitochondrial DNA. *J. Mol. Evol.*, 22:160–174, 1985.

[10] T. H. Jukes and C. R. Cantor. Evolution of Protein Molecules. In Munro, H. N., editor, *Mammalian Protein Metabolism*, volume 3, pages 21–132. Academic Press, New York, NY, USA, 1969.

[11] M. Kimura. A Simple Method for Estimating Evolutionary Rates of Base Substitutions through Comparative Studies of Nucleotide Sequences. *J. Mol. Evol.*, 16:111–120, 1980.

[12] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A Tool for Multiple Sequence Alignment. *Proc. Natl. Acad. Sci. USA*, 86:4412–4415, 1989.

[13] F. Meyer and C. Schleiermacher. Genefisher. http://bibiserv.TechFak.Uni-Bielefeld.DE/genefisher/, 1996.

[14] M. Schöniger and A. von Haeseler. Simulating Efficiently the Evolution of DNA Sequences. *CABIOS*, 11(1):111–115, 1995.

[15] G. E. Schulz, E. Schiltz, A. G. Tomasselli, R. Frank, M. Brune, A. Wittinghofer, and R. H. Schirmer. Structural Relationships in the Adenylate Kinase Family. *Eur. J. Biochem.*, 161:127–132, 1986.

[16] J. Stoye. DCA: Divide and Conquer Multiple Sequence Alignment, version 1.0. http://bibiserv.TechFak.Uni-Bielefeld.DE/dca/, 1997.

[17] J. Stoye, V. Moulton, and A. W. M. Dress. DCA: An Efficient Implementation of the Divide-and-Conquer Approach to Simultaneous Multiple Sequence Alignment. *CABIOS*, To appear.

[18] J. L. Thorne, H. Kishino, and J. Felsenstein. An Evolutionary Model for Maximum Likelihood Alignment of DNA Sequences. *J. Mol. Evol.*, 33:114–124, 1991.

[19] J. L. Thorne, H. Kishino, and J. Felsenstein. Inching toward Reality: An Improved Likelihood Model of Sequence Evolution. *J. Mol. Evol.*, 34:3–16, 1992.

[20] T. D. Wu and D. L. Brutlag. Identification of Protein Motifs Using Conserved Amino Acid Properties and Partitioning Techniques. In *Proc. of the Third Conference on Intelligent Systems for Molecular Biology, ISMB 95*, pages 402–410. AAAI Press, Menlo Park, CA, USA, 1995.

Bisher erschienene Reports an der Technischen Fakultät
Stand: 18. April 1997

94–01 Modular Properties of Composable Term Rewriting Systems
(Enno Ohlebusch)

94–02 Analysis and Applications of the Direct Cascade Architecture
(Enno Littmann und Helge Ritter)

94–03 From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction
(Robert Giegerich und Stefan Kurtz)

94–04 Die Verwendung unscharfer Maße zur Korrespondenzanalyse in Stereo Farbbildern
(Andrè Wolfram und Alois Knoll)

94–05 Searching Correspondences in Colour Stereo Images — Recent Results Using the Fuzzy Integral
(Andrè Wolfram und Alois Knoll)

94–06 A Basic Semantics for Computer Arithmetic
(Markus Freericks, A. Fauth und Alois Knoll)

94–07 Reverse Restructuring: Another Method of Solving Algebraic Equations
(Bernd Bütow und Stephan Thesing)

95–01 PaNaMa User Manual V1.3
(Bernd Bütow und Stephan Thesing)

95–02 Computer Based Training-Software: ein interaktiver Sequenzierkurs
(Frank Meier, Garrit Skrock und Robert Giegerich)

95–03 Fundamental Algorithms for a Declarative Pattern Matching System
(Stefan Kurtz)

95–04 On the Equivalence of E-Pattern Languages
(Enno Ohlebusch und Esko Ukkonen)

96–01 Static and Dynamic Filtering Methods for Approximate String Matching
(Robert Giegerich, Frank Hischke, Stefan Kurtz und Enno Ohlebusch)

96–02 Instructing Cooperating Assembly Robots through Situated Dialogues in Natural Language
(Alois Knoll, Bernd Hildebrandt und Jianwei Zhang)

96–03 Correctness in System Engineering
(Peter Ladkin)

96–04 An Algebraic Approach to General Boolean Constraint Problems
(Hans-Werner Güsgen und Peter Ladkin)

96–05 Future University Computing Resources
(Peter Ladkin)

96–06 Lazy Cache Implements Complete Cache
(Peter Ladkin)

96–07 Formal but Lively Buffers in TLA+
(Peter Ladkin)

96–08 The X-31 and A320 Warsaw Crashes: Whodunnit?
(Peter Ladkin)

96–09 Reasons and Causes
(Peter Ladkin)

96–10 Comments on Confusing Conversation at Cali
(Dafydd Gibbon und Peter Ladkin)

96–11 On Needing Models
(Peter Ladkin)

96–12 Formalism Helps in Describing Accidents
(Peter Ladkin)

96–13 Explaining Failure with Tense Logic
(Peter Ladkin)

96–14 Some Dubious Theses in the Tense Logic of Accidents
(Peter Ladkin)

96–15 A Note on a Note on a Lemma of Ladkin
(Peter Ladkin)

96–16 News and Comment on the AeroPeru B757 Accident
(Peter Ladkin)

97–01 Analysing the Cali Accident With a WB-Graph
(Peter Ladkin)

97–02 Divide-and-Conquer Multiple Sequence Alignment
(Jens Stoye)

97–03 A System for the content-based retrieval of textual and non-textual documents based on natural language queries
(Alois Knoll, Ingo Glöckner, Hermann Helbig und Sven Hartrumpf)