**ReadEVAluator** is a python program for analyzing paired end read data from illumina sequencing and providing genomewide and local assessments that quantitatively and qualitatively describe a run.

Minimal Command Line:

**python REVA\_xy11\_012120.py RefFile=<ReferenceGenome> R1File=<R1File> R2File=<R2File>**

**I. File Inputs for ReadEVAluator:**

**<ReferenceGenome>** sequence file in FastA format (needs .fa or .fasta extension and can be compressed as a .gz file). This is a list of sequences that are expected (or unexpected) in the dataset. Sequences can be in upper or lower case. Any character other than acgtAGCT will be ignored. As with any FastA file, the header line for each sequence (>mysequence) will be used an identifier (name) to refer to the following sequence. For convenience, the first word of the identifier (everything after the '>' and up to the first space) will be used as an abbreviated name to refer to the sequence in output. The default FastA file is "**ws220OP50PhiX.fa**" (for use with C. elegans sequencing data) containing chromosome sequences assigned in wormbase release ws220 (the most current complete genome as of 09/2016), along with an approximation of the E. coli OP50 sequence based on an incomplete list of OP50 snps published in 2009 entered into different E. coli B strain. The phi-X-174 sequence (used as a marker in some illumina experiments) is also included.

**<R1File> <R2File>**. The Illumina read files: fastq files that can be .gz compressed. They should retain "R1" and "R2" somewhere in their names (if not, you can use the explicit specification r1file=<R1File> and r2file=<R2File>). If they are equivalently named, you'll only need to specify one of the two files. You can also specify several files for R1 and or R2 and (if there are appropriate companion files), REVA will use all of them in order.

**<Feature\_File>**. A GTF or GFF file annotating introns, exons, genes, and other features. This allows the user to get a feature-by-feature output of hit numbers. Default is to take each feature in the list and calculate incidence of reads that start and end in that feature, as well as overall k-mer count for the dataset. Any file listed in the command line ending in gtf/gff/gff3 will be used in this way, or such a file can be specified with FeatureFile=<filename> in the command line. File 'Caenorhabditis\_elegans.WBcel235.96.gtf.gz' is included with the REVA distribution and may need to be put into the working directory if not found. More gtf files are available at the Ensemble website. Note that for these files, sequence counts are assumed to be 1-based (so sequence starts at position 1). k-mer count is based on k-mers for which >=50% reside in the indicated feature, while starts and ends are called only for starting and ending k-mer where the start of the k-mer is within the feature. For reads with some or all repeated sequences, the first k-mer matching the genome will be used to determine which features "own" the read. Reads (and k-mers) can be owned by as many features that contain them. Note that flags FeatureCategory=gene and FeatureCategory=exon limit the analyzed features to features that are named "gene" or "exon" respectively in the third column of the GFF/GTF table.

**<BaseByBase\_File>**. An input file describing regions of the genome where the user will obtain a list of match counts on a base-by-base level. A specific base range (e.g., i:1000-2000;ii:2500-3000) or a GFF or GTF file can be specified which indicates which areas (features) will be used for a base-by-base output. If not specified, the region covered by the BaseByBase output will default to all regions GTF/GFF file specified above and if that is not specified, the entire reference sequence will be flagged for base-by-base output (very long file if the reference is a complete genome of any complex organism)..

**<DFAM\_File>**. These are special repeat element annotation files for genomes that provide a repeat-by-repeat list of identified (known) repetitive elements. Generally the subset of elements indexed in such files is only a fraction of the repeats actually present in the genome. For those repeats, the assignments are (by nature) also somewhat ambiguous. Many repeated regions are related to each other and assignments can be arbitrary. REVA uses an assignment process that assigns each genomic region to the segment with the highest assigned evalue. This will often (but not always) be the largest repeat model that spans a given region. Further, the assignment of reads to repeats is dependent on a cutoff value that is set... DFAM assigns each instance of each repeat an evalue and some are very significant whereas others are not. Over-aggressive or under-aggressive repeat assignments could each result in both over- and under-counting individual repeat types. The default setting for MinRepeatEvalue is 10-6, but it can be changed using MinRepeatEvalue=<floating cutoff value>. Default is not to use DFAM file. Setting the flag -dfam looks for the file 'ce10\_dfam.hits.gz' if the reference sequence file is C. elegans build ws220 and 'hg38\_dfam.hits.gz' if the C. elegans build is hg38.

**II. Setting Additional Parameters (all optional and most have defaults-- generally only a few are needed for any given analysis; parameters in red are filenames that are generally set for each individual experiment [i.e., no clear default]):**

Settings can be assigned in the command line in two ways (Parameter names are case-insensitive):

a. **Using equals signs**: e.g., Klen=25 circlemax=2000

b. **Using *-parameter value* syntax:** e.g, *-Klen 25 -circlemax 2000*

**A. Specifying the Reference Genome**

**RefFile** This can be a fastA format file or simple text file with the reference sequence of one or more genomes. A zip archive with one or more references files is also allowed.

**ChrBase** Prefix needed to obtain a working UCSC browser link for a given chromosome. This character set should be the default "chr" for worm (ws220) or human (hg38) but may need to change for other species depending on how the UCSC browser is configured.

**CircularChromosomeLimit** (default 13500000) . Above this size all chromosomes are considered linear. This ensures that prokaryotic chromosomes and organelle genomes are considered circular and individual eukaryotic chromosomes are generally linear. This could be reconfigured in the code but is not particularly relevant for most analyses due to telomere repeats that obscure any mapping difference between circular and linear on eukaryotic chromosomes. Note as of version xf that the circularity of a given chromosomal segment can also be specified in the FastA file in the descriptor line for each segment, with the text "Topology=Linear" or "Topology=Circular".

**MultiplicityMax**. Sets maximum number of copies of a k-mer in the genome that is accounted for. Default is 63, which means that anything with a greater copy number is assigned the number "63 copes in the genome. No data is lost by keeping this at 63, one just loses distinction between 63 and >63 copies. Setting this value to a higher value (16383 or 2\*\*30-1 are the necommended next steps, provides greater granularity in examining very very high copy number sequences).

**IndexFile**. Can specify the premade index file if it is in another directory or does not have the expected name (see below). Default is to look for the IndexFile based on the name of the reference sequence. If such a file exists it will be used with no user intervention. So there is no reason to normally change or set this parameter unless the IndexFile is present but in a different directory.

I**ndexConstructionBins**. To avoid massive memory use during reference-sequence index construction at the start of the run, REVA can set up the index in stages using a binning approach that avoids undue memory use. REVA will try this automatically (default value of IndexConstructionBins is zero, which says to set using a basic algorithm). If memory errors appear during index construction, then it may be possible to avoid them by setting IndexConstructionBins to a higher value (up to 64 is feasible). This doesn't affect the eventual running of the program (either speeed or memory) and should not affect the results. Note that REVA will require a minimum level of memory to operate. For a k-mer length of 28, this is 20x(number of different k-mers in one strand of genome)+ 2-4gb. For the human genome, this should be about ~54GB of RAM. Approximately 3x of this operational ram would be required to set up the index with IndexConstructionBins==1 (no disk caching), thus necessitating the use of the disk cache on most systems for human DNA. For worm or other small genomes this can generally be set to 1 or zero.

**B. Specifying the read datasets and their characteristics**

**R1File** <Value is a fastq filename or file path, or a comma-delimited list of such names>. Any .fastq or any .fasta file with a single line per sequence read can be used for input (important for fasta files that each read be on a single line). If R1 somewhere in the name, REVA will automatically look for an equivalent R2 file. If for some reason your Read1 filename doesn't have "R1" in it, you will need can specify the name with "R1File=<myfile>". The files can be gzipped ('.gz file name expected).

Descriptors: An additional descriptor or the file can be added to each file specifications using an '#' indicator (e.g. R1File=MyReadFile\_R1 can specify the name of the R1 file for analysis, **R1File=MyReadFile\_R1#CeGenomic\_121719** adds a descriptor to output ('**CeGenomic\_121719'**) wherever the file is mentioned without changing the file name used by the program for analysis.

Wildcards: You can use a wildcard in a descriptor for files, such as R1File=\*.fasta (which will include all files in the current directory with the extension '.fasta'. If you use a wildcard, the Mnemonic for the summary output files will be your input with \* replaced by '\_all\_'. At the moment, only asterisks are accepted as wildcards.

Using a long list of input files organized into a file: For a long list of input files, you can use R1File=<Listing\_File.files>, e.g. R1File=MyReadFilesList.files (or MyReadFilesList.files.txt). REVA will parse the file into individual words, determine which of these words refer to possible input files, and open each as a datafile for mapping and counting. Entries can be one per line or any other fomat, with a # as above for each file providing an opportunity to add a descriptor (e.g. R1FileA.fastq.gz#MyFirstFile, R1FileB.fastq.gz#MySecondFile will input two files R1FileA.fastq.gz and R1FileB.fastq.gz and associate these with the mnemonics MyFirstFile and MySecondFile. There are two specific constraint for using such a "MetaFile" list, the MetaFile list file must end with ".files" or ".files.txt" and there can be no spaces, commas, number signs in any of the file names.

Direct use of NCBI Short Read Archive Data: If an R1File value is given with no '.fasta' or '.fastq' or '.'gz' extension, REVA will assume it is an NCBI SRR ID and will try to use fastq-dump (SRA tools) to download the data from NCBI in gzipped-fastA format into the temporary folder, where it will be renamed and used for analysis. This capability depends on the fastq-dump program being updated (recently downloaded from NCBI), on the program name being accessible in the current directory in the shell used to run python (may not be true with all IDEs), and on a functional internet connection. REVA will look for a copy of fastq-dump to do the download but this is very dependent on the operating system. To ensure that this works, you may need to explicitly tell REVA where the fastq-dump program file is with the command line item FastQDumpProgram=<full path to fastq-dump>.

**R2File** <Value is a fastq filename or file path>. In general REVA will impute R2 file names as best it can based on the Read1 file list. To do this, each Read1 file should have the character string "R1" exactly once, so that Reva can identify this and replace R1 with R2 to get the Read2 file name. Alternatively (and for a single R1 and R2 file as input) you can explicitly specify the R2 file by name. Any .fastq or .fastq filenames with R2 somewhere in the name can just be listed in the parameter list and REVA will identify this as your R2 read file. But in general it is most straightforward to specify an R1 file in the current directory where an equivalently named R2 file is in the same directory. REVA should then find the R2 file. So specifying the R2 file with "R2File=<myfile>" is only needed if for some reason your Read2 filename doesn't have "R2" in it, has more than one R2, or has other differences in location or name from the R1 file.

**Barcode** <Value is a string of bases [seq], Default is empty [no 5' barcode]> Looks for a 5' barcode sequence [seq] at the beginning of each R1 read and removes this. <bcseq>can be a barcode of any length. Setting <bcseq> to string of <len> N's (e.g. Barcode='NNN') will trim of the first <len> bases. Default is for barcoding to affect only the R1 read. Setting Barcode2, BarcodeRequire2 will affect the R2 read (also see Buffer5 and Buffer3 below).

**DeleteN** <Value is a boolean instructing REVA on what to do with N's in the read data. Default is to convert these arbitrarily to "G" residues, "DeleteN=True" deletes any Ns>.

**BarcodeRequire** <Value is True or False, Default is False>**.** Setting BarcodeRequire to True tells REVA to ignore any sequence without the barcode (otherwise sequences without the barcode are analyzed without trimming).

**Linker** <Value is a string of bases [seq], Default is empty [no linkers]> Looks for a linker starting with [seq]at the end of each read and removes this (does this as a search from the right (rsearch) so that the shortest unique sequence at the beginning of the linker can be used. Thus for the linker TGGAACCCGGAA, the sequence TGG is unique and Linker=TGG can be used to filter.

**LinkerRequire** <Value is True or False, Default is False>. Setting BarcodeRequire to True tells REVA to ignore any sequence without the linker (otherwise sequences without the linker are analyzed without trimming). Default is for barcoding to affect only the R1 read. Setting Linker2=, LinkerRequire2 will affect the R2 read (also see Buffer5 and Buffer3 below).

**R1Buffer5, R1Buffer3, R2Buffer5, R2Buffer3** <values are integers, defaults are zero>. These will remove a fixed length of sequence (zero is nothing, 10 is 10 bases, etc) on the 5' or 3' end of R1 and R2 respectively. This is to accomodate poor sequence quality in random primed regions in some protocols and in some cases to ignore fixed-length "unique molecular identifiers" that may be present on one end of other of a sequence.

**LinesToProcess**. <Value is an integer, Default is -1, which causes all lines of the fastq input files to be processed> Number of lines to process in the input files. Generally set to -1 (all lines) except for debugging or quick (partial) analysis of output. A value of 100000 processes 100,000 read pairs. Default is -1 (process all lines in fastq files).

**FivePrimeExtensionDisallowed** <Setting this to true will filter out any read/read-pair where the very first k-mer in R1 doesn't match the genome. Value is true or false, default is False [no filtering]. Setting this to true ignores any match that doesn't start at the first base of the read (barcode removal and trimming the 5' are done before carrying this out.

**StartHomology** <Setting this to a base sequence will filter out any match not starting that base sequence at the beginning of the homology. Value is a base sequence or the empty string, default is the empty string [no filtering]>. Example: setting this to 'G' ignores any match that doesn't start with a 'G' (barcode removal and trimming the 5' are done before carrying this out).

**MinHomology** <Setting this to a nonzero number sequence will filter out any match not starting with a homology of at least that length. Value is an integer, default is zero [no filtering]>. Example: setting this to 30 ignores any match that doesn't have at least 30 base match at the beginning of the matched region of R1 (barcode removal and trimming the 5' are done before carrying this out).

**MaxHomology** <Allows user to filter out any match starting with a homology above a certain length. Value is an integer, default is a very large number longer than any conceivable sequence read [999999999]= [no filtering]>. Example: setting this to 30 ignores any match that more than a 30 base match at the beginning of the matched region of R1 (barcode removal and trimming the 5' are done before carrying this out). Example and note: setting MinHomology=22, MaxHomology=22, StartHomology='G', FivePrimeExtensionDisallowed will only look at so-called '22G' sequence matches.

**C. Instructing REVA on details of how to locate reads and assign possible junctions**

**Klen** <K-mer length; value is an integer, default is 25>. This is the word-length that the program uses to position sequences. Generally a word length of 20 is sufficient to position a majority of unique sequences, but longer word lengths can be useful to position sequences with somewhat degererate character or in partly repetitive regions. Very long k-mer lengths will lose some assignments due to either errors or rearrangements. The default setting for Klen1 is 25 but it can be set as high as 32 with no reworking of the program.

**DomainLength** <A parameter that determines which repeats are focal and which are chromosome-wide; Values are integers; default value is 3000000 [=3MB]>. All instances of the k-mer must be within this limit and on the same chromosome for the repeat to be considered "**local**". Otherwise, the repeat is considered "**chromosomal**".

**bitsindexed** <K-mer length; value is an integer, default is set by REVA to optimize memory> This sets a parameter related to memory optimization in making the index. There should generally be no reason to manually specify this parameter (and changing it will make no difference for the eventual match counts and can make the assignments either very slow or very memory intensive), but the option is included because every sketchily documented and arcane piece of scientific software should have at least one completely obscure setting that allows the user to break the system at will.

**D. Instructing REVA on bin-by-bin, chromosome-by-chromosome, and feature-by-feature count output.** REVA will provide some basic summaries of read coverage that it compiles as it goes through individual reads. The default is to provide this on a chromosome-by-chromosome basis and a bin-by-bin basis, with the variable SeparationGranularity (default value = 10000) providing the bin length for display. In addition, providing REVA with a Feature file in GFF or (preferably) GTF format will yield an output of coverage statistics for individual features.

Several Parameters determine what is reported in the output for all three types of count summary:

**ReportStarts/ReportEnds/ReportCovering** <Report Starts, Ends , and Total K-mer Coverage; default is True>. Setting these to "True" (default) sets REVA to count starts (and ends and total coverage) of k-mers for each feature, bin, chromosome. Setting to "False" avoids this and minimizes output columns.

**ReportUnique/ReportRepeats** <instructs REVA whether to count uniquely mapping k-mers, repeated k-mers, or any combination thereof>. Values are boolean and default is True for both. Setting to "False" sets REVA to skip calculating and reporting for repeated (or unique) k-mers.

**ReportSense/ReportAntisense/ReportBothStrands** <instructs REVA whether to count k-mers, from sense-mapping reads, antisense-mapping reads, combine the two into a single value, or report all of the above. Values are boolean and default is to report a single summed value [ReportBothStrands]>. Setting ReportSense=True ReportAntisense=True ReportBothStrands=False sets REVA to separately report numbers or matches in sense-mapping and antisense-mapping reads. Note that for features, the orientation of the overall feature is taken into account so that for a reverse oriented feature "sense" counts are those on the negative strand of the larger chromosome.

**ReportReads/ReportPositions** <instructs REVA on the distinction between reporting the total number of reads that contain (or start or end with) a k-mer in a given region or reporting the total number of k-mer positions that are covered (or used as starts or ends). In the former case, the maximum value is essentially infinite, while in the latter case the maximum value is the total number of k-mers in the relevant interval. Sense and Antisense K-mers are (for this purpose) accounted for as distinct entities, so for a toally unique region of 10kb, the maximum value for ReportPositions is 20000. Both can be set to true, in which case both sets of values (number of positions and number of k-mers that match in the sequence data) are reported. Total count of matched k-mers is referred to as SummedKmerCoverage (the sum of total k-mer counts for every k-mer in the region), and CoveredPositions (the total number of Kmers in the region that are covered.

**FeatureFile** <Specifies a GFF/GTF file with feature locations in order to obtain a feature-by-feature set of match counts; value is a file name or path> This sets the file name for an optional GFF/GTF annotation file for a given genome-- feature positions in that file are taken literally as bins to allow feature-by-feature counting of matching reads. Each read (R1 or R2) is matched to a single k-mer- the first that can be uniquely mapped to the genome. Additional parameters that can (optionally) to be set to tailor GTF/GFF feature counting as follows. Providing the FeatureFile automatically yields a Feature-By-Feature output file

**FeatureCategory** <comma delimited list of feature types; value is a list of words [e.g. "exon,gene"] default is not specifying FeatureCategory, which yields counts for all categories>. This restricts reporting of features to only those features that are identified in column2 of the GFF/GTF feature file as one of the precise words in the comma-delimited list (no spaces). Shortcut: **-e or -g or -exon** or **-gene** are equivalent to 'FeatureCategory=exon' and 'FeatureCategory=gene'. These specific restrictions would limit the analysis of individual features to those identified as exons and genes respectively.

**FeatureTag** <comma delimited list of feature tags; value is a list of words [e.g. "ncRNA,tRNA"] default is not specifying FeatureTag, which yields counts for all features that match the FeatureCategory criteria above>. Setting this parameter tells REVA to look for a specific sequence tag anywhere in a gff/gtf line (e.g. "pseudogene" will limit the count output to gff/gtf lines with the string "pseudogene" somewhere in the line. FeatureTag can either be a single string to search for or a set of strings, delimited by commas (e.g. FeatureTag=pseudogene,piRNA yields any gff feature containing either "pseudogene" or "piRNA". (Note that phrases to search for and comma-delimited lists in the command line need to not have spaces in them, and that the searches are case insensitive)

**BinByBin** <Specifies an output of basic count information on a BinByBin basis using settings and parameters above; Default is True> Relevant settings are: **ReportStarts/ReportEnds/ReportCovering**, **ReportUnique/ReportRepeats**, **ReportSense/ReportAntisense/ReportBothStrands**,  **ReportReads/ReportPositions, and SeparationGranularity).**

**ChromosomeByChromosome** <Specifies an output of basic count information on a ChromosomeByChromosome basis using settings and parameters above; Default is True> Relevant settings are: **ReportStarts/ReportEnds/ReportCovering**, **ReportUnique/ReportRepeats**, **ReportSense/ReportAntisense/ReportBothStrands**,  **ReportReads/ReportPositions,.**

**E. Instructing REVA on detailed positional bin-by-bin event summary and rearrangement count output**

**SeparationGranularity** <Granularity of output (bin size); value is an integer, default is 10000>; A value of 1000 means that the program will output counts of hits for bins of 1000bp. So for the worm genome, this value splits the genome (100MB) into 100000 segments.

**KmerBinCoverageAll** <Tells REVA whether to provide bin-by-bin reports of repeated sequence mapping counts as well as unique position counts; value is True or False; default is True>. Number of k-mers in each bin is normally counted for local, chromosomal, and dispersed repeats in addition to unique k-mers present only once in genome. Setting KmerBinCoverageAll to False yields just the unique counts in the Positional Summary output, effectively removing a bunch of columns from the file.

**F. Instructing REVA on a output of hit frequencies and other characteristics for repeat regions ("DFAM" output)**

**DFAMFile** <Specifies a DFM file with feature locations in order to obtain a repeat-by-repeat set of match counts; value is a file name or path>This sets the file name for an optional annotation of known repeated sequences in the genome in a format that can be obtained from the DFAM database. Very importantly, these are only annotations on a small fraction of repeated sequences in a genome that have been characterized by the DFAM group. Providing the DFAM file automatically generates a repeat-by-repeat accounting of hit numbers in output. DFAM filenames are generally in the format "genome\_dfam.hits" (e.g. ce10\_dfam.hits or hg38\_dfam.hits) and can also be provided in .gzip format.

**G. Instructing REVA on an output of overall genome coverage ("Incidence" output)**

**FullCoverage** <Allows the user to specify whether an extensive set of coverage statistics will be calculated for all (unique and repetitive) kmers in the reference sequence; value is True or False, Default is True> Setting **FullCoverage =False** will save some computational time under certain circumstances, albeit with loss of some data for non-Unique counts.

**FullCoverageByFile**  <Tells REVA to provide a running total of all sequence categories (unique, local, chromosomal, and dispersed repeats) after each pair of read files is processed; Value is True or False; Default is "False" in which case the full coverage statistics are provided only once when the program has completed the entire set> "True" provides a summary after each file.

**FullCoverageInterval** <Tells REVA how often to calculate coverage statistics for the Incidence output; Value is an integer, defauolt is 0, which suppresses coverage estimates so they are only printed out at the end of the program (or after each file if FullCoverageByFile is set to True>. Setting to a positive value will produce interim reports in the incidence file (e.g. CoverageInterval=1000000 for calculate every million reads.

**BriefCoverage** <Instructs REVA how often to provide a less extensive (and less processor intensive) report of coverage, indicating how many unique k-mers are covered as the program progresses; Value is an integer; default is 200000>. The default sets REVA to provide the very basic interim coverage report every 200000 read pairs.

**CoverageAccumulationInterval**. <Gives the interval (how many read pairs) between interim calculations of coverage; Value is an integer; Default is 200000> This is used both by FullCoverage and BriefCoverage Routines. , which can be increased to provide less frequent updates (but will tie up memory if set to a much higher value, e.g., >50 million).

**CoverageDType**. By default, coverage values max out at 16-bit representation (**'uint16'**, recording coverage up to 65535. To decrease this (and free up a small amount of memory) this can be set to **'uint8'**. Conversely setting this to uint32 or uint64 would enable accurate reporting of sequences covered many mote times.

**H. Instructing REVA on an output of potential junctions ("Junction" output)**

**FindStrucutralAnomalies**. Instructs REVA to output a file (JunctionEvents....tdv) with a list of unexpected reads that are consistent with structural anomalies, particularly shedding of circles from a genome. Default is True. Setting this to "False" will somewhat speed up the program if the desired output doesn't include such rearrangements.

**RequireKMerOnly**. Instructs REVA to require only a single k-mer on each side of a junction to report a structural anomaly. Default is **RequireKMerOnly=True**, which reports all cases where the first and last uniquely mapping k-mer from a single read are not juxtaposed as expected in the reference genome. The (legacy) setting **RequireKMerOnly=False** only reports anomalies in which all bases before encountering a unique k-mer on either side perfectly match one end of the inferred structural variant. The **RequireKMerOnly=False** (legacy)setting becomes increasingly inappropriate if the reference genome has extensive mismatches with the actual sample, if the error rates increase toward 1/kMerLength, or if there are variants in imperfect tandem repeats. Note that **RequireKMerOnly=True** may need more post-REVA filtering to identify “true” variants.

**MatePairSubstitutionMax/SplitReadSubstitutionMax**. Sets number of substitutions allowed between actual read and reference in finding ends for SV calling and coverage estimations for Mate-Pair and Split-Read situations respectively. As of wn1\_112718, the default values for these are 1; default for earlier versions was 0.

**MatePairIndelLengthMax/SplitReadIndelLengthMax, MatePairIndelCountMax/SplitReadIndelCountMax**. Sets number and total length of indels allowed (read to reference) in aligning ends for SV calling and coverage estimation (1/3 should be okay for high quality/high coverage data on coding regions and unless a sequencing data set is particularly indel-prone). As of REVA wn1\_112718, the default values were 1 for number of indels 3 for max total length; defaults for earlier versions were zero.

**CircleMax/DeletionMax** (Maximum circle size/deletion sizes): These gives the maximum size of circular DNA detection and deletions that will be called by the variant finder. Default is set at 100kb for circles, 1MB for deletions. This affects both split-read detection and read-pair . Use a single value for both maximum in order to compare the two as controls for each other.

**CircleMin/InsertionMin** (Minimum circle/insertion size). As (2) but minimum size. Operationally this is used to avoid certain sequencing errors being inaccurately classified as evidence for DNA circles. Operationally a value of 10 (the default) should avoid any issues.

**Tn5DupMax**, **Tn5DupMin** . Maximum and minimum read overlap in looking for singly tagmented circles. Mechanistically, the values would be 12 and 8. Operationally these can be set to 16 and 0 respectively (which are the defaults), which will cast a wider net and allow a more specific measure of whether tagmentation events with the expected overhang (usually a 9bp overlap) are indeed overrepresented.

**ReadSeparationMax**. Largest fragment we expect to be able to capture and sequence on the flow cell. Fragments longer than this are examined as possible structural variants. Default is currently set as 3000bp (for Illumina platform)

**VerboseOutput** . If "True", causes the program to output numerous additional attributes of each potential structural variant (circle or deletion) including sequence reads and corresponding sequences in the reference genome. Makes the output file much larger but provides considerable extra information. Default is "True".

**Short** / **Long** . Cutoffs in total span for a read pair to be considered short, medium, or long. Default values: Short=100, Long=200. These cutoffs are inclusive on the short and long ends-- so setting short at 100 and long at 200 defines "Short" as anything up to 100 inclusive, "Long" as anything above 200 inclusive, and "Medium" as anything between 101 and 199 (inclusive).

**I. Instructing REVA on an output with base-by-base counts of matches, starts, ends, and 5' or 3' extensions**

**BaseByBaseRegionsFile** <default is main GFF/GTF if provided, the entire sequence if not> BaseByBase allows the option to output a single-base-resolution summary of match numbers and characteristics. A gff/gtf file can be used to specify the limits of the regions for single-base-level accounting, and this is recommended as a base-by-base output for the whole genome would be extremely large for even a modestly sized genome. **BaseByBaseRegionsFile** defaults to the main **FeatureFile** if this parameter is not provided (and to the entire genome if nothing is provided). GTF files are generally preferred over GFF/GFF3 but this is not an absolute preference and one may work where the other fails in any given case due to file format heterogeneity.

The following options further restrict the examined regions by chromosome, base-coordinates, feature category, and description tag.

**BaseByBaseCategory** <default is to not filter> Setting this to a value of a known feature type (e.g. 'gene' or 'exon') will restrict the BaseByBase output to only those features which have that designation (can be a comma delimited list of designations, allowing **any** in the list, e.g. 'exon,intron,gene')

**BaseByBaseTags** <default is to not filter> Setting this to a value (or a comma-delimited list of values) restricts the single-base-level output to only features where **any** of the indicated tags appear in the description line (sample tags 'ncRNA', 'kinase'), comparisons are not case sensitive.

**BaseByBasePositions** <default is all positions> Setting this will limit the single-base-output to the indicated chromosomes (can be a comma delimited list, format should be (e.g., BaseByBasePositions=i:1200-1299,ii:1400-1699 will limit output to chromosome i from position 1200 to 1299 and chromosome ii from 1400-1499 (inclusive)). Note these segments are in the format chromosome.position\_start-position\_end and can be comma separated (no spaces please).

Additional note/examples on the two filtering opetions for BaseByBase -- BaseByBaseCategories keeps only those records that contain one of the indicated words as the (complete) entry in column3 of the gff/gtf table (e.g. gene or exon).  BaseByBaseTags keeps only those records that contain an entry somewhere on the gff line

**BaseByBaseCategories=aardvark**

      use only gff/gtf lines where the third column in the line is explicity "aardvark"

**BaseByBaseCategories=aardvark,moose**

      use only gff/gtf lines where the third column in the line is explicity "aardvark" or "moose"

**BaseByBaseTags=aardvark**

      use only gff/gtf lines containing "aardvark" anywhere in the line

**BaseByBaseTags=aardvark,moose**

      use only gff/gtf lines containing "aardvark" or "moose" in the line

**BaseByBaseCategories=aardvark BaseByBaseTags=moose**

      use only gff/gtf lines where both hold (third column in the line is explicity "aardvark" and line contains "moose")

**BaseByBaseColumns** <default is all>. Setting this will select a defined set of columns for base-by-base output

**C** Chromosome Name

**P** Absolute position in chromosome (1 based)

**F** Feature Name

**O** Feature Orientation

**B** Base (A,G,C, or T). For output by feature, this will be on the + strand of feature, otherwise + strand of chromosome

**R** Nature of k-mer centered here (Unique,Local,Chromosomal,Dispersed[**U**,**C**,**L**,**D**], or **E**=(k-mer assigned elsewhere)

**M** Multiplicity character of k-mer centered on that base (zero if k-mer assigned elsewhere in the genome)

**K** Observation frequency of k-mer centered on that base (relative to feature+ if recording only for GFF features)

**S** Start frequency of k-mer centered on that base (relative to feature+ if recording only for GFF features)

**E** End frequency of k-mer centered on that base (relative to feature+ if recording only for GFF features)

**T** Unaligned base strings at start and end of sequence

e.g., *BaseByBaseColumns=FPBS*asks for output for Features [F], position in feature [P], base [B], starts [S]). Default: all values.

**MaxExtension** <default is 4> This sets the longest unaligned string at the beginning or end of an alignment that will be recorded amongst the extension list for that base in the BaseByBase analysis. Setting this to 4 (indeed the default) records extensions of <4 (e.g., "5'A", "3'AGAG") as is, but truncates longer extensions (e.g. "3'AGAGTT" would be truncated to "3'AGAG")

**CoverageSEMax** <default is 255> This sets the maximum number of reads starting at a given position that will be outputted in BaseByBase summary. CoverageSEMax values set below 64 are interpreted as number of bits reserved in memory (e.g. CoverageSEMax=16 uses a 16 bit word to store read counts, setting the maximum count to 2^16-1), CoverageSEMax values set above 64 are interpreted as an absolute limit (e.g. CoverageSEMax=65535 sets the count limit at 65535).

**KeepDoubleExtension** <default is True>This affects the reporting of 5' and 3' non-aligned extensions for single read datasets (not paired reads). Setting KeepDoubleExtension to True instructs REVA to not report sequences that have extensions on both ends. This avoids reporting certain partially aligning sequences from unannotated regions of the reference genome or from contaminating species.

**SnpFilter** <default is 2>this will filter out sequences that have an extended match followed by mismatch followed by match as likely not having extensions. Setting this to 2 will filter out any apparent extension (5' or 3') which has 2 or more matches following the first mismatch.

**J. Instructing REVA on a simple output with combinations of 5' and 3' ends**

**Combinations** <default is None, which suppresses output of a "combinations" file> Output a simple list of start-end combinations for paired end reads. Setting **Combinations=All** outputs a list of paired read positions for the entire input genome. Setting **Combinations=BaseByBase** only tallies read pairs where one read is in the BaseByBase regions. The latter will default to all **FeatureFile** regions if no BaseByBase range or file is specified. Setting **Combinations=Tn5** outputs a list of paired read positions that match the expectations of a singly-tagmented circle (paired ends in either unique or repetitive DNA with the two ends of a paired read oriented so that there is a 8-10b duplicated region present at the two ends. Note this will only detect sequences where some type of mapping has occured.

**K. Allowing REVA to generate links to the UCSC genome browser**

**UCSCLinks1:** <Value should be set to "True" if REVA is being instructed to attempt to generate UCSC links for each region>

**UCSC\_Assembly** <Value is the name of the UCSC database correpsonding to the assembly used> This just used for setting UCSC browser links> "Worm" or "Human" provides links to ce11 (worm) and human (hg38) by default, but other appelations can be used for other species for which there is a UCSC browser track.

**UCSCBuffer**. Sets the length of the buffer upstream of and downstream of a gene or other feature that is linked to on the UCSC browser (default=1kb).

**K. Some run-time settings for debugging and other uses**

**ReportGranularity**. How many sequence reads between progress checkins as the program runs. Default is 20000 reads.

**MetaData** <default is to include MetaData on run in every output file [MetaData=True]> Setting this to False will eliminate some but not all of the MetaData placements used by REVA to make the pedigree/history of REVA-derived files very clear.

**Debug** <default is non-debug mode> Invoking the Debug option sets REVA to stop reading the reference file after the first fastA entry (generally the first chromosome).

**III. The Intermediate (Index) file:**

A large file called "**REvalIndex\_**<details>**.9re**" holds a set of pre-sorted binary structures that allow rapid location of individual k-mers (effectively speeding up location of reads). If the file is not found, the program will (somewhat slowly) find it. The structure of the file name provides some context:

'REvalIndex\_'+RefAbbrev1+'\_k'+str(klen1)+'\_b'+str(BitsIndexed1)+'\_wv1.9re'

RefAbbrev1 is an abbreviation of the reference sequence file name, klen1 is the kmer length, BitsIndexed is an internal variable related to optimizing indexing (no need to reset this, but it needs to be consistent at "runtime" with the value that was used to construct the index). 'v1b' is the version (version 1 beta) and may be updated if the program adds different pre-calculated features to speed up operation. '9re' is an arbitrary file extension not currently used by any other program.

In constructing the index for large genomes, REVA creates a set of temporary files ('\*\_reva.tmp'). These should be deleted by REVA but if they for some reason aren't ,they can simply be deleted by the user from the operating system in batch. Note that index building may work more quickly from an SSD or M.2 drive, although the difference from a fast hard drive would be quite limited.

**IV. Output.**

**A.** A file with the prefix "**JunctionEvents\_**" provides a list of read pairs that indicate a likely junction in the DNA that corresponds either to a circularization event or an internal deletion. The criteria entail both mappability and consistency and the program is looking for precise sequences that evidence novel junctions (not partial matches). There is flexibility for sequence duplications, insertions, and deletions at the site of joining, but no flexibility in the "k-mer" sequences used for mapping (which must be unique, with two overlapping k-mers required for discordant read calls). This file is abbreviated under conditions where "Verbose=False" and contains many additional aspects of the designated sequence location assignments if "Verbose"=False.

Several kinds of events are "called" by the program

**Circle\_Junction** represents a split read (R1 or R2) which clearly/uniquely/perfectly maps at its start and end to points in the genome which indicate a circular permutation. The nature of the match to the genome is indicated by the suffix on this: as an example "1a1a" indicates that the junction is between k-mers both on the antisense strand from the R1 read. Similarly, a split read junction on the sense strand coming from the R2 read will have the suffix 2s2s. This call requires that both positions in the split read be defined by unique k-mers (ie nowhere else in the genome) somewhere between the beginning/end of the read and the provisional junction, with the entire read up to the end of the k-mer matching the genome precisely. The maximum and minimum circle sizes that will be "called" by the program are set as above (circlemin, circlemax), but note that a circle that is shorter than the readlength is unlikely and is more consistent with an insertion junction.

I**nsertion\_Junction** represents a split read which contains an apparent insertion within the read (generally short since the entire insertion and a unique k-mer on each side would need to be present for the call to be made.)

**Deletion\_Junction** represents a split read in R1 or R2 that indicates a deletion (with the lower and upper limits on deletion size being circlemin and circlemax)

**Circular\_Discordant** represents a pair of reads that each map perfectly to the reference but which are arranged discordantly (e.g. sense read downstream of antisense read). The program is set so both reads need to be positioned by at least two non-overlapping k-mers that are unique in the reference sequence. A "perfect" match to the genome throughout the individual reads. This stringency will certainly lose some instances of matches where there were point errors in sequencing or PCR, but more importantly will avoid counting other types of events that show up as "best matches" using conventional aligners but are in reality much more complex rearrangements induced during library production and/or amplification. This type of approach will be most effective when using highly accurate reads of moderate length (e.g. 75base paired ends). Only circles below the maximum size circlemax will be reported. Suffix on Circle\_Discordant indicates the nature of the reads (e.g., '1s2a' indicates an R1 in sense orientation with R2 in antisense orientation. To be circularly permuted in this case, R1 would need to be downstream of R2)

**Deletion\_Contraction.** Similar to Circle\_Discordant with the two reads mapping perfectly and uniquely (2 different k-mers) to the genome, but with the distance between the reads long enough to preclude a simple paired end situation. Assumes a maximum size for fragment amplification on the illumina flow cell of **ReadSeparationMax**, which is 3000 by default (but can be reset).

**Circle\_Contraction.** This is a specific and very precise event in which two reads that overlap at their start and that can be uniquely and precisely positioned (perfect match to the reference and with at least two non-overlapping k-mers that uniquely and consistently map each read of the pair). This configuration is indicative of a single tagmentation reaction of a circle that occurs at least (readlength) away from the circular junction in sufficiently unique DNA sequence to allow both read-halfs to align with the minimal two non-overlapping k-mers. For reasons not-entirely-clear to me, these events are almost vanishingly rare even in circular DNA populations.

**B.**  A file with the prefix "**IncidenceSummary\_**" provides a global and chromosome-by-chromosome list of specific event categories. The structure of each line in this file is "PropertyName Value Incidence(all\_RefSeq) Incidence(Individual\_RefSeqs)"

Properties listed by default are

BriefCoverage: A "running total" that shows how many different k-mers of a given class have been "covered" on one or both strands at a given point in the analysis. At the top of the IncidenceSummary file, these values are interim calculations (unless otherwise specified, just counting unique k-mers). Both number and percentage are given. As an example, a 1.25% value in coverage either strand means that 1.25% of all k-mer sequence in the target genome or segment have been covered on at least one of the two strands of at least one read. Numbers for both-strand coverage are also given. A very slight anomaly may occur in these calculations with even k-mer lengths; while not significant, the use of odd k-mer lengths is recommended if these are the key objectives in analyzing a particular sequencing run.

Multiplicity: How many read starts (the first k-mer in a read) are mappable uniquely and how many map to multiple sites in the reference file. Mutiplicities are "capped" in the current version of the program (MaxMultiplicity=63) so any k-mer with more than 63 copies in the genome is assigned to 63 copies. This can be modified by setting Multiplicity to a larger value as a program parameter. Operationally settng Multiplicity to anything above 63 will require a bit more memory (4 bytes instead of 2 bytes per position in sequence). The next useful point to set is 2\*\*14-1 (16383), with the next being 2\*\*30-1, etc. Setting Multiplicity=-1 will use a long (64 bit unsigned) integer to keep track of very high values.

Redundancy: How many well-mapped positional pairs (each read can be uniquely placed by at least two non-overlapping k-mers with the resulting positioning being consistent between the two k-mers) are represented by a single read pair and how many by multiple read pairs. Libraries that are substantially bottlenecked will be expected to show higher values for redundancy.

Coverage: What fraction of k-mers in the reference genome are covered in the present dataset. An extensive and unbiased sequence dataset would in principal cover every k-mer in the reference genome, with any deviation from the reference genome, inability to capture or sequence certain fragments, or incompleteness in the library (almost assured) yielding a fraction of uncovered k-mers. This value is reported separately for unique k-mers, k-mers that are repeated locally but not globally or in reverse, k-mers that are repeated but present only on a single chromosome, and k-mers that are dispersed on multiple chromosomes.

Separation: How many reads from each reference sequence are mappable to the sequence with the two ends separated by x bp.

**C.** A file with the prefix "**BinByBinEventSummary\_**" provides a position-by-position summary of located reads. Column outputs are as follows (see below for a detailed list of all 55 columns)

0: Chromosome

1: BinStart (Position of beginning of bin)

2: BinLength (usually SeparationGranularity but may be less in the last bin on each chromosome)

3. MappableSingleReads (two for a single read pair mapped on both sides).

4. FocalRepeatSingleReads (Focally repeated k-mers are defined as k-mers that occur multiple times in the genome but for which all occurrences are confined to a single chromosome in a limited range of base-pair distance (set by the "domain" variable and defaulted at 3MB). Each focal repeat k-mer is mapped to the first occurrence of that k-mer in the genome.

5. ChromosomeWideRepeatSingleReads ('ChromosomeWide' repeated k-mers are defined as k-mers that occur multiple times in a chromosome separated at the ends by a distance greater than the "domain" variable, but for which all occurrences are confined to a single chromosome. Each such repeat k-mer is mapped to the first occurrence of that k-mer in the chromosome.

6. DispersedRepeatSingleReads (Dispersed repeats are repeated k-mer that fails the "focal repeat" test. In addition to truly dispersed k-mers, this will also include any focally-repeated k-mer that is present at several different chromosomal loci, and any focally repeated chromosome whose "domain" spans a greater distance on a single chromosome than the "domain" variable allows.

7. UniqueWellPositionedReadPairSpecies: This value counts the number of different read pair start/stop positional combinations for which both reads are uniquely and unambiguosly mapped by k-mers to the reference genome. This value is "collapsed (while values in columns 2,3,4 are not collapsed) in the sense that two reads with the same pair of ends (R1 and R2) will be counted for this column) and the following columns just once.

8. MeanSepUniqueWellPositionedPairs: The mean separation between uniquely mapping read pairs in a given bin..

9. StdDSepUniqueWellPositionedPairs: The standard deviation corresponding to the above mean.

10. UniqueShortFragReadPairSpecies. Pairs of reads that are uniquely mappable and indicate a short captured fragment. (length of inferred captured fragment <=**Short**).

11. UniqueMedFragReadPairSpecies\_Len. Pairs of reads that are uniquely mappable and indicate a medium captured fragment. (R1-R2 span >**Short** but <**Long**)

12. UniqueLongFragReadPairSpecies\_Len. Pairs of reads that are uniquely mappable and indicate a medium captured fragment. (R1-R2 span >=**Long**)

13. Events that appear to be single-tagmentation of a short circle, producing reads that overlap at their base by 9bp

14. Bin-by-bin measurements of coverage (total k-mers of each category in each bin and total that were observed in the sample (on each strand) as well as counts of local, chromosomal, and dispersed repeat classes where the first matching kmer is in the relevant bin. The latter metrics require setting the flag KmerBinCoverageAll.

**D.**  A file with the prefix "**ChromosomeByChromosomeReadCountSummary\_**" provides a list of counts for each bin of k-mers in the reference sequence and number of matches in the read data.

**E.**  A file with the prefix "**BinByBinReadCountSummary\_**" provides a list of counts for each bin of k-mers in the reference sequence and number of matches in the read data. Some overlap with file F above in specific metrics, this file is provided for consistency with chromosome-by-chromosome and feature-by-feature read counts, and for use with PreREVA.

**F.**  A file with the prefix "**LogSummary\_**" provides a summary of the program as it was run including the version and code and a defining list of the parameters and identifying properties of the Illumina and Reference files. Even if this information in the Log is not immediately used (in general it will not be), it is **highly** recommended that these files be kept with the data generated by the program.

**G.** An optional output file **"FeatureSummary\_"** gives individual hits on a feature-by-feature basis, e.g. for individual exons. You will need a starting list of features (e.g., a gtf file, which can be gzipped]) which will be copied into the FeatureByFeature file with additional columns showing k-mer coverage in the dataset. Counts for sense and antisense k-mers, starts and ends for unique and repeated subregions of each feature are provided. Note that sense and antisense counts refer to the orientation of the feature (not of the whole genome). So sense counts for a "+" oriented feature are counts on the sense strand of the larger chromosome, but sense counts for a "-" oriented feature are counts on the antisense strand. The program does no checking on orientation (or anything else) from the gff/gtf files-- it just uses the coordinants and positions (which it assumes are 1-based and inclusive). Thus having a properly constituted GFF/GTF file is critical.

**H.** An optional output file with the prefix **"BaseByBase"** gives individual counts of matches for start, end, and internal k-mer coverage on a base-by-base basis, generally for just a fraction of the original chromosomal territory.

**I.** An optional output file **"DFAM\_RepeatSummary"** gives individual coverage and counts on a repeat-by-repeat basis.

**J.** An optional output file **"Combinations"** gives a list of start-end pairs encountered by REVA in a dataset. For a paired-end dataset, this represents pairs of start alignments for the R1 and R2 reads. For single-read datasets, this represents combinations of 5' and 3' ends.

**K.** PreREVA produces fastq files which have been filtered to contain only rearrangement candidates. These have names that start with the characters PreREVA and end with .fastq. In between is the original fastq file name (in lower case) plus a run-specific sequence of characters as an identifier. PreREVA also produces counts

**V. Environments: Python versus IronPython versus PyPy:**

The program will run from any version of Python that can import a reasonably complete version of the numerical analysis package Numpy. HYPERLINK "https://www.continuum.io/downloads" Anaconda (Continuum Analytics) and Canopy (Enthought) are good sources for Python distributions with numpy. Building from scratch from the standard python and numpy distributions is also possible but in general not needed. The current version of REVA will run with the accelerated "PyPy" version of python although speed is not improved (and may be degraded). PreREVA is a variant of REVA designed to minimize overhead and take advantage of the potential speeding up of PyPy. For simple counting of mapped reads and identification of rearrangements, PreREVA run with PyPy can be up to 100x faaster than REVA. To run REVA or PreREVA with PyPy you will need to have Numpy installed as a PyPy Module. Here are the current (1/20) instructions for this. (note that there have been two parallel Numpy packages for PyPy, a "Cython" version and a more native but somewhat limited "NumPyPy" version. Both work, but the Cython version is very slow. The following procedure (adapted from the Numpy website; http://pypy.org/download.html) instally Numpypy on a linux or mac system

1. Download the binary package (for linux or mac) from the pypy website. This is currently just a compressed folder with all of the executable code to run pypy in a folder and in folders contained within.

2. Uncompress the folder using the Mac or Linux archive manager (double clicking on the compressed file should be sufficient for that).

3. Move the folder to somewhere convenient (e.g. applications folder on mac or Desktop directory on Mac or Linux)

4. Open a terminal and navigate first to the folder you've just placed, then to the 'bin' directory under this folder

5. Do the following commands (you may need to install "git" first, (which would be sudo apt-get install git on linux)

./pypy -m ensurepip

./pypy -m pip install <git+https://bitbucket.org/pypy/numpy.git>

6. To allow easy access to the pypy program from any directory in the terminal, add a link to the relevant pypy file as one of your preferred paths. On a Mac, this is done as follows

edit ~/.bash\_profile

*add* the line

export PATH="/Applications/pypy2-v5.4.1-osx64/bin/:$PATH"

to the end of this file (you will need to change the version <v5.4.1-osx64> to the relevant directory name used for the pypy folder you just created)

7. Restart the terminal

**VI. Packing List:**

A. The python code (REVA\_ xy11\_012120.py).

B. A FastA reference file ws220OP50PhiX.fa carrying the ws220 worm genome with approximations of OP50 and PhiX. This is a "gzipped" file and may need to be ungzipped on some systems (gunzip <filename>). Also provided: a similar ws235 version.

C. Two test read files in nearly-FASTQ format (TestSVsUnc13R1.fastq.gz, TestSVsUnc13R2.fastq.gz). These allow a quick check to see if the program is functioning (six novel junctions should be identified in the ws220OP50PhiX reference file).

D. A script (cPlotSeparation<version>.py) that plots the Separation lengths identified by paired uniquely mapping reads in the resulting IncidenceSummary file.

E. A script (ManhattanProfile<version>.py) that plots the incidence of unique and mappable repetitive individual sequence tags based on their position along the genome.

Running the two scripts above (ManhattanProfile and cPlotSeparation) can be done either specifying the relevant REVal output file in the command line (e.g. *python ManhattanProfile101316.py PositionalSummary<xxxxx>.tdv*) will simply draw the plot. *python ManhattanProfile101316.py* will bring up a dialog which asks the user to locate the relevant data file ("PositionalSummary<xxxxx>.tdv")

F. The Script VSG\_Module, which is used for graphic capabilities by the referenced plotting script

G. Two files for parsing hits by worm exon: Caenorhabditis\_elegans. Caenorhabditis\_elegans.WBcel235.96.gtf.gz and c\_elegans.WS235.geneIDs.txt

G. Repeat strength and positional assignment lists from the DFAM group (ce10\_dfam.hits and hg38\_dfam.hits [Note the hg38 file was sufficiently large that it will need to be downloaded separately).

H. This documentation File (REVA\_Docs\_xr5\_100319.docx)

I. An experimental script for parsing Junctions, identifying good candidates for individual reads that are bone-fide DNA junctions identified from illumina sequencing. This script does an automated alignment and audit of the read with the putative left and right junctions, determining whether an individual candidate junction has good alignments to the putative right and left parent fragments, and also has a sufficient number of mismatches in regions of common homology to be distinguished from various alignment artefacts. Experimental.

J. PreREVA, a variant of REVA that prefilters large read sets to obtain only rearrangements and some count data on non-rearranged reads. PreREVA uses it's own index, but otherwise has the same input parameters as REVA. It is much faster when run with a PyPy interpreter. In addition to a subset of relevant REVA output files, PreREVA produces a new set of FastA or FastQ files that contain only reads that are candidates for rearrangement or other structural variant. These FastQ files can then be run with identical parameters using REVA. For multifile input, PreREVA FastA and FastQ output maintains a history of the File Name and read number for the original read file. This is in the descriptor line (first line of the block) for both FastA and FastQ files.

**VII. Bug List and History:**

A. In some cases if there is a crash during its assembly, the index file can be corrupted and need to be deleted. If you are getting odd errors, try deleting the existing .9re files and rerunning the program (this will reassemble the index files). If this is still problematic, try running the program once with Python (not pypy), using the included dummy file (TestSVsUnc13R1.fastq.gz) as the fastq input.

B. 11/27/18 (version wn). This is a very "beta" version of REVA with many recent changes, including the allowance of incomplete matches upstream of an anchoring k-mer match. Please record and report any bugs (log files are useful for this!).

C. 2/24/19 (version wu). Fixed a bug that caused the program to hang with a very GC rich k-mer

D. 2/26/19 (version wv). Memory for Multiplicity, Local, Chromosomal, and Dispersed indications is incorporated into a single array, saving a reasonable amount of RAM. Default copy number max for default accounting went from 255 to 63, meaning somewhat less granularity in reporting very high-copy-number k-mers in reference sequence (but no data is lost, just reports copy numbers at 63 for anything >63. MultiplicityMax can now be set to a greater value, using a bit more memory. Note that index files now have a different identifier and are not back-compatible (nor are old index files usable). REVA will detect this and assemble new index files as needed. 'N' Nucleotides are now excluded in searching for k-mers as are all 'N'-containing k-mers in the reference genome. Note that numbering is preserved while doing this. Added an option to segment the construction of the REVA index, saving memory. This should allow REVA to work with the human genome on a 64GB workstation.

E. 4/29/19 (version wy). Switched to Ensembl format GTF files for formatting. Many thanks to Matt McCoy for suggestions.

F. 5/13/19 (version xa). Fixed a bug that crashed with blank lines in fasta input file and added metrics that keep track of k-mers in each bin. Many thanks to Massa Shoura and Nimit Jain for suggestions.

G. 06/01/19 (version xb). Fixed a bug that crashed with reads that formally map beyond the end of a sequence. Many thanks to Massa Shoura and Nelson Hall.

H. 06/20/19 (version xf). Fixed a bug that gave somewhat inaccurate counts of covered k-mers in incidence output (position-specific output was unaffected). Added more precise bookkeeping for coverage of circular DNAs, also added the ability to specify for any given DNA the topology in the descriptor line of the FastA file (with Topology=Linear or Topology=Circular).

I. 07/09/19 (version xk). Added general capacity to take an input GFF or GTF feature file and add columns to that file indicating k-mer coverage and start and end coverage, also added capability to specify 5' barcode and 3' linker for captured RNA datasets (e.g. small RNA). Adding an "N" barcode (e.g. NNNN) can be used to blindly trim the first few bases (eg. barcode=NNNN trims the first four bases from read1, Barcode2=NNNNN trims the first 5 bases from read2).

J. 07/23/19 (version xl). Fixed a bug in which multiple files resulted in accumulated positional counts added with each file (now done once after all files finished at end). This was a significant bug for any job aggregating multiple files as the aggregate instance counts had been added to the total for each file ending. Now corrected so only added once. Also set recording of repeat counts in PositionalSummary to default (rather than skipping of repeats counts as default).

K. 09/14/19 (version xq). Added base-by-base capabilities, a descriptor at the top of each output file with basic outline of experimental and computational input, and capabilities for a pairwise map of paired end reads. Also added sufixes to column headings that indicate the version of the reference sequence that they apply to (for values based solely on the reference sequence), or the sequence read [e.g. fastq] dataset (for data-derived values).

L. 10/03/19 (version xr). Added option to set a buffer sequence length on the 5' and/or 3' of each R1 or R2 read to accomodate unique molecular identifiers (UMIs) in sequencing data and to avoid ambiguity due to random hexamer priming in some libraries. Completing shift to allow full key-value user interface. Added options FivePrimeExtensionDisallowed, MinHomology, MaxHomology, StartHomology, which allow user to specify the nature of matches and filter for sequences with characteristics like 21U or 22G.

M. 10/16/19 (version xs). Fixed order of output for positional summary fields and a bug involving readfile names with the string R2 that are actually R1 (e.g. SRR2...).

N. 10/28/19 (version xt). REVA should now be compatible with any recent version of Python3. String formatting from Python2 is still used but this seems to be supported by Python3 as well.

O. 11/24/19 (version xu). Major bug fixes and clarifications for BaseByBase. xu4 adds the ability to set the highest coverage for base-by-base reads above 255 (to any number).

P. 12/17/19 (version xv). Added ability to annotate read file specifications using a '#' indicator (e.g. R1=MyReadFile\_R1 can specify the name of the R1 file for analysis, **R1=MyReadFile\_R1#ceGenomic\_121719** adds a descriptor to output ('ceGenomic\_121719) without changing the file name. Added ability to call SRR files from the NCBI short read archive directly-- these are temporarily downloaded in compressed form from NCBI. Added ability to use fasta and gzipped fasta files as read input.

Q. 01/05/20 (version xy). Added consistency to BinByBin and FeatureByFeature outputs, added ChromosomeByChromosome Output. Added consistency with PreReva filtering version (included). Additional changes: Renamed the earlier BinByBin Event summary, which is distinct from the BinByBin counts list (a merging tool for the two files will be forthcoming). Also corrected all outputs (as far as I know) to be 1-based rather than zero-based in numbering. Added an option "FirstKTable" that allows trouble shooting of Illumina hardware issues, providing the position of the first k-mer in each sequence that matches the genome. This value should be small for good sequencing reads (generally less than 3). If it is larger for a given region of the flowcell, there is a potential focusing problem at one or more synthesis cycles. This may also be evident in quality scores (or not). Default handling of N residues in sequence reads is now to convert them to "Gs". An option "DeleteNs" was added in case users wish to delete all N residues in reads (no effect on Reference genomes, for which Ns are converted to Gs.

**REVA Version xy11 (01/21/2020).** Code written by A. Fire with guidance from L. Hansen and experimental inputs from M. Shoura, N. Jain, M. McCoy, and L. Wahba, K. Artiles, N. Hall), Department of Pathology, Stanford University School of Medicine. Distributed with no guarantees of quality, efficacy, accuracy, or anything in life, let alone the functioning of software (**Copyright 2016-2020 Stanford**)

Appendix 1: Fields in Structural Variant Junction Summary File (tab delimited “spreadsheet”) with “Verbose” turned on.

0. EventTypes

1. EventChr

2. EventStartLo

3. EventStartHi

4. EventEndLo

5. EventEndHi

6. EventOvrelap

7. UCSC\_Link

8. r1

9. r1L\_Chromosome

10. r1L\_StartBase

11. r1L\_HomologyLen

12. r1L\_ReferenceSeq

13. r1R\_Chromosome

14. r1R\_StartBase

15. r1R\_HomologyLen

16. r1R\_ReferenceSeq

17. r1\_RtoL\_Distance\_On\_Genome

18. r1\_RtoL\_Homology\_Overlap

19. r2

20. r2L\_Chromosome

21. r2L\_StartBase

22. r2L\_HomologyLen

23. r2L\_ReferenceSeq

24. r2R\_Chromosome

25. r2R\_StartBase

26. r2R\_HomologyLen

27. r2R\_ReferenceSeq

28. r2\_RtoL\_Distance\_On\_Genome

29. r2\_RtoL\_Homology\_Overlap

30. ReadFileAndReadNumberInFile

Appendix 2: Positional Summary Fields (note that fields 37-61 can be skipped by setting KmerBinCoverageAll=False in command line)

1. Chr
2. BinStart
3. BinLength
4. MappableSingleReads
5. FocalRepeatSingleReads
6. ChromosomeLimitedRepeatSingleReads
7. DispersedRepeatSingleReads
8. MappableSingleReadSpecies(dedup)
9. FocalRepeatSingleReadSpecies(dedup)
10. ChromosomeLimitedRepeatSingleReadSpecies(dedup)
11. DispersedRepeatSingleReadSpecies(dedup)
12. MappableReadPairs
13. MappableReadPairSpecies(dedup)
14. DMR\_DualMappableReads(Dual-kmer-mapped)
15. DMRP\_DualMappableReadPairs(Dual-kmer-mapped)
16. DMRP\_DualMappableReadPairSpecies(Dual-kmer-mapped\_dedup)
17. MeanSepDMRP
18. StdDSepDMRP
19. UniqueShortFragReadPairSpecies\_Len<=+str(Short1)
20. UniqueMedFragReadPairSpecies\_Len<+str(Short1)+\_Len>+str(Long1)
21. UniqueLongFragReadPairSpecies\_Len>=+str(Long1)
22. UniqueSingleTagmentationCandidateReads
23. UniqueSingleTagmentationCandidateSpecies
24. Unique+RepeatedSingleTagmentationCandidateReads
25. Unique+RepeatedSingleTagmentationCandidateSpecies
26. G\_Bases
27. A\_Bases
28. T\_Bases
29. C\_Bases
30. N\_Bases
31. UniqueKNumber (All unique k-mers on the sense strand in this bin [whether observed or not])
32. UniqueKObservedS (Total number of unique k-mers in the bin observed in sense reads in fastq)
33. UniqueKObservedA (Total number of unique k-mers in the bin observed in antisense reads in fastq)
34. UniqueKCountS (S = sum of counts on sense strand)
35. UniqueKCountA (A= sum of counts on antisense strand)
36. UniqueKCountS2 (S2 = sum of squares of counts on sense strand)
37. UniqueKCountA2 (A2 = sum of squares of counts on antisense strand)
38. LocalKNumber (Total local-repeat k-mers where the first occurrence is in this bin)
39. LocalKMultiplicity (Total multiplicity for all local-repeat K-mers where the canonical [first] occurence is in this bin)
40. LocalKObservedS
41. LocalKObservedA
42. LocalKCountS
43. LocalKCountA
44. LocalKCountS2
45. LocalKCountA2
46. ChromosomalKNumber (Total chromosomal-repeat k-mers where the first occurrence is in this bin)
47. ChromosomalKMultiplicity (Total multiplicity for all chromosomal-repeat k-mers where the first occurrence is in this bin)
48. ChromosomalKObservedS
49. ChromosomalKObservedA
50. ChromosomalKCountS
51. ChromosomalKCountA
52. ChromosomalKCountS2
53. ChromosomalKCountA2
54. DispersedKNumber (Total dispersed-repeat k-mers where the first occurrence is in this bin)
55. DispersedKMultiplicity (Total multiplicity for all dispersed-repeat k-mers where the first occurrence is in this bin)
56. DispersedKObservedS
57. DispersedKObservedA
58. DispersedKCountS
59. DispersedKCountA
60. DispersedKCountS2
61. DispersedKCountA2
62. UCSCLink

Some additional command lines (work in progress)

Examining small RNA data for counts by bin and by gene, and for hits to ncRNA, and for base-by-base hits to a specified subset of loci (in this case rDNA loci in the file rRNABases.gtf). For this dataset each read has a 3' linker starting with the bases 'TGG' that don't appear downstream in the linker. The reads here were from VC2010-G6-17x\_S1\_L001\_R1\_001.fastq.gz and the reference file was ws235OP50PhiX.fa (worm plus OP50 plus phiX), while the features are from the standard GTF file Caenorhabditis\_elegans.WBcel235.96.gtf.gz (available from Ensembl) and only features labeled in that file as "gene" (in the second GTF column) and having the additional tag "ncRNA" somewhere in the GTF description are taken for counting.

python

REVA\_xv6\_121719\_Defn.py

RefFile=ws235OP50PhiX.fa

R1=VC2010-G6-17x\_S1\_L001\_R1\_001.fastq.gz

R2=VC2010-G6-17x\_S1\_L001\_R2\_001.fastq.gz

FeatureFile=Caenorhabditis\_elegans.WBcel235.96.gtf.gz

FeatureCategory=gene

klen=19

Linker=TGG

RequireLinker=True

BaseByBaseFile=rRNABases.gtf

LinesToProcess=200000

Some command line shortcuts:

1. Any .fastq or .fastq.gz file with "R1" in the name that is mentioned in the command line will be utilized as an input read file. If there is an equivalent "R2" file in the same directory, that will be used as well (so you need only specify one file of each read pair).

2. Any .fasta or .fa file (with or without zip or gz compression) will be treated as the reference input

3. Any .gff or .gtf file (with or without gz compression) will be treated as a feature input file

4. Any integer will be treated as a total read count after which to stop

5. Specifying any BaseByBase component (tag, category, positions, inputfile) will automatically turn on the BaseByBase feature

6. Specifying -gene or -exon will instruct REVA to only pay attention to the indicated feature category(ies)