**Comprehensive Guide for APOBEC3 Analysis at Branch Level**

Apocount and Apoplot are a set of Python scripts designed to analyze and visualize APOBEC3 mutations within DNA sequences, particularly at a branch level. These tools and their associated scripts are valuable for categorizing APOBEC3 mutants and understanding their genomic distribution.

* Apobec (Utility Module): This foundational module contains the core logic for defining APOBEC patterns, identifying APOBEC-relevant sites, and performing sequence manipulations like reverse-complementing.
* Apocount: This script identifies and quantifies characteristic APOBEC3 mutation patterns within sequences, relative to a reference. It outputs detailed mutation tables that can be quantified to give statistical summaries.
* Apoplot: This script generates graphical representations of mutations along DNA sequences, specifically highlighting APOBEC3-related mutations with distinct colors and markers, relative to a reference.
* Mergesnps: This utility combines all unique SNP mutations from a set of sequences into a single "merged" sequence, based on the most common base at each mutated site.

Characteristic APOBEC3 Mutations: For DNA sequences, the hallmarks of APOBEC3-induced mutations are:

* G>A transitions within specific trinucleotide contexts.
* C>T transitions within specific trinucleotide contexts (the reverse complement of the above).

Availability: These scripts are available within the jt-lanl/cov-voc repository, [jt-lanl/cov-voc: covid+monkepox+apobec](https://www.google.com/search?q=https://github.com/jt-lanl/cov-voc) [jt-lanl/cov-voc: Routines for tracking and analyzing variants of covcern for the SARS-CoV-2 virus](https://www.google.com/search?q=https://github.com/jt-lanl/cov-voc)

See 5. For a complete example workflow.

**1. The Core: apobec.py**

This script provides the fundamental definitions and functions for APOBEC analysis that apocount.py and apoplot.py build upon.

**1.1. Key Definitions**

The apobec.py script defines the APOBEC signature patterns based on trinucleotide contexts.

* Default (Strict) APOBEC Patterns:
  + Forward (G>A mutations): APOBEC\_FD is defined as a set of trinucleotides where a G is followed by an A or G, and then any base.
    - GA followed by A, G, or T (e.g., GAA, GAG, GAT)
    - GG followed by A, G, or T (e.g., GGA, GGG, GGT)
  + Reverse Complement (C>T mutations): APOBEC\_RC is defined as a set of trinucleotides where any base is followed by a C or T, and then a C. This is the reverse complement of the forward patterns.
    - A, C, or T followed by TC (e.g., ATC, CTC, TTC)
    - A, C, or T followed by CC (e.g., ACC, CCC, TCC)
* Loose APOBEC Rules:
  + The loosen\_apobec\_rules() function modifies the global APOBEC\_FD and APOBEC\_RC sets. If called, it expands the last base in the pattern to include C as well as A, G, T (or ACGT).
    - Forward (G>A mutations): GA or GG followed by *any* base.
    - Reverse Complement (C>T mutations): *Any* base followed by TC or CC.
  + This is why apocount.py and apoplot.py have --strict and --loose arguments; they directly control which set of rules is used from this module.

**2. mergesnps.py: Creating a Consensus Mutation Sequence**

mergesnps.py is a utility script designed to aggregate all observed mutations from a set of sequences (relative to a reference) into a single, new "merged" sequence. At each mutation site, the merged sequence will adopt the *most common* variant base found among the input sequences.

2.1. Key Actions

* Identifies All Mutations: It iterates through all input sequences (excluding the first, which is treated as the reference) and finds every site where they differ from the reference using apobec.get\_all\_mutsites().
* Determines Most Common Base: For each mutation site, it collects all the different bases observed across all input sequences at that position. It then determines the most frequently occurring base at that site.
* Constructs Merged Sequence: It creates a new sequence that is initially identical to the reference. Then, at every identified mutation site, it replaces the reference base with the most common variant base determined in the previous step.
* Appends to Output: The new "Merged-SNPs" sequence is added to the list of sequences. When writing to an output file, this merged sequence will appear as the second sequence, immediately after the original reference.

Bash

python3.7 /path/to/jt-lanl-cov-voc-2d20d77/mergesnps.py \

--input /path/to/your\_alignment.fasta \

--output /path/to/your\_alignment\_with\_merged.fasta \

-v > /path/to/mergesnps\_log.txt

2.2. Command-Line Arguments

* -i or --input <path/to/fasta> (Required):
  + Specifies the path to your input sequence file in FASTA format. The first sequence in this file will be considered the reference, and all other sequences will be compared against it.
* -o or --output <path/to/fasta> (Optional):
  + Specifies the output filename for the FASTA file. This file will contain:
    1. The original reference sequence.
    2. The newly created "Merged-SNPs" sequence.
    3. All other original input sequences.
* -v or --verbose (Optional, can be repeated):
  + Increases the verbosity level. -v for basic info, -vv for more detailed messages.

2.3. Output

* A FASTA file (.fasta or .fa) containing the reference sequence, the new merged SNP sequence, and all original sequences.
* Verbose output to standard error (or redirected to a log file) if --verbose is used.

**3. apocount.py: Quantification and Statistics**

apocount.py quantifies APOBEC patterns and mutations. It reads sequences from a FASTA file, using the first sequence as a reference, and then compares all other sequences to it.

3.1. Key Actions

* Identifies APOBEC Sites: It first determines all potential G-to-A APOBEC sites (and their C-to-T reverse complements) in the *reference sequence* based on the apobec.py rules (strict or loose).
* Counts Mutations: For each non-reference sequence, it counts:
  + G>A mutations at identified APOBEC sites.
  + C>T mutations at identified APOBEC sites.
  + All other mutations (including non-APOBEC and other base changes).
* Statistical Analysis: It constructs a 2x2 contingency table for each sequence, comparing the occurrence of G>A/C>T mutations at APOBEC sites versus non-APOBEC sites. It then performs a Fisher's Exact Test to calculate a p-value and odds ratio, indicating the statistical significance and enrichment of APOBEC mutations.

3.2. Command-Line Arguments (Summary)

Bash

python3.7 ./jt-lanl-cov-voc-2d20d77/apocount.py \

--input your\_alignment.fasta \

--summary apobec\_summary.txt \

--table \

--strict \

-vv > apocount\_log.txt

apocount.py [-h] [--input INPUT] [--nseq NSEQ] [--reversecomplement] [--fwd FWD] [--strict] [--loose] [--table] [--summary SUMMARY] [--verbose]

apocount.py: error: unrecognized arguments: --output

* --input (Required): Input FASTA file (first sequence is reference).
* --nseq: Limit number of sequences to analyze.
* --reversecomplement: Reverse complement sequences before analysis.
* --fwd (F, R, FR): Specify forward (G>A), reverse complement (C>T), or both mutation types to consider for APOBEC.
* --strict / --loose: Apply strict or loose APOBEC pattern definitions (uses apobec.loosen\_apobec\_rules()).
* --table: Print the detailed contingency table.
* --summary <file>: Write detailed mutation summary to a file (space-separated, not CSV).
* -v / --verbose: Increase verbosity.

3.3. Output

* Standard Output: For each sequence, it prints a line showing the sequence name, count of "other" mutations, a condensed contingency table, Fisher's exact test p-value, and odds ratio.
* Summary File (--summary): Provides a detailed breakdown of mutations by type (e.g., APO-GA, APO-CT, TC, AG, etc.) and their alignment-based genomic positions.

**4. apoplot.py: Visualization of Mutation Patterns**

apoplot.py generates graphical plots that show the locations and types of mutations along DNA sequences. It's particularly useful for visualizing hotspots and patterns of APOBEC mutagenesis.

4.1. Key Actions

* Plot Generation: Creates a matplotlib plot where each sequence (or sequence pair) is represented by a horizontal line.
* Mutation Ticks: Vertical tick marks are drawn at mutation sites.
  + Above line: G>A mutations.
  + Below line: C>T mutations.
  + Color-coding: Blue/Cyan for APOBEC-signature mutations (based on strict/loose rules from apobec.py), Red for non-APOBEC G>A/C>T, Gray for other mutations.
* Gap/N Indication: Can optionally show insertions, deletions, and 'N' (ambiguous) bases using colored fills.
* Gene Boundaries: Can optionally mark the hardcoded F13L gene boundaries.
* Key Generation: Can create a separate plot explaining the color codes.

4.2. Command-Line Arguments (Summary)

Bash

python3.7 ./jt-lanl-cov-voc-2d20d77/apoplot.py \

--input your\_alignment.fasta \

--output mutation\_plot\_original\_ref.png \

--showgaps \

--showgene \

--keepfirst \

--strict \

--keyfile apobec\_key.png \

-v

* -h, --help: show this help message and exit
* --input INPUT, -i INPUT: input sequence file
* --nseq: ##, if specified, then only plot this many sequences
* --reversecomplement, -r, reverse complement the strings as they are read in
* --strict, Use stricter APOBEC pattern definition
* --loose, Use looser APOBEC pattern definition
* --keepfirst, Keep the reference sequence in the plot
* --merge, Add an extra merge sequence
* --trim ##, Trim this many bases off of front and back of sequence
* --pairs PAIRS.txt, file with pairs of sequence names
* --output OUTPUT, -o OUTPUT, write plot to output file, add .svg, .png, .pdf to end of output name to specify format
* --showgaps show where the reference sequence gaps are in the alignment
* --showgene show where the F13L gene is
* --nosnp NOSNP.txt write sequences names with no SNPs into file
* --numlabel NUMLABEL, name of file to put names of sequences, associated with numbers
* --keyfile KEYFILE, name of color keyfile
* --verbose, -v verbose4.3. Output

4.3. Outputs

A graphical image file (e.g., .png) displaying the mutation plots.

Optionally, a separate image file for the color key.

Optional text files listing sequences with no SNPs or numerical labels for the plot.

**5. Running the Tools: A Common Workflow**

5.1. General Setup

Bash

# It's good practice to ensure your environment is set up.

# If using conda:

# Install dependencies:

pip install numpy matplotlib scipy

# If you get 'Tkinter' errors when apoplot tries to display a plot,

# set the Matplotlib backend to 'agg' before running apoplot.py:

export MPLBACKEND=agg, see example

5.2. Example Workflow

Here's a typical workflow that might involve all three scripts:

1. Prepare your input FASTA file: Ensure it's an MSA (Multiple Sequence Alignment) for best results, especially if you plan to use --showgaps with apoplot.py. The first sequence in the file will be treated as the reference.
2. Generate a merged SNP sequence using mergesnps.py: This can be useful for creating a "consensus of variation" among your sequences, which you might then use as a reference for further analysis or plotting.

Bash

python3.7 ./jt-lanl-cov-voc-2d20d77/mergesnps.py \

--input your\_alignment.fasta \

--output your\_alignment\_with\_merged.fasta \

-v > mergesnps\_log.txt

This will create your\_alignment\_with\_merged.fasta which includes the original reference, the new Merged-SNPs sequence, and all your original sequences.

1. Quantify APOBEC mutations with apocount.py: You can use your original alignment or the one with the merged SNPs.

Bash

python3.7 ./jt-lanl-cov-voc-2d20d77/apocount.py \

--input your\_alignment.fasta \

--summary apobec\_summary.txt \

--table \

--strict \

-vv > apocount\_log.txt

1. Visualize mutations with apoplot.py: You might plot your original alignment, or plot against the new Merged-SNPs sequence if you want to see variations relative to this consensus.

Bash

export MPLBACKEND=agg # Important for headless environments

# Example 1: Plotting original sequences against the original reference

python3.7 /path/to/jt-lanl-cov-voc-2d20d77/apoplot.py \

--input /path/to/your\_alignment.fasta \

--output /path/to/mutation\_plot\_original\_ref.png \

--showgaps \

--showgene \

--keepfirst \

--strict \

--keyfile apobec\_key.png \

-v

# Example 2: Plotting against the newly created merged SNP sequence (Requires a pairs file or you'd need to reorder the fasta for default mode)

You need to manually create a pairs.txt file like:

Txt File could contain any of these pairs where the first listed sequence is the reference:

Merged-SNPs SequenceA

Merged-SNPs SequenceB

Sequence A SequenceB

Reference Sequence B

Or, if you want to use the default mode, you'd need to rearrange your fasta file (I’ve found this hyper complicated; the list is easier).

For simplicity, using paired mode here is clearer for custom references:

python3.7 /path/to/jt-lanl-cov-voc-2d20d77/apoplot.py \

--input /path/to/your\_alignment\_with\_merged.fasta \

--output /path/to/mutation\_plot\_merged\_ref.png \

--pairs /path/to/your\_merged\_pairs.txt \

--showgaps \

--strict \

-v