**Instructions for updating RVDB**

**1. Create folders for update**

First, create a directory for the update. This should be in “RVDBv$version” format, for example the following line would create the folder for version 13.0:

mkdir RVDBv13.0

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 2018, the folders could be created using Windows cmd.exe using the following command:

cd RVDBv13.0

mkdir GenBank\_raw\_data\_apr.2018 && mkdir TPA\_raw\_data\_apr.2018 && mkdir RefSeq\_raw\_data\_apr.2018

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.2018

mkdir log && mkdir scripts && mkdir poskw\_out\_apr.2018 && mkdir sizemirna\_out\_ apr.2018 && mkdir negkw\_out\_ apr.2018

**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) (note, from cmd.exe you have to ftp into [ftp.ncbi.nih.gov](ftp://ftp.ncbi.nih.gov) first, provide a username and password, type “binary” and “prompt” to switch the binary mode to “I” and to turn off file download prompts, then navigate to 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 opening a web browser and download some files. Log on to: <https://www.ncbi.nlm.nih.gov/genome/viruses/> and under “Download Viral Genome Data” click on “Accession list of all viral genomes”. Save target 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.23.2018.nbr”) in the Refseq main folder. 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.23.2018.csv”). You can delete the original .nbr file after completing this step.

**Phage.** There is an in-house 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/release.notes/> and download the most recent file. Save this file in “gb\_releasenotes\_v$version\_$month.$year.txt” format, for example gb\_releasenotes\_v225\_apr.2018.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.** Use the following concatenated commands (described individually beneath the command block):

python E:/UPDATE\_SCRIPTS\_LOGS/parse\_raw\_refseq\_PIPE.py E: apr.2018 13.0 viral.1.1.genomic.fna.gz viral.2.1.genomic.fna.gz && python E:/UPDATE\_SCRIPTS\_LOGS/multiple\_gzunzip\_PIPE.py E: apr.2018 13.0 viral.1.genomic.gbff.gz viral.2.genomic.gbff.gz viral.genomic.gbff && python E:/UPDATE\_SCRIPTS\_LOGS/fileops\_PIPE.py E: apr.2018 13.0 gbff 1000000 && python E:/UPDATE\_SCRIPTS\_LOGS/rs\_acc\_mapping\_PIPE.py E: apr.2018 13.0 && python E:/UPDATE\_SCRIPTS\_LOGS/VDBunzip\_reformat\_gb\_to\_fasta\_PIPE.py E: apr.2018 13.0 gb && python E:/UPDATE\_SCRIPTS\_LOGS/VDBupdate\_checkpoint2\_PIPE.py E: apr.2018 13.0 gb\_releasenotes\_v225\_apr.2018.txt && python E:/UPDATE\_SCRIPTS\_LOGS/SEM-R\_PIPE.py E: apr.2018 13.0 poskw gb && python E:/UPDATE\_SCRIPTS\_LOGS/SEM-R\_PIPE.py E: apr.2018 13.0 sizemirna gb && python E:/UPDATE\_SCRIPTS\_LOGS/SEM-R\_PIPE.py E: apr.2018 13.0 negkw gb

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

python E:/UPDATE\_SCRIPTS\_LOGS/parse\_raw\_refseq\_PIPE.py E: apr.2018 13.0 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). “F:” is the home directory, “dec.2017” is the date of the update, “12.0” 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, F:/RVDBv12.0.

python E:/UPDATE\_SCRIPTS\_LOGS/multiple\_gzunzip\_PIPE.py E: apr.2018 13.0 viral.1.genomic.gbff.gz viral.2.genomic.gbff.gz viral.genomic.gbff

Combines the two GenBank flat files for refseq viral into one. “E:” is the home directory, “apr.2017” is the date of the update, “13.0” is the version of RVDB; these parameters are needed to identify the directory for the update.

python E:/UPDATE\_SCRIPTS\_LOGS/fileops\_PIPE.py E: apr.2018 13.0 gbff 1000000

Splits the combined GenBank flat file into multiple files, so that each can be read into Python. “E:” is the home directory, “apr.2018” is the date of the update, “13.0” 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 E:/UPDATE\_SCRIPTS\_LOGS/rs\_acc\_mapping\_PIPE.py E: apr.2018 13.0

Using the GenBank flat file metadata for RefSeq viral, finds the duplicate entries’ accessions (original entries, upon which RefSeq viral entries were based). “E:” is the home directory, “apr.2018” is the date of the update, “13.0” 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, “E:/RVDBv13.0/RefSeq\_raw\_data.apr.2018/ refseqviral\_neighbors\_mapping.apr.23.2018.csv”; this filename is hard-coded into the script). The neighbors are saved in the file (here “E:/RVDBv13.0/RefSeq\_raw\_data.apr.2018/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 “E:/RVDBv13.0/RefSeq\_raw\_data.apr.2018/refseq\_viral\_originalaccs.txt” ; this filename is also hard-coded as input for the unzipping script.

python E:/UPDATE\_SCRIPTS\_LOGS/VDBunzip\_reformat\_gb\_to\_fasta\_PIPE.py E: apr.2018 13.0 gb

Unzips the GenBank division files, labels sequences that are RefSeq viral neighbors during the unzipping. “E:” is the home directory, “apr.2018” is the date of the update, “13.0” is the version of RVDB; these parameters are needed to identify the directory for the update. Please note that a modified form of the GenBank Scanner.py script (found in Biopython, typically in Python sub-folder: Lib/site-packages/Bio/GenBank) should be used, in place of the original. The modified version of the Scanner.py script (available on <https://github.com/ArifaKhanLab/RVDB>) can simply be copied and pasted into the same directory as the original Scanner.py script (overwrite previous). To avoid having to change additional lines in calling scripts to accommodate a different name for Scanner.py, the name of Scanner.py was not changed. The modifications should not hinder any existing functionality of the script, so it can safely be used in place of the original. The modified Scanner.py contains a try/except block in two places, to correct for an error that occurred in a small number of entries, related to the ‘Structured Comments’ metadata. In a very small number of cases, it was noted that the standard Scanner.py script was trying to extract the ‘Structured Comments’ metadata, when this metadata was in fact not present.

python F:/UPDATE\_SCRIPTS\_LOGS/VDBupdate\_checkpoint2\_PIPE.py E: apr.2018 13.0 gb\_releasenotes\_v225\_apr.2018.txt

“E:” is the home directory, “apr.2018” is the date of the update, “13.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “gb\_releasenotes\_v225\_apr.2018.txt” is the name of the release notes file that was downloaded from the GenBank ftp site.

Runs checkpoint2, generates four output files: E:/UPDATE\_SCRIPTS/\*month\*\*year\*\_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\_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 E:/UPDATE\_SCRIPTS\_LOGS/SEM-R.py E: 13.0 apr.2018 poskw gb

Runs the positive keyword screen. “E:” is the home directory, “apr.2018” is the date of the update, “13.0” 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 E:/UPDATE\_SCRIPTS\_LOGS/SEM-R\_PIPE.py E: 13.0 apr.2018 sizemirna gb

Runs the size/mirna screen. “E:” is the home directory, “apr.2018” is the date of the update, “13.0” 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 E:/UPDATE\_SCRIPTS\_LOGS/SEM-R\_PIPE.py E: 13.0 apr.2018 negkw gb

Runs the negative keyword screen. “E:” is the home directory, “apr.2018” is the date of the update, “13.0” 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 E:/UPDATE\_SCRIPTS\_LOGS/VDBunzip\_tpa\_PIPE.py E: apr.2018 13.0 fsa\_nt.gz && python E:/UPDATE\_SCRIPTS\_LOGS/SEM-R\_PIPE.py E: apr.2018 13.0 poskw tpa && python E:/UPDATE\_SCRIPTS\_LOGS/SEM-R\_PIPE.py E: apr.2018 13.0 sizemirna tpa && python E:/UPDATE\_SCRIPTS\_LOGS/SEM-R\_PIPE.py E: apr.2018 13.0 negkw tpa

“E:” is the home directory, “apr.2018” is the date of the update, “13.0” 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\_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 enw 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 (“RVDBv13.0.removeaccs.txt”). This is done manually. The prep\_manual\_review.py script is called from the command line as follows:

> python E:/UPDATE\_SCRIPTS\_LOGS/prep\_manual\_review.py E: apr.2018 13.0 E:/RVDBv12.2/U-RVDBv12.2.fasta

“E:” is the home directory, “apr.2018” is the date of the update, “13.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “E:/RVDBv12.2/U-RVDBv12.2.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. “RVDBv13.0.missing.csv” and “RVDBv13.0.new.csv”.

**6. Clustering and creation of RVDB fasta files**

**Creation of U-RVDB fasta file.** Following manual review, the unclustered RVDB fasta file can be generated using the script “create\_U-RVDB\_file.py”. In windows cmd.exe, navigate to the working directory and enter the following command:

>python E:/UPDATE\_SCRIPTS\_LOGS/create\_U-RVDB\_file.py E: apr.2018 13.0 RVDBv13.0.removeaccs.txt

“E:” is the home directory, “apr.2018” is the date of the update, “13.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “RVDBv13.0.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).

**Clustering.** The unclustered fasta file for the updated RVDB should be uploaded onto the high-performance computing cluster (HPC), BLUEFIN (BF). The target directory on BF should be: /projects/arifakhanlabs/cdhit\_runs. The script to modify is called cdhit.sh, and what needs to be modified are the -i and -o parameters in the execute line. For example, for clustering (at 98% sequence identity, which is standard) of U-RVDBv13.0, the execute line should read:

time $APP -i U-RVDBv13.0.fasta -o U-RVDBv13.0.98 -c 0.98 -n 11 -M 0 -d 0 -g 1 -T $NSLOTS

You may also wish to modify the name of the job (top of script, line starting with -N) – but this will not alter the outcome of the clustering. Once clustering is complete, the .clstr file should be downloaded from the HPC into the main folder for the update. Note that three files are generated as output by CD-HIT-EST on the HPC, but only the one ending in .clstr needs to be downloaded.

**Editing the clustering output.** There are two things that need to be modified in the clustered (.clstr) output file. The first is the headers – in the clustered output, the headers have been abbreviated to about 20-30 characters, cutting off in mid-description. So the headers need to be restored by copying and pasting the full headers from the U-RVDB file. The second modification needed is to promote a RefSeq Viral or, in its absence, a RefSeqViral neighbor sequence to cluster representative status, in that order. This is performed where one of these is present but not the longest sequence in the cluster (and hence were not made the cluster representative by CD-HIT). Both edits are taken care of by running the edit\_rawcdhit\_output.py script in the following manner:

python E:/UPDATE\_SCRIPTS\_LOGS/edit\_rawcdhit\_output.py E: apr.2018 13.0 U-RVDBv13.0.98.clstr

“E:” is the home directory, “apr.2018” is the date of the update, “13.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “U-RVDBv13.0.98.clstr” is the name of the raw clustering output file from CD-HIT-EST.

**Creation of C-RVDB fasta file.** After clustering, the clustered RVDB fasta file can be generated using the following steps. The following are instructions using CD-HIT-EST clustered output. In windows cmd.exe, navigate to the working directory for the update and enter the following command:

python E:/UPDATE\_SCRIPTS\_LOGS/create\_C-RVDB\_file.py E: apr.2018 13.0 U-RVDBv13.0.fasta U-RVDBv13.0.98.promoted.clstr

“E:” is (in this example) the home directory or directory below the update directory, “apr.2018” is the date of the update, “13.0” is the version RVDB, “U-RVDBv13.0.fasta” is the name of the unclustered fasta file, and “RVDBv13.0.98.promoted.clstr” is the name of the edited CD-HIT-EST clustering output file. The script will generate a file called “C-RVDBv13.0.fasta” in the main directory for the update. Note that the create\_C-RVDB\_file.py script prints out the total number of headers and the total number of lines in the cmd.exe shell, after completing. It should be verified that the second number is exactly twice the former – to ensure that every entry has exactly two lines, a header and a sequence line.

**7. 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 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-RVDBv13.0.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 from windows.cmd:

>python E:/UPDATE\_SCRIPTS\_LOGS/RVDB\_characterization.py E: apr.2018 13.0 U-RVDBv13.0.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:

E:/RVDBv13.0/U-RVDBv13.0\_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:

E:/RVDBv13.0/RVDBv13.0.fasta.EX.headers.txt

E:/RVDBv13.0/RVDBv13.0.fasta.ENRV.headers.txt

E:/RVDBv13.0/RVDBv13.0.fasta.ERV.headers.txt

E:/RVDBv13.0/RVDBv13.0.fasta.LTR-RETO.headers.txt

E:/RVDBv13.0/RVDBv13.0.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.

**8. 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 7) 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 E:/postmay2018/UPDATE\_SCRIPTS\_LOGS/make\_alter\_build\_sqlite3db\_v2.py E:/postmay2018/RVDB\_postpub apr.2018 13.0 U-RVDBv13.0.fasta

**9. Upload to GW HIVE**

**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 E:/UPDATE\_SCRIPTS\_LOGS/create\_relnotes.py E: apr.2018 13.0 4 may 2018 apr 2018 apr 2018 U-RVDBv13.0.fasta

python E:/UPDATE\_SCRIPTS\_LOGS/create\_relnotes.py E: apr.2018 13.0 4 may 2018 apr 2018 apr 2018 C-RVDBv13.0.fasta

“E:” is the home directory, “apr.2018” is the date of the update, “13.0” is the version of RVDB; these parameters are needed to identify the directory for the update. “4 may 2018” are the day, month, and year for the update (usually the date of running of the create\_relnotes.py script). The first “apr 2018” is the month and year of the GenBank download, while the second “apr 2018” is the month and year of the RefSeq download. Finally, “U-RVDBv13.0.fasta”and “C-RVDBv13.0.fasta” are the names of the input RVDB fasta files. 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 from Windows cmd.exe. Please remove all whitespace characters before copying and pasting.

certUtil -hashfile $pathtoRVDBfile MD5

So, for example:

certUtil -hashfile E:/RVDBv13.0/U-RVDBv13.0.fasta MD5

The size and #bases fields in the release notes should be left blank – GW Hive calculate these themselves once the files are uploaded.

**Registering for and accessing the GWU HIVE.** Web address for HIVE (go here both to register initially and to log in) <https://hive.biochemistry.gwu.edu/home> . Go here to obtain an account and to do the upload. If you are logging in to do the upload, click “Login” at top right. On the following page, make sure you click the “click here” link in smaller font: “Please click here if you are trying to login to the HIVE platform”. Then enter your email and password. Note that if you do not log in regularly (a month approx. – don’t know exactly) it expires your password and the next time you log on you will be asked to reset to a new password. Once logged in, upload the five files (U-RVDB and C-RVDB fasta files, U-RVDB and C-RVDB release notes, and U-RVDB sql file). To do this, click the icon of a globe with an upwards-pointing purple arrow. When you hover over the icon, it should say “Upload a file”. Click it and upload each of the 5 files. Finally, you have to share permissions with Naila Gulzar (who is also the contact for any technical issues: Naila Gulzar, [naila\_gulzar@email.gwu.edu](mailto:naila_gulzar@email.gwu.edu)) or, if she is not available (and she is out until July, 2018 at least) Janisha Patel [janishapatel@gwmail.gwu.edu](mailto:janishapatel@gwmail.gwu.edu). To share permissions, first select all five files in the scrolling files menu, that have just uploaded. Then look for the “sharing” icon at the bottom of the screen below the scrolling files list. This is another globe-icon, but without an arrow on it. Click the “sharing” icon, selected Naila Gulzar’s name from the drop-down menu, and then selected all permissions except delete and administer. Hit “apply”. Update is now complete from our end – all that remains is for GW HIVE to calculate # bases, size, and verify the MD5/checksum value.

And most of all, enjoy RVDB.