1 Introduction

The aim of this software is to assess the performance of various methods of identifying pSNPs from sequence reads obtained from different technologies. It takes a template genomic sequence and then creates a tandem repeat of that template, while adding pSNPs at known positions. It then simulates a sequencing experiment on the tandem repeat segment, including the addition of sequencing errors according to a given error model. aligns the results to the original template, and uses this alignment to predict the positions of any pSNPs. Finally the software compares predicted pSNPs with the known positions of the simulated pSNPS to produce statistics assessing the accuracy of the various methods.

Figures 1 and 2 indicate the two main use cases of the software, analysing either simulated .fastq output from an Illumina or PacBio experiment, or both a .fastq file and a .h5 file from a PacBio experiment. There follows a brief outline of the main routines as described in figures 1 and 2, with a more detailed outline of the software developed within this project in Sections 2 to 7.

Template.fasta This .fasta file is the base sequence for the tandem repeat under study. For example for the yeast strain *S cerevisiae* S288c the template sequence of a single unit is 9237bp in length. The template is duplicated by pSNPgenerator.R to create a tandem repeat sequence, which can be used by the alignment software and VariantLister as a reference sequence. For some analyses it is useful to create a double length template consisting of a tandem repeat for mapping, as this can reduce alignment errors at the ends of the sequence.

Input bas.h5 This bax.h5 or bas.h5 file contains details of PacBio sequencing reads. AnalyseBax will use them to produce distributions of the various quality fields, which can be used by Simbas to simulate similarly distributed reads. They are also used to improve the accuracy of the alignment software BLASR (**?**) and the pSNP caller VariantLister.

AnalyseBax A C++ command line program that takes a bas.h5/bax.h5 file and produces two files. One file contains a distribution of the Phred scores associated with the five .h5 quality variables, DeletionQV, InsertionQV, MergeQV, SubstitutionQV and QualityValue. This distribution can be used as a bas.h5/bax.h5 file model for input to the Simbas program. The second file contains contingency tables comparing the DeletionQV, InsertionQV and SubstitutionQV variables and can give insight into any mutual dependencies between these variables.

pSNPgenerator.R An R script that takes as input a template .fasta file and produces a specified number of simulated tandem repeats with randomly added pSNPs, as well as a .csv file with the positions of the added pSNPs. The resultant .fasta can be used as a template in a read simulator, e.g

pIRS or PBSIM. The positions of the pSNPs can be used later by software such as Plotroc to analyse the success of differing pipelines at identifying pSNPs.

Simbas A bas.h5/bax.h5 file for simulating PacBio experiments. It takes as input a .fasta file with the genomic region from which to simulate reads, plus a quality model as outputted from the AnalyseBax program. The model is human readable and could also be created by hand. The result is a bax.h5 file of simulated reads with the required coverage and length of the reads defined by input options.

Alingment Third party programs were used for alignment of sequence reads to the template .fasta file. BLASR (**?**) and BWA MEM were used for the long Pacific Bioscience reads, and Stampy (**?**) and BWA ALN (**?**) for Illumina and short PacBio reads. The mapped reads are output as a .sam file.

samtools view A command line tool to convert a .sam file into a binary compressed .bam file (**?**).

VarientLister A command line program written in C++ that takes as input a .bam file containing reads mapped to a template .fasta file. The output files contain pSNP calling information. P-values of potential pSNPs that can be used by the Plotroc program, plus human readable .csv files with information on each position. The three methods of pSNP calling currently supported by this software are binomial, Poisson and Known Frequency. If the reads aligned in the .bam file were generated by a PacBio machine or simulator, then the associated bas.h5/bax.h5 file, containing reads plus extra quality information, can be also be used as input to increase the accuracy of pSNP calling.

.pvalues.csv Output from VariantLister. This file comprises an ordered list of p-values, where each value matches a template position, giving an estimate of the probability that the detected variants are errors rather than pSNPs.

Plotroc A command line program written in C++. It assumes we have the results of a simulation experiment and takes a file with suffix .pvalues.csv outputted from VariantLister plus the .csv file with the list of added pSNPs outputted by pSNPgenerator.R. Plotroc then outputs a .pdf file giving a ROC curve showing true positive rate verses false positive rate. This gives a good visual indication of how accurate was the pSNP calling method under investigation.

Pipwrap This tool, written in C++, provides an intuitive framework for constructing and running sequences of shell commands. It enables the setting of parameters to third-party and bespoke software in the software pipeline.

All the software developed during this project is available at the GitHub site https://github.com/Sriep/pipewrap . With the exception of the R script

pSNPgenerator, all the software was developed in C++ using the Qt Development environment (**?**).

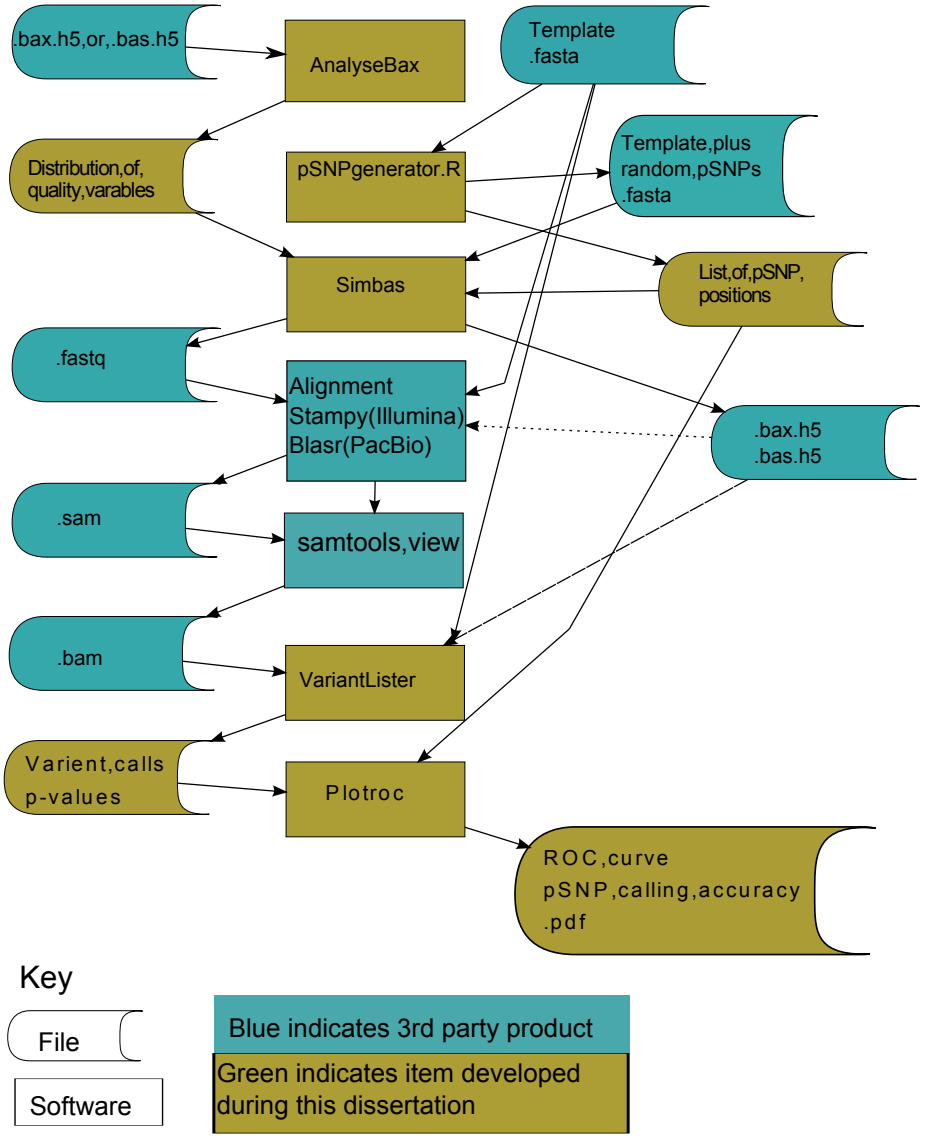

Figure 1: Use case for simulating a PacBio experiment using .h5 machine output files. Blue indicates 3rd party software or data formats, green indicates software or data formats developed as part of this project.

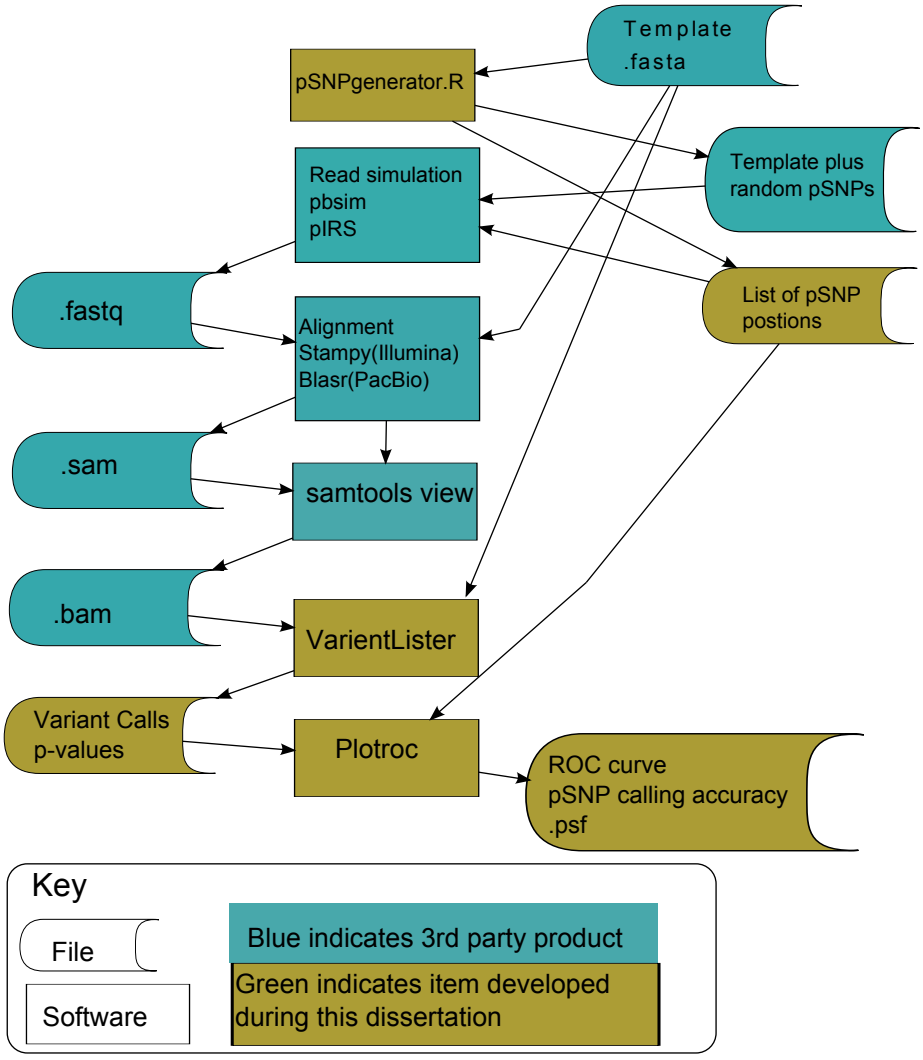

Figure 2: Use case for simulating a PacBio or Illumina experiment using just .fastq machine output files. Blue indicates 3rd party software or data formats, green indicates software or data formats developed as part of this project.

2 pSNPgenerator.R

pSNPgenerator.R is a command line application written in the R scripting language. It takes a gnomic sequence in a .fasta file format, and creates a tandem repeat of that sequence while introducing pSNPs at random locations. It outputs a .fasta file of the tandem reapeat and a .csv file with the positions of the inserted pSNPs.

The output tandem repeat .fasta file is used as input to such sequencing read simulators as pIRS (**?**), PBSIM (**?**) , alchemy (**?**) and Simbas (see section 4). The pSNP positions can be used as input to the Plotroc (see section 6) application which creates ROC curves to visualise the accuracy of pSNP calling.

The user enters the number of tandem repeats to be created plus a set of pSNP frequencies and the number of pSNPs of each frequency to produce. The program continuously cycles through the tandem repeats, inserting a pSNP in each one in turn. Each relative pSNP position is determined randomly, then that pSNP is inserted into the correct number of tandem repeats to make up the requested frequency.

Two consequences of this algorithm are that firstly that each tandem repeat has to within one the same number of pSNPs and secondly that any particular pSNPs appear in either one blocks of adjacent tandem repeats or two blocks at either end of the tandem repeats.

The output file containing the pSNP positions, is tab delimited with the following columns

index Row number.

position Position of pSNP relative to template.

csBase Original base in the template.

pSNPBase New pSNP base, changed from the template.

frequency The frequency that this particular pSNP appears across the tandem repeats..

StgartTR The position of the first tandem repeat to which this pSNP was added.

The R library SeqinR was used to read and write .fasta files (**?**).

2.1 Command line options

i Input template file to be tandemly repeated, in .fasta format.

o Output .fasta file which will hold the tandem repeats plus randomly generated pSNPs.

c Output file containing pSNP positions.

t Number of tandem repeats to generate.

p Number of pSNPs to create in each bin.

b This option is best entered last. It contains a space delimited list of pSNP percentage frequencies. For each list entry N say a number of pSNPs equal to the -b option is generated. If T is the number of tandem repeats specified with the -t option, then each of these pSNPS are inserted into $TN/100$ tandem repeats.

3 AnalyseBax

AnalyseBax is a command line application written in C++. It takes output from a PacBio sequencing experiment and preforms two types of analysis. The first analysis examines empirical distributions of PacBio-specific base call error scores, while the second looks for dependencies between these quality scores. The outputs are required for simulating Pac-Bio-like sequence reads. (**?**).

3.1 Distributions of PacBio base call error probabilities

AnalyseBax outputs the distributions of the four base call error probabilities DeletionQV, InsertionQV, MergeQV and SubstitutionQV found within .h5 files along with an aggregate QualityValue field which gives the overall probability of a base call error (**?**). These probabilities are all stored as Phred values (**?**) in the .h5 file, thereby taking integral values between 0 and 100.

The output of this analysis is a tab-separated file with 101 rows after the header and six columns. One column represents the Phred index with the remaining five representing the DeletionQV, InsertionQV, MergeQV, SubstitutionQV and QualityValue fields. Each cell gives the percentage of reads that have the indexed Phred score.

The output Phred distributions can be used as input to Simbas, the bas.h5/bax.h5 simulator, which will then simulate a .h5 file with the same quality distribution as the analysed file. Table 1 gives an example of the output produced from a PacBio sequencing run of *S. cerevisiae* strain S288c. Figure 3 shows a graphical representation of the quality score distributions in Table 1.

3.2 Dependencies between PacBio base call error probabilities

AnalyseBax also creates contingency tables showing the dependencies between the four quality values DeletionQV, InsertionQV, MergeQV and SubstitutionQV. Analyses of real PacBio .h5 files indicated that base call error probabilities often possess a bimodal distribution, with two peaks at the extremes of their range and with a plateau in between. With this in mind, the values taken

by each of the four variables were split into three bins, as shown in table 2. The ranges of these bins are currently hard coded, but could be made configurable in later updates of this software.

Contingency tables, output as tab delimited files, are shown as three-way interactions between all combinations (27 in total) of the DeletionQV, InsertionQV and SubstitutionQV bins. For example, table 3 shows an example of this output for *S. cerevisiae* strain S288c. MergeQV values could be added to the output file in a later software update.

At the end of each table row and column aggregate observed and expected values are given, where the observed value is the column or row total, and the expected value is based on the assumption of independence between the bins. Hence a significant difference between the observed and expected totals for a row or column would suggest a dependency between the relevant bins. The values at the bottom left and right of the table give the column and row totals for the entire contingency table, found by summing over all three variables.

Expected values are obtained by multiplying the row and column totals of the relevant bin combinations. For example, in table 3 the expected value for the first section of the first column, which corresponds to InsertionQV Bin1, DeletionQV Bin1 and all three SubstitutionQV Bins is 1.57885. This is calculated by multiplying the observed value for DeletionQV Bin1 (the row total 7.26258) by that for InsertionQV Bin1 (the column total 21.4398), and bearing in mind that all values are percentages:

$$1.57885 = \frac{21.7398x7.26258}{100} \tag{1}$$

Table 1: An example of the first 32 rows of a quality distribution file output from an AnalyseBax analysis of input file m130724_104949_42149_c100529652550000001823089211101375_s1_p0.bas.h5 obtained from a sequencing run of *S. cerevisiae* strain S288c.

phred	DeletionQV	InsertionQV	MergeQV	SubstitutionQV	QualityValue
0	0	0	0	0	6
1	0	0	0	0	5
2	0	7	0	0	5
3	0	5	0	0	5
4	0	4	0	1	5
5	2	4	0	3	5
6	1	4	1	4	5
7	1	3	1	5	6
8	1	4	1	5	6
9	0	4	1	4	6
10	0	4	2	4	7
11	0	4	2	3	7
12	0	5	3	3	7
13	0	5	3	2	8
14	0	5	3	2	9
15	0	5	4	2	0
16	88	6	4	2	0
17	0	6	4	2	0
18	0	6	4	2	0
19	0	6	4	2	0
20	0	4	4	1	0
21	0	0	4	1	0
22	0	0	4	1	0
23	0	0	3	1	0
24	0	0	3	1	0
25	0	0	2	1	0
26	0	0	2	1	0
27	0	0	1	1	0
28	0	0	1	1	0
29	0	0	1	1	0
30	0	0	1	29	0
31	0	0	0	0	0

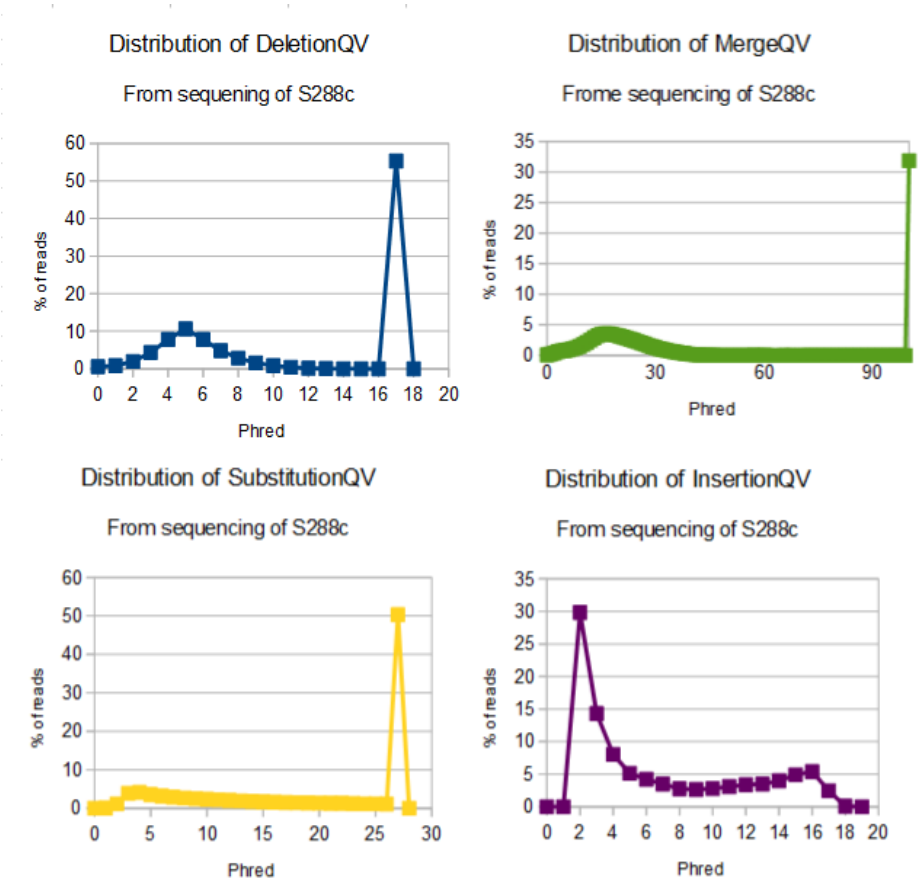

Figure 3: Graphical representation of the quality distributions output from an AnalyseBax analysis of a PacBio sequencing run of *S. cerevisiae* strain S288c, for input file m130724_152807_42149_c100529652550000001823089211101377_s1_p0.mcd.h5. The four curves shows the Phred score frequencies for the DeletionQV, MergeQV, SubstitutionQV and InsertionQV base call errors found within the .h5 file.

Table 2: Bins used for construction of contingency tables

Varaible	Bin1	Bin2	Bin3
DeletionQV	0-7	8-15	16+
InsertionQV	0-5	6-12	13+
SubstitutionQV	0-8	0-22	23+

Table 3: Example output of a three-way contingency table from a PacBio sequencing run of *S. cerevisiae* strain S288c. The input file used is m130724_104949_42149_c100529652550000001823089211101375_s1_p0.bas.h5.

Del 0-7	Ins 0-5	Ins 6-12	Ins 13+	Observed	Expected	Total
Sub 0-8	2.64978	0.391664	0.0629712	3.10441	1.63108	
Sub 9-22	1.31136	0.591697	0.417034	2.32009	2.77051	
Sub 23+	0.669987	0.65293	0.515159	1.83808	2.86099	
Observed	4.63112	1.63629	0.995164	7.26258		
Expected	1.57887	2.27731	3.4064			
Del 8-15	Ins 0-5	Ins 6-12	Ins 13+	Observed	Expected	Total
Sub 0-8	0.729959	0.190343	0.0419006	0.962202	0.844683	
Sub 9-22	0.526133	0.486761	0.302315	1.31521	1.43476	
Sub 23+	0.371617	0.639327	0.472702	1.48365	1.48162	
Observed	1.62771	1.31643	0.816918	3.76106		
Expected	0.817646	1.17935	1.76407			
Del 16+	Ins 0-5	Ins 6-12	Ins 13+	Observed	Expected	Total
Sub 0-8	8.084	6.48718	3.82085	18.392	19.9829	22.4586
Sub 9-22	4.88004	10.1133	19.5192	34.5125	33.9425	38.1478
Sub 23+	2.51691	11.8036	21.7514	36.0718	35.051	39.3936
Observed	15.481	28.404	45.0914	88.9764		
Expected	19.3433	27.9001	41.733			
Total	21.7398	31.3567	46.9035			

The first (non-header) row of table 3 shows (3.10441, 1.63108) as the observed and expected values corresponding to DeletionQV Bin1 and SubstitutionQV Bin1 (across all InsertionQV bins). The moderate difference between these observed and expected values indicates a possible correlation between low DeletionQV Phred scores and low SubstitutionQV Phred scores. A χ^2-test could be performed to confirm this.

The third row of the same table shows (0.995164, 3.4064) as the observed and expected values for InsertionQV Bin3 and DeletionQV Bin1 (across all SubstitutionQV bins). These values suggest a possible inverse correlation between high InsertionQV Phred scores and low DeletionQV Phred scores.

After inspecting the contingency tables of several contrasting datasets, we did not detect a high level of dependency between the quality variables in any of them. Hence in the rest of our study we ignore such dependencies, effectively assuming that Phred scores for the different base call errors are independent of one other.

3.3 Command line options

The following command line options are available for AnalyseBax:

i The name of the input bax.h5 or bas.h5 to be analysed.

o The prefix name for the output files. The distribution output file will append the suffix "_dist.csv" to this prefix name, while the contingency table output will have the suffix "Dependencies.csv" added.

d Determine and output the distribution of the four variables DeletionQV, InsertionQV, MergeQV and SubstitutionQV.

c Determine and output the contingency tables for the .h5 file, examining correlations between the three variables DeletionQV, InsertionQV, and SubstitutionQV.

h Display help information.

v Display version information.

4 Simbas

Simbas is a command line application written in C++ that will produce a simulated .bax.h5 file of reads from a PacBio sequencing experiment. It takes a template .fasta file from which to construct the reads, plus a distribution file, which gives a model of the DeletionQV, InsertionQV, MergeQV and SubstitutionQV variables.

A distribution file is most easily created by running AnalyseBax on a bas.h5/bax.h5 file with the -d option. Table 1 gives an example of such a distribution model file. At present the QualityValue column is not used and

can be filled with dummy values if constructing the file by hand. Currently all simulated reads are of the same length. This could later be extended to allow a range of lengths as determined from analysing a .h5 file.

4.1 Command line options

t Template in .fasta format. If the first line starts with '>' it is ignored. The rest of the file is examined and used to generate reads. For each read, a start position on the template is determined at random. Then during a read simulation a growing simulated read points to a position on the template. If the template pointer reaches the end of the template it is subsequently wrapped around to position zero.

p File containing the distribution of the DeletionQV, InsertionQV, MergeQV and SubstitutionQV base call error probabilities. It is advised to generate this file using AnalyseBax.

d The depth to which the simulated reads will cover the template file.

l The length of the reads. All reads will be the same length.

4.2 Implementation

Simbas uses the lhdf5 and ldf5_cpp third party librarie (**?**) to exposed a C++ API for accessing .h5 files. Figure 4 illustrates the Simbas read generation algorithm. The Simbas classes are as follows:

SimBax The main class for creating the simulated .h5 file. It is a functor, with the default constructor and an overload of the function call operator the only two public functions.

Options A wrapper around the getopt_long function (**?**) for parsing and accessing command line options. All accessor members have static scope.

BaxH5 Derived from the ldf5 cpp class H5File, used to create and write to the .h5 file.

BaseCaller Base class for creating random reads.

DiscreteCaller Derived from BaseCaller, uses the information in the quality distribution file to generate random base calls.

Figure 4 gives the algorithm where each read is determined. For each read, a start position on the template is determined randomly. Then during a read simulation we a growing simulated read and a pointer to a position on the template. If the template pointer is incremented of the end of the template then the pointer is wrapped around to position zero. As long as the growing read is less than the required length base calls are added using the simulatedat()

method. A random number is generated to determine whether there is an error then a second random number to determine which kind of error plus as below:

No Error The matching template base is added to the growing read, and the template pointer is incremented. Random non-template DeletionTag and SubstitutionTag bases (see below) are saved.

Deletion The template position is incremented, without any base being added to the read. A DeletionTag base is determined as below, and the simulate-data method is called recursively with a flag to indicate that no further DeletionTag base needs to be generated.

Insertion A random base is added to the read, as well as a random DeletionTag and SubstitutionTag. A flag is set to indicate that the template position should not be incremented.

Substitution A random base not matching the template is added to the read. The SubstitutionTag field is determined as below, and a random Deletion-Tag base is added.

DeletionTag The distributions of DeletionQV and SubstitutionQV are assumed to follow a bimodal distribution, for example see figure 3, where a significant proportion of the probability mass is around the maximum Phred score in the range of values taken. During base calling, when a substitution or deletion error has likely occurred there is typically a clear alternative or missing base call. Consequently, if an error has been made the correct base call can still often be identified. In this context the DeletionQV is the Phred score of the most likely candidate for a missing base, and the DeletionTag is the identity of this base. The remaining two bases are assumed to be possible deletions with Phred scores equal to the background Phred score. In Simbas we have set this background rate backgroundDel to be the maximum Phred score with non-zero occurrence as given in the inputted distribution model. For the purposes of our simulation if we have generated a deletion error, we assume the alternate base is always correct so generate a random probability $p \epsilon [0,1]$ and set DeletionTag to the template base if $p < \frac{DeletionQV}{DeletionQV+3\times backgroundDel}$ and otherwise set DeletionTag to be one of the remaining two bases at random.

SubstitutionTag The SubstitutionQV, SubstitutionTag pair works the same way to DeletionTag, with SubstitutionTag the most likely correct base if a substitution error has occurred. If such a substitution error has occurred then SubstitutionQV is the Phred value corresponding to the probability that that substitution has happened while the background substitution rate backgroundSub gives the chance that either one of the remaining two bases is actually the correct base. As for deletions we generate a random probability $p \epsilon [0,1]$ and set SubstitutionTag to the template base if $p < \frac{SubstitutionQV}{SubstitutionQV+3\times backgroundSub}$ and otherwise set SubstitutionTag to be one of the remaining two bases at random.

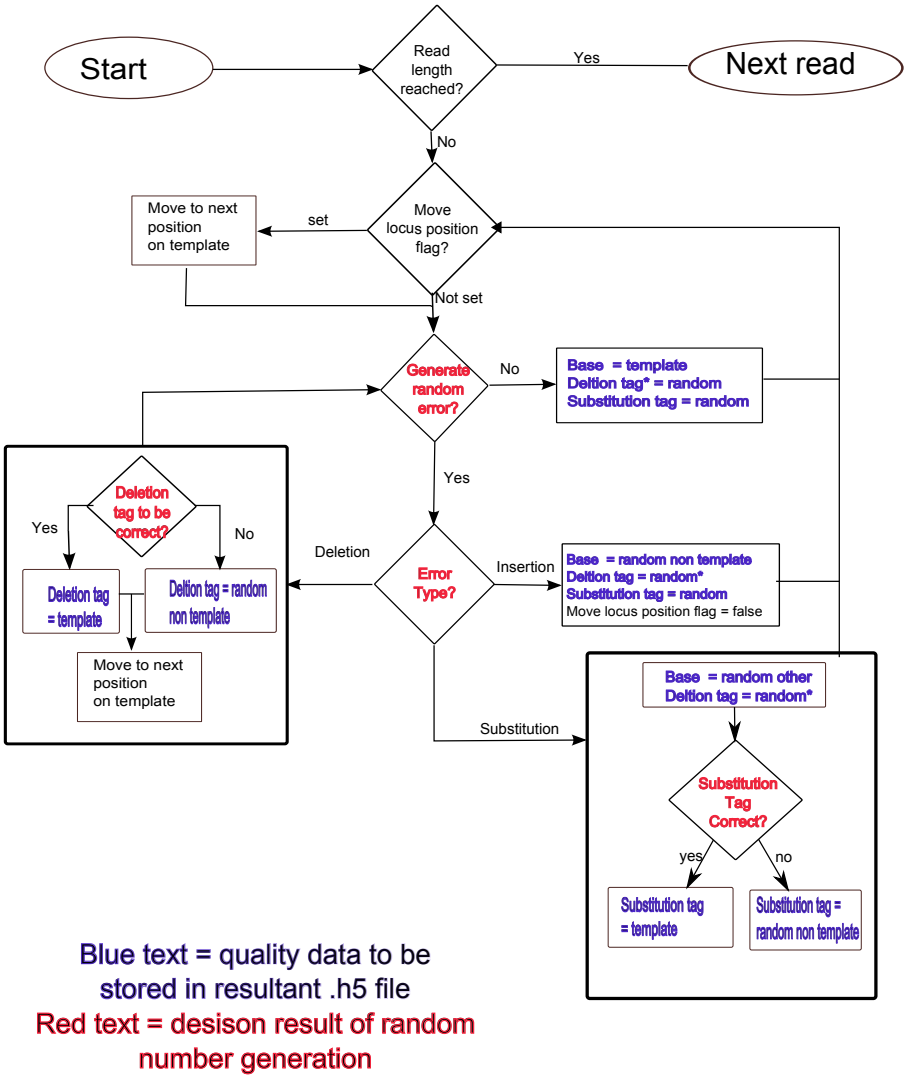

Figure 4: Flow diagram of the Simbas simulation algorithm, showing how the base calls that make up a read are determined. Blue indicates where the base and quality values are determined, and red indicates the result of random decision generation. *Do not create another DeletionTag base if reached recursively after deletion error

5 VariantLister

VariantLister is a command line application written in C++. It takes sequencing alignment information relative to a template sequence in the form of a .bam file, and uses information on differences between sequence reads and the template to identify pSNPs. It currently supports three algorithms for calculating the probability that all the variants matching a particular template position are errors: binomial, Poisson and Known Frequency. VariantLister is the core program in our pipeline to compare sequencing machines using simulated reads. It requires a .bam file and a template .fasta file. Furthermore, pSNP calling information can be improved for PacBio sequencing experiments by including the associated bas.h5/bax.h5 file.

VariantLister can output several optional data files. This information is output as a newline delimited list of floating point values, to a file with suffix .pvalue.csv that can be used as input to the Plotroc application (see section 6). VariantLister also outputs a locus information, tab separated file, with each row containing information about each position along the template. Fields ouput in this file are as follows, where one of the final three will be included, depending on the method selected in the command line options:

Locus Index of base position relative to start of template sequence.

Base The base found in the template sequence.

Coverage The number of reads included in p-value calculations. Note that this will in general be less than the total coverage provided by the sequencing experiment due to filtering.

AveQuality The average Phred quality value for reads included in the coverage.

pSNP The variant call with highest frequency . A '=' character indicates that no variants have been found.

pSNPCount The number of reads possessing the highest frequency variant.

pSNP% The percentage of reads possessing the highest frequency variant relative to coverage.

pvBionomial Probability that all variants are errors, calculated using the binomial algorithm.

pvPoisson Probability that all variants are errors, calculated using the Poisson algorithm.

pvKF Probability that all variants are errors, calculated using the Known Frequency algorithm.

5.1 Command line options

i Input .bam file containing sequence reads mapped against the template sequence.

t Input .fasta file containing the template sequence. The .bam file must have been aligned against this sequence or a tandem repeat of the input sequence. If the .bam file has been aligned against a repeat of the input template then all indices are determined as modulo the (unit) template length.

x Input .bas.h5 or .bax.h5 with extra PacBio quality information. This file can be used to improve p-value calculation.

r Output filename of an optional data file containing a row of information on each read at each position.

l Output filename of an optional data file containing a row of aggregate information on reads at each position in the template.

b Use p-values calculated by the binomial method.

p Use p-values calculated by the Poisson method.

k Use p-values calculated by the Known Frequency method.

f Number of frequency partitions to be used when calculating p-values via the Known Frequency method.

d Optional quality filter (for bas.h5/bax.h5 files only). Pre-filters reads with DeletionQV Phred score less than the entered value.

e Optional quality filter (for bas.h5/bax.h5 files only). Pre-filters reads with InsertionQV Phred score less than the entered value.

u Optional quality filter (for bas.h5/bax.h5 files only). Pre-filters reads with SubstitutionQV Phred score less than the entered value.

q Optional quality filter. Pre-filters reads with the Phred quality score (QualityValue field if .h5 file available) less than the entered value.

a Optional quality filter. Post-filters reads by an intermediately-calculated quality value, which is the Phred score from the .bam file unless an .h5 file is provided. For a .h5 file, a new quality Phred score is calculated. Reads can then be filtered if this Phred score is too high.

h Displays help information to standard output.

v Displays version information to standard output.

5.2 Implementation

VariantCaller uses the lhdf5 and ldf5_cpp third party libraries (**?**) which exposed a C++ API for accessing .h5 files. The Bamtools C++ API was used for accessing the alignment information in the .bam file (**?**).

VariantCaller The main function class. It exposes a default constructor and an overload of the function call operator.

Options A wrapper around the getopt_long function (**?**) for parsing and accessing command line options. All accessor members have static scope.

LocusInfo A wrapper around the data for each locus position in the template.

PValues Wraps the binomial, Poisson and Known Frequency algorithms for calculating p-values. Called from the LocusInfo class when a p-value is requested for that position. It is logically a static class with no data members.

MatchMismatches Helper class derived from the std::string class in the C++ standard library. It takes the aligned bases and cigar from an entry in a .bam file and forms the corresponding sequence that can be directly mapped against the template.

Hdf5BasFile Derived from the ldf5_cpp library class H5File, this class handles read only access to the optional .h5 file.

5.2.1 VariantCaller algorithm

The main algorithm consists of two loops. The first loop iterates through all the read, position pairs in the .bam file and filters out unsuitable reads depending on user options while making a note of the number and nature of the remaining reads at each position, specifically the called base and quality information.

Figure 5 gives a graphical outline of this loop. Each sequence read entry in the .bam file is examined in turn, beginning with a set of filters. If the sequence alignment position from the .bam file is zero this indicates that the read does not match our template and can be ignored. Any optional quality pre-filters are handled next. If the quality value from the .bam file is less than the -q option Phred score then the read is ignored. Furthermore, if any of the -d, -e or -u options have been set, then the DeletionQV, InsertionQV or SubstitutionQV values respectively from the bas.h5/bax.h5 file are compared with the entered values, such that a read is ignored if one or more relevant scores are lower than these values.

Next the current read, position pair is compared with the template. If there is a match then the match counter of the appropriate LocusInfo object is incremented and the quality score from the .bam file is pushed onto a list. If there is a mismatch and there is no optional .h5 file, then the mismatch and

mismatch base counters of the corresponding LocusInfo object are increased and the quality score is pushed onto a list. The loop then proceeds to the next read, position pair.

The algorithm becomes significantly more complex in the case of a mismatch in the presence of an optional .h5 file. Firstly the corresponding read in the .h5 file needs to be located. We rely on the read id in the .bam file containing the position of the read against the template in the \PulseData\BaseCalls\ZMW\NumEven field between two '\' characters. From this we can locate the read in the corresponding .h5 file by comparing the .bam position with the \PulseData\BaseCalls\Basecall field and hence we can read off the corresponding DeletionQV, InsertionQV, MergeQV and SubstitutionQV scores.

We now calculate the probability that a sequencing error has occurred. Let W be the size of any run of consecutive bases matching the template base, with a maximum value currently hard coded as 20. Due to the high probability of insertion and deletion base call errors within PacBio sequence reads there could be several ways that such a match could have occurred. template/tACGGGA /t/t|||||| read/tACTGGA In the above, ACTGGA could have been generated from ACGGA by either a TG substitution or deletion of any of the Gs plus a T insertion. To account for this we multiple the indel probability by the number of adjacent copies of the substituted base in the template.

Del represents the probability of a deletion error. The DeletionTag gives the most likely alternate base. The Phred score corresponding to the alternate base is assumed to be that given by DeletionQV while the other two bases are assumed to appear with a background probability currently hard coded as 20. If the DeletionTag matches the template base then a sequencing error becomes likely, otherwise a pSNP becomes highly likely (**?**) (**?**). If Q_d is the DeletionQV Phred score and B_D the background value then

$$Del = \begin{cases} \frac{Q_d}{Q_d+2B_d} & \text{If DeletionTag matches the template} \\ \frac{B_d}{Q_d+2B_d} & \text{If DeletionTag does not match the template} \end{cases}. \quad (2)$$

Sub represents the probability of a substitution error. The SubstitutionTag gives the most likely alternate base. The Phred score corresponding to the alternate base is assumed to be that given by SubstitutionQV while the other two bases are assumed to appear with a background probability currently hard coded as 30. If the SubstitutionTag matches the template base then a sequencing error becomes likely, otherwise a pSNP becomes highly likely (**?**) (**?**). If Q_s is the SubstitutionQV Phred score and B_s the background value then

$$Sub = \begin{cases} \frac{Q_s}{Q_s+2B_s} & \text{If SubstitutionTag matches the template} \\ \frac{B_s}{Q_s+2B_s} & \text{If SubstitutionTag does not match the template} \end{cases}. \quad (3)$$

Let I represent the InsertionQV Phred score and M the MergeQV Phred score. Using these variables we calculate the probability of mismatch between

a sequence read and the template sequence being due to a sequencing error is,

$$Prob(Error) = Sub + Del(W+1)10^{\frac{-I-M}{10}} \tag{4}$$

Once calculated, the Phred score corresponding to this $Prob(Error)$ is compared with the quality post-filter value (option a), and if it is less than the entered value the read is skipped. Otherwise the mismatch and mismatch base counters of the corresponding LocusInfo object are increased and the quality score is pushed onto a list. The loop then proceeds to the next read, position pair.

The second major loop within the VariantLister algorithm iterates along the template sequence, calling the p-value calculation algorithm requested in the command line options. Details of these algorithms are given in Chapter 1. Following the identification of putative pSNPs with p-values below a threshold, the required results are written to various output files.

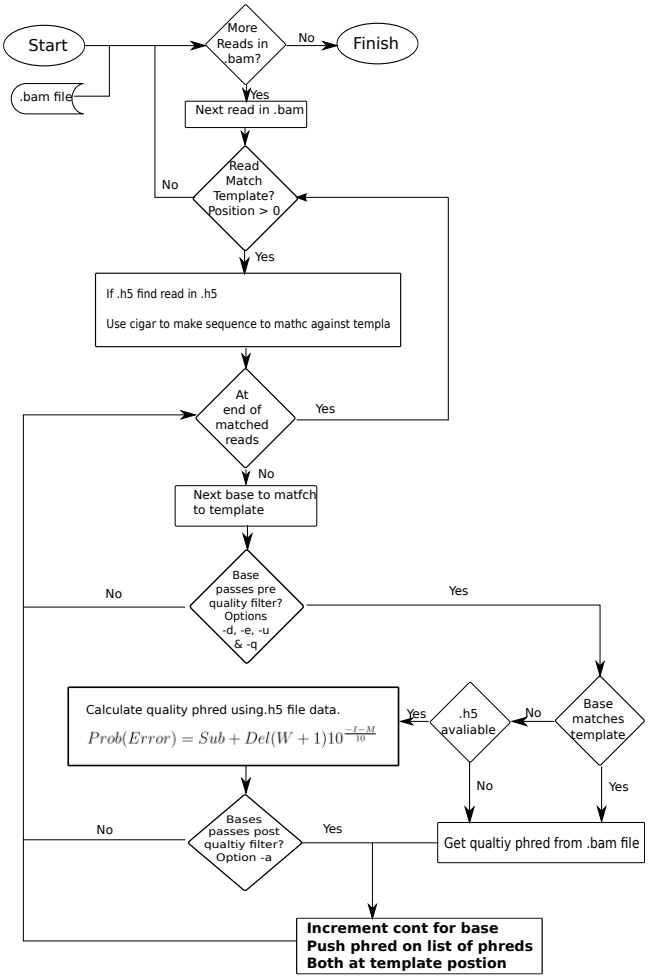

Figure 5: Flow diagram showing the main VariantCaller loop for processing alignment information from a .bam and optional .h5 file. The aim of this routine is to identify quality values associated with mismatches between each template position and suitable sequence reads that map to them. This provides enough information at each template position for subsequent pSNP calling with all three supported p-value calculation algorithms.

6 Plotroc

Plotroc is a command line application written in C++ that produces a receiver operating characteristic (ROC) curve. The main use case for Plotroc within this project involves analysing the predictive success of different pSNP calling algorithms upon sequence reads produced by different sequencing technologies. Specifically, the R script pSNPgenerator.R is used to create a .csv file containing the known positions of randomly generated pSNPs along a simulated tandem repeat of a template sequence. Then a sequencing experiment is simulated on the template sequence and the resulting reads are mapped back to the template before using VariantLister to identify the positions of putative pSNPs. VariantLister produces a file containing an ordered newline separated list of p-values for each locus. The p-values represent the probability that all the variants at that position or locus in the mapped template are errors and hence not a pSNP. The results of alternative variant callers, such as VarScan2 (**?**), can also be edited to produce such a file of p-values.

We are interested in analysing threshold p-values, θ say, where all p-values less than that value are to be called pSNPs. By using the known true pSNP positions, for each possible threshold, we can determine the number of true positives (TP_θ = number of loci with a true pSNP and p-value $< \theta$) and false positives (FP_θ = number of loci known to not have a pSNP but with a p-value $< \theta$). Similarly we can define true negatives (TN_θ = number of loci known to not have a pSNP and with a p-value $> \theta$) and false negatives (TN_θ = number of loci known to have a pSNP but with a p-value $> \theta$). We can now define the true positive rate $TPR_\theta = \frac{TP_\theta}{TP_\theta + FN_\theta}$ and false positive rate $FPR_\theta = \frac{FP_\theta}{FP_\theta + TN_\theta}$ of pSNP identification.

Plotroc creates a scatter graph of (FPR_θ, TPR_θ) values for an ever decreasing sequence of values of θ, such as $\{10^{-n}\}$, where n gets ever smaller until the minimum strictly positive number the computer can handle is reached or the minimum p-value for the set of results undergoing analysis has been passed. We then join the scatter points to give a ROC graph. This scatter graph gives a visual impression of how good the pSNP calling process has been. The best pSNP caller will produce a graph that gets closest to the point (FPR = 0, TPR = 1) while the worse case is a graph that approximates to the line $TPR = FPR$.

6.1 Command line options

s Output file from the R script pSNPgenerator.R containing information on true pSNP positions. If this option is set then a ROC curve of FPR to TPR is generated for each p-value file with the suffix ".p-value.csv". All files in the working directory with suffix ".pvalue.csv" are assumed to be a newline delimited list of p-values corresponding to the template sequence from which the known pSNPs were generated.

t Title to give the ROC curve.

I Locus information file, output from the VariantLister application. Variants with error p-values less than the hard coded values 10^{-50}, 10^{-100}, 10^{-150} and 10^{-200} are considered true positive pSNPs for the purposes of the ROC curve.

P Locus information file, output from the VariantLister application. Used to compare a second sequencing experiment with the base experiment entered with option I. The resultant curve gives an indication of how similar this pSNP profile is to that entered with option I. The closer the resultant curve gets to the point (0,1) the more similar the pSNPs in the two result sets are.

7 Pipewrap

Pipewrap is a GUI application written in C++. Figure 6 gives a screen shot of the software in action. Pipewrap allows the construction and running of sequences of shell commands. It enables the setting of parameters to third-party and bespoke software in the software pipeline. After a pipeline is run all the results are moved to a folder in a timestamped results sub-directory.

As well as the command result files, three documents are created by Pipewrap. The first of these files is a shell script containing the shell commands executed. Running this script will re-run the pipeline. The second is a log .cvs file containing the list of commands executed along with the time taken and exit values. The third file is a colour-coded output log containing a dump of all the standard output and standard error produced by the Pipewrap commands. This file also contains the commands run, start and end times and exit codes.

The main Pipewrap display contains two columns. The first column has the list of commands in the pipeline to be run with each option appearing underneath. The second column contains the options' input values. Commands can be added either by using the Add button in the toolbar, the main menu or the context menu. This brings up a list of supported commands. Selecting one of these commands further brings up a dialog where the command options can be entered. The list of commands supported and the details of the options available for each of these commands is held in a database. One database table contains the list of commands and a second table contains a the list of options for each command. A new command can be supported by adding a row to the Biotools table (see table 4) and a row for each option to the ToolOptions table (see table 5).

Shell commands currently supported by Pipewrap are the third-party tools pIRS simulate, Stampy, VarScan somatic, samtools view, samtools sort, samtools index, samtools mpileup, samtools merge, PBSIM, Alchemy, BLASR, BWA bwasw, BWA index, BWA mem, BWA aln and BWA sampe along with

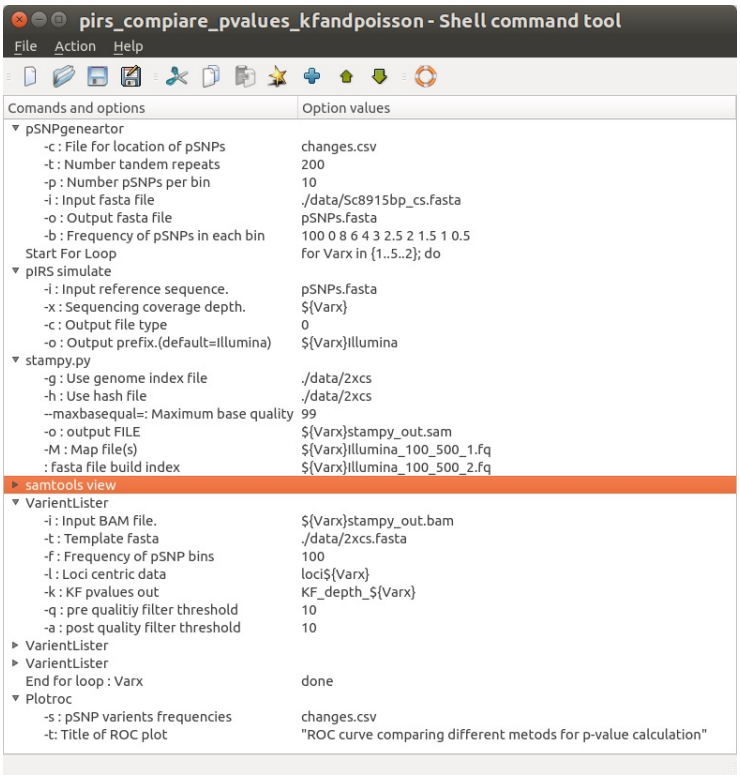

Figure 6: Screenshot of a Pipewrap pipeline in action.

pSNPgenerator, Simbas, AnalyseBax, VariantLister and Plotroc, which were developed within this project.

The Pipewrap user interface can be split into three areas: the main menu, toolbar and context menu on the central area. The actions available within these areas are listed below.

7.0.1 Pipewrap user actions

The following actions are available from the main menu. Key shortcuts, where available, are given in square brackets. Most actions are also available from the toolbar.

File |New Clears pipeline from central area, and resets pipe title to "New pipe" [Ctl N].

File |Open Brings up the file dialog to load a new pipe from disk. [Ctl O]

File |Save Saves the current pipe. [Ctr S]

File |SaveAs Brings up file dialog to save current pipe to disk. [Ctr Shft S]

File |<**filename**> Loads a pipe from the recent file list.

File |**E**x**it** Closes program. [Ctr Q]

Action |**Add** Adds a new command to pipe below current position.

Action |**Cut** Removes currently selected command or command option to a clipboard. Activates paste menu option. [Ctr X]

Action |**Copy** Copies currently selected command or command option to a clipboard. Activates paste menu option.

Action |**Paste** Inserts the command or options in Pipewrap's clipboard into the pipe below the current location. [Ctr V]

Action |**Execute** Executes the pipeline currently in the central window. Puts results in a time stamped directory in the results sub-directory.

Action |**Up** Moves selected command or options up one.

Action |**Down** Moves selected command or options down one.

Help |**help contents** Brings up help browser. [F1]

Action |**About** Brings up software information.

A context menu is available by left clicking from the central window. It provides the following actions.

Add Adds a new command to pipe below current position.

Cut Removes currently selected command or command option to a clipboard. Activates paste menu option. [Ctr X]

Paste Inserts the command or options in Pipewrap's clipboard into the pipe below the current location. [Ctr V]

Edit options Brings up the dialog for the currently selected command for editing.

Vary parameters Sets up a for loop bracketing everything from the current command to the end of the pipe. Currently only available when an option with integer values is selected. Brings up a dialog to enter the start, end and loop increment values.

Add input pipe Pipes standard output from previous command into current command's standard input.

Add output pipe Pipes standard output into the standard input for the command immediately below.

Execute Executes the pipeline currently in the central window. Puts results in a time stamped directory in the results sub-directory.

The second column in the central window is editable, allowing option values to be changed directly. To add a new option, however, it is necessary to use the Edit options from the context menu.

7.1 Implementation

The details of each command supported by Pipewrap are held in a database. PostgreSQL was used for development of this database while support was maintained for SQLite. The various dialogs used for data entry are then built up using the data in the database. The two database tables, encapsulating information on Pipewrap commands and their options respectively, are shown in tables 4 and 5.

Table 4: Biotools database table with list of supported commands

Name	Description
id_bio_tools	primary key
tool_name	Displayed name of tool
tool_tooltip	Help information describing tool
tool_shell_string	Shell string to use when calling tool
path_to_tool	Not used
tool_type	Not used

Table 5: ToolOptions database with options for each tool

Name	Description
id_tool_options	primary key
op_command	command line, key into Biotools table
op_string	Option value
op_data_typ	Datatype of option
op_short_description	Short description, label on auto-generated dialog
op_long_description	Longer description, tool tip on dialog
op_in_out	not used
op_group	not used
op_postion	Used to order options in command string
op_default_text	not used
op_default_int	not used

During development extensive use was made of the Qt examples (**?**). In

particular the menu, toolbar, previous file list, help browser and database connection examples were used.