AnchoRNA: Full virus genome alignments through conserved anchor regions

Tom Eulenfeld [1,2,3,*], Sandra Triebel [1,2], Peter F. Stadler [4,5,6,7,8,9,10], Manja Marz [1,2,3,6,11,*]

[1] RNA Bioinformatics and High-Throughput Analysis, Friedrich Schiller University Jena, Germany
[2] European Virus Bioinformatics Center, Jena, Germany
[3] Cluster of Excellence Balance of the Microverse, Friedrich Schiller University Jena, Germany
[4] Bioinformatics Group, Department of Computer Science & Interdisciplinary Center for Bioinformatics, Leipzig University, Germany
[5] Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany
[6] German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, Germany
[7] Department of Theoretical Chemistry, University of Vienna,Austria
[8] Facultad de Ciencias, Universidad National de Colombia, Bogotá, Colombia
[9] Center for Non-coding RNA in Technology and Health, University of Copenhagen, Denmark
[10] Santa Fe Institute, USA
[11] Fritz Lipmann Institute-Leibniz Institute on Aging, Jena, Germany
[*] Contact: tom.eulenfeld@uni-jena.de, manja@uni-jena.de

Multiple sequence alignment of full viral genomes can be challenging due to factors such as long sequences, large insertions/deletions (spanning several 100 nucleotides), large number of sequences, sequence divergence, and high computational complexity in particular when computing alignments based on RNA secondary structures. Standard alignment methods often face these issues, in particular when processing highly variable sequences or when specific phylogenetic analysis is required on selected subsequences.

We present an algorithm to determine high quality anchors that define partitions of sequences and guide the alignment of viral genomes to respect well conserved, and therefore functionally significant, regions. This new approach is implemented in the Python-based command line tool `AnchoRNA`, which is designed to identify conserved regions, or anchors, within coding sequences. By default, anchors are searched in translated coding sequences accounting for high mutation rates in viral genomes. `AnchoRNA` enhances the accuracy and efficiency of full-genome alignment by focusing on these crucial conserved regions. `AnchoRNA` guided alignments are systematically compared to the results of 3 alignment programs. Utilizing a dataset of 55 representative *Pestivirus* genomes, `AnchoRNA` identified 55 anchors that are used for guiding the alignment process. The incorporation of these anchors led to improvements across tested alignment tools, highlighting the effectiveness of `AnchoRNA` in enhancing alignment quality, especially in viral genomes.

Keywords: *Multiple sequence alignment, Conserved regions, Viral genomes, Anchors*

1 Introduction

Multiple sequence alignment (MSA) is a crucial step in biological sequence analysis, serving as the basis for many downstream applications, including functional annotation, evolutionary studies, and structural predictions (e.g. Lamkiewicz et al., 2023). Despite its importance, MSA remains a challenging problem due to its inherent complexity and the diversity of biological sequences, particularly in the context of viral genomes (e.g. Hufsky et al., 2023, 2020; Madhugiri et al., 2014; Hölzer and Marz, 2020). The traditional approach, which focuses on local alignments, has evolved significantly, leading to advancements in alignment accuracy with tools like `ClustalW` (Thompson et al., 1994), `T-Coffee` (Notredame et al., 2000), `MUSCLE` (Edgar, 2004), `MAFFT` (Katoh, 2002), and `DIALIGN` (Morgenstern et al., 1998). These methods have improved the alignment process by integrating local alignment information into global alignment frameworks and incorporating homology data from databases. However, aligning highly variable sequences, such as full viral genomes, still poses significant challenges that require more specialized approaches. Despite these advancements, current MSA tools face limitations when handling certain complexities. Specifically, they often struggle with long sequences, large insertions or deletions (as known for e.g. pestiviruses), and highly divergent sequences, which can lead to suboptimal full virus genome alignments or excessive computational demands, e.g., when simultaneously annotating conserved RNA secondary structures with `LocARNA` (Will et al., 2012).

Anchors help by focusing on conserved regions and by improving alignment accuracy. The precise meaning of anchors is not unequivocally defined in the literature. `DIALIGN` is designed to accommodate constraints on pairwise alignments (Morgenstern et al., 2006). The concept of anchors used here originated in synteny analysis and refers to orthologous sequence elements that appear in all input sequences and therefore subdivide multiple sequence alignments into orthologous blocks delimited by the anchors. Velandia-Huerto et al. (2016) and Berkemer et al. (2017) used such anchors to systematically investigate the turnover of repetitive elements and multi-copy genes such as tRNAs. Anchors in these sense are highly conserved local alignment blocks. In genome-wide synteny analysis, 1-1 orthologous protein coding genes are commonly used as anchors. An annotation-free method to detect them on large scales in closely related genomes was introduced by Käther et al. (2025). By partitioning the global alignment problem into independent regions between consecutive anchors, anchors can drastically reduce the computational demands for long sequences. The smaller search space of the sub-problems not only saves computational resources, but also increases the accuracy. Anchors therefore improve alignments in the presence of large insertions and deletions, or high sequence divergence.

Tools with anchors have been developed for (i) very large genomes: `MUMmer` (Delcher et al., 1999), `Mugsy` (Angiuoli and Salzberg, 2010); (ii) handling duplications and genome rearrangements: `Mauve` (Darling et al., 2004), `progressiveMauve` (Darling et al., 2010), `TBA` (Blanchette et al., 2004); (iii) comparative genomics especially in the context of phylogenetics: `LAGAN`, `Multi-LAGAN` (Brudno et al., 2003b), `AVID` (Bray et al., 2002); (iv) precise alignments based on anchor points: `CHAOS/DIALIGN` (Brudno et al., 2003a), `VISTA genome pipeline` (Dubchak et al., 2009); (v) advanced algorithmic techniques, such as graph-based methods or spaced seeds: `ACANA` (Huang et al., 2005), `FSWM` (Leimeister et al., 2018), `OWEN` (Ogurtsov et al., 2002); (vi) extended general multiple sequence alignment: `ClustalW` (Thompson et al., 1994), `T-Coffee` (Notredame et al., 2000), `MAFFT` (Katoh, 2002); and (vii) larger insertions and deletions:

`DIALIGN` (Morgenstern et al., 1998; Morgenstern, 1999). Viral genomes are small, compact, and generally less structurally complex compared to the genomes of larger organisms. Tools designed for large genomes which emphasize complex rearrangements and extensive duplications are thus not well-suited for viral genomes. Most alignments of virus sequences are computed with `MAFFT`, which is designed for large numbers of sequences, but is not sufficiently accurate for full-genome alignments, in particular in applications where large insertions/deletions occurred or RNA secondary structures are of importance of the study (e.g. Madhugiri et al., 2014). For RNA virus genomes, with a length of up to 40000 nt, the focus is on the development of tools that can efficiently handle high mutation rates and accommodate large numbers (typically 50 to 2000) of genomes in a multiple sequence alignment.

Here, we introduce a method to compute accurate anchors that is designed specifically for application to viral genomes. `AnchoRNA` produces anchors that are gap-less local alignments. This subdivides the alignment problem, leading to a better performance both in terms of resource consumption and accuracy of the alignment. By focusing on these key conserved regions, `AnchoRNA` addresses the challenges of long viral genomes, large insertions/deletions, and highly divergent sequences, offering a more accurate and efficient solution for aligning both nucleotide and protein sequences.

2 Algorithm and Implementation

The `AnchoRNA` algorithm, summarized in Fig. 1, assumes that there are no genome rearrangements and proceeds in three stages: first, a window-based similarity search is used on the translated coding regions to find short matches as initial candidates. These are then processed by merging overlapping candidates to larger more significant gap-less alignments. In the final phase, inconsistencies between candidates are resolved by removing less significant candidates, resulting in a co-linear sequence of anchors. Denote the input sequences by $S^{(1)}$, $S^{(2)}$, $\dots S^{(n)}$. An *anchor* consists of a set of n intervals with the same length, called *flukes*, one in each sequence, see Fig. 1. Each anchor corresponds to a gap-less alignment of the subsequences of $S^{(1)}$, $S^{(2)}$, $\dots S^{(n)}$ defined by the flukes. `AnchoRNA` uses one of the input sequences, say $S^{(1)}$, as a reference. We will refer to $S^{(1)}$ as the *guiding sequence*. A sliding window with user-defined fixed length w on the guiding sequence specifies a candidate fluke $F^{(1)} = [j_1, j_1']$. For each sequence $S^{(k)}$, $2 \leq k \leq n$, `AnchoRNA` searches for a matching fluke in a search window $[p_k, q_k]$ where $p_k := j_1 - L + \min(0, n_k - n_1)$ and $q_k := j_1' + L + \max(0, n_k - n_1)$ and $n_i := |S^{(i)}|$ denotes the length of sequence $S^{(i)}$, and L is the user-defined `search_range` parameter. That is, if $S^{(1)}$ and $S^{(k)}$ have the same length, the search for a matching fluke is limited to $[j_1 - L, j_1' + L]$. If $S^{(k)}$ is shorter than the guiding sequence $S^{(1)}$, then the search interval is extended by the length difference $|n_k - n_1|$ to the left, if it is longer, the interval is extended to the right.

`AnchoRNA` attempts to determine an anchor candidate for each position of the guiding sequence, starting with query word F_1 corresponding to the fluke $F^{(1)}$ in $S^{(1)}$. The fluke $F^{(1)}$ becomes part of the candidate anchor $\mathcal{A}$. The search proceeds then iteratively using an expanding list Q of query words starting from $Q = \{F_1\}$. In each step, the search covers the intervals $[p_j, q_j]$ of all sequences $S^{(j)}$ that have not been assigned a fluke in a previous step and determines the word W in these intervals with the maximal similarity score $\sigma(W) := \max_{F \in Q} \sigma(W, F)$ over all query words. If σ equals or exceeds a threshold $\sigma_{\text{add_anchor}}$, then the position of word W is added as

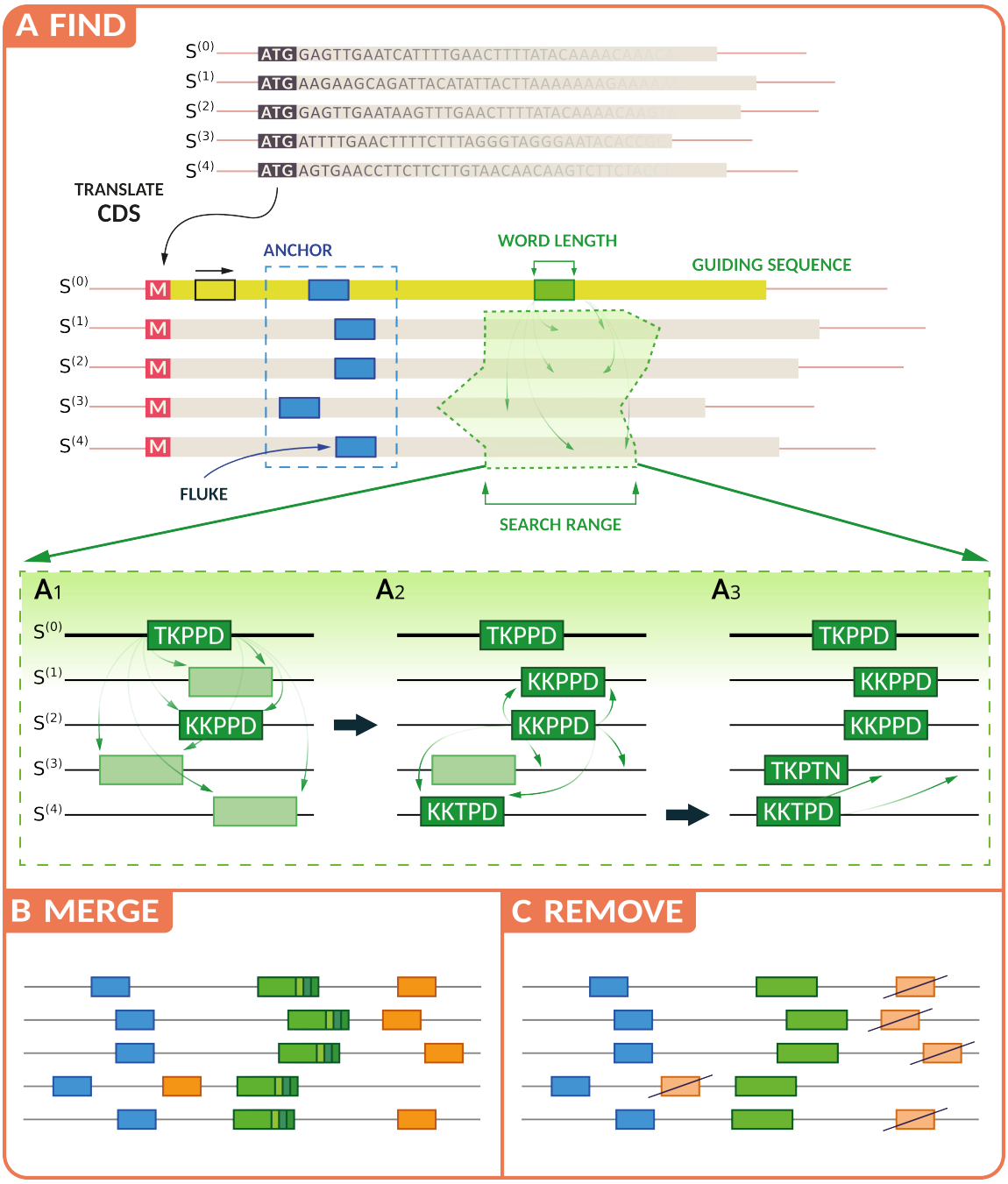

Figure 1: Illustration of the anchor detection algorithm. In this example the translation of the viral RNA sequence is used. Anchor construction is subdivided into three phases: **A** Detection of preliminary anchors starts from a query word in the guiding sequence. For each input sequence, a search window with user-defined width is defined. In the first pass, at most one exact match of the query word in all other sequences in detected. Then, the most similar matches in the remaining sequences are iteratively used as queries. A preliminary anchor consists of one fluke in each sequence, each of which is sufficiently similar to one of the query words. **B** Preliminary anchors are merged if they overlap and their flukes are separated by the same distance in all sequences. The merged anchor candidates are assigned a similarity score. **C** A simple greedy algorithm is used to remove all candidate anchors that still overlap or cross another anchor with higher similarity score. The result is a co-linear sequence of anchors, each of which corresponds to a gap-less local alignment.

fluke $F^{(j)}$ to the anchor candidate $\mathcal{A}$. If σ equals or exceeds a threshold $\sigma_{\text{add_word}} \geq \sigma_{\text{add_anchor}}$, then W is added to the set of query words. By default, the two thresholds are the same. In the first step, therefore, flukes that correspond to identical matches of F_1 are added to $\mathcal{A}$, and in later stages, increasingly different words W are obtained. The search ends if either all sequences have been assigned a fluke or if the highest-scoring match W does not reach the threshold $\sigma_{\text{add_anchor}}$, and thus no further fluke can be added to the anchor candidate. The candidate is discarded if the fraction of sequences with flukes does not reach a user-defined threshold. By default only complete anchors, i.e., those with a fluke in every sequence are retained as preliminary anchors.

Each preliminary anchor $\mathcal{A}$ is associated, by construction, with the set $Q_{\mathcal{A}}$ of query words that were used for its construction. Note that, in general, $Q_{\mathcal{A}}$ contains only a subset of the words F_k defined by flukes $F^{(k)}$. The initial query word W corresponding to the guiding fluke $F^{(1)}$, however, is always contained in $Q_{\mathcal{A}}$. In order to score the preliminary anchor $\mathcal{A}$, we proceed as follows: For each fluke $F^{(k)}$ of $\mathcal{A}$, we first determine the median $\hat{\sigma}_k$ of the similarity scores $\sigma(F^{(k)}, W)$ with all query words $W \in Q_{\mathcal{A}}$ and then compute the *anchor score* as the minimum of these median scores over all flukes in the candidate anchor: $\sigma(\mathcal{A}) := \min_k \hat{\sigma}_k$.

These initial anchors are processed in two steps. First, anchors $\mathcal{A}$ and $\mathcal{A}'$ with overlapping flukes $F^{(k)}$ and $F^{(k)'}$ are merged if the difference between the fluke positions, i.e., $f'_k - f_k$ is the same for all k. After merging, each anchor consists of flukes $\bar{F}^{(k)}$ in $S^{(k)}$ with a common length. The fluke and anchor scores are recomputed after merging.

In the second step anchors are removed to ensure consistency. Note that any overlapping anchors that have not been merged in the previous step necessarily contain contradictory position-wise matches in at least one pair of sequences. Two anchors $\mathcal{A}$ and $\mathcal{A}'$ *cross* if the start position of their respective guiding flukes satisfy $f_1 < f'_1$ and there is a sequence $S^{(j)}$ such that the corresponding fluke positions satisfy $f'_j < f_j$. Two preliminary anchors $\mathcal{A}$ and $\mathcal{A}'$ are *inconsistent* if they are still overlapping or crossing. A greedy algorithm is then applied to remove *inconsistent* anchors: Starting from the highest scoring preliminary anchor $\mathcal{A}$, all preliminary anchors $\mathcal{A}'$ that are inconsistent with $\mathcal{A}$ are removed. This step leaves a set of non-overlapping, co-linear anchors, each of which corresponds to a gap-less local alignment. The greedy removal procedure can be restricted to crossing anchors only by setting `aggressive_remove` to false.

Since `AnchoRNA` is designed to address in particular the high levels of sequence divergence commonly observed among viral genomes, the coding RNA sequence is translated to the corresponding amino acid sequence. This feature can also be omitted, e.g. to handle UTR sequences. For amino acid sequences, the `BLOSUM62` (Henikoff and Henikoff, 1992) matrix is used to evaluate the similarity score σ, since it accounts for synonymous mutations and mutations for which the corresponding codon is translated to an amino acid with similar properties. Alternatively, the user can choose one of the substitution matrices distributed within the `sugar` package (Eulenfeld, 2025b), which is particularly pertinent when analyzing nucleic acid sequences.

```
### Configuration for AnchoRNA in TOML format
fname = "pesti_example.gff"# Seq file with CDS annotation
w = 5                      # Word length
gseqid = "KC533775"        # ID of the guiding sequence
search_range = 100         # Search range [#letters]
scoring = "blosum62"       # Scoring matrix
score_add_word = 22        # Score to add word to Q set
         # flukes with lower score are marked as "poor"
thr_quota_add_anchor = 1   # Discard anchor if
                           # quota is not met (1=100%)
thr_score_add_anchor = 22  # Score threshold for quota
score_use_fluke = 22       # Score to use fluke in
                           # export, view, cutout
```

Figure 2: Configuration parameters for `AnchoRNA`.

Complexity and scalability

The time complexity of `AnchoRNA` is dominated by the initial search for preliminary anchors, Figure 1A. The effort depends linearly on the length n of the guiding sequence since `AnchoRNA` considers each position as a potential initial fluke for an anchor. Calculating the similarity scores of all words in the search range of the remaining sequences has a complexity of $\mathcal{O}(wSN)$, where S is the `search_range` parameter L plus the maximum of the absolute length differences between the guiding sequence and all other sequences, w is the word length of the query, and N is the number of input sequences. Finally, updating the best-scoring fluke candidates by a repeated search with new query words results in a runtime that depends quadratically on N resulting in worst case complexity of $\mathcal{O}(nwSN^2)$. For large $N \gg 100$, it is advisable to select a representative subset of sequence, see below.

Since `AnchoRNA` needs to store all sequences, the memory consumption amounts to $\mathcal{O}(nN)$. It is worth noting that the searches for matching flukes at different position j_1 of the guiding sequence are independent of each other and thus allow a straight-forward parallelization using multiple CPU cores (command line flag `--njobs`) leading to a linear reduction in runtime as the number of cores increases.

Implementation and Usage

`AnchoRNA` is implemented as a command line tool in python and makes use of the `sugar` (Eulenfeld, 2025b) package, a lightweight toolkit for manipulating sequence and annotation data. The behavior of `AnchoRNA` is affected by several parameters, most importantly the word length w, the size L of the flanking region defining the search range, the score cutoff for word matches, and the fraction of sequences with a valid fluke to form an anchor. `AnchoRNA` takes two arguments: a file containing the sequences and a configuration file (default name: `anchorna.conf`), see Fig. 2, that sets the parameters and specifies information such as the annotation of coding sequence and the choice of guiding sequence. The sequence file is parsed by the `sugar` package, which currently supports either a `.gbk` file, including annotations of viral protein positions; or a `.gff` file with `fasta` directive.

The `AnchoRNA` algorithm is executed by the `anchorna go` command. The resulting anchors are written to a file with a special format based on the specification of GFF. Support for the creation of the configuration file is provided by running `anchorna create`. The software is distributed with an example data set comprising 55 *Pestivirus* genomes, allowing the user to

recreate all examples shown in this contribution. In addition, a detailed tutorial is provided on the GitHub landing page (Eulenfeld, 2024).

Default parameter settings have been chosen to work well with this, and similar, data sets. The default length for the sliding window and initial anchors for protein sequences is $w = 5$ amino acids. The size of the query sequence can be increased for data with a higher conservation level. For Hepatitis C Viruses genomes, for example, we recommend $w = 8$. Applying `AnchoRNA` directly to nucleotide sequences, such as 5' UTRs and 3' UTRs, also requires larger values. We found $10 \leq w \leq 15$ to work well in such a scenario. The size of `search_range` is set to $L = 100$ by default. The length of the interval in which flukes are searched for is $2L + w$. Larger values are likely of use for input sequences with large length variations and for nucleic acid sequences. A potential anchor becomes a fixed anchor if a fraction of `thr_quota_add_anchor` (default: 100%) of the flukes score above $\sigma_{\text{add_anchor}}$ (`thr_score_add_anchor`, default 22 for protein sequences scored with the `BLOSUM62` similarity matrix). Reducing the `thr_quota_add_anchor` parameter makes it possible to find anchors that are present in a subset of input sequences only. While this can be useful in the presence of larger deletions in some sequences, it makes the results strongly dependent on the choice of the guiding sequence. It is beneficial to select the guiding sequences as the longest sequences in such cases. It should be kept in mind that threshold score, window length w, and scoring matrix options are dependent on each other. The default values were chosen so that the product of the word length and the identity score of a single *common* letter in the `BLOSUM62` matrix is slightly smaller than the score threshold.

General workflows

The current implementation of `AnchoRNA` supports input data comprising up to a few hundred viral genomes. For well-studied viruses, much larger sets of genomic sequence are available, many of which are then identical or near similar. It this case it is advisable to first perform a clustering by sequence similarity as described e.g. by Triebel et al. (2024, 2025). Sequences within a cluster are sufficiently similar to be aligned globally without the need of anchors. A representative or the consensus sequence of each cluster then serves as input for an `AnchoRNA`-based alignment. If necessary the alignment of the sequences within a cluster can also be performed with the help of `AnchoRNA`, which can be expected to produce a much denser set of anchors within clusters than among cluster representatives. The `AnchoRNA` software implements not only the `AnchoRNA` algorithm but also provides several additional functionalities to inspect and modify the anchor data, see Fig. 3 and Fig. S1 in the appendix. The `anchorna view` command allows the user to inspect the anchors in Jalview. Anchors can be overlaid on raw sequences or a preliminary alignment, facilitating the identification of erroneous anchors by visual inspection. The `anchorna combine` command can be used to manually delete erroneous anchors or select anchors. It can also be used to combine multiple anchor files by adjusting positions to each other. This may be necessary when using `AnchoRNA` in an iterative manner to search for anchors in "empty" regions of the sequence set where no anchors were identified in the initial, genome-wide run, see Fig. S1 in the appendix. The necessity to combine multiple anchor files can also occur, when anchors are determined separately in different coding sequences of the same viral genome. This strategy has proved useful for *Filoviridae*. `anchorna cutout` can be used to create, for each virus, a subsequence defined by locations relative to 5' and 3' anchors. The location strings must be from the group of strings defined by the regular

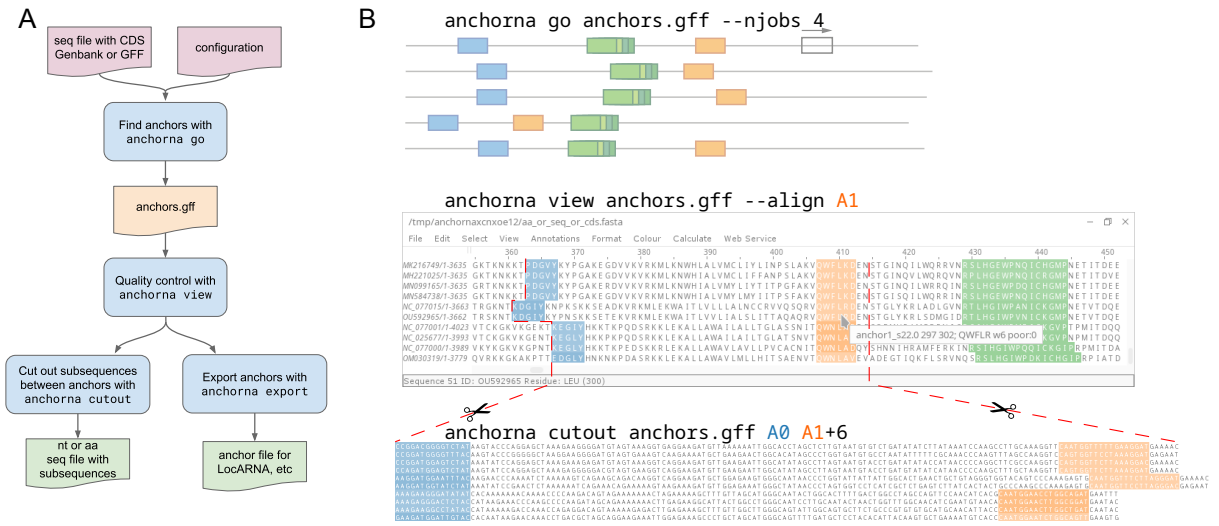

Figure 3: Basic `AnchoRNA` workflow. **A** The `anchorna go` command takes a sequence file and a configuration file and outputs anchors in GFF format. Anchor GFF files can be viewed with `Jalview` (Waterhouse et al., 2009) using the `anchorna view` command. The original or translated sequences can be cut at positions relative to the anchors with the command `anchorna cutout`. Anchors can be exported to different formats with the command `anchorna export`. A more advanced workflow is shown in Fig. S1 in the appendix. **B** Example usage of `AnchoRNA` commands based on the tutorial dataset. `anchorna go` is used with 4 CPUs; `anchorna view` shows anchors, here A0 (blue), A1 (orange, aligned), A2 (green); `anchorna cutout` cuts the nucleotide sequences from anchor A0 to anchor A1 and adds 6 nucleotides to the right.

expression `(A?\d+|ATG|*|start|end)([<^>])?([+-]\d+)?` (case is ignored). The regular expression consists of three groups: The first group gives the anchor number. Start and stop codons, as well as start and end of sequences, can also be used as anchors. The optional second group defines the location inside the anchor. It can be the 5' or 3' end, or the center. By default, the 5' or 3' end is used for anchors at the 5' or 3' end of the cutout region, respectively. The third group defines an optional constant character offset. For example the location `A1>+6` is six characters behind the 3' end of the second anchor. Sequences can be cut in such a way that the original sequences can be reconstituted using the `sugar merge` command provided by the `sugar` package (Eulenfeld, 2025b). This makes it possible to use `sugar merge` also to concatenate subalignment files output by alignment tools and generate the full alignment. This facility is useful in particular when alignment tools are used that do not natively support anchors and thus have to process the sequences between the anchors independently. Several alignment tools allow to input anchor files directly to guide the alignment process. The `anchorna export` command creates corresponding anchor files for the `Dialign` and `LocARNA` programs. The command can also be used to create `.gff` files with anchor positions in the nucleotide or amino acid sequences for each virus genome.

3 Results

3.1 Full-genome alignment of pestiviruses

We selected the *Pestivirus* taxon as showcase application because its genomes exhibit both conserved regions that can easily be aligned using standard tools and regions that require

Figure 4: *Pestivirus* genome of around 12 kb length adapted from Viralzone (Hulo et al., 2010) with anchors indicated by black and colored vertical bars. 55 valid anchors were found using the tutorial dataset of 55 *Pestivirus* sequences. Different alignments of the genome region marked with the horizontal bar and light gray box are shown in Fig. 6. The first four colored anchors in this region are displayed in Fig. 5. The distance between neighboring anchors ranges from 1 aa to around 1100 aa depending on position in the genome and sequence.

anchors for alignment. Moreover, Triebel et al. (2025) provided a carefully curated ground truth against which `AnchoRNA` can be compared. We downloaded all 1,684 *Pestivirus* genomes (as of October 2023) from the BV-BRC and NCBI databases (Olson et al., 2022; Sayers et al., 2021). Using the `ViralClust` pipeline (Lamkiewicz et al., 2025), as described by Triebel et al. (2024, 2025), the dataset was reduced to 55 representative genomes with an approximate length of 12 kb. These genomes were selected to ensure a well-distributed and balanced representation across the phylogenetic tree of the genus Pestivirus. The corresponding sequences are included as example data set with the `AnchoRNA` distribution. As a downstream application we aim to predict RNA secondary structures across the entire RNA genome alignment to evaluate the dependence of the prediction on the quality of the genome-wide alignments. We use `LocARNA` ($\mathcal{O}(n^2)$), a sparsified heuristic adaptation of the Sankoff algorithm ($\mathcal{O}(n^6)$) (Sankoff, 1985).

Using `AnchoRNA`, we translated all protein-coding regions into amino acids and identified 56 anchors across the entire genome. We identified a single erroneous anchor A8 with the `anchorna view` command, leaving 55 valid anchors. The anchors are 5–22 amino acids in length, spaced between 1 and 1100 amino acids apart, see Fig. 4. For a detailed view, we selected 24 sequences (at least one from each genotype) and highlighted anchor no. 11–13, which are easy to align, and anchor 14, which only `AnchoRNA` successfully aligned (Fig. 5), see below. Triebel et al. (2025) used these anchors to split up the *Pestivirus* genome sequences into several subsequences which they fed into the `LocARNA` algorithm. We use their resulting hand-curated *Pestivirus* alignment as ground truth.

3.2 Impact of anchors on different alignment tools

To validate our anchors, we first overlaid them on a `MAFFT` alignment using the `anchorna view` command. All anchor regions were aligned by `MAFFT`, except for the erroneous anchor mentioned above and anchor A14. The region between anchors A13 and A15 appears difficult to align due to mutations and large deletions and insertions. This region spans between 430–700 amino acids, and we will focus on it below.

We compared the impact of `AnchoRNA` anchors on several alignment tools. We chose `MAFFT` for its popularity and widespread use, and `MAFFT` with `AnchoRNA` to assess the effect of anchor-based refinement. `Dialign` was included because it is well-suited for handling insertions and deletions, while `Dialign` with `CHAOS` has been described to utilizes pairwise anchors to enhance alignment quality. We evaluated `Dialign` with `AnchoRNA` to integrate our anchor-based approach into the `Dialign` workflow. Finally, we also employed `LocARNA` in its standard form, alongside `LocARNA`

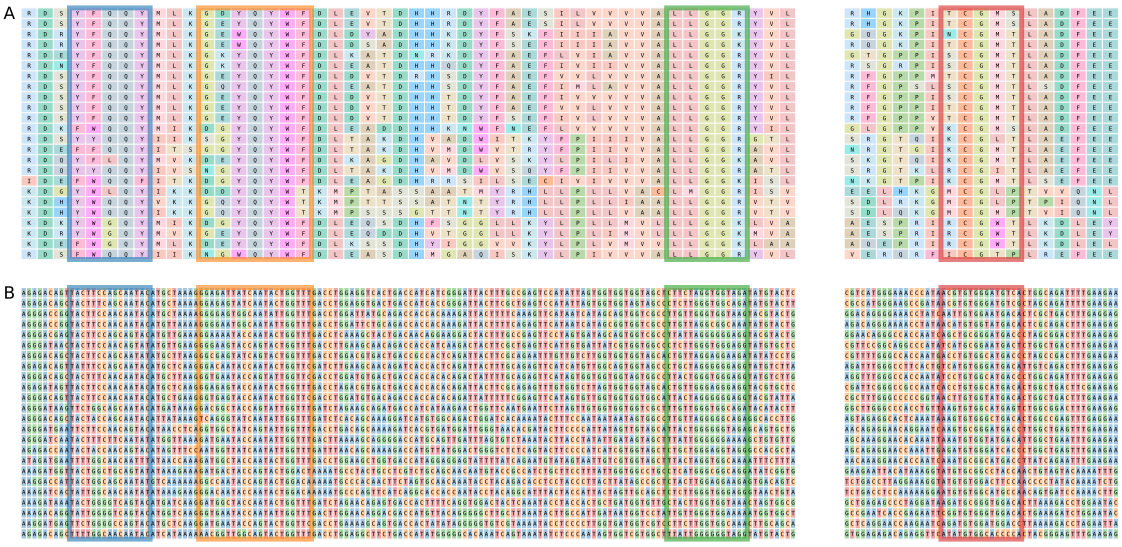

Figure 5: Sequences with four exemplary anchors found with `AnchoRNA` for the *Pestivirus* dataset. For visualization purposes only 24 sequences (each genotype represented) are displayed, with 1,000-1,600 nt omitted (white space). The four anchors correspond to anchor 11-14, which are displayed with the same colors in Fig. 4 and Fig. 6. `AnchoRNA` works on the amino acid level by default (**A**) and the anchors are transferred to the nucleotide level only afterwards (**B**).

guided by `AnchoRNA` anchors, and `LocARNA` applied to subsequences. The choice of using the computationally expensive tool `LocARNA` reflects the growing importance of analyzing RNA secondary structures across entire viral genomes.

The various tools necessitate different adaptations for alignment. We used the `Dialign` web service[1], which cannot process more than 50 sequences. Therefore, we reduced the dataset accordingly. On this reduced dataset of 50 sequences, `AnchoRNA` identified the same 55 valid anchors as in the full dataset of 55 sequences. For `AnchoRNA+Dialign`, we used the `anchorna export` command to export the anchors A14 and A41, which were not aligned by vanilla `Dialign`, along with the surrounding anchors, into the `Dialign` anchor file, which serves as a constraint for the `Dialign` alignment process. Note that not all 55 valid anchors were exported because Dialign could not handle so many constraints, which resulted in unaligned anchor regions. For `AnchoRNA+MAFFT`, we only chose anchor A14 as constraint because Vanilla `MAFFT` aligned all other valid anchor regions out of the box. We utilized the `anchorna cutout` command to split the genome sequences into two subsets at anchor A14, as `MAFFT` does not provide an option to directly incorporate anchors. This allowed us to force `MAFFT` to align the anchor, after which we concatenated the resulting alignments with the `sugar merge` command. For `AnchoRNA+LocARNA`, we exported all valid anchors, to the `LocARNA` anchor format using the `anchorna export` command. These anchors were then utilized as constraints for `LocARNA`, enabling more accurate alignment by guiding the alignment process based on the identified anchor points. In another approach, we split the genome at the anchors and fed each subsequence set into `LocARNA`, subsequently concatenating the resulting alignments.

For our tests, we use a standard PC with four 3.6 GHz i7-7700 cores, each with two threads, and 67 GB of memory. Two threads were used with `LocARNA`, while `MAFFT` used only a single

[1] `https://dialign.gobics.de/`, last accessed 07. April 2025

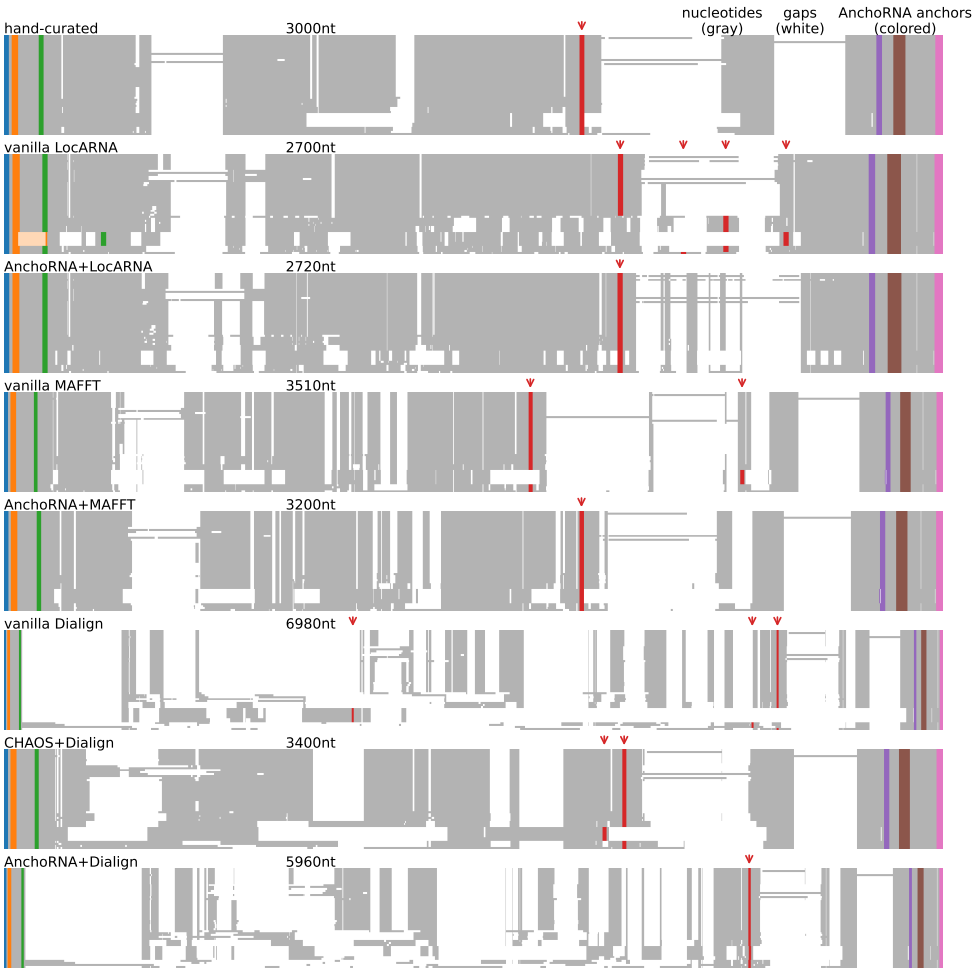

Figure 6: Part of the alignments produced by different tools with and without guidance by anchors. Anchors calculated by `AnchoRNA` are displayed over each alignment with colored bars. The positions of anchor 14 are marked with red arrows. Length of the displayed parts of alignments rounded to 10 nt are indicated. Each panel corresponds to one cell in Tab. 1. The guidance by anchors decreases alignment length for most tools.

Table 1: Mean alignment scores and alignment lengths (rounded to 100 nt) of the curated *Pestivirus* sequences using different alignment programs with and without guidance by anchors. The scores displayed are mean scores per letter using the `NUC4.4` matrix. A perfect mean alignment score is 5. `CHAOS` calculates anchors for pairs of sequences. It is therefore difficult to combine CHAOS anchors with `MAFFT` and `LocARNA`. NA – not applicable.

	vanilla	anchors with `CHAOS`	anchors with `AnchoRNA`	
`MAFFT`	0.30/16100	NA	0.44/15500	
`Dialign`	-1.11/24900	0.22/16900	-0.74/22300	
`LocARNA`	0.57/14300	NA	0.58/14300	with `export` command
			0.56/14300	with `cutout` command
hand-curated			0.57/14800	
runtime	≈ 2 min	≈ 2 h	≈ 12 h	≈ 2 d

thread. Fig. 6 displays parts of the resulting alignments for the various alignment approaches for *Pestivirus* genomes between `AnchoRNA` anchor no. 11 and 17. The most compact alignment-part, produced by `LocARNA`, measures ≈2,700 nt in length, while the widest alignment, optimized for longer insertions using `vanilla Dialign`, extends to ≈6,980 nt. Notably, while this example highlights a single anchor region at the nucleotide level that is not aligned by standard multiple sequence alignment methods, in more complex real-world scenarios, this issue typically affects numerous such regions. To measure a precise alignment score, we used Todd Lowe's `NUC4.4` scoring system[2], which assigns +5 for identical letters, -4 for different letters, additionally we set -15 for a new gap, and -1 for gap extensions. We calculate the alignment score for each sequence pair, compute the mean of these scores, and divide by the constant median length of the 50 sequences (12 296.5). This results in a mean alignment score per letter, with a perfect score of 5. Tab. 1 compares various alignment methods for the 50 *Pestivirus* genomes, both with and without anchor guidance. Without anchors, `MAFFT` produced a reasonable alignment with a length of 16.1 kb and a mean score of 0.30. In contrast, `Dialign` generated a much longer alignment being gap-rich of 24.9 kb length, resulting in a low mean score of -1.11. `LocARNA` produced the alignment with the highest score of 0.57 and an alignment length of ≈14.3 kb. When anchor guidance by `CHAOS` was introduced, `Dialign` benefits substantially, achieving a score of 0.22 with a more manageable alignment length of 16.9 kb. This result can be attributed to the fact that `CHAOS` calculates anchors specifically for pairs of sequences. In general, these pairwise anchors do not necessarily determine gap-less local alignment blocks and thus, do not induce a partition of the global multiple alignment problem. Pairwise anchors therefore cannot be coupled with arbitrary alignment algorithms. Instead, the alignment algorithm itself must be capable of handling pairwise alignment constraints. No such feature is available for `MAFFT` and `LocARNA`. We note that pairwise anchors need to satisfy non-trivial consistency conditions, which have been specifically designed into `Dialign` (Morgenstern et al., 2006).

All alignment methods benefited from the guidance of `AnchoRNA` anchors. `MAFFT` could be improved to a mean score of 0.44 and an alignment length of 15.5 kb. `Dialign` in combination with `AnchoRNA` led to an improvement with a score of -0.74 and an alignment length of 22.3 kb, but this was not as efficient as the results obtained with `CHAOS` anchors, which use a larger number of pairwise anchors. Note that we could have further improved the `AnchoRNA` +`Dialign` alignment by searching and incorporating anchors that span a subset of sequences in the crucial region between anchors A13 and A15. Finally, we input `AnchoRNA` anchors into `LocARNA`, which resulted in a slightly improved alignment score of 0.58, however, it still had an extensive runtime of around two days. Therefore, we also ran `LocARNA` on subsequences based on the calculated anchors, which yielded a slightly lower alignment score of 0.56 and an alignment length of 14.3 kb, but with a more reasonable runtime of 2 h. This result has a similar score as the manually curated ground truth alignment by Triebel et al. (2025), which achieved an alignment score of 0.57 and an alignment length of 14.8 kb. The manually curated alignments are partially based on pure secondary structure conservation, which leads to less optimal sequence alignments. Consequently, while the alignment score is similar, it is considered to more accurately reflect the evolution of the virus family.

[2] `ftp://ftp.ncbi.nih.gov/blast/matrices/`, last accessed on 21 Dec 2024, also distributed with the `sugar` package

4 Conclusion

The alignment of complete viral genomes is challenging due to the complexities associated with long sequences, extensive insertions and deletions, and high sequence divergence. Our study demonstrates that `AnchoRNA` effectively addresses these challenges by identifying conserved regions, or anchors, within the coding sequences, demonstrated at the example of pestiviruses. Utilizing a dataset of 55 representative genomes, `AnchoRNA` identified 55 valid anchors that are crucial for guiding the alignment process. The incorporation of these anchors led to major improvements across tested alignment tools, highlighting the effectiveness of `AnchoRNA` in enhancing alignment quality, especially in viral genomes where standard methods often fall short. Even in cases where alignments are not improved in terms of score, forcing the sequences to align in conserved anchor regions leads to a biologically more meaningful alignment (Triebel et al., 2025).

While not the primary focus of our tool, `AnchoRNA` is capable of identifying anchors in both amino acid and non-coding sequences without requiring translation. Though, effectiveness of the tool will drop when sequence conservation is low and structural anchors are desired. In practical terms, we found that `AnchoRNA` operates efficiently on a standard PC with four cores, processing the *Pestivirus* dataset of 55 sequences – each approximately 12 kb in length – in about one minute. This rapid processing time makes `AnchoRNA` a viable option for researchers requiring timely results without sacrificing the accuracy of alignment. The ability to handle viral genomes and other highly variable sequences sets `AnchoRNA` apart from existing methods, making it a valuable addition to the MSA toolkit.

Furthermore, the abundance and distribution of anchors identified by `AnchoRNA` represent a significant contribution for primer design across entire virus families. The conserved nature of these anchors ensures effective amplification of genetic material, which is crucial for diagnostics, epidemiology, and virology research. Therefore, integrating `AnchoRNA` into alignment and primer design processes not only enhances alignment quality but also makes a meaningful contribution to the broader field of viral research and management.

Availability and requirements

Project name AnchoRNA
Project homepage `https://github.com/rnajena/anchorna`
Releases are distributed via PyPI and archived at Zenodo (Eulenfeld, 2024)
Operating systems Linux, MacOS, Windows
Programming Language Python
Other requirements Python 3.11+, rnajena-sugar
License MIT

Data availability The necessary code, data, and results to reproduce Figures 5 and 6, Table 1, and parts of Figures 3B and 4 are available on GitHub at `https://github.com/rnajena/anchorna_paper` and archived on Zenodo (Eulenfeld, 2025a).

Acknowledgments We thank Daria Kubrak for drawing the scheme of the method. We happily developed the manuscript *in pitu* and therefore thank Niklas Basler from the ITN for his remarks. This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2051, Project-ID 390713860; NFDI4Microbiota (NFDI 28/1); and the Ministry for Economics, Sciences and Digital Society of Thuringia (TMWWDG), under the framework of the Landesprogramm ProDigital (DigLeben-5575/10-9). PFS acknowledges the financial support by the Federal Ministry of Education and Research of Germany (BMBF) through DAAD project 57616814 (SECAI, School of Embedded Composite AI), and jointly with the Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the programme Center of Excellence for AI-research *Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig*, project identification number: SCADS24B.

References

Angiuoli SV and Salzberg SL (2010), Mugsy: Fast multiple alignment of closely related whole genomes, *Bioinformatics*, 27(3):334–342, doi:10.1093/bioinformatics/btq665.

Berkemer S, Hoffmann A, Murray C, and Stadler P (2017), SMORE: Synteny Modulator of Repetitive Elements, *Life*, 7(4):42, doi:10.3390/life7040042.

Blanchette M, Kent WJ, Riemer C, Elnitski L, Smit AFA, Roskin KM, Baertsch R, Rosenbloom K, Clawson H, Green ED, Haussler D, and Miller W (2004), Aligning multiple genomic sequences with the threaded blockset aligner, *Genome Research*, 14(4):708–715, doi:10.1101/gr.1933104.

Bray N, Dubchak I, and Pachter L (2002), AVID: A global alignment program, *Genome Research*, 13(1):97–102, doi:10.1101/gr.789803.

Brudno M, Chapman M, Göttgens B, Batzoglou S, and Morgenstern B (2003a), Fast and sensitive multiple alignment of large genomic sequences, *BMC Bioinformatics*, 4(1), doi:10.1186/1471-2105-4-66.

Brudno M, Do CB, Cooper GM, Kim MF, Davydov E, Program NISCCS, Green ED, Sidow A, and Batzoglou S (2003b), LAGAN and Multi-LAGAN: Efficient tools for large-scale multiple alignment of genomic DNA, *Genome Research*, 13(4):721–731, doi:10.1101/gr.926603.

Darling ACE, Mau B, Blattner FR, and Perna NT (2004), Mauve: Multiple alignment of conserved genomic sequence with rearrangements, *Genome Research*, 14(7):1394–1403, doi:10.1101/gr.2289704.

Darling AE, Mau B, and Perna NT (2010), progressiveMauve: Multiple genome alignment with gene gain, loss and rearrangement, *PLOS One*, 5(6):e11147, doi:10.1371/journal.pone.0011147.

Delcher AL, Kasif S, Fleischmann RD, Peterson J, White O, and Salzberg SL (1999), Alignment of whole genomes, *Nucleic Acids Research*, 27(11):2369–2376, doi:10.1093/nar/27.11.2369.

Dubchak I, Poliakov A, Kislyuk A, and Brudno M (2009), Multiple whole-genome alignments without a reference organism, *Genome Research*, 19(4):682–689, doi:10.1101/gr.081778.108.

Edgar RC (2004), MUSCLE: Multiple sequence alignment with high accuracy and high throughput, *Nucleic Acids Research*, 32(5):1792–1797, doi:10.1093/nar/gkh340.

Eulenfeld T (2024), AnchoRNA, `https://github.com/rnajena/anchorna`, doi:10.5281/zenodo.12190267.

Eulenfeld T (2025a), Code repository for the AnchoRNA publication, `https://github.com/rnajena/anchorna_paper`, doi:10.5281/zenodo.15807301.

Eulenfeld T (2025b), Sugar: A Python framework for bioinformatics, *Journal of Open Source Software*, 10(111):8122.

Henikoff S and Henikoff JG (1992), Amino acid substitution matrices from protein blocks., *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, doi:10.1073/pnas.89.22.10915.

Huang W, Umbach DM, and Li L (2005), Accurate anchoring alignment of divergent sequences, *Bioinformatics*, 22(1):29–34, doi:10.1093/bioinformatics/bti772.

Hufsky F, Abecasis AB, Babaian A, Beck S, Brierley L, Dellicour S, Eggeling C, Elena SF, Gieraths U, Ha AD, Harvey W, Jones TC, Lamkiewicz K, Lovate GL, Lücking D, Machyna M, Nishimura L, Nocke MK, Renard BY, Sakaguchi S, Sakellaridi L, Spangenberg J, Tarradas-Alemany M, Triebel S, Vakulenko Y, Wijesekara RY, González-Candelas F, Krautwurst S, Pérez-Cataluña A, Randazzo W, Sánchez G, and Marz M (2023), The International Virus Bioinformatics Meeting 2023, *Viruses*, 15(10):2031, doi:10.3390/v15102031.

Hufsky F, Lamkiewicz K, Almeida A, Aouacheria A, Arighi C, Bateman A, Baumbach J, Beerenwinkel N, Brandt C, Cacciabue M, Chuguransky S, Drechsel O, Finn RD, Fritz A, Fuchs S, Hattab G, Hauschild AC, Heider D, Hoffmann M, Hölzer M, Hoops S, Kaderali L, Kalvari I, von Kleist M, Kmiecinski R, Kühnert D, Lasso G, Libin P, List M, Löchel HF, Martin MJ, Martin R, Matschinske J, McHardy AC, Mendes P, Mistry J, Navratil V, Nawrocki EP, O'Toole AN, Ontiveros-Palacios N, Petrov AI, Rangel-Pineros G, Redaschi N, Reimering S, Reinert K, Reyes A, Richardson L, Robertson DL, Sadegh S, Singer JB, Theys K, Upton C, Welzel M, Williams L, and Marz M (2020), Computational strategies to combat COVID-19: Useful tools to accelerate SARS-CoV-2 and Coronavirus research, *Briefings in Bioinformatics*, 22(2):642–663, doi:10.1093/bib/bbaa232.

Hulo C, de Castro E, Masson P, Bougueleret L, Bairoch A, Xenarios I, and Le Mercier P (2010), ViralZone: a knowledge resource to understand virus diversity, *Nucleic Acids Research*, 39(suppl_1):D576–D582, doi:10.1093/nar/gkq901.

Hölzer M and Marz M (2020), PoSeiDon: a Nextflow pipeline for the detection of evolutionary recombination events and positive selection, *Bioinformatics*, 37(7):1018–1020, doi:10.1093/bioinformatics/btaa695.

Katoh K (2002), MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform, *Nucleic Acids Research*, 30(14):3059–3066, doi:10.1093/nar/gkf436.

Käther KK, Remmel A, Lemke S, and Stadler PF (2025), Unbiased anchors for reliable genome-wide synteny detection, *Algorithms for Molecular Biology*, 20(1), doi:10.1186/s13015-025-00275-9.

Lamkiewicz K, Esquivel Gomez LR, Kühnert D, and Marz M (2023), Genome structure, life cycle, and taxonomy of Coronaviruses and the evolution of SARS-CoV-2, pages 305–339, Springer International Publishing, doi:10.1007/978-3-031-15640-3_9.

Lamkiewicz K, Triebel S, and Marz M (2025), ViralClust: A Nextflow pipeline to cluster viral sequences, `https://github.com/rnajena/viralclust`, doi:10.5281/zenodo.14586897.

Leimeister CA, Dencker T, and Morgenstern B (2018), Accurate multiple alignment of distantly related genome sequences using filtered spaced word matches as anchor points, *Bioinformatics*, 35(2):211–218, doi:10.1093/bioinformatics/bty592.

Madhugiri R, Fricke M, Marz M, and Ziebuhr J (2014), RNA structure analysis of alphacoronavirus terminal genome regions, *Virus Research*, 194:76–89, doi:10.1016/j.virusres.2014.10.001.

Morgenstern B (1999), DIALIGN 2: Improvement of the segment-to-segment approach to multiple sequence alignment., *Bioinformatics*, 15(3):211–218, doi:10.1093/bioinformatics/15.3.211.

Morgenstern B, Frech K, Dress A, and Werner T (1998), DIALIGN: Finding local similarities by multiple sequence alignment., *Bioinformatics*, 14(3):290–294, doi:10.1093/bioinformatics/14.3.290.

Morgenstern B, Prohaska SJ, Pöhler D, and Stadler PF (2006), Multiple sequence alignment with user-defined anchor points, *Algorithms for Molecular Biology*, 1(1), doi:10.1186/1748-7188-1-6.

Notredame C, Higgins DG, and Heringa J (2000), T-Coffee: A novel method for fast and accurate multiple sequence alignment, *Journal of Molecular Biology*, 302(1):205–217, doi:10.1006/jmbi.2000.4042.

Ogurtsov AY, Roytberg MA, Shabalina SA, and Kondrashov AS (2002), OWEN: Aligning long collinear regions of genomes, *Bioinformatics*, 18(12):1703–1704, doi:10.1093/bioinformatics/18.12.1703.

Olson RD, Assaf R, Brettin T, Conrad N, Cucinell C, Davis J, Dempsey D, Dickerman A, Dietrich E, Kenyon R, Kuscuoglu M, Lefkowitz E, Lu J, Machi D, Macken C, Mao C, Niewiadomska A, Nguyen M, Olsen G, Overbeek J, Parrello B, Parrello V, Porter J, Pusch G, Shukla M, Singh I, Stewart L, Tan G, Thomas C, VanOeffelen M, Vonstein V, Wallace Z, Warren A, Wattam A, Xia F, Yoo H, Zhang Y, Zmasek C, Scheuermann R, and Stevens R (2022), Introducing the Bacterial and Viral Bioinformatics Resource Center (BV-BRC): a resource combining PATRIC, IRD and ViPR, *Nucleic Acids Research*, 51(D1):D678–D689, doi:10.1093/nar/gkac1003.

Sankoff D (1985), Simultaneous solution of the RNA folding, alignment and protosequence problems, *SIAM Journal on Applied Mathematics*, 45(5):810–825, doi:10.1137/0145048.

Sayers EW, Bolton EE, Brister JR, Canese K, Chan J, Comeau D, Connor R, Funk K, Kelly C, Kim S, Madej T, Marchler-Bauer A, Lanczycki C, Lathrop S, Lu Z, Thibaud-Nissen F, Murphy T, Phan L, Skripchenko Y, Tse T, Wang J, Williams R, Trawick B, Pruitt K, and Sherry S (2021), Database resources of the national center for biotechnology information, *Nucleic Acids Research*, 50(D1):D20–D26, doi:10.1093/nar/gkab1112.

Thompson JD, Higgins DG, and Gibson TJ (1994), CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice, *Nucleic Acids Research*, 22(22):4673–4680, doi:10.1093/nar/22.22.4673.

Triebel S, Eulenfeld T, Ontiveros N, Sweeney B, Tautz N, and Marz M (2025), First full-genome alignment representative for the genus *Pestivirus*, *bioRxiv*, doi:10.1101/2025.05.22.655560.

Triebel S, Lamkiewicz K, Ontiveros N, Sweeney B, Stadler PF, Petrov AI, Niepmann M, and Marz M (2024), Comprehensive survey of conserved RNA secondary structures in full-genome alignment of Hepatitis C virus, *Scientific Reports*, 14(1), doi:10.1038/s41598-024-62897-0.

Velandia-Huerto CA, Berkemer SJ, Hoffmann A, Retzlaff N, Romero Marroquín LC, Hernández-Rosales M, Stadler PF, and Bermúdez-Santana CI (2016), Orthologs, turn-over, and remolding of tRNAs in primates and fruit flies, *BMC Genomics*, 17(1), doi:10.1186/s12864-016-2927-4.

Waterhouse AM, Procter JB, Martin DMA, Clamp M, and Barton GJ (2009), Jalview version 2–a multiple sequence alignment editor and analysis workbench, *Bioinformatics*, 25(9):1189–1191, doi:10.1093/bioinformatics/btp033.

Will S, Joshi T, Hofacker IL, Stadler PF, and Backofen R (2012), LocARNA-P: Accurate boundary prediction and improved detection of structural RNAs, *RNA*, 18(5):900–914, doi:10.1261/rna.029041.111.

Appendix: Extended workflow

Fig. S1 shows an extension of the workflow chart in Fig. 3. It includes commands to refine the anchor search in specific regions of the virus genome.

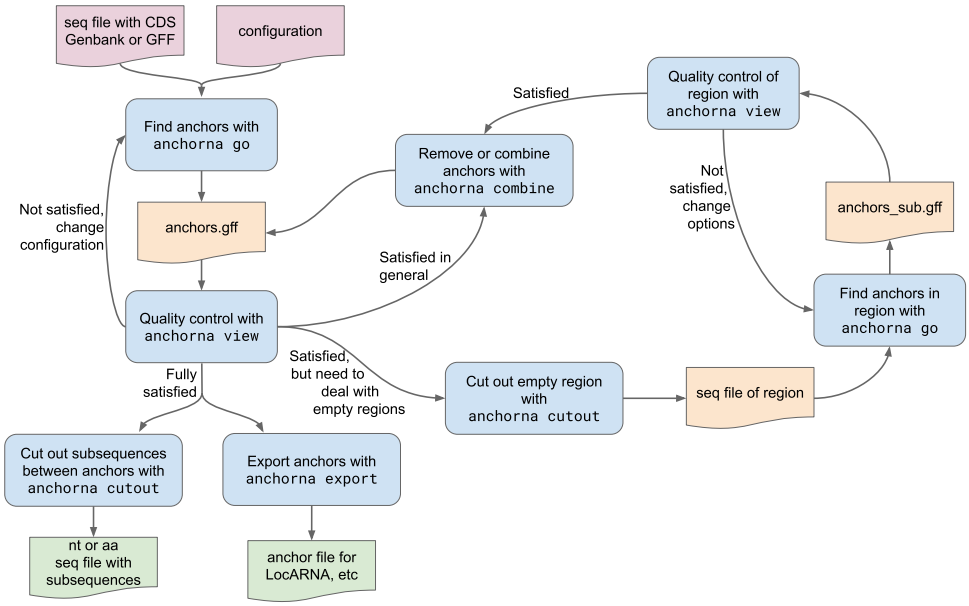

Figure S1: The place of the different `AnchoRNA` commands in the general workflow. The output of several commands, represented by orange data boxes, may be directly passed to other commands using a pipe.