Correcting reads with Blue (v1.1.0)

Blue is written in C#. This means that it will run natively on Windows, and can run under *mono* on Linux systems. Mono comes pre-installed on many Linux distributions, but can be downloaded from http://www.go-mono.com/mono-downloads/download.html and installed manually. Blue doesn't use any of the very latest C# language features so it should be compatible with almost all current mono/Linux distributions. The following commands and examples are for running Blue and its related tools on Linux, and on Windows the 'mono' should be dropped (and the '.exe' is optional).

Tessel

Blue works by correcting reads using a k-mer consensus table (a set of k-mers and counts), and optionally a set of k-mer pairs. The set of kmers+counts is generated by running Tessel.

```
mono Tessel.exe -k k-merLength -g genomeLength [-t #threads]
                  [-f fasta|fastq] [-tmp tempDir] [-m minDepth]
                  [-s] [-canonical|asread] [-text textFN]
                  [-textFormat pairs|sum|faPairs|faSum]
                  cbtName readsPattern or list of reads files
```

-k	*kmerLength*	is the k-mer length to be used (normally 25 but it can be anything up to 32. Do not use k values less than 20 as Blue needs a fair degree of k-mer uniqueness).
-g	*genomeSize:*	is a guess at the 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. *'-genome'* can also be used for this option.
-t	*noOfThreads*	is how many parallel threads to use for the tiling. This parameter is **optional** and 1 thread will be used by default. Tessel scales well with the number of threads, up to the point where it is limited by reading and writing the files. *'-threads'* can also be used for this option.
-f	*reads format*	fasta or fastq. This parameter is **optional** and the file format will normally be determined by looking inside the first of the reads files. *'-format'* can also be used for this option.
-tmp	*tempDir*	is an **optional** parameter that names a directory used for the temporary k-mer singleton files. I added this parameter so I could place these files on a different disk and reduce the impact of the temp file writes on reading the sequence files. It should only be used if you have a physically separate disk drive to use for these temp files, and faster local disk is much better than slower or remote storage.
-m	*minDepth*	is an **optional** parameter that specifies the minimum depth needed before a k-mer (and its count) will be written to the output .cbt file. The default value for this parameter is 1, so all k-mers will be written out, regardless of how many times they appear in the reads files. '*-min*' can also be used for this option. This option is used to discard k-mers that come from sequencing errors, and will be discarded by Blue anyway when the k-mers are loaded into its tables.
-s		Recursively search directories when looking for sequence data files.
-canonical		(default) regard a k-mer and its reverse complement as equivalent. Separate counts are maintained for both forms of each k-mer.

-asread	k-mers 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.
-text textFN	tells Tessel to write out the final k-mers and their counts to a text file. Tessel will also always write out the k-mer and counts 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 k-mers+counts to the text file. . The 'sum' and 'faSum' options are compatible with Jellyfish text file formats.
pairs	one line per k-mer: k-mer (tab) count (tab) rc-count (default)
sum	one line per k-mer: k-mer (tab) summed-count
faPairs	FASTA file. Header is >count rcCount. Sequence line is k-mer.
faSum	FASTA file. Header is >summed-count. Sequence line is k-mer.
cbtName	is used to construct the output file names. The k-mer consensus will be written to cbtName_kmerLength.cbt (e.g. Cspor_25.cbt) and the k-mer repetition depth histogram will be saved as cbtName_kmerLength_histo.txt (e.g. Cspor_25_histo.txt). If this parameter is simply a name, such as 'Cspor', then the output files will be placed in the current directory. If it is a full file name or path, such as 'Healed/Cspor', then these files will be written to this directory. If the '-asread' option is used, a '.abt' file suffix is used instead of '.cbt'.
reads pattern|FNs	these parameters specify the set of files to be tiled into k-mers. You either supply a list of space separated files names or a filename pattern. On Windows you would normally use a pattern and let Tessel 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.

Example:

```
mono Tessel.exe -k 25 -g 6000000 -t 4 Cspor s_1_?_sequence.fastq

mono Tessel.exe -k 25 -g 6000000 -t 4 Healed/Cspor s_1_?_sequence.fastq
```

GenerateMerPairs

Blue can also take a 'pairs' file ('.prs'). These are pairs of short k-mers separated by a small gap, and effectively extend the view of the correction algorithm further than a single k-mer. then you can use GenerateMerPairs will take a set of reads, and a .cbt file (from tessel) and produce a pairs file with the same name as the .cbt file (except with '.cbt' replaced by '.prs')

Pairs should only be generated (and used) if there is enough depth of coverage in the sequence data. Do not use a pairs file if there is not approximately one pair for each base in the expected 'genome'. For example, for a read set corresponding to 6Mbp bacterial genome, you should be looking for at least 6M pairs. If there are fewer pairs than this, just delete the pairs file and Blue will do a better job of correcting the reads. The first of the numbers from the run statistics will tell you how much coverage you have: 'wrote 5475432/124412923 pairs' means that GenerateMerPairs saw 5475432 unique pairs, and so your genome is probably about this length (actually a bit smaller).

```
mono GenerateMerPairs.exe [-t #threads] [-m minDepth]
                          cbtFN readsPattern or list of reads files
```

-t *noOfThreads*	is how many parallel threads to use when generating the pairs. This parameter is **optional** and 1 thread will be used by default. Currently around 6-8 is enough if you have that many processors available. *'-threads'* can also be used for this option.
-m *minDepth*	is an **optional** parameter that specifies the minimum depth needed before a k-mer (and its count) will be loaded into the k-mers table, and also the minimum depth needed for a k-mer within a pair. The default value for this parameter is 3. Use of this parameter can reduce the amount of memory needed by GenerateMerPairs. Setting the –min option for Tessel will have similar effects. '*-min*' can also be used for this option.
cbtFN	is the name of the .cbt file produced by Tessel from the set of reads specified by the next parameter.
reads pattern|FNs:	is a set of reads files, either a pattern or a list of file names.

Example:

```
mono GenerateMerPairs.exe -t 8 -m 10 Cspor_25.cbt s_1_?_sequence.fastq
mono GenerateMerPairs.exe -t 8 -m 10 Healed/Cspor_25.cbt s_1_?_sequence.fastq
```

Blue

Once you have a k-mer consensus file (.cbt), and optionally the set of corresponding k-mer pairs (.prs), you can go ahead and correct your reads using Blue.

```
mono Blue.exe [-help] [-r run] [-stats statsFN]-m minReps [-f fasta|fastq]
              [-hp] [-good %] [-t threads] [-o outputDir] [-fixed] [-variable]
              [-paired] [-unpaired]cbtFN readsPattern or list of reads files
```

-r	*runName*	is the name of this healing run. It is inserted into the names of each of the reads files to produce the name of the corresponding file of corrected reads. For example, the corrected form of 'ERR022075_1.fastq' will be 'ERR022075_1_runName.fastq'. This parameter is **optional**, and its default value is 'corrected_minReps'. *'-run'* can also be used for this option.
-s	*statsFN*	is the name of the file used to save the statistics from this run of Blue. This parameter is **optional** and by default Blue will generate a statistics file from the first reads file name parameter and the runName.
-m	*minReps*	is a k-mer repetition depth used (at times) to distinguish between 'good' and 'bad' k-mers. 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' k-mers at all.
-f	*reads format*	fasta or fastq. This parameter is **optional** and the file format will normally be determined by looking inside the first of the reads files. *'-format'* can also be used for this option.
-hp		The -hp option sets a flag that is checked when Blue is scanning along a read trying to find 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 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 very rare indels at the end of the hp run resulting in runs of 5 or 7 As. With 454-like data, we'd probably get 5 As as frequently as 6 As so depth of coverage would say that neither of them are 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.
-g	*%k-mers*	tells Blue to only save reads that look to be 'good' after correction. The '%k-mers' value specifies how many of the k-mers in a corrected read have to be above the 'good' threshold. Rejected reads are written to 'problems' files. Blue maintains the pairedness of its input files, and if one read of a pair fails the 'good' test, both the failing read and its pair will be dropped and written to the 'problems' file rather than the corrected reads files. This parameter is **optional**, and by default Blue will write all corrected reads to the output files. *'-good'* can also be used for this option.
-t	*noOfThreads*	is how many parallel threads to use for the tiling. This parameter is **optional** and 1 thread will be used by default. 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. *'-threads'* can also be used for this option.
-o outputDir	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. *'-output'* can also be used for this option.
-fixed	read lengths are maintained by padding and trimming. Padding is done by adding bases if there is no doubt as to what bases come next, and Ns in all other cases. This is the **default** behavior.
-variable	read lengths are allowed to change during correction as a result of insertions and deletions.
-paired	if Blue is asked to correct a set of reads, it will treat them as a set and 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 set of reads if any one of them fails the goodness test. This is the **default** option.
-unpaired	tells Blue not to treat a set of reads as a mate set. The main use of this option is with the *–good* option as it allows a set of files to be corrected in a single run of Blue, without having good reads from one file discarded because the corresponding (but unrelated)
cbtFN	is the name of the k-mer tiles 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 pattern|FNs	these parameters specify the set of files to be tiled into k-mers. You either supply a list of space separated files names or a filename pattern. On Windows you would normally use a pattern and let Tessel 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.

Examples:

```
mono Blue.exe -m 50 -t 8 Cspor_25.cbt s_1_?_sequence.fastq

mono Blue.exe –r g80 -m 50 -t 8 –g 80 –o Healed
                  Healed/Cspor_25.cbt s_1_?_sequence.fastq
```

Setting the minReps parameter

Blue scans along each read, detecting broken k-mers by looking up their repetition depth in the k-mer consensus table produced by Tessel. Figure 1 shows the types of depths encountered along a real 454 read containing a few errors (blue line). 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 k-mers in the read, and if there are not, Blue will use the average depth from the entire consensus to set the OK level, and the value specified for minReps for the error threshold.

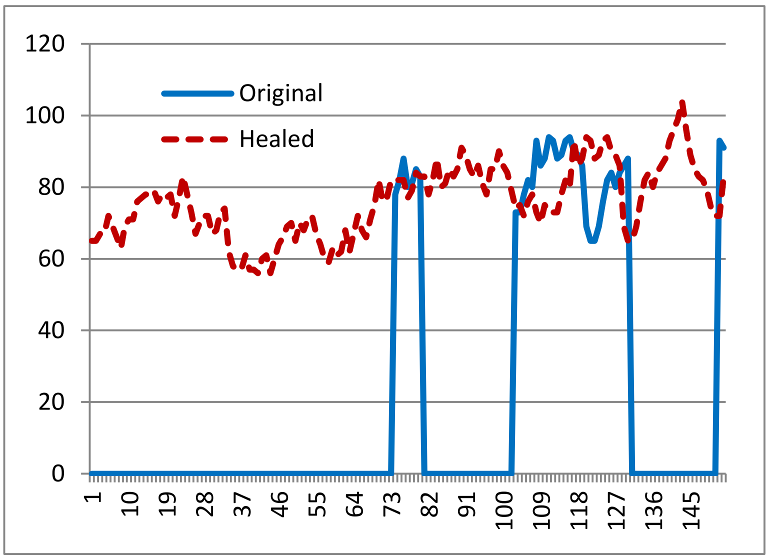

Figure 1: k-mer depths along a read

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 k-mers actually found in the genome of the organism being sequenced, and a *minReps* value of around 50 to 70 would be suitable for this dataset.

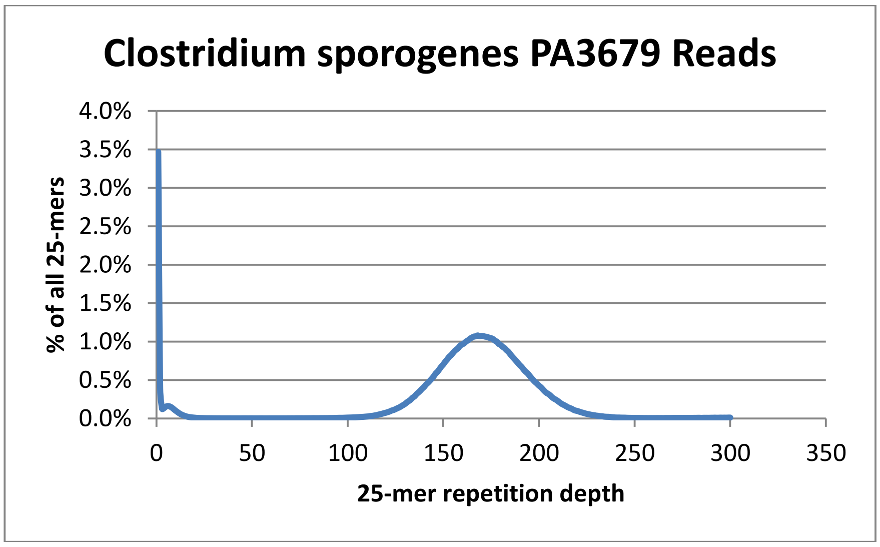

Figure 2: k-mer depth histogram

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 k-mers, and the other heterozygous k-mers. The same procedure for choosing a *minReps* value applies though, just find somewhere in the valley between the error k-mer 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.

Notes on Mono

Current releases of Mono support both the older Boehm memory manager ('garbage collector') and the newer SGen engine. Using SGen is recommended if it is supported in by your Mono release. You can do this by either running 'mono-sgen' (instead of 'mono'), or using the '—gc=sgen' option with 'mono'. The SGen garbage collector supports larger memory allocations, essential if you are correcting large files of reads. Blue (with SGen) has been used to correct 1.7B Illumina HiSeq reads (4 lanes), using about 80GB of memory without a 'pairs' table, and about 130GB with 'pairs'. This was done using Mono 2.10.8.1. The Boehm garbage collector is prone to asking to be recompiled when handling large memory allocations (and big sequence data files).

More details on running Mono can be found at http://docs.go-mono.com/?link=man%3amono(1).