1.) Some general facts about 454 Pyrosequencing

Three generations of 454 pyrosequencing with increasing read lengths:

<table>
<thead>
<tr>
<th>Generation</th>
<th>Year</th>
<th>Flows (cycles)</th>
<th>Read length (avg.)</th>
<th>Number of reads per run</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS 20</td>
<td>2005</td>
<td>168 (42)</td>
<td>100 base pairs</td>
<td>~250 000</td>
</tr>
<tr>
<td>GS FLX</td>
<td>2007</td>
<td>400 (100)</td>
<td>200-300 bp</td>
<td>~350 000</td>
</tr>
<tr>
<td>GS FLX Titanium</td>
<td>2008</td>
<td>800 (200)</td>
<td>400-500 bp</td>
<td>~1-1.2 million</td>
</tr>
</tbody>
</table>

File format:

“Standard flowgram format” (.sff), often two files per run

The technology in brief:

- Cyclic flowing of nucleotide reagents ("TACG – TACG – ...")
- Chemical reaction produces an observable light signal
- The strength of the light signal is proportional to the homopolymer length of the corresponding nucleotide
Nucleotides are flowed sequentially in a fixed order across the PicoTiterPlate device during a sequencing run.

During the nucleotide flow, hundreds of thousands of beads each carrying millions of copies of a unique single-stranded DNA molecule are sequenced in parallel.

If a nucleotide complementary to the template strand is flowed into a well, the polymerase extends the existing DNA strand by adding nucleotide(s).

Addition of one (or more) nucleotide(s) results in a reaction that generates a light signal that is recorded by the CCD camera in the instrument.

The signal strength is proportional to the number of nucleotides incorporated in a single nucleotide flow.

Figure 1: Sequencing reaction of the Genome Sequencer System. Millions of copies of a single clonal fragment are contained on each DNA Capture Bead.

Image Processing

The Genome Sequencer System software tracks the location of DNA carrying beads on a XY axis. Each bead corresponds to a XY-coordinate on a series of images. The signal intensity per nucleotide flow is recorded for each bead over time and is plotted to generate a flowgram. Each 10-hour sequencing run with the GS FLX Titanium series will typically produce over 1,000,000 flowgrams, one flowgram per read (Figure 2).

Figure 2: Genome Sequencer System Image processing overview.
2.) Data Characteristics

a) Each read consists of 800 flows (Titanium only)

4 flows = 1 flow cycle (TACG)

*lowercase letters indicate that the base has been trimmed

Example (from the blue line to the bottom): “tcagGTTTGTGGAGATGGTGA…”

- The first four bases of each read contain the “tcag” control tag and are trimmed.
- Raw data output and read summary is possible with “flower”:

  http://blog.malde.org/index.php/flower/

- Raw data output and splitting up sff files is possible with “sffinfo” and “sffile” (by 454)
b) Flow values follow a statistical distribution which...
- is almost symmetrical around the corresponding integer
- gets broader for longer homopolymer lengths
- gets broader towards the end of the read

c) Over- and under-calls arise from overlapping distributions
- substitution errors can only occur when an over-call follows an under-call or vice versa

d) The 454 software calculates a per-base quality score
- ~ probability that the base in question is not an over-call / that the homopolymer length has been determined correctly
- Algorithm uses flow value distributions

e) The 454 software performs several steps of whole-read filtering and read-trimming
- see "Genome Sequencer Data Analysis Software Manual" by Roche Diagnostics for details

---

For details in data characteristics see the Flowsim paper at the end of the handout.
3.) Clonesim & Flowsim

Available on:

http://blog.malde.org/index.php/flowsim/

(including a detailed description of how Flowsim works, see attachment 2)

Clonesim...
- creates an arbitrary number of random genome fragments from a fasta file
- can handle different clone size distributions (which influences the read length distribution)

Flowsim...
- creates .sff files from clonesim input
- draws flow values from either parametrical or empirical flow value distributions
- creates reads of given length (influenced by clone size and 454 generation)
- creates quality scores from different algorithms
- can simulate read degradation

More options to be implemented, e.g. paired ends
Clonesim / Flowsim parametrization:

Clonesim v0.2.7, copyright 2010 Ketil Malde

clonesim [FLAG] [FILE]
  simulate sequence cloning

-? --help=[FORMAT]  Show usage information (optional format)
-V --version        Show version information
-v --verbose        Higher verbosity
-q --quiet          Lower verbosity
-l --lengths=DIST   model for clone lengths (default=Uniform[400, 800])
-c --count=INT      number of reads to generate (default=10)

Available distributions (DIST):
  Uniform a b, Normal mu sigma, LogNormal mu sigma

Flowsim v0.2.7, copyright 2010 Ketil Malde

flowsim [FLAG] [FILE]
  simulate 454 pyrosequencing

-? --help=[FORMAT]  Show usage information (optional format)
-V --version        Show version information
-v --verbose        Higher verbosity
-q --quiet          Lower verbosity
-G --generation=GEN 454 generation to simulate (default=Titanium)
  --degradation=DIST model for degradation of the flow model
-m --model=FILE     empirical distribution for flow generation
  --qualitymethod=STRING method for calculating quality
  --discardfilters=DFILT discarding filters to apply
-t --trimfilters=TFILT trimming filters to apply
  --flowkey=VALUE    sequence key to start each read (TCAG)
-h --hplinput=FILE  input genome for HPL count estimate
  --flowlength=VALUE number of flow cycles to run
  --flowcycle=VALUE  sequence nucleotides in each flow cycle (TCAG)
-o --output=VALUE   output file

Generations (GEN): GS20, Titanium, EmpTitanium
Distributions (DIST): Uniform a b, Normal mu sigma, LogNormal mu sigma
Discarding filters (DFILT): ....
Trimming filters (TFILT): ....

4.) For further reading:


Attachment 1: Flowsim Paper
Characteristics of 454 pyrosequencing data—enabling realistic simulation with flowsim

Susanne Balzer¹, ², *, Ketil Malde¹, *, Anders Lanzén³, ⁴, Animesh Sharma² and Inge Jonassen² ³

¹Institute of Marine Research, PO Box 1870, N-5817, ²Department of Informatics, University of Bergen, PO Box 7803, N-5020, ³Computational Biology Unit, Bergen Center for Computational Science, Thorvaldensgate 55, N-5008 and ⁴Department of Biology, University of Bergen, PO Box 7803, N-5020, Bergen

ABSTRACT

Motivation: The commercial launch of 454 pyrosequencing in 2005 was a milestone in genome sequencing in terms of performance and cost. Throughout the three available releases, average read lengths have increased to ~500 base pairs and are thus approaching read lengths obtained from traditional Sanger sequencing. Study design of sequencing projects would benefit from being able to simulate experiments.

Results: We explore 454 raw data to investigate its characteristics and derive empirical distributions for the flow values generated by pyrosequencing. Based on our findings, we implement Flowsim, a simulator that generates realistic pyrosequencing data files of arbitrary size from a given set of input DNA sequences. We finally use our simulator to examine the impact of sequence lengths on the results of concrete whole-genome assemblies, and we suggest its use in planning of sequencing projects, benchmarking of assembly methods and other fields.

Availability: Flowsim is freely available under the General Public License from http://blog.malde.org/index.php/flowsim/

Contact: susanne.balzer@imr.no; ketil.malde@imr.no

1 INTRODUCTION

During the last few years novel sequencing technologies have been introduced. The platforms that are currently commercially available are marketed by Roche (454), Illumina (Solexa/Genome Analyzer), and Applied Biosystems (SOLiD), and they give new challenges for bioinformatics due to data volumes, short read lengths, and difference in errors and quality compared to traditional Sanger sequencing. So far, most bioinformatics methods available have been developed for Sanger sequencing data.

In this article, we characterize the data produced by the 454 system and in particular by its latest version named GS FLX Titanium (referred to as Titanium in the rest of the article). We analyze Titanium data sets from genomes for which the sequence has been determined. Specifically, we map each Titanium read to the reference strand in this well is extended with additional nucleotid(es) by a polymerase. This hybridization results in a reaction that generates an observable light signal which is recorded by a camera. The light intensity is converted into a ‘flow value’, a two-decimal non-negative number that is proportional to the length of a homopolymer run, i.e. it designates the number of nucleotides included in the flow, estimated by simply rounding the number to the closest integer (Margulies et al., 2005).

The term ‘noise flow values’ (in literature sometimes referred to as ‘negative flow values’, in practical terms being between 0 and 0.49) means that the light signal—although existing—is weak and judged not to result from a chemical reaction. A ‘positive flow value’ thus indicates incorporation of at least one base, and the number of bases (the homopolymer length) is determined from the flow value. Flow values for one bead (one read) can be used to plot a flowgram (Fig. 1a) from which the associated sequence can be determined.

The cyclically flowed nucleotides and the corresponding flow values build the basis for not only base calling, but also per-base quality score calculation (integrated in Titanium output). Obviously, the key to a correct base calling lies in the accuracy of the light signals. The 454 methodology differs from traditional Sanger sequencing in that substitution errors are a lot less frequent than insertions or deletions. Data properties have slightly changed over the three 454 generations (Roche Applied Science, 2008). We focus on the Titanium technology for all further calculations.

1.1 Pyrosequencing

The 454 pyrosequencing technology is based on sequencing-by-synthesis and consists in the cyclic flowing of nucleotide reagents (repeatedly flowing T, A, C, G) over a PicoTiterPlate™. The plate consists of approximately one million wells, and each well contains at most one bead carrying a copy of a unique single-stranded DNA fragment to be sequenced. When the flowed nucleotide is complementary to the template strand in a well, the existing DNA strand in this well is extended with additional nucleotid(e)s by a polymerase. This hybridization results in a reaction that generates an observable light signal which is recorded by a camera. The light intensity is converted into a ‘flow value’, a two-decimal non-negative number that is proportional to the length of a homopolymer run, i.e. it designates the number of nucleotides included in the flow, estimated by simply rounding the number to the closest integer (Margulies et al., 2005).

The term ‘noise flow values’ (in literature sometimes referred to as ‘negative flow values’, in practical terms being between 0 and 0.49) means that the light signal—although existing—is weak and judged not to result from a chemical reaction. A ‘positive flow value’ thus indicates incorporation of at least one base, and the number of bases (the homopolymer length) is determined from the flow value. Flow values for one bead (one read) can be used to plot a flowgram (Fig. 1a) from which the associated sequence can be determined.

The cyclically flowed nucleotides and the corresponding flow values build the basis for not only base calling, but also per-base quality score calculation (integrated in Titanium output). Obviously, the key to a correct base calling lies in the accuracy of the light signals. The 454 methodology differs from traditional Sanger sequencing in that substitution errors are a lot less frequent than insertions or deletions. Data properties have slightly changed over the three 454 generations (Roche Applied Science, 2008). We focus on the Titanium technology for all further calculations.

1.2 Use of flow values in data analysis

Although 454 sequences can be analyzed as Fasta files with standard bioinformatics tools, the flow values contain information that is not available in the pure nucleotide sequence. Consequently, several
Characteristics of 454 pyrosequencing data

(a) A 454 flowgram: cyclic flowing during one read. The light signal strengths (flow values) are directly translated into homopolymer runs. (b) Absolute frequencies of flow values (E. coli). Left: original data, no quality-trimming; right: quality-trimmed. The trimming algorithm enhances the separation of the homopolymer length distributions and levels out discrepancies between the nucleotides such that the curves for the four nucleotides are nearly identical.

Many groups have proposed algorithms to utilize flow values directly. This approach is referred to as operating in ‘flowspace’ as opposed to ‘nucleotide space’ and inhibits information loss. For example, the PyroNoise method (Quince et al., 2009) uses a maximum likelihood approach to decide whether a set of flowgrams is likely to result from one or several distinct underlying biological sequences. In an analogous manner, using Bayesian statistics, the PyroBayes method (Quinlan et al., 2008) determines the length of each homopolymer run as the most likely number of bases given the observed flow value. If the probability for an extra base exceeds a certain threshold, the extra base is added to the homopolymer run. This increases the number of insertion errors, but decreases the number of deletions and substitutions since it is intrinsic to 454 pyrosequencing that substitution errors can only arise from coherent over- and undercalls. This tendency to call more bases in homopolymer runs thus enables a higher SNP identification rate.

For small RNA discovery, direct mapping of flowgrams against a target genome (‘FLAT’, flowgram alignment tool) has been proved to be an efficient method (Vacic et al., 2008). It is also possible to achieve higher per-base accuracy rates in sequence assembly by building consensus sequences in flowspace from highly oversampled data (Huse et al., 2007; Margulies et al., 2005). Metagenomics is another field where the quality of 454-pyrosequenced data has received much attention (Gomez-Alvarez et al., 2009; Huson et al., 2007; Quince et al., 2009).

Studies have shown that there are several artifacts that heavily influence the processing of data for different purposes (Gomez-Alvarez et al., 2009; Huse et al., 2007), and especially methods that do not directly use flow values are sensitive to the characteristics of pyrosequencing data. For example, when matching 454 sequences with an indexing approach one can collapse all homopolymer subsequences to length one since pyrosequencing is likely to introduce errors in homopolymer lengths (Miller et al., 2008).

Especially for long homopolymers, many errors are caused by broad and overlapping signal distributions leading to ambiguous base calls, although there has also been work on improving 454 sequencing from the chemical aspect (Margulies et al., 2005). In addition to the correct determination of homopolymer lengths, the under- or over-calling of bases is especially critical for weak light signals (i.e. noise flow values). A flow value of 0.49 is treated as noise by the 454 base caller although it is almost as likely to originate from a single base call.

1.3 Simulating shotgun data

With Genfrag (Engle and Burks, 1994) and celsim (Myers, 1999), there have been earlier attempts to simulate shotgun read data, but, to the best of our knowledge, MetaSIM (Richter et al., 2008) is the only simulator that allows for generating 454 pyrosequencing data. MetaSIM targets Metagenomics. Internally, it uses parametric models for simulating flow values, but its output is Fasta files, and thus it is of limited use for applications that operate in flowspace.

2 FLOW VALUE DISTRIBUTIONS

One of the main challenges in 454 pyrosequencing is the correct determination of homopolymer lengths from flow values. The latter originate from a mixture of overlapping distributions. This is illustrated in Figures 1b and 3, where each distribution is assigned to one homopolymer length and one distribution to noise values. Incorrect homopolymer lengths lead to insertions and deletions during base calling (relative to the underlying biological sequence),
After having compared Titanium raw data from two different species, we characterize the distributions of flow values coming from each homopolymer length. We find our resulting empirical distributions to be almost symmetric around the corresponding integers, with relatively low standard deviation for short homopolymer runs. However, when analyzing data from the three 454 generations, we also found that the degree of symmetry varies between them. Quinlan et al. (2008) report a significantly higher insertion than deletion rate, which is consistent with an asymmetry in the tails of the distributions, but we found the asymmetry to decrease towards newer generation data.

### 2.1 Parametric versus empirical approaches

In earlier studies one has approximated flow values by normal, log-normal (Margulies et al., 2005) or non-central student’s t distributions (Quinlan et al., 2008). However, for our data the fit of these distributions is not satisfying (Fig. 3). An alternative is to use non-parametric empirical distributions estimated from real Titanium data for which reference sequences are available. By mapping 454 flowgrams to the matching genomic region, assigning each flow value to the corresponding true homopolymer length as known from the reference genome. Thus, we collected the flow values assigned to each homopolymer length distribution from 0 to 5, and the linear regression for these parameters based on normal distributions fitted to homopolymer lengths 1 to 5.

### 2.2 Sequence comparisons

After having compared Titanium raw data from two different species, *Escherichia coli* and seabass (*Dicentrarchus labrax*, referred to as *D. labrax*, respectively in the rest of the article), we decided to combine them—equally weighted—into one empirical distribution per homopolymer length. However, we also decided to include the four different nucleotide types in the same distributions since they appear to give rise to very similar distributions. In order to find the distribution of flow values that arises from one particular homopolymer length, we mapped Titanium flowgrams to a reference genome for the same organism, based on one Titanium plate each for an *E. coli* K-12 strain (Blattner et al., 1997) and *D. labrax* (Kuhl et al., 2010). We used BLAST (Altschul et al., 1990) to identify the location of reads that could be aligned unambiguously to one location on the genome, with default BLAST parameters, except for gap open and extend penalties, which were set to 1.

### 2.3 Calculation of empirical distributions

We aligned the flowgrams to the matching genomic region, assigning each flow value to the corresponding true homopolymer length as known from the reference genome. Thus, we collected the flow values assigned to each homopolymer length distribution from 0 to 5, as shown in Figure 3. For homopolymer lengths greater than 5, our data is sparse, and it is therefore better to approximate the real distributions by extrapolating parametric distributions from the shorter homopolymer lengths. Table 2 shows the observed mean and standard deviation of the empirical distributions for homopolymer lengths 0 to 5, and the linear regression for these parameters based on normal distributions fitted to homopolymer lengths 1 to 5.

### 2.4 Degradation and Noise

We find our resulting empirical distributions to be almost symmetrical around the corresponding integers, with relatively low standard deviation for short homopolymer runs. However, when analyzing data from the three 454 generations, we also found that the degree of symmetry varies between them. Quinlan et al. (2008) report a significantly higher insertion than deletion rate, which is consistent with an asymmetry in the tails of the distributions, but we found the asymmetry to decrease towards newer generation data.

Nevertheless, we can clearly observe two kinds of degradation: since standard deviation increases for increasing homopolymer lengths, these belong to broader distributions with overlapping tails, where the latter generally means a higher risk of under- and under-calls.

Second, analysis of the flow values associated with sequence parts that have been trimmed off (during standard 454 quality-trimming) indicates that 454 quality-filtering and trimming calibrates discrepancies between the four nucleotides and increases the separations of the distributions, involving deeper valleys.
Characteristics of 454 pyrosequencing data

Fig. 2. (a) Absolute frequencies of flow values by flow cycle. A total of 200 flow cycles of a Titanium run correspond to \(200 \times 4 = 800\) flows. The first two flow cycles contain the TCAG tag and are omitted here. Towards the end of a run, flow values tend to be further away from their ideal values (integers), but are obviously less in number because many values from later flow cycles have been trimmed away. (b) Standard deviation of flow values (difference in relation to their closest integer), by flow cycle. Standard deviation increases almost linearly. Only flow values \(<5.5\) were included.

2.5 Read lengths

The length of un-trimmed reads in 454 pyrosequencing is limited by either the number of flows (168 in GS20, 400 in GS FLX and 800 in GS FLX Titanium) or the length of the clones. The longest reads are thus obtained when the clone length exceeds the number of flows, such that the DNA strands in the well are extended until the very last flow cycle.

As quality decreases towards the end of a read, several filters are applied on the reads, which again gives a different read length distribution. We can thus distinguish between the distribution of clone lengths, the distribution of read lengths before filtering and quality-trimming and that after application of those filters. A detailed description of the filtering algorithms is given in the 454 manual (Roche Applied Science, 2008). As visible in Figure 2b, they eliminate (some of the) artifacts in the distributions by trimming low-quality flow values from the end of each read.

3 FLOWSIM—A SIMULATOR FOR 454 DATA

To take advantage of the empirical distributions, we implemented Flowsim, a simulator for pyrosequencing data.

3.1 Implementation of Flowsim

Given an input sequence in Fasta format, Flowsim selects substrings of this sequence with random position and strand, and generates a flowgram by converting the nucleotide sequences to sequences of homopolymer lengths. Each homopolymer length is then altered according to its flow distribution, where the latter is allowed to vary (degrade) with the flow position in the simulated read. To emulate degradation, we derived 20 different sets of empirical distributions from our mapping results (Fig. 3), where each of them represents 10 consecutive Titanium flow cycles, which sums up to 800 flow values.

The simulated flowgram is then analyzed to call nucleotide sequence and quality scores. Finally, all generated information is stored in an SFF file, similar to the ones produced by the 454 software.

One can further specify the number of desired output reads and also incorporate user-defined empirical distributions, either position-specific (degrading) or not.

3.2 Quality scores

It is crucial to assign a quality score to each called base, since sequenced bases are not filtered individually during quality-filtering and -trimming, but rather in the context of their reads. Quality scores are e.g. useful for assembly projects, although some assemblers do not use them. If they do, however, they might rely on them for incorporating Sanger reads since 454 quality scores are expressed as a phred equivalent (Margulies et al., 2005; Roche Applied Science, 2008). On the other hand, scores can also be used by assemblers built for Sanger sequences when assembling 454 sequences.

Although the method for determining quality has been described both for GS20 (Margulies et al., 2005) and Titanium (Brockman et al., 2008), the exact parameters are not known. Instead, Flowsim
Fig. 3. Empirical distributions (smoothed average of *E. coli* and *D. labrax*) on logarithmic scale. In gray: fitted (log-)normal distributions.

calculates the error probability (‘the base in question is an over-call’), using Bayes’ Theorem, and transfers it into a phred equivalent. Thus, the quality score corresponds to the true quality of the simulated base call, rather than to the quality the 454 software would produce for the same flowgram.

Flowsim currently supports two quality calling methods based on Bayesian statistics. One produces decreasing quality scores for the bases in a homopolymer, similar to GS20. The second produces a series of identical values for each base in a homopolymer, as in Titanium, but otherwise builds on the same Bayesian approach as the GS20 algorithm. Compared to the quality scores assigned to Titanium by the Roche analysis pipeline, our quality scores are lower. As GS20 appears to use a fixed table mapping each flow value to a set of qualities, there is also a third option of assigning qualities from a table derived from GS20 data.

Bayes’ theorem requires both the prior probability for each homopolymer length and the conditional probability for a flow value given a certain homopolymer length. In contrast to Margulies et al., we use both empirical priors (from the input Fasta file) and empirical conditional probabilities (from our empirical distributions). This allows us to assess the quality of our simulated data as accurately as possible. When position-specific empirical distributions are used in Flowsim, we also use these for quality score calculation.

3.3 Simulating data sets

We used Flowsim to generate synthetic data sets, using our empirical distributions as the flow model. Each of the 20 distributions was used for 10 flow cycles (40 flows), giving a realistic degradation of quality along the sequence. We also simulated data sets using 400 flow cycles, simulating a hypothetical 454 generation with twice the read length of the current Titanium generation. The *E. coli* genome (K-12 strain, GenBank ID: 49175990) was used as the input genome.

<table>
<thead>
<tr>
<th>Coverage</th>
<th>Real</th>
<th>200 cycles (simulated)</th>
<th>400 cycles (simulated)</th>
</tr>
</thead>
<tbody>
<tr>
<td>De novo-based N50 for E. coli</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>649</td>
<td>651</td>
<td>995</td>
</tr>
<tr>
<td>5</td>
<td>2406</td>
<td>7045</td>
<td>7623</td>
</tr>
<tr>
<td>10</td>
<td>23613</td>
<td>132,913</td>
<td>104,012</td>
</tr>
<tr>
<td>15</td>
<td>67,231</td>
<td>173,592</td>
<td>178,129</td>
</tr>
<tr>
<td>20</td>
<td>86,902</td>
<td>172,127</td>
<td>203,000</td>
</tr>
<tr>
<td>25</td>
<td>95,348</td>
<td>176,747</td>
<td>207,011</td>
</tr>
<tr>
<td>30</td>
<td>97,821</td>
<td>171,819</td>
<td>207,011</td>
</tr>
<tr>
<td>Reference-based N50 for <em>E. coli</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>895</td>
<td>1093</td>
<td>1681</td>
</tr>
<tr>
<td>5</td>
<td>8905</td>
<td>31,730</td>
<td>40,321</td>
</tr>
<tr>
<td>10</td>
<td>76,607</td>
<td>207,827</td>
<td>234,849</td>
</tr>
<tr>
<td>15</td>
<td>110,013</td>
<td>207,856</td>
<td>2496,857</td>
</tr>
<tr>
<td>20</td>
<td>118,387</td>
<td>207,740</td>
<td>2497,013</td>
</tr>
<tr>
<td>25</td>
<td>161,266</td>
<td>207,899</td>
<td>2497,058</td>
</tr>
<tr>
<td>30</td>
<td>177,489</td>
<td>207,845</td>
<td>2724,990</td>
</tr>
</tbody>
</table>

3.4 Simulation results

We have performed both de-novo and reference-based assembly using Newbler assembler version 2.3 (Roche), approximating various coverage (1x, 5x, 10x, 15x, 20x, 25x and 30x). A simulation with 200 flow cycles shows ~1% inferred error, while 400 flow cycles result in an error rate of ~0.8%, which is the same as for the real data (Titanium, i.e. 200 flow cycles).

Our results indicate that Flowsim can be useful to estimate the quality of an assembly that can be expected from using Titanium to shotgun sequence a genome. However, the assemblies resulting from our simulations were consistently better in terms of contig sizes (through the N50 summarizing statistic, see Table 3) for the simulated data sets than for the real ones. This may partly be due to all simulated reads coming from the reference genome and thus avoiding strain-specific discrepancies, which leads to the fact that 100% of the reads for 200 and 400 flow cycle simulations can be mapped back to genome, while real data reach only ~98.7% for all studied coverage values. There may also be other factors such as possible biases in terms of genome coverage in the experimental protocols used to generate the shotgun libraries for Titanium sequencing. Further work will include exploring such biases and other sources of variability as well as characterizing their influence on the simulation accuracy of Flowsim. Also Flowsim will be extended to include simulation of paired-reads, which will be of high value for simulation and planning of projects for de-novo whole-genome sequencing.

4 DISCUSSION

This study aims to sketch the opportunities that arise from analyzing pyrosequencing raw data, culminating in the use of empirical distributions. The empirical distributions give us a very realistic picture of the underlying characteristics of the light signal values that are later translated into DNA sequences. In contrast, earlier approaches to modeling flow data have built on parametric
distributions, and the same distributions were used for whole reads, without respect to flow or read positions.

Our findings and the empirical distributions are based on large amounts of data from three different species (E. coli, D. labrax, Gadus morhua), four sequencing labs, both shotgun and paired-end reads with different gap sizes. The empirical flow value distributions are very similar, and we have not observed any factors which influence the shape of the distributions apart from the 454 generation. Thus, we have a good reason to believe that the distributions used in Flowsim are representative.

The flow values that result from 454 sequencing exhibit many interesting characteristics and artifacts, and we do not address them all here. Some of these are generation-specific, some of them have remained stable over the years, and some of them only appear on one certain plate, for one certain species or in one lab. One known artifact, exact or almost-exact duplicates, has been not only described for metagenomics in the literature (Gomez-Alvarez et al., 2009), but we also observed them in shotgun sequences for E. coli and D. labrax.

We do emulate the degradation in empirical flow distributions, and we also calculate the corresponding quality scores. In contrast, we neglect some of the artifacts that we have observed in the empirical distributions, but are not able to interpret properly yet, such as for example: shifts in peaks that lead to systematic over- or under-calls, jumps, neighboring peaks, i.e. subpeaks around the next or preceding integer. These are particularly strong for the noise distribution (with a neighboring peak around 1) and the 1-distribution (with neighboring peaks around 0.1 and 2), but the values causing these peaks are not many in number. Analyzing the corresponding data including the related alignments we found that the subpeaks are likely to be caused by real biological differences. This will be explored further in a separate study. In this context, we also performed a weak smoothing process that helped to reduce subpeaks and jumps.

Furthermore, the 454 image analysis software implements a set of quality filters that sets trimming coordinates to identify the high-quality part of each read. In addition, some reads are eliminated entirely based on quality metrics. Although these filters are documented (Roche Applied Science, 2008), the documentation is not sufficient to re-implement them, and the current version of Flowsim does not attempt to simulate them. We hope to address this in a future release (Fig. 4).

In conclusion, our simulator produces sufficiently realistic 454 files as we model all important phenomena that we have observed. Furthermore, Flowsim allows the user to specify many of its parameters, making it adaptable to new real or hypothetical 454 generations.

ACKNOWLEDGEMENTS

The authors wish to thank Dr Alexander Sczyrba, DOE Joint Genome Institute, Dr Richard Reinhardt, Max Planck Institute for Molecular Genetics, Berlin, for kindly providing us with Titanium raw data and Dr Christopher Quince, University of Glasgow, for the fruitful discussions. Notur is acknowledged for access to the Titan cluster in Oslo.


Conflict of Interest: none declared.

REFERENCES

Roche’s 454 sequencing is one of the new (or “next generation”) sequencing technologies that are currently available. FlowSim is a program, written in Haskell of course, that simulates the 454 sequencing process in order to produce synthetic sequencing data as close to the real thing as possible.

**Background**

Of existing sequencing simulators, MetaSim is the only one I’m aware of that targets pyrosequencing, but it is limited to producing Fasta formatted output. FlowSim generates native SFF files, and is thus suitable for experimenting with software that operates directly on flowgrams. (Of course, the SFF file embeds nucleotide sequence and quality values as well, so you can still use it with tools that expect Sanger sequences.)

Previous efforts usually model flow values from parametric distributions (usually normal distributions around the integrals), FlowSim provides a variety of options, including user-supplied parametric or empiric distributions, with or without degradation.

**Features and examples**

The basic configurations in FlowSim are available as generations, and specified with -G option. So to simulate a GS20 run, it should suffice to say:

```
flowsim -G GS20 input_genome.fasta -o gs20.sff
```

In addition to “GS20", you can also specify “Titanium”, which uses a different set of defaults. If you, being the type not to trust documentation, check the output of

```
flowsim --help
```

you might notice two things. First, the built-in help text lists the available distributions, and second, there’s also a distribution called “EmpTitanium”. More about that later.

**Generating the input using ‘clonesim’**

Previous versions of flowsim let you specify the number of reads to generate using a -n option. The clone generation has now been factored out into a separate program, and flowsim now just generates one read per input sequence. So, in order to generate a full GS20 run of, say 120K sequences, you must do

```
clonesim -c 120000 input_genome.fasta | flowsim -G GS20 -o gs20.sff
```

The default is to generate 10 reads. Clonesim has additional options to set the distribution of clone sizes, and flowsim lets you override the flowgram length, the sequence of flows, and the initial key (four letter prefix that is added to all the clones before sequencing). You can also specify verbosity (-v) to get some more output while flowsim is running. Again, –help is your friend.

**Parametric and empirical flow distributions**

The basic “engine” of FlowSim is the perturbation of homopolymer lengths to generate flow values. As you probably should be aware before reading this, 454 sequencing produces a sequence of flow values, each corresponding to a homopolymer (i.e. a run of the same nucleotide) length. The idea is that for a given homopolymer length, you get a number drawn from some stochastic distribution more or less centered around the homopolymer length. Base calling rounds to the nearest integer, so if this distribution is too wide (and it is), you get sequencing errors. The models are built in, but the user can specify a file containing an empirical model to use:
flowsim -G Ti --model=emp.txt -o emp.sff

This file consists of whitespace-separated columns of numbers. The first column is ignored, the second is the integers from zero and up, representing flow values times a hundred, and then columns 2+ contain probabilities for getting each flow value, for homopolymers of length 0+ (so column 2 is the noise flows, column 3 represents homopolymers of length 1, and so on). For homopolymer lengths larger than the amount of columns, FlowSim will fall back to the distribution specified by the generation (Titanium in the example).

Another option is to specify a list of parametric distributions:

./flowsim -G Ti -v --model="LogNormal -2.5 0.2,Normal 1 0.2" -o degr.sff

This will use the default Titanium log-normal distribution parameters of -2.5 and 0.2 for the noise flows, and a wider (sd = 0.2, vs the default of 0.08) distribution for homopolymer length 1. Anything else will fall back to Titanium defaults. The format is a comma-separated list of distributions, and you may of course specify distributions for as long homopolymers as you wish.

The “EmpTitanium” is a special generation containing a set of twenty empirical distributions for Titanium. It uses a sequence of 20 empirical distributions, each derived from segments of ten flows from Titanium data. Using this switches off degradation (see below).

Degradation

In real 454 data, quality is quite high at the start of each read, but quite poor at the end. The flow model is allowed to degrade as the read simulation progresses, this is modeled by drawing a number from a separate “degradation distribution”, and cumulatively increasing the standard deviation of the flow distribution with this number. The Titanium default is to use a normal distribution with mu 0.002 and sd 0.001. This means that the expected increase in sd for the flow distributions is about 0.1, but also that quality has a small, but non-zero chance of improving, at least locally.

The degradation is user-configurable, to produce quickly-degrading reads, you could do:

flowsim -G Ti --degrad="Normal 0.004 0.001" -o output.sff

Quality methods

Base call quality values are, of course, an important by-product of sequencing, and used by a number of tools. Although GS20 base calling appears to be a simple lookup table (with the flow values as index), Titanium is a more complicated matter, and not entirely adequately described outside of Roche’s sources. But since we know the real distributions used for generating flow values, we don’t need to emulate this to get accurate qualities. We can instead calculate the quality values based on the actual probability of the call being correct, using Bayes’ formula.

There are two ways to do this: the first (GS20-style) produces a decreasing sequence of qualities for each homopolymer, based on the probability of this being an overcall or not. The second calculates the probability of the homopolymer length being correct, and repeats this for the whole homopolymer – i.e. all qualities within a homopolymer are the same. We think this is the intent behind Titanium quality calling, but we’re not sure (and we’ve seen some cases where the quality varies within a homopolymer, so it might be wrong).

There’s a third way, too, using a GS20-based lookup table.

Quality filters

454 implements a set of quality filters, some that trim sequences by quality by setting the qual_clip_right parameter, and some that discard sequences of too low quality. Although most of these are implemented in the biolib, the documentation isn’t clear enough to ensure that they are implemented correctly, and for
some the documentation is too difficult to understand to implement at all. The facilities are available in FlowSim, though, so perhaps in the future this will be better supported.

Implementation

FlowSim has grown from humble beginnings into a somewhat complex beast, but hopefully the individual modules are digestible. The perhaps easiest parts are the modules representing Generations, currently Titanium.hs, EmpTitanium.hs and GS20.hs. These basically just populate the data structure defined in GenBase.hs with what is hopefully reasonable defaults.

In order to support the variety of statistics distributions, I implemented (yet another) Statistics module. This centers on a Distribution, which is an algebraic data type, and can contain either Normal, LogNormal, Uniform, or Empiric distributions. A function ‘sample’ pulls a random value from this, using the Random monad. Speaking of which, this provides a nice way to work with stochastic values. In retrospect, I guess I could have used one of the available statistics/random modules, the Statistics module is easily the most troublesome piece of s..oftware I've written for some time. Programming with indexes is just evil.

The configuration work is done in Config.hs, which uses Neil Mitchell’s CmdArgs module for command line parameters. This is very nice in that you give it a data structure representing your configuration, and the module works out all the magic, including naming your parameters. Sometimes the magic breaks, and you end up with errors that you normally wouldn't expect in a statically typed language. But we can forgive it for being quick and dirty, as long as it's mostly quick. Which it is.

Performance

I briefly tested performance on my 1.3GHz HP laptop. I can generate a simulated GS20 run — 120K reads with 168 flows each — in about 18 minutes, or 6-7000 reads per minute. Titanium is a bit slower, mostly because of longer reads (800 flows). Experimenting with generating 1K, 2K, 4K and 8K sequences reveals a quite linear growth (5min 10s for 8K) of about 1500 reads per minute, so a run of 400K reads is expected to take a bit over four hours. There also seems to be a constant memory consumption of about six megabytes resident, so if it's too slow for you, you are free to run several in parallel.

Todo

Some fairly obvious features are still missing, e.g. paired ends is important for de-novo sequencing and should be supported. CDNA (i.e. transcript sequencing) might be something to look into. Chimerae, duplicate clones, etc.