Improved Alignments Based on a Combination of Amino Acid and Nucleic Acid Sequence Information

DIPLOMARBEIT
zur Erlangung des akademischen Grades
Magister rerum naturalium

Vorgelegt der
Formal- und Naturwissenschaftlichen Fakultät
der Universität Wien

von
Roman Stocsits

am Institut für
Theoretische Chemie und Molekulare Strukturbiologie

im Mai 1999

Abstract

Many advanced techniques of sequence analysis are dependent upon the availablity of high quality multiple sequence alignments. A recent procedure for extracting conserved secondary structure elements from a moderate size sample of related RNA sequences is one example. For such a procedure a high quality alignment of nucleic acid sequences is essential.

In many cases, the sequences under consideration code for proteins. It is well known that amino acid sequences can be aligned much more reliably than their underlying genomic DNA or RNA sequences. In this work a procedure is described that utilizes the information contained in the amino acid sequences to construct an improved multiple alignment of the underlying nucleic acid sequences. This algorithm, which consists of a combination of amino acid and nucleic acid based partial alignments, is implemented in the program package `RALIGN`.

It is demonstrated that the `RALIGN` approach is indeed feasible and leads to significant improvements. In particular, the number of small gaps is reduced, and `RALIGN` guarantees that insertions and deletions at the nucleic acid level within a coding region match insertions and deletions at the level of protein sequences.

The program `RALIGN` is applied to finding conserved RNA secondary structures in the pregenomic RNA of human hepatitis B virus sequences. Apart from the well known ϵ-element at the very 5'-end of the sequence, we find a number of highly significant secondary structure elements that so far have not been described in the literature. A Y-shaped element located in the ENH enhancer region is of particular interest.

Zusammenfassung

Bei der Verwendung vieler Methoden im Bereich der Sequenzanalyse ist man angewiesen auf die Verfügbarkeit von multiplen Sequenzalignments höchster Qualität. Eine neue Methode zur Detektion von konservierten Sekundärstrukturelementen in einem Datensatz verwandter RNA-Sequenzen ist ein Beispiel. Für solch eine Anwendung ist ein Nukleinsäure-Alignment höchster Qualität absolut essentiell.

In vielen Fällen codieren die betrachteten Sequenzen für Proteine. Es ist eine bekannte Tatsache, dass Aminosäuresequenzen viel eher erfolgversprechend aligniert werden können als die entsprechenden genomischen DNA- oder RNA-Sequenzen. In dieser Arbeit wird eine Methode beschrieben, die die Information nutzt, die in Aminosäuresequenzen enthalten ist, um ein verbessertes multiples Alignment der entsprechenden Nukleinsäuresequenzen zu konstruieren. Dieser Algorithmus, der aus einer Kombination von Teilalignments auf der Ebene von Nukleinsäure- und Aminosäuresequenzalignments besteht, wurde implementiert in dem Programmpaket `RALIGN`.

Es konnte gezeigt werden, dass der `RALIGN`-Ansatz tatsächlich gut funktioniert und zu signifikanten Verbesserungen führt. Im speziellen wird die Anzahl der kurzen *gaps* reduziert, wobei durch `RALIGN` gewährleistet wird, dass Insertionen und Deletionen auf der Ebene von Nukleinsäuren innerhalb einer codierenden Region exakt den Insertionen und Deletionen auf der Ebene der jeweiligen Proteinsequenzen entsprechen.

Das Programm `RALIGN` wurde angewendet, um konservierte RNA-Sekundärstrukturelemente in der prägenomischen RNA von humanen Hepatitis B-Viren zu detektieren. Neben dem gut bekannten ϵ-Element am 5'-Ende der Sequenz konnten noch einige hochsignifikante Sekundärstrukturelemente gefunden werden, die bis jetzt in der Literatur nicht beschrieben worden sind. Interessant ist vor allem ein Y-förmiges Element, das in der ENH enhancer-Region liegt.

Danksagung

Ich danke Peter Stadler für die hervorragende und freundschaftliche Betreuung meiner Arbeit und die Möglichkeit, bei ihm an dieser Diplomarbeit zu arbeiten. Danke an Ivo Hofacker für viele sehr hilfreiche Ratschläge und Hinweise.
Ich danke Peter Schuster für die freundliche Aufnahme an seinem Institut.
Vielen Dank auch an Christoph Flamm für seine immerwährende Höflichkeit und Hilfsbereitschaft.
Danke an alle meine Freunde und Kollegen: Ronke Babajide, Jan Cupal, Daniela Dorigoni, Martin Fekete, Dagmar Friede, Kurt Grünberger, Christian Haslinger, Stefan Müller, Susi Rauscher, Andreas Svrcek-Seiler, Michael Kospach, Andreas Wernitznig, Günther Weberndorfer, Bärbel Stadler.
Zuletzt vielen Dank an meine Eltern, die mir dieses Studium ermöglicht haben.

Contents

1 Introduction

There is a huge amount of various data sets available in different sequence data banks like e.g. `GenBank`. However, often there is no satisfactory mechanism to process the data [82]. One of the essential tools in molecular biology is the simultaneous alignment of nucleotide or amino acid sequences. Multiple alignments are used to find diagnostic patterns to characterize protein families; to detect or demonstrate homology between new sequences and existing families of sequences; to help predict the secondary and tertiary structures of new sequences; to suggest oligonucleotide primers for PCR; and as an essential prelude to molecular evolutionary analysis [51]. The rate of appearance of new sequence data is steadily increasing and the development of efficient and accurate automatic methods for multiple sequence alignments is of major importance [2, 1].

Recent research in our group aims at finding conserved secondary structure elements that are part of the genomes of RNA viruses and the pregenomic RNA intermediates of some DNA viruses like the Hepatitis B viruses [58, 69]. For this reason predicted secondary structures of known genomic RNA sequences have to be compared on the basis of a reliable multiple sequence alignment. Almost all RNA molecules and consequently also almost all subsequences of a large RNA molecule form secondary structures. The development and implementation of computational methods capable of reliably predicting functional structural elements on the basis of sequence information will provide immense benefits in terms of our understanding of the relationship between sequence and structure [18, 11, 25, 49]. Such methods will also help greatly in tasks such as drug discovery or the study of molecular evolution [27]. They could be applied to huge quantities of sequence data at our disposal nowadays to discover important structural motifs and trends in various macromolecules, without measuring the 3D structure of each macromolecule which is very laborious and expensive [5, 10, 12, 57, 73, 72].

Secondary structures of ssRNA viruses or (as in the case of Hepatitis B virus) pregenomic RNA intermediates are known to play an important role in the regulation of the viral life cycle [31, 56]. See figure 1.

Since only functional secondary structures are likely to be conserved, a method that detects and highlights conserved structural elements based solely on already available sequence data could be used e.g. to guide experimental mutagenesis or deletion studies [29, 30, 50].

An important prerequisite of this method is the quality of the sequence alignment. However, sequence heterogenity on the level of nucleic acids makes good alignments often infeasible even for phylogenetically closely related sequences. In many cases one observes too many gaps. This is caused by the inherent redundancy of the genetic code: most amino acids have more than one codon on the level of nucleic acids. As a result it is possible that two different nucleic acid sequences code for the same protein sequence. In the extreme case all three codon positions are different. In a protein alignment these amino acids would match each other while the differences on the level of nucleic acids can produce gaps in a nucleic acid alignment. This specific problem leads to various gaps within coding regions where they are not really necessary, because the biologically important part of the system is protein in this region of the genome. On the level of protein alignments many of these gaps could have been avoided.

The purpose of this thesis is to improve the quality of sequence alignments of RNA viruses by designing and implementing a combined alignment algorithm. This program should detect possible coding regions in all input data sets and process them on amino acid level while non coding regions would be handled as nucleic acids. Finally the various alignments of the parts of an input sequence have to be combined.

The idea behind the combined amino acid and nucleic acid based align-

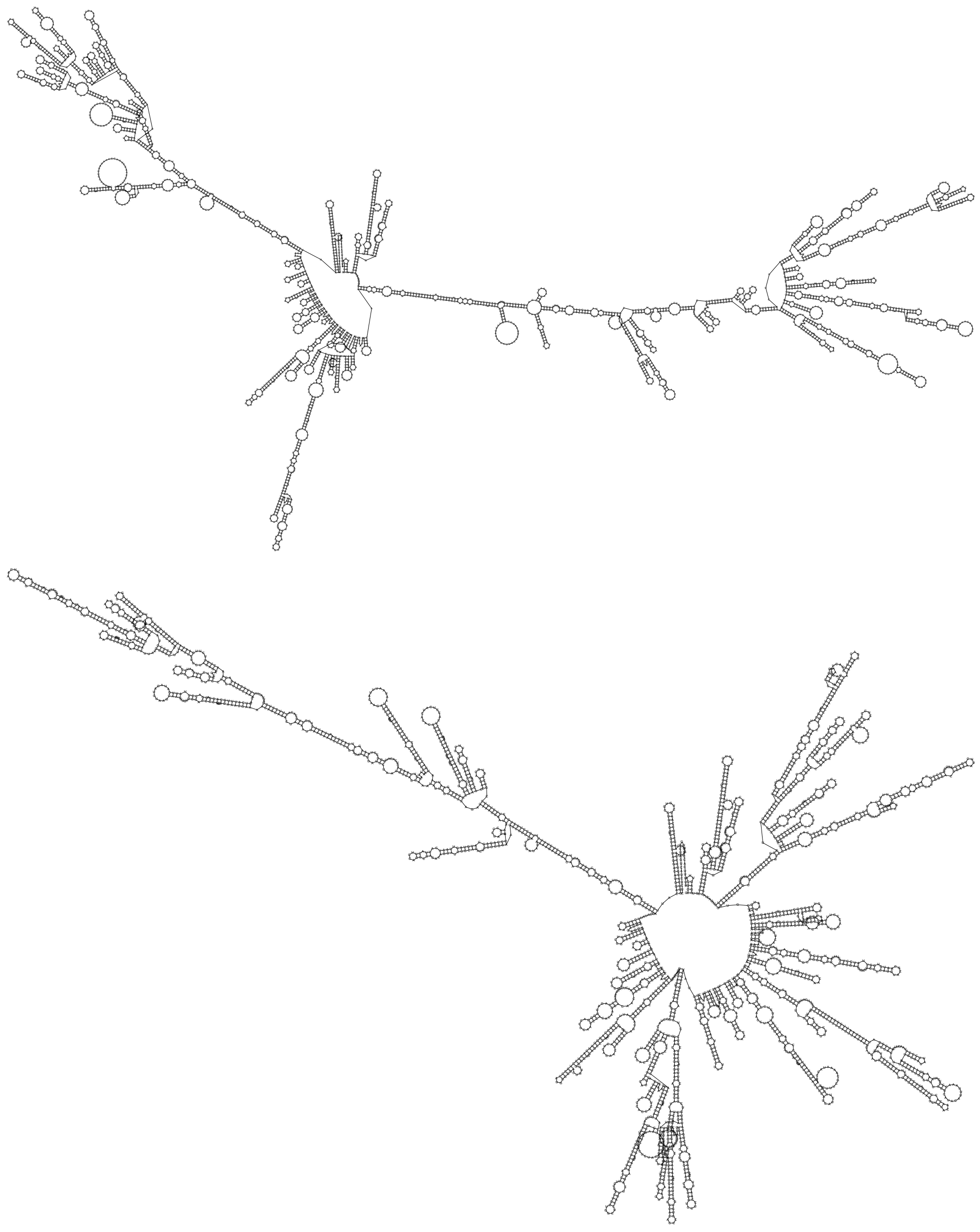


Figure 1: Two predicted secondary structures of the complete RNA pregenomes of human hepatitis B viruses (AB014360 and HBD50521). Note that there is hardly similarity between the predictions.

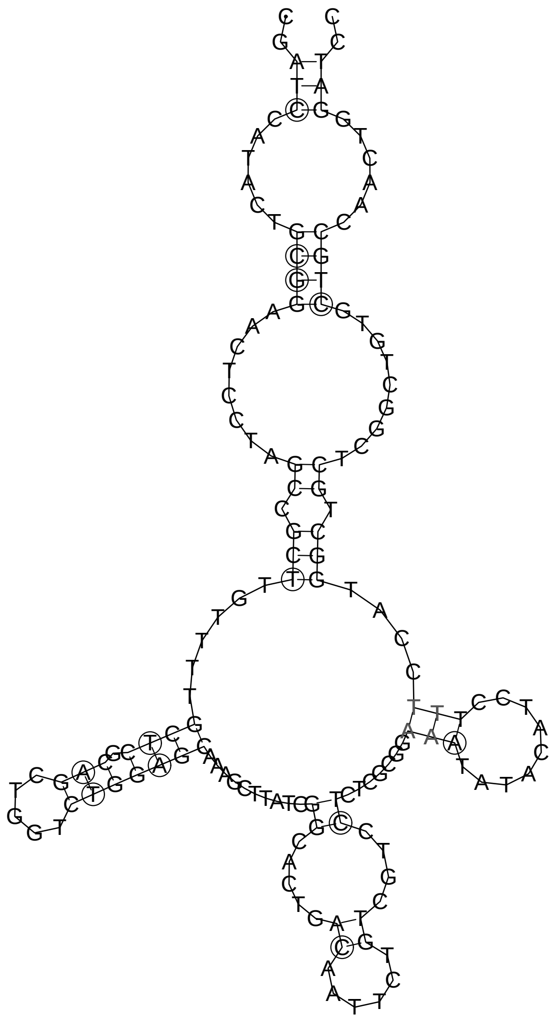


Figure 2: Example for a predicted secondary structure. It consists of the nucleotides 2794 to 2937 of mammalian hepatitis B virus.

ments is that coding regions on the level of protein vary less than on the level of nucleic acid, because most amino acids are coded by more than one codon (base triplet) and some different nucleic acid sequences can produce the same protein sequence after translation. Protein sequences can still show substantial homology when the corresponding nucleic acid sequences are already essentially randomized.

In the following some examples will be shown how it was made possible to improve alignments of viral genomes and to reduce the quantity of gaps using the combined amino acid and nucleic acid based alignment algorithm. Here we use HIV1 and hepatitis B viruses as examples.

The alignment results of Hepatitis B viruses were used then as an input for secondary structure prediction using the Vienna RNA package [41, 43]. The starting point of this approach is the list of all predicted base pairs. The multiple sequence alignment is very useful to establish which base pairs from different sequences correspond to each other. Then the individual base pairs are ranked by certain filtering procedures. The procedure is described in detail in [42, 40].

The algorithm proposed here has the following features: first the program detects all possible coding regions of a minimum size without user intervention. The next step is translation of the detected coding regions into amino acid sequences.

Corresponding open reading frames are identified and then aligned. While the program can operate fully automatically, manual alteration of the list of corresponding ORFs by the user is possible and often recommended.

The protein alignment results are then reverse translated back into nucleic acids and the non coding regions between the open reading frames are aligned as nucleic acids using the alignments of the open reading frames as constraints.

The last step consists of joining all reverse translated protein sequence alignments with the nucleic acid alignments of the non coding regions.

Secondary structure prediction was performed using the algorithms of the `Vienna RNA package` like `RNAfold` which calculates the minimum free energy structure and the partition function and base pairing probability matrix of an RNA sequence. Further used programs were `alidot` and `pfrali`. The programs `alidot` and `pfrali` detect conserved secondary structure elements in relatively small sets of RNAs by combining multiple sequence alignments and secondary structure predictions. While `alidot` works with single structures, its variant `pfrali` uses base pairing probability matrices. In both cases the best possible quality of the input alignment is of crucial importance. It will be shown that the alignments are indeed improved by the combined amino acid and nucleic acid based alignment algorithm and that the number of gaps is reduced significantly.

2 Theory

2.1 Background on Alignments

An alignment is the most basic sequence analysis task. It is used to tell whether two or more sequences are related and to give an impression how close relationship is in terms of sequence similarity. To find the best possible alignment of sequences is of central importance for bioinformatics and data processing after routine laboratory procedures like sequencing nucleic acids. Some alignment algorithms exist which are used to find an optimal alignment, and, of course, a scoring system is necessary to rank alignments. In principle all known algorithms are based on two criteria, (i) maximum similarity or (ii) minimum (Hamming-) distance [23, 28, 39].

For evaluating the difference between two sequences we have three possibilities of pairs of opposite symbols: (i) identity, (ii) substitution or mismatch and (iii) insertion or deletion. The procedure is usually done by first aligning the sequences and then deciding whether that alignment is more likely to have occurred because the sequences are related, or just by chance. In any case the scoring system should help to answer this question regarding to identical and similar positions in the alignment. (Similar pairs of residues in amino acid alignments are those which have a positive score in the substitution matrix used to score the alignment, e.g. aspartate-glutamate pairs, D-E, both negatively charged amino acids.)

2.2 Scoring Systems

Careful thought must be given to the scoring system used to evaluate an alignment when we are looking for evidence that they have diverged from a common ancestor by a process of mutation and selection. As mentioned above, the basic mutational processes that are considered are substitutions,

```
HBA_HUMAN   GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
            ++ ++++H+ KV   + +A  ++          +L+ L+++H+ K
LGB2_LUPLU  NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG
```

Figure 3: This figure shows a protein sequence alignment between a fragment of human alpha globin and leghaemoglobin from yellow lupin. Some identities are shown and some similar positions which have a positive score in the substitution matrix (indicated by '+'). This is a biologically meaningful alignment, in that we know that these two sequences are evolutionarily related.

which change residues in a sequence, and insertions and deletions, which add or remove residues and are together referred to as 'gaps'. The total score we assign to an alignment is a sum of terms for each aligned pair of residues, plus terms for each gap. Informally, using an additive scoring system we expect identities and conservative substitutions to be more likely in good (biologically relevant) alignments than we expect by chance, and so they should contribute positive score terms. And on the other hand non conservative changes are expected to be observed less frequently in real alignments than we expect by chance, and so these contribute negative score terms. This system also corresponds to the assumption that we can consider mutations at different sites in a sequence to have occurred independently (treating a gap of arbitrary length as a single event). All alignment algorithms depend crucially on such a scoring scheme and from a biological point of view the assumption of independence appears to be a reasonable approximation for DNA and protein sequences, although we know that intramolecular interactions between residues of a protein play a very important role in determining protein structure. Regarding the secondary structures of RNAs, where base pairing introduces very critical long range dependencies, the model of independent mutations is biologically inaccurate [46, 52, 54].

As mentioned above we need score terms for each aligned residue (or base) pair. We derive substitution scores from a probabilistic model that gives a measure of the relative likelyhood that the sequences are related as opposed to being unrelated. We do this by having models that assign a probability to the alignment in each of the two cases. Then we consider the ratio of the two probabilities. The *random* model R assumes that a letter in the sequence (for proteins an amino acid or one of the four bases in the case of DNA or RNA) occurs independently with some frequency q, and hence the probability of the two sequences is the product of the probabilities of each amino acid (or base):

$$P(x,y|R) = \prod_i q_{x_i} \prod_j q_{y_j} \tag{1}$$

where x and y is a pair of sequences, x_i is the ith symbol in x and y_j is the jth symbol in y. These symbols come from an alphabet (A, G, C, T, U in the case of nucleic acids or an amino acid in the case of protein). In the alternative *match* model M, aligned pairs of residues occur with a joint probability p_{ab}. This value p_{ab} can be thought of as the probability that the residues a and b have each independently been derived from some unknown original residue c in their common ancestor (c might be the same as a and/or b). This gives a probability for the whole alignment:

$$P(x,y|M) = \prod_i p_{x_i y_i} \tag{2}$$

The ratio of these two likelihoods is the odds ratio:

$$\frac{P(x,y|M)}{P(x,y|R)} = \frac{\prod_i p_{x_i y_i}}{\prod_i q_{x_i} \prod_i q_{y_i}} = \prod_i \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}} \tag{3}$$

We want to arrive at an additive scoring system, so we have to take the logarithm of this ratio, known as the log-odds ratio:

$$S = \sum_i s(x_i, y_i) \tag{4}$$

where

$$s(a,b) = \log(\frac{p_{ab}}{q_a q_b}) \tag{5}$$

is the log likelihood ratio of the residue pair(a,b) occurring as an really valid aligned pair, as opposed to an unaligned pair (or by chance joined pair of residues or nucleic acids). We can see that S in this equation is a sum of individual scores $s(a,b)$ for each aligned pair of residues. And these individual scores, these log-odds values can be rounded to the nearest integer for purposes of computational efficiency and then arranged in a matrix. For instance, in the case of proteins the matrix is a 20×20 matrix which gives an individual score $s(a_i, b_j)$ for sequences a and b in position i and j. The highest positive entries in the matrix are given for identical residue pairs, lower, but also positive, values do the conservative substitutions have while non conservative substitutions give a negative score. So it is possible to derive scores, in fact $s(a,b)$ in the above equation, for every pair of residues in the alignment. Any matrix like this is making a statement about the probability of observing ab pairs in real (biologically relevant) alignments and is called substitution matrix or score matrix or weight matrix. Examples of substitution matrices are the `BLOSUM50`, the `BLOSUM62` [36] or the `PAM250` matrix [20].

The next point is penalising gaps. There are two possibilities: the standard cost associated with a gap of length g could be given by a linear score

$$\gamma(g) = -gd \tag{6}$$

where d is called the gap open penalty. But it seems to be more legitimate to make a difference whether a gap is newly opened or an existing gap is just extended. A type of score could be used which is known as the affine score

$$\gamma(g) = -d - (g-1)e \tag{7}$$

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5

Figure 4: The BLOSUM50 substitution matrix. The log-odds values have been scaled and rounded to the nearest integer for purposes of computational efficiency.

where e is called the gap extension penalty. This penalty should be set to something less than the gap open penalty d, so that extension of existing insertions (or deletions) is penalised less than opening further gaps (as it would be by the linear gap cost). Gap penalties also correspond to a probabilistic model of alignment. We assume that the probability of a gap occurring at a particular site in a given sequence is the product of a function $f(g)$ of the length of the gap, and the combined probability of the set of inserted residues,

$$P(\text{gap}) = f(g) \prod_{i \in \text{gap}} q_{x_i}. \tag{8}$$

The form of this equation as a product of $f(g)$ with the q_{x_i} terms corresponds to an assumption that the length of the gap is not correlated to the residues it contains. The natural values for the q_a probabilities here are the same as those used in the random model above, because they both correspond to unmatched independent residues. When we divide by the probability of this region according to the random model to form the odds ratio, the q_{x_i} terms cancel out. This leaves us with a term dependent on length $\gamma(g) = log(f(g))$. Gap penalties correspond to the log probability of a gap of that length.

But on the other hand, if there is evidence for a different distribution of residues in gap regions then there should be residue-specific scores for the unaligned residues in gap regions, equal to the logs of the ratio of their frequencies in gapped versus aligned regions. This might happen if it is expected that polar amino acids are more likely to occur in gaps in protein alignments than indicated by their average frequency in protein sequences, because the gaps are more likely to be in loops on the surface of the protein structure than in the buried core.

After having determined a certain scoring system we need to have an algorithm for finding an optimal alignment for a pair of sequences. Without using gaps there is only one possible global alignment for two sequences when both have the same length n. But the alignment becomes much more complicated once gaps are allowed. We have

$$\binom{2n}{n} = \frac{(2n)!}{(n!)^2} \simeq \frac{2^{2n}}{\sqrt{2\pi n}} \tag{9}$$

possible global alignments between two sequences of length n. The quantity of possible alignment solutions grows by about 4^n This means for sequences of length 30 there are 10^9 possibilities, and with length 60 we have 10^{18} possible alignments. But in terms of molecular biology sequences of length 30 or even 60 are comparatively short and often it is necessary to find the best alignment between sequences which have a length of a few thousand amino acids or nucleotides (like in the case of virus genomes). It is of course not computationally feasible to enumerate all these, even for moderate values of n.

So we need to find a way which gives us the possibility to gain optimal alignments without testing and valueing every possible solution. The algorithms for finding optimal alignments given an additive alignment score of the type described above is called dynamic programming. Dynamic programming algorithms are of central importance for computational sequence analysis. They imply that we get the best possible alignment between two sequences as a result of an optimal alignment till each current position. Using the introduced scoring scheme as a log-odds ratio, better alignments have higher scores. So what we have to is to maximise the score to find the optimal alignment as opposed to other interpretations of scoring which search for minimal distances or costs. Both approaches have been used in the biological sequence comparison literature. Dynamic programming algorithms apply to either case. The differences are, simply said, just exchanges of 'min' for 'max' [61, 63, 64].

2.3 The Needleman-Wunsch Algorithm

The most important dynamic programming algorithm in biological sequence analysis for obtaining the optimal global alignment between two sequences, allowing gaps, is the Needleman-Wunsch algorithm [63, 8], introduced in 1970.

The idea behind all versions is to build up an optimal alignment using previous solutions for optimal alignments of smaller subsequences. A matrix F of the two sequences is constructed, indexed by i and j, one index for each sequence, where the value $F(i,j)$ is the score of the best alignment between the initial segment $x_{1\dots i}$ of x up to x_i and the initial segment $y_{1\dots j}$ of y up to y_j. The score value $F(i,j)$ is builded recursively and we start by initialising $F(0,0) = 0$.

As mentioned above we have three possibilities of pairs of opposite symbols: (i) identity, (ii) substitution or 'mismatch' and (iii) insertion or deletion. So we proceed to fill the matrix from top left to bottom right, from the first letters of the sequences to their ends. Along the top horizontal row (where $j = 0$) and the first vertical column (where $i = 0$) we write the pairs of one sequence's letters with a gap in the second sequence and get the scores of these gaps by multiplying the position of the letter by the gap penalty. So the values $F(i,0)$ represent alignments of a prefix of x to all gaps in y and we can define $F(i,0) = -id$. Likewise down the left column $F(0,j) = -jd$. If $F(i-1,j-1), F(i-1,j)$ and $F(i,j-1)$ are known, it is possible to calculate $F(i,j)$. There are three possible ways that the best score $F(i,j)$ of an alignment up to x_i, y_j could be obtained: x_i could be aligned to y_j, in which case $F(i,j) = F(i-1,j-1) + s(x_i,y_j)$, where $s(x_i,y_j)$ is the individual score for this pair of amino acids or nucleotides; or x_i is aligned to a gap, in which case $F(i,j) = F(i-1,j) - d$, where d is the gap penalty; or y_j is aligned to a gap, in which case $F(i,j) = F(i,j-1) - d$. The best score up to (i,j) is the largest of these three options. Therefore, we have

$$F(i,j) = \sup \left\{ \begin{array}{c} F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{array} \right\} \qquad (10)$$

This equation is applied repeatedly to fill in the matrix of $F(i,j)$ values, calculating the value in the bottom right-hand corner of each square of four cells from one of the other three values (above left, left, or above). And as we fill in the $F(i,j)$ values, we also keep a pointer in each cell back to the cell from which its $F(i,j)$ was derived. Finally the value in the bottom right cell of the matrix $F(n,m)$ is by definition the best score for an alignment of $x_{1...n}$ to $y_{1...m}$. To gain the alignment itself, we must find the path of choices which led to this final value. The procedure for doing this is called backtracking. We build the alignment in reverse, starting from the final cell, and following the pointers that we stored when building the matrix. At each step in the backtracking process we go back from the current cell (i,j) to the one of the cells $(i-1,j-1),(i-1,j)$ or $(i,j-1)$ from which the value $F(i,j)$ was derived. So with every step we get a pair of symbols and add it to the growing alignment: x_i and y_j if the step was to $(i-1,j-1)$, x_i and the gap character '-' if the step was to $(i-1,j)$, or '-' and y_j if the step was to $(i,j-1)$. Finally we reach the starting point of the matrix, $i=j=0$. The reason that the algorithm works is that the score is made of a sum of independent pieces, so the best score up to some point in the alignment is the best score up to the point one step before, plus the incremental score of the new step.

2.4 Multiple Alignments

As described above it is practice to use dynamic programming in order to align just two sequences. This guarantees a mathematically optimal alignment, given a table of scores for matches and mismatches between all amino acids or nucleotides and penalties for insertions or deletions of different lengths. But attempts at generalising dynamic programming to multiple alignments are limited to small numbers of short sequences [55]. For much more than ten or so proteins of average length, the problem is infeasible given current computer power. Therefore, all of the methods capable of handling larger problems in practical timescales make use of heuristics. Nowadays, the most widely used approach is to exploit the fact that homologous sequences are evolutionary related. We can produce a multiple alignment progressively by a series of pairwise alignments, following the branching order in a phylogenetic tree [22]. We first align all possible pairs of sequences and derive a distance matrix in order to calculate the initial guide tree which is built up by the distances between the sequences. Then the most closely related sequences get aligned progressively according to the branching order in the guide tree, gradually adding in the more distant ones when we already have some information about the most basic mismatches or gaps. Some information which is derived from the first pairwise alignments of the most closely related sequences.

This approach is fast enough to allow alignments of virtually any size. Further, in most (simple) cases, the quality of the alignments is very good, as judged by the ability to correctly align corresponding domains from sequences of known secondary or tertiary structures [6]. So this approach also works well if the data sets consist of sequences of different degrees of divergence. By the time the most distantly related sequences are aligned, one already has a sample of aligned sequences which gives important information about the variability at each position. The placement of gaps in alignments between closely related sequences is much more accurate than between distantly related ones. Therefore, the positions of the gaps which were introduced during the early alignments of the closely related sequences are not changed as new

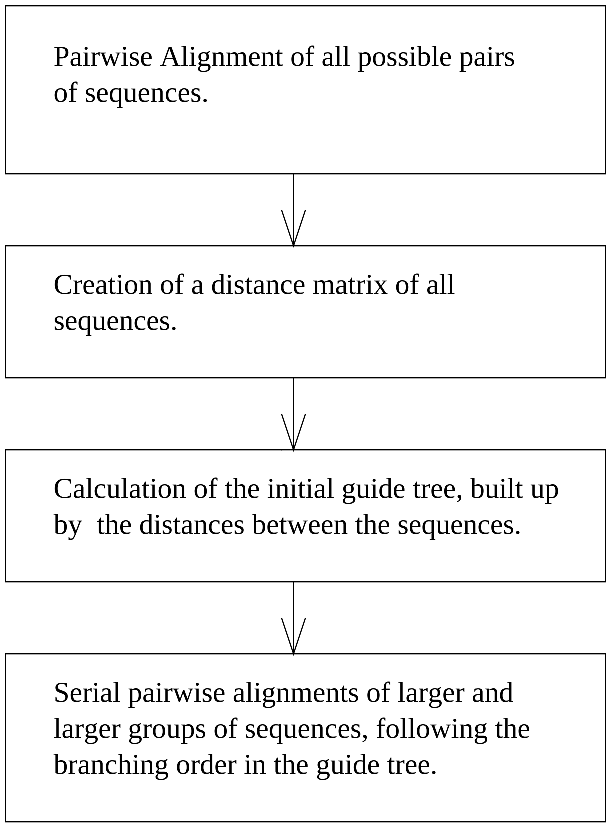

Pairwise Alignment of all possible pairs of sequences.
Creation of a distance matrix of all sequences.
Calculation of the initial guide tree, built up by the distances between the sequences.
Serial pairwise alignments of larger and larger groups of sequences, following the branching order in the guide tree.

Figure 5: A flow chart representing the main steps of `CLUSTAL W`. See the text for details.

```
B-YU2           --------------CUAGCUAUAGGUUGAUAAGUUGUAACACCUCAGUCAUUACACAGGC
B-896           --------------CUAAGUAUAGGUUAAUAAGUUGUAACACCUCAGUCAUUACACAGGC
B-JRCSF         --------------CCAAAUAUAGGUUAAUAAGUUGUAACACCUCAGUCAUUACACAAGC
B-WEAU          ------------------CUAUACGUUGAUAAAUUGUAAAUCCUCAACCAUUACACAGGC
D-ELI           -----AUAG---UACCAAUUAUAGGUUAAUAAAUUGUAAUACCUCAGCCAUUACACAGGC
D-NDK           -----AUAG---UACUAAUUAUAGGUUAAUAAAUUGUGAUACCUCAACCAUUACACAGGC
ADI-MAL         ---------------UAGUUAUAGGCUAAUAAAUUGUAAUACCUCAGUAAUUACACAGGC
SIVCPZGAB       ------------------AUAUAGGAUAAUUAAUUGCAAUACUACAGCCAUAACACAAGC
O-ANT70         -------------AAAAUGUAUACAUUAACUAAUUGUAACUCCACAACCAUCACGCAAGC
O-MVP5180       -------------ACAACAUAUAUGUUAACUAAUUGUAACUCCACAAUUAUCAAGCAGGC
                                   * **   * *    ***  *     **   *  *  ** **
```

Figure 6: This is an example for a multiple nucleic acid sequence alignment. On the left side some gaps are shown. The last line is the consensus line which indicates identities.

sequences are added. One problem is that this approach becomes less reliable if all of the sequences are highly divergent. More specifically, any mistakes like misaligned regions made early in the alignment process cannot be corrected later as new information from other sequences is added. Thus, there is no guarantee that the global optimal solution has been found and the alignment is not captured in a local minimum. This risk increases with the divergence of the initially aligned sequences and is thought of as mainly resulting from an incorrect branching order in the initial tree. As mentioned above initial trees are derived from a matrix of distances between the separately aligned pairs of sequences in the first steps of the multiple alignment process. Most relevant errors occur during these initial alignments.

Furthermore, the parameter choice problem is very important. One chooses a weight matrix and two gap penalties (one for opening a new gap and one for extension of an existing gap) and expects that these should work well over all parts of all the sequences in the data set. When the sequences are closely related this works in most cases. But this problem increases as the sequences diverge. All residue weight matrices give most weight to identities. If identities dominate an alignment, almost any weight matrix will find approximately the correct solution. With very divergent sequences the scores given to non-identical residues will become critically important, because there are more mismatches than identities.

Another problem arises with the choice of the best gap penalties. The range of gap penalty values which will find the correct or best possible solution can be very broad for highly similar sequences, but as more and more divergent sequences are used, the exact values of the gap penalties become very important for success [78]. Further, in protein alignments, gaps do not occur randomly. They occur far more often between the major secondary structural elements like helices than within [68].

2.5 CLUSTAL W in Some Details

A widely used program is `CLUSTAL W` [74]. `CLUSTAL W` addresses the alignment parameter choice problem and dynamically varies the gap penalties in a position- and residue-specific manner. The observed relative frequencies of gaps adjacent to each of the 20 amino acids are used to locally adjust the gap opening penalty after each residue. Short stretches of hydrophilic residues usually indicate loop or random coil regions and the gap opening penalties are locally reduced in these stretches. In addition, the locations of the gaps found in the early alignments are also given reduced gap opening penalties. And because it has been observed that gaps in alignments between sequences of known structure tend not to be closer than roughly eight residues on average, `CLUSTAL W` also increases the gap opening penalty within eight residues of an existing gap. The two main series of amino acid weight matrices used today are the `PAM` series [36] and the `BLOSUM` series [20]. In each case there is a range of matrices to choose from. Some matrices are appropriate for aligning very closely related sequences where most weight by far is given to identities, with only the most frequent conservative substitutions receiving high scores. Other matrices work better at higher evolutionary distances where less importance is attached to identities. As the alignment proceeds, `CLUSTAL W` chooses different weight matrices depending on the estimated divergence of the sequences to be aligned at each stage.

Besides, sequences are weighted by `CLUSTAL W` to correct for unequal sampling across all evolutionary distances in the data set [77, 78]. This down-weights sequences which are very similar to other sequences in the data set and up-weights the most divergent ones. The weights are calculated directly from the branch lengths in the initial guide tree [75, 74]. In `CLUSTAL W` the initial guide tree used to guide the multiple alignment, is calculated using the Neighbour-Joining method [71] which is quite robust against the effects of unequal evolutionary rates in different lineages and gives good estimates of individual branch lengths. These branch lengths are used to derive the sequence weights. And finally it is possible for the user to choose between fast approximate alignments [7] or full dynamic programming for the distance calculations used to make the guide tree.

As mentioned above the first step of the basic multiple alignment algorithm is aligning separately all pairs of sequences in order to calculate a distance matrix giving the divergence of each pair of sequences. Accurate scores for constructing the best pairwise alignments are derived from full dynamic programming alignments using two gap penalties (for opening or extending gaps) and a full amino acid weight matrix. The relevant distance scores for the matrix are then calculated as the number of identities in the best alignment divided by the number of residues compared (gap positions are excluded). These scores are then initially calculated as per cent identity scores and are converted to distances by dividing by 100 and substracting from 1.0 to give number of differences per site.

The trees used to guide the final multiple alignment process are calculated from the distance matrix derived in the first step. This produces unrooted trees with branch lengths proportional to the estimated divergence. Then the root of one tree is established at a position where the means of the branch lengths on either side of the root are equal. These trees are then also used to derive a weight for each sequence.

The weights are dependent upon the distance from the root of the tree but sequences which have a common branch with other sequences share the weight derived from the shared branch. For example, a sequence's weight consists of the distance for its own branch plus half the length of the branch shared with its neighbour in the tree plus one third of the length shared with its second, less homologous neighbour, and so on. This sums a total weight for each sequence. Then the weights are normalised in such way that the biggest one is set to 1.0 and the rest are all less than 1.0. Groups of closely related sequences receive smaller weights because they contain much duplicated information. Highly divergent sequences without any close relatives receive high weights.

Then the progressive alignments start. The basic procedure at this stage is to use a series of pairwise alignments to align larger and larger groups of sequences, following the branching order in the guide tree. First the most similar sequences at the tips of the tree get aligned. Then this alignment gets aligned with the third most similar sequence which is something more divergent from the first two sequences and so on till the last sequence with the least homology gets aligned with an alignment consisting of all other sequences. At each stage a full dynamic programming algorithm [62] is used with a residue weight matrix and penalties for opening and extending gaps. So each step consists of an alignment of two existing alignments or sequences. Gaps that are present in former alignments remain fixed, but new gaps that are introduced at each state of alignment get full opening and extending penalties even if they are introduced inside old gap positions. The score

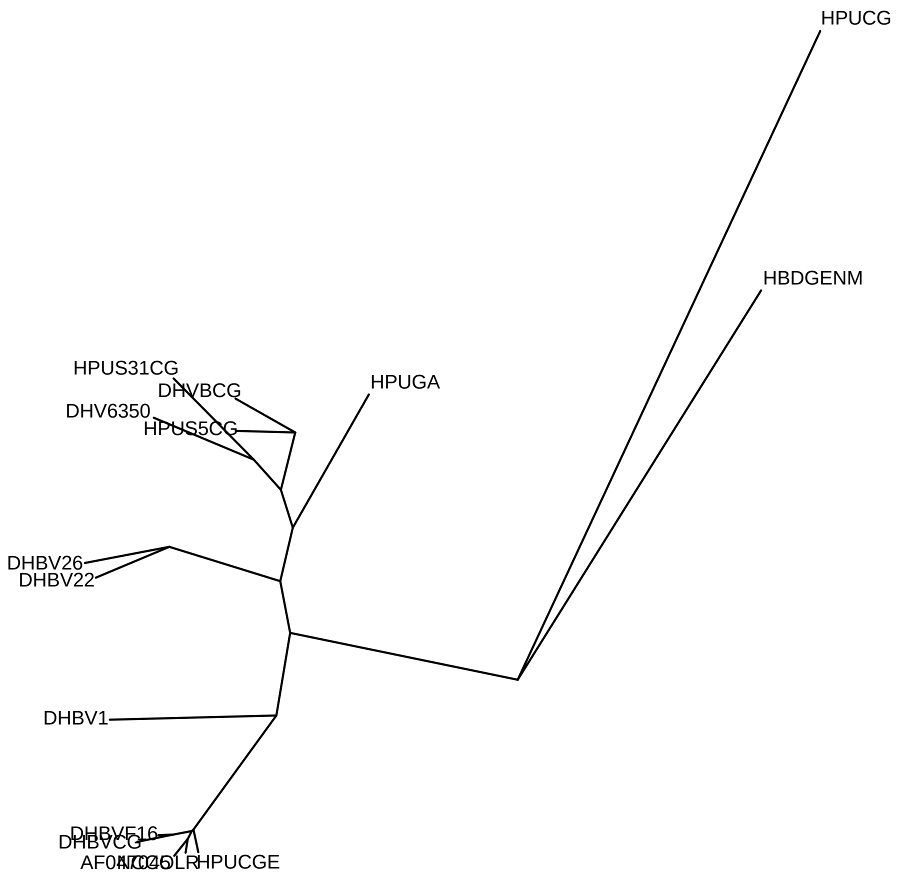


Figure 7: A distance matrix derived guide tree for avian hepatitis B viruses. The branch lengths are proportional to the estimated divergences.

between a position from one sequence or alignment and one from another is calculated as the average of all pairwise amino acid weight (substitution) matrix scores in the two sets. For example, aligning two alignments with 2 and 4 sequences the final score for this position is the average of 8 (2x4) comparisons. If one of the sequences contains a gap at this specific position, the gap versus a residue is scored as zero. Besides, the default amino acid weight matrices used in this context are rescored to have only positive values. Thus, this treatment of gaps treats the score of a residue versus a gap as having the worst possible score. And also, because sequences are weighted to correct for unequal sampling across all evolutionary distances in the data set, as mentioned before, each weight (substitution) matrix value for the residues is multiplied by the weights of the two sequences.

Two different gap penalties are used: a gap opening penalty, which gives the cost of opening a new gap of any length, and a gap extension penalty, which gives the cost of every item in the gap. `CLUSTAL W` varies gap penalties used with different weight (substitution) matrices to improve the accuracy of the sequence alignments. Further, the per cent identity of the two (groups of) sequences to be aligned is used to increase the gap opening penalty for closely related sequences and to decrease it for more divergent sequences. Also, if there are already gaps at a position, then the gap opening penalty is reduced in proportion to the number of sequences with a gap at this position and the gap extension penalty is lowered by a half. If a position does not have any gaps but is within 8 residues of an existing gap, the gap open penalty is increased. In the case of proteins the gap open penalty is reduced by one third in hydrophilic stretches.

Finally, `CLUSTAL W` offers two main series of weight matrices to the user: the `Dayhoff PAM` series [20] and the `BLOSUM` series [36]. In each case there is a choice of matrices ranging from strict ones, useful for comparing very closely related sequences, to less strict ones which are useful for aligning more divergent sequences. Depending on the distances between the two sequences or groups of sequences to be compared, `CLUSTAL W` switches between 4 different matrices in each series. The distances are measured directly from the guide tree.

2.6 RNA Secondary Structure Prediction

RNA polymers are macromolecules, consisting of a linear arrangement of building blocks, the monomers [32]. RNA polymers have the ability to fold back on themselves, due to interactions between individual base pairs. For biopolymers like proteins or RNA these interactions are specific, and can lead to the adaption of a unique compact conformation called 'native state'. During the structure formation process both, RNA and proteins, try to minimize the solvent exposure of hydrophobic residues by burying these residues in the interior of the structure [37]. But it is self-evident from the different chemical nature of RNA and proteins that the ways how these macromolecules achieve their compact conformation is different. For proteins the driving force of the collapse into compact conformations is the formation of a hydrophobic core. For RNA the formation of compact conformation is promoted by the tendency to maximize the stacking interaction between base pairs [38]. And it is essential for living cells that this formation of the correct and functional conformation is achieved in biologically relevant sufficiently short time [80, 79, 81, 33].

From a theoretical point of view, the problem of how biopolymers achieve their native state splits up into two aspects. The first aspect is the structure prediction problem. The second aspect deals with the dynamics of the folding process itself [13, 14, 35].

Since the sequence of a biopolymer specifies its three-dimensional structure, it should be possible, at least in principle, to predict its native structure solely from the knowledge of the sequence [45]. And in fact, as is becoming increasingly clear, biopolymers like proteins or RNA are flexible and rapidly fluctuating molecules whose structural mobilities have functional significance [15, 59].

The native states of RNA consist of a large ensemble of closely related and rapidly inter-converting conformational sub-states of nearly equal stabilities. Theoretical methods for structure prediction require extensive computation. The secondary structure of RNA is defined as the pattern of base pairs, which is formed by hydrogen bonds between atoms of the four bases [67, 66].

For RNA folding powerful algorithms [65, 83] based on the method of dynamic programming [8] and experimentally measured energy parameters have been developed [26, 34, 49, 76].

`RNAfold` as part of the `Vienna RNA package` reads RNA sequences from `stdin` and calculates their minimum free energy structure, partition function and base pairing probability matrix [41, 60]. It returns the minimum free energy structure in bracket notation, its energy, the free energy of the thermodynamic ensemble and the frequency of the minimum free energy structure in the ensemble to `stdout`. It also produces PostScript files with plots of the resulting secondary structure graph and a "dot plot" of the base pairing matrix. The dot plot shows a matrix of squares with area proportional to the pairing probability in the upper half, and one square for each pair in the minimum free energy structure in the lower half. The results of `RNAfold` are used as an input for `alidot` and `pfrali` [43, 44, 40].

2.7 Detection of Conserved RNA Structures

The programs `alidot` and `pfrali` detect conserved secondary structure elements in relatively small sets of RNAs by combining multiple sequence alignments and secondary structure predictions. Both a (good) sequence alignment and secondary structure predictions for each sequence in the alignment must be provided as inputs. While `alidot` works with single structures, its variant `pfrali` uses base pairing probability matrices.

The starting point of the analysis of conserved secondary structure elements is a list of all predicted base pairs. This list will in general not be a valid secondary structure, because it is possible that one certain nucleotide takes part in more than one base pairs and it is possible that base pairs cross.

The quality of the input sequence alignment is of crucial importance. So the approach of the combined amino acid and nucleic acid based alignments should provide one with better sequence alignments with less gaps.

The basic idea behind both `alidot` and `pfrali` is to sort the individual base pairs by their *credibility* and to reduce the number of entries in the list by subsequent filtering steps until only those secondary structure elements are left that are consistently predicted. Of course the sorting procedure is very important. For each predicted base pair the nucleotides occurring in the corresponding positions in the sequence alignment are stored. A sequence is *non-compatible* with a base pair *(i.j)* if the two nucleotides at positions i and j would form a non-standard base pair such as **GA** or **UU**. A sequence is *compatible* with base pair *(i.j)* if the two nucleotides form either one of the following six combinations: **GC**, **CG**, **AU**, **UA**, **GU**, **UG**.

When different standard combinations are found for a particular base pair *(i.j)* we may speak of *consistent* mutations. If we find combinations where both positions are mutated at once we have *compensatory* mutations. The occurrance of consistent and, in particular, compensatory mutations strongly supports a predicted base pair, at least in the absence of non-consistent mutations.

We call a base pair *(i.j) symmetric* if j is the most frequently predicted pairing partner of i and if i is the most frequently predicted pairing partner of j. Foe each sequence position i there is at most one symmetric base pair involving i.

In the first step of `alidot` and `pfrali`, a list of 'believable base pairs' is extracted from the set of all pairs which are contained in the input. In the first processing step of `alidot`, all but the most frequent pair *(i.j)* for each base i is removed. The remaining list is then sorted according to some hierarchical criteria like: (i) the more sequences are non-compatible with *(i.j)*, the less credible is the base pair, (ii) symmetric base pairs are more credible than other base pairs, (iii) a base pair with more consistent mutations is more credible, (iv) base pairs are more credible with smaller values of a certain pseudo-entropy which is derived from the frequencies f_{ij} with which *(i.j)* is predicted in the sample of sequences and which is a measure for the reliability. In contrast, `pfrali` does not prune the list in a preprocessing step and uses a two-step mechanism for sorting the list of base pairs: (i) the more sequences are non-compatible with *(i.j)*, the less credible is the base pair, (ii) if the number of non-compatible sequences is the same, then the pairs are ranked by the product of the mean probability and the number of different pairing combinations.

The next step is common to both programs: the sorted list is reduced by running through it and removing all base pairs that cross with higher ranking ones and hence would not yield a valid secondary structure.

The resulting secondary structure will, in general, still contain ill-supported base pairs. These are removed by three subsequent filtering steps. First all pairs are removed that have more than two non-compatible sequences, as well as pairs with two non-compatible sequences adjacent to a pair that also has non-compatible sequences.

Next all isolated base pairs are omitted. The remaining pairs are collected into helices. Only those helices are retained that satisfy the following conditions: (i) the highest ranking base pair must not have non-compatible sequences, (ii) for the highest ranking base pair the product of the mean

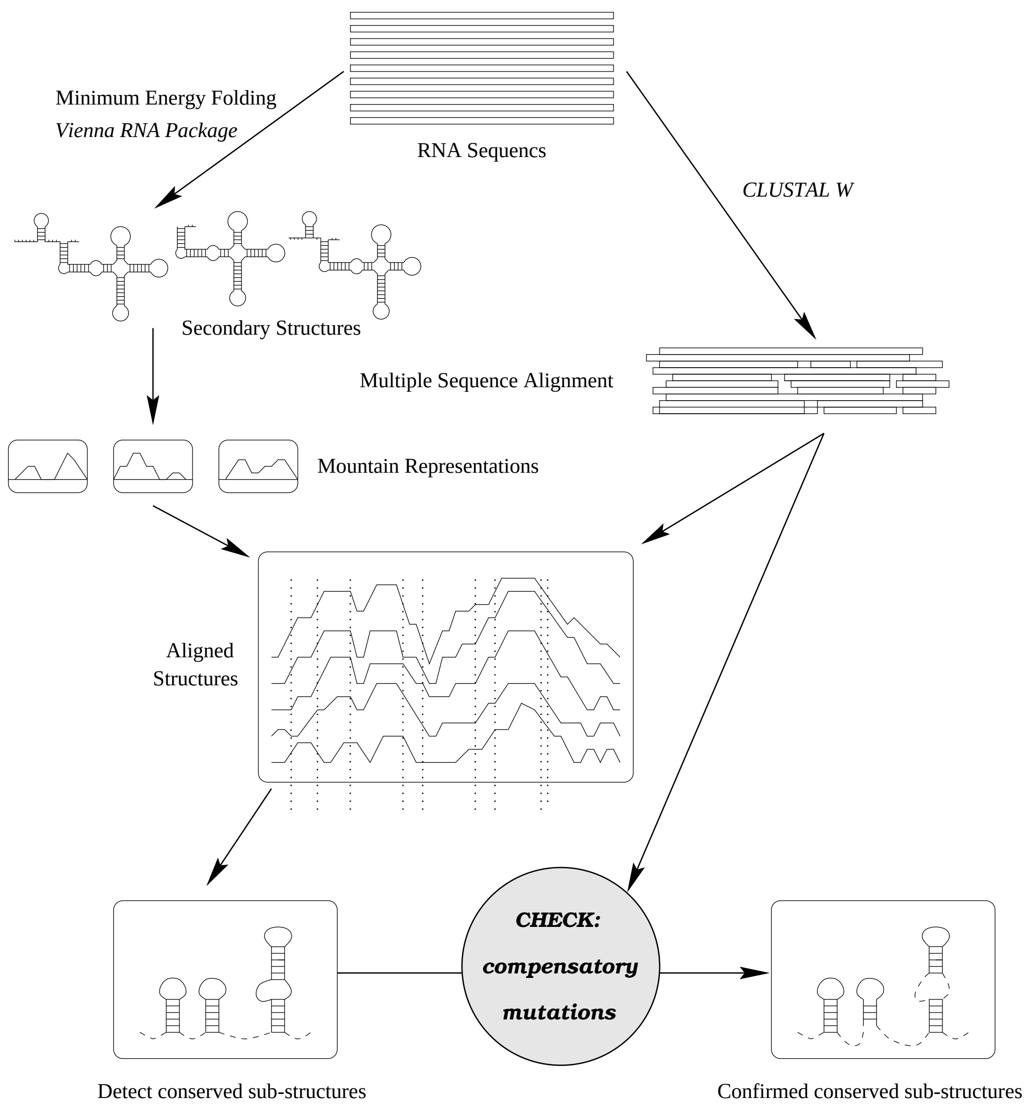


Figure 8: Scheme of the secondary structure analysis of viral genomes. Sequences are aligned using a standard multiple alignment procedure. Secondary structures for each sequence are predicted and gaps are inserted based on the sequence alignment. The resulting aligned structures can be represented as aligned mountain plots. From the aligned structures consistently predicted base pairs are identified. The alignment is used to identify compensatory mutations that support base pairs and inconsistent mutants that contradict pairs. This information is used to rank proposed base pairs by their credibility and to filter the original list of predicted pairs.

probability and the number of different pairing combinations must be greater than 0.3, (iii) if the helix has length 2, it must not have more non-compatible sequences than consistent mutations.

The search algorithms for detecting conserved RNA structure elements are based on both a multiple sequence alignment and predicted secondary structures. The multiple sequence alignment is contained in a single file in `CLUSTAL W` format. The predicted secondary structures are contained in individual files (e.g minimum free energy files or base pairing probabilities produced by RNAfold as part of the `Vienna RNA package`).

Both programs produce both text and `PostScript` output. The text output contains some statistics, all base pairing data and the conserved structure in bracket notation. The `PostScript` output are dot plots, a simple graphical representation of structures, where each base pair corresponds to a small sqare in a matrix of size sequence length $\times$ sequence length. The size and color of this 'dot' are used to encode additional information like the frequency of prediction and the number of different consistent base pairs.

3 The RALIGN Project

3.1 Difficulties of Nucleic Acid Alignments

Alignments of nucleic acid sequences can bear one main problem: the sequence heterogenity on the level of nucleic acid makes good alignments often impossible. The resulting alignments contain too many gaps although the sequences should be very similar regarding their high degree of relationship. While protein sequences can still show substantial homology, the corresponding nucleic acid sequences are already essentially randomized. This is caused by the inherent redundancy of the genetic code: most amino acids have more than one codon on the level of nucleic acid. As a result it is possible that two different nucleic acid sequences code for the same protein sequence. In a protein alignment these amino acids would match each other while the differences on the level of nucleic acids can produce gaps in a nucleic acid alignment. This specific problem leads to various gaps within coding regions where they are not really necessary, because the biologically important part of the system is protein at this region of the genome. Furthermore, on the level of protein alignments many of this gaps could have been avoided.

Therefore, in most cases it is possible to obtain better alignments on the level of protein than on the level of nucleic acids. The scores (the per cent homologies) are higher and the number of gaps within the protein sequences is not as high as it would be in the case of nucleic acids. Reducing the gaps within an alignment improves the resulting alignment which may be used as input into other sequence data processing programs like those for secondary structure prediction.

```
SARGLSSTVSLGQFEHWSPR
+AR+LS+TVSL+QF+H SPR
NARNLSDTVSLSQFDHPSPR

AGTGCAAGAGGATTAAGTAGTACAGTAAGTTTAGGACAATTTGAACATTGGAGTCCAAGA
   GC  G G   T       AC G      T    CA TT GA CA       CC  G
GACGCCCGCGACCTCTCCGACACCGCTTCCCTCTCCCAGTTCGACCACCCCTCCCCCCGC
```

Figure 9: Example for the problem of higher sequence heterogenity on the level of nucleic acids. It shows an hypothetical amino acid alignment on top which represents a high degree of similarity between both protein sequences allowing for an unambiguous alignment. Below the same sequences are aligned on the level of nucleic acids. It is clearly visible that the sequences are much more heterogenous: the pairwise identity is only 33%. This is only slightly above the 25% identity expected for two random nucleic acid sequences.

3.2 Some Characteristics of virus genomes

Virus genomes contain various open reading frames within their nucleic acid sequences as they are available as data sets in various data banks (e.g. `GenBank`). The lengths of small virus genomes can vary from some 3500 bp as in hepatitis B up to about 20000 bp as in the case of Ebola. The typical genome size is about 10000 bp. The genomes can consist of single-stranded or double-stranded DNA or single- or double-stranded RNA. Also some viruses with relatively large double-stranded genomes exist, like the pox viruses. Retrotranscribing viruses are the retroviruses (e.g. HIV), the hepatitis B viruses as well as caulimoviruses which have a DNA genome but use RNA as an intermediate during their replication. RNA viruses have enormously high mutation rates of up to 10^{-3} per position and replication. The number of the open reading frames depends on the type of virus considered. In addition, the organization of virus genomes is extremely variable. Overlapping open reading frames are possible, hence one part of the nucleic acid sequence codes for more than one protein in different frames. Theoretically, three open reading frames can be covered by the same nucleic acid sequence in all three possible reading frames. This possibility is actually realized in the hepatitis B virus. In addition, various non coding regions can exist in a certain virus genome.

3.3 The Idea behind RALIGN

The idea behind the combined amino acid and nucleic acid based alignments (`RALIGN`) is that coding regions on the level of protein vary less than on the level of nucleic acid, because most amino acids are coded by more than one codon (base triplet) and some different nucleic acid sequences can produce the same protein sequence after translation. Thus, our approach was to improve the quality of sequence alignments of RNA viruses (especially the pregenomic RNA intermediate of Hepatitis B virus) by creating and implementing a combined alignment algorithm.

One could argue that the quality of sequence alignments could be raised simply by translating the entire nucleic acid sequence into protein and processing on the level of proteins. But a very important factor is that the viral genomic sequence could consist of more than just one open reading frame (various coding regions in different frames) as well as some non-coding regions. These non-coding regions should, of course, be processed as nucleic acids, and every open reading frame should be processed in the correct frame.

3.4 The RALIGN Algorithm

The combined amino acid and nucleic acid based alignment procedure is made available in a program called `RALIGN`. The source code of the package is written in the programming language C and will run on computers with a conforming C compiler.

RALIGN reads GenBank nucleic acid sequences from sequence files in Pearson's format and GenBank format. Besides, it is possible for the user to define one or more than one codon tables for each sequence or a group of sequences. Every input file can be processed using its own codon table. The standard codon table is the universal genetic codon table which fits most cases. Entering 'ralign' without any options or input files displays a list of the various available codon tables. These user-defined codon tables are then used by the program for translation and, of course, for finding the correct start- and stop-codons in the nucleic acid sequences. Then the program finds all possible open reading frames which have a previously defined minimal length.

GenBank files may contain information about the exact positions of start- and stop-codons, the genomic structure of exons and introns or the protein sequence after translation. If some information like this (e.g. regarding exons and introns) is present in the GenBank file, it can be obtained and used as preferred information.

The detected coding regions are translated, using the correct codon table, and the resulting proteins are compared to the protein sequences in the GenBank file, if available. An output file is created which contains all data about the detected open reading frames, either derived by reading the data in the GenBank file or as a result of the automatic search done by the program. From this file the user can get information about all open reading frames, about their length, their start and stop, and the lengths of their proteins after translation. Also a second file is created: a PostScript output file which gives a graphical representation of the found open reading frames either in one of the three frames or, beyond these, as derived from the GenBank file input with all introns.

In many cases we can see significant differences in the genetic structure regarding the number and order of various open reading frames even between very closely related sequences. This makes it difficult to decide which ORFs

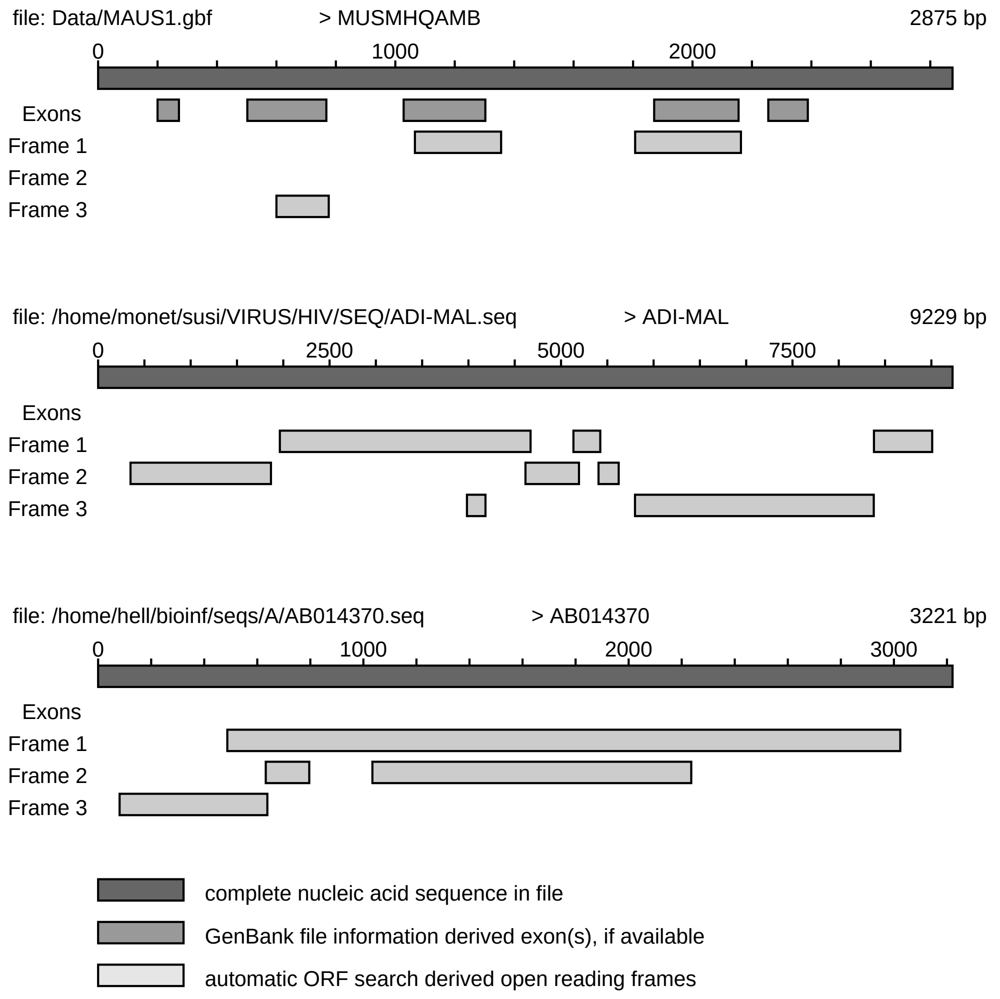

file: Data/MAUS1.gbf
> MUSMHQAMB
2875 bp
0
1000
2000
Exons
Frame 1
Frame 2
Frame 3
file: /home/monet/susi/VIRUS/HIV/SEQ/ADI-MAL.seq
> ADI-MAL
9229 bp
0
2500
5000
7500
Exons
Frame 1
Frame 2
Frame 3
file: /home/hell/bioinf/seqs/A/AB014370.seq
> AB014370
3221 bp
0
1000
2000
3000
Exons
Frame 1
Frame 2
Frame 3
complete nucleic acid sequence in file
GenBank file information derived exon(s), if available
automatic ORF search derived open reading frames

Figure 10: An example for the `PostScript` output of `RALIGN`. The figure shows a graphical representation of the found open reading frames of three unrelated sequences. A mouse gene, the genome of HIV1 and the genome of hepatitis B virus. The exon-intron organization in the mouse gene is shown (as obtained through the `GenBank` file) and also all found open reading frames in the three sequences.

```
NUMBER OF SEQUENCE INPUT: 1
PATH OF SEQUENCE INPUT FILE: Data/MAUS1.gbf
NAME OF SEQUENCE: MUSMHQAMB
found ORF's: 4

ORF number (order of appearance): 1
exon information: GenBank_file
start: 200    stop: 2388    length: 1035
protein sequence length: 344
No differences to protein sequence in GenBank file.

ORF number (order of appearance): 4
exon information: search_result
start: 1807    stop: 2163    length: 357
protein sequence length: 118
Not covered by GenBank file information.

ORF number (order of appearance): 3
exon information: search_result
start: 1066    stop: 1356    length: 291
protein sequence length: 96
Not covered by GenBank file information.

ORF number (order of appearance): 2
exon information: search_result
start: 600    stop: 776    length: 177
protein sequence length: 58
Not covered by GenBank file information.
```

Figure 11: An example for the text file output of `RALIGN`. Various data about the input sequences are shown, like the name, the positions of start and stop codons, the lengths of nucleic acid and protein sequences and information whether the open reading frame was found automatically or derived from the GenBank file.

correspond to each other in the various sequences.

Overlapping open reading frames are quite frequent in virus genomes. If a certain part of the sequence is coding for two or three proteins, a decision has to be made which open reading frame is used for the protein alignment. `RALIGN` constructs a hierarchy which considers the lengths of the open reading frames. The longest coding region gets highest priority, the second longest second highest priority and so on. The largest selected coding regions from every sequence get aligned first as a protein alignment.

The program makes a first decision, which coding regions are maintained through the alignments as protein sequences and what regions get aligned on the level of nucleic acids. `RALIGN` gives the highest priority to the longest open reading frames and checks simply, going from the longest to the shorter ones, which of these regions are located at about the same position. The proposed assignment is presented to the user in a file listing the open reading frames chosen for protein alignment. The user now has the possibility to alter this assumption and to tell the program exactly what coding regions are to be used for alignments on protein level. The information in both output files (text and `PostScript`) turned out to be quite helpful to make meaningful decisions about the choice of the open reading frames which get processed as protein sequences in the various alignment steps. In many cases this possibility to change the program's assumption is important.

After the user has either manipulated or accepted the chosen open reading frames, `RALIGN` computes alignments of the homologous sequence parts. In the current implementation `CLUSTAL W` is used for this purpose.

Normally, an alignment like this produces end gaps. End gaps are not penalised by the `CLUSTAL W` algorithm, so they occur very often. Here we are aligning only a piece of the genomic sequence; thus this treatment of end gaps is not desireable. Since `CLUSTAL W` is used as a 'black box' via a system call, we have to resort to a trick:

`RALIGN` cuts off the end gaps such that the remaining 'central alignment block' has no gaps both at the first and the last position. The sequence pieces that have been cut off are joined to the neighbouring sequence parts before and after the now aligned protein parts of the sequence. In the case of overlapping coding regions these cut off parts are again handled on the level of the proteins that these regions code for. On the other hand, if the neighbouring sequences are non-coding, the cut off sequence pieces are handled directly as nucleic acids.

Then the second protein alignment of the second largest open reading frames (with second priority) is started. Again the central alignment block is generated and the cut off end gap regions are again joined to the neighbouring parts. If, in the case of overlapping coding regions, the central block of the first alignment (the alignment of higher priority) is still overlapping the second open reading frame, then the second protein alignment processes only this part of the second open reading frame which is not covered by the prior alignment. In order to be able to smoothly join the first and second central alignment block we have to suppress the generation of end gaps in the second alignment. This is achieved by adding a tag to each of the sequences to be aligned. In the present implementation this tag consists of 12 copies of the string `THISISATAG`, which is quite unlikely both for a native amino acid and nucleic acid sequence. Im almost all cases, therefore, `CLUSTAL W` aligns the artifical tag sequences with each other and hence provides us with well defined edges for the alignment of the real sequence. The ends of the second protein alignment part which lie adjacent to the first central alignment block are therefore forced to lie exactly one above the other.

After having aligned all protein subsequences (all chosen open reading frames), after having removed all end gap containing regions, and after having linked them to the neighbouring parts of the sequences, the alignments of the non-coding regions start. Again the ends of the aligned sequence parts are forced to lie one above the other, if these ends are adjacent to formerly aligned protein parts. That way all parts can be joined smoothly together.

The protein alignments are then reverse translated. At every position where the protein alignments contain a gap of length n, a gap of length $3n$ is inserted into the corresponding nucleic acid sequence at the corresponding site.

Finally, all alignments, either on the level of proteins or nucleic acids, get combined and a resulting alignment output file is created which contains the complete nucleic acid sequence alignment.

In some rare cases `CLUSTAL W` will not properly align the tag regions added to suppress end gaps. Gaps inserted into the tags can lead to imperfect removal of the tags and thereby corruption of the sequences.

In a last step the final alignment is checked for such errors. Currently, the only recourse is to remove the offending sequence from the alignment. A permanent fix will be implemented at a later time.

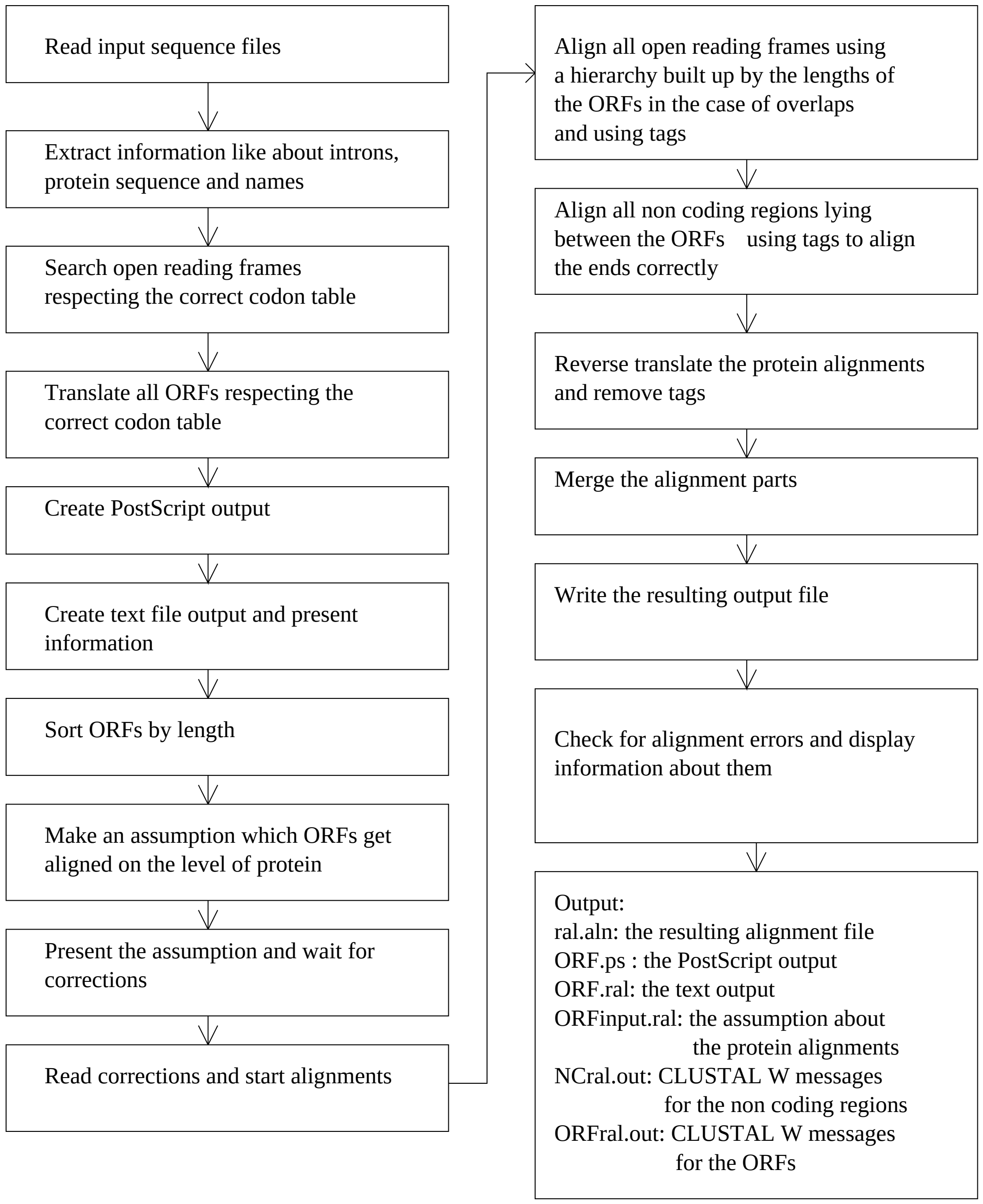

Read input sequence files
Extract information like about introns, protein sequence and names
Search open reading frames respecting the correct codon table
Translate all ORFs respecting the correct codon table
Create PostScript output
Create text file output and present information
Sort ORFs by length
Make an assumption which ORFs get aligned on the level of protein
Present the assumption and wait for corrections
Read corrections and start alignments
Align all open reading frames using a hierarchy built up by the lengths of the ORFs in the case of overlaps and using tags
Align all non coding regions lying between the ORFs using tags to align the ends correctly
Reverse translate the protein alignments and remove tags
Merge the alignment parts
Write the resulting output file
Check for alignment errors and display information about them
Output:
ral.aln: the resulting alignment file
ORF.ps : the PostScript output
ORF.ral: the text output
ORFinput.ral: the assumption about the protein alignments
NCral.out: CLUSTAL W messages for the non coding regions
ORFral.out: CLUSTAL W messages for the ORFs

Figure 12: This flow chart shows the main steps of `RALIGN`.

4 Results

4.1 Examples for Improved alignments

4.1.1 Human Immunodeficiency Viruses

Although individual cases of unexplained immunosuppresion accompanied by opportunistic infection were recognized in industrialized societies during the 1960s and 1970s, 1981 proved to be a turning point in the recognition of a new syndrome. Outbreaks of immunodeficiency-associated conditions such as Kaposi's sarcoma, mucosal candidiasis or *Pneumocystis carinii* pneumonia were described. As the underlying infectious agent for this newly described immunodeficiency syndrome human retroviruses were identified, named *human immunodeficiency virus type I* (HIV-1). In 1986, a second HIV (HIV-2) was isolated in West Africa and subsequently in Europe and North America.

The CD4 receptor, present on most T-helper cells, many cells of monocyte-macrophage lineage, and certain other cell types, appears to be the principal receptor for attachment of both HIV-1 and HIV-2. But also other mechanisms for viral entry into certain cells are possible and the search for additional receptors (like the Fc receptor) that facilitate virus entry into cells has been intense.

Over the course of infection, the virus an individual carries broadens in tropism and biologic variability. Small changes in the envelope glycoprotein amino acid composition can lead to large differences in phenotype. Sequence variation occurs rapidly. Although a predominant HIV species is maintained over time, swarms of quasispecies of subtly altered viruses emerge with broadened tropism and possibly increased cytopathic capacity.

HIV-1 is a highly complex retrovirus [42, 44]. Its genome is dense with information for coding of proteins and biologically significant RNA secondary structures [48]. The latter play a role in both the entire genomic HIV-1

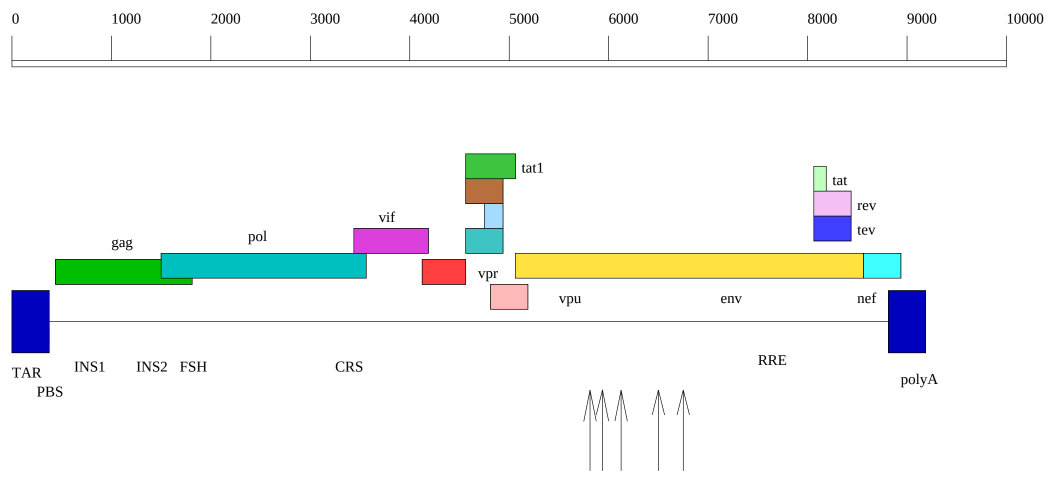


Figure 13: Organization of a retrovirus genome like HIV-1. Proteins are shown on top, known features of the RNA are indicated below. For details about the genes see the text. Arrows indicate the positions of the following five examples for improved alignment results.

sequence and in the separate HIV-1 mRNAs, which are basically (combined) fragments of the entire genome [3].

The major genes of HIV-1 are *gag*, *pol*, *env*, *tat* and *rev*. See the figure for graphical representation.

The *gag* gene codes for structural proteins for the viral core. The *pol* gene codes among others for the reverse transcriptase and the protein that integrates the viral DNA (after reverse transcription) into the host DNA. The *env* gene codes for the envelope proteins. The *tat* and *rev* genes code for regulatory proteins, Tat and Rev, that can bind to TAR and the RRE, respectively. INS1, INS2 and CRS are RNA sequences that destabilize the transcript in the absence of the Rev protein. FSH refers to the hairpin that is involved in the ribosomal frameshift from *gag* to *pol* during translation. Poly(A) refers to the polyadenylation signal. PBS is the primer binding site.

In the following, parts of an alignment of HIV-1 genomes will serve as examples how the number of gaps could be reduced. These parts are cut out of an complete alignment with a total length of the order of 10 000 nt.

There is no significant difference between the alignment generated by `CLUSTAL W` and by `RALIGN` regarding the length of the alignment. But the number of gaps is much lower in the `RALIGN` generated alignment while the gaps are longer. Besides, `RALIGN` produces no gaps which cannot be divided by 3 in the regions of the protein alignments. See the figures on the next pages which show cut out parts of a conventional alignment produced by `CLUSTAL W` in contrast to an alignment produced by the combined amino acid and nucleic acid based alignment algorithm as implemented in `RALIGN`. It is shown that the number of gaps in the alignment could have been reduced distinctly using `RALIGN`.

```
ADI-MAL         UAAACUGCACUAAUGUGAAUGGGACUGCUGU---GAAUGGGACUAAUGCUGGGAGUAAUA
AE-90CF402      UACAUUGUACCAAGG--CU----------------AGUUUUACUAAUGCCA---------
AE-CM240        UAAAUUGUACCAAUG--CU----------------AAUUUGACCAAUGGCAGUAGCAAAA
B-896           UAAAUUGCACUAAUUUGAAUAUCACUA------AGAAUACUACUAAUCCCACUAGUAGCA
B-ACH320A       UAAAUUGCACUGAU---UUUG------------GGAAUGCUACUAAUACCACUAGUAGUA
B-BCSG3         UAAAUUGCACUGAUGAGUUGA------------AGAAUGCUACUAAUACCACUAGUACUA
B-CAM1          UAAUUUGCACUAAUG-UA-----------------AAUAAUACUAGGACCAAUAGUAGUG
B-D31           UAAAUUGCACUGAUC-UGAAG--------------AAUGCUACUAAUACCAAUAAUAGUA
B-HIV1AD8       UAAAUUGCACUGAUU-UGAGG--------------AAUGUUACUAAUAUCAAUAAUAGUA
B-HXB2          UAAAGUGCACUGAU---UUGA------------AGAAUGAUACUAAUACCAAUAGUAGUA
B-JRCSF         UAAAUUGCAAAGAUGUGAA------------------UGCUACUAAUACCACUAGUAGUA
B-LAI           UAAAGUGCACUGAU---UUGG------------GGAAUGCUACUAAUACCAAUAGUAGUA
B-MANC          UAGAUUGCACUGAUUAUGUAG------------GGAAUGCUACUAAUACCACUAGCACUA
B-OYI           UAGAUUGCACUGAUGUUAAUA------------CCACUAGUAGUAGUUUGAGGAAUGCUA
B-SF2           UAAAUUGCACUGAUU-UGGGG--------------AAGGCUACUAAUACCAAUAGUAGUA
B-WEAU          UAAAUUGCACUAAUGUGAAUGUGACUAAUUUGAAGAAUGAGACUAAUACCAAUAGUAGUA
B-YU2           UAAAUUGCACUGAU---UUAA------------GGAAUGCUACUAAUACCACUAGUAGUA
B-pNL43         UAAAGUGCACUGAU---UUGA------------AGAAUGAUACUAAUACCAAUAGUAGUA
D-ELI           UAAACUGUAGUGAU--------------------GAAU-UGAGGAACA---AUGGCACUA
D-NDK           UAAACUGCACUGAU--------------------GAAU-UGAGGAACAGCAAGGGCA--A
O-ANT70         UGGAGUGU---------------------------------ACAAACAUAGCUGGAACAA
O-MVP5180       UGAACUGUGUAGAUC---------------U------GCAAACAAAUAAAACAGGCCUAU
SIVCPZGAB       UGCAGUGCAGUAAGG------------------CUAACUUUAGCCAGGCAAAAAACCUAA
```

Figure 14: In the central region of these part of an conventional `CLUSTAL W` alignment of HIV-1 genomes there are some gaps of length 3, separated from the neighbouring gaps by just 4 nucleotides. In some lines (like the third one) we have gaps of length 1. And in the last line there are two gaps separated just by one single nucleotide. So almost each line contains more than one gap. The total number of gaps in this alignment part is 39, 17 can not be divided by 3.

```
ADI-MAL         UAAACUGCACUAAUGUGAAUGGGACUGCUGUGAAUGGGACUAAUGCUGGGAGUAAUAGGA
AE-90CF402      UACAUUGUACCAAG---------------------------GCUAGUUUUACUAAUGCCA
AE-CM240        UAAAUUGUACCAAU---------------------------GCUAAUUUGACCAAUGGCA
B-896           UAAAUUGCACUAAUUUGAAUAUC---------------ACUAAGAAUACUACUAAUCCCA
B-ACH320A       UAAAUUGCACUGAUUUUGGG------------------------AAUGCUACUAAUACCA
B-BCSG3         UAAAUUGCACUGAU---------------------GAGUUGAAGAAUGCUACUAAUACCA
B-CAM1          UAAUUUGCACUAAUGUAAAU---------------AAUACUAGGACCAAUAGUAGUGAUU
B-D31           UAAAUUGCACUGAUCUGAAG------------------------AAUGCUACUAAUACCA
B-HIV1AD8       UAAAUUGCACU------------------------GAUUUGAGGAAUGUUACUAAUAUCA
B-HXB2          UAAAGUGCACUGAUUUGAAG------------------------AAUGAUACUAAUACCA
B-JRCSF         UAAAUUGCAAAGAUGUG---------------------------AAUGCUACUAAUACCA
B-LAI           UAAAGUGCACUGAUUUGGGG------------------------AAUGCUACUAAUACCA
B-MANC          UAGAUUGCACUGAUUAUGUAGGG---------------------AAUGCUACUAAUACCA
B-OYI           UAGAUUGCACUGAUGUUAAUACCACUAGUAGU---AGUUUGAGGAAUGCUACUAAUACCA
B-SF2           UAAAUUGCACUGAU------------------------UUGGGGAAGGCUACUAAUACCA
B-WEAU          UAAAUUGCACUAAUGUGAAUGUGACU---------AAUUUGAAGAAUGAGACUAAUACCA
B-YU2           UAAAUUGCACUGAU------------------------UUAAGGAAUGCUACUAAUACCA
B-pNL43         UAAAGUGCACUGAUUUGAAG------------------------AAUGAUACUAAUACCA
D-ELI           UAAACUGUAGUGAUGAAUUGAGGAACAAUGGC---------ACUAUGGGGAACAAUGUCA
D-NDK           UAAACUGCACUGAUGAAUUGAGGAAC---------------AGCAAGGGCAAUGGGAAGG
O-ANT70         UGGAGUGU---------------------------------ACAAACAUAGCUGGAACAA
O-MVP5180       UGAACUGUGUAGAUCUGCAA---------------------ACAAAUAAAACAGGCCUAU
SIVCPZGAB       UGCAGUGCAGUAAGGCUAACUUUAGC---------CAGGCAAAAAACCUAACAAACCAGA
```

Figure 15: This figure shows about the same region of the HIV-1 genomes as above, after an alignment using the combined amino acid and nucleic acid based alignment algorithm as implemented in `RALIGN`. It is clearly visible that the number of gaps could have been decreased dramatically. Each line contains just one gap of different lengths. The number of gaps is 22, this is 44% less than in the above `CLUSTAL W` generated alignment part.

```
ADI-MAL         GGGAGUAAUAGGACUA-------AUGCAG--AAUUGAAA--AUGGAAAUUGGAGAAGUGAAAAAC
AE-90CF402      A---------CCAGUG-----------ACAGAAUA---AAAAUG---GAAGAUGCAGUAAGAAAC
AE-CM240        AGUAGCAAAACCAAUGU-----CUCUAACAUAAUAGGAAAUAUA---ACAGAUGAAGUAAGAAAC
B-896           ACUAGUAGCA---------------GCUGGGGAAUGA-----UGGAGAAAGGAGAAAUAAAAAAU
B-ACH320A       ACUAGUAGUA---------------GCGGGGUUAUAAUA----G-AGAAAGGAGAAAUAAAAAAC
B-BCSG3         ACUAGUACUAAUACCCCUAGUGGUAGCUGGAAAAAGAU-----GGAAAGAGGAGAAAUAAAGAAC
B-CAM1          AAUAGUAGUGAUU-----GGGACAGGAGGGAAGGAGAAAAGAUG---AAAGGAGAAAUAAAAAAC
B-D31           AAUAAUAGUAGU-----------UGGACGAUGACAGGAGAAAUG---AAAGGAGAAAUAAAAAAC
B-HIV1AD8       AAUAAUAGUAGU-------------GAGGGAA----------UG---AGAGGAGAAAUAAAAAAC
B-HXB2          AAUAGUAGUA---------------GCGGGAGAAUGAUA--AUGGAGAAAGGAGAGAUAAAAAAC
B-JRCSF         ACUAGUAGUA---------------GUGAGGGAAUGA-----UGGAGAGAGGAGAAAUAAAAAAC
B-LAI           AAUAGUAGUAAUACCAAUAGUAGUAGCGGGGAAAUGAUG--AUGGAGAAAGGAGAGAUAAAAAAC
B-MANC          ACUAGCACUAAUAAUACCGCUAGUGGAAGUUGG--GGAGCGAUG---AGAGGGGAAAUAAAAAAC
B-OYI           AGGAAUGCUACUAAUACCACAAGUAGUAGUUGG--GAAACGAUGGAGAAAGGAGAAUUAAAAAAC
B-SF2           AAUAGUAGUAAUU-----GGAAA--GAAGAAA----------UA---AAAGGAGAAAUAAAAAAC
B-WEAU          AAUAGUAGUAGU------------GGAGGGGAAAAGA-----UGGAGGAGGGAGAAAUGAAAAAC
B-YU2           ACUAGUAGUA---------------GCUGGGAAACGAU-----GGAGAAAGGAGAAAUAAAAAAC
B-pNL43         AAUAGUAGUA---------------GCGGGAGAAUGAUA--AUGGAGAAAGGAGAGAUAAAAAAC
D-ELI           -AUGGCACUA-U-------------GGGGAACAAUGUCACUACAGAGGAGAAAGGAAUGAAAAAC
D-NDK           AAGGGCA--A-U-------------GGG----AAGGUA---GAAGAGGAGGAAAAAAGGAAAAAC
O-ANT70         GCUGGAACAA--------------------CAAUGA-------------AAACCUUAUGAAGAAG
O-MVP5180       ACAGGCCUAU--------------------UAAAUGAGA------CAAUAAAUGAGAUGAGAAAU
SIVCPZGAB       AAAAACCUAA----------------CAAACCAGACAUCUUCUC-CGCCUCUCGAAAUGAAAAAC
```

Figure 16: Again a part of an usual alignment created by CLUSTAL W. Almost each line contains more than one gap, there are also some lines containing up to three gaps, like the first one. This is a rather bad alignment. The total number of gaps is 53, 34 can not be divided by 3.

```
ADI-MAL         ACUAAUGCAGAAUUGAAAAUGGAA---------------------AUUGGAGAAGUGAAAAAC
AE-90CF402      ACCAGUGACAGA------------------------AUAAAAAUGGAAGAUGCAGUAAGAAAC
AE-CM240        AGUAGCAAAACCAAUGUCUCU------AACAUAAUAGGAAAUAUAACAGAUGAAGUAAGAAAC
B-896           ACUAGUAGCAGC------------UGGGGA------AUGAUGGAGAAAGGAGAAAUAAAAAAU
B-ACH320A       ACUAGUAGU---------------AGCGGG---GUUAUAAUAGAGAAAGGAGAAAUAAAAAAC
B-BCSG3         ACUAGUACUAAU---ACCCCUAGUGGUAGCUGGAAAAAGAUGGAAAGAGGAGAAAUAAAGAAC
B-CAM1          UGGGACAGGAGGGAAGGAGAA---------------AAG---AUGAAAGGAGAAAUAAAAAAC
B-D31           AAUAAUAGUAGUUGGACGAUGACAGGA------------GAAAUGAAAGGAGAAAUAAAAAAC
B-HIV1AD8       AAUAAUAGUAGUGAGGGA------------------------AUGAGAGGAGAAAUAAAAAAC
B-HXB2          AAUAGUAGU---------------AGCGGGAGAAUGAUAAUGGAGAAAGGAGAGAUAAAAAAC
B-JRCSF         ACUAGUAGUAGU------------GAGGGA------AUGAUGGAGAGAGGAGAAAUAAAAAAC
B-LAI           AAUAGUAGUAAUACCAAUAGUAGUAGCGGGGAAAUGAUGAUGGAGAAAGGAGAGAUAAAAAAC
B-MANC          ACUAGCACUAAUAAUACCGCUAGUGGAAGUUGG---GGAGCGAUGAGAGGGGAAAUAAAAAAC
B-OYI           ACAAGUAGU------------------AGUUGGGAAACGAUGGAGAAAGGAGAAUUAAAAAAC
B-SF2           AAUAGUAGU------------------AAUUGGAAAGAAGAAAUAAAAGGAGAAAUAAAAAAC
B-WEAU          AAUAGUAGUAGUGGAGGGGAA---------------AAGAUGGAGGAGGGAGAAAUGAAAAAC
B-YU2           ACUAGUAGU------------------AGCUGGGAAACGAUGGAGAAAGGAGAAAUAAAAAAC
B-pNL43         AAUAGUAGU---------------AGCGGGAGAAUGAUAAUGGAGAAAGGAGAGAUAAAAAAC
D-ELI           ACUACAGAGGAGAAAGGAAUG------------------------------------AAAAAC
D-NDK           GUAGAAGAGGAGGAAAAAAGG------------------------------------AAAAAC
O-ANT70         ACAAAUGAAAACCUU---------------------------------------AUGAAGAAG
O-MVP5180       UUAAAUGAGACAAUAAAU---------------------------------GAGAUGAGAAAU
SIVCPZGAB       ACAUCUUCUCCG---------------------------------CCUCUCGAAAUGAAAAAC
```

Figure 17: This figure shows again about the same region of HIV-1 genomes, but, in contrast, created by RALIGN. Again clearly visible is the reduction of the number of gaps: most of the lines contain just one gap, with few lines containing two gaps. There is no line with more than two gaps. The number of gaps could be reduced by about the half (26 gaps).

```
ADI-MAL         CAAAU----AG-----------------AUGAUAGUGAUAAUAG--------AUAG---UACUAAUUAUAGGUUAAUAAAUUG
AE-90CF402      CAAAUUAGUAGUAGUGUACAAAAUAAUAAUAACAGUAAUACUAGUGGACAAAAUAAUAGUCAUAAGUUUAGAUUAAUACAUUG
AE-CM240        CAAAUU-GAAG-----------------AUAAGAAGACUAGUAGUG------------------AGUAUAGGUUAAUAAAUUG
B-896           CCAAU----AG-----------------AAAAUACUAAUAAUA------------------CUAAGUAUAGGUUAAUAAGUUG
B-ACH320A       CCAAU----AG-----------------AUAAUAAUAAUACUAAUA------CCAGCUAUACCAGCUAUAGGUUGAUAAGUUG
B-BCSG3         CCAAU----AG-----------------AUAAUGAUAAGAAUAG---------------UACCAAAUAUAGGUUGAUAAGUUG
B-CAM1          CCAAU----AG-----------------AUAAGGCUAAUACAAGU---------------------UAUACAUUGAUACAUUG
B-D31           CCAAU----AG-----------------AUAAUGACAAUACUAG---------------------CUAUAGGUUGAUAAGUUG
B-HIV1AD8       CCAAU----AG-----------------AUAAUGAUAAUACUAG---------------------CUAUAGGUUGAUAAAUUG
B-HXB2          CCAAU----AG-----------------AUAAUGAUAC---------------------UACCAGCUAUAAGUUGACAAGUUG
B-JRCSF         CCAAU----AG-----------------AUAAUAAGAAUAAUA------------------CCAAAUAUAGGUUAAUAAGUUG
B-LAI           CCAAU----AG-----------------AUAAUGAUAC---------------------UACCAGCUAUACGUUGACAAGUUG
B-MANC          CCAAU----AG-----------------AAAAGAAGAAUACUAG---------------------CUUUAGAUUGAUAAGUUG
B-OYI           CCAAU----AG-----------------AUAAGAAUGAUACUAA---------------------AUUUAGGUUAAUACAUUG
B-SF2           CCAAU----AG-----------------AUAAUGCUAGUACUACUA------CCAACUAUACCAACUAUAGGUUGAUACAUUG
B-WEAU          CCAAU----AG-----------------AUCAUGAUAAUACAAG---------------------CUAUACGUUGAUAAAUUG
B-YU2           CCAAU----AG-----------------AUAAUG---------------------------CUAGCUAUAGGUUGAUAAGUUG
B-pNL43         CCAAU----AG-----------------AUAAU---------------------------ACCAGCUAUAGGUUGAUAAGUUG
D-ELI           CCAAU----AG-----------------ACAAUGAUAGUAGUACCA------AUAG---UACCAAUUAUAGGUUAAUAAAUUG
D-NDK           CCAAU----AG-----------------ACAAUAAUAAUAGGACCA------AUAG---UACUAAUUAUAGGUUAAUAAAUUG
O-ANT70         GAACUG--AAUGAG----ACAAGCAGCACAAAUAAGACAAACAGC---------------AAAAUGUAUACAUUAACUAAUUG
O-MVP5180       AAGGUU--AAUGA-------------CUCAAAUGCAGUAAAUGGA---------------ACAACAUAUAUGUUAACUAAUUG
SIVCPZGAB       AACCU----AG-----------------GGAAUGAGAACAACAC---------------------AUAUAGGAUAAUUAAUUG
```

Figure 18: This alignment part shows two very regular gaps of length 4 and 17, separated by just 2 nucleotides. On the right side another important gap containing region lies. Each line contains three or four gaps. The total gap number is 69, 45 can not be divided by 3. Created by an conventional alignment using `CLUSTAL W`.

```
ADI-MAL         CAAAUAGAUGAU---------------------------------------AGUGAUAAUAGUAGUUAUAGGCUAAUAAAUUG
AE-90CF402      CAAAUUAGUAGUAGUGUACAAAAUAAUAAUAACAGUAAUACUAGUGGACAAAAUAAUAGUCAUAAGUUUAGAUUAAUACAUUG
AE-CM240        CAAAUUGAA---------------------GAU---------------AAGAAGACUAGUAGUGAGUAUAGGUUAAUAAAUUG
B-896           CCAAUAGAAAAUACUAAU---------------------------------------AAUACUAAGUAUAGGUUAAUAAGUUG
B-ACH320A       CCAAUAGAUAAUAAUAAUACUAAU---------------------------ACCAGCUAUACCAGCUAUAGGUUGAUAAGUUG
B-BCSG3         CCAAUAGAUAAUGAUAAG------------------------------------AAUAGUACCAAAUAUAGGUUGAUAAGUUG
B-CAM1          CCAAUAGAUAAGGCU------------------------------------------AAUACAAGUUAUACAUUGAUACAUUG
B-D31           CCAAUAGAUAAUGAC------------------------------------------AAUACUAGCUAUAGGUUGAUAAGUUG
B-HIV1AD8       CCAAUAGAUAAUGAU------------------------------------------AAUACUAGCUAUAGGUUGAUAAAUUG
B-HXB2          CCAAUAGAUAAUGAU------------------------------------------ACUACCAGCUAUAAGUUGACAAGUUG
B-JRCSF         CCAAUAGAUAAUAAGAAU---------------------------------------AAUACCAAAUAUAGGUUAAUAAGUUG
B-LAI           CCAAUAGAUAAUGAU------------------------------------------ACUACCAGCUAUACGUUGACAAGUUG
B-MANC          CCAAUAGAAAAGAAG------------------------------------------AAUACUAGCUUUAGAUUGAUAAGUUG
B-OYI           CCAAUAGAUAAGAAU------------------------------------------GAUACUAAAUUUAGGUUAAUACAUUG
B-SF2           CCAAUAGAUAAUGCUAGUACUACU---------------------------ACCAACUAUACCAACUAUAGGUUGAUACAUUG
B-WEAU          CCAAUAGAUCAUGAU------------------------------------------AAUACAAGCUAUACGUUGAUAAAUUG
B-YU2           CCAAUAGAUAAU------------------------------------------------GCUAGCUAUAGGUUGAUAAGUUG
B-pNL43         CCAAUAGAUAAU------------------------------------------------ACCAGCUAUAGGUUGAUAAGUUG
D-ELI           CCAAUAGACAAUGAUAGUAGU------------------------------ACCAAUAGUACCAAUUAUAGGUUAAUAAAUUG
D-NDK           CCAAUAGACAAUAAUAAUAGG------------------------------ACCAAUAGUACUAAUUAUAGGUUAAUAAAUUG
O-ANT70         GAACUGAAUGAGACAAGCAGCACAAAUAAGACAAACAGC---------------------AAAAUGUAUACAUUAACUAAUUG
O-MVP5180       AAGGUUAAUGACUCAAAUGCAGUAAAUGGAACAACA------------------------------UAUAUGUUAACUAAUUG
SIVCPZGAB       AACCUAGGGAAUGAG------------------------------------------AACAACACAUAUAGGAUAAUUAAUUG
```

Figure 19: This alignment part gives distinct evidence for the powerful algorithm of `RALIGN`. The two plain and regular gaps of length 4 and 17 in the central region could have been avoided completely. The already existing right-handed gaps were just extended. The number of gaps is 23, this is 66% less than above.

```
ADI-MAL       CUCUAUACAACAGGGAUAGU--------A----GGAG--AUAU-AAGAAGAGCAUAUUGUACU
AE-90CF402    UUCCAUACAACAGGAAACAU-------AAAU--GGUG--AUAU-AAGAAAAGCAUAUUGUGAA
AE-CM240      UUCUAUAGAACAGGAGAUAU-------AAUA--GGAA--AUAU-AAGAAAAGCAUAUUGUGAG
B-896         UUUUAUGCAAGAAGAAACAU-------AAUA--GGAG--AUAU-AAGACAAGCACAUUGUAAC
B-ACH320A     UUUUAUGCAACAGGACAAAU-------AAUA--GGAG--AUAU-AAGACAAGCACAUUGUAAC
B-BCSG3       UAUUAUACAACAGGAGAAAU-------AGUA--GGAG--AUAU-AAGACAAGCACAUUGUAAC
B-CAM1        GUUUAUGCAACAGACAGAAU-------AAUA--GGAG--AUAU-AAGACAAGCACAUUGUAAC
B-D31         UUUUAUACAAAAGGAAAAAU-------AAUA--GGAG--AUAU-AAGACAAGCACAUUGUAAC
B-HIV1AD8     UUUUAUACAACAGGAGACAU-------AAUA--GGAG--AUAU-AAGACAAGCACAUUGCAAC
B-HXB2        UUUGUUACAAUAGGAAAAAU-------AG-----GAA--AUAU-GAGACAAGCACAUUGUAAC
B-JRCSF       UUUUAUACAACAGGAGAAAU-------AAUA--GGAG--AUAU-AAGACAAGCACAUUGUAAC
B-LAI         UUUGUUACAAUAGGAAAAAU-------AG-----GAA--AUAU-GAGACAAGCACAUUGUAAC
B-MANC        UUUCAUGUAACAAGAGCCGU-------AACA--GGAG--AUAU-AAGACAAGCACAUUGUAAC
B-OYI         UUUCAUACAACAAAACAAAU-------AAUA--GGAG--AUAU-AAGACAAGCACAUUGUAAC
B-SF2         UUUCAUACAACAGGAAGAAU-------AAUA--GGAG--AUAU-AAGAAAAGCACAUUGUAAC
B-WEAU        CUUUAUACAACAGGAGAAAU-------AAUA--GGAG--AUAU-AAGACGAGCACAUUGUAAC
B-YU2         UUGUAUACAACAGGAGAAAU-------AAUA--GGAG--AUAU-AAGACAAGCACAUUGUAAC
B-pNL43       UUUGUUACAAUAGGAAAAAU-------AG-----GAA--AUAU-GAGACAAGCACAUUGUAAC
D-ELI         CUCUAUACUACAAGAUCAA----GAUCAAUA----------AU-AGGACAAGCACAUUGUAAU
D-NDK         CUCUAUACAAUAACAGGAAAAAAGAAGAAAACAGGAU--ACAU-AGGACAAGCACAUUGUAAA
O-ANT70       UACAGCAUGGGAAU----AGGGGGAACAGCAGGAAAC--AGCUCAAGGGCAGCUUAUUGCAAG
O-MVP5180     CGCAGUAUGACACUUAAAAGAAGUAACAAUACAUCACCAAGAUCAAGGGUAGCUUAUUGUACA
SIVCPZGAB     UUUUAUAAUAUAGAAAAUGU-------AGUA--GGAG--AUAC-CAGAUCUGCCUACUGUAAG
```

Figure 20: This region in the usual alignment shows four different gaps in the central region, most of them are very short (just one or two nucleotides). They are separated by very short sequence fragments of 4 nucleotides only. This alignment result consists of many disrupted fragment and is no good alignment. The number of gaps is 83, all of them can not be divided by 3.

```
ADI-MAL       CUCUAUACAACAGGGAUAGUAGGA---------------GAUAUAAGAAGAGCAUAUUGUACU
AE-90CF402    UUCCAUACAACAGGAAACAUAAAUGGU------------GAUAUAAGAAAAGCAUAUUGUGAA
AE-CM240      UUCUAUAGAACAGGAGAUAUAAUAGGA------------AAUAUAAGAAAAGCAUAUUGUGAG
B-896         UUUUAUGCAAGAAGAAACAUAAUAGGA------------GAUAUAAGACAAGCACAUUGUAAC
B-ACH320A     UUUUAUGCAACAGGACAAAUAAUAGGA------------GAUAUAAGACAAGCACAUUGUAAC
B-BCSG3       UAUUAUACAACAGGAGAAAUAGUAGGA------------GAUAUAAGACAAGCACAUUGUAAC
B-CAM1        GUUUAUGCAACAGACAGAAUAAUAGGA------------GAUAUAAGACAAGCACAUUGUAAC
B-D31         UUUUAUACAAAAGGAAAAAUAAUAGGA------------GAUAUAAGACAAGCACAUUGUAAC
B-HIV1AD8     UUUUAUACAACAGGAGACAUAAUAGGA------------GAUAUAAGACAAGCACAUUGCAAC
B-HXB2        UUUGUUACAAUAGGAAAA---AUAGGA------------AAUAUGAGACAAGCACAUUGUAAC
B-JRCSF       UUUUAUACAACAGGAGAAAUAAUAGGA------------GAUAUAAGACAAGCACAUUGUAAC
B-LAI         UUUGUUACAAUAGGAAAA---AUAGGA------------AAUAUGAGACAAGCACAUUGUAAC
B-MANC        UUUCAUGUAACAAGAGCCGUAACAGGA------------GAUAUAAGACAAGCACAUUGUAAC
B-OYI         UUUCAUACAACAAAACAAAUAAUAGGA------------GAUAUAAGACAAGCACAUUGUAAC
B-SF2         UUUCAUACAACAGGAAGAAUAAUAGGA------------GAUAUAAGAAAAGCACAUUGUAAC
B-WEAU        CUUUAUACAACAGGAGAAAUAAUAGGA------------GAUAUAAGACGAGCACAUUGUAAC
B-YU2         UUGUAUACAACAGGAGAAAUAAUAGGA------------GAUAUAAGACAAGCACAUUGUAAC
B-pNL43       UUUGUUACAAUAGGAAAA---AUAGGA------------AAUAUGAGACAAGCACAUUGUAAC
D-ELI         CUCUAUACUACAAGAUCAAGAUCA---------------AUAAUAGGACAAGCACAUUGUAAU
D-NDK         CUCUAUACAAUAACAGGAAAAAAGAAGAAAACAGGA---UACAUAGGACAAGCACAUUGUAAA
O-ANT70       UACAGCAUGGGAAUAGGGGGAACAGCAGGAAACAGC------UCAAGGGCAGCUUAUUGCAAG
O-MVP5180     CGCAGUAUGACACUUAAAAGAAGUAACAAUACAUCACCAAGAUCAAGGGUAGCUUAUUGUACA
SIVCPZGAB     UUUUAUAAUAUAGAAAAUGUAGUAGGA------------GAUACCAGAUCUGCCUACUGUAAG
```

Figure 21: Using RALIGN the number of gaps could have been decreased again. The resulting alignment contains one larger gap in most cases and only few sequences have a second, shorter gap. The successful reduction of gaps by the combined amino acid and nucleic acid based alignments is clearly visible. Total number of gaps in this region: 25 (70% less gaps).

```
ADI-MAL      AU-AGUACAUGGCAGAAUAAUGGUGC----AAGA------CU-AA--GU----AAUAGCACAGAGUC---AACUGGUAGUAUCACACUCCCAUG
AE-90CF402   AU-AGUACUUGGAUA---------------AAUGGAACCAUGCAGGAGGUU--AAUGGCACAAACUC---A---GGCAAUAUCACACUUCCAUG
AE-CM240     AU-AAUACUUGCCUAG------------GAAAUGAAACCAUGGCGGGGUGU--AAUGACAC------------------UAUCACACUUCCAUG
B-896        AU-AGUACUUGGAAU-------G-------U-UA------CUGGAGGGACA--AAUGGCACUGAAGG---AAAUGACAUAAUCACACUCCAAUG
B-ACH320A    AU-AGUACUUGG------AAUGAUACUGGGAAUGUUA---CUGAAAGGUCA--AAUAACAAUGA------AAAU------AUCACACUCCCAUG
B-BCSG3      AU-AGUACUUGGGCUGGGAAUAAUACUUGGAAUAGUAGUGCUGAAAGGUCA--GAUGACACUGGAGG---AAAU------AUCACACUCCCAUG
B-CAM1       AU-ACUACUUGGCUGUUUAAUGGUACUUGGAAUGAUA---CUGAAGGGUUA--AAUAACACUGAAAG---AAAU------AUUACACUUCCAUG
B-D31        AU-AGUACUUGGAAU------------------GAUA---CUAAAGAGUCA--AAUAACACAAAU---------GGAACUAUCACACUCCCAUG
B-HIV1AD8    AU-AGUACUUGGAAUUUUAAUGGUACUUGGAAUUUAA---CACAAUCG-----AAUGGUACUGAAGG---AAAUGACACUAUCACACUCCCAUG
B-HXB2       AU-AGUACUUGGUUU---AAUAGUACUUGGAGUA------CUGAAGGGUCA--AAUAACACUGAAGG---AAGUGACACAAUCACCCUCCCAUG
B-JRCSF      AU-AGUACUUGGAAU-------G-------A-UA------CUGAAAAGUCA--AGUGGCACUGAAGG---AAAUGACACCAUCAUACUCCCAUG
B-LAI        AU-AGUACUUGGUUU---AAUAGUACUUGGAGUA------CUGAAGGGUCA--AAUAACACUGAAGG---AAGUGACACAAUCACACUCCCAUG
B-MANC       AU-AGUACUUGGAAUACUGGG----------AAUGAUA---CUAGAGAGUCA--AAUGACACAAAUAA---UACUGGAAAUAUCACACUCCCAUG
B-OYI        AU-AGUACUUGGAAU------------------GAUA---CUACAAGGGCA--AAUAGCACUGAA----------GUAACUAUCACACUCCCAUG
B-SF2        AU-AAUACAUGGAGGUUAAAU-------------------CACACU-G-----AA-GGAACUAAAGG---AAAUGACACAAUCAUACUCCCAUG
B-WEAU       AU-AGUACUUGGCAUGCUAAUGGUACUUGGAAGAAUA---CUGAAGGGGCA--GAUAACAAU------------------AUCACACUCCCAUG
B-YU2        -------CUUGG------AAUGAUACUAGAAA---------------GUUA--AAUAACACUGGAAG---AAAU------AUCACACUCCCAUG
B-pNL43      AU-AGUACUUGGUUU---AAUAGUACUUGGAGUA------CUGAAGGGUCA--AAUAACACUGAAGG---AAGUGACACAAUCACACUCCCAUG
D-ELI        AU-AGUACAUGGAAUAU---UAGUGCAUGGAAUAAUAU---UACAGAGUCA--AAUAAUAGCACAAA---CAC---AAACAUCACACUCCAAUG
D-NDK        AU-AGUACAUGGAAU----------CA--GACUAAUAG---UACAGGGUUC--AAUAAUGGCACAG---------------UCACACUCCCAUG
O-ANT70      AUUA-UACCUUU-UCA----------UGUAACGGAACCACCUGUAGUGUUAGUAAUGUUAGUCAAGG------UAACAAUGGCACUCUACCUUG
O-MVP5180    ACUA-UACUUUUAUCAA--------CUGUACAAAGUCCGGAUGCCAGGAGAUCAAAGGGAGCAAUGAGACCAAUAAAAAUGGUACUAUACCUUG
SIVCPZGAB    CUGACAACAUUA-------------------------------------CA--AAUGGCAUU------------------AUAAUACUGCCAUG
```

Figure 22: This part of the HIV-1 genome alignment is extremely disrupted. We have a lot of gaps, some of them very short, some a little bit longer. An alignment like this is not usable for secondary structure prediction where the quality of the input alignments is of great importance. 116 gaps, 39 can not be divided by 3.

```
ADI-MAL      AUAGUACAUGGCAGAAUAAUGGU---GCAAGACUAAGUAAUAGCACAGAGUCAACUGGU------------------------AGUAUCACACUCCCAUG
AE-90CF402   AUAGUACUUGGAUA------------AAUGGAACCAUGCAGGAGGUUAAUGGCACAAACUCA------------------GGCAAUAUCACACUUCCAUG
AE-CM240     AUAAUACUUGCCUAGGA---------AAUGAAACCAUGGCGGGGUGUAAUGACACU------------------------------AUCACACUUCCAUG
B-896        AUAGUACUUGGAAU------------------GUUACUGGAGGGACAAAUGGCACUGAAGGA---------------AAUGACAUAAUCACACUCCAAUG
B-ACH320A    AUAGUACUUGGAAUGAUACUGGG------AAUGUUACUGAAAGGUCAAAUAACAAUGAA------------------------AAUAUCACACUCCCAUG
B-BCSG3      AUAGUACUUGGGCUGGG------------------AAUAAUACUUGGAAUAGUAGUGCUGAAAGGUCAGAUGACACUGGAGGAAAUAUCACACUCCCAUG
B-CAM1       AUACUACUUGGCUG------------------UUUAAUGGUACUUGGAAUGAUACUGAAGGG---UUAAAUAACACUGAAAGAAAUAUUACACUUCCAUG
B-D31        AUAGUACUUGGAAU------------------GAUACUAAAGAGUCAAAUAACACAAAU---------------------GGAACUAUCACACUCCCAUG
B-HIV1AD8    AUAGUACUUGGAAU------------------UUUAAUGGUACUUGGAAUUUAACACAAUCGAAUGGUACUGAAGGAAAUGACACUAUCACACUCCCAUG
B-HXB2       AUAGUACUUGGUUUAAUAGUACU------UGGAGUACUGAAGGGUCAAAUAACACUGAAGGA---------------AGUGACACAAUCACCCUCCCAUG
B-JRCSF      AUAGUACUUGGAAU------------------GAUACUGAAAAGUCAAGUGGCACUGAAGGA---------------AAUGACACCAUCAUACUCCCAUG
B-LAI        AUAGUACUUGGUUUAAUAGUACU------UGGAGUACUGAAGGGUCAAAUAACACUGAAGGA---------------AGUGACACAAUCACACUCCCAUG
B-MANC       AUAGUACUUGGAAUACU---GGG------AAUGAUACUAGAGAGUCAAAUGACACAAAUAAU---------------ACUGGAAAUAUCACACUCCCAUG
B-OYI        AUAGUACUUGGAAU------------------GAUACUACAAGGGCAAAUAGCACUGAAGUA---------------------ACUAUCACACUCCCAUG
B-SF2        AUAAUACAUGGAGG------------------UUAAAUCACACUGAAGGAACUAAAGGAAAU---------------GAC---ACAAUCAUACUCCCAUG
B-WEAU       AUAGUACUUGGCAU------------------GCUAAUGGUACUUGGAAGAAUACUGAAGGG------------GCAGAUAACAAUAUCACACUCCCAUG
B-YU2        -----ACUUGGAAU------------------GAUACUAGAAAGUUAAAUAACACUGGAAGA---------------------AAUAUCACACUCCCAUG
B-pNL43      AUAGUACUUGGUUUAAUAGUACU------UGGAGUACUGAAGGGUCAAAUAACACUGAAGGA---------------AGUGACACAAUCACACUCCCAUG
D-ELI        AUAGUACAUGGAAUAUUAGUGCAUGGAAUAAUAUUACAGAGUCAAAUAAUAGCACAAACACA---------------------AACAUCACACUCCAAUG
D-NDK        AUAGUACAUGGAAUCAGACU---------AAUAGUACAGGGUUCAAUAAUGGCACA------------------------------GUCACACUCCCAUG
O-ANT70      AUUAUACCUUUUCA------UGUAACGGAACCACCUGUAGUGUUAGUAAUGUUAGUCAA------------------GGUAACAAUGGCACUCUACCUUG
O-MVP5180    ACUAUACUUUUAUCAAC---UGUACAAAGUCCGGAUGCCAGGAGAUCAAAGGGAGCAAUGAG------------ACCAAUAAAAAUGGUACUAUACCUUG
SIVCPZGAB    -----------------------------------------ACUGACAACAUUACAAAUGGC---------------------AUUAUAAUACUGCCAUG
```

Figure 23: This figure again demonstrates the excellent capability of the new `RALIGN` algorithm to obtain better alignments. The number of gaps could has been reduced dramatically as a result of the powerful combined amino acid and nucleic acid based alignment algorithm. Most lines still have only two remaining gaps. 46 gaps (60% less gaps).

5 An Application

5.1 Detecting Conserved Structures in Hepatitis B

Secondary structures of ssRNA viruses or (as in the case of Hepatitis B virus) pregenomic RNA intermediates are known to play an important role in the regulation of the viral life cycle. Much research is performed to find conserved secondary structure elements as part of the genomes of RNA viruses and the pregenomic RNA intermediates of some DNA viruses like the Hepatitis B viruses [58, 69]. Almost all RNA molecules and consequently also almost all subsequences of a large RNA molecule form secondary structures [17, 19, 21, 24, 25]. Since only functional secondary structures are likely to be conserved, a method that detects and highlights conserved structural elements based solely on already available sequence data could be used e.g. to guide experimental mutagenesis or deletion studies. For this reason predicted secondary structures of some known genomic RNA sequences are compared. The development and implementation of computational methods capable of reliably predicting functional structural elements on the basis of sequence information will provide immense benefits in terms of our understanding of the relationship between sequence and structure [16].

In fact, this method for detecting conserved structures depends crucially on the best possible quality of sequence alignments. But even very closely related types of certain viruses can often not be aligned in a satisfying way. Alignment inaccuracies are a substantial problem when dealing with rather diverse sets of viral sequences.

5.2 The Properties of the Hepatitis B Virus

5.2.1 About the Morphology

The Hepatitis B virus infects mammals and members the family of the Hepadnaviridae. Their morphology is spherical, occasionally pleomorphic, 40-48 nm in diameter (40-42nm in the case of HBV, 46-48nm in the case of duck HBV) but with no evident surface projections. The outer 7 nm thick, detergent-sensitive envelope contains the surface antigens and surrounds an icosahedral, 27-35 nm diameter (HBV: 27nm, DHBV: 35nm) nucleocapsid core with 180 capsomers. The core is composed of one major protein species, the core antigen, and encloses the viral genome (DNA) and some associated minor proteins. Some viruses occur with filamentous forms of variable length and 22 nm diameter and others with spherical 16-25 nm structures that lack cores.

The genome consists of a single molecule of non-covalently closed, circular DNA that is partially double-stranded and partially single-stranded. The G+C content is about 48 %. One strand in negative sense (complementary to the viral mRNAs) is full-length (3100 -3400 nt), the other is shorter and varies in size. In the hepatitis B virus genome, which is a member of the Orthohepadnavirus genus, the full-length negative sense DNA strand has a nick at a unique site corresponding to a position 242 nt downstream from the 5' end of the positive sense strand. The ssDNA may represent up to 60 % of the circle. In the second genus (Avihepadnavirus, e.g. duck hepatitis B) of the family Hepadnaviridae the nick in the negative sense DNA is about 50 nt from the end and genomes may be fully double-stranded. The uniquely located 5' ends of the two strands overlap by about 240 nt so that the circular configuration is maintained by base pairing of cohesive ends. The 5' end of the negative sense DNA has a covalently attached terminal protein. The 5' end of the positive sense DNA has a covalently attached 19 nt, 5' capped

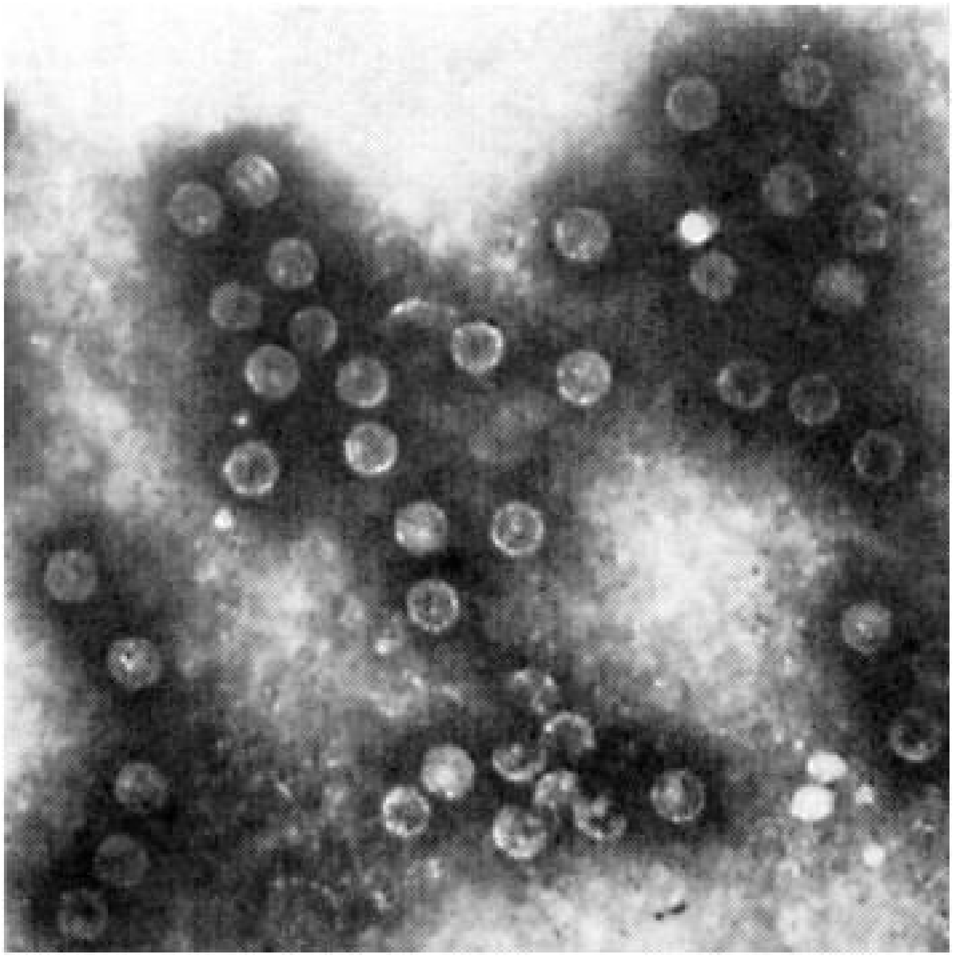


Figure 24: Electron micrograph of hepatitis B virus particles, including virions, 20-nm spheres, and filaments.

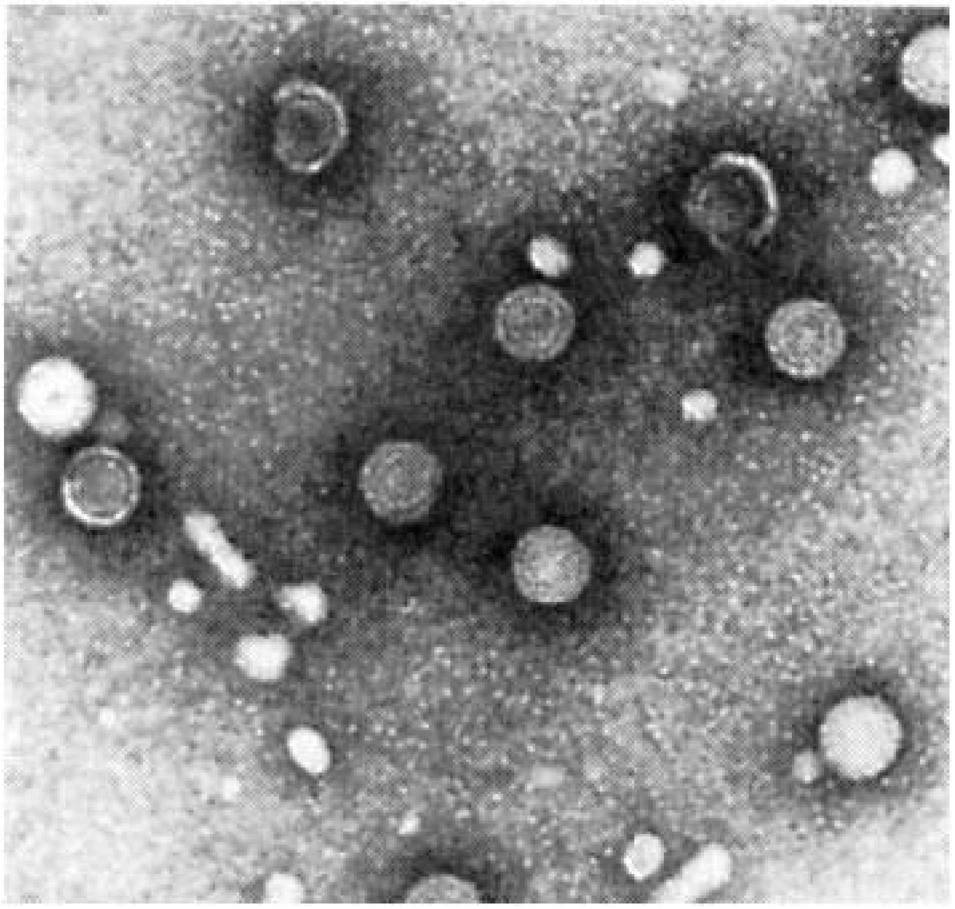


Figure 25: Electron micrograph of virion cores produced by detergent treatment of virions.

oligoribonucleotide primer.

In orthohepadnaviruses the envelope (surface antigen) proteins of virions consist of three groups of antigenically complex proteins: S-proteins (p24, gp27), M-proteins (p33, gp36) and L-proteins (p39, gp42). All three have common carboxy termini and differ in amino termini (due to different sites of translation initiation) and in the presence and form of glycosylation. In hepatitis B virus the S-proteins appear to have the same amino acid composition and, beside this, gp27 has a single glycosylation site that is shared by the M-proteins p33 and gp36 which are composed of p24 with additional 55 amino acids and a further glycosylation site. The L-proteins contain about further 120 amino acids and their N-termini are myristylated.

The virion core is composed principally of the 22 kDa core antigen, a phosphoprotein. Enzymes associated with virions include a protein kinase and reverse transcriptase with RNA- and DNA- dependent DNA-polymerase and RNase H activities (P-gene products). Other functional components include the terminal protein covalently attached to the 5' end of the full length DNA strand. This terminal protein has been shown to be a component of the about 90 kDa P gene product.

The envelope lipids of virions are derived from the host cell membranes.

5.2.2 The Genomic Organization of Hepadnaviruses

The hepatitis B virus genome has four partially overlapping genes (S, C, P, X), all oriented in the same direction. The duck hepatitis B virus (genus Avihepadnavirus) consists of three genes (S, C, P). There appear to be no intervening sequences. The S-gene ORF codes for the surface antigens. In the S-gene the p24 protein is preceded by pre-S2 which, in turn, is preceded by pre-S1. Each has an in-frame ATG start codon for the initiation of protein synthesis of all surface antigens (S-, M- and L-proteins). The C-gene ORF specifies the core protein. This is preceded by a short pre-C region that varies in size between different viruses and is larger in avihepadnaviruses than in mammalians.

The P-gene covers 80 % of the entire genome and overlaps the other three ORFs. It codes for the reverse transcriptase, with DNA polymerase and RNase H activities, and the genome-linked terminal protein. Finally, the X-gene specifies a protein with a probable transactivation function. It varies in size among the HBV serotypes.

After having entered the hepatocytes, the single-stranded regions are made full length double-stranded DNA. The terminal protein of the negative strand is removed, also the terminally redundant region and the oligoribonucleotide of the positive strand, and the DNA is converted into a covalently closed circular DNA by ligation.

After location of the double-stranded DNA into the nucleus of infected cells, transcription of viral mRNAs starts enhanced by the X-protein. Transcription yields various species of mRNAs with various lengths which code for the viral proteins. Following transcription, translation of the gene products ensues in the cytoplasm of the infected cells. The 3.4 kb pre-genome is transcribed which is greater in size than the genome, because of terminally redundant sequence parts. The pre-genome is initiated near the start of the pre-C ORF and terminates about 100 nucleotides downstream of the pre-C initiation site after making a complete copy of the genome (240 nucleotides downstream in the case of avian hepatitis B virus). The mRNAs are unspliced and are made from distinct promotors. Two regions of the HBV genome have transcription enhancer activities, another is similar to glucocorticoid responsive elements.

Integration of viral DNA into the host genome is possible, but singly integrated forms cannot serve as templates for the synthesis of the 3.4 kb pre-genome (which requires circularized or concatenated copies of integrated DNA).

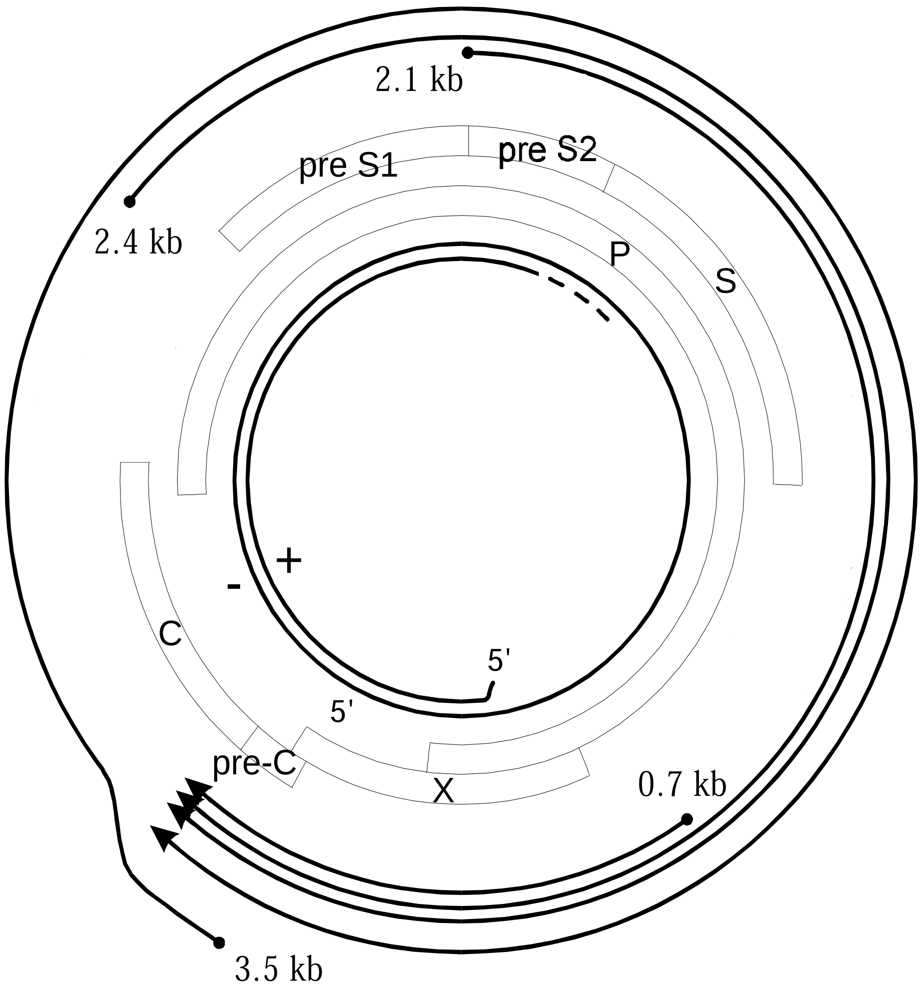


Figure 26: Diagram of the genome organization of hepatitis B virus indicating the DNA arrangement, the positions of the four open reading frames (C, P, S, X) and the mRNA transcripts.

Current evidence indicates that following the generation of a covalently closed circular DNA and synthesis of the 3.4 kb pre-genome, this RNA associates with viral core particles where it serves as a template for synthesis of minus-strand DNA by reverse transcription using a protein primer. Then the minus-strand DNA serves as a template for plus-strand DNA synthesis. The plus-strand DNA strand is incomplete in most core particles at the time of virion assembly and release from infected cells. And finally the carboxy-terminal domain of C-protein probably is required for packaging the DNA.

5.2.3 The ϵ-Structure: a 5'-proximal Encapsidation Signal

After reverse transcription of the terminally redundant RNA pregenome, the specific packaging of the transcript into viral capsids is mediated by interaction of the reverse transcriptase, P protein, with the 5'-proximal encapsidation signal ϵ [53]. ϵ-function is correlated with the formation of a hairpin containing bulges and a loop. This interaction is not only central to pregenome packing but also to capsid assembly. Furthermore, it is essential for initiation of reverse transcription. There is striking assymetry in the importance of primary sequence in the 5'- and the 3'-part of the ϵ-structure. The motif CU is important in the proximal bulge position. It has been proposed that these nucleotides are in close contact with P protein [70]. Deletion of the bulge prevents incapsidation.

5.3 Results: Conserved Secondary Structure Elements

The combined amino acid and nucleic acid based alignment algorithm as implemented in `RALIGN` is especially useful when used with rather closely related sequences. In these cases the reduction of gaps bears quite good alignment results. Therefore, as an example for an application human hepatitis B virus genomes were used. With 16 various pregenomic RNA sequences the com-

AB014370	E00010	AB014366	HBD50521
AB014360	AB014361	AB014362	HBVAYWE
HBVAYWGEN	HBVAYWMCG	HBV131567	HBV131568
AF046996	HHVBFFOU	HBVADW4A	HVHEPB

Figure 27: The 16 input sequences: human hepatitis B virus pregenomes.

bined amino acid and nucleic acid based alignment was done.

The redundant terminal part of the pregenomes was attached to the sequence end. Using `RALIGN` led to an alignment which contains very few gaps although the input sequences were not the most closely related. The 16 input sequences are members of 6 different subgroups of human hepatitis B virus which contain some different sequence parts. And this resulting alignment is a rather good input for algorithms which depend on good alignment results like `alidot` and `pfrali`. Using `alidot` and `pfrali` led to dot plots which show a lot of conserved secondary structure elements. In most cases `alidot` and `pfrali` gave the same results, but there are also some differences in the dot plots. The dot plots then were used to construct the secondary structure graphs and the Hogeweg mountain representations. Before producing the secondary structure graphs the consensus sequences were calculated.

The colour codes are as follows:

`red`: all sequences have the same two nucleotides

`ocre`: two types of base pairs occur

`green`: three types of base pairs occur

`turquoise`: four types of base pairs occur

`blue`: five types of base pairs occur

`violet`: all six types of base pairs occur

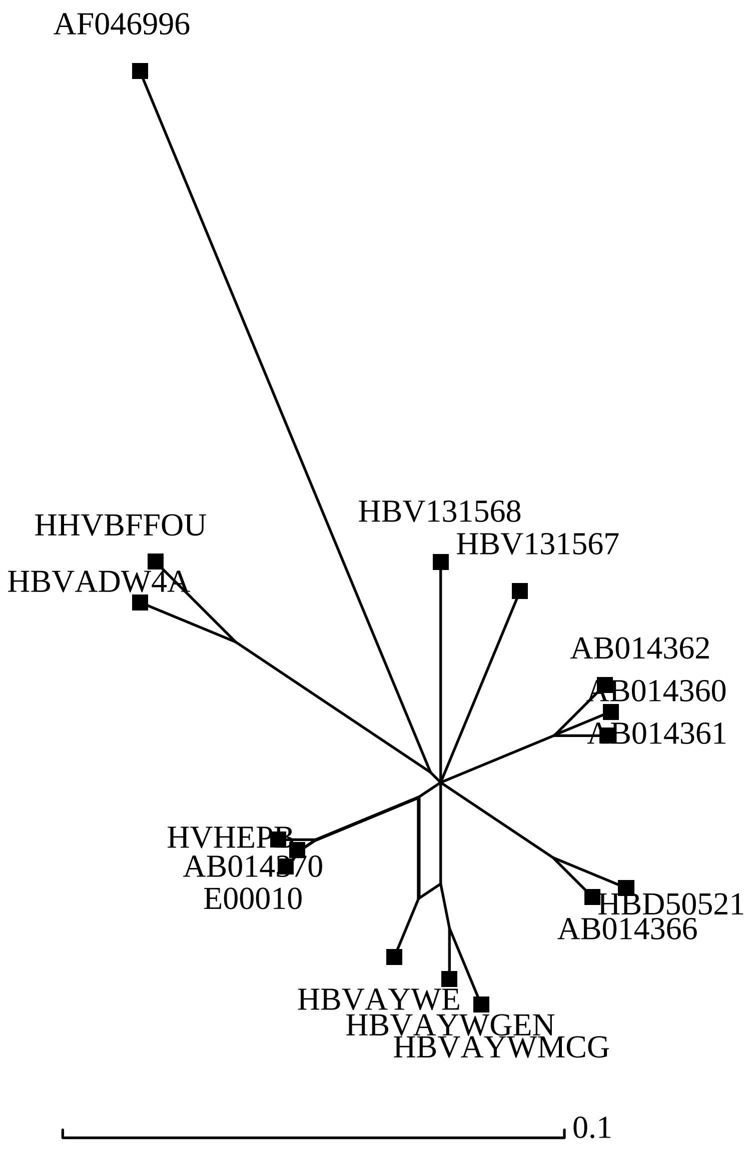


Figure 28: Phylogenetic relationships among the 16 human HBV sequences used here. The tree was produced using `Splitstree-2.4.1` [47], The graph encapsulates 83.3% of distances, the rest is assigned to the so-called *split-prime* part [4] and corresponds to noise in the data.

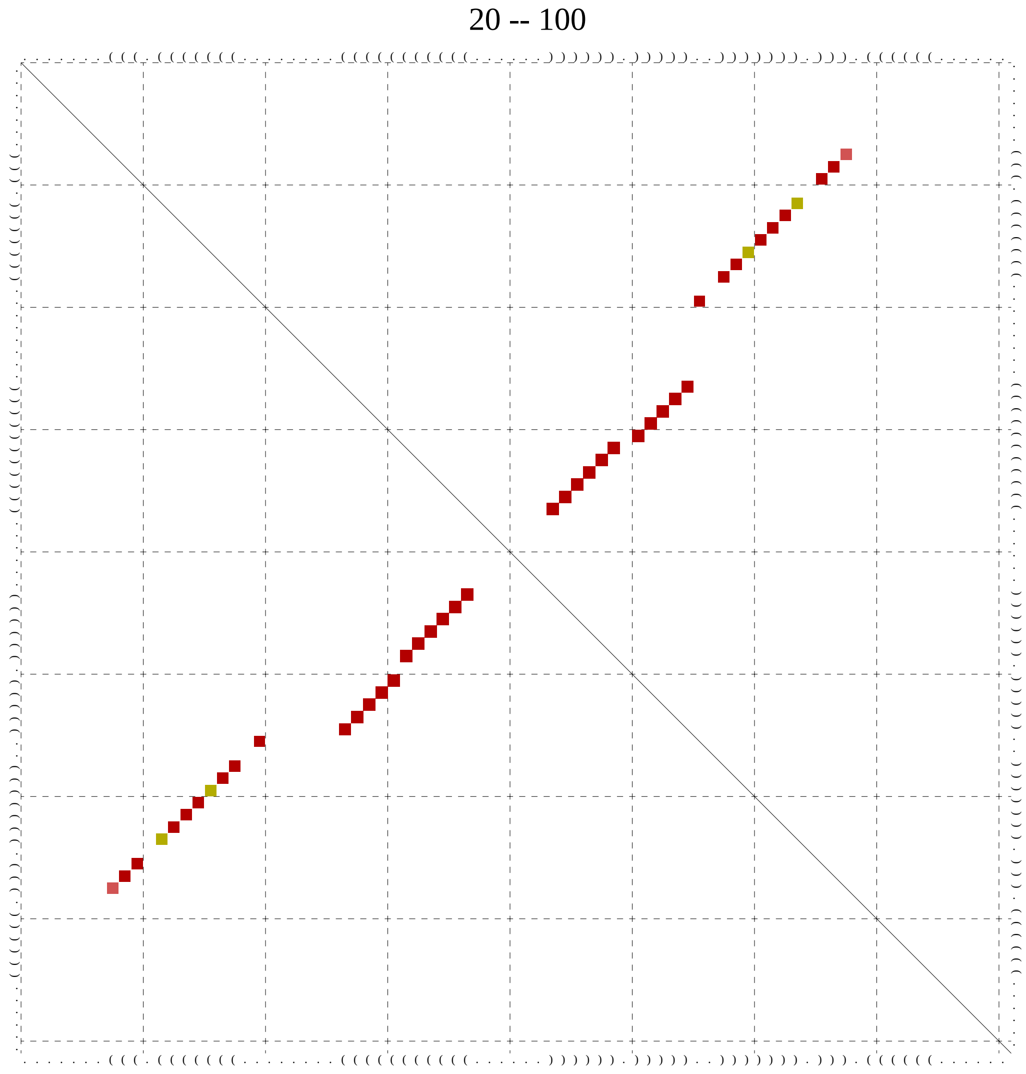


Figure 29: This figure shows the dot plot of the well known ϵ-structure in the hepatitis B pregenome. As the color code shows almost all sequences have the same two nucleotides. This indicates that the sequences are highly conserved. But twice two types of base pairs occur as shown by the ocre dots. These consistent mutations strongly support the pairing at their positions. And we have one inconsistent mutation at the end of the stem structure.

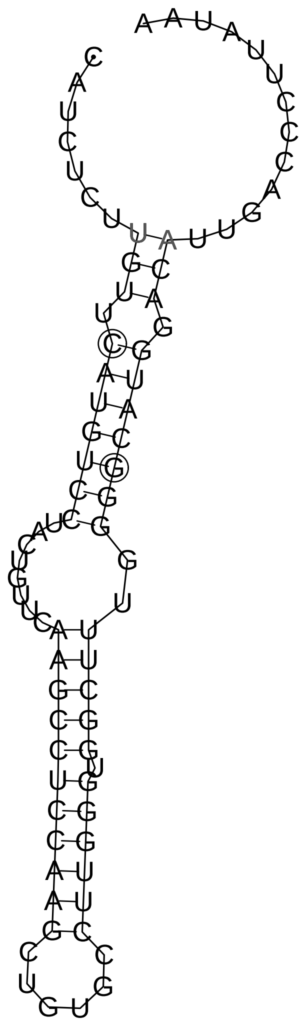


Figure 30: Conventional secondary structure representation of the ϵ-structure at the beginning of the sequence. Nucleotides 20 - 100 are shown. Consistent mutations are indicated by circles. Also one incompatible sequence is in the data setaffecting the terminal U-A pair of the structure (shown in gray).

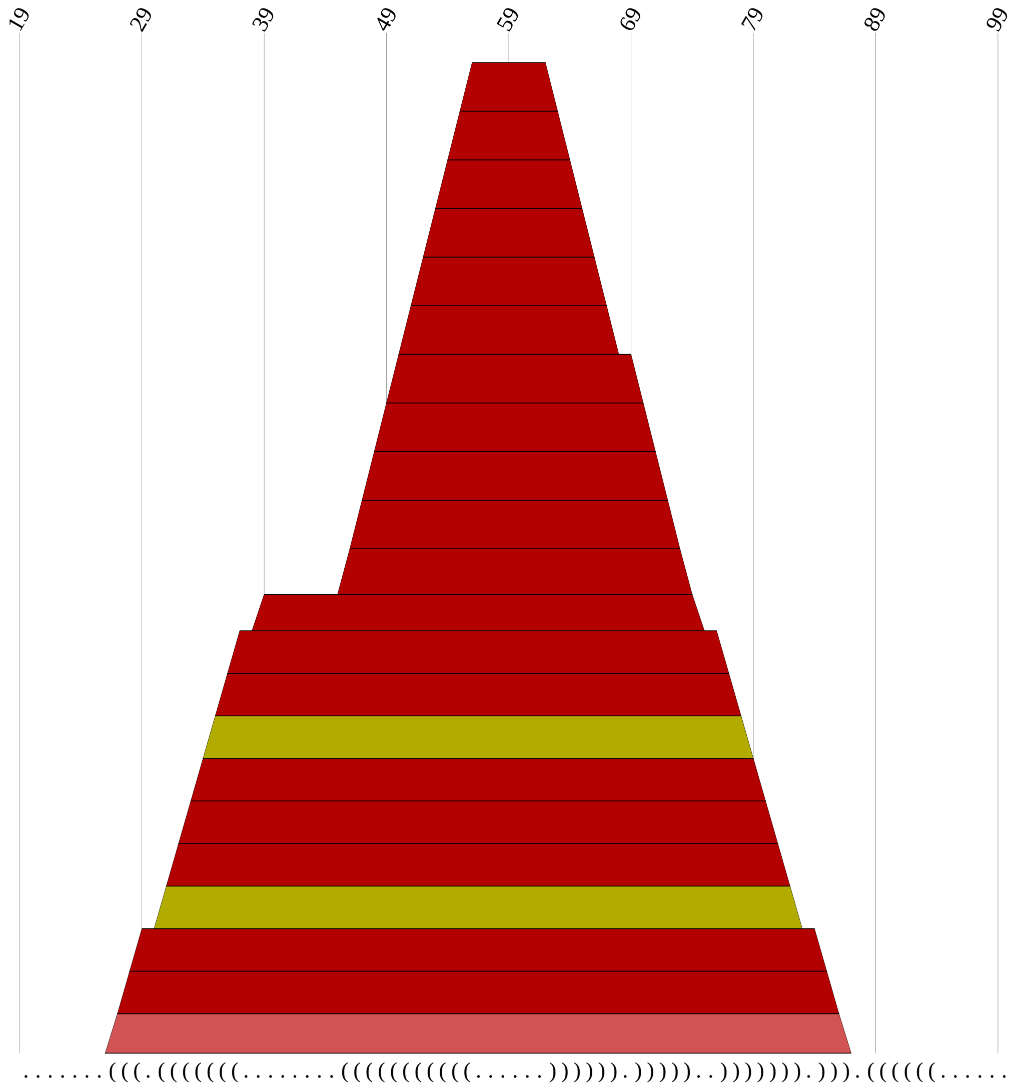


Figure 31: The Hogeweg mountain representation of the ϵ-structure. Two ocre coloured consistent base pairs strongly support the structure.

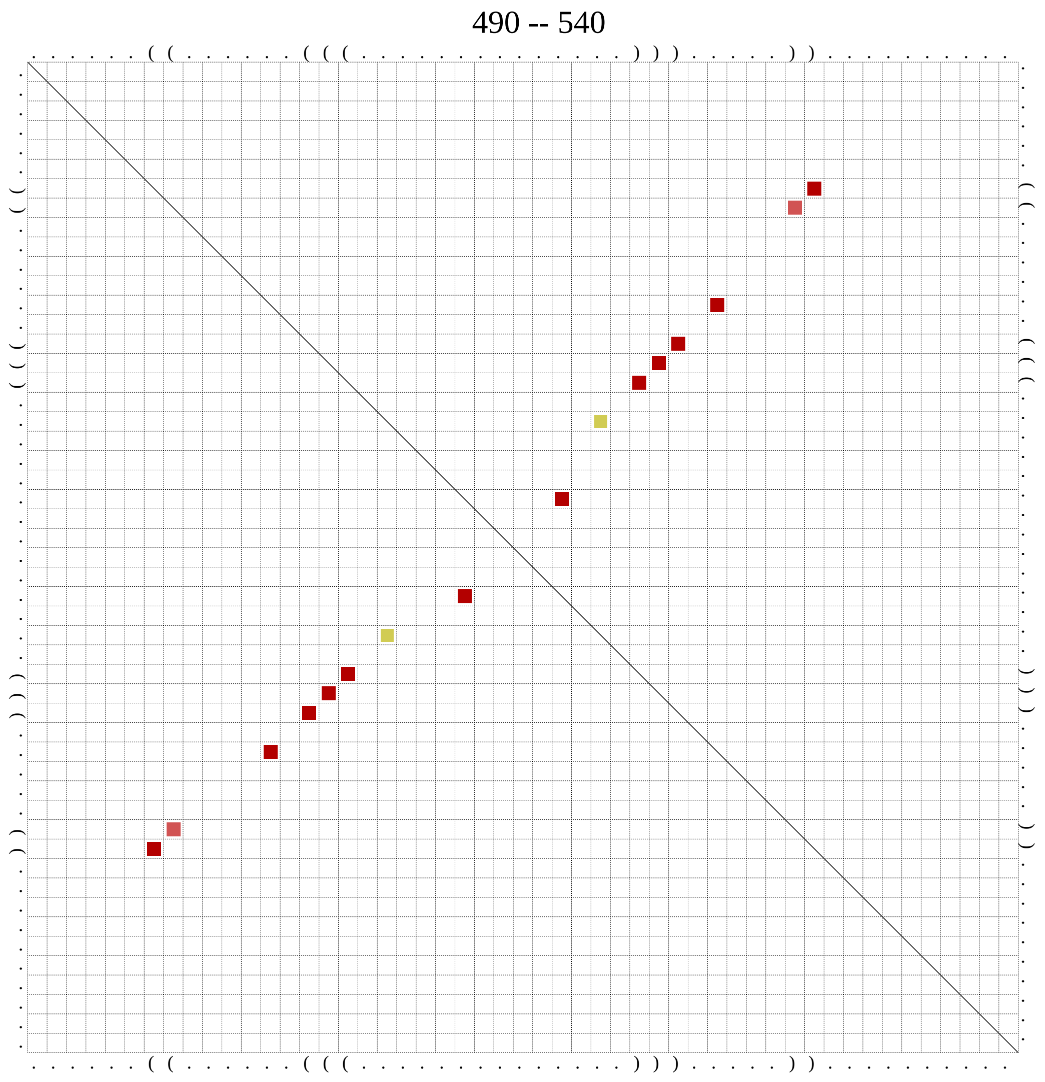


Figure 32: This `alidot` derived dot plot shows the region 490 - 540. At one position two types of base pairs occur. We have a consistent mutation at this position. Also at this certain position one sequence is incompatible. In addition, we have some more incompatible sequences in the dot plot indicated by the pale colours. Data such as these probably do not correspond to functional structures.

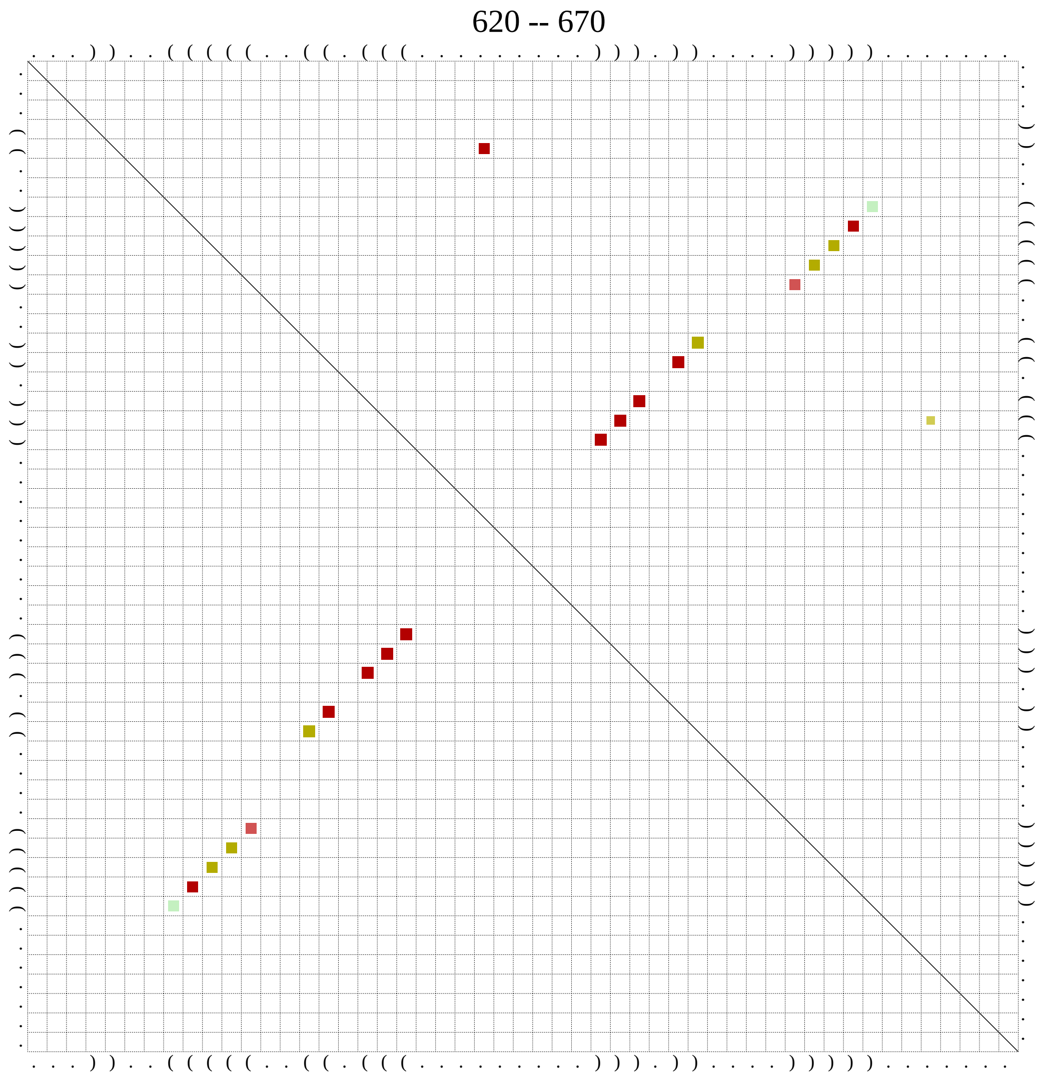


Figure 33: This is an example for a conserved secondary structure which is detected by **alidot**, but not by **pfrali**. Nucleotides 620 - 670 are shown. At three positions two types of base pairs occur (indicated by the ocre coloured dots). At one positions four types of base pairs are found (turquoise colour). The three positions with consistent mutations (ocre dots) strongly support this secondary structure element.

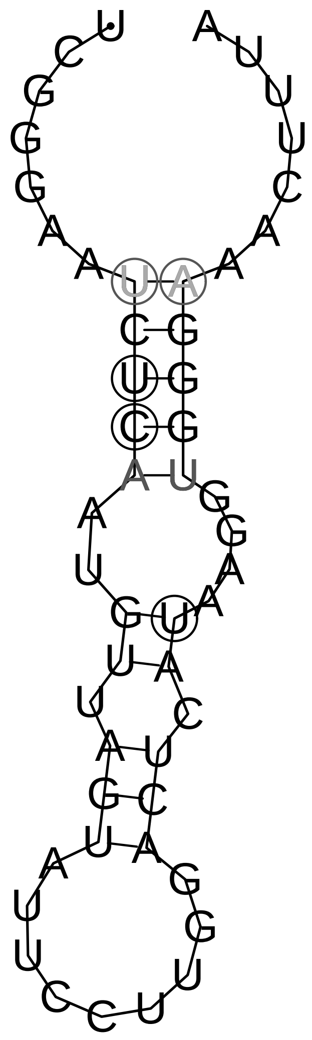
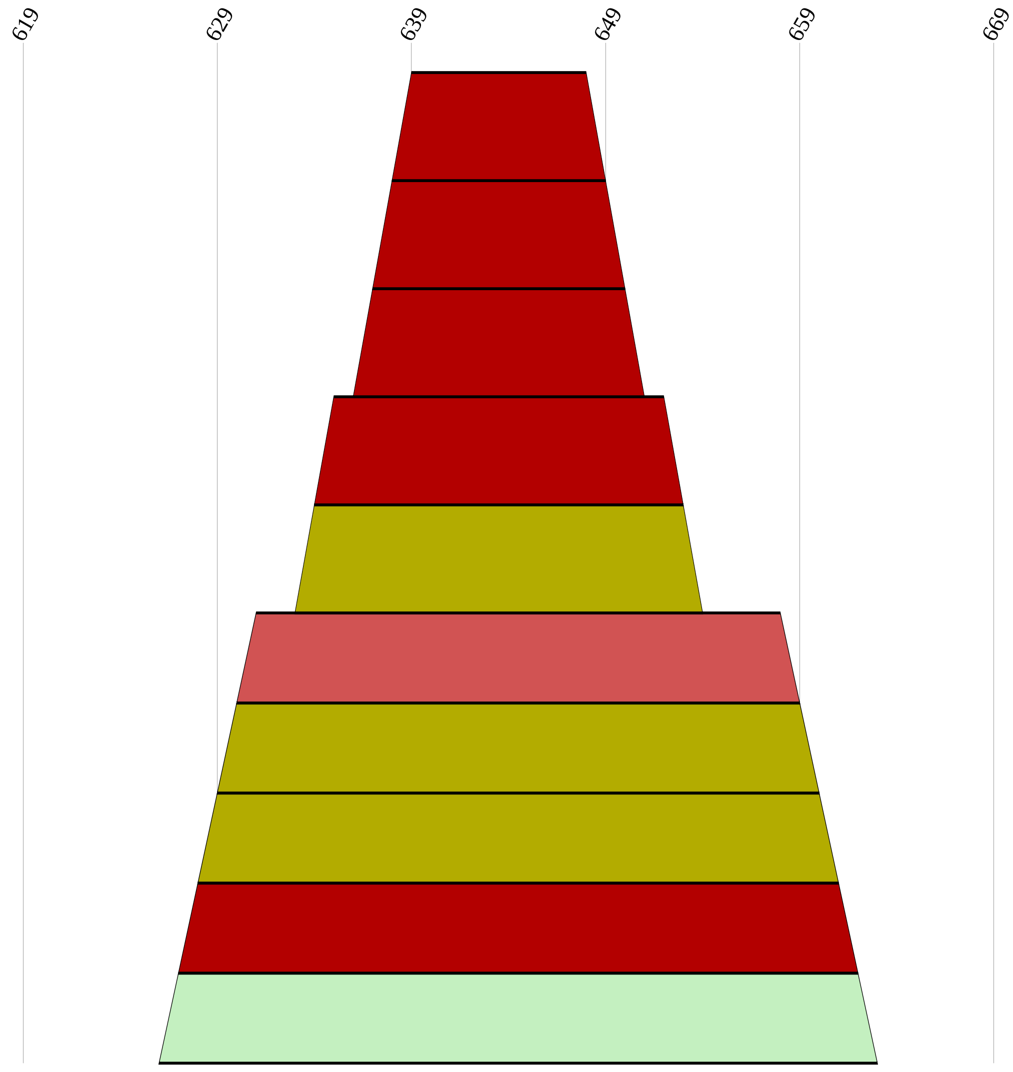


Figure 34: The conserved structure found by `alidot` but not by `pfrali`. Position 620 - 670. Compensatory and consistent mutations are idicated by circles.



...))..(((((..((.(((.........))).))....))))).......

Figure 35: The Hogeweg mountain representation of the region 620 - 670 as derived from `alidot`. The ocre and turquoise coloured regions show the positions of the compensatory and consistent mutations which strongly support the secondary structure shown before.

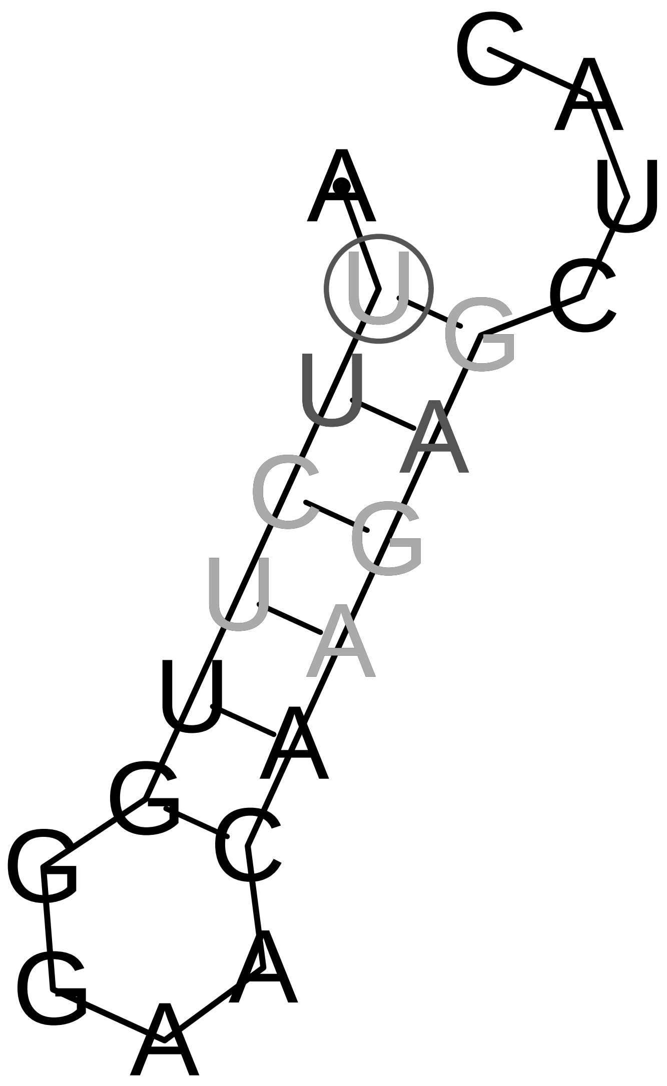


Figure 36: The predicted secondary structure at position 1010 - 1030 as predicted by `alidot`. One consistent mutation, but with one base pair one sequence is incompatible, and with three base pairs two sequences are incompatible. This is an example for a predicted structure which is not credible.

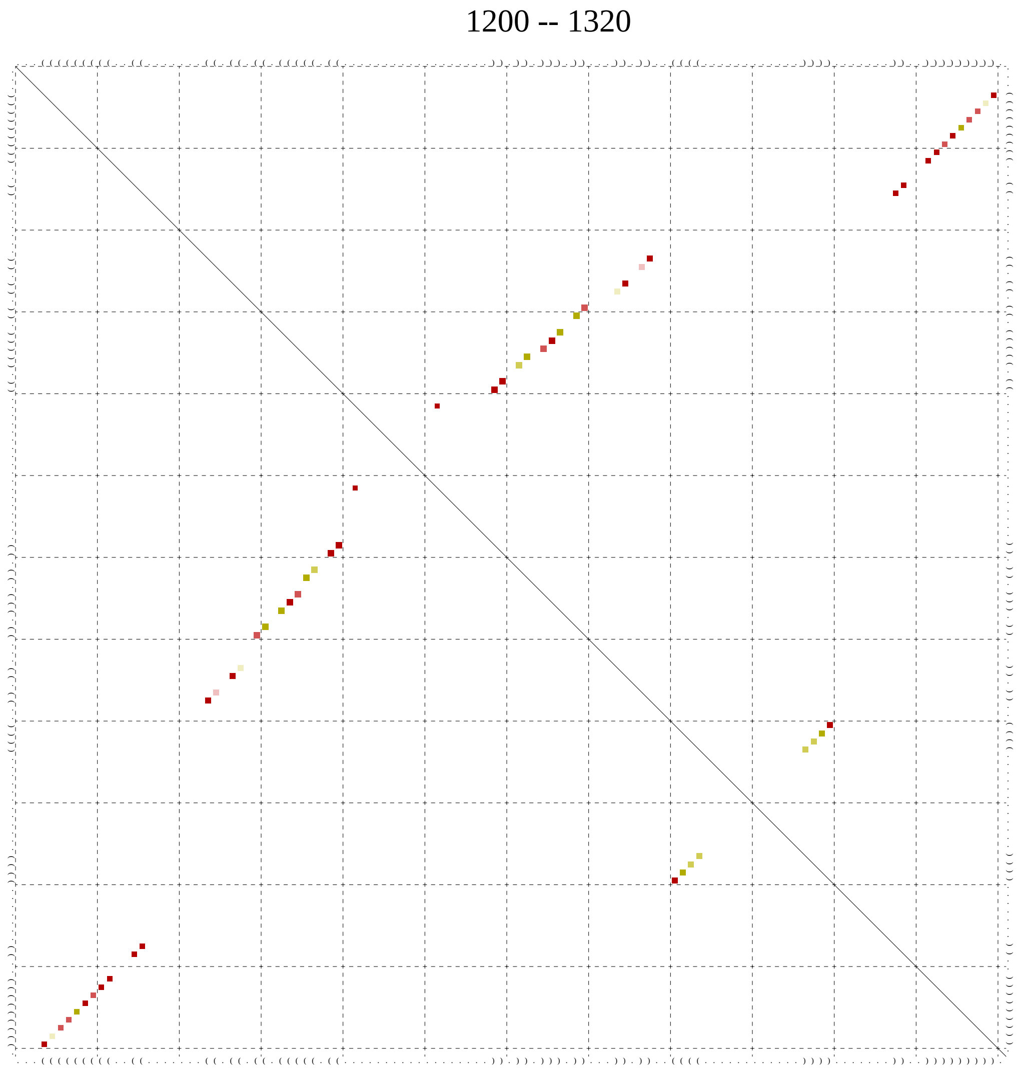


Figure 37: The dot plot of the region 1200 - 1320 as produced by `alidot`. A lot of ocre dots indicating that two types of base pairs occur. And also some dots with pale colours which show that one or two sequences in the sample are inconsistent. Here we have some consistent mutations which strongly support the predicted secondary structure elements. The ENH enhancer element is located in this area. Whether the potentially conserved structural elements detected here are related to the functioning of ENH is unknown.

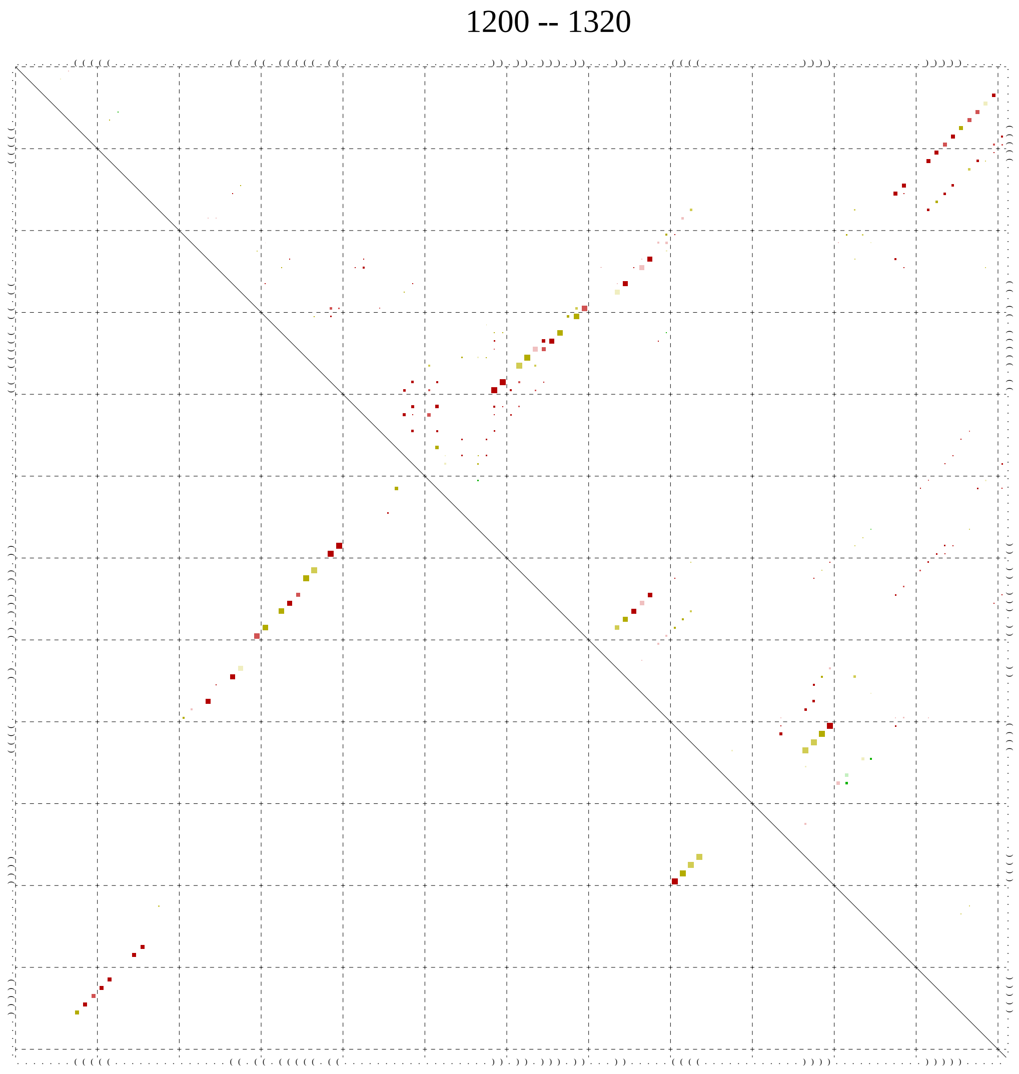


Figure 38: The dot plot of the same region as before, but produced by `pfrali`. The large predicted structures are essentially the same result as derived by using `alidot`.

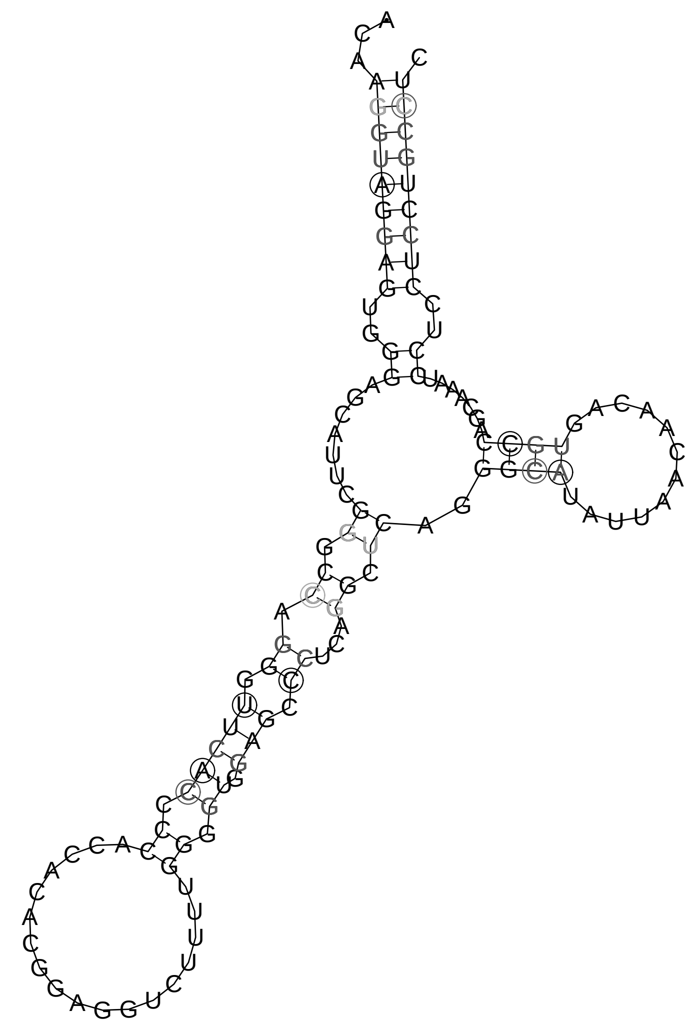


Figure 39: The secondary structure of the region 1200 - 1320 as predicted by `alidot`. Consistent mutations which strongly support the predicted base pairs are indicated by circles. But also here some incompatible sequences exist.

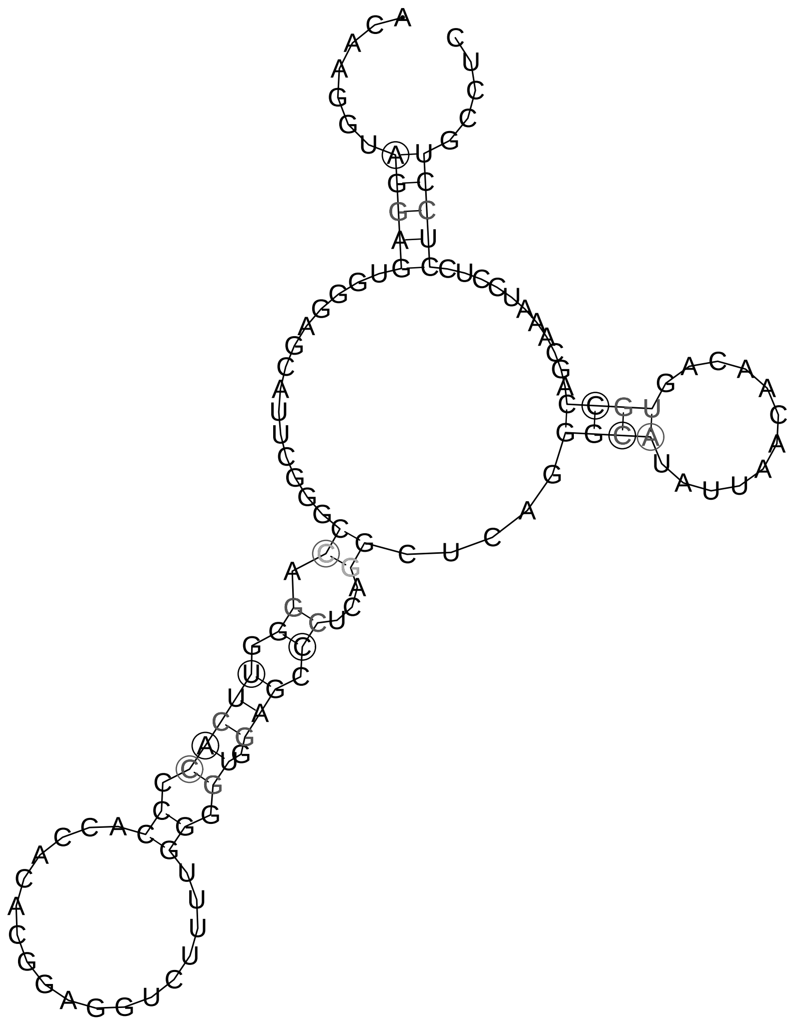


Figure 40: The secondary structure of the same region as before, but predicted as a result of using `pfrali` instead of `alidot`. The main features of the organization of the secondary structure elements have the same appearance, but the central bulge is much larger and the hairpin on top of the figure is not as long as predicted by `alidot`. The two predictions are consistent in the sense that there are no conflicting base pairs.

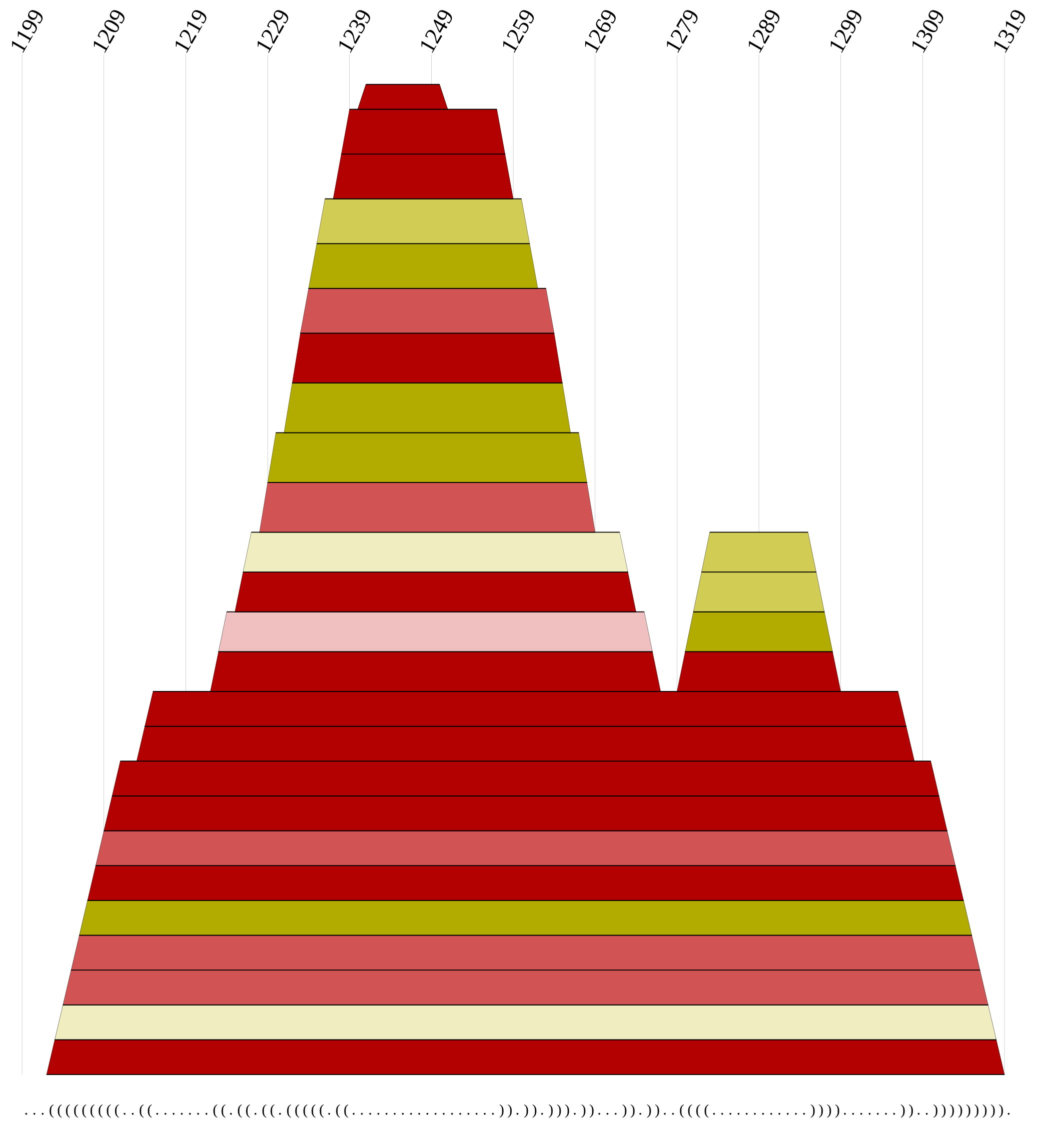


Figure 41: The Hogeweg mountain representation of the region 1200 - 1320 as derived from `alidot`.

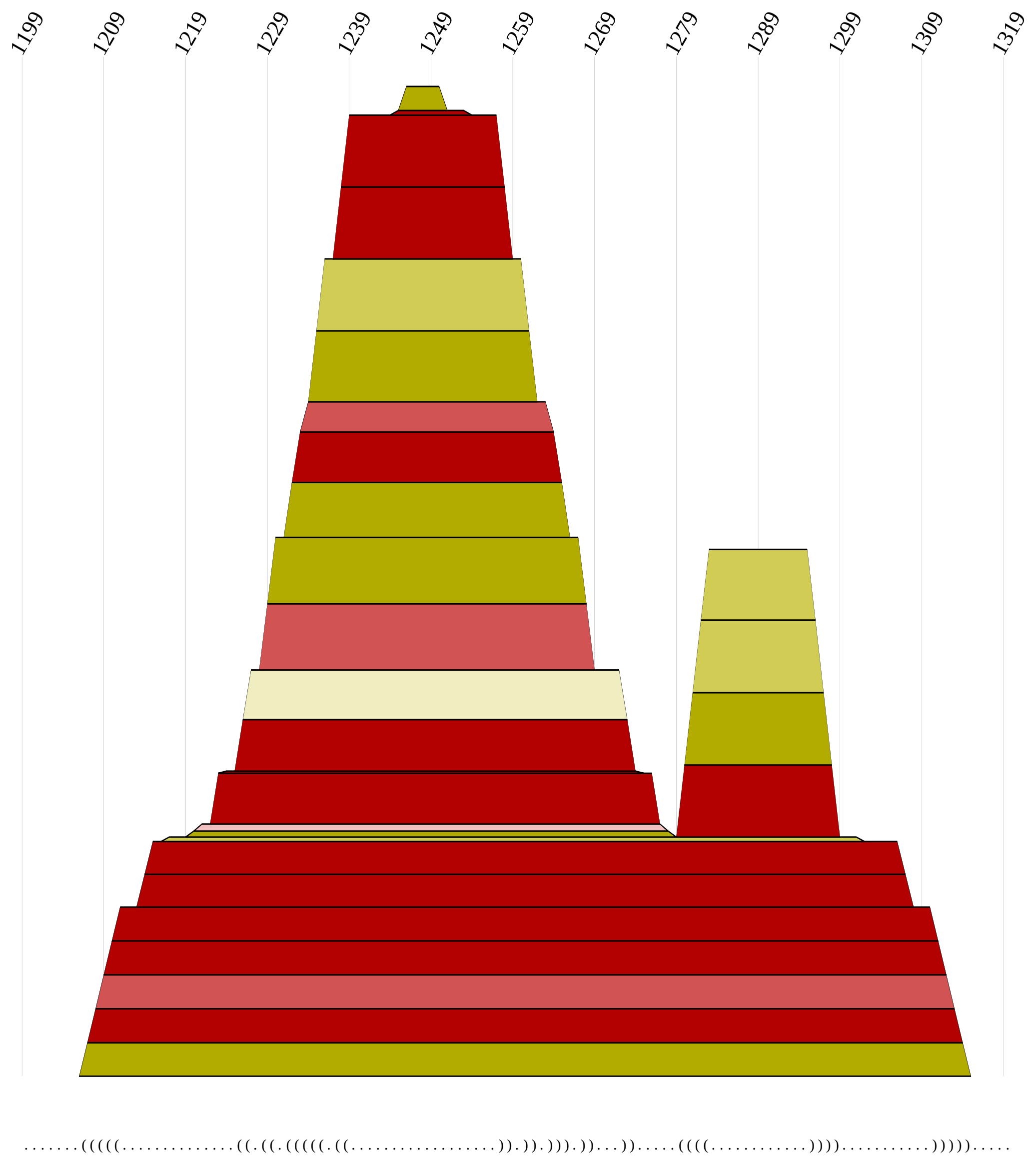


Figure 42: The Hogeweg mountain representation of the region 1200 - 1320 as derived from `pfrali`.

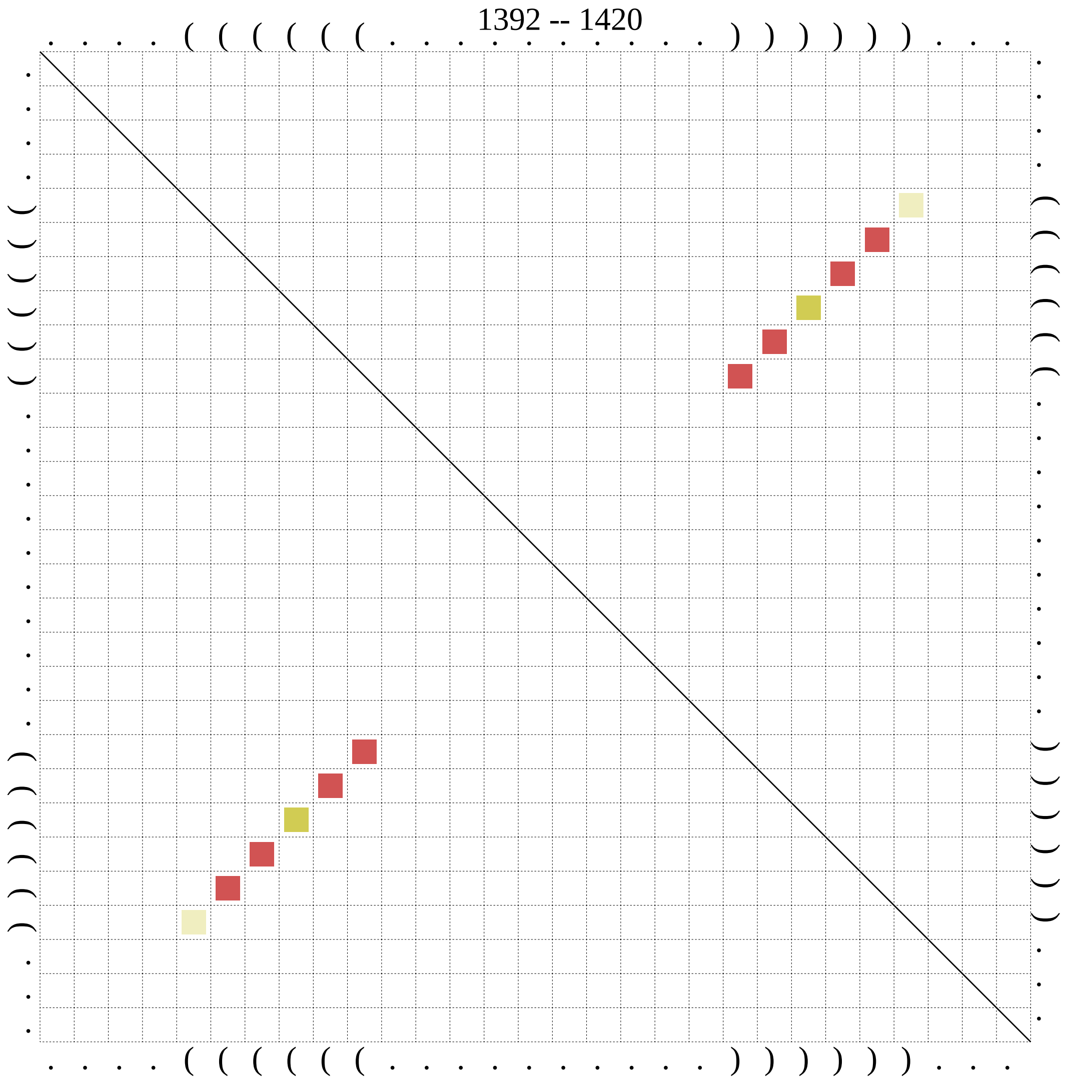


Figure 43: The position 1392 - 1420 as predicted by `alidot`. Six conserved base pairs were found, at two positions two types of base pairs occur; but the pale colour indicates that at each position one sequence in the sample is incompatible, at one position two sequences are incompatible. And further investigation led to the result that one of the input sequences has a deletion at this position.

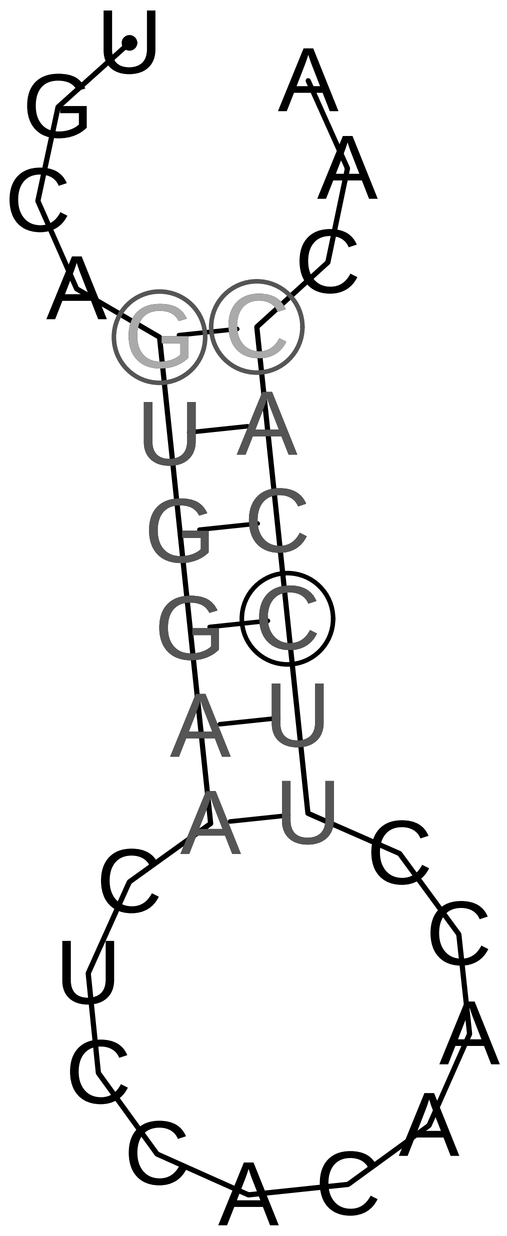


Figure 44: The secondary structure as predicted through `alidot`, strongly supported by one compensatory mutated base pair and one consistent mutation. But a deletion in one of the input sequences leads to pale grey colour of the base pairs. `Pfrali` shows no secondary structure at this position.

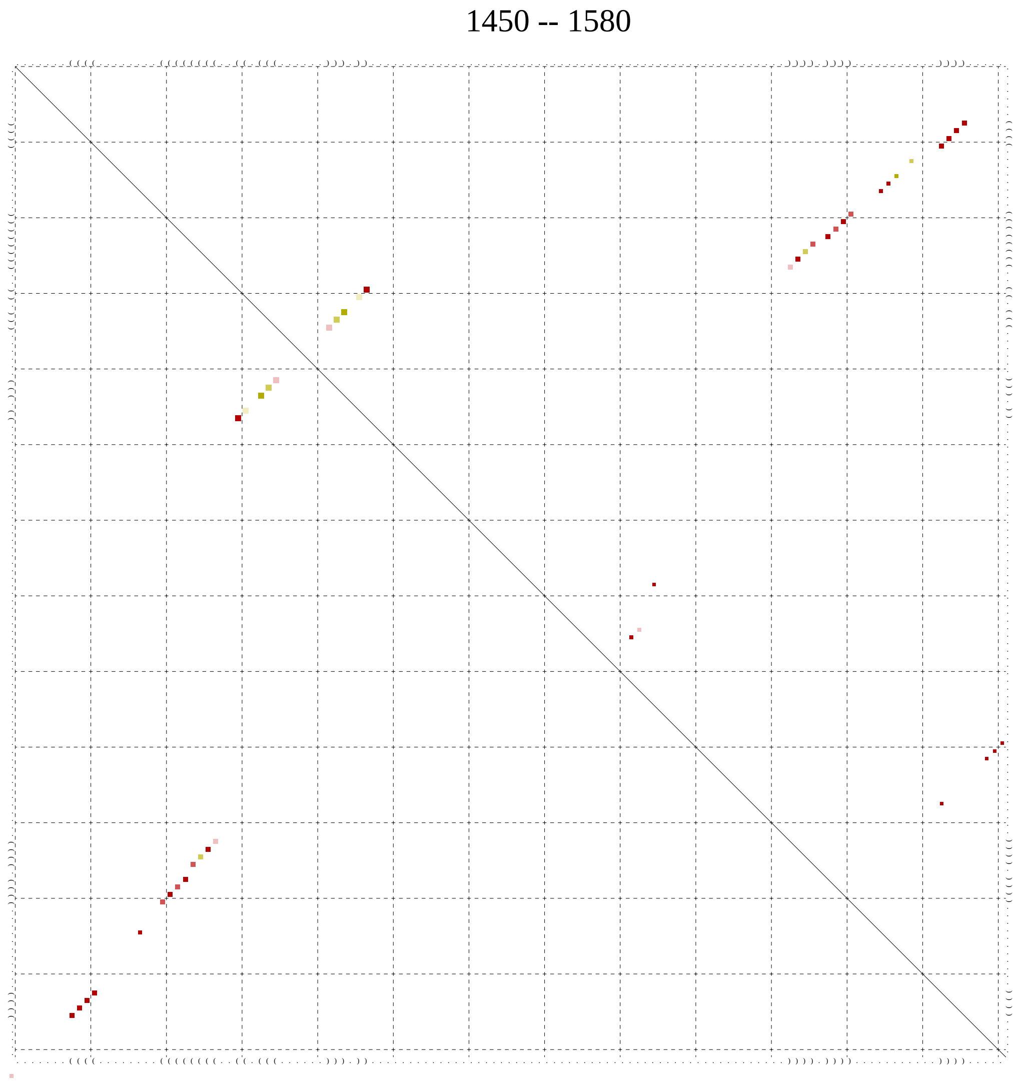


Figure 45: The region 1450 - 1580 as predicted by `alidot`. Various consistent mutations, indicated by the ocre dots (two types of base pairs) support the prediction of this found secondary structure element. But there is a lot of incompatible sequences in the data set. This predicted secondary structure element is an example for a not credible prediction.

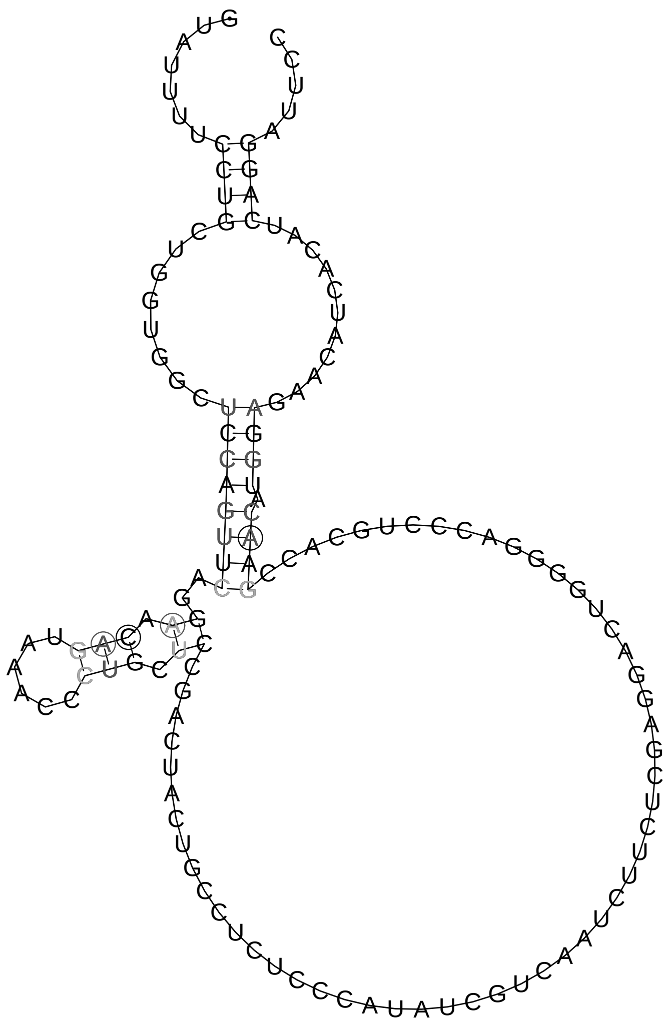


Figure 46: The predicted secondary structure of the consensus sequence. Some consistent mutations support this structure. But there are also some incompatible sequences in the input data set as indicated through the pale grey colours. Not credible.

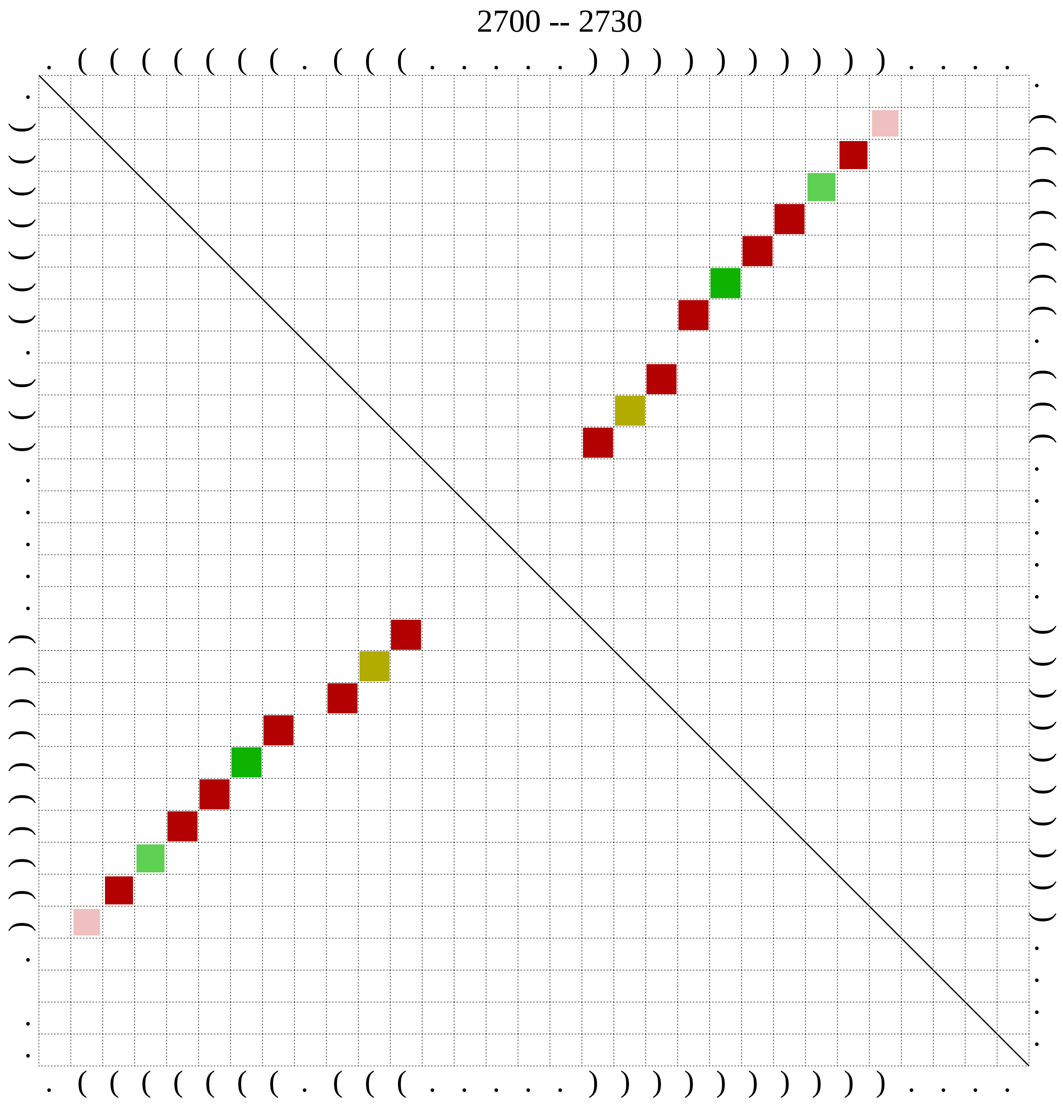


Figure 47: This `alidot` derived dot plot shows the region 2700 - 2730. At two positions three types of base pairs occur as compensatory mutations (green dots). One of these positions has also one incompatible sequence in the input data set. At one position two types of base pairs occur as indicated by the ocre dot. This is a strongly supported predicted secondary structure element.

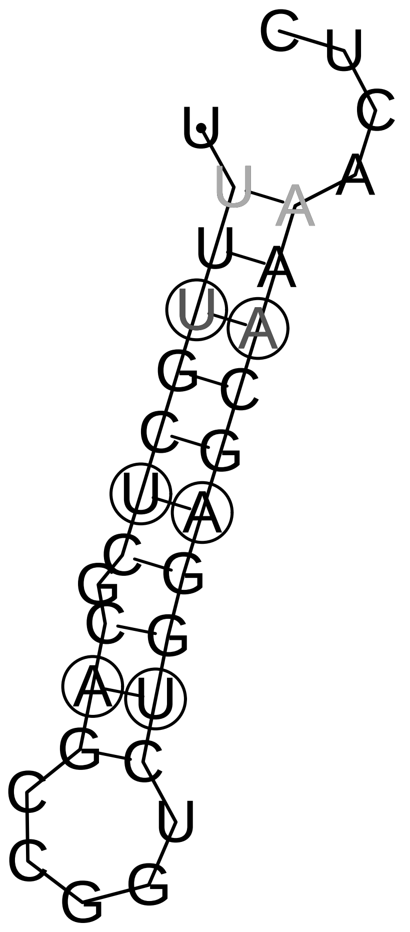


Figure 48: Three compensatory mutations strongly support this secondary structure element. Position 2700 - 2730. Only few inconsistent sequences are detected.

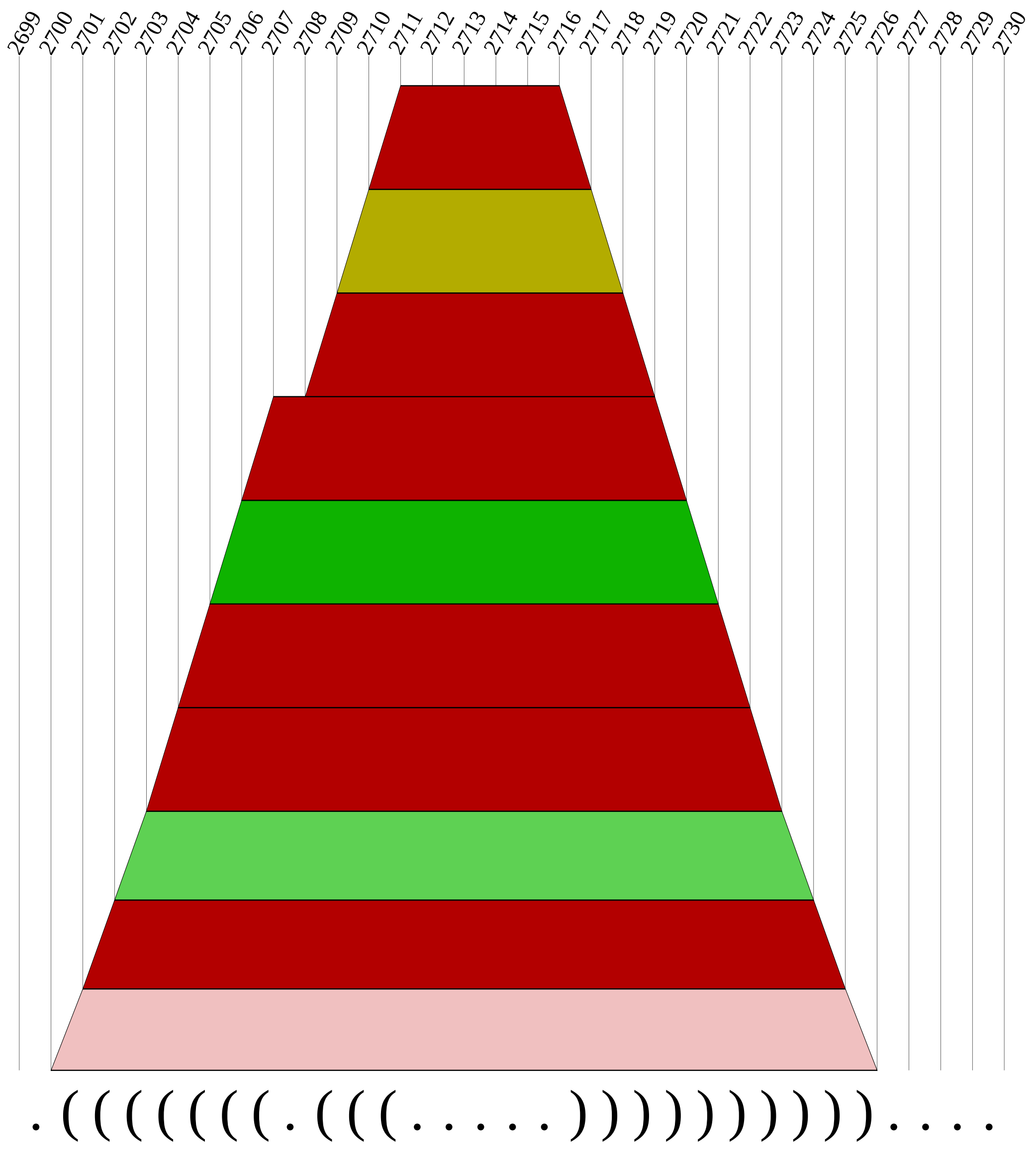


Figure 49: At two positions three types of base pairs occur, at one position two types. Only at two positions inconsistent sequences in the input data set.

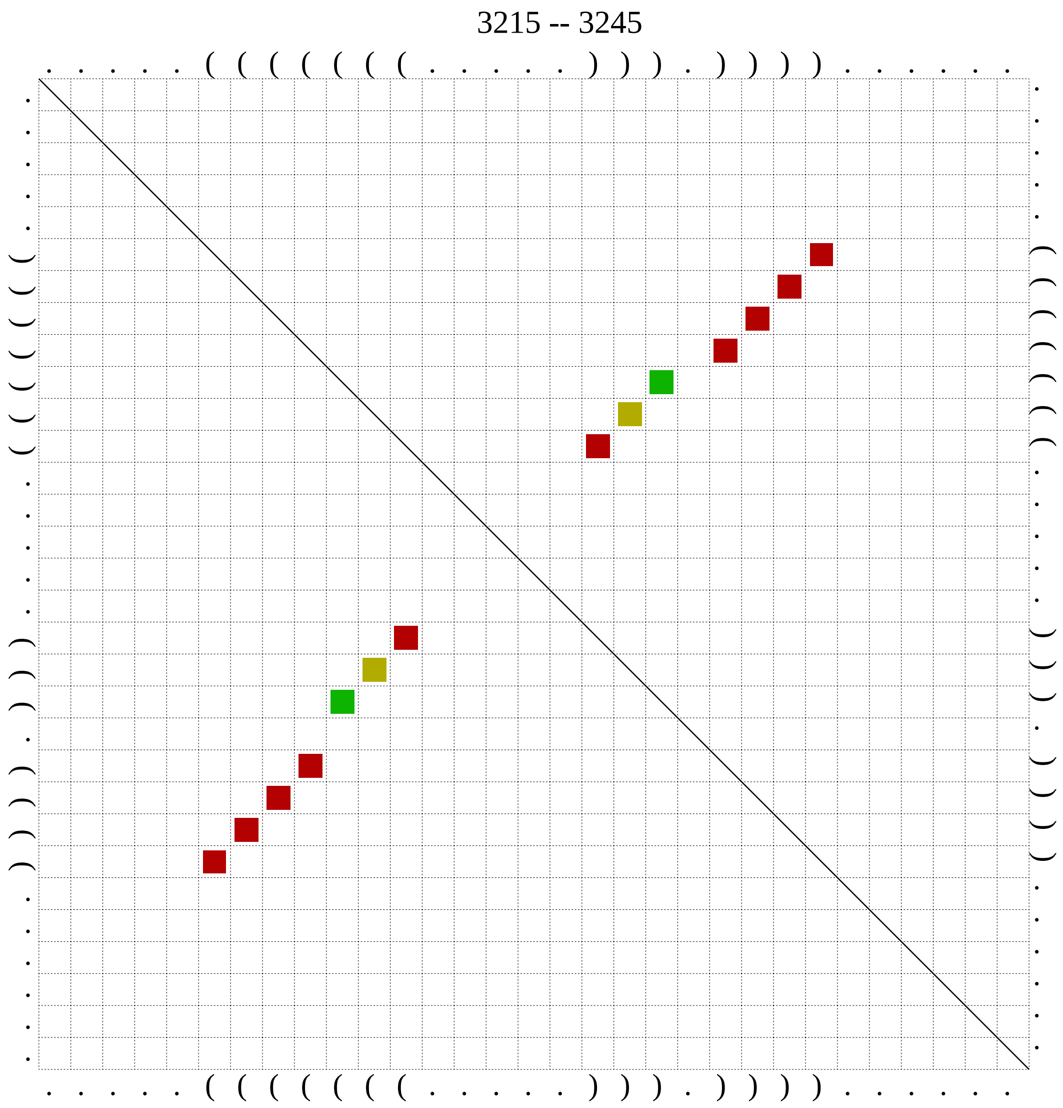


Figure 50: A secondary structure element with three types of base pairs at one position and two types at another (green and ocre dots).

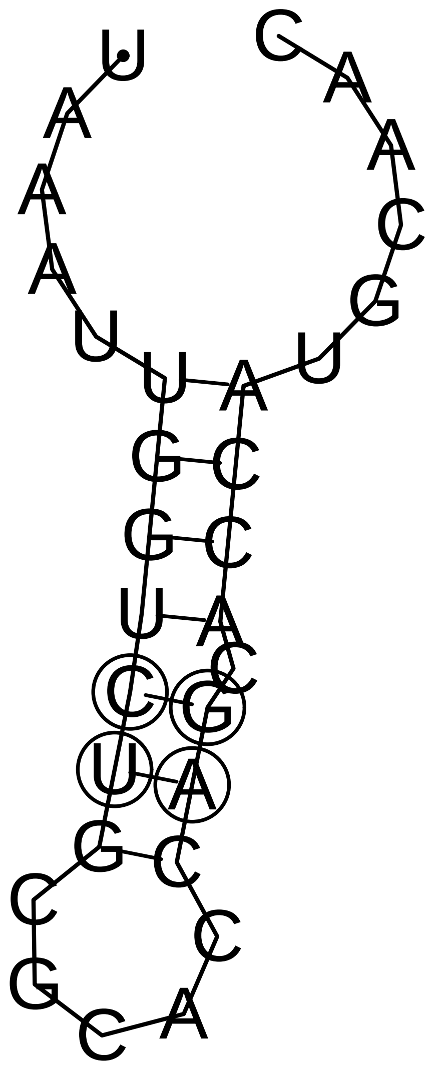


Figure 51: Two compensatory mutated base pairs strongly support this secondary structure element at the position 3215 - 3245.

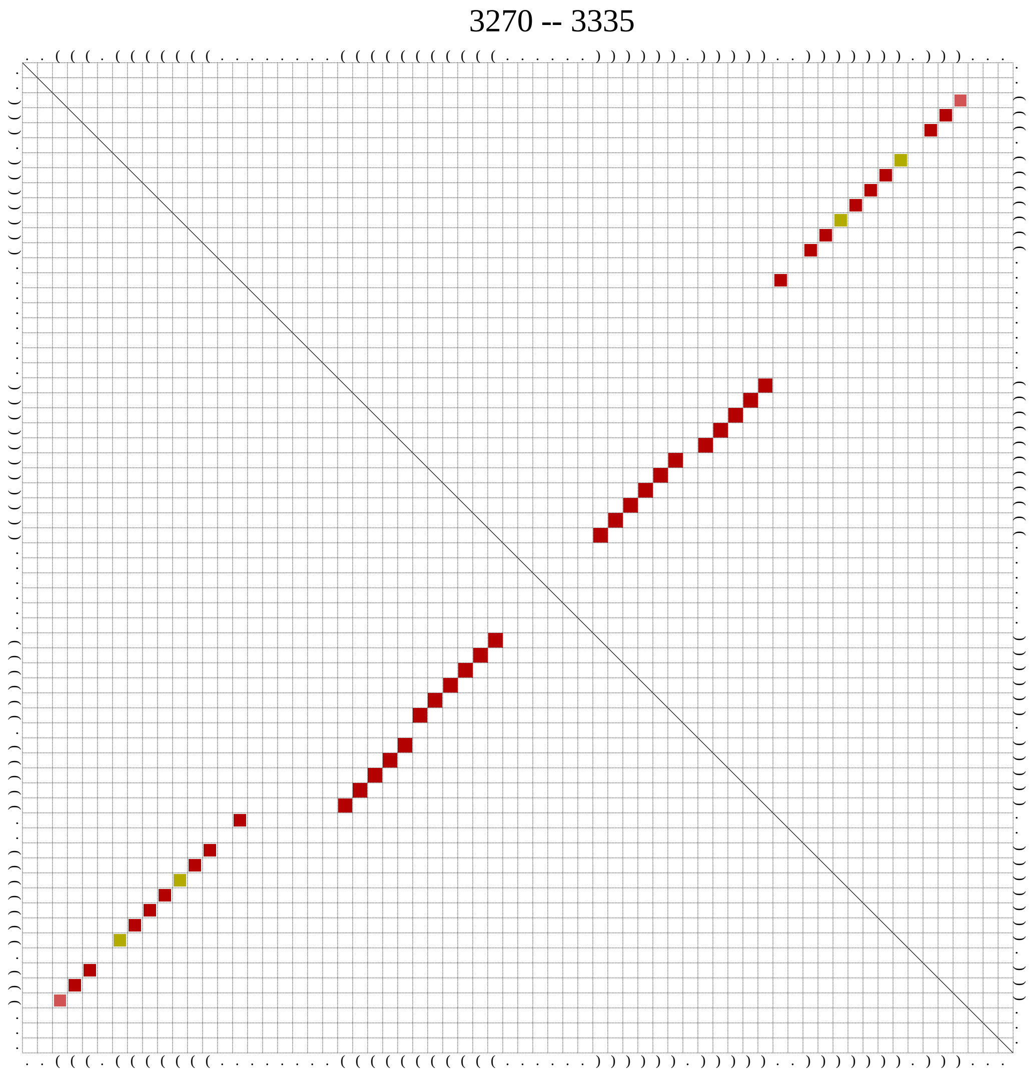


Figure 52: Because of the redundant nature of the HBV genomes this predicted secondary structure element represents again the same sequence as the known ϵ-structure. This structure is is exactly the same as the predicted secondary structure of the ϵ-structure at the beginning of these sequences. Two positions with two types of base pairs.

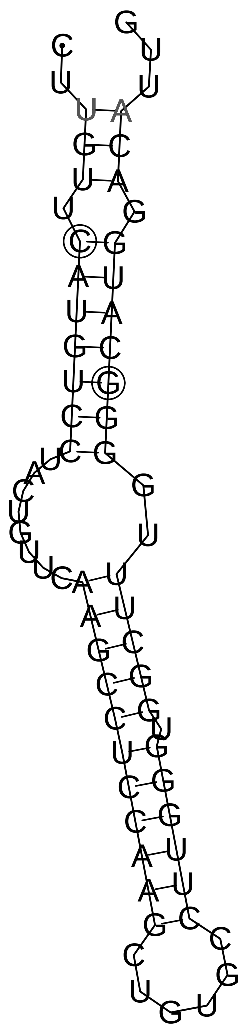


Figure 53: The predicted secondary structure element for position 3270 - 3335. It is the same as the ϵ-structure at the beginning of the sequence. And it has the same sequence.

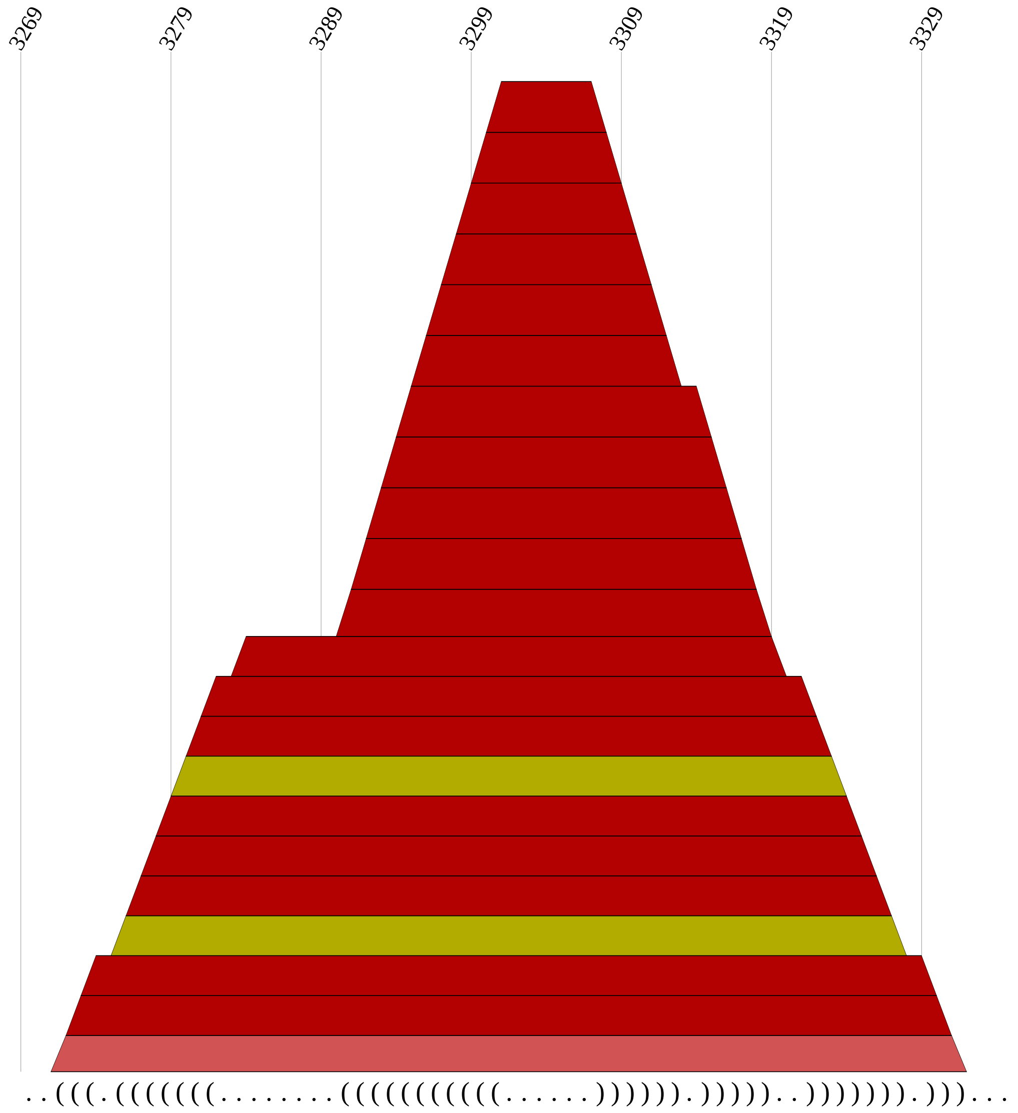


Figure 54: The Hogeweg mountain representation of the ϵ-structure at the end of the HBV pregenome.

6 Discussion

It was the aim of this thesis to use the information contained in the amino acid sequences translated from the coding regions of a (virus) genome as a means of improving the quality of the alignment of the genomic DNA or RNA sequences. The program `RALIGN` is the first implementation of this approach.

`RALIGN` is written in `ANSI C` and hence easily portable to different operating systems. It was developed on PC's running `Linux` and works well with different `Unix` dialects. The standard multiple alignment package `CLUSTAL W` is invoked by means of a system call.

The results reported in the previous sections show that `RALIGN` yields significantly improved aligmnments, compared to the output of the `CLUSTAL W` alignments of the same nucleic acid sequences. As examples for the significant reduction of the number of short gaps we have discussed the alignment of complete HIV-1 RNA genomes. While the length of the entire alignment was almost the same, we found the number of gaps was reduced by 44% up to 70 % in certain parts of the multiple alignment. The `RALIGN` approach guarantees that insertions and deletions within coding regions correspond exactly to insertions and deletions at the protein level. A number of very short gaps is therefore often collected into a single longer one.

As a first application for the combination of amino acid and nucleic acid based alignments, the output of `RALIGN` was used as the basis for the search of conserved RNA secondary structure elements. The programs `alidot` and `pfrali` can be used to scan a sample of RNA sequences for conserved secondary structure elements and to predict their consensus structure. A good multiple alignment and structure predictions for each of the sequences under consideration are required as input for this procedure.

Not surprisingly, the performance of the `pfrali` program depends on the quality of the multiple sequence alignment that is used as an input.

Table 1: Summary of predicted secondary structure elements in the RNA pregenome of Human Hepatitis B Virus.

Position	Figures	Remark
20-100	29, 30, 31	ϵ-element
620-670	33, 34, 35	?
1200-1320	37-42	possible enhancer
1392-1420	43, 44	?
2700-2730	47, 48, 49	?
3215-3245	50, 51	?
3270-3335	52, 53, 54	repeat of ϵ

Although the results are surprisingly robust with respect to minor alignment problems, we expect that alignment inaccuracies will become a substantial problem when dealing with more diverse data sets. Hence, the advantages of `RALIGN` are demonstrated here in the example of viral sequences.

A number of probably significant secondary structure elements were predicted from a sample of 16 sequences pregenomic RNA of different isolates of the Human Hepatitis B virus, see table 1. In particular, the occurrance of consistent and, in particular, compensatory mutations strongly supports a predicted base pair The well-known ϵ-structure was correctly predicted by both `alidot` and `pfrali` from the `RALIGN` alignment of human hepatitis B virus genomes. Furthermore, a number of secondary structure elements have been detected by this method that have not been described in the literature so far.

The region around positions 1200-1320, see figure 37, is of particular interest. The ENH enhancer element is located in this area. Whether the potentially conserved structural elements detected here are related to the functioning of ENH is unknown. It would certainly be very interesting to see if mutations that affect the secondary structure but are neutral at the level of the proteins have an effect on the viability of the virus. Unfortunately, only experimental evidence will eventually decide whether the RNA structure in the region is of importance for the viability of the virus.

7 Directions for Future Improvements

The `RALIGN` procedure could be improved in various ways in the future:

Instead of just looking for open reading frames one could use for instance hidden Markov Models to search for *likely* coding regions. These methods is fairly good at descriminating between open reading frames that do not code for a protein and actual coding sequences, see e.g. [9].

The alignment might be further improved by using alternative hierachies of open reading frames. In particular, it might be advantageous to use the more conserved (instead of the longer) ORFs with higher priority. In general, the current mechanism for detecting homologous reading frames is not very robust. The procedure could probably be improved significantly by explicitly comparing the pairwise sequence homologies of all ORFs.

Even in the case of overlapping reading frames, it might be useful to consider the alignments of each reading frame and to combine them, for instance, using local optimzation rules, instead of selecting a single reading frame.

In some cases, `CLUSTAL W` introduces gaps into the 'tagging sequences' that are used to combine the alignments of the individual ORFs. At present, `RALIGN` returns an error message if this problem occurs. While this a rare problem, a more reliable procedure would be desireable.

We have been able to show that `RALIGN` is capable of substantially improving the quality of alignments. It will be necessary in the future to demonstrate this effect quantitatively. In particular, a detailed statistics of the distribution of types and lengths of gaps produced by different alignment procedures, including `RALIGN`, as a function of the average sequence homology in the sample is still missing.

Currenly, `RALIGN` invokes `CLUSTAL W` using a system call. One might want to use different multiple alignment procedures instead, if only for the sake of comparison. On the other hand, system calls cause a substantial loss in performance, so that it would be desirable to have the multiple alignment procedure available as a run-time library.

The user interface is fairly terse in the present implementation. A graphical user interface, following the example of `CLUSTAL X` would be desirable.

Finally, `RALIGN` at present does not produce auxiliary information such as phylogentic trees. While it is straight forward to implement such procedures, this was beyond the scope of this work reported here.

8 Appendix A

8.1 The manual page

8.1.1 NAME

ralign - combined amino acid and nucleic acid based alignments

8.1.2 SYNOPSIS

```
ralign [-c[CTN]] FN FN FN ...  [-c[CTN]] FN FN FN ...
```

8.1.3 DESCRIPTION

`Ralign` reads `GenBank` files and files in Pearson's format. It finds all theoretically possible open reading frames which have a minimal length. It is possible for the user to define one or more than one codon tables for each sequence or a group of sequences. Every input file can be processed using its own codon table. The standard codon table is the universal genetic code. The found coding regions get translated, using the correct codon table, and the resulting protein sequences are compared to the protein sequences in the `GenBank` file, if available. Two output files are created: a text file (`ORF.ral`) which contains all data about the found open reading frames, either derived by reading the data in the `GenBank` file or as a result of the automatic search. The second file is a `PostScript` output (`ORF.ps`) which gives a graphical representation of the found open reading frames either in one of the three frames or as derived from the `GenBank` file input with all introns. `Ralign` then chooses the coding regions which are maintained through the alignments as protein sequences. `Ralign` checks, going from the longest coding region to the shorter ones, which of these regions are located at the same position in a list sorted by size. In the case of overlapping coding regions the highest priority is given to the longest ORF. The shorter overlapping

ORF then consists only of these parts which are not covered by the longer. This assumption is presented to the user in the file `ORFinput.ral` and can be altered. Then the ORFs get aligned as proteins (using `CLUSTAL W`) while the alignments are sorted by the length of the open reading frames. After the protein alignments the non coding regions get aligned as nucleic acids (using `CLUSTAL W`). Finally the partial alignments are merged and tested for alignment mistakes and the complete sequences are written into a resulting alignment file in `CLUSTAL W` style.

8.1.4 OPTIONS

`-c CTN` where `CTN` is a codon table name. The following codon tables are available:

`univ`: universal genetic code (standard)

`acet`: Acetabularia

`ccyl`: Candida cylindrica

`tepa`: Tetrahymena, Paramecium, Oxytrichia, Stylonychia, Glaucoma

`eupl`: Euplotes

`mlut`: Micrococcus luteus

`mysp`: Mycoplasma, Spiroplasma

`mitocan`: canonical mitochondrial code

`mitovrt`: Vertebrates - mitochondrial code

`mitoart`: Arthropods - mitochondrial code

`mitoech`: Echinoderms - mitochondrial code

`mitomol`:	Molluscs - mitochondrial code
`mitoasc`:	Ascidians - mitochondrial code
`mitonem`:	Nematodes - mitochondrial code
`mitopla`:	Plathelminths - mitochondrial code
`mitoyea`:	Yeasts - mitochondrial code
`mitoeua`:	Euascomycetes - mitochondrial code
`mitopro`:	Protozoans - mitochondrial code

8.1.5 VERSION

This man page documents version 1 of ralign.

8.1.6 AUTHOR

Roman R. Stocsits

8.1.7 BUGS

Comments and bug reports should be sent to roman@tbi.univie.ac.at.

9 Appendix B

9.1 Program Description

```
struct seq_identification *process_command(int argc,
char **argv);
```

This routine takes over the integer value argc which represents the number of arguments, the program was started with, and the two dimensional character array `**argv` which lists all arguments. These arguments are on one hand the paths and the names of the input sequence files and on the other hand the synonyms for the codon tables that should be used for finding the open reading frames in the sequences and for translating them. Every desired codon table follows the argument '`-c`' and is used then by the program for all the input sequence files that follow this codon table name in the command line till the next '`-c`' with an codon table name.

The standard codon table is the universal genetic code. This will be used if no other codon table name is specified and it will work well in most cases.

The sequence input files can contain one ore more sequences, they can be `GenBank` files or files in Pearson's format. If the sequences have distinct names within the files, these names are used. If the sequence input file contains just the sequence without any name, the name and path of the file become the sequence name. This routine also creates an array of structures of the type `seq_identification`, one structure for each input sequence. It also writes the desired codon table and the file name into this structure. If some of the input files are `GenBank` files it calls the routine `readGBF` (see below) for them, otherwise, if Pearson's sequence files should be handled, it calls `readText` (see below) and gives up a pointer at the file to be read to the subroutine. Finally, the information derived through these subroutines, is written into the structure array of the type `seq_identification`.

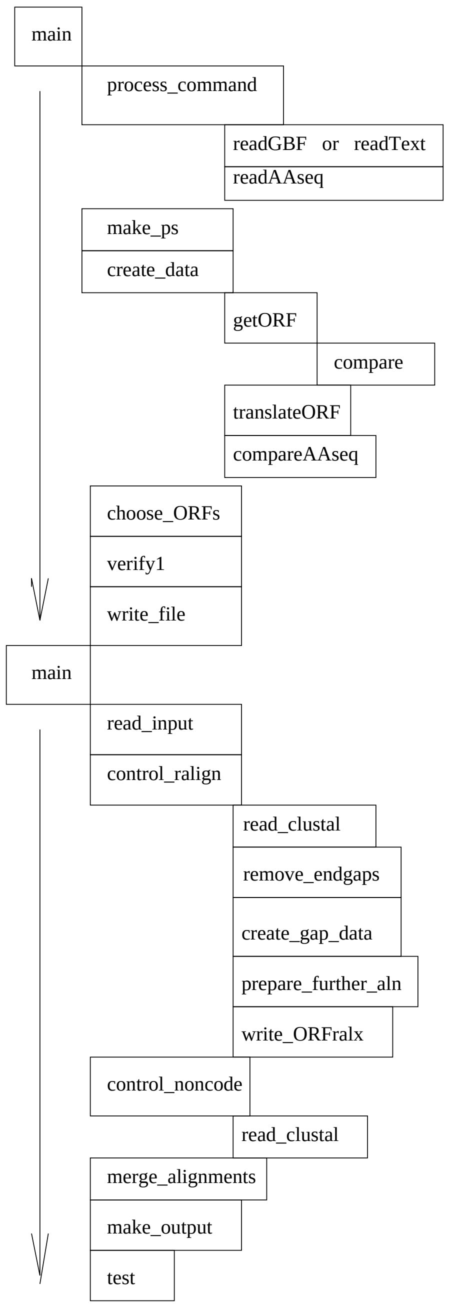


Figure 55: This flow chart gives a graphical representation of the main subroutines of `RALIGN`. See the text for further description.

`char **readGBF(FILE *fp);`

This routine reads `GenBank` files and extracts some necessary information like the name and the length of the sequence.

It also reads the nucleic acid sequence and starts a small subroutine `char *readAAseq(char *line,FILE *fp);` which reads the amino acid sequence in the `GenBank` file, if available. Finally, it gives the name of the sequence, the nucleic acid sequence and the protein sequence back to `process_command` as a double character pointer.

`char ***readText(FILE *fp);`

This routine takes over a pointer at either a file in Pearson's format, which contains one or more sequences with names, or a file which only contains one sequence without any name. It reads the names, if available, and the sequence(s) and gives them back to `process_command` as a triple character pointer which contains names and sequences sorted by the succession of their input.

`void use_me_this_way(void);`

If the program gets started in a not correct way, e.g. with wrong usage of the codon table names, this routine displays some information about the correct commands and the available codon tables and exits the program.

`void make_ps(void);`

This routine opens a file pointer and writes all necessary data for creating a `PostScript` output into this file. This `PostScript` output gives some graphical representation of the found open reading frames and the structure of exons and introns in the nucleic acid sequences.

The data about the ORFs per se is not yet written into this file, but just the surrounding `PostScript` source code and the commands. The data about the coding regions is linked later to this `PostScript` environment.

```
struct ORFdata **create_data(struct seq_identification *SID);
```

This routine takes over the formerly created pointer at the structure of the type `seq_identification` and creates a two dimensional array of structures of the type `ORFdata`.

It calls the subroutine `getORF` (see below) and passes the information in the structure `seq_identification` on to this subroutine. The transmitted information is the nucleic acid sequence, the names of the sequence and the file, the name of the used codon table and a pointer at the `PostScript` file which the data about the open reading frames and the introns and exons are appended to. Every found coding region, whether derived through the `GenBank` file information or as a result of the open reading frame search of `getORF` , is written into a discrete structure of the type `ORFdata`. Also all available information about the ORFs is written into these structures and some variables which are part of the structure `ORFdata` and get used in later parts of the program are initialized and set to zero. The first array of `struct ORFdata` are all coding regions in one nucleic acid sequence (this array is the result of the routine `getORF`), the second array are the nucleic acid sequences (with all open reading frames).

A text file called `ORF.ral` is created also by this routine which contains information about all open reading frames found.

Then the translation routine `translateORF` is called and translates all open reading frames into protein sequences. Also these protein sequences are written into the `ORFdata` structures. Finally, after this, a small subroutine `compareAASeq` is called which checks for differences between the translated protein sequences and the protein sequences derived from `GenBank` files, if available.

`struct ORFdata *getORF(char *FileName, char *seq, char *CTN, FILE *ORFps, char *SeqName);`

This routine is called by `create_data` and it takes over the information stored in the structure `seq_identification`. For every structure variable, of which the data come, an array of the structure `ORFdata` is created, one structure variable for each found coding region. In the case of `GenBank` files the information about exons and introns (if present) is read and the coding regions are linked, the positions of start- and stop-codons, the length of the found coding region and the coding region itself as a sequence are written into the structure `ORFdata`. Beside this all `GenBank` files and, of course, all other files containing sequences are scanned in all three frames for further start- and stop- codons considering the codon table which is used for the sequence looked at. Substrings are forbidden, what means that an open reading frame which has the same stop-codon as a longer coding region, starting prior to this shorter one, is not recognized as an own ORF. And a recognized coding region which is written into its own structure `ORFdata` must also have a definitive minimum length.

Then the array of `ORFdata` structures is sorted by length of the found open reading frames using the qsort function which needs a compare function `static int compare(const void *a, const void *b);`.

Finally, all data about the exons and introns, about the `GenBank` derived and search derived open reading frames, are appended to the `PostScript` file which shows these data then as a graphical output.

`char *translateORF(char *orf, char *CodonTableName);`

This routine takes over a character pointer which represents the nucleic acid sequence of an open reading frame and a certain codon table name. It then translates the sequence into protein using the desired codon table and gives back the protein sequence.

```
struct temp_no *choose_ORFs(struct ORFdata **allORF, struct
seq_identification *SID);
```

This routine takes over the double array of structures of the type `ORFdata`, which represents all found coding regions, and the array of structures of the type `seq_identification`, which contains the input sequences' data. It makes a first choice of coding regions which should get aligned as a protein alignment, in principle, it chooses all open reading frames that have a earlier defined minimum length and constructs an array of structures called `temp_no` which consist in an integer value for the number of the input nucleic acid sequence and an array of integer values for the numbers of the certain open reading frames choosen.

```
struct temp_no *verify1(struct temp_no *inputno);
```

This routine gets the formerly constructed array of `temp_no` structures and modifies it. All open reading frames that were choosen by `choose_ORFs` in all input nucleic acid sequences, and that are conserved enough to stand on the same positions in the lists of coding regions sorted by length, become verified as choosen. And the modified structure `temp_no` array is given back.

```
int write_file(struct temp_no *inputno1);
```

This routine takes over the array of structures of the type `temp_no` and writes a file called `ORFinput.ral`. This file contains the information which open reading frames were choosen automatically for protein alignment. Also a message is brought if the automatic choice of coding regions fails, because the sequences' genetic structures seem to be too divergent for combined alignments. After this point of the program the user gets a chance for intervening: it is possible to change the automatically choosen ORFs, to edit this file and to tell the program which ORFs should be aligned on the level of protein. If one wants to edit `ORFinput.ral`, an `emacs` window is opened by the program, the changes can be made and saved. Before starting the protein alignments the program reads this file and respects changes which overrule the automatic choice.

Finally, an integer value is returned which gives the number of expected protein alignments to be carried out.

Annotation: it is possible that some open reading frames are choosen for protein alignment which lie completely within another, larger coding region in one of the two other frames. This is no 'bug', the reason for this is that after the protein alignments the end gap containing regions of the alignments are cut away, so that just the central 'block' of each alignment is the result. And it is possible that such a shortened open reading frame sequence is not still long enough that the second, shorter coding region still lies completely within it, but the ends of the second sequence may project beyond the ends of the now shorter first coding region. The cut away end gap containing parts of the larger ORF could be covered now by the second ORF that lies within, and so they could also get aligned on the level of protein as part of the second ORF.

```
struct aln_results read_clustal(FILE *aln);
```

This routine takes over a pointer at a alignment results file of `CLUSTAL W`. It reads the names of the aligned sequences and the sequences themselves as they got modified by the alignment and contain gaps now at certain positions. The names and the sequences belonging to them are written into a structure of the type `aln_results` as parts of an array of character pointers. This structure is given back.

```
struct aln_results remove_endgaps(struct aln_results
ORF1aln_results);
```

The input into this routine is the formerly constructed structure of the type `aln_results` containing the results of the protein alignment. The first important function of this routine is to remove the tags which were appended to the protein sequences prior to the alignment in the case that these open reading frames had overlapping regions with other ORFs. Always, if the program has to handle overlapping coding regions, a certain hierarchy is established which gives the longer coding region higher priority. In an area of overlapping ORFs the ORF with highest priority gets aligned without restrictions. Then the end gap containing parts of the alignment are cut away and become part of the alignment of the ORFs with lower priority. As a result of this the sequence ends, which were part of the prior alignment and were cut away and appended to these sequences of second priority, have to lie one above each other in the alignment. This is achieved by appending the same nonsense tag to each sequence at this certain end. But after the alignment these tags have to be removed again, of course.

The second function of this routine is to remove these regions at the two ends of the alignment which contain end gaps. The end parts are cut away as long as the longest end gap, so that, as a result, the aligned sequences begin in each case with a letter (an amino acid). The cut away sequences and also the lengths of these sequences are written into a structure of the type `aln_results`, together with the aligned sequences and their names.

```
struct ORFdata **create_gap_data(struct aln_results ORF_noend,
struct ORFdata **allORF, int no_aln, int *orfnos1);
```

This routine transfers the information of the structure `aln_results` to the double array of the structure `ORFdata` which represents the open reading frames. The cut away sequences, their lengths and also the resulting aligned central sequences are transferred, the definitive starting- and end points of the open reading frames in the input nucleic acid sequences are written into this double array structure. And also the rank of the alignment regarding the level of priority in the case of overlapping coding regions and a marker that this ORF was used for protein alignment, are recorded.

```
struct ORFdata **prepare_further_aln(struct ORFdata **allORF,
int *orfnos1);
```

This routine is of major importance if the program has to handle overlapping open reading frames. Again the double array of structures of the type `ORFdata` is taken over, representing the found coding regions, and also an integer array that holds the information which ORFs are used for an alignment on the level of protein. The routine manages the organization between the different open reading frames, e.g. the beginning and end of a coding region after cutting away the end gap containing regions, or if a second ORF with lower priority is still overlapping after this modifications. In the case of overlapping ORFs it is not possible that the end of the first coding region exactly fits the beginning of the second in an other frame, even if their ends are adapted each other. The reason for this is that a coding region in an other frame is shifted one or two nucleotides against the first coding region. This fact is also respected and managed by this routine. Before starting the protein alignments the starting- and end-positions of the coding regions are determined and prepared for the alignment. This preparation includes appending the tags to the ORFs in order to lay the ends of the sequences exactly one above the other during the alignment in the case of overlapping open reading frames. The tags consist in the string '`THISISATAG`' which is appended twelve fold at the ends that lie immediately in the neighbourhood of a coding region with higher priority. It is also possible that a coding region is bordered by two open reading frames with higher priority at both ends, so that the tags are appended at both sides of this ORF. The sequences which were prepared for the alignments this way are written into the `ORFdata` structures, each sequence into the ORF's structure it belongs to. And the double array of structures is given back.

```
int write_ORFralx(struct ORFdata **allORF, int *orfnos2);
```

This routine takes over the structures of the type `ORFdata` which were prepared for the protein alignments, and the integer array that contains the numbers of the open reading frames which get aligned as proteins. It writes an input file for `CLUSTAL W` and gives back a marker which shows that a file has been written.

```
int **read_imput(int alnno, struct seq_identification *SID);
```

The routine reads the file `ORFinput.ral` which could be modified by the user and produces the integer arrays which hold the information about the open reading frames that were choosen by the program or the user for alignment on the level of proteins.

```
struct ORFdata **control_ralign(struct ORFdata **allORF,
int **orfnos);
```

This routine is of central importance for the control of the alignments of the open reading frames as proteins. It manages the alignments of the different priorities and starts `CLUSTAL W` by a system call. It also calls and manages the formerly explained subroutines `read_clustal`, `remove_endgaps`, `create_gap_data`, `prepare_further_aln` and also `write_ORFralx`. It gives back the double array of the structure `ORFdata` which now contains all changes made by the called subroutines.

```
struct aln_results *control_noncode(struct ORFdata **allORF,
struct seq_identification *SID, int alnno);
```

This is the routine that controls and manages the alignments on the level of nucleic acids of the non coding regions between the ORFs. First, it creates the `CLUSTAL W` input files containing the sequences with the tags for lying the ends of the sequences one above the other in the alignments. Second, it calls `CLUSTAL W` and the `read_clustal` subroutine after the alignment. It creates an array of the structure `aln_results` which contains the results of each nucleic acid alignment of the non coding regions and it manages the order of the alignments. It also cuts away the tags after the alignments, attached on one or both sides of the sequences. Finally, it returns the array of the result containing structure `aln_results`.

```
struct aln_results merge_alignments(struct aln_results *final,
struct ORFdata **allORF, struct seq_identification *SID);
```

This routine takes over the double array of the structure `ORFdata` which contains the results of the protein alignments, and the array of the structure `aln_results` which contains the results of the non coding regions' alignments, and it also takes over the array of the structure `seq_identification` which is used to check the order of the manipulated sequences. By this routine the resulting parts of the single alignments are merged again to the complete sequences, now containg all gaps. It also calls the subroutine `rev_translate` which writes the protein sequences as a result of the protein alignments back into nucleic acid sequences. The complete sequences are finally written back into a structure of the type `aln_results` which contains, of course, also all names.

```
char *rev_translate(struct ORFdata ORF);
```

A reverse translation is done by this routine. It takes over one structure of the type `ORFdata`, which represents one open reading frame that was aligned as a protein sequence. It uses the nucleic acid sequence of this coding region, which is also stored in the structure, to construct the reverse translated sequence with all gaps. Every gap in the protein sequence results in a gap of three fold length on the level of nucleic acids. The nucleic acid sequence is given back.

```
int make_output(struct aln_results last);
```

This routine uses the structure of the type `aln_results` produced by `merge_alignments` to construct an output file in the style of `CLUSTAL W` output files. The head of the file is modified: it contains a short comment which tells the user that this file was created by `RALIGN`.

```
int test(struct seq_identification *SID,
struct aln_results last);
```

This routine finally checks if the merging of the partial alignments was successful or if some pieces of the tags were inserted into the sequences by an alignment error (when `CLUSTAL W` inserts a gap into the tags, this occurs). It is also checked whether the sequences after all manipulations are the same as the input sequences or some undefined mistake changed them.

9.2 The structures

```
struct ORFdata{
  char *foundExon, *aaseq, *seqname, *beginAAseq, *endAAseq;
  char *centerAAseq, *prep_aln_aaseq;
  char exon_info[20];
  int start, stop, ORFlen, ORFcount, AAlen, SEQcount;
  int blockstart, blockstop, endAA, beginAA, rank_of_aln;
  int used_for_aln, prep_start, prep_stop;
  int start_overlap_minus1or2, stop_overlap_minus1or2;
  int found_higher_rank, higherrankORFarr;
};
```

This structure contains the nucleic acid sequence of the found open reading frame, the amino acid sequence, the name, the cut away end gap containing sequences, the central alignment sequence, the sequence prior to the alignment, information if the found exons are search derived or `GenBank` file derived, the starting point, the end, the length, the number of the ORF, the length of the protein sequence and the number of the input sequence. Further some information which is used by the program to manage the different alignments on the level of proteins and nucleic acids.

```
struct seq_identification {
  char *sequ, *FileName, *CodonTableName, *read_aaseq;
  char *seqname;
};
```

This structure contains the input nucleic acid sequence, the name of the input file, the desired codon table, the amino acid sequence as derived from the `GenBank` file and the name of the sequence.

```
struct temp_no {
  int SEQno, *ORFno;
};
```

This structure contains the information which ORFs get aligned as proteins.

```
struct aln_results {
  char **alnseqs, **names;
  int gaplen1, gaplen2, *beginAA, *endAA;
};
```

This structure represents the results of alignments. The aligned sequences, their names, the lengths of the cut away end gap containing parts and information about the number of amino acids cut away in a certain case.

References

[1] J. P. Abrahams, M. van den Berg, E. van Batenburg, and C. Pleij. Prediction of RNA secondary structure, including pseudoknotting, by computer simulation. *Nucl. Acids Res.*, 18:3035–3044, 1990.

[2] V. V. Anshelevich, A. V. Vologodskii, A. V. Lukashin, and M. D. Frank-Kamenetskii. Slow relaxational processes in the melting of linear biopolymers: A theory and its application to nucleic acids. *Biopolymers*, 23:39–58, 1984.

[3] G. Awang and D. Sen. Mode of dimerization of HIV-1 genomic RNA. *Biochemistry*, 32:11453–11457, 1993.

[4] H. J. Bandelt and A. W.M. Dress. A canonical decomposition theory for metrics on a finite set. *Advances in mathematics*, 92(1):47, 1992.

[5] A. R. Banerjee, J. A. Jaeger, and D. H. Turner. Thermal unfolding of a group I ribozyme: The low-temperature transition is primarily disruption of tertiary structure. *Biochemistry*, 32:153–163, 1993.

[6] G. J. Barton and M. J. E. Sternberg. A strategy for the rapid multiple alignment of protein sequences. confidence levels from tertiary structure comparisons. *J. Mol. Biol.*, 198:327–337, 1987.

[7] D. Bashford, C. Chothia, and A. M. Lesk. Determinants of a protein fold. unique features of the globin amino acid sequences. *J. Mol. Biol.*, 196:199–216, 1987.

[8] R. Bellman. On the theory of dynamic programming. *Proc. Natl. Acad. Sci. USA*, 38:716–719, 1952.

[9] M. Borodovsky and J. McIninch. Genmark: parallel gene recognition for both dna strands. *Comp. & Chem.*, 17:123–133, 1993.

[10] J. Boyle, G. T. Robillard, and S. H. Kim. Sequential folding of transfer RNA. A nuclear magnetic resonance study of successively longer tRNA fragments with a common 5' end. *J. Mol. Biol.*, 139:601–625, 1980.

[11] N. Breton, C. Jacob, and P. Daegelen. Prediction of sequentially optimal RNA secondary structures. *J. Biomol. Struct. Dyn.*, 14:772–740, 1997.

[12] J. H. Cate, A. R. Gooding, E. Podell, K. Zhou, B. L. Golden, A. A. Szewczak, C. D. Kundrot, T. R. Cech, and J. H. Doudna. RNA tertiary structure mediation by adenosine platforms. *Science*, 273:1696–1699, 1996.

[13] J. H. Cate, A. R. Gooding, E. Podell, K. Zhou, B. L. Golden, A. A. Szewczak, C. D. Kundrot, T. R. Cech, and J. H. Doudna. Crystal structure of a group I ribozyme domain: Principles of RNA packing. *Science*, 273:1678–1685, 1996.

[14] T. R. Cech and B. L. Bass. Biological catalysis by RNA. *Annu. Rev. Biochem.*, 55:599–630, 1986.

[15] M. Chastain and I. Tinoco. Nucleoside triples from the group I intron. *Biochemistry*, 32:14220–14228, 1993.

[16] D. M. Crothers, P.E. Cole, C. W. Hilbers, and R. G. Shulman. The molecular mechanism of thermal unfolding of *escherichia coli* formylmethionine transfer RNA. *J. Mol. Biol.*, 87:63–88, 1974.

[17] Jan Cupal, Christoph Flamm, Alexander Renner, and Peter F. Stadler. Density of states, metastable states, and saddle points. Exploring the energy landscape of an RNA molecule. In T. Gaasterland, P. Karp, K. Karplus, Ch. Ouzounis, Ch. Sander, and A. Valencia, editors, *Proceedings of the ISMB-97*, pages 88–91, Menlo Park, CA, 1997. AAAI Press.

[18] Jan Cupal, Ivo L. Hofacker, and Peter F. Stadler. Dynamic programming algorithm for the density of states of RNA secondary structures. In R. Hofstädt, T. Lengauer, M. Löffler, and D. Schomburg, editors, *Computer Science and Biology 96 (Proceedings of the German Conference on Bioinformatics)*, pages 184–186, Leipzig (Germany), 1996. Univeristät Leipzig.

[19] A. E. Dahlberg. The functional role of ribosomal RNA in protein synthesis. *Cell*, 57:525–529, 1989.

[20] M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. *Atlas of Protein Sequence and Structure*, volume 5 of *3*, chapter 3, pages 345–352. NBRF Washington, 1978.

[21] D. E. Draper. Strategies for rna folding. *Trends Biochem. Sci.*, 21:145–149, 1996.

[22] D. F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, 25:351–360, 1987.

[23] W. M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science*, 155:279–284, 1967.

[24] W. Fontana, D. A. M. Konings, P. F. Stadler, and P. Schuster. Statistics of RNA secondary structures. *Biopolymers*, 33:1389–1404, 1993.

[25] W. Fontana, P. F. Stadler, E. G. Bornberg-Bauer, T. Griesmacher, I. L. Hofacker, M. Tacker, P. Taranzona, E. D. Weinberger, and P. Schuster. RNA folding and combinatory landscapes. *Phys. Rev. E*, 47:2083–2099, 1993.

[26] S. M. Freier, R. Kierzek, J. A. Jaeger, N. Sugimoto, M. H. Caruthers, T. Neilson, and T. H. Turner. Improved free energy parameters for prediction of RNA duplex stability. *ProcṄatl. Acad. Sci. USA*, 83:9373–9377, 1986.

[27] T. C. Gluick and D. E. Draper. Thermodynamics of a pseudoknotted mRNA fragment. *J. Mol. Biol.*, 241:246–262, 1994.

[28] W. B. Goad and M. I. Kanehisa. Pattern recognition in nucleic acid sequences. A general method for finding local homologies and symmetries. *Nucl. Acids Res.*, 10:247–263, 1982.

[29] A. P. Gultyaev. The computer simulation of rna folding involving pseudoknot formation. *Nucl. Acids Res.*, 19:2489–2494, 1991.

[30] A. P. Gultyaev, F. H. D. van Batenburg, and C. W. A. Pleij. The computer simulation of RNA folding pathways using a genetic algorithm. *J. Mol. Biol.*, 250:37–51, 1995.

[31] A. P. Gultyaev, F. H. D. van Batenburg, and C. W. A. Pleij. Dynamic competition between alternative structures in viroid RNAs simulated by an RNA folding algorithm. *J. Mol. Biol.*, 276:43–55, 1998.

[32] R. W. Hamming. *Coding and Information Theory*, pages 44–47. Prentice-Hall, Englewood Cliffs, 2 edition, 1989.

[33] T. P. Hausner, J. Atmadja, and K. H. Nierhaus. Evidence that the G2661 region of 23S rRNA is located at the ribosomal binding site of both elongation factors. *Biochemie*, 69:911–923, 1987.

[34] L. He, R. Kierzek, J. SantaLucia, A. E. Walter, and D. H. Turner. Nearest-neighbour parameters for G-U mismatches. *Biochemistry*, 30:11124–11132, 1991.

[35] R. Hecker, Z. Wang, G. Riesner, and D. Steger. Analysis of RNA structures by temperature-gradient gel electrophoresis: Viroid replication and processing. *Gene*, 72:59–74, 1988.

[36] S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919, 1992.

[37] D. Herschlag. RNA chaperones and the RNA folding problem. *J. Biol. Chem.*, 270:20871–20874, 1995.

[38] P. G. Higgs and S. R. Morgan. Thermodynamics of RNA folding. when is an RNA molecule in equilibrium. In F. Morán, A. Moreno, J. J. Merelo, and Chacón, editors, *Advances in Artificial Life*, pages 852–861, Berlin, 1995. ECAL 95, Springer Verlag.

[39] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Comm. Assoc. Comp. Mach.*, 18:341–343, 1975.

[40] I. L. Hofacker, M. Fekete, C. Flamm, M. A. Huynen, S. Rauscher, P. E. Stolorz, and P. F. Stadler. Automatic detection of conserved RNA structure elements in complete RNA virus genomes. *Nucl. Acids Res*, 26:3825–3863, 1998.

[41] I. L. Hofacker, W. Fontana, P. F. Stadler, and P. Schuster. `Vienna RNA Package`. `http://www.tbi.univie.ac.at/~ivo/RNA/`, 1994. (Free Software).

[42] I. L. Hofacker and P. F. Stadler. Automatic detection of conserved base pairing patterns in RNA virus genomes. *Comp. & Chem.*, 23:401–414, 1999.

[43] Ivo L. Hofacker, Walter Fontana, Peter F. Stadler, Sebastian Bonhoeffer, Manfred Tacker, and Peter Schuster. Fast folding and comparison of RNA secondary structures. *Monatsh. Chemie*, 125:167–188, 1994.

[44] Ivo L. Hofacker, Martijn A. Huynen, Peter F. Stadler, and Paul E. Stolorz. Knowledge discovery in RNA sequence families of HIV using scalable computers. In Evangelos Simoudis, Jiawei Han, and Usama Fayyad, editors, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR*, pages 20–25, Menlo Park, CA, 1996. AAAI Press.

[45] P. Hogeweg and B. Hesper. Energy directed folding of RNA sequences. *Nucl. Acids Res.*, 12:67–74, 1984.

[46] J. W. Hunt and M. D. McIlroy. An algorithm for differential file comparison. Technical Report Comp. Sci. 41, Bell Laboratories, 1976.

[47] Daniel H. Huson. SplitsTree: analyzing and visualizing evolutionary data. *Bioinformatics*, 14:68–73, 1998.

[48] Martijn A. Huynen, Alan S. Perelson, Wayne A. Vieira, and Peter F. Stadler. Base pairing probabilities in a complete HIV-1 RNA. *J. Comp. Biol.*, 3:253–274, 1996.

[49] J. A. Jaeger, D. H. Turner, and M. Zuker. Improved predictions of secondary structures for RNA. *Proc. Natl. Acad. Sci. USA*, 86:7706–7710, 1989.

[50] B. R. Jordan. Computer generation of pairing schemes for RNA molecules. *J. Theor. Biol.*, 34:363–378, 1972.

[51] G. F. Joyce. In vitro evolution of nucleic acids. *Curr. Opin. Struct. Biol.*, 4:331–336, 1994.

[52] M.I. Kanehisa and W. B. Goad. Pattern recognition in nucleic acid sequences. An efficient method for finding locally stable secondary structures. *Nucl. Acids Res.*, 10:265–277, 1982.

[53] A. H. Kidd and K. Kidd-Ljunggren. A revised secondary structure model for the 3'-end of hepatitis b virus pregenomic rna. *Nucl. Acids Res.*, 24:3295–3301, 1996.

[54] M. Kunze and G. Thierrin. Maximal common subsequences of pairs of strings. *Congr. Num.*, 34:299–311, 1982.

[55] 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.

[56] P. Loss, M. Schmitz, G. Steger, and D. Riesner. Formation of a thermodynamically metastable structure containing hairpin II is critical for infectivity of potato spindle tuber viroid RNA. *EMBO J.*, 10:719–727, 1991.

[57] M. Lu and D. E. Draper. Bases defining an ammonium and magnesium ion-dependent tertiary structure within the large subunit ribosomal RNA. *J. Mol. Biol.*, 244:572–585, 1994.

[58] C. W. Mandl, H. Holzmann, T. Meixner, S. Rauscher, P. F. Stadler, S. L. Allison, , and F. X. Heinz. Spontaneous and engineered deletions in the 3' noncoding region of tick-borne encephalitis virus: construction of highly attenuated mutants of a flavivirus. *J. Virology*, 72:2132–2140, 1998.

[59] H. M. Martinez. An RNA folding rule. *Nucl. Acids Res.*, 12:323–324, 1984.

[60] J. S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29:1105–1119, 1990.

[61] D. L. Mills, editor. *A new algorithm to determine the Levenshtein distance between two strings*, Conference on Sequence Comparison. University of Montreal, 1978.

[62] E. W. Myers and W. Miller. Optimal alignments in linear space. *CABIOS*, 4:11–17, 1988.

[63] S. B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequences of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[64] J. M. Norman. *Elementary dynamic programming.* Crane, Russak and Co., New York, 1975.

[65] R. Nussinov and A. B. Jacobson. Fast algorithm for predicting the secondary structure of single-stranded RNA. *Proc. Natl. Acad. Sci. USA*, 77:6309–6313, 1980.

[66] R. Nussinov, I. Tinoco, and A. Jacobsen. Secondary structure model for the complete simian virus 50 late precursor RNA. *Nucl. Acids Res.*, 10:351–363, 1982.

[67] R. Nussinov, I. Tinoco, and A. B. Jacobson. Small changes in free energy assignments for unpaired bases do not affect predicted secondary structures in single stranded RNA. *Nucl. Acids Res.*, 10:341–349, 1982.

[68] S. Pascarella and P. Argos. Analysis of insertions/deletions in protein structures. *J. Mol. Biol.*, 224:461–471, 1992.

[69] S. Rauscher, C. Flamm, C. Mandl, F. X. Heinz, and P. F. Stadler. Secondary structure of the 3' noncoding regions of flavivirus genomes: Comparative analysis of base pairing probabilities. *RNA*, 3:779–791, 1997.

[70] A. Rieger and M. Nassal. Distinct requirements for primary sequence in the 5'-and 3'-part of a bulge in the hepatitis b virus rna encapsidation signal revealed by a combined in vivo selection/in vitro amplification system. *Nucl. Acids Res.*, 23:3909–3915, 1995.

[71] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.

[72] D. Sankoff, R. J. Cedergren, and G. Lapalme. Frequency of insertion-deletion, transversion and transition in the evolution of 5S ribosomal RNA. *J. Mol. Evol.*, 7,:133–149, 1976.

[73] D. Sankoff, C. Morel, and R. J. Cedergren. Evolution of 5S RNA and the nonrandomness of base replacement. *Nature New Biology*, 245:232–234, 1973.

[74] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignments through sequence weighting, position specific gap penalties and weight matrix choice. *Nucl. Acids Res.*, 22:4673–4680, 1994.

[75] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Improved sensitivity of profile searches through the use of sequence weights and gap excision. *CABIOS*, 10:19–29, 1994.

[76] D. H. Turner, N. Sugimoto, and S. Freier. RNA structure prediction. *Ann. Rev. Biophys. Chem.*, 17:167–192, 1988.

[77] M. Vingron and P. R. Sibbald. Weighting in sequence space: A comparison of methods in terms of generalized sequences. *Proc. Natl. Acad. Sci. USA*, 90:8777–8781, 1993.

[78] M. Vingron and M. S. Waterman. Sequence alignment and penalty choice. Review of concepts, case studies and implications. *J. Mol. Biol.*, 235:1–12, 1994.

[79] A. E. Walter, D. H. Turner, J. Kim, M. H. Lyttle, P. Mueller, D. H. Mathews, and M. Zuker. Co-axial stacking of helices enhances binding of oligoribonucleotides and improves predictions of RNA folding. *Proc. Natl. Acad. Sci. USA*, 91:9218–9222, 1994.

[80] M. S. Waterman. *Studies on foundation and combinatorics, advances in mathematics supplementary studies*, volume 1, chapter secondary structures of single stranded nucleic acids, pages 167–212. Academic Press N. Y., 1978.

[81] M. S. Waterman and T. Byers. A dynamic programming algorithm to find all solutions in the neighborhood of the optimum. *Math. Biosci.*, 77:179–188, 1985.

[82] W. J. Wilbur and D. J. Lipman. Rapid similarity searches of nucleic acid and protein data banks. *Proc. Natl. Acad. Sci. USA*, 80:726–730, 1983.

[83] M. Zuker and P. Stiegler. Optimal computer folding of larger rna sequences using thermodynamics and auxiliary information. *Nucl. Acids Res.*, 9:133–148, 1981.

Curriculum Vitae

Name:	Roman Rudolf Stocsits
Date of Birth:	1971-02-04
Place of Birth:	Vienna
1977 - 1981:	Elementary School, Volkschule Knöllgasse
1981 - 1985:	Secondary School, BG V Rainergasse
1985 - 1989:	Classical Secondary School, Humanistisches Gymnasium V
1989:	School Leaving Certificate, passing with distinction
1989:	Entering University of Vienna, study of biology and molecular biology
1994, Nov. 15th:	Achieving B.Sc.
1998:	Start of diploma work at the Institute for Theoretical Chemistry, working with Ao. Prof. Dr. Peter F. Stadler on the field of multiple sequence alignments
Current Address:	Angeligasse 59/14 A-1100 Vienna Tel.: +43 1 600 74 02 e-mail: roman@tbi.univie.ac.at