**Instructions for updating RVDB\_PY3**

1. **Prerequisites**

RVDB Python3 (RVDB\_PY3) scripts requires several third-party packages for data processing and most importantly, quality checking and clustering SARS-CoV-2 sequences. The prerequisites and their installation instructions are described as below.

1. Environment setup

We highly recommend to create an isolated Python3 environment by package managers such as Anaconda or Mamba. RVDB\_PY3 was tested under the Python3 environment create by Mamba. The Mamba instruction is available at <https://github.com/mamba-org/mamba?tab=readme-ov-file>.

1. Installation of prerequisites packages

We have provided a YAML Ain't Markup Language (YAML) file named “RVDB\_PY3.YAML” which contains the setup instruction for RVDB\_PY3 running under Mamba environment. To create a mamba environment named “rvdb\_pipeline” and install all prerequisites, please prompt:

$ mamba env create -n rvdb\_pipeline -f RVDB\_PY3.yaml

1. Environment activation

To activate “rvdb\_pipeline” environment, please prompt:

$ mamba activate rvdb\_pipeline

That’s it. You are ready to proceed to Step 1.

**1. Create folders for update**

First, create a parent directory for the update. For example, the following prompt would create a parent folder name RVDB

$mkdir RVDB

and place the folder “UPDATE\_SCRIPTS\_LOGS\_PY3” to here. This folder contains Python 3 scripts and accessories to update RVDB.

Next, create a sub-directory under the parent directory. This should be in “RVDBv$version” format, for example the following line would create the folder for version 30.0:

$cd RVDB

$mkdir RVDBv30.0

The folder structure for an RVDB update is three separate main folders for GenBank, TPA, and RefSeq sequences, in \*\_“$month”.”$year” format. For example, if the month were apr and the year 2025, the folders could be created using the following command:

$cd RVDBv30.0

$mkdir GenBank\_raw\_data\_apr.2025 && mkdir TPA\_raw\_data\_apr.2025 && mkdir RefSeq\_raw\_data\_apr.2025

The sub-folder structure for an RVDB update depends on the main folder. All main folders have “log” and “scripts” sub-folders. The main GenBank and TPA folders also have poskw\_out\_”$month”.”$year”, sizemirna\_out\_”$month”.”$year”, and negkw\_out\_”$month”.”$year sub-folders. So, for example, for the GenBank main folder you could enter the following commands:

$cd GenBank\_raw\_data\_apr.2025

$mkdir log && mkdir scripts && mkdir poskw\_out\_apr.2025 && mkdir sizemirna\_out\_ apr.2025 && mkdir negkw\_out\_ apr.2025

**2. Download raw sequences from NCBI FTP**

**RefSeq viral.** Navigate to the RefSeq main folder, log on to NCBI’s RefSeq ftp site, [ftp.ncbi.nih.gov/refseq/release/viral](ftp://ftp.ncbi.nih.gov/refseq/release/viral), and download the four files: viral.1.1.genomic.fna.gz (fasta file), viral.2.1.genomic.fna.gz (fasta file), viral.1.genomic.gbff.gz (genbank flat file), and viral.2.genomic.gbff.gz (genbank flat file). This can be done using the ftp command:

$ftp [ftp.ncbi.nih.gov](ftp://ftp.ncbi.nih.gov)

anonymous

anonymous

cd refseq

cd release

cd viral

binary

prompt

mget viral\*genomic\*gz

**RefSeq viral neighbors.** This is one of only two parts that requires manually downloading some files:

Save the downloaded file in the Refseq data folder as “refseqviral\_neighbors\_mapping.$date.nbr”, where $date is the full date in “month.day.year” format (example “refseqviral\_neighbors\_mapping.apr.23.2025.nbr”) in the Refseq main folder.

$wget <https://ftp.ncbi.nlm.nih.gov/genomes/Viruses/Viruses_RefSeq_and_neighbors_genome_data.tab> -O refseqviral\_neighbors\_mapping.apr.23.2025.nbr

Open the .nbr file in Excel using the “delimited” option with only “tab” selected (this should be the default). Resave as a .csv (example “refseqviral\_neighbors\_mapping.apr.23.2025.csv”). You can delete the original .nbr file after completing this step.

**Phage.** There is an list of phage keywords that are used to identify and remove phage sequences. It should be saved in the RefSeq main folder, log sub-folder, as “phage\_kws.txt”. It contains the following search strings:

‘ phage’

‘corticovir’

‘cystovir’

‘fusellovir’

‘ inovir’

‘plectrovir’

‘levivir’

‘lipothrixvir’

‘microvir

‘myovir

‘plasmavir’

‘podovir’

‘rudivir’

‘siphovir’

‘tectivir’

**GenBank.** Navigate to the Genbank main folder**, l**og on to NCBI’s Genbank ftp site, <ftp://ftp.ncbi.nih.gov/genbank> , and download gb flat files from the following 10 divisions: ENV, HTC, INV, MAM, PLN, PRI, ROD, VRL, VRT. This can be done using the following ftp command:

$ftp [ftp.ncbi.nih.gov](ftp://ftp.ncbi.nih.gov)

anonymous

anonymous

cd genbank

binary

prompt

mget gbenv\*seq.gz gbhtc\*.seq.gz gbinv\*.seq.gz gbmam\*.seq.gz gbpln\*.seq.gz gbpri\*.seq.gz gbrod\*.seq.gz gbvrl\*.seq.gz gbvrt\*.seq.gz

Also, the official release notes must be downloaded from the GenBank website using a web browser. While this could be done using ftp, the name of the release notes file has to be passed as a parameter later, so it’s best to directly download it and save the file name for later. Visit: [ftp://ftp.ncbi.nih.gov/genbank/](ftp://ftp.ncbi.nih.gov/genbank/release.notes/) and download ”gbrel.txt”. Save this file in “gb\_releasenotes\_v$version\_$month.$year.txt” format, for example gb\_releasenotes\_v265\_apr.2025.txt.

**TPA.** Navigate to the TPA main folder**, l**og on to NCBI’s TPA ftp site, [ftp.ncbi.nih.gov/tpa/release](ftp://ftp.ncbi.nih.gov/tpa/release) , and download TPA sequence files tpa\_cu.fsa\_nt.gz and con\_tpa\_cu.fsa\_nt.gz. Note that there is no meta-data and therefore not .gbff format files for TPA sequences. The download can be done using the following ftp command:

$ftp [ftp.ncbi.nih.gov](ftp://ftp.ncbi.nih.gov)

anonymous

anonymous

cd tpa

cd release

binary

prompt

mget \*tpa\*nt.gz

**3. Running the main pipeline – RefSeq and GenBank.**

The main pipeline performs the core series of operations on the downloaded RefSeq, GenBank, and TPA files. In order, this includes unzipping RefSeq viral, removing phage, pulling in viral neighbor annotation, identifying duplicates of RefSeq (original GenBank entries from which RefSeq entries were created), unzipping and formatting GenBank entries, running checkpoint2 to cross-reference GenBank file contents with the official release notes, and running the positive, size/mirna, and negative screens on GenBank files.

**Main pipeline – RefSeq and GenBank - command block.** Navigate to the parent folder (“RVDB” in this example). Use the following concatenated commands (described individually beneath the command block):

$python UPDATE\_SCRIPTS\_LOGS\_PY3/parse\_raw\_refseq\_PIPE.py . apr.2025 30.0 viral.1.1.genomic.fna.gz viral.2.1.genomic.fna.gz && python UPDATE\_SCRIPTS\_LOGS\_PY3/multiple\_gzunzip\_PIPE.py . apr.2025 30.0 viral.1.genomic.gbff.gz viral.2.genomic.gbff.gz viral.genomic.gbff && python UPDATE\_SCRIPTS\_LOGS\_PY3/fileops\_PIPE.py . apr.2025 30.0 gbff 1000000 && python UPDATE\_SCRIPTS\_LOGS\_PY3/rs\_acc\_mapping\_PIPE.py . apr.2025 30.0 && python UPDATE\_SCRIPTS\_LOGS\_PY3/VDBunzip\_reformat\_gb\_to\_fasta\_PIPE.py . apr.2025 30.0 gb && python UPDATE\_SCRIPTS\_LOGS\_PY3/VDBupdate\_checkpoint2\_PIPE.py . apr.2025 30.0 gb\_releasenotes\_v265\_apr.2025.txt && python UPDATE\_SCRIPTS\_LOGS\_PY3/SEM-R\_june62018\_PIPE.py . apr.2025 30.0 poskw gb && python UPDATE\_SCRIPTS\_LOGS\_PY3/SEM-R\_june62018\_PIPE.py . apr.2025 30.0 sizemirna gb && python UPDATE\_SCRIPTS\_LOGS\_PY3/SEM-R\_june62018\_PIPE.py . apr.2025 30.0 negkw gb

**Description of commands and scripts.** These scripts called in the command block above do the following:

$python UPDATE\_SCRIPTS\_LOGS\_PY3/parse\_raw\_refseq\_PIPE.py . apr.2025 30.0 viral.1.1.genomic.fna.gz viral.2.1.genomic.fna.gz

Takes the two RefSeq viral files and outputs a eukaryotic viral fasta file formatted with two lines per entry (header and sequences), as well as a phage file (same format). “.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update. The directory for the update is, in this case, ./RVDBv30.0.

$python UPDATE\_SCRIPTS\_LOGS\_PY3/multiple\_gzunzip\_PIPE.py . apr.2025 30.0 viral.1.genomic.gbff.gz viral.2.genomic.gbff.gz viral.genomic.gbff

Combines the two GenBank flat files for refseq viral into one. “.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update.

$python UPDATE\_SCRIPTS\_LOGS\_PY3/fileops\_PIPE.py . apr.2025 30.0 gbff 1000000

Splits the combined GenBank flat file into multiple files, so that each can be read into Python. “.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “gbff” is the file type used as input, and 1000000 is the number of entries to include in each split.

$python UPDATE\_SCRIPTS\_LOGS\_PY3/rs\_acc\_mapping\_PIPE.py . apr.2025 30.0

Using the GenBank flat file metadata for RefSeq viral, finds the duplicate entries’ accessions (original entries, upon which RefSeq viral entries were based). “.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update. Also uses the RefSeq viral neighbors mapping file to complete the mapping (here, “./RVDBv30.0/RefSeq\_raw\_data.apr.2025/ refseqviral\_neighbors\_mapping.apr.23.2025.csv”; this filename is hard-coded into the script). The neighbors are saved in the file (here “./RVDBv30.0/RefSeq\_raw\_data.apr.2025/neighbor\_accs.txt”); this filename is hard-coded into the next script, which is the unzipping script. The RefSeq duplicate accessions are saved in the file “./RVDBv30.0/RefSeq\_raw\_data.apr.2025/refseq\_viral\_originalaccs.txt” ; this filename is also hard-coded as input for the unzipping script.

$python UPDATE\_SCRIPTS\_LOGS\_PY3/VDBunzip\_reformat\_gb\_to\_fasta\_PIPE.py . apr.2025 30.0 gb

Unzips the GenBank division files, labels sequences that are RefSeq viral neighbors during the unzipping. “.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update.

$python UPDATE\_SCRIPTS\_LOGS\_PY3/VDBupdate\_checkpoint2\_PIPE.py . apr.2025 30.0 gb\_releasenotes\_v265\_apr.2025.txt

“.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “gb\_releasenotes\_v265\_apr.2025.txt” is the name of the release notes file that was downloaded from the GenBank ftp site.

Runs checkpoint2, generates four output files: ./RVDB[version]/GenBank\_raw\_data\_month.year/log/[version]\_checkpt2[a,b,c,d].log”. Note, the names of unzipped files are hard-coded into the semantic screen script that is called next: SEM-R\_june62018\_PIPE.py, which is described below.

The first file (“a.log” ending) is a print-out of a running total of files / division, seqs / division, after each .seq.gz file is read. This file is time-stamped, so it’s main purpose is to show a continuous timeline of the unzipping process.

The second file (“b.log” ending) is a summary of the unzipping process, showing total #sequences for each file (basically shortened version of 2a). This is more convenient for looking at entry totals. This format is the same format as the official release notes.

The third file (“c.log” ending) is a side-by-side list of all file counts, the official release notes counts and the downloaded + unzipped counts.

The fourth file (“d.log” ending) is a side-by-side list of all file division counts, the official release notes counts and the downloaded + unzipped counts. This is a summary form of c.log, with totals by division rather than file.

$python UPDATE\_SCRIPTS\_LOGS\_PY3/SEM-R\_june62018\_PIPE.py . apr.2025 30.0 poskw gb

Runs the positive keyword screen. “.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “poskw” is the type of screen, “gb” is the source database. Generates files ending in “pscreen” as output. All files generated as output are used as input for the sizemirna screen.

$python UPDATE\_SCRIPTS\_LOGS\_PY3/SEM-R\_june62018\_PIPE.py . apr.2025 30.0 sizemirna gb

Runs the size/mirna screen. “.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “sizemirna” is the type of screen, “gb” is the source database. Two type of files are generated as output: those ending in “FLAG” and those ending in “OK”. Files ending in “OK” pass the sizemirna screen and are used as input for the negkw screen.

$python UPDATE\_SCRIPTS\_LOGS\_PY3/SEM-R\_june62018\_PIPE.py . apr.2025 30.0 negkw gb

Runs the negative keyword screen. “.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “negkw” is the type of screen, “gb” is the source database. Generated four types of files as output” those ending in “FLAG”, those ending in “OK”, those ending in “AMB”, and those ending in “VRL”. The files ending in “OK”, “AMB” (for “ambiguous”), and “VRL” (coming from the GenBank “VRL” division) can be manually reviewed (see below, section 5) to generate the U-RVDB.

**4. Running the main pipeline – TPA.**

The main pipeline consists of unzipping the TPA files and running the positive, size/mirna, and negative screens on the unzipped TPA files.

**Main pipeline – TPA – command block.** Use the following concatenated and piped commands (described individually below).

$python UPDATE\_SCRIPTS\_LOGS\_PY3/VDBunzip\_tpa\_PIPE.py . apr.2025 30.0 fsa\_nt.gz && python UPDATE\_SCRIPTS\_LOGS\_PY3/SEM-R\_june62018\_PIPE.py . apr.2025 30.0 poskw tpa && python UPDATE\_SCRIPTS\_LOGS\_PY3/SEM-R\_june62018\_PIPE.py . apr.2025 30.0 sizemirna tpa && python UPDATE\_SCRIPTS\_LOGS\_PY3/SEM-R\_june62018\_PIPE.py . apr.2025 30.0 negkw tpa

“.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update. For the positive screen, sizemirna screen, and negkw screen the output files are the same as for the GenBank pipeline, with the except that no “VRL” files are generated by the negkw screen because there is no VRL division in TPA. The files ending in “OK” and “AMB” (for “ambiguous”), can be manually reviewed (see below, section 5) to generate the U-RVDB.

**5. Manual review**

Following the running of the main pipeline, all sequences passing the SEM-R screen will be in files in the RefSeq, GenBank, and TPA directories. The script “prep\_manual\_review.py” collects all sequences that have passed all three parts of the SEM-R\_june62018\_PIPE.py screen (poskw, sizemirna, and negkw). These are sequences that are present in files with “OK” or “VRL” endings in either the GenBank or the TPA fodlers, negkw\_out sub-folders. Also, these are sequences from the “viral.genomic.eukviral.accs.txt” file in the RefSeq folder. The prep\_manual\_review.py script takes accessions of these “passing” sequences and compares them to accessions from the previous version of the U-RVDB, and finally outputs two sheets: one with headers of “missing” sequences in the update, or those that were present in the previous version but not the update, and one with headers of “new” sequences in the update, or those that are present in the update but were not present in the previous version. Both files should be manually reviewed for consistency, i.e. to make sure that the missing sequences have all gone obsolete or been modified to a new version numbers or switched to a different, non-RVDB GenBank division. Additionally, sequences in files ending in “AMB” (for ambiguous) should be manually reviewed (generally these are not included). Finally, as a result of manual review of new sequences, a list of accessions (one accession per line) that are non-viral should be made using the following syntax and stored in the main directory for the update: “RVDBv$version.removeaccs.txt”, for example (“RVDBv30.0.removeaccs.txt”). This is done manually. The prep\_manual\_review.py script is called from the command line as follows:

$python UPDATE\_SCRIPTS\_LOGS/prep\_manual\_review.py . apr.2025 30.0 ./RVDBv29.0/U-RVDBv29.0.fasta

“.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “./RVDBv29.0/U-RVDBv29.0.fasta” is the full path to the previous, unclustered version of the database (if you want, you can also compare to any other version of RVDB). The script outputs two files into the main folder of the update: “RVDBv$version.missing.csv” and “RVDBv$version.new.csv”, e.g. “RVDBv30.0.missing.csv” and “RVDBv30.0.new.csv”.

**6. Creation of raw RVDB fasta files**

In the old Python 2 RVDB production pipeline, this is the last step to generate U-RVDB. We have implemented several improvements and refinements described in the publication along with the Python 3 conversion: **1)** Taxonomy-based phage removal, **2)** poly N filtration for SARS-CoV-2, and **3)** Replacement of clustering algorithm from CD-HIT-EST to Many-against-Many sequence searching (MMseqs2). Therefore, the U-RVDBv[version].fasta is now served as the intermediate, raw U-RVDB for the downstream process instead of the final product in the old Python 2 pipeline. We chosed not to replace the Python 3 script name to ensure the backward compatibility.

**Creation of raw U-RVDB fasta file.** Following manual review, the unclustered raw RVDB fasta file can be generated using the script “create\_U-RVDB\_file.py”. Navigate to the home or parent directory (“RVDB” in this example) and enter the following command:

$python UPDATE\_SCRIPTS\_LOGS\_PY3/create\_U-RVDB\_file.py . apr.2025 30.0 RVDBv30.0.removeaccs.txt

“.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “RVDBv30.0.removeaccs.txt” is a file containing accessions for all of the entries that are to be excluded from the final fasta file. The create\_U-RVDB\_file.py script also screens out any duplicate entres – entries upon which RefSeq Viral entries were based (using accs in the files “refseq\_viral\_originalaccs.txt” in the RefSeq folder for the update, generated by rs\_acc\_mapping in the main command block).

**7. Raw U-RVDB post-processing**

The post-processing of U-RVDB contains the removal of phage sequences based on the taxonomy, post-editing of newly-added list from Step 5, and removal of SARS-CoV-2 sequences with ≥ 1% poly Ns.

Firstly, navigate to the RVDB release folder (RVDBv30.0 in this example):

$ cd RVDBv30.0

1. **Taxonomy-based phage removal**

This step contains the retrieval of all phage-associated taxonomy IDs (TaxIDs) down to the species level, cross- referencing phage-associated TaxIDs to accession IDs (AccIDs), and filtrating phage-associated AccIDS from raw-U-RVDB.

First, navigate to the RVDB sub-directory (RVDBv30.0 in this example), and prompt the following commands:

$awk -F ";" '{print $2}' ../rvdb\_update\_pipeline/phage\_taxid.list | xargs -I{} get\_species\_taxids.sh -t {} > allphage\_taxid.list

From the phage\_taxid.list, get all phage-associated TaxIDs down to the species level. The file format of phage\_taxid.list is [family\_name;taxID] in case that users want to append the phage family.

The following bash script downloads the AccID-TaxID cross reference file from NCBI and remove its header to facilitate the joining process.

$ wget ftp://ftp.ncbi.nlm.nih.gov/pub/taxonomy/accession2taxid/nucl\_gb.accession2taxid.gz && gunzip -f -d nucl\_gb.accession2taxid.gz

$ tail -n +2 nucl\_gb.accession2taxid > nucl\_gb.accession2taxid\_noheader

The following bash script joining the allphage\_taxid\_list with nucl\_gb.accession2taxid\_noheader by using taxID field as the key. The output “allphage\_acc.list” contains phage-associated AccIDs.

$ join -1 3 -2 1 -t $'\t' -o 1.1,1.2,1.3,2.1 <(sort -k 3,3 nucl\_gb.accession2taxid\_noheader) <(sort -k 1,1 allphage\_taxid.list) > allphage\_acc.list

The following two bash scripts retrieve the fasta headers from raw-U-RVDB and output phage-associated headers.

$ grep ">" raw-U-RVDBv30.0.fasta > raw-U-RVDBv30.0.fasta\_header

$ join -1 2 -2 2 -t $'\t' -o 1.1,2.2,1.2,1.3,1.4,1.5,1.6 <(awk -F "|" '{print $2"\t"$3"\t"$4"\t"$5"\t"$6"\t"$7}' raw-U-RVDBv30.0.fasta\_header|sort -k 2,2) <(sort -k 2,2 allphage\_acc.list) > U-RVDBv30.0\_phage\_candidate.list

The following three bash scripts scans the presence of phage sequences in the newly-added list to facilitate the manual review process (Step 5)

$ csvformat -T RVDBv30.0.new.csv > RVDBv30.0.new.tsv

$tail -n +2 RVDBv30.0.new.tsv > RVDBv30.0.new.tsv\_noheader

$ join -1 2 -2 2 -t $'\t' -o 1.1,2.2,1.2,1.3 <(awk -F "|" '{print $0}' RVDBv30.0.new.tsv\_noheader|sort -k 2,2) <(sort -k 2,2 allphage\_acc.list) > U-RVDBv30.0\_phage\_newly-added.list

1. **Post-editing the newly-added list**

The purposes to post-edit the newly-added list are to remove the following entries to reduce the burdon of manual review process (Step5): **1)** the SARS-CoV-2 records from newly-added list since they are abundant and truly viral, **2)** non-viral records manually identified in the previous RVDB release(s), if any, and **3)** the phage-associated records identified in the previousstep to be excluded. To be noted the records removed in **2)** and **3)** are retained and handled by the downstream process since they are *bona-fide* unwanted and/or non-viral sequences to be removed from the final RVDB release.

The following bash script removes millions of SARS-CoV-2 records from the newly-added list:

$awk -F "\t" '{if ($4!="Severe acute respiratory syndrome coronavirus 2") print $0}' RVDBv30.0.new.tsv > RVDBv30.0.new\_wo\_SARSCoV2.tsv

The following bash script scans the records in the removal list from previous RVDB release (if any) and excludes them from the current newly-added list:

$awk -F "\t" 'NR == FNR {a[$1]; next} !($2 in a)' ../RVDBv29.0/v29.0\_removeaccs.txt RVDBv30.0.new\_wo\_SARSCoV2.tsv > RVDBv30.0.new\_wo\_SARSCoV2\_v30.0\_removelist.tsv

In case you don’t have the previous removal list, please provide an empty file named “dummy.txt” and run the following bash script:

$awk -F "\t" 'NR == FNR {a[$1]; next} !($2 in a)' dummy.txt RVDBv30.0.new\_wo\_SARSCoV2.tsv > RVDBv30.0.new\_wo\_SARSCoV2\_v30.0\_removelist.tsv

The following bash script removes the phage records generated from Step 7A:

$awk -F "\t" 'NR == FNR {a[$2]; next} !($2 in a)' U-RVDBv30.0\_phage\_newly-added.list RVDBv30.0.new\_wo\_SARSCoV2\_v30.0\_removelist.tsv > RVDBv30.0.newly-added\_wo\_SARSCoV2\_wo\_phage.tsv

The final newly-added list named “RVDBv30.0.newly-added\_wo\_SARSCoV2\_wo\_phage.tsv” is ready for the manual review.

1. **Separation of non-SARS-CoV-2 and SARS-CoV-2 sequences**

Since the non- SARS-CoV-2 and SARS-CoV-2 sequences would be subjected to the different collapsing or clustering algorithm, and the poly N filtration is being implement to SARS-CoV-2 sequence exclusively. Therefore it is necessary to separate non- and SARS-CoV-2 sequences.

Please prompt the following three bash scripts to separate non- SARS-CoV-2 and SARS-CoV-2 sequences from raw-U-RVDBv30.0.fasta

$grep ">" raw-U-RVDBv30.0.fasta|sed 's/>//' > raw-U-RVDB v30.0\_header && awk -F "|" '{if ($5 == "Severe acute respiratory syndrome coronavirus 2") print $0}' raw-U-RVDBv30.0\_header > raw-U-RVDBv30.0\_sarscov2

$filterbyname.sh in=raw-U-RVDBv30.0.fasta out=raw-U-RVDBv30.0\_sarscov2.fasta names=raw-U-RVDBv30.0\_sarscov2 include=t ow=t

$filterbyname.sh in=raw-U-RVDBv30.0.fasta out=raw-U-RVDBv30.0\_nonsarscov2.fasta names=raw-U-RVDBv30.0\_sarscov2 include=f ow=t

“raw-U-RVDBv30.0\_nonsarscov2.fasta” and “raw-U-RVDBv30.0\_sarscov2.fasta” contain the collection of non- and SARS-CoV-2 sequences, respectively.

1. **Scanning the ambiguity nucleotide base in SARS-CoV-2 sequences**

The SARS-CoV-2 sequences are subjected to the ambiguity nucleotide (poly N) screening to remove the sequences containing ≥ 1% poly Ns. The binary file “N\_counter” in the folder “UPDATE\_SCRIPTS\_LOGS\_PY3” was written in C and precompiled by Unix gcc. The source code of N-counter is available under the same folder named “N\_counter.c” for users to compile based on their operation system and setup. Please prompt the following command to make it executable:

$ chmod +x ../UPDATE\_SCRIPTS\_LOGS\_PY3/N\_counter

And run the following bash script to generate the list of ambiguity nucleotide composition for SARS-CoV-2 sequences:

$../UPDATE\_SCRIPTS\_LOGS\_PY3/N\_counter < raw-U-RVDBv30.0\_sarscov2.fasta |sort -t $'\t' -nrk5 > raw-U-RVDBv30.0\_sarscov2\_N\_counter.list

The following bash script exstracts the records with ≥ 1% poly Ns to be included in the removal list:

$awk -F "\t" '{if ($5>=0.01) print $0}' raw-U-RVDBv30.0\_sarscov2\_N\_counter.list| awk -F "|" '{print $3}' > raw-U-RVDBv30.0\_SARSCoV2\_N1.00\_removeaccs

1. **Removing SARS-CoV-2 sequences with ≥ 1% poly Ns**

From the list generated from Step 7D, the list of fasta headers is generated by the following bash script:

$awk -F "|" 'NR==FNR{a[$0];next}$3 in a' raw-U-RVDBv30.0\_SARSCoV2\_N1.00\_removeaccs raw-U-RVDBv30.0\_header | sed 's/>//' > raw-U-RVDBv30.0\_SARSCoV2\_N1.00\_remove\_header

The fasta headers indicating to ≥ 1% poly Ns are used to remove from the RVDB release by the following bash script:

$filterbyname.sh in=raw-U-RVDBv30.0\_sarscov2.fasta out=raw-U-RVDBv30.0\_sarscov2\_N1.00.fasta names=raw-U-RVDBv30.0\_SARSCoV2\_N1.00\_remove\_header include=f ow=t

“raw-U-RVDBv30.0\_sarscov2\_N1.00.fasta” contains the final SARS-CoV-2 collection

**8. Clustering**

We have implemented the different clustering algorithm and workflow to produce clustered C-RVDB (C-RVDB) to overcome the challenge of SARS-CoV-2 redundancy (please see the detail information in the publication).

In brief, SARS-CoV-2 sequences are aligned to the Wuhan strain (NCBI AccID: NC\_045512.2) by minimap2. These with ≥98% identity are represented by NC\_045512.2. The remnant sequences with ≤98% identity are subjected to the second stage clustering and collapsing by MMseqs2.

For the non-SARS-CoV-2 sequences, they are not filtered by the ambibuity nucleotide screening (Step 7D-E) and directly subjected to the clustering and collapsing by MMseqs2.

1. **Clustering of SARS-CoV-2 sequences**

NC\_045512.2.fasta is included in “UPDATE\_SCRIPTS\_LOGS\_PY3” folder. To align SARS-CoV-2 sequences against NC\_045512.2, please run the following bash script:

$ minimap2 -t 20 ../ UPDATE\_SCRIPTS\_LOGS\_PY3/NC\_045512.2.fasta raw-U-RVDBv30.0\_sarscov2\_N1.00.fasta -o raw-U-RVDBv30.0\_sarscov2\_N1.00.fasta\_minimap2\_NC\_045512.2.out

Where “-t 20” indicates the number of CPUs to run minimap2. Please assign the number according to your hardware spec.

The following two bash scripts populate the lists containing the records of ≥ or < 98% identity against NC\_045512.2:

$ awk -F "\t" '{if ($10/$11 >= 0.98) print $0}' raw-U-RVDBv30.0\_sarscov2\_N1.00.fasta\_minimap2\_NC\_045512.2.out > raw-U-RVDBv30.0\_sarscov2\_A0.98\_minimap2\_NC\_045512.2.out

$ awk -F "\t" '{if ($10/$11 < 0.98) print $0}' raw-U-RVDBv30.0\_sarscov2\_N1.00.fasta\_minimap2\_NC\_045512.2.out > raw-U-RVDBv30.0\_sarscov2\_B0.98\_minimap2\_NC\_045512.2.out

The following two bash scripts generate the fasta headers of SARS-CoV-2 sequences with ≥ 98% identity against NC\_045512.2:

$awk -F "|" '{print $3}' raw-U-RVDBv30.0\_sarscov2\_A0.98\_minimap2\_NC\_045512.2.out|sort|uniq > raw-U-RVDBv30.0\_sarscov2\_A0.98\_minimap2\_NC\_045512.2\_acc

$awk -F "|" 'NR==FNR{a[$0];next}$3 in a' raw-U-RVDBv30.0 \_sarscov2\_A0.98\_minimap2\_NC\_045512.2\_acc raw-U-RVDBv30.0\_header | sed 's/>//' > raw-U-RVDBv30.0\_sarscov2\_A0.98\_minimap2\_NC\_045512.2\_fasta\_header

From the fasta headers, the SARS-CoV-2 sequences with < 98% identity are extracted and subjected to the clustering by MMseqs2:

$filterbyname.sh in=raw-U-RVDBv30.0\_sarscov2\_N1.00.fasta out=raw-U-RVDBv30.0\_sarscov2\_N1.00\_B0.98\_minimap2\_NC\_045512.2.fasta names=raw-U-RVDBv30.0\_sarscov2\_A0.98\_minimap2\_NC\_045512.2\_fasta\_header include=f ow=t

$ mmseqs easy-cluster raw-U-RVDBv$30.0\_sarscov2\_N1.00\_B0.98\_minimap2\_NC\_045512.2.fasta raw-C-RVDBv30.0\_sarscov2\_0.98 tmp --split-memory-limit 32G --cluster-mode 2 --cov-mode 1 --min-seq-id 0.98 --threads 20 -k 11

“--split-memory-limit 32G” assigns the RAM size used for dividing the target database in parts that fit into memory. “--threads 20” indicates the CPUs to be used for clustering. Please assign these two parameters according to your hareware spec.

“-k 11” and “--min-seq-id 0.98” indicate the k-mer length and the identity threshold to be clustered into the same clade. These two parameters are the same as the old CD-HIT-EST algorithm to generate the similar clustering result. Please refer to MMseqs2 user manual (https://mmseqs.com/latest/userguide.pdf), Page 72 for more detail.

To suppress fragments from becoming representative sequences, it is recommended to use --cluster-mode 2 in conjunction with --cov-mode 1 to align with the older RVDB production pipeline that the representative in each clade has to be the longest sequence. Please refer to MMseqs2 user manual (https://mmseqs.com/latest/userguide.pdf), Page 71 for more detail.

1. **Clustering of non-SARS-CoV-2 sequences**

Before clustering non-SARS-CoV-2 sequences, the removal sequences list, which is consisted of the previous removal list (if any), current removal list determined by the manual review process, and the phage list, is used to remove the unwanted sequences. The removal process is done by the following bash scripts.

Firstly, please include the AccIDs to be removed from the RVDB release in the file named “v30.0\_removeaccs\_raw.txt”, which is determined by the manual review process:

$cat <(awk -F "\t" '{print $3}' U-RVDBv30.0\_phage\_candidate.list) v30.0\_removeaccs\_raw.txt > v30.0\_removeaccs\_phage\_raw.txt

The following bash scripts removes the redundant records and remove the space before and/or after AccID, if any. The final removal list is produced, and the raw one is removed to avoid any confusion:

$ cat v30.0\_removeaccs\_phage\_raw.txt |tr -d "[:blank:]"|sort|uniq > v30.0\_removeaccs.txt && rm v30.0\_removeaccs\_raw.txt

The following two bash script retrieve the fasta header of unwanted sequences, and remove them from the RVDB release:

$ awk -F "|" 'NR==FNR{a[$0];next}$3 in a' v30.0\_removeaccs.txt raw-U-RVDBv30.0\_header | sed 's/>//' > raw-U-RVDBv30.0\_header\_for\_remove\_list

$ filterbyname.sh in=raw-U-RVDBv30.0\_nonsarscov2.fasta out=raw-U-RVDBv30.0\_clean\_nonsarscov2.fasta names=raw-U-RVDBv30.0\_header\_for\_remove\_list include=f ow=t

“raw-U-RVDBv30.0\_clean\_nonsarscov2.fasta” contains the non-SARS-CoV-2 sequences ready for being clustered or consolidated with quality-checked SARS-CoV-2 sequences to make U-RVDBv30.0.

The following bash script clusters non-SARS-CoV-2 sequences by MMseqs2:

$ mmseqs easy-cluster raw-U-RVDBv30.0\_clean\_nonsarscov2.fasta C-RVDBv30.0\_nonsarscov2 tmp --split-memory-limit 32G --cluster-mode 2 --cov-mode 1 --min-seq-id 0.98 --threads 20 -k 11

Please see Step 8A for the detail of MMseqs2 parameters

**9. Production of U-RVDB and C-RVDB**

1. **Production of U-RVDB**

The following bash script combines non-SARS-CoV-2 and quality checked SARS-CoV-2 sequences to make U-RVDBv30.0 and remove duplicate sequences if existed  
$ cat raw-U-RVDBv30.0\_clean\_nonsarscov2.fasta raw-U-RVDBv30.0\_sarscov2\_N1.00.fasta | seqkit rmdup -o U-RVDBv30.0.fasta -D duplicate\_U-RVDBv30.0.txt

Where “raw-U-RVDBv30.0\_sarscov2\_N1.00.fasta” (from Step 7E) contains the quality-checked SARS-CoV-2 sequences with ≤ 1% ambiguity nucleotides. “duplicate\_U-RVDBv30.0.txt” contains the records of duplicated sequences in U-RVDBv30.0. There should not be any duplicates.

1. **Production of C-RVDB**

The following bash script combines clustered non- SARS-CoV-2 (Step 8B) and SARS-CoV-2 (Step 8A) sequences to make the C-RVDB release:

$cat C-RVDBv30.0\_nonsarscov2\_rep\_seq.fasta raw-C-RVDBv30.0\_sarscov2\_0.98\_rep\_seq.fasta |seqkit rmdup -o C-RVDBv30.0.fasta -D duplicate\_C-RVDBv30.0.txt

Where “C-RVDBv30.0\_nonsarscov2\_rep\_seq.fasta” (from Step 7E) and “raw-C-RVDBv30.0\_sarscov2\_0.98\_rep\_seq.fasta” contain the clustered non-SARS-CoV-2 and SARS-CoV-2 sequences produced in Step 8B and 8A, respectively. “duplicate\_C-RVDBv30.0.txt” contains the records of duplicated sequences in C-RVDBv30.0. There should not be any duplicates.

**10. Characterization**

**Overview.** The “RVDB\_characterization.py” script was used to partition the sequences into five Level 1 categories: exogenous viral (EX), endogenous nonretroviral (ENRV), endogenous retroviral (ERV), LTR-retrotransposon (LTR\_Reto), and unassigned viral gene /fragments (Unassigned). This partitioning was done using some of the SEM-R positive keywords, and organizing them by categories. Sequences possessing headers with specific positive keywords from SEM-R screen were placed into the corresponding categories. For instance, the keywords “retrotranspos”,”retro transpos”,”retroelem”,”blastopia “,” copia “,” delta element”,” gipsy “,” gypsy element”,” gypsy like “,” gypsy type “, “insertion element”, ” mdg1 “, ” mdg3 “, ”micropia”, “ sire “,” ty element”, and “ ty insertion” were used to classify sequences as belonging to the LTR-retrotransposon category. There is also a regular expression for finding strings of the form “ ty” + either “1” or “3”, with / without a space, and there are also a handful of rules for pulling in the less common LTR-retrotransposons: the string “transpos” + either “ bel “, “ pao “, “ roo “, or “morgane”. There are similar combinations of keywords, regular expressions, and rules for the other four groups.

**Running RVDB\_characterization.py.** The RVDB\_characterization.py script can be run by a single line in the command shell, containing python command, the script name, and then 5 parameters: the home or parent directory (one level below the update folder), the date tag for the update, the current version of the update, the name of the fasta file to be characterized (e.g. “U-RVDBv30.0.fasta”), and a filename containing a filterset, an accession list for a subset of sequences to be characterized. The last two parameters can be selected so that the script can be run not just on the base unclustered form of RVDB, but also the clustered form of RVDB, or any special-purpose sub-version created by the user. Please note, if all sequences from the supplied fasta file are to be characterized, there is no filterset and the final parameter can be a random letter (e.g. “NA”). Below is an example of running the script:

$python UPDATE\_SCRIPTS\_LOGS\_PY3/RVDB\_characterization.py . apr.2025 30.0 U-RVDBv30.0.fasta NA

The counts for each group are recorded in a log file, $fastafilename”\_char\_output\_log.txt”, which is in the current update folder. For example:

RVDB/RVDBv30.0/U-RVDBv30.0\_char\_output\_log.txt

In addition, the script generates output files of headers for each of the categories. These are also written to the current update folder and are named $fastafilename”.”$group”.headers.txt”, for example:

RVDB/RVDBv30.0/RVDBv30.0.fasta.EX.headers.txt

RVDB/RVDBv30.0/U-RVDBv30.0.fasta.ENRV.headers.txt

RVDB/RVDBv30.0/U-RVDBv30.0.fasta.ERV.headers.txt

RVDB/RVDBv30.0/U-RVDBv30.0.fasta.LTR-RETO.headers.txt

RVDB/RVDBv30.0/U-RVDBv30.0.fasta.UNASSIGNED.headers.txt

**Manual review of characterization output.** In our characterization efforts, we did find it necessary to perform some final manual review. In particular, we found that some sequences that had been labelled by the RVDB\_characterization.py script as endogenous nonretroviral, were in fact endogenous retroviral according to their name. Also, some of the sequences that had been labelled by the script as “viral gene/fragment” were in fact exogenous viral or LTR-retrotransposon.

**11. Creation of RVDB SQL**

Having an SQL implementation of the RVDB is useful for the same reason that all SQL databases are useful – rapid and flexible querying of content. Most parameters by which one can query the RVDB SQL correspond to fields in the header, but in addition there are category (as defined using RVDB\_characterization.py; see section 10) as well as sequence length parameters. The RVDB SQL database is created in sqlite form using the sqlite API in Python. The SQL database is created by calling the script “make\_alter\_build\_sqlite3db\_v2.py” in the following manner:

$python UPDATE\_SCRIPTS\_LOGS\_PY3/make\_alter\_build\_sqlite3db\_v2.py . apr.2025 30.0 U-RVDBv30.0.fasta

**12. Generation of release notes**

After both the unclustered and clustered forms of the database have been generated as fasta files, the release notes need to be generated. The release notes script exists to generate summary statistics as well as update information in a simple but standardized format. These statistics include total counts of sequences in each category (RefSeq Viral, RefSeq Viral neighbor, GenBank division, or TPA) as well as date (month and year) information for the download of RefSeq and GenBank flat files. The release notes scripts are called using the following two commands:

$python UPDATE\_SCRIPTS\_LOGS\_PY3/create\_relnotes.py . apr.2025 13.0 4 may 2025 apr 2025 apr 2025 U-RVDBv30.0.fasta 228 265

$python E:/UPDATE\_SCRIPTS\_LOGS/create\_relnotes.py . apr.2025 30.0 4 may 2025 apr 2025 apr 2025 C-RVDBv30.0.fasta 228 265

“.” is the home or parent directory, “apr.2025” is the date of the update, “30.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “4 may 2025” are the day, month, and year for the update (usually the date of running of the create\_relnotes.py script). The first “apr 2025” is the month and year of the GenBank download, while the second “apr 2025” is the month and year of the RefSeq download. Finally, “U-RVDBv30.0.fasta”and “C-RVDBv30.0.fasta” are the names of the input RVDB fasta files. “228” and “265” are RefSeq and GenBank release number, respectively. The script needs to be run twice, once with the name of the unclustered fasta file and once with the name of the clustered fasta file, because two release notes need to be generated. Note that once you have generated the release note, you also have to generate the checksum/MD5 value separately and paste it into the release notes. This can be accomplished in a single command. Please remove all whitespace characters before copying and pasting.

$md5sum $PathToRVDB > MD5\_Output

So, for example:

$md5sum U-RVDBv30.0.fasta > U-RVDBv30.0.fasta.md5

$md5sum C-RVDBv30.0.fasta > C-RVDBv30.0.fasta.md5

$md5sum U-RVDBv30.0.sqlite.db > U-RVDBv30.0.fasta.md5

And most of all, enjoy RVDB!