Correcting reads with Blue (v2.1.4)

16th August 2018

Blue is a kMer-based DNA error correction tool. Blue v1 is described in *Blue: correcting sequencing errors using consensus and context*. *Bioinformatics. 2014;30:2723-32*. The significant difference between Blue v1 and Blue v2 is a change in intent, moving from reducing the number of errors in a set of reads to producing only correct reads. Blue v1 would correct whatever errors it could correct, and at times would leave uncorrected errors at the end of reads. The intent was to correct what could be safely corrected, and pass the resulting 'improved' reads on to other tools. The 'good' parameter in Blue v1 mitigated this behaviour somewhat by discarding reads with insufficient numbers of 'good' kMers after correction, ensuring that the most erroneous and uncorrectable reads were dropped rather than being passed on to the next tool in a pipeline.

Blue v2 tries to ensure that all the reads it are completely corrected, and uses tail trimming to remove uncorrectable parts of reads and drops any reads that appear to still contain errors after correction and trimming. Blue is quite averse to 'rewriting' reads. Incremental kMer-based algorithms can often infer which base should follow any given kMer, and such extended 'reads' can quickly diverge from the read actually being corrected. This produces chimeric 'corrected' reads whose start comes from the read being corrected and whose tail comes from elsewhere in the genome. Blue avoids this 'rewriting' by tracking the number of corrections being made to a read, and will abandon a read if too many 'corrections' are being made too close together. Blue will also abandon a read if reaches a point where there are no valid 'next kMers'. Blue v1 would pass on these abandoned but partially corrected reads, after checking that the read had sufficient good kMers. Blue v2 will trim such reads back to the last good kMer, and discard it if it is now too short (as defined by the repurposed 'good' parameter).

The overall result of this change, and numerous others, is much improved accuracy (in relative terms). On the E. coli DH10B dataset referred to in the paper, Blue now has 99.98% of the corrected (and possibly trimmed) reads aligning with zero changes against the reference sequence, up from 99.90% previously (with –good 80 used in both cases).

Blue and Tessel are part of the WorkingDogs pack of kMer-based bioinformatics tools. They can be downloaded from https://github.com/PaulGreenfieldOz/WorkingDogs.

Blue and Tessel are written in C#. This means that they will run natively on Windows, various flavours of Linux and on OSX. They can also be run under *mono* on these non-Windows platforms. The non-Windows 'native' code files are produced through the use of mkbundle (part of the mono-dev package). Instructions on compiling these programs can be found in the appropriate README.md files.

Tessel

Tessel is a fast, scalable kMer tiling program that creates the kMer sets (and kMer pairs) used by Blue. It takes any number of sequence data files (fasta or fastq), tiles them for kMers, and writes a set of the distinct kMers present in the reads files, and how many times each kMer was found.

Tessel usually writes these kMers+counts to a '.cbt' file that contains binary (2-bits/base) canonical kMers. Canonical here means that only the lexicographically lowest of a kMer and its reverse complement is kept. This is useful for real sequence data where reads can come from either the forward or reverse strand, and saves programs that want to look for a kMer in a kMer set from having to check for both its as-read and reverse-complement forms. It is also possible to get Tessel to only track 'as-read' kMers, rather than

converting them to canonical form, and this is useful when tiling genomes and assembled contigs. Tessel also supports a number of other text-based output formats, both for debugging and for compatibility with JellyFish.

Tessel keeps two counts for each kMers - a separate count for both the kMer as it appears in the hash table and its reverse-complement. Keeping these two counts separate lets Blue check for unbalanced kMers/reads, often a sign of sequencing errors or remnant adapter sequences.

As of v2.1, Tessel has subsumed the GenerateMerPairs program and, by default, will also tile the sequence data reads for kMer 'pairs'. These are pairs of short kMers, separated by a gap, and are effectively longer kMers. They are used by Blue to partially resolve the ambiguity that comes from using shorter kMers.

Tessel is a command-line program with the following cryptic usage hint:

```
Tessel -k kMerLength -g genomeLength [-t #threads] [-tmp tempDir]
       [-min minCount] [-trim nn] [-trimQual nn] [-s]
       [-pairs] [-nopairs] [-canonical|asread]
       [-text textFN] [-textFormat pairs|sum|faPairs|faSum]
       cbtName readsFNs... or readsPattern
```

A typical run of Tessel is something like…

```
Tessel -k 25 -g 10000000 -t 16 -tmp /tmp/DH10B -min 2 -trimqual 30 DH10B
       MiSeq_Ecoli_DH10B_110721_PF_R?_paired.fastq
```

…which will tile whatever files match MiSeq_Ecoli_DH10B_110721_PF_R?_paired.fastq for 25-mers and write out a file of all the distinct canonical kMers found and their counts. The kMers will be written to DH10B_25.cbt. Tessel produces a number of temporary files and these will be written to the /tmp/DH10B directory. Reads will be tail-trimmed before tiling, using a qual score of 30 as the acceptable-quality level. Tessel will also tile the reads again for kMer pairs, and these will be written to DH10B_25.prs.

The full set of Tessel parameters is:

Option	Parameter	Description
-h -help		Writes a summary of the options and parameters available with Tessel, and then exits.
-k	*kMerSize*	Length of the kMers to be generated. 'k' Should typically be >=20 and must be <=32. The default is k=25.
-g -genome	*genomeSize*	A guess at the total size of the final (assembled) genome or genomes in the sample. This is used for the initial sizing of some arrays and any rough guess will probably be OK. Default is 10,000,000 which will handle bacterial genomes quite well.
-t -threads	*no_of_threads*	Number of parallel tiling threads. Tessel scales well with the number of threads, up to the point where it is limited by reading and writing the files. Default is 1.
-tmp	*tmpDir*	Temporary files are written to this directory. This can be used to improve performance by pushing these temporary files to the fastest available storage. This is especially helpful with remote HDD storage and less important with local SSDs. Default is to write these temp files to the current directory.

-m -min	*minCount*	kMers found fewer than minCount times will be dropped when writing the final kMer file. Useful for reducing the size of the kMer tables, especially when Blue will probably discard low abundance kMers while loading the kMer tables. The default is to discard singletons (-min 2).
-trim	*trimLength*	The tails of all reads will be trimmed by this length before tiling.-trimQual is a better tail-trimming technique in most cases. Default is no trimming.
-tq -trimQual	*min_qual_value*	Tail-trims each read prior to tiling and reduces the number of kMers coming from the error-prone read tails. Uses a sliding window to cut back the tails of reads until the specified quality level is reached. Default is no trimming.
-s		Recursively search through sub-directories for matching file names. Default is to only look in the current directory if no explicit directory name is given as part of the readsFN parameter.
-pairs		Tile the reads for paired short kMers. This option replaces the use of GenerateMerPairs the Blue v1 pipeline. This is the default setting.
-nopairs		Do not tile the reads for kMer pairs.
-canonical		Regard kMers and their reverse complements as equivalent. Separate counts are maintained for both forms of each kMer. Default option. Causes a .cbt file to be used for the binary kMers+counts.
-asread		kMers are saved in the form that they are found, and not converted into a canonical form. This option is intended only for use on contigs/genomes. Causes a .abt file to be used for the binary kMers+counts.
-text *textFN*		The following formats tell Tessel how to write out the kMers and their counts to a text file, as well as in binary format (a .cbt or .abt file). The format of the text file is specified by the-textFormat option
-textFormat		Specifies the format used when writing kMers+counts to the text file. The 'sum' and 'faSum' options are compatible with Jellyfish text file formats.
	Pairs	One line per kMer: kMer (tab) count (tab) rc-count. This is the default text format.
	Sum	One line per kMer: kMer (tab) summed-count.
	faPairs	FASTA file. Header is >count rcCount. Sequence line is kMer.
	faSum	FASTA file. Header is >summed-count. Sequence line is kMer.
Cbtname		This parameter is used to construct the output file names. The binary-formatted tiled kMers+counts will be written to cbtName_kmerLength.cbt (e.g. DH10B_25.cbt) and the kMer repetition depth histogram will be saved as cbtName_kmerLength_histo.txt (e.g. DH10B_25_histo.txt). If this parameter is simply a name, such as 'DH10B', then the output files will be placed in the current directory. If it includes a directory, such as 'Healed/DH10B', then the file will be written to this directory. If the '-asread' option is used, a '.abt' file suffix is used instead of '.cbt'. The 'pairs' file, if requested' is written using the same file name, but with a suffix of '.prs'.
readsFNs readsFNP		List of file names/patterns of reads to be filtered. E.g. MiSeq_Ecoli_DH10B_110721_PF_R?_paired.fastq. There can be any number of these files names or file name patterns.

Canonical binary k-mers are written to a '.cbt' file, and using the '-asread' option causes them to be written to a '.abt' file. The format of these files is:

Int32 'k' length used. The top 8 bits of this Int32 are reserved for use as a file format marker (currently 0).
Sets of {uint64, int32, int32} triplets, one for each k-mer.
The uint64 value is a kMer in packed binary format.
Two bits per base (A=00, C=01, G=10, T=11).
kMers are left-adjusted within the 64-bit word.
A kMer can be no bigger than 32 bases long.
The first int32 is the number of times the corresponding kMer was seen.
The next int32 is the number of times it was seen in its reverse complement form.

The 'pairs' file format is similar:
Int32 'gap' in bases between the pair of 16-mers making up the 'pair'. The top 8 bits of this Int32 are reserved for use as a file format marker (currently 0).
Sets of {uint64, int32} pairs, one for each kMer pair.
The uint64 value is a kMer pair in packed binary format.
Top 32-bits are the first 16-mer,
Low 32-bits are the second 16S-mer in the pair.
The int32 is the number of times the corresponding kMer pair was seen.

Blue

Once you have a set of kMers (a .cbt file), and (preferably) a set of corresponding kMer pairs (a .prs file), you can go ahead and correct your reads. Blue is a command-line program with the following cryptic usage hint:

```
Blue [-help] [-r run] –m minReps [-hp] [-t threads] [-l length] [-fixed]
       [-variable] [-good nn%] [-problems] [-extend nn] [-output dir]
       [-paired] [-unpaired] cbtFN readsFNs or patterns
```

A typical run of Blue is something like…

```
Blue –r hv80 –m 20 –t 16 –g 80 –mq 25 –v DH10B_25.cbt
       MiSeq_Ecoli_DH10B_110721_PF_R?_paired.fq
```

… which will correct the pair of reads files that matched MiSeq_Ecoli_DH10B_110721_PF_R?_paired.fq, writing corrected reads to MiSeq_Ecoli_DH10B_110721_PF_R1_paired_hv80.fq and MiSeq_Ecoli_DH10B_110721_PF_R2_paired_hv80.fq, using the kMers found in DH10B_25.cbt and the pairs from DH10B_25.prs. The read pairing in the starting reads will be maintained in the corrected reads. The expected minimum depth for a good read is set to 20. Corrected reads will be trimmed back to the final 'good' kMer (if the read cannot be completely corrected), and the trimmed reads must be at least 80% of their starting length (no more than 20% of the read can be trimmed away). The reads will be left at their trimmed length (if they were trimmed). The 'error prone tail' of the region will be estimated using a qual score of 25.

The full set of Blue parameters is:

Option	Parameter	Description
-r -run	*runName*	This is the name of this healing run. It is appended to the name of each of the reads files to produce the name of the corresponding file of corrected reads. For example, with '-r h' the corrected form of 'ERR022075_1.fastq' will be 'ERR022075_1_h.fastq'. The default value is 'corrected_<minReps>'.
-s -stats	*statsFN*	This is the name of the file to be used to save the statistics from this run of Blue. Blue will generate a default statistics file from the first reads file name parameter and the runName, e.g. ERR022075_h_stats.txt.
-t -threads	*noOfThreads*	This option says is how many parallel threads to use for the tiling. The default is to use just 2 threads. Blue scales well with the number of threads, but will eventually be limited by the time needed to read and write the sequence files. There is also some per-thread memory allocations, so using more threads will also use more memory.
-m -min	*minReps*	This is the minimum kMer repetition depth expected in good kMers and is used (rarely) to distinguish between 'good' and 'bad' kMers. In general, this should be set to somewhere in the dip between the LHS error spike and the start of the Poisson curve derived from the good reads (more on this later). This value isn't very critical as the good/poor/bad depths are calculated dynamically for all reads if they contain any 'good' kMers at all.
-max	*maxReps*	Reads with an average depth of coverage above this level will not be corrected. This rarely useful option was added to better handle a metagenomic dataset with a very deep-coverage repeat region that was best ignored during correction and assembly.
-l -length	*maxLength*	Reads are trimmed to this length before correction. This option may be of use when correcting poor quality data with very noisy tails, such as some MiSeq 300-mer reads. The minQual parameter may also prove to be useful with this type of noisy data. The default is to do no length-based trimming.
`-mq` `-minqual`	*minQual*	Sets the minimum qual score to be used when estimating the starting point of the noisy tail in each read. Once Blue reaches this point when correcting a read, it stops trying quite so hard, reducing the recursive depth allowed during tree exploration and only checking for substitution errors for Illumina-style data. The effect of this option is to allow Blue to successfully correct more of the most error-ridden reads. The default value is a qual score of 30.
-hp		The –hp option tells Blue to try even harder to when finding errors that could be corrected. There are a number of tests done at every base position, all based on depth of coverage. These tests will pick up random indel errors, but indels are so common at the end of homopolymer runs in 454 and IonTorrent data that multiple different hp run lengths all look to be OK as well. For example, if our genome had AAAAAA then with Illumina data this is what we'd see almost all the time. With 454-like data, we'd probably get 5 or 7 As as frequently as 6 As, so depth of coverage test would indicate that none of them were errors. The –hp flag causes Blue to look for the end of hp runs and forces an attempt at correction at that point, regardless of depth of coverage. Blue will always attempt to detect and correct indel errors regardless of the setting of this option, and

		setting –hp has a performance cost so it should not be set for Illumina data.
-g -good	*%length*	The meaning of this parameter has changed in Blue v2. It now specifies the minimum read length required after correction and trimming (and possible extension). The default value is 70 which means that corrected reads must be at least 70% of their original length after trimming. Blue maintains the pairedness of its input files, so if one read of a pair fails the 'good' test, both the failing read and its mate will be dropped (but written to a 'singles' file in case you want to use them). Those reads that fail this minimum length test will be written to a 'problems' file if you have set the '-problems' option.
-o -output	*outputDir*	This specifies the output directory where Blue will write all its files, including the corrected reads. By default, these files are written to the directory where the reads were found.
-problems		Asks Blue to save any uncorrectable reads in a '_problems' file for later examination.
-v -variable		With Blue v2, read lengths are allowed to change during correction as a result of insertions, deletions and the trimming of uncorrected read tails. By default, reads are written just as they are at the end of correction/trimming, and so can vary in length.
-fixed		This option asks Blue to try to extend any trimmed reads to get them closer to their starting lengths. This extension works by starting at the end of a trimmed read and trying to extend it unambiguously, one base at a time. The extension process is stopped if two or more viable 'next' kMers are found at any point. This extended length is the one that is checked against the minimum length set through the 'good' option.
-fixPadded		The extended reads produced by '-fixed' can still vary in length as the extension process will stop as soon as a read cannot be unambiguously extended. This option forces all corrected reads to be exactly the same length as they were prior to correction by extending them and then padding them out with Ns to the required length. This option is provided for compatibility with any tools that require all reads to be the same length.
-extend	*No. of bases to extend*	This option gets Blue to try to unambiguously extend all corrected reads, regardless of whether they were trimmed or not. This option can result in longer corrected reads, but the extension process is not perfect.
-paired		By default, if Blue is asked to correct pairs of reads files, it will maintain strict pairing between the reads written to each of the output files. If the '*-good'* option is used, then Blue will discard a complete pair of reads if any either of them fails the goodness test.
-unpaired		This option tells Blue not to treat pairs of reads files as a pair set. The main use of this option is with the *–good* option as it allows a set of unpaired files to be corrected in a single run of Blue, without having good reads from one file discarded because the corresponding (but unrelated) read in another file was uncorrected.
cbtFN		This the name of the kMer file to be used (produced by Tessel). Blue will also look for a .prs file with same name and use it if it finds it.

reads patterns FNs		These parameters specify the names of the sequence data files to be corrected. You can either supply a list of space separated files names or a filename pattern. On Windows you would normally use a pattern and let Blue turn it into a set of matching file names. On Linux, the same pattern will normally be turned into a list of file names by the shell, with equivalent results.

Setting the minReps parameter

Blue scans along each read, detecting broken kMers by looking up their repetition depth in the kMer consensus table produced by Tessel. The blue line in Figure 1 shows kMer repetition counts (depth of coverage) encountered along a real 454 read containing a few errors. The red line shows the depth of coverage for the read after correction. Blue examines each read and calculates two depths – an 'OK' level for the read around 70-80 in this case; and a depth that indicates an error, around 20 here. This depth calculation can only be done if there are enough good kMers in the read, and if there are not, Blue will calculate the OK level use the average depth from the entire kMer set to, and the value specified for minReps for the error threshold.

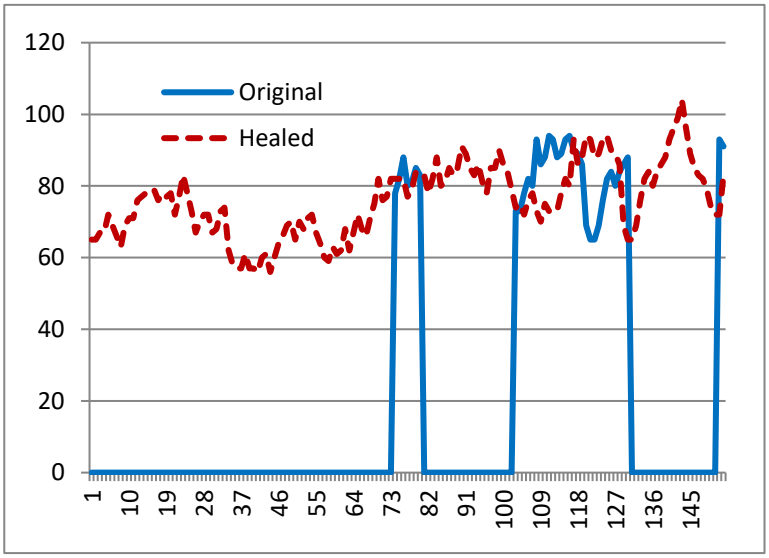

Figure 1: k-mer depths along a read

The normal way to set the *minReps* parameter is to use the repetition depth histogram generated by Tessel. Figure 2 shows the histogram produced from tiling 16,000,000 Illumina GAII reads for a 5Mbp organism. The Poisson-like curve represents the kMers actually found in the genome of the organism being sequenced, and a *minReps* value of around 50 would be suitable for this dataset.

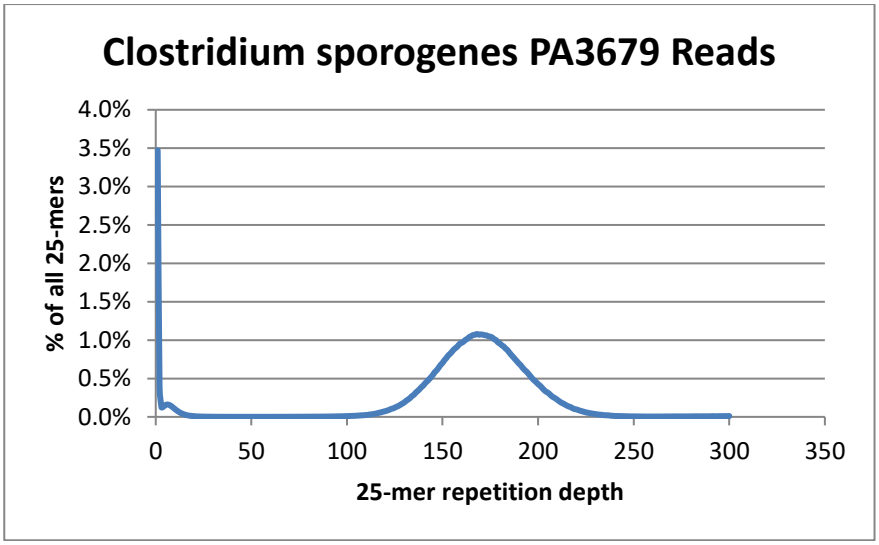

Figure 2: kMer depth histogram

The curve shown in Figure 2 comes from a pure microbial sample – the simplest case. For diploid organisms, there will often be two peaks, one representing homozygous kMers, and the other heterozygous kMers. The same procedure for choosing a *minReps* value applies though, just find somewhere in the valley between the error kMer spike (around 1-2) and the first peak. For metagenomic datasets, there will typically be one or more peaks corresponding to the dominant organisms in the community, so just choose a minReps value somewhere on the left, and away from any peaks that may represent organisms of interest.