**BarcodeRabbit-- Version ca05 073125**

**SSU rRNA-based Reference-agnostic phylogenetic survey of cellular composition from NGS datasets**

**->Fuction:**

BarcodeRabbit is designed to make use of a highly conserved "core" motif in defining composition of source material for NGS datasets. The region used for detection of potential rRNA novelty is a highly conserved portion seen in most known small rRNA genes that has been called by various names including the G530 loop. This region projects a retractable guanine base into the A site of the ribosome, a process that stabilizes tRNA:mRNA interactions and that seems to be an "invention" that predates the divergence of Archae, Bacteria, and Eukaryotes. Not all known SSU rRNAs have an loop that matches the criteria for this, but the cases where it has been lost appear to be relatively "recent", while its presence is sufficiently ancient to cover all three domains of life.

- The general question is "I have a(some) NGS datasets and want to know how rich the source material was in both known and unknown species.

- Two applications are envisioned for BarcodeRabbit

- 1. Provisional assessment of cellular phylogenetic composition that may provide a broader species assessment than standard k-mer tools

- 2. As an initial tool for discovery of potentially novel species/genus or larger phylogenetic groups

- Note that BarcodeRabbit shares considerable conceptual foundation with a number of other phylogenetic tools (e.g. Kraken), with

- its unique characteristics centered on the possibility of discovering novel cellular categories in sequencing datasets

- A further (major) footnote is that many different processes in molecular evolution and the experimental determination of sequences

- can produce sequence reads that would appear novel or otherwise of potential interest. Thus BarcodeRabbit is intended solely to nominate

- candidates rather than to produce output that can be directly interpreted (so expect further work computational and potentially experimental)

**->Syntax**

- python BarcodeRabbit<ver>.py Data=<file,file..> <Threads=# [e.g.8]>

**->Operation (brief intro)**

- BarcodeRabbit uses an ancillary program (VSG\_ModuleFV.py). This should be placed in the same directory that BarcodeRabbit is run from

- Two other files are needed (also in the same directory):

- i) a Reference directory of SSU rRNA 'barcode' sequences. Currently this is derived from Silva and is

**'Rabbit\_SSU\_Barcodes\_bb01\_040625.fasta.gz'**

- Other datasbases (ncbi, greengenes2, pr2) may have some advantages and formatted versions of these will be available 'soon'

- ii) a list of various illumina linkers 'illuminatetritis1223wMultiN.single.fa'

- The 'Threads' variable is optional to speed up analysis of large numbers of files (without providing this, BarcodeRabbit will run with a single core)

- Example:

- pypy3 BarcodeRabbit\_ca05\_073125.py data=''/Users/rabbit/media/datadrive/\*\*/\*.fastq.gz'

- This command will run BarcodeRabbit and analyze all \*.fastq.gz files in the indicated directory and any subdirectories

- And you'll need these four files in the directory where BarcodeRabbit is being run from (and where the output will appear)

- 'BarcodeRabbit\_ca05\_073125.py'

- 'VSG\_ModuleFV.py'

- 'illuminatetritis1223wMultiN.single.fa'

- 'Rabbit\_SSU\_Barcodes\_bb01\_040625.fasta.gz'

- 'bardwords.txt'

**->Output**

- The outputs of RabbitSSUBarcodeCounter are spreadsheets (.tsv files) with a list of data for barcode sequences that may be either

- Barcode\_segments from known SSU genes (with some identity information to indicate what species could have provided these

- Barcode\_segment-like regions that don't match any known SSU segment (as initially assessed using the Silva database used as input

- Many of the "novel" matches are actually not indicative of new species or genera. Some examples

- SSU rRNA segments that were not in the Silva database but are in other databases such as NCBI nr

- Various chimeras, sequencing artefacts, unannotated variants, and sequencing error/noise

- But there are novel rRNAs out there, and the program is a first-pass means to find them

- By default, two .tsv files are produced. One includes all SSU rDNA barcodes (novel or not) and one includes strong candidates for

novelty based on the provided database If there are no novel-rRNA candidates, the only file produced is the full list.

- If novel rRNA core candidate.

**- Output Columns:**

- 0 Barcode: the sequence of the (novel or known) barcode segment

- 1 Count: number of occurences in all datasets

- 2 SampleCounts: number of occurences by sample name

- 3 DataFileCounts: number of occurences by data file name

- 4 CoreSeq: sequence of the core segment used to originally nominate the barcode

- 5 CoreScore: mismatches between the core in this barcode and the canonical core (crw0)

- 6 UpsLenCount: the number of different upstream lengths observed in all data. If this number is small (1 or 2, there is limited diversity in the observed reads, which may be indicative of certain artefacts)

- 7 maxDwnLen: the longest downstream segment in any individual read. If this is small, the segment is always near the 3' end of the observed DNA/RNA fragment, somewhat raising concerns of a linker chimera.

- 8 Whole\_BestMatchScore: Differences between this barcode and the closest match in the reference dataset (Silva or other database). Single base snps contribute 1 to the score, indels 2

- 9 Whole\_BestMatchSeq: The sequence of the closest matching reference sequence

- 10 Whole\_BestMatchTaxa: This is just one example of a reference sequence amongst the 'best match' category (there may be others also)

- 11 Ups\_BestMatchScore: Best Match score for the sequence of length sups1 upstream of the core. Small values indicate likely close relationship to a known SSU; a low score (e.g. 0 or 1) here with a higher score in the WholeBestMatch score suggests a chimeric or truncated read, rather than a truly novel SSU segment

- 12 Dwn\_BestMatchScore: Best Match score for the sequence of length sdwn1 downstream of the core. Small values indicate likely close relationship to a known SSU; a low score (e.g. 0 or 1) here with a higher score in the WholeBestMatch score suggests a chimeric or truncated read, rather than a truly novel SSU segment

- 13 U2\_BestMatchScore: Best Match score for a relatively short sequence (sups2) just upstream of the core. A perfect or near perfect match here with poor matche by the full upstream sequence (col 11) indicates possible chimerism or artefact within the upstream flank.

- 14 D2\_BestMatchScore: Best Match score for a relatively short sequence (sdwn2) just downstream of the core. A perfect or near perfect match here with poor matche by the full upstream sequence (col 11) indicates possible chimerism or artefact within the upstream flank.

- 15 Conserved\_9\_Score: A metric indicating the number of bases in the core that match a consensus of 9 residues that are generally conserved between Eukaryotes, Bacteria, Archae, Mitochondria, and Chloroplasts. A low score here (<3) is strongly indicative of potential SSU rRNA character, while larger values may indicate either a non-rRNA locus that just happened to have the core or a very highly diverged rRNA SSU segment.

- 16 Clamp\_Affinity\_Metric: A metric indicating the quality of any potential duplex just outside the conserved 20-mer that "clamps" this sequence in the ribosome structure. Higher values (e.g.>20) seem strongly indicative of true ribosomal origin.

- 17 WashUSS\_Provisional: A diagram of possible secondary structure for a clamp, using the WashUSS conventions (see documentation for Infernal for definitions).

- 18 Clamp\_Percentile: A percentile of clamp quality (energy) amongst a set of sequences where bases outside the core-20 mer are randomly shuffled (default 1000 different shuffles) to yield a bootstrap comparision (higher number indicates greater significance of clamp structure).

- 19 High\_Novelty: A boolean flag indicating which sequences match the variety of flags set up to detect novelty amongst barcodes.

- 20 Bard\_Mnemonic: This is an experimental mnemoic derived from teh sequence and energy properties of each barcode. In some cases, similar barcodes will have similarities in their mnemonic. Words are from W.Shakespeare with some editing to attempt to avoid profanity.

- 21 AssemblyLocal: A very naive local assembly that may be of use in a blast search or other analysis. Not of quality expected for a program like Spades or Megahit-- this is merely to facilitate interim assessment.

**->Required Parameter** (just one! tell the program where to find your data files [fasta's or fastq;s with or without gzipping])

- **Direct FileName Option**: Data=<fasta/q file[s]> : Here you input one or more sequencing data files for analysis

- This can be a single file or a comma-delimited list and can include wildcards (\*)

- "Data" can also be a directory (in this case, all fasta/fastq/fastq.gz/fastq.gz in the directory will be analyzed).

- By default the sample name will be the filename and Rabbit "AutoPair" R1/R2 data file pairs as a single ('Rx') entity

- **SampleToData MetaFile Option**: For more flexibility, provide SampleName/ReadFileNames combinations in your own file

- This is invoked with SampleToData=<your SampleDescriptorFile> instead of Data= in command line

- SampleDescriptor files have simple line-by-line format where each line consists of

- A sample name, followed by tab, followed bya comma-delimited list of relevant the file paths with data from that sample

**->Optional Parameters**

- Although several different characteristics of the input handling and output can be varied, the program is effectively tuned with

- a single set of parameters. Detailed parameter lists are declared and defined in the first non-commented section of the code.

- NoNs=<False> : Setting this to true ignores any read with an N

- OutFile=<file name assigned by program if not specified> : Allows user to specify base filenames for data output

- For paired data output, separate R1 and R2 file names with a semicolon, e.g. OutFile='myoutfile\_R1.fastq;myoutfile\_R2.fastq'

- ReportGranularity=<100000> : How often to report progress (default every 100000 reads)

- MaxReadsPerFile=<0> : Max number to check for each substrate file (default is zero (all reads)

- GzipOut = <True/False>: Gzips the output (default is false, can also be set by providing an output file with extension '.gz')

- Threads = <number>: Running with multiple threads may be faster on a system where disk access is not limiting (default=1)

**->Additional Feature Options**

- \*\*\*Pair/Trimming Function-- Options to pair trim Reads of linker sequences

- Two types of trimming are used

- (a) To avoid reading into linkers, any read that contains the complement of the first k-mer (of len TrimFk1) from its paired read will be trimmed to remove all sequences after then end of that k-mer

- (b) Any read that contains a k-mer of len TrimTk1 matching a linker will be trimmed to remove all sequences from the start of that k-mer

- TrimFk=<16> : sets a value for k-mer length to avoid paired reads that extend into linker

- TrimTk=<13>: sets a value for k-mer to remove tetritis

- TrimTf=<illuminatetritis1223.fa> provides a FastA file name to extract k-mers for tetritis removal. (a file that can be used for this is available on the GITHub site)

**->Other Notes**

- All features should work on Linux/Mac/Windows.

- Requires Python 3.7+.

- The pypy interpreter (www.pypy.org) will speed up the program many-fold.