FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

Friedrich-Schiller-Universität Jena

Fakultät Mathematik und Informatik

Wintersemester 2020/21

Evalution of VeGETAs core functions on *Influenza B virus* subsets

bearbeitet von: Alexander Henoch
Martikel-Nr.: 182709
betreut von: Daniel Desirò

Jena, den 22.10.2020

Contents

List of Abbreviations

A

AA aminoacid

B

BM2 BM2 ion channel

C

CDS coding sequence

CM2 CM2 ion channel

D

DMS dimethyl sulfat

G

GORS global ordered RNA structures

H

HA hemagglutinin

HEF hemagluthinin-esterase-fusion

I

IAV *Influenza A Virus*

IBV *Influenza B Virus*

ICV *Influenza C Virus*

IRD Influenza Research Database

IRES internal ribosome entry site

M

M1 matrix protein 1

M2 M2 ion channel

MED mean of euclidean distance

MMD mean of manhattan distance

mRNA messenger RNA

MSA multiple sequence alignment

N

NA neuraminidase

NB NB ion channel

NEP nuclear export protein

NS1 nonstructural protein 1

O

ORF open reading frame

P

PA polimerase acid protein

PB1 polimerase basic protein 1

PB2 polimerase basic protein 2

S

SD standard deviation

SNP single nucleotide polymorphism

ssRNA single stranded RNA

U

UTR untranslated region

V

vRNP viral ribonucleoprotein

List of Figures

List of Tables

1 Introduction

Virus research is a crucial field of molecular biological science, especially in the light of new pandemics nowadays. While not harming their natural host with any sign of illness, the viral infections, may lead to death or cause tremendous late effects in an accidental one (Wahlgren 2011). In many cases, humans can be these accidental hosts. An accidental host is a organism with permissive cells which allow the virus to replicate by circumvent the native defenses but, unlike the natural host, with pathogenic consequences. In contrast, the infection of a natural host leads to a lifelong infection, with high viral loads but no negative effects on the organism (Wahlgren 2011). Becoming the accidental host of a virus can happen by close contact with an already infected natural or accidental host, e.g. a human who has close contact to a mallard duck infected by the *Influenza A Virus* (IAV) (Jourdain et al. 2010). The IAV is the most widespread species of the virus family *Orthomyxoviridae* and also commonly known as flu (Eisfeld et al. 2015). While the mallard duck is one of the natural host of the IAV and the infection shows therefore no pathogenic behavior, the subsequent infection of the human shows a high immune response related with fever and a possible mortal outcome (Julkunen et al. 2000). The process of a human to get infected by a pathogen from a non-human animal is called zoonosis. Known examples are avian flu, swine flu and HIV (Van Reeth 2007; Sharp and Hahn 2011).

By entering the organism in a way that allows the virus particles to access types of cells that provide an opportunity to replicate themselves, an organism can become infected. The *Orthomyxoviridae* can only replicate themselves in epithelial respiratory cells, so the particles need to enter tissue of the respiratory tract, e.g. the lungs (Julkunen et al. 2000). The complete virus particles are called virions. They contain all necessary informations to build new virions, by replication of its genome and arrangement of new build proteins around it, to secure and enable the transport of the genome. The structure around the genome is called capsid (Cann 2005). Outside of a host cell, viral genomes are highly fragile and can be easily destroyed by physical forces or proteins build for degradation of nucleic acids like nucleases (Cann 2005). Capsids are most commonly formed as icosahedrons which is a highly effective trade-off between stability, volume and surface area. While protecting the genome from the hostile environment around the cells, they enable the injection of the genome into the cell by proteins on their surface (Cann 2005).

After replication of the virions inside the host cell and the following cell death, the new particles can infect other host cells and repeat the process.

Orthomyxoviridae genomes consist of negative single stranded RNA (ssRNA), that is not coated by a capsid. Instead the genome is surrounded by three layers of protection. The first layer consists of viral ribonucleoproteins (vRNPs) bound to the genome (Eisfeld et al. 2015). The vRNPs are surrounded by proteins called matrix protein 1 (M1) building the matrix as second layer and coated by the lipid envelope build from host cell lipids, as third and final layer (Eisfeld et al. 2015). Viruses possessing this more complex composition are called enveloped viruses. These three layers allow a more effective production of new virions because the host cell do not need to die to release new virions and can be slaved as a manufactory instead (Cann 2005). To enter the host cell for replication, a the protein on the surface called hemagglutinin (HA) attaches the virion to the sialic acid receptor (Eisfeld et al. 2015). The complete virion is then taken into the cell by a process called endocytose. After entering the cell, the lipid envelope is fused with the vesicle membrane of the host cell to migrate the genome into the nucleus for replication and production of proteins (Eisfeld et al. 2015). The genome is replicated in the host nucleus by the RNA polymerase complex, already bound to the virus genome consisting of polimerase acid protein (PA), polimerase basic protein 1 (PB1) and polimerase basic protein 2 (PB2) (Suarez 2000). Simultaneously, the genome is transcribed to positive stranded messenger RNA (mRNA) by the same polymerase complex and translated into proteins needed for the same process in the new virion. Export of the replicated genome from the nucleus is induced by the nuclear export protein (NEP) (Eisfeld et al. 2015). After assembly of the build proteins and replicated genome into a new virion, it leaves the cell through the plasma membrane with the help of neuraminidase (NA) while coating itself in its lipids (Eisfeld et al. 2015). All new virions created from the infection by one specific strain are called quasispecies (Kuroda et al. 2010).

In the virions lifecycle, synthesizing of proteins is inevitable. Protection of the new replicated genome by vRNPs, the building of its replication machinery from PA, PB1 and PB2, enable nuclear transport by NEP, or building the structure of the virion by HA, M1 and NA (Eisfeld et al. 2015). For a more effective way of replication, the genome of the *Orthomyxoviridae* consists of not only one very long negative ssRNA strand but instead between seven or eight smaller ones. The number of these small genomes, called segments, depends on the species (Eisfeld et al. 2015). Each segment is coding for at least one of the major proteins needed for the life cycle of the virus (Eisfeld et al. 2015). Subtypes of the species IAV from the *Orthomyxoviridae* are named after variations of the surface proteins HA and NA, e.g. H5N1 or H1N1 (Unknown author 1980). Only by connecting a receptor of the host cell with a matching antibody structure of the HA protein on the virion surface, a cell can become infected, and only by matching NA antibody, a

new build virion can leave the host cell (Eisfeld et al. 2015).

Changes in the composition of the virion, especially related to the proteins on the surface can lead to drastic changes in the infectivity and may result in even more dangerous pathogenic behavior (Eisfeld et al. 2015). These changes can occur by random mutations in the genome or by a process called reassortment. Mutation rates in negative ssRNA viruses like IAV are very high. One or two single nucleotide polymorphisms (SNPs) occur in every virion from the quasispecies (Duffy 2018). SNPs are the change of one specific nucleotide to another by e.g. missing error control of the RNA polymerase complex. When a SNP results in the change of the HA or PA protein structure, antigenic drift occurres (Webster 1999). The second and more drastic change in the virion structure of segmented viruses, like the IAV, can happen by simultaneous infection of one host cell by more than one virion. The new replicated segments can be mixed together in the new virions potentially creating more dangerous combinations from viruses with different surface proteins and potentially enabling zoonosis. This process is called reassortment and leads to events called antigenic shift (Nelson et al. 2008). Very dangerous, but human non-transferable subtypes, simultaneous infecting a non-human host cell with a human transferable, but harmless subtype, can reassort into a human transferable and dangerous subtypes (Nelson et al. 2008).

Species from the family *Orthomyxoviridae*, especially the IAV and the IBV, are a major threat to the human race by potentially developing deadly pandemics like the Spanish flu in 1918 (Nelson et al. 2008). To prevent the composition of new deadly strains or weakening existing ones, better vaccines need to be developed. One way is to get a better knowledge about the secondary structure of the viruses, which is the way nucleotides bind to each other to accomplish various functions, like starting the protein translation (Kieft 2008). With the mentioned virion structure and ways of infection in mind, the molecular mechanisms in the translation are of greater interest to find possible weak spots in the formation of vital structure proteins like HA and NA (Eisfeld et al. 2015).

To synthesize the necessary proteins, each genomic negative ssRNA segment is transcribed into positive mRNA. The single mRNAs can be divided into three pieces, the 5'-untranslated region (UTR), coding sequence (CDS) and 3-UTR (Moss et al. 2011). The CDS contains the information to be translated into aminoacids (AAs) by ribosomes in form of three nucleotide long codons (Cann 2005). The area of the CDS starting with the start codon ATG and ending with a stop codon, usually AUG, is called open reading frame (ORF) and synthesizes a chain of AAs later folded into a protein. Surrounding the CDS on both sides are the 3'-UTR on the 3'-end of the strand and the 5'-UTR on the other side (Cann 2005). UTRs are non-coding sequences and usually not translated into peptides. Instead they can form important secondary structures to enable translation mechanisms,

e.g. forming an internal ribosome entry site (IRES) which is in many viruses necessary for translation (Kieft 2008).

Considering the high mutation rate of *Orthomyxoviridae* and possible mechanisms to change its structure by reassembling its genome, it is not enough to get the secondary structure and weak spots of one quasispecies alone. To find wide-ranging vaccines, it is necessary to cluster complete strains of species and their various quasispecies into groups, find representatives of these groups and generate their secondary structure all together. That way weak spots in recurring and hopefully vital genomic regions can be discovered (Handl et al. 2005). These groups are called clusters. For clustering various methods can be used. Clusters can be build by concepts, like trying to accomplish the best compactness of the sequences or using hierarchical approaches for connection (Handl et al. 2005). Clustering techniques based on compactness ranging from comparison based on the nucleotide sequence identity threshold, like CD-HIT or usearch, to approaches using k-means algorithm (Handl et al. 2005; Li and Godzik 2006; Edgar 2010; Macqueen 1967). Clustering based on hierarchical clustering, like HDBSCAN connect sequences to cluster trees and use a given threshold to decide which sequences belongs to the same cluster (Campello et al. 2013).

The state of the art clustering tools usearch and CD-HIT are compared to a new tool named VeGETA using HDBSCAN for its clustering function (Lamkiewicz 2020; Edgar 2010; Li and Godzik 2006; Campello et al. 2013). They were used on random subsets of various FASTA files containing the (segmented) genomes of all strains of IBV. Clustering IAV segments is skipped here, because the enormous number of highly different sequences for every segment makes random subsets extremely various. Some random sequences out of many ten thousands with highly different sequence, result in nearly the same number of clusters as used random sequences, especially with sequence identity based algorithms, like the ones used by usearch and CD-HIT. ICV is skipped because there exist roughly 100 sequences per segment and the differences in these sequences are negligible, compared in a multiple sequence alignment (MSA). This is resulting in a small amount of clusters per tool, mostly only one, which is obstructive for the creation of a rating pipeline and meaningful statements about the clustering quality. The clusters of IBV were compared and their quality rated by a calculated score, using the clusters representative, with a vector based rating method created in this project. After calculation of the used algorithms clustering quality, VeGETA is used on the same random subset of sequences, to calculate the overall secondary structure of the clustered sequences. These resulting secondary structures are validated by literature.

2 Materials and Methods

All mentioned tools were installed by the Conda package distribution system version 2-2.4.0 (Miniconda) for a easier replication of the results (Anaconda Software Distribution 2020). By the use of Conda, Python packages and full software tools can be installed into environments without touching the system by single commands, making installation of many tools easy and fast. Therefore by execution of the Unix-Shell commands in Sec. 2.1 the same environment can be created as it was used for this project.

For complete replication of the results the file structure for the *script.sh* needs to be recreated in Unix-Shell by:

```
mkdir -p {Scripts,A/{01_PB2,02_PB1,03_PA,04_HA,05_NP,06_NA,07_MP,08_NS
    },B/{01_PB1,02_PB2,03_PA,04_HA,05_NP,06_NA_NB,07_M1_BM2,08_NS},C
    /{01_PB2,02_PB1,03_P3,04_HE,05_NP,06_M1_CM2,07_NS}}
```

The *script.sh* needs to be placed into the root of the structure on the same level as the folders *Scripts*, *A*, *B* and *C*. All other files needs to be moved to the *Scripts* folder.

After recreation the up to date sequences for every segment of every desired type must be retrieved from the Influenza Research Database (IRD) by entering

1. *SEARCH DATA*
2. *Search Sequences*
3. *Nucleotide Sequences*
    a) *Data Type: Genome Segments*
    b) *Virus Type: X*
    c) *Complete Genome: Complete Genome Only*
    d) *Select Segments: Y*
    e) *Complete: Y*
4. *search*
5. *download*

download the files and move them to the respective folder in the file structure.

The tools in the pipeline of the Sec. 2.2 were executed on every segment of the selected Influenza type. Data was obtained from the IRD on 13.06.2019 (Zhang et al. 2017). The final goal of the Sec. 2.2 pipeline is to rate the used clustering algorithms based on vector calculations, as well as a visualization of the evolutionary relationship and clustering differences of the genomes to every segment of the selected Influenza type. Possible secondary structures are furthermore predicted by VeGETA in the pipeline and to be validated.

2.1 Conda preparations

In the following, the used conda environments and related tools are listed as Unix-Shell commands. The names of the listed conda environments match the used environments in the pipelines and are necessary for execution without changing the pipeline scripts.

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
conda config --add channels r

conda create -n projektarbeit
conda activate projektarbeit

conda install ruby
conda install cd-hit
conda install mafft
conda install raxml
conda install newick_utils
conda install viennarna
conda install parallel
conda install jalview

conda deactivate

git clone https://github.com/klamkiew/vegeta.git
export PATH="$HOME/vegeta/bin:$PATH"
    #include in .Bashrc or .profile for permanent PATH modification

conda create -n vegeta python=3.6
conda activate vegeta

conda install -c pip
conda install -c cd-hit
conda install -c mafft
```

```
conda install -c locarna
conda install -c glpk

pip install numpy biopython colorlog umap-learn hdbscan docopt scipy

conda deactivate
```

2.2 Project pipeline

A visual summary of the full pipeline is shown in Fig. 2.1. The pipeline can be executed by the following Bash script. For execution of all Bash scripts GNU Bash is used in version 5.0.17(1)-release (Fox 1989).

> **script.sh**
>
> -p [number of processes]
> -i [path input directory]

```
cd Projektarbeit/Cluster/
bash -i script.sh -p 8 -i B/ > .log
```

The pipeline was only executed on *Influenza B Virus* (IBV). By expanding the command with `-i A/ -i C/`, the pipeline can be used on every segment from all Influenza types including *Influenza A Virus* (IAV) and *Influenza C Virus* (ICV). The `-p` option changes the number of parallel processes for used tools and can be changed for performance reasons.

2.2.1 Preparation of sequences

The FASTA file containing the segment sequences needed to be prepared by deleting duplicates, checking of sequences direction (positive or negative strand), extracting 500 random sequences (to keep a rational calculation time) and building a sequence alignment from the 500 sequences.

> **cd-hit-est**
>
> -M [memory limit]
> -s [length difference cutoff]
> -C [sequence identity threshold]

-T [number of processes]
-i [path input FASTA]
-o [path output FASTA]

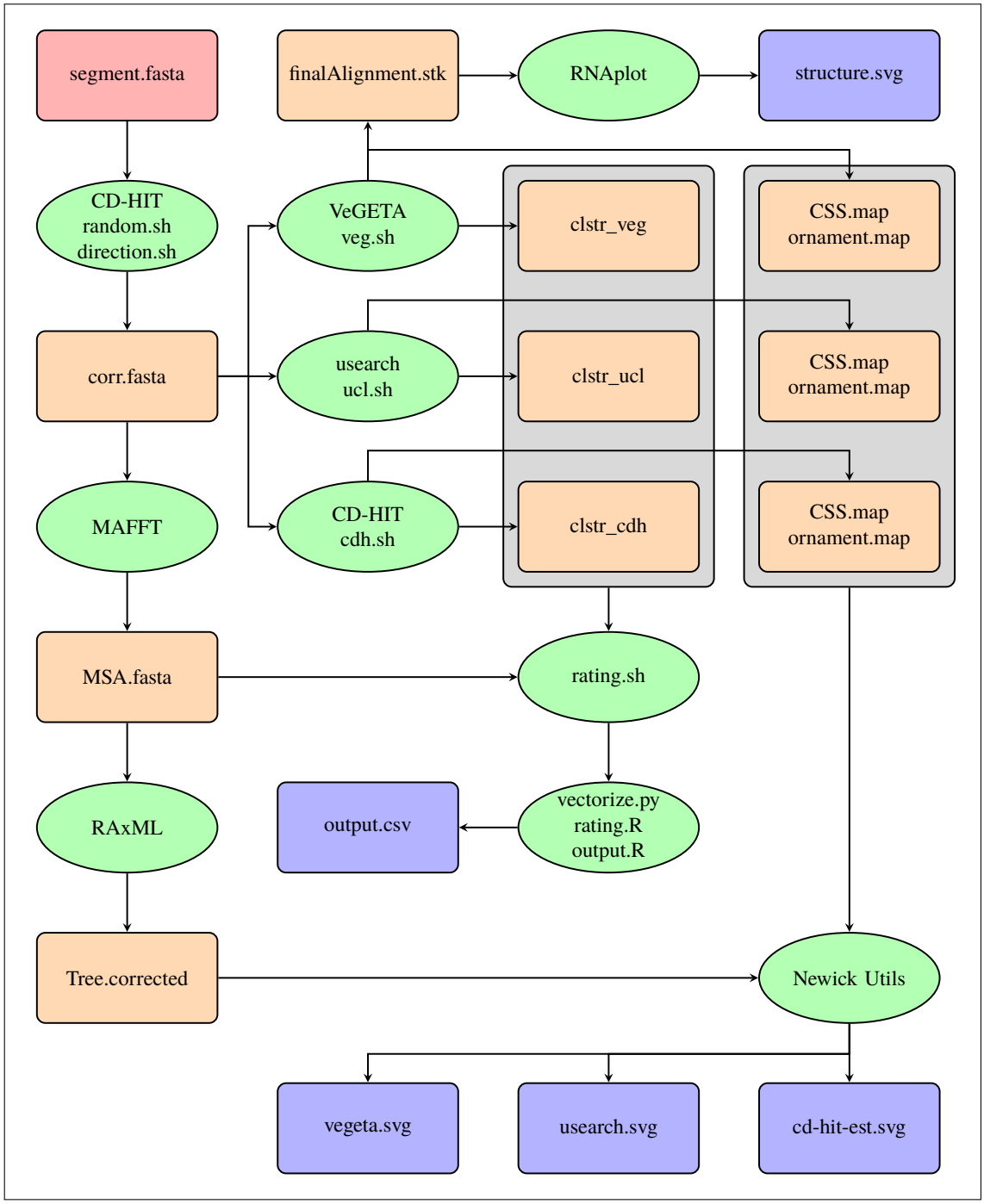


Fig. 2.1 Project pipeline. Visual representation of the described pipeline. Filenames boxed with red coloring are the input for the pipeline. Results are boxed in blue. The used tools described in the following are circled in green. If more than one tool is inside one circle the tools interact with each other or were used simultaneously. Interim results resulting from the used tools are colored in orange. Interim results inside a grey box pointing to a tool are used as input together.

Deletion of duplicate sequences were realised by CD-HIT version 4.8.1 using parameters to ensure that only duplicates were deleted that have a sequence identity of 100% and the same length by using `-C 1.0` and `-s 1.0`. Thus no shorter sequences or sequences with single nucleotide polymorphisms or other differences were deleted. Resulting from this execution, a FASTA file named *unique.fasta* was created (Li and Godzik 2006).

random.sh

-i [path input FASTA]
-o [path output FASTA]
-n [max. number of random sequences]

For randomization a Bash script was used to first mark every sequence in the *unique.fasta* with a sequential number and then generate 500 random but unique numbers between one and the number of sequences in the file. Based on the 500 random numbers the complete sequences with header matching these numbers were copied to a new FASTA file named *rndm.fasta*. Every sequence from the original file was copied if the original FASTA contained less than 500 sequences.

direction.sh

-i [path input FASTA]
-o [path output FASTA]

The correction of the sequence direction was, like the randomization, implemented by a Bash script. Every sequence from the FASTA file containing the random sequences were examined. The sequences in original and reverse transcribed direction as well as with one and two missing nucleotides at the sequence start in original and reverse transcribed direction were split in chunks of three nucleotides. These chunks in the six possible variations were translated to aminoacids (AAs). Every possible reading frame can be evaluated to find the longest open reading frame (ORF). The sequence variation containing the longest ORF in Influenza genomes has the highest possibility to match the real sequence direction. In case the direction wasn't the original sequence direction, the original sequence without missing nucleotides was reverse transcribed and copied to a new FASTA file named *corr.fasta* otherwise the original sequence without missing nucleotides was copied.

mafft

--auto automatic options (switch)

--reorder	aligned output order (switch)
--quiet	don't show progress (switch)
--thread	[number of processes]
	[path input FASTA]
>	[path output alignment]

MAFFT version v7.471 was then used on the *corr.fasta* file to build a multiple sequence alignment (MSA) named *MSA.fasta* with standard options plus reordering the sequences by option `--reorder` (Katoh 2002).

jalview

-nodisplay	don't show GUI (switch)
-colour	[colorscheme]
-open	[path input alignment]
-svg	[path output svg]

The sequence alignment was afterwards visualized by Jalview version 11.0.8-internal to have the possibility to examine differences in regions with noticeable structures. As a coloring scheme the standard nucleotide scheme was used and saved as *MSA.svg* (Clamp et al. 2004).

2.2.2 Clustering of sequences

The sequences from the file produced by the preparation steps *corr.fasta* was clustered by usearch v11.0.667_i86linux32, VeGETA v0.3 (alpha) and CD-HIT version 4.8.1 to rate differences in the clustering result (Edgar 2010; Li and Godzik 2006; Lamkiewicz 2020).

cd-hit-est

-T	[number of processes]
-n	[word length]
-C	[sequence identity threshold]
-i	[path input FASTA]
-o	[path output representatives file]

First clustering was performed by CD-HIT with a sequence identity of 95% by `-C 0.95` using the *corr.fasta* as input resulting in files containing the clusters and representatives (Li and Godzik 2006).

cdh.sh

-c [path input cluster file]
-r [path input representatives file]
-o [path output directory]

For the coloring in the evolutionary tree, the cluster and representative files from CD-HIT were restructured and exported as *ornament.map* and *css.map* files by the given Bash script. Simultaneously, the script restructured the information again into a file holding the clusters with related representative and sequence names for the rating step named *clstr_cdh*.

usearch

-cluster_fast [path input FASTA]
-id [sequence identity threshold]
-centroids [path output representatives file]
-clusters [path output cluster files]

Second clustering was performed by usearch with a sequence identity matching the one used with CD-HIT by setting `-id 0.95` and with the same FASTA used before (Edgar 2010).

ucl.sh

-c [path input cluster files]
-r [path input representatives file]
-o [path output directory]

In the same manner as the script before this Bash script restructured and exported the informations given by the usearch files into *ornament.map* and *css.map* files, as well as a custom file named *clstr_ucl*.

vegeta

[path input FASTA]
-o [path output dir]
-p [number of processes]

Third clustering was performed by VeGETA with standard settings on the same FASTA which was used in the other cluster tools (Lamkiewicz 2020).

veg.sh

-c	[path input cluster file]
-r	[path input representatives file]
-o	[path output directory]

Again, the informations were exported into *ornament.map*, *css.map* and a custom cluster information file named *clstr_veg* by the given Bash script fitted for execution on VeGETA cluster and representatives files. In conclusion, three *ornament.map* files, three *css.map* files and three cluster information files were generated by the three Bash scripts. One of each for one used clustering tools.

RNAplot

-i	[path input stockholm file]
-a	input is in stockholm format (switch)
-o	[output format]
-t	[layout-type]

The stockholm file created by VeGETA was visualized using RNAplot from the ViennaRNA package version 2.4.15 with svg output settings and circular layout type `-t 2` (Lorenz et al. 2011).

mafft

--auto	automatic options (switch)
--reorder	aligned output order (switch)
--quiet	don't show progress (switch)
--thread	[number of processes]
	[path input FASTA]
>	[path output alignment]

To be able to find conserved sequences later, MAFFT was used again with the same settings to calculate the MSA of the representatives for every tool.

2.2.3 Evolutionary distance analysis

raxmlHPC-PTHREADS-SSE3

-T [number of processes]
-N [number of alternative runs]
-f [used algorith]
-x [integer number]
-p [random number seed]
-m [model of substitution]
-s [path input alignment]
-w [path output directory]
-n [name of the output file]

RAxML was used to build trees giving information of the evolutionary distances of the genomes from the MAFFT alignment *MSA.fasta*. Recommended settings from the RAxML manual (`-N 100` , `-x 1234` , `-p 1234` and `-m GTRGAMMA`) were used with a custom name *RAxML_bipartitionsBranchLabels.Tree* for the output file (Stamatakis 2014).

raxml2drawing.rb

[path input RAxML tree file]

For the visualization of the RAxML tree, a Ruby script written by Martin Hölzer was used to shorten the evolutionary distance values in the file itself, for clearer display (Matsumoto 1995).

nw_topology

[path input RAxML tree file]
> [path output RAxML topology file]

To visualize the RAxML tree as a radial cladogram, nw_topology and nw_display from the Newick Utilities 1.6 were used (Junier and Zdobnov 2010).

nw_display

-v [number of pixels between leaves]
-i [CSS for inner node labels]
-l [CSS for leaf node labels]

-I [position of the inner node label]
-w [set width or scale]
-s output as svg (switch)
-r output as radial tree (switch)
-b [CSS for branch length labels]
-c [path input css file]
-o [path input ornament file]
[path input RAxML topology file]
> [path output svg]

The topological information was generated by nw_topology and then visualized and saved as svg by nw_display. The settings used were of decorative nature to generate a clear and readable svg-file. Furthermore, the generated *css.map* and *ornament.map* files from the Bash scripts were used here to color the generated clusters and representatives by usearch, VeGETA and CD-HIT in the radial tree. In conclusion, three different svg-files with the same radial tree but, different coloring were created, one for every used clustering tool with the corresponding clustering of the sequences represented by the color scheme (Junier and Zdobnov 2010).

2.2.4 Rating of clustering quality

To rate the clustering quality of each used clustering tool, a Bash script was used which takes a combined text file containing the custom made cluster information files and the *MSA.fasta* generated in the earlier steps as input.

rating.sh

-m [path input alignment]
-o [path output directory]
-i [path input cluster information file]

parallel

--keep-order keep order of input (switch)
--progress show progress (switch)
--jobs [number of parallel jobs]
[command] [arguments]

vectorize.py

[path input codon table]
[path output directory]
[step size]
[window size]
[path input cluster FASTA file]

rating.R

[path input cluster vector file]

output.R

[path input temporary rating files]
[path output rating file]

Based on the clusters generated by the clustering tools, the script calculates a vector for every sequence in the cluster by calculating the distance to the representative of the cluster. This was done by using GNU Parallel 20200722 for multiple executions of a python 3.8.5 script (Tange 2020; Van Rossum and Drake Jr 1995). The calculation of the distance is based on the differences of the nucleotides at the same positions by a cost function using the length uniformed sequences from *MSA.fasta* (siehe Tab. 2.1). This was done for every cluster from every clustering tool.

	A	**C**	**G**	**U**	**N**	**-**
A	0	3 (1)	2 (1)	3 (1)	0	3
C	3 (1)	0	3 (1)	2 (1)	0	3
G	2 (1)	3 (1)	0	3 (1)	0	3
U	3 (1)	2 (1)	3 (1)	0	0	3
N	0	0	0	0	0	3
-	3	3	3	3	3	0

Tab. 2.1 Costs used for cluster rating. For conversion of the differences between the representative (nucleotides on top) and the sequences in the cluster (nucleotides left) the given costs were used. If no difference is found, no costs were recorded. Single nucleotide polymorphisms (SNPs) without changing the AA, the triplet is coding for, is called silent mutation and rated by the value inside the braces. If the nucleotide changed is still a purin- or still a pyrimidinbase the SNP called transition is rated by the cost of value two. In all other cases of SNP as well as deletion or insertion it is rated by value three.

Following the calculation of the vectors the bash script executed two scripts using R version 3.6.3. The first one to calculate the mean of euclidean distance (MED), as well as the mean of manhattan distance (MMD), for result comparison with different distance functions,

on all m vectors related to one cluster together (Eq. 2.1 and Eq. 2.2). The calculation was done by two consecutive summation containing the euclidean or manhattan distance calculation in the middle. That way the distance between all the m vectors were measured by using the distance functions on vector n and o from m and finally divided by the number of calculations necessary.

$$\text{MED} = \sum_{n=1}^{m-1}\left(\sum_{o=n+1}^{m}\left(\sqrt{\sum_{i}^{j}(n_i - o_i)^2}\right)\right)\cdot\binom{m}{2}^{-1} \tag{2.1}$$

$$\text{MMD} = \sum_{n=1}^{m-1}\left(\sum_{o=n+1}^{m}\left(\sum_{i}^{j}|n_i - o_i|\right)\right)\cdot\binom{m}{2}^{-1} \tag{2.2}$$

This process was repeated for every cluster of every used tool (Tab. 2.2). The second R script outlines these results for every cluster tool, by calculating the weight of each MED of MMD and sum them up for every specific tool. This was done by building the product of the MED or MMD with the fraction of clustersize and total number of used sequences for clustering (Tab. 2.3 and Eq. 2.3 to Eq. 2.5) (R Development Core Team 2006).

tool	**cluster**	**MED**	**MMD**	**cluster size m**
CD-HIT	0	8.901	65.963	102
usearch	0	8.922	66.324	102
VeGETA	0	6.595	38.765	66
VeGETA	1	4.259	14.588	17
VeGETA	2	13.318	88.444	10
VeGETA	3	6.994	43.800	6
VeGETA	4	145.894	7095.000	1
VeGETA	5	145.894	7095.000	1
VeGETA	6	145.894	7095.000	1

Tab. 2.2 Example result from *rating.R*. By execution of *rating.R* the vectors of every sequence in the cluster containing the costs were used to calculate the MED inside the cluster.

The MED and MMD of single sequences clusters, were calculated by creating two vectors, one with the highest penalty three in every entry and one with the lowest zero. The distance of these two vectors is the highest possible for the given length. The resulting very high MMD and MED values punish tools for building of single sequence clusters.

tool	MED	MMD
CD-HIT	8.901	65.963
usearch	8.922	66.324
VeGETA	10.985	247.438

Tab. 2.3 Example result from *output.R*. By execution of *output.R* the results from *rating.R* were weighted by the cluster sizes to the total number of sequences and summed up for every tool.

$$8.901 = 8.901 \cdot 102 \cdot 102^{-1} \tag{2.3}$$

$$8.922 = 8.922 \cdot 102 \cdot 102^{-1} \tag{2.4}$$

$$\begin{aligned} 10.985 &= 6.595 \cdot 66 \cdot 102^{-1} \\ &+ 4.259 \cdot 17 \cdot 102^{-1} \\ &+ 13.318 \cdot 10 \cdot 102^{-1} \\ &+ 6.994 \cdot 6 \cdot 102^{-1} \\ &+ 145.894 \cdot 1 \cdot 102^{-1} \cdot 3 \end{aligned} \tag{2.5}$$

2.3 Tool versions

The versions of the mentioned tools used in Sec. 2.2 are listed in Tab. 2.4.

Tool	Usage	Version
GNU Bash	Preparation of sequences Clustering of sequences Rating of clustering quality	5.0.17(1)-release
CD-HIT-EST	Preparation of sequences Clustering of sequences	4.8.1
MAFFT	Preparation of sequences Clustering of sequences	v7.471
Jalview	Preparation of sequences	11.0.8-internal
usearch	Clustering of sequences	v11.0.667_i86linux32
VeGETA	Clustering of sequences	v0.3 (alpha)
RNAplot	Clustering of sequences	2.4.15
RAxML	Evolutionary distance analysis	8.2.12
Ruby	Evolutionary distance analysis	2.6.6p146
Newick Utils	Evolutionary distance analysis	1.6
GNU Parallel	Rating of clustering quality	20200722
Python	Rating of clustering quality	3.8.5
R	Rating of clustering quality	3.6.3

Tab. 2.4 Tool Versions. The versions of all the used tools and programming languages in Sec. 2.2, in order of usage without repetition.

3 Results and Discussion

3.1 Rating of clustering quality

3.1.1 Detailed vectorization

Execution of the *rating.sh* script was performed on the files generated in the used pipeline (Chap. 2), containing the clustering tools and their related clusters, as well as the representatives of the clusters and the sequences in the clusters. In *rating.sh*, the sequences related to the cluster were extracted from a multiple sequence alignment (MSA) and converted to a frameshift-, gap, and open reading frame (ORF)-aware sequence difference vector, by a python script named *vectorize.py* (Fig. 3.1).

To calculate these vectors, the following procedure was performed once for every sequence in the cluster, as well as once for the clusters representative. First the gaps were removed from the strand, resulting in the original sequence before the alignment. Following the removal, the ORF of the sequence was extracted by starting from the first nucleotide and proceeding in triplets, until a start codon was found. The nucleotides following the start codon were saved while counting the length of the ORF. When a stop codon was found, the sequence from start to stop was saved in a second variable and the first one was emptied. Read-in of the triplets proceeded until a new start codon was found, from which on the sequence was saved again in the emptied first variable, while the length was counted again. Only if the second sequence from start codon to stop codon was longer than the first one, the second variable was emptied and filled by the second found sequence. The same procedure was repeated until the whole strand was searched for every possible ORF. The longest ORF and its related candidate nucleotide sequence was then saved in a third variable. Furthermore, a second and third candidate sequence was searched by shifting the triplets by one and two and repeat the process again. Finally the longest candidate was accepted as the most likely ORF related to this sequence in the cluster and finally saved.

With the found ORF sequence in mind, the length of the 5'-untranslated region (UTR) and 3'-UTR were calculated. The saved ORF sequence was converted to a list of single

nucleotides. Every nucleotide from the list was then replaced by its related aminoacid (AA), e.g., the nucleotides *A*, *U*, *G* were replaced by methionine. Single gaps were afterwards added to the start of the list, their number according to the length of the 5'-UTR nucleotides and likewise added to the end of the list matching the 3'-UTR length. The previous removed gaps were also included at their exact removed positions. In conclusion, the procedure created a list reflecting a related AA to every nucleotide of the strand if the nucleotide was in the ORF or otherwise a gap (Fig. 3.1 bottom left). This more complex approach served the conservation of related AAs to the single nucleotides in case of frameshifts, gaps in the triplets, and longer or shorter UTRs.

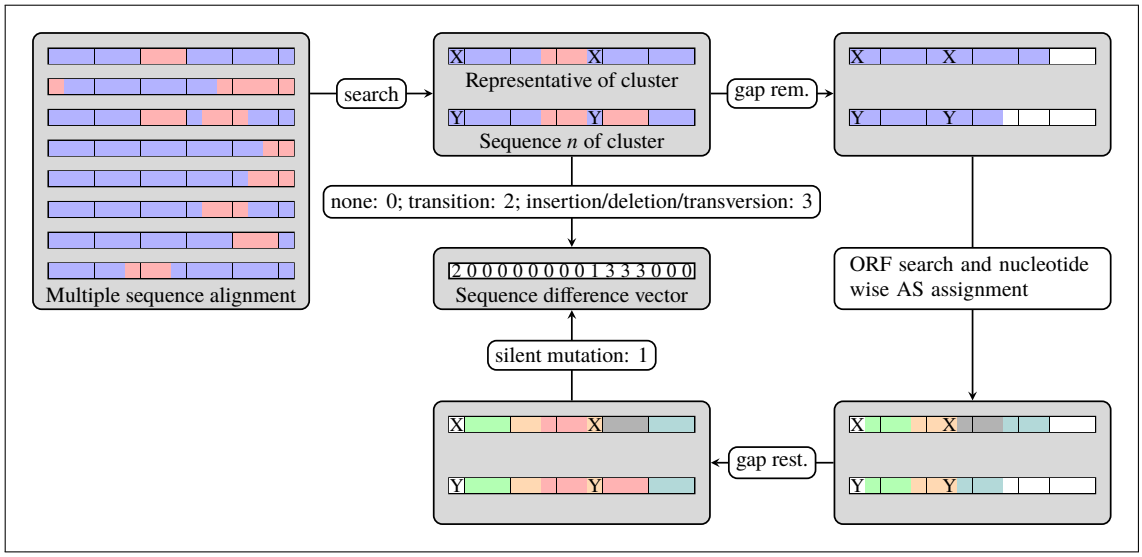


Fig. 3.1 Detailed procedure of *vectorize.py*. For the rating of clustering quality the representative of the cluster, as well as the sequence *n* was extracted from the previous generated MSA. Gaps in both sequences were removed and the AAs in the ORF annotated to the related nucleotides. With information of the AAs in mind, both sequences from the MSA were compared nucleotide-wise for no change, transition, insertion, deletion, transversion or silent mutation and rated accordingly in the sequence difference vector. This process was repeated for every sequence *n* of the cluster.

The procedure was first performed on the representative and then iteratively for every sequence in the cluster. After the procedure was finished for the first sequence of the cluster, there were two sequences from the MSA, one for the representative and one for the first cluster sequence, as well as two lists representing their AAs constellation. The first vector of sequence difference was then calculated by simultaneously iterating the two MSA sequences. If the nucleotides matched, the value zero was saved in the vector at the same position as the nucleotide. If the nucleotides did not match, the related positions in the two lists were compared for the same AA and by success rated with the value one. If there was still no match, the two MSA sequences were compared for transition at the position (A $\leftrightarrow$ G or U $\leftrightarrow$ C) and rated with the value two. Insertion, deletion, or all other types of single nucleotide polymorphisms (SNPs) were rated by the value three (the highest penalty). After the first vector was finished, only the MSA sequence and list for

the first cluster sequence were deleted and the process started again, for the next cluster sequence. MSA sequence and list for the representative were kept for the next comparison.

The substitution matrix used in the process was build by considering evolutionary aspects (insertion, deletion, transversion, transition, silent mutations), to assign nucleotide differences with specific scores, for a reasonable estimation of the clustering quality (Altschul 2008). Despite that, the rating process was build for easy customization in mind. The components were build to allow a way more complex transition table instead, by only changing some lines in the script pipeline. While calculation of the penalty, the nucleotide of the representative and sequence as well as the related triplet of the representative and sequence are available and considered. By changing the penalty calculation from the simple progressive if checkback to a lookup of the four components in a more specific transition table, a more accurate rating can be accomplished and the pipeline can be modified for specific questions.

3.1.2 Run time estimation of the rating process

Under the assumption that the used MSA of the genomes from segment X contains m sequences of length o and the cluster to be rated contained n of these sequences, the duration was calculated as follows:

$$m \cdot (n+1) + 6 \cdot o \cdot (n+1) + 2 \cdot o \cdot n + n^2 = O(n(m+n+o))$$

Every sequence in the cluster was extracted from the MSA once. The clusters representative instead twice, as a normal sequence in the cluster and as the clusters representative to compare the other sequences with. This resulted in

$$m \cdot (n+1) = O(m \cdot n)$$

operations.

All extracted sequences an the representative $(n+1)$ were lapsed six times to remove the gaps, search for the ORF three times (to search in every shifting possibility), annotate the AAs and restore the gaps:

$$6 \cdot o \cdot (n+1) = O(o \cdot n)$$

For the calculation of the sequence to the representative difference vector, every sequence was lapsed once together with the representative, so

$$2 \cdot o \cdot n = O(o \cdot n)$$

operations were needed.

Finally, the distance between every vector to every other vector was calculated:

$$n^2 = O(n^2)$$

The segment wise rating of the clustering quality of three tools for eight segments from *Influenza B Virus* (IBV) with 500 sequences took less than five minutes.

3.1.3 Implementation of step and window size

Investigation of the possibility to use window and step sizes to reduce calculation time in the distance calculation was carried out by rating the same cluster from CD-HIT, usearch and VeGETA repeatedly with different sizes. In detail, the penalty of the nucleotides in the window was summed up to reduce the vector size and then iteratively shifted by the step size. The distance function in the R script calculated the distance between every vector in the cluster to each other, therefore, a reduction of the calculation time by shrinking the vector size was expected. By coloring of the resulting mean of euclidean distances (MEDs) based on the used step and window sizes, the clustering quality of the tools with different settings could be observed (Fig. 3.2). Changing the step and window sizes for calculation on the same cluster resulted only in a change of the resulting MED magnitude, but did not change the quality the slightest, since the quality difference from tool to tool was preserved. While this result reflected the expected robustness of the rating script, the change in window and step size unexpectedly did not reduce the duration of the calculation. Instead, the calculation time was increased slightly due to the double calculation of penalties in the overlapping area of window and step size, since every window was calculated one by one. In conclusion, the vector calculation step was the run time determining one, instead of the vector distance calculation. By keeping this in mind, the implementation for step and window size was declared obsolete and removed.

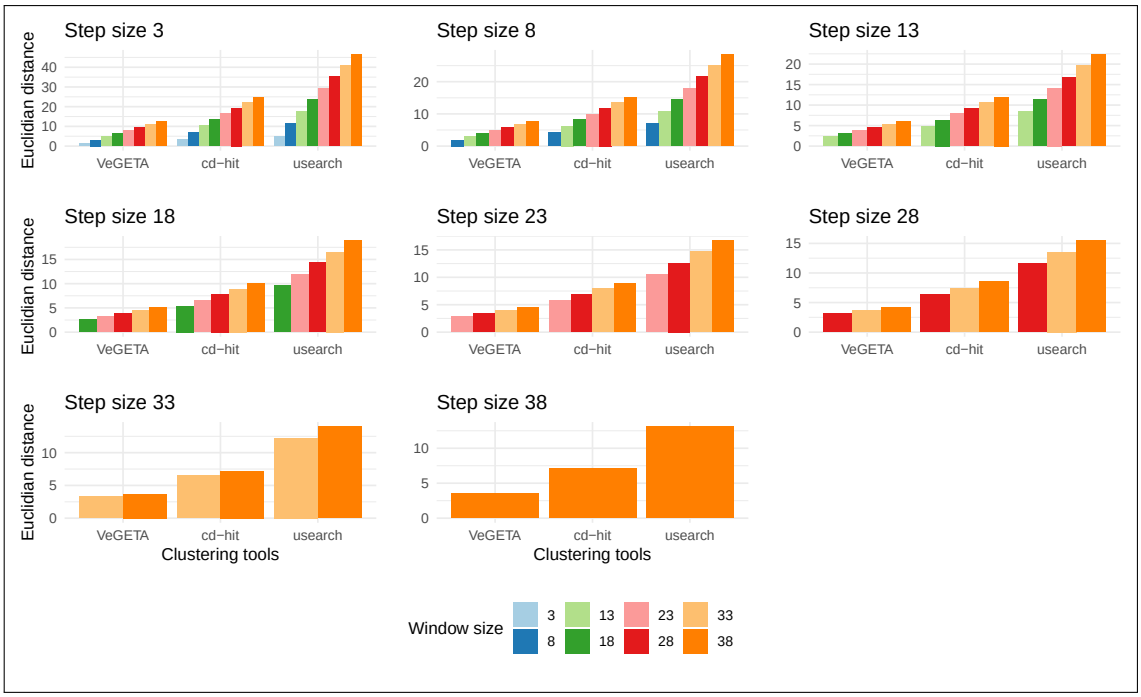


Fig. 3.2 Window and step size. Comparison of different used step and window sizes in combination with the rating pipeline. Euclidean distance was used for the vector comparison only. Step size can only be less equal to the window size to prevent skipping nucleotides in the rating process. Clustering quality of the used tools can be compared in the same window and step sizes.

3.1.4 Comparison of distance functions

For the best possible cluster rating, the results of the *rating.sh* were calculated with euclidean and with manhattan distance function. The best clustering tool was chosen by having the least distance (MED with euclidean function and mean of manhattan distance (MMD) with manhattan function). While the comparison of the used functions in the results of the rating for all the segments of IBV (Fig. 3.3 and Fig. 3.4) showed mostly similar behavior, with a change in magnitude, the results for segment three and four differed (exact distance values in Tab. A.1). This was explained by the calculation of the distance function and the resulting stronger impact of the higher distance of single sequence clusters in the MMD. While single sequence clusters should increase the overall distance to impair the tool in the ranking, the increasing of MMD by single sequence clusters was disproportionate. For the use of MMD as distance, the weighting of single sequence clusters should be reconsidered. While the using of MMD may overestimate the penalty of single sequence clusters, VeGETA still seemed to have the overall lowest distance in both methods, with exception of segment three and four with with the MMD.

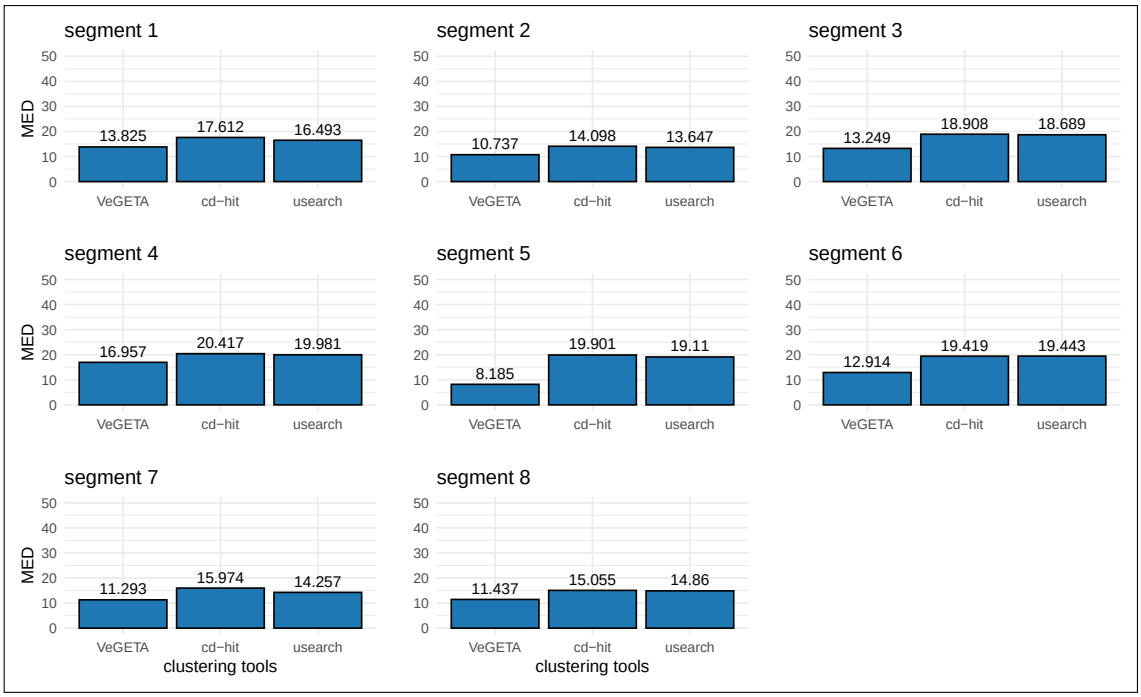


Fig. 3.3 Clustering quality of all segments measured with MED. For all segments of IBV the rating pipeline with MED as measurement was used to rate the quality of the clusters generated by CD-HIT, usearch and VeGETA. The distances are weighted by the cluster sizes generated by the tools.

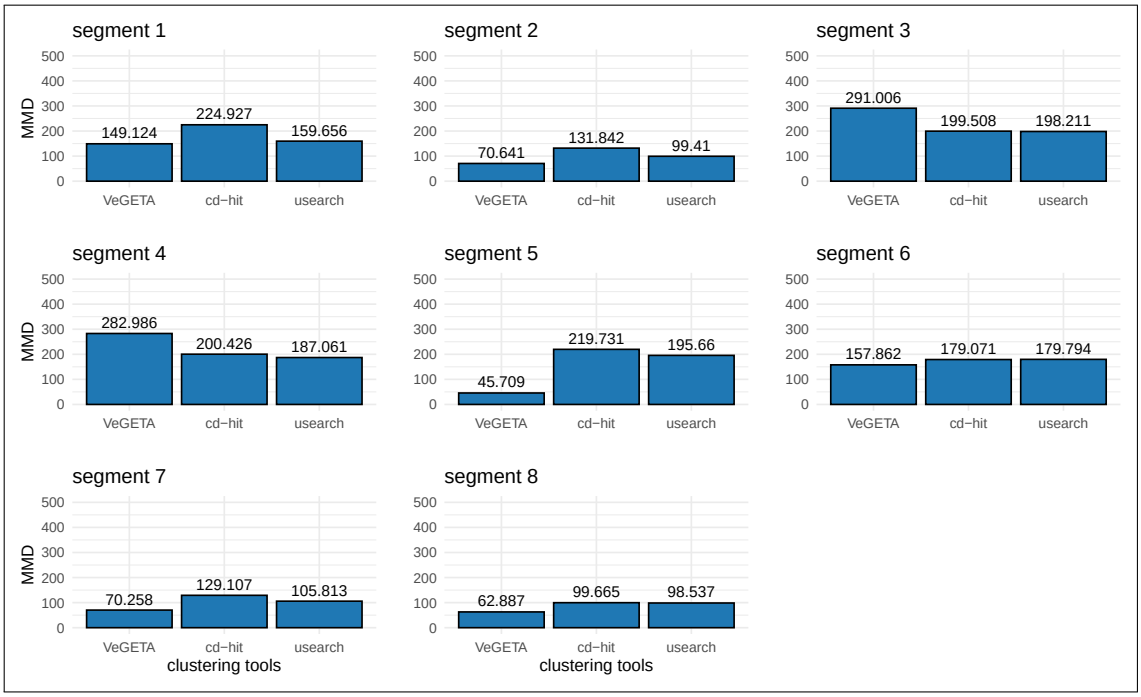


Fig. 3.4 Clustering quality of all segments measured with MMD. For all segments of IBV the rating pipeline with MMD as measurement was used to rate the quality of the clusters generated by CD-HIT, usearch and VeGETA. The distances are weighted by the cluster sizes generated by the tools.

3.2 CD-HIT and usearch identity threshold comparison

Questioning the robustness of the rating related to the CD-HIT and usearch identity threshold settings, rating of segment five was repeated multiple times with different settings (Fig. 3.5 and Fig. 3.6). While the clustering quality of both tools increased slightly with the use of the identity threshold 0.98, noticeable by a decrease of MED and MMD, there were no differences related to the ranking. With higher setting of 0.99, the number of single sequence clusters increased leading to an proportional increase of MED and MMD. By changing the threshold to 1.00, the clusters were mostly single sequence clusters leading to MED higher than 50 and MMD higher than 500. Repetition of comparison for the other segments may be necessary to find the optimum value for the identity threshold for usearch and CD-HIT yet the quality of VeGETA seems to be superior to the optimal CD-HIT and usearch settings.

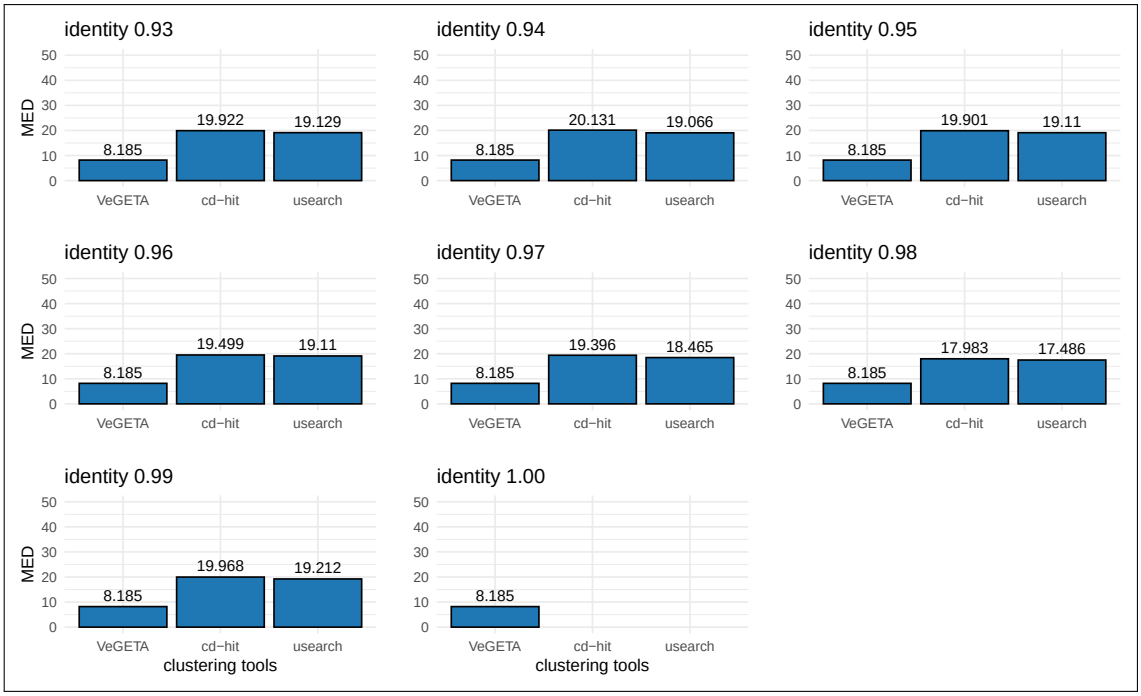


Fig. 3.5 Clustering quality of different thresholds measured with MED. For segment five of IBV and different sequence identity thresholds of CD-HIT and usearch, the rating pipeline with MED as measurement was used to rate the quality of the clusters generated by CD-HIT, usearch and VeGETA. The distances are weighted by the cluster sizes generated by the tools.

3.3 Clustering of *Influenza B Virus* segment 8

After elaboration of a reliable approach to rank clustering tools, as well as rate their clustering quality, the used tool VeGETA was compared to sequence identity based tools

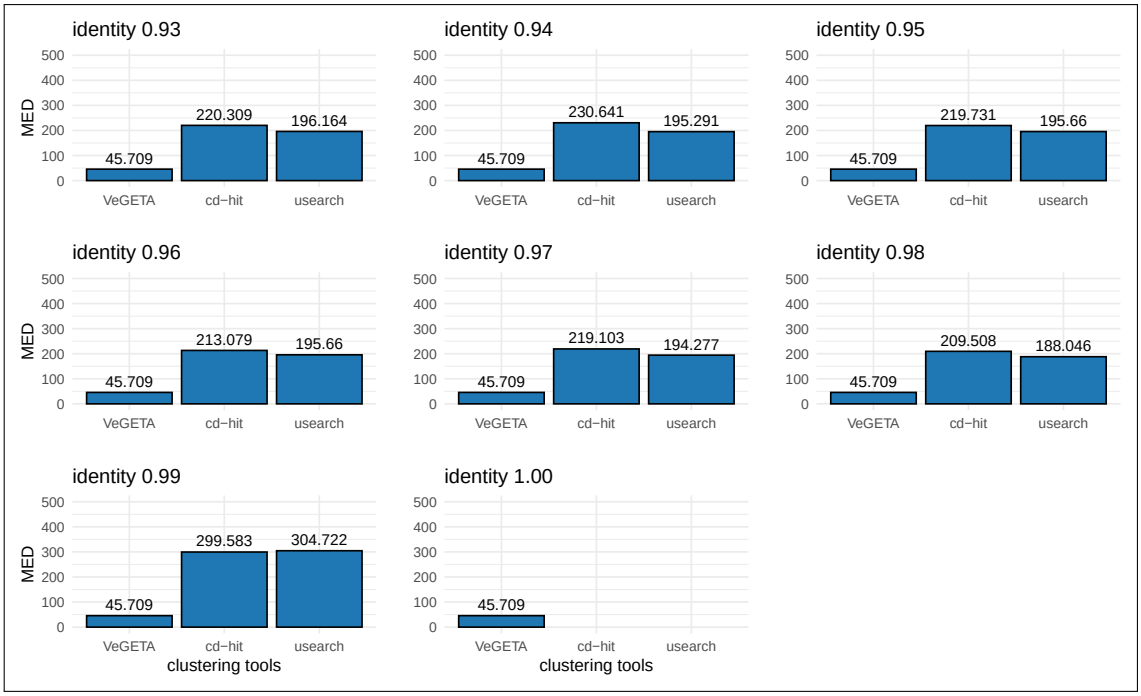


Fig. 3.6 Clustering quality of different thresholds measured with MMD. For segment five of IBV and different sequence identity thresholds of CD-HIT and usearch, the rating pipeline with MMD as measurement was used to rate the quality of the clusters generated by CD-HIT, usearch and VeGETA. The distances are weighted by the cluster sizes generated by the tools.

tool	cluster	MED	MMD	cluster size m
CD-HIT	0	15.056	99.688	488
CD-HIT	1	15.036	98.727	12
usearch	0	14.886	98.557	432
usearch	1	14.702	98.413	68
VeGETA	0	9.883	41.719	18
VeGETA	1	10.415	49.856	18
VeGETA	2	8.100	37.806	34
VeGETA	3	8.076	35.103	70
VeGETA	4	6.445	28.145	11
VeGETA	5	10.861	58.606	126
VeGETA	6	12.362	65.820	25
VeGETA	7	15.873	102.860	106
VeGETA	8	5.663	14.156	10
VeGETA	9	12.628	69.896	73

Tab. 3.1 MED of IBV segment eight clusters. The MED and MMD for all the clusters generated by the used tools with their related cluster size.

usearch and CD-HIT. For a quick overview, a radial cladogram of the used sequences was build and colored based on the clustering of VeGETA (Fig. 3.7) (Lamkiewicz 2020). By focusing on the marked windows in Fig. 3.8, the high number of clusters calculated by VeGETA became visible. Comparison with the same windows for the radial cladograms

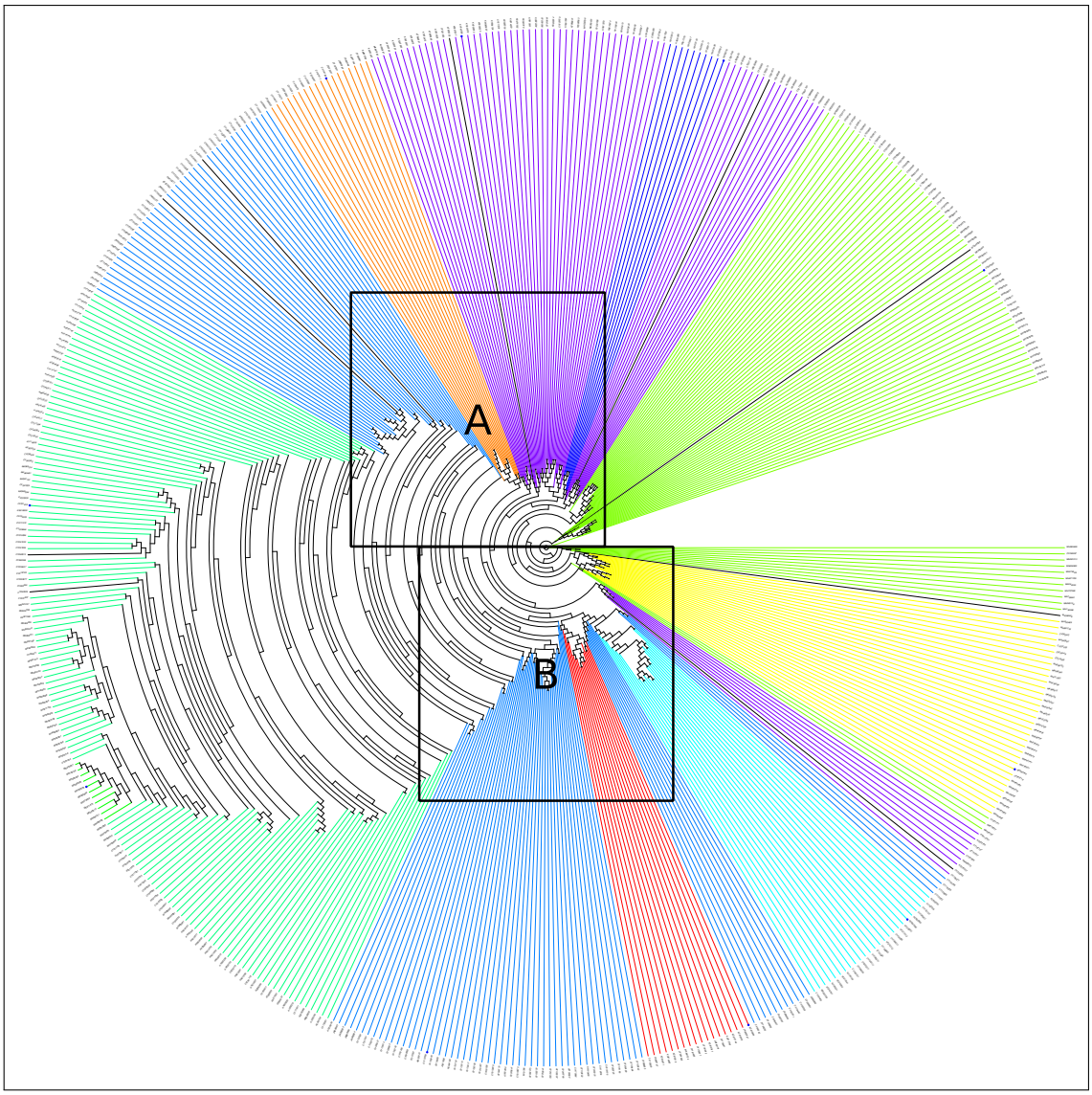
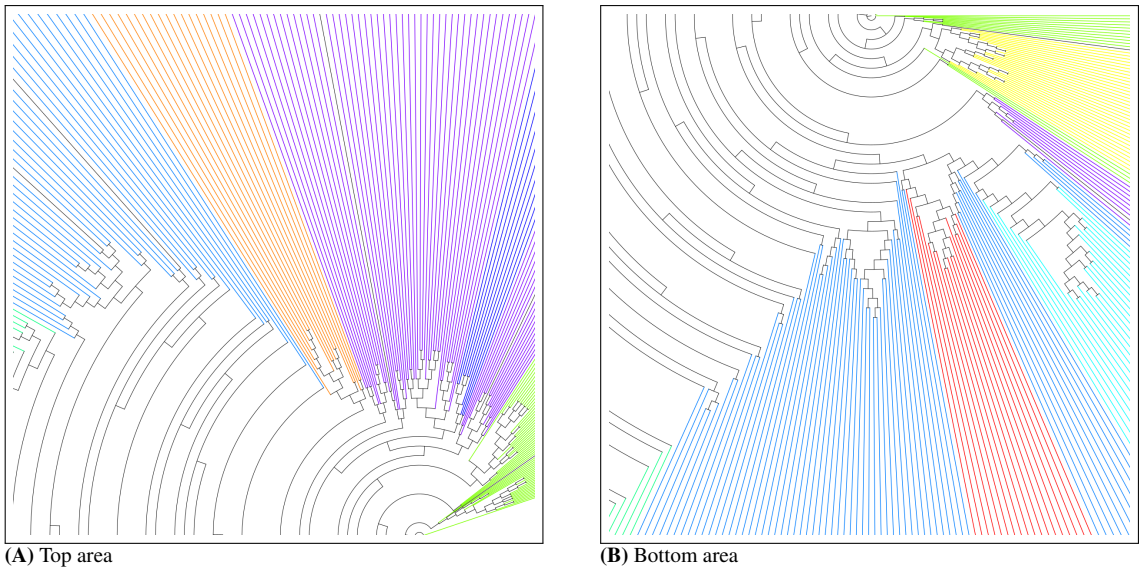


Fig. 3.7 Radial cladogram representing VeGETA clustering. The evolutionary tree of the used segment eight sequences of IBV, created by RAxML and colored related to the clusters created by VeGETA. For better comparison the areas A and B were cropped and enlarged in Fig. 3.8.

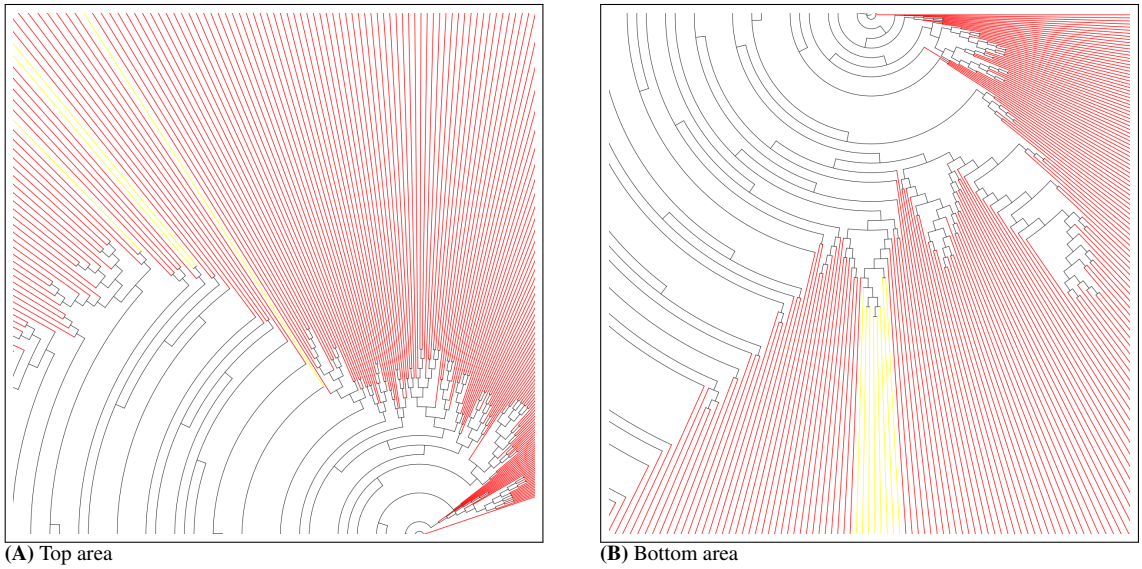
of CD-HIT Fig. 3.9 and usearch Fig. 3.10, indicated these big differences in the clustering algorithms compared to VeGETA. This was leveraged by Tab. 3.1. While usearch and CD-HIT only calculated two clusters each, one big cluster of 488 sequences in CD-HIT and 432 sequences in usearch as well as a small one with 12 in CD-HIT and 68 in usearch, VeGETA split the sequences in 10 clusters.

Precise inspection of the MED per cluster in Tab. 3.1 showed smaller distances in nine of ten clusters, calculated by VeGETA, than in the two clusters from usearch and CD-HIT. The summary of these results in Tab. 3.1 indicated the same conclusion. While the cluster with the biggest mean of MED in VeGETA with 15.873 was only slightly bigger than the clusters with the biggest mean of MED in CD-HIT (15.056) and usearch (14.886),



(A) Top area **(B)** Bottom area

Fig. 3.8 Focus on VeGETA colored cladogram Fig. 3.7. Enlargement of area A and B of Fig. 3.7 for better comparison of the clustering behavior of VeGETA. The coloring resemble the created clusters.

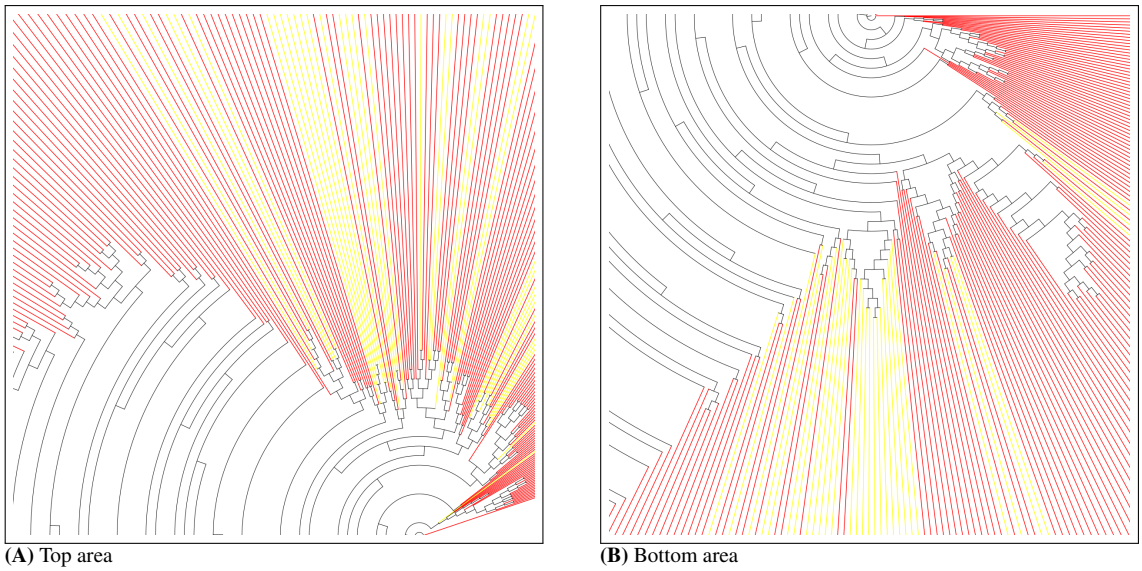


(A) Top area **(B)** Bottom area

Fig. 3.9 Focus on CD-HIT colored cladogram Fig. A.1. Enlargement of area A and B of Fig. A.1 for better comparison of the clustering behavior of CD-HIT. The coloring resemble the created clusters.

there were even plenty of better ones in VeGETA. This concluded to the fact, that even with a sequence based rating approach, like the one proposed in this project, the new tool VeGETA seemed to create more compact clusters. This assumption was supported by the fact, that VeGETA had the smallest MED values, in the visualization of the MED for all segments of IBV in Fig. 3.3, stating the compactness of the created clusters.

This result was still obtained by using a sequence identity based rating approach on different clustering methods. While even changing the identity threshold of CD-HIT and usearch did not seem to change the results in any way, it might still be necessary to validate these



(A) Top area **(B)** Bottom area

Fig. 3.10 Focus on usearch colored cladogram Fig. A.2. Enlargement of area A and B of Fig. A.1 for better comparison of the clustering behavior of usearch. The coloring resemble the created clusters.

result with a second cluster rating approach based on evolutionary distance. Repeating the pipeline with an optimal identity threshold for usearch and CD-HIT may still be important for more confident ranking. Nevertheless, the clustering of VeGETA was distinguished as most compact and best fitting for IBV clustering by this version of cluster quality rating pipeline.

3.4 Secondary structures of *Influenza B Virus* segment 8

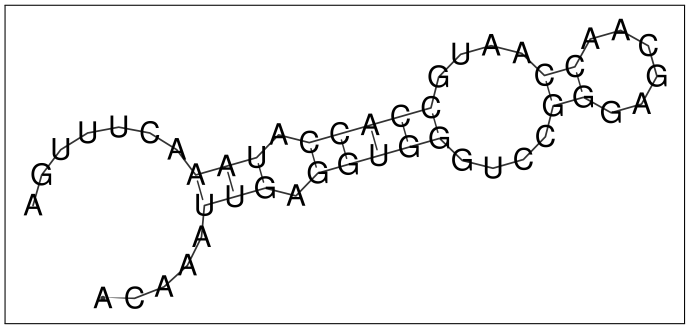


Fig. 3.11 Hairpin structure 5'-splice site. Recreation of the hairpin structure in segment eight of IBV, proposed by Dela-Moss et al. (2014) with a MSA from the segments representatives, found by VeGETA.

The overall secondary structure calculated by VeGETA from the subset of IBVs segment eight and visualized by RNAplot showed many structure elements (areas colored in purple, yellow, green and blue) (Fig. 3.12). While most structures could not be validated, literature claimed a hairpin structure in the 5'-splice site of IBV segment eight with matching

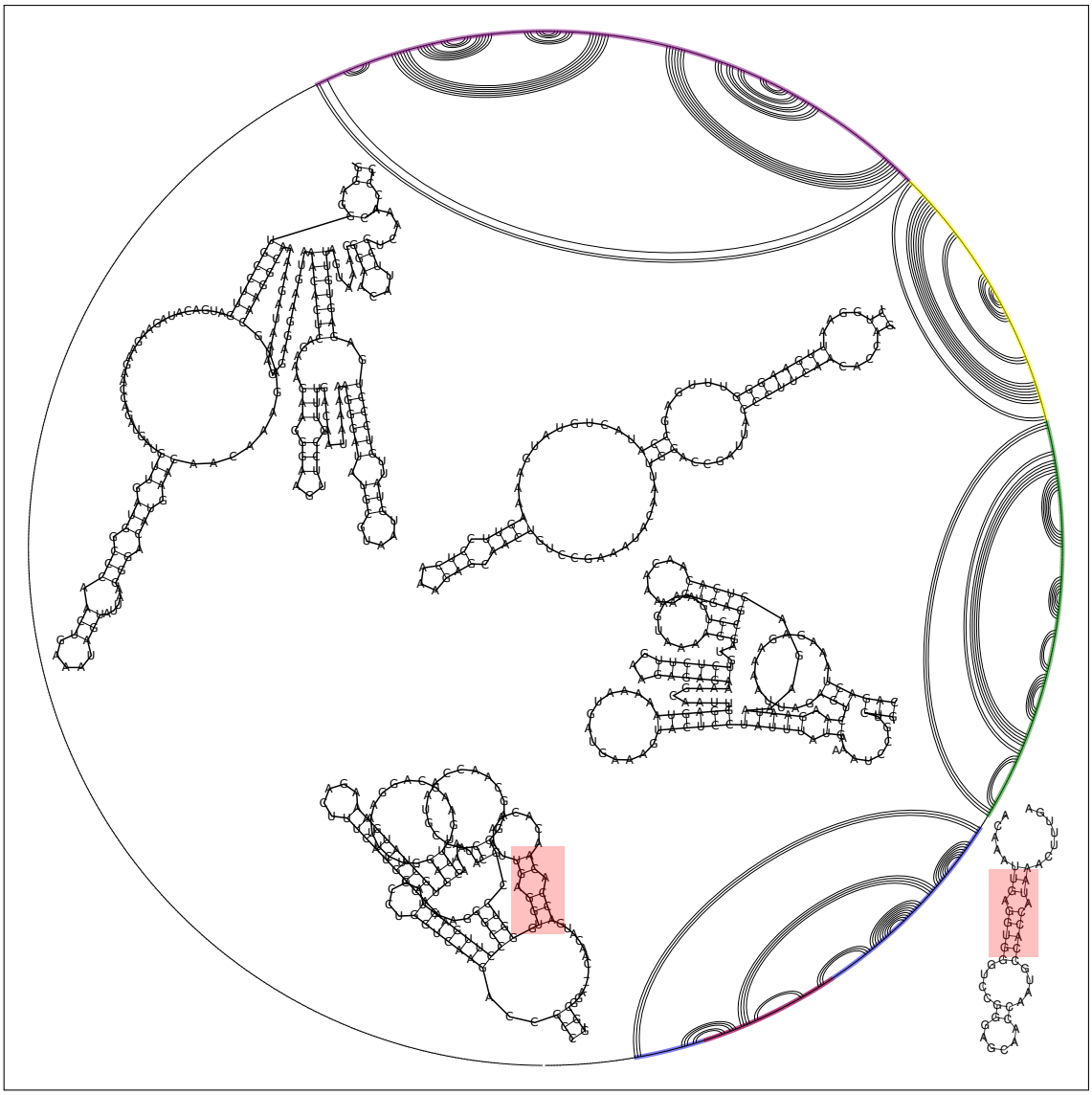


Fig. 3.12 Secondary structure of IBV segment eight calculated by VeGETA. All the secondary structure areas of the segment eight sequences of IBV found by VeGETA, with their related folded structures for better visualization.

structure area found by VeGETA (Dela-Moss et al. 2014). The position of the proposed 5'-splice site structure was marked red inside the blue colored secondary structure region Fig. 3.12. While the structure was only partly found by VeGETA in the first place, by extraction of the structure building part from a MSA build on the representatives and subsequent folding by RNAalifold, the full structure was reproduced (similar elements of VeGETA and RNAalifold highlighted with light-red square) Fig. 3.11.

The folded structure from RNAalifold had a energy of -11.6 kcal $\cdot$ mol^{-1} instead of -12.5 kcal $\cdot$ mol^{-1}, as proposed in Dela-Moss et al. (2014). Possible reasons for only finding a matching area instead of the full proposed structure, were differences in the used subsets, leading to differences in the folding energy which may led VeGETA to skip parts of this

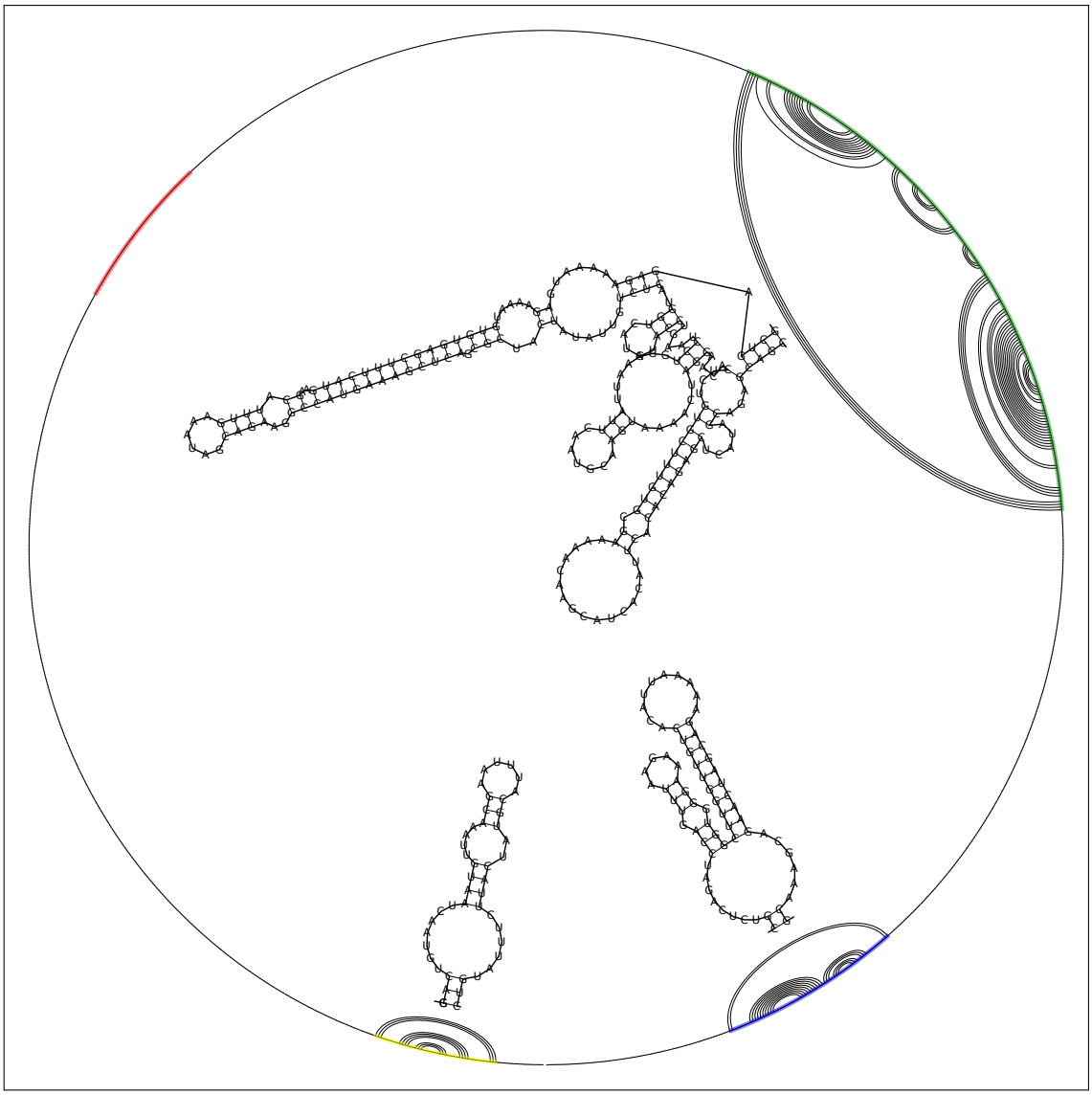


Fig. 3.13 Secondary structure of IBV segment seven calculated by VeGETA. All the secondary structure areas of the segment seven sequences of IBV found by VeGETA, with their related folded structures for better visualization.

structure for more likely structures in the area (Fig. 3.12). Time may be an important factor too, since the paper was published in 2014, for this project all available segment eight sequences for IBV as of 13th june, 2019 from the Influenza Research Database (IRD) were used. Possible new SNPs in this segment may have developed in the last five years and contradict the assumed folding of the conserved region, besides the sequences originate from different databases in the first place.

Furthermore, RNAz was used in the paper, to predict thermodynamic stable and conserved noncoding areas, followed by RNAalifold for folding and predicting the secondary structures (Dela-Moss et al. 2014; Lorenz et al. 2011; Washietl and L. Hofacker 2007). VeGETA on the other hand, use calculated shannon entropies, to predict possible areas

containing secondary structures, followed by LocARNA to build alignments on these areas (Lamkiewicz 2020; Will et al. 2012). The base pairing probability of these alignments are calculated by RNALalifold (Lorenz et al. 2011). The final structure is then predicted by arranging the base pairs based on linear optimization by integer linear program (Lamkiewicz 2020). Different approaches were used to predict the secondary structures. Since many secondary structures were found by VeGETA, an overall validation of the existence of secondary structures in the coding region of segment eight was done by Priore et al. (2013a).

In conclusion, best validation of the structures found by VeGETA can be done by analyzing the messenger RNA (mRNA) of IBV segments by *in vivo* analysis using dimethyl sulfat (DMS) probing, to reflect the natural folding of the mRNA (Simon et al. 2019). By comparing *in silico* prediction methods to each other, it is necessary to keep the principle of these methods in mind. Various results can arise by using different tools or pipelines and all results not necessarily reflect the real *in vivo* structure and merely are predictions. However, probing approaches were, to my best knowledge and until the present day, only occasionally done for IBV segments as *in vitro* by Powell et al. (2008) and more frequently done for *Influenza A Virus* (IAV) (Simon et al. 2019). Furthermore, structures found in segment seven in the matrix protein 1 (M1) ORF termination region of IBV mentioned, Powell et al. (2008) using RNA structure probing were indeed not found in VeGETAs prediction, but are also based on *in vitro* probing (Fig. 3.13).

4 Conclusion

As of today, VeGETA is still in version v0.3 (alpha) and, therefore, not in the final release state. Using VeGETA in the proposed pipeline, without subsetting the data to reduce the necessary computation power, may result in ram overflow errors at the moment, especially when using the prediction of secondary structures function of VeGETA. However, there are improvements made every day and the overall clustering quality of VeGETA seems to be promising, as described in the project and visualized in Fig. 3.3 for *Influenza B Virus* (IBV). Therefore, it seems to be useful to repeat the rating with full datasets of the IBV segments using VeGETAs clustering alone or as described in the pipeline following by the secondary structure prediction at some point in the future and compare the results again.

The rating process used in this project was created to enable comparisons based on sequence difference with translation to aminoacids (AAs) in mind. But as already mentioned, for the final validation of the results, it may be necessary to repeat the rating with other concepts, like using the evolutionary distance, instead of the nucleotide to nucleotide comparisons and compare these rating approaches for even results. Furthermore, future use of better weighting approaches related to the single sequence clusters of the rating pipeline may be useful.

For better validation of predicted secondary structures, it may be useful to repeat the pipeline with *Influenza A Virus* (IAV) segments and a later version of VeGETA. Structures for segments of IAV generated by *in vivo* probing (Simon et al. 2019) and generated by *in silico* prediction were published already (Michalak et al. 2019; Jiang et al. 2014; Priore et al. 2013b) and can be used for comparison.

The use of fully assembled pseudogenomes of the Influenza species can be another approach to search for secondary structures between the segments instead of inside them. At the moment, VeGETA is only able to calculate structures on the positive strand but this may change in near future. On top of that, the assembly of all combinations of IAV segments by iteration is nearly a computationally impossible task. To assemble only the proven and existing combinations in a reasonable amount of time can still be accomplished, by using a pipeline which utilizes the advantages of databases. Using joins, the segments in the database can be assembled to build the existing genome combinations and using VeGETA

on this resulting FASTA file. Even with version v0.3 (alpha) and the positive strands only, many possible structures were found in the existing pseudogenomes of *Influenza C Virus* (ICV) assembled that way (Fig. 4.1).

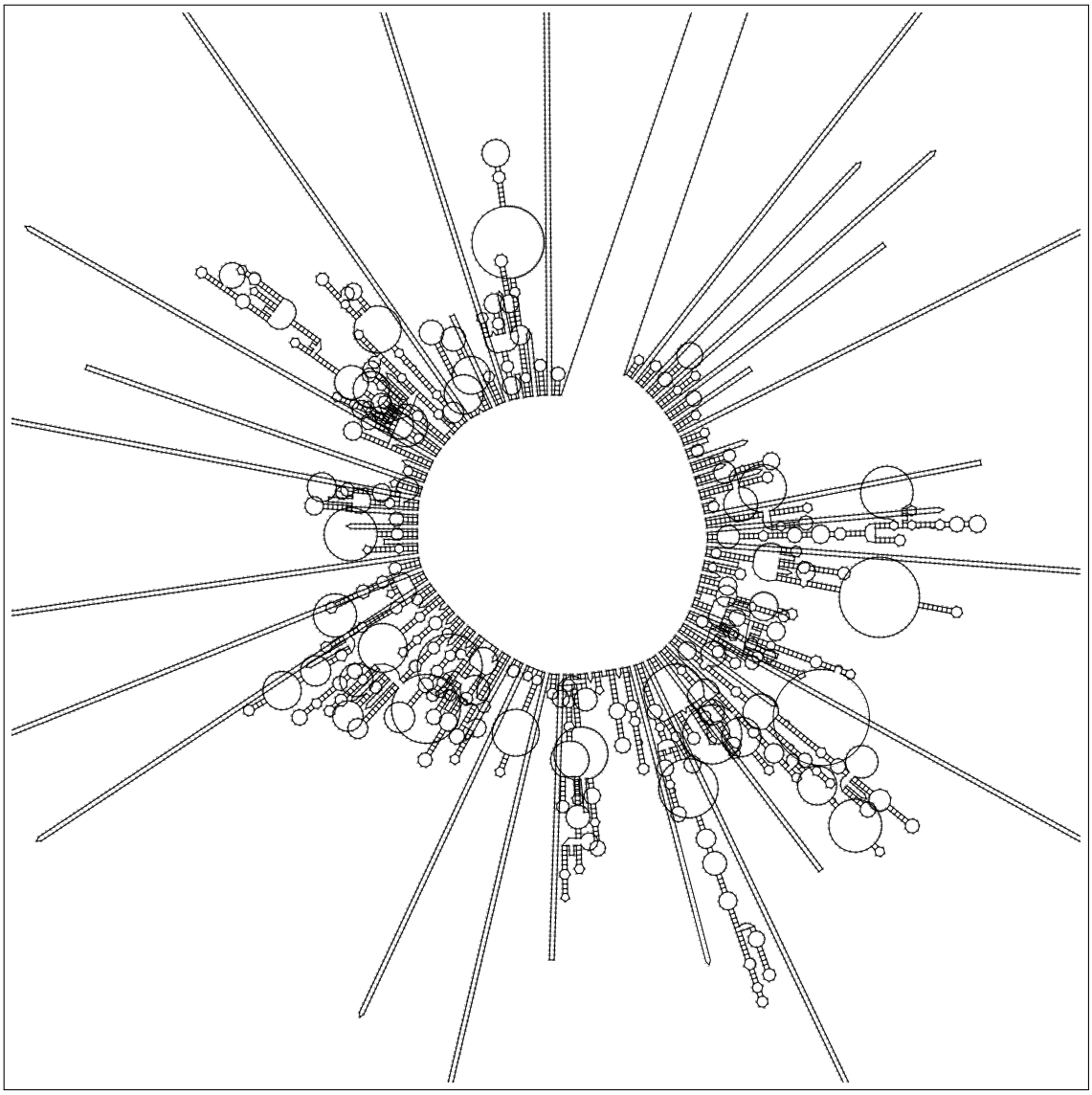


Fig. 4.1 Structures of the pseudogenomes of ICV. Secondary structures found by running VeGETA on the existing sequence combination (pseudogenomes) of ICV. Most secondary structures were still predicted inside the segments instead of between, but this correspond with the only possible use of positive sequences by VeGETA at the moment.

References

Altschul, SF (2008). "Substitution Matrices". In: *eLS*. Chichester, UK: John Wiley & Sons, Ltd, a0005265.pub2. DOI: 10.1002/9780470015902.a0005265.pub2.

Anaconda Software Distribution (2020). *Anaconda*. URL: https://anaconda.com.

Campello, RJGB, Moulavi, D, and Sander, J (2013). "Density-Based Clustering Based on Hierarchical Density Estimates". In: *Advances in Knowledge Discovery and Data Mining*. Vol. 7819. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 160–172. DOI: 10.1007/978-3-642-37456-2_14.

Cann, A (2005). *Principles of molecular virology*. 4th ed. Amsterdam ; Boston: Elsevier Academic Press. 315 pp.

Clamp, M, Cuff, J, Searle, SM, and Barton, GJ (2004). "The Jalview Java alignment editor". In: *Bioinformatics* 20.3, pp. 426–427. DOI: 10.1093/bioinformatics/btg430.

Dela-Moss, LI, Moss, WN, and Turner, DH (2014). "Identification of conserved RNA secondary structures at influenza B and C splice sites reveals similarities and differences between influenza A, B, and C." In: *BMC Research Notes* 7, 22/1–22/12, 12 pp. DOI: 10.1186/1756-0500-7-22.

Duffy, S (2018). "Why are RNA virus mutation rates so damn high?" In: *PLOS Biology* 16.8, e3000003. DOI: 10.1371/journal.pbio.3000003.

Edgar, RC (2010). "Search and clustering orders of magnitude faster than BLAST". In: *Bioinformatics* 26.19, pp. 2460–2461. DOI: 10.1093/bioinformatics/btq461.

Eisfeld, AJ, Neumann, G, and Kawaoka, Y (Jan. 2015). "At the centre: influenza A virus ribonucleoproteins". In: *Nature Reviews Microbiology* 13.1, pp. 28–41. DOI: 10.1038/nrmicro3367.

Fox, B (1989). *GNU Bash*. URL: https://www.gnu.org/software/bash/.

Handl, J, Knowles, J, and Kell, DB (Aug. 1, 2005). "Computational cluster validation in post-genomic data analysis". In: *Bioinformatics* 21.15, pp. 3201–3212. DOI: 10.1093/bioinformatics/bti517.

Jiang, T, Kennedy, SD, Moss, WN, Kierzek, E, and Turner, DH (2014). "Secondary Structure of a Conserved Domain in an Intron of Influenza A M1 mRNA." In: *Biochemistry* 53, pp. 5236–5248. DOI: 10.1021/bi500611j.

Jourdain, E, Gunnarsson, G, Wahlgren, J, Latorre-Margalef, N, Bröjer, C, Sahlin, S, Svensson, L, Waldenström, J, Lundkvist, Å, and Olsen, B (2010). "Influenza Virus in

a Natural Host, the Mallard: Experimental Infection Data". In: *PLoS ONE* 5.1, e8935. DOI: 10.1371/journal.pone.0008935.

Julkunen, I, Melén, K, Nyqvist, M, Pirhonen, J, Sareneva, T, and Matikainen, S (2000). "Inflammatory responses in influenza A virus infection". In: *Vaccine* 19, S32–S37. DOI: 10.1016/S0264-410X(00)00275-9.

Junier, T and Zdobnov, EM (2010). "The Newick utilities: high-throughput phylogenetic tree processing in the UNIX shell". In: *Bioinformatics* 26.13, pp. 1669–1670. DOI: 10.1093/bioinformatics/btq243.

Katoh, K (2002). "MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform". In: *Nucleic Acids Research* 30.14, pp. 3059–3066. DOI: 10.1093/nar/gkf436.

Kieft, JS (2008). "Viral IRES RNA structures and ribosome interactions". In: *Trends in Biochemical Sciences* 33.6, pp. 274–283. DOI: 10.1016/j.tibs.2008.04.007.

Kuroda, M, Katano, H, Nakajima, N, Tobiume, M, Ainai, A, Sekizuka, T, Hasegawa, H, Tashiro, M, Sasaki, Y, Arakawa, Y, Hata, S, Watanabe, M, and Sata, T (Apr. 23, 2010). "Characterization of Quasispecies of Pandemic 2009 Influenza A Virus (A/H1N1/2009) by De Novo Sequencing Using a Next-Generation DNA Sequencer". In: *PLoS ONE* 5.4, e10256. DOI: 10.1371/journal.pone.0010256.

Lamkiewicz, K (2020). *VeGETA - Viral GEnome sTructure Alignments*. URL: https://github.com/klamkiew/vegeta.

Li, W and Godzik, A (2006). "Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences". In: *Bioinformatics* 22.13, pp. 1658–1659. DOI: 10.1093/bioinformatics/btl158.

Lorenz, R, Bernhart, SH, Höner zu Siederdissen, C, Tafer, H, Flamm, C, Stadler, PF, and Hofacker, IL (2011). "ViennaRNA Package 2.0". In: *Algorithms for Molecular Biology* 6.1, p. 26. DOI: 10.1186/1748-7188-6-26.

Macqueen, J (1967). "Some methods for classification and analysis of multivariate observations". In: pp. 281–297.

Matsumoto, Y (1995). *Ruby*. URL: https://www.ruby-lang.org/en/.

Michalak, P, Soszynska-Jozwiak, M, Biala, E, Kesy, J, Szutkowska, B, Lenartowicz, E, Kierzek, R, Kierzek, E, and Moss, WN (2019). "Secondary structure of the segment 5 genomic RNA of influenza A virus and its application for designing antisense oligonucleotides". In: *Scientific reports* 9, p. 3801.

Moss, WN, Priore, SF, and Turner, DH (June 1, 2011). "Identification of potential conserved RNA secondary structure throughout influenza A coding regions". In: *RNA* 17.6, pp. 991–1011. DOI: 10.1261/rna.2619511.

Nelson, MI, Viboud, C, Simonsen, L, Bennett, RT, Griesemer, SB, St. George, K, Taylor, J, Spiro, DJ, Sengamalay, NA, Ghedin, E, Taubenberger, JK, and Holmes, EC (Feb. 29, 2008). "Multiple Reassortment Events in the Evolutionary History of H1N1 Influenza

A Virus Since 1918". In: *PLoS Pathogens* 4.2, e1000012. DOI: 10.1371/journal.ppat.1000012.

Powell, ML, Napthine, S, Jackson, RJ, Brierley, I, and Brown, TDK (Sept. 16, 2008). "Characterization of the termination-reinitiation strategy employed in the expression of influenza B virus BM2 protein". In: *RNA* 14.11, pp. 2394–2406. DOI: 10.1261/rna.1231008.

Priore, SF, Moss, WN, and Turner, DH (2013a). "Influenza B virus has global ordered RNA structure in (+) and (-) strands but relatively less stable predicted RNA folding free energy than allowed by the encoded protein sequence". In: *BMC Research Notes* 6.1, p. 330. DOI: 10.1186/1756-0500-6-330.

Priore, SF, Kierzek, E, Kierzek, R, Baman, JR, Moss, WN, Dela-Moss, LI, and Turner, DH (2013b). "Secondary structure of a conserved domain in the intron of influenza A NS1 mRNA." In: *PLoS One* 8, e70615. DOI: 10.1371/journal.pone.0070615.

R Development Core Team (2006). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: http://www.R-project.org.

Sharp, PM and Hahn, BH (2011). "Origins of HIV and the AIDS Pandemic". In: *Cold Spring Harbor Perspectives in Medicine* 1.1, a006841–a006841. DOI: 10.1101/cshperspect.a006841.

Simon, LM, Morandi, E, Luganini, A, Gribaudo, G, Martinez-Sobrido, L, Turner, DH, Oliviero, S, and Incarnato, D (July 26, 2019). "In vivo analysis of influenza A mRNA secondary structures identifies critical regulatory motifs". In: *Nucleic Acids Research* 47.13, pp. 7003–7017. DOI: 10.1093/nar/gkz318.

Stamatakis, A (2014). "RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies". In: *Bioinformatics* 30.9, pp. 1312–1313. DOI: 10.1093/bioinformatics/btu033.

Suarez, DL (2000). "Evolution of avian influenza viruses". In: *Veterinary Microbiology* 74.1, pp. 15–27. DOI: 10.1016/S0378-1135(00)00161-9.

Tange, O (2020). *GNU Parallel 20200722 ('Privacy Shield')*. DOI: 10.5281/zenodo.3956817.

Unknown author (1980). "A revision of the system of nomenclature for influenza viruses: a WHO memorandum". In: *Bulletin of the World Health Organization* 58.4, pp. 585–591.

Van Reeth, K (2007). "Avian and swine influenza viruses: our current understanding of the zoonotic risk". In: *Veterinary Research* 38.2, pp. 243–260. DOI: 10.1051/vetres:2006062.

Van Rossum, G and Drake Jr, FL (1995). *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam.

Wahlgren, J (Jan. 2011). "Influenza A viruses: an ecology review". en. In: *Infection Ecology & Epidemiology* 1.1, p. 6004. DOI: 10.3402/iee.v1i0.6004. (Visited on 09/30/2020).

Washietl, S and L. Hofacker, I (Sept. 2007). “Identifying Structural Noncoding RNAs Using RNAz”. In: *Current Protocols in Bioinformatics* 19.1. DOI: 10.1002/0471250953.bi1207s19.

Webster, RG (1999). “Antigenic Variation in Influenza Viruses”. In: *Origin and Evolution of Viruses*. Elsevier, pp. 377–390. DOI: 10.1016/B978-012220360-2/50015-5.

Will, S, Joshi, T, Hofacker, IL, Stadler, PF, and Backofen, R (May 1, 2012). “LocARNA-P: Accurate boundary prediction and improved detection of structural RNAs”. In: *RNA* 18.5, pp. 900–914. DOI: 10.1261/rna.029041.111.

Zhang, Y, Aevermann, BD, Anderson, TK, Burke, DF, Dauphin, G, Gu, Z, He, S, Kumar, S, Larsen, CN, Lee, AJ, Li, X, Macken, C, Mahaffey, C, Pickett, BE, Reardon, B, Smith, T, Stewart, L, Suloway, C, Sun, G, Tong, L, Vincent, AL, Walters, B, Zaremba, S, Zhao, H, Zhou, L, Zmasek, C, Klem, EB, and Scheuermann, RH (2017). “Influenza Research Database: An integrated bioinformatics resource for influenza virus research”. In: *Nucleic Acids Research* 45 (D1), pp. D466–D474. DOI: 10.1093/nar/gkw857.

A Appendix

Tool	Distance	Segment	Function
CD-HIT	17.612	1	Euclidean
usearch	16.493	1	Euclidean
VeGETA	13.825	1	Euclidean
CD-HIT	14.098	2	Euclidean
usearch	13.647	2	Euclidean
VeGETA	10.737	2	Euclidean
CD-HIT	18.908	3	Euclidean
usearch	18.689	3	Euclidean
VeGETA	13.249	3	Euclidean
CD-HIT	20.417	4	Euclidean
usearch	19.981	4	Euclidean
VeGETA	16.957	4	Euclidean
CD-HIT	19.901	5	Euclidean
usearch	19.11	5	Euclidean
VeGETA	8.185	5	Euclidean
CD-HIT	19.419	6	Euclidean
usearch	19.443	6	Euclidean
VeGETA	12.914	6	Euclidean
CD-HIT	15.974	7	Euclidean
usearch	14.257	7	Euclidean
VeGETA	11.293	7	Euclidean
CD-HIT	15.055	8	Euclidean
usearch	14.86	8	Euclidean
VeGETA	11.437	8	Euclidean
CD-HIT	224.927	1	Manhattan
usearch	159.656	1	Manhattan
VeGETA	149.124	1	Manhattan
CD-HIT	131.842	2	Manhattan

Continued on next page

Tool	Distance	Segment	Function
usearch	99.41	2	Manhattan
VeGETA	70.641	2	Manhattan
CD-HIT	199.508	3	Manhattan
usearch	198.211	3	Manhattan
VeGETA	291.006	3	Manhattan
CD-HIT	200.426	4	Manhattan
usearch	187.061	4	Manhattan
VeGETA	282.986	4	Manhattan
CD-HIT	219.731	5	Manhattan
usearch	195.66	5	Manhattan
VeGETA	45.709	5	Manhattan
CD-HIT	179.071	6	Manhattan
usearch	179.794	6	Manhattan
VeGETA	157.862	6	Manhattan
CD-HIT	129.107	7	Manhattan
usearch	105.813	7	Manhattan
VeGETA	70.258	7	Manhattan
CD-HIT	99.665	8	Manhattan
usearch	98.537	8	Manhattan
VeGETA	62.887	8	Manhattan

Tab. A.1 Weighted result from *output.R* for IBV

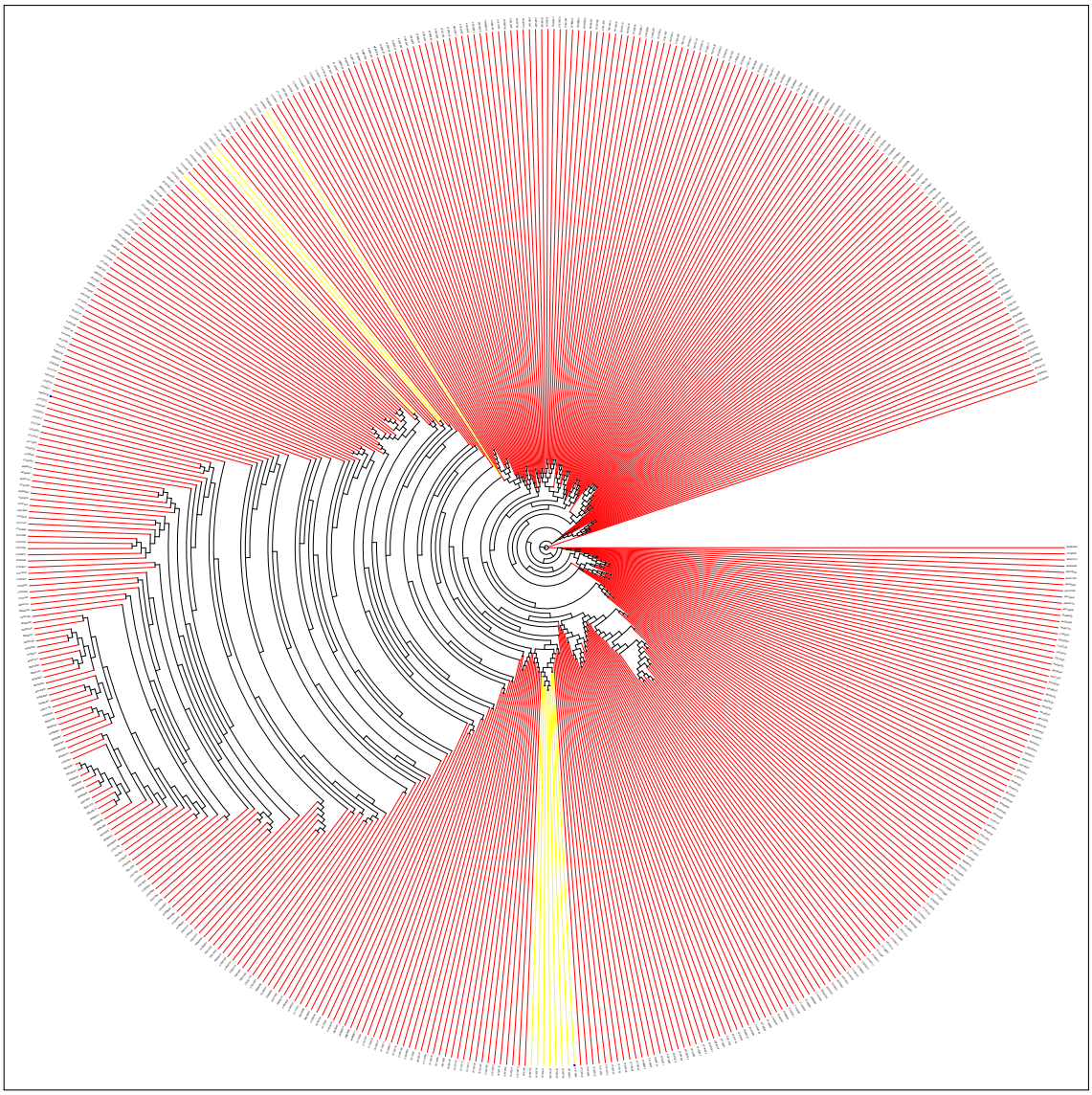


Fig. A.1 Radial cladogram representing CD-HIT clustering

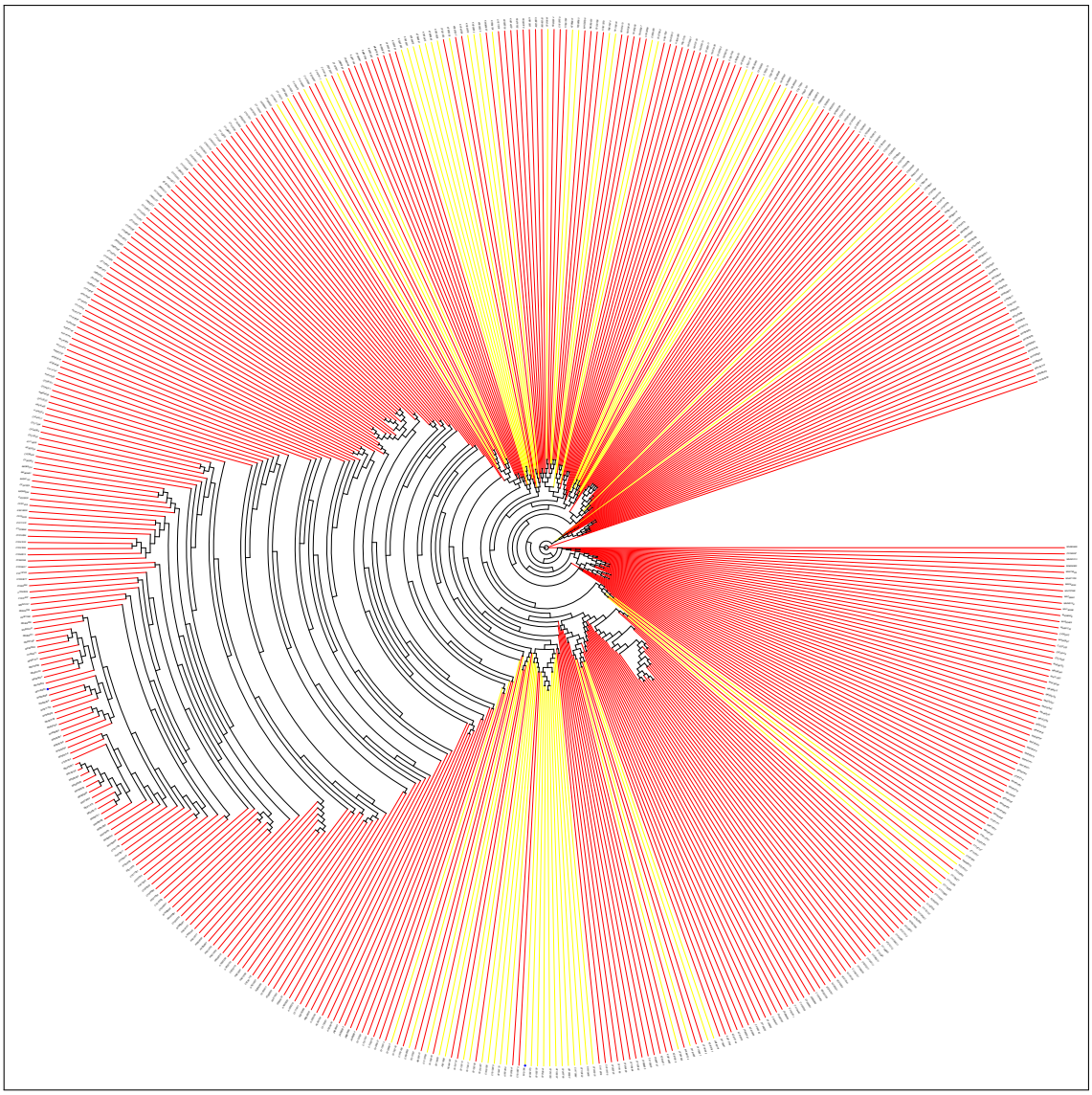


Fig. A.2 Radial cladogram representing usearch clustering