Determining Error Biases in Second-Generation DNA Sequencing Data

Project Proposal

Neza Vodopivec
Applied Math and Scientific Computation Program
nvodopiv@math.umd.edu

Advisor:
Aleksey Zimin
Institute for Physical Science & Technology
alekseyz@ipst.umd.edu

ABSTRACT

I will study base substitution error biases in Illunima data. I will compute how often each of the twelve substitution errors occur and determine which type of error is most dominant for bases with a given quality score. I will also examine how the frequency of substitution errors changes over intervals of the genome as a function of their GC content. I will then determine the most common types of substitutions that occur for each GC content.

INTRODUCTION

Sanger sequencing technology revolutionized genomics, providing insight into the DNA of organisms from simple viruses and bacteria, to humans and other mammals. This technology, however, is expensive – the cost to produce the sequencing data for the chicken genome, for example, was over ten million dollars (Dalloul *et al*., 2010). The high cost and low throughput of Sanger data put a big constraint on the number of genomes that could be assembled.

The emergence of second-generation sequencing technologies is making it possible to sequence and assemble many more organisms. This new data is much less expensive: the turkey genome, comparable in size to the chicken genome, for example, was forty times cheaper to sequence with second-generation data (Dalloul *et al*., 2010). With the invention of these new technologies, it has even become feasible to initiate a 10K Genome project, whose aim is to assemble ten thousand genomes of vertebrate species (Genome 10K Community of Scientists, 2009). However, there is a caveat: while second-generation data is cheaper and much faster to produce, it comes with its own challenges. Each fragment of data sequenced is much smaller, generating extremely short strings of DNA 'letters' (the nucleic acids A, G, C, and T). This makes assembly significantly more difficult.

Second-generation data comes with additional complications. While no sequencing technology produces completely reliable data, errors in the new data are particularly problematic. Illumina, currently the most widely used of the second-generation technologies, has estimated error rates of 0.6% to 1.0% (Dohm *et al*., 2008). Moreover, these errors do not occur randomly. Illumina data has biases in the types of errors that are likely to be present and where they are likely to occur. While the insertion and deletion of a base, or letter, is relatively rare, a substitution of one letter for another occurs much more frequently.

Knowing the nature of base substitution error biases can produce a better assembly. A preliminary step in the assembly routine is data pre-processing, whose fundamental component is the detection and correction of errors in the data. Much work has gone into improving error-correction techniques. Among this work is research examining error biases in Illumina data, involving the substitution of one letter {A, G, C, T} for another. Dohm *et al*. (2008) determined that the most frequent type of base substitution error is an A to a C and the least frequent one is a C to a G. Kelly *et al*. (2010) found that dominant substitution errors also depend on a base's quality score, the probability that base was sequenced correctly.

In this project, I study the frequency of various types of substitution errors in Illumina sequencing data. I explore how these substitution errors vary with respect to the quality scores assigned to each base. I also examine which types of errors are dominant in parts of the genome with different GC contents, the percentage of the letters that are either G or C.

DATABASES

I will use Illumina data fragments for the genome of the Rhodobacter sphaeroedes bacteria as reported by the sequencing center. Each data fragment, called a 'read', is a sequence of letters from the set {A,C,G,T}, containing one letter for each base. Each base is assigned a quality score $q \in \{0,\ldots,40\}$ where the probability *P* that the base was sequenced incorrectly is P= *10^(-q/10).* I will also use the 'finished' sequence for the Rhodobacter genome. The finished sequence is the final assembled sequence with all gaps and ambiguities resolved experimentally using a chemical process. I will assume that this finished sequence is the 'ground truth', that is, the true genome sequence for the bacteria.

I will use additional databases (faux data, published results, and similar data for Staphylicoccus aureus) for validation and testing.

Formally, my input data for Rhodobacter sphaeroedes bacteria will consist of the following:

- Two matrices, R and Q, each (n x l), containing read sequences and quality scores, respectively, where l=101 is the read length and n=2,000,000 is the number of reads.
- A string (vector) of letters of length ~4.6M that represents the finished genome sequence for the bacteria.

APPROACH

This project will be split into the following three major parts:

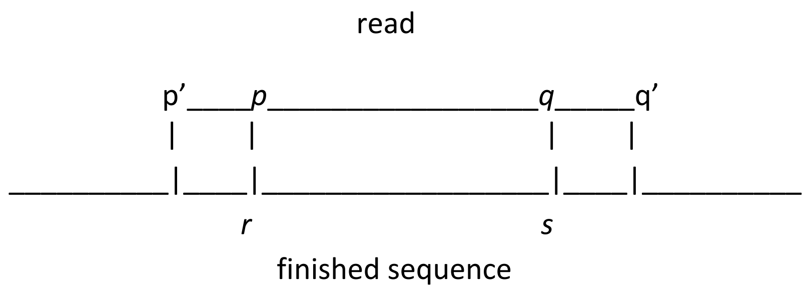
1. **Annotating the Reads.** I will align the reads to the finished sequence and find out where the substitution errors are.
2. **Determining the Most Frequent Errors as a Function of Quality Scores.** I will construct a substitution error Matrix E_q for each quality score *q* and use it to determine the most frequent types of errors for that quality score.
3. **Determining the Most Frequent Errors as a Function of GC Content.** I will construct a substitution error matrix E_g for various GC contents, the percentage *g* of the sequence that consists of the letters G and C. I will use each matrix to determine the frequencies of different types of substitution errors for each GC content available.

Part 1: Annotating the Reads

In this part of my code, I first determine the best global alignment of each read to the finished sequence. Next, I find the best local alignment by identifying all insertion and deletion errors on the read in order to align its individual bases onto the finished sequence correctly. Finally, I compare the corresponding bases in the aligned sequences to determine the substitution errors. I create annotated reads marking these substitution errors.

I begin by aligning each read to the finished sequence. I will determine the preliminary (global) alignment positions of reads on the finished sequence using Nucmer (Delcher *et al.*, 2002).

Figure 1: Aligning a read to the finished sequence.



Nucmer outputs coordinates (*p,q*) on the read and (*r,s*) on the finished sequence where a read aligns. It provides the general (global) location where the read maps. However [p. q] and [r, s] may not be the largest possible intervals where the sequences align. Such intervals could stretch as long as |*q'-p'*|, the length of the read. Nucmer doesn't provide any information about possible insertions and deletions of bases in the alignment.

In order to (locally) align individual bases on the read to their correct corresponding bases on the finished sequence, I must identify any insertions and deletions of bases in the read. Marking these insertion and deletion errors also prevents them from being erroneously classified as substitution errors. I will use the *Smith-Waterman alignment* algorithm (Smith *et al*. 1981).

First, I will build the Smith-Waterman alignment matrix A. Let A be an m x n matrix and let a_i be the ith letter in string *a* and b_j be the jth letter in string *b*.

Define A (i,0) = 0 for $0 \leq i \leq m$ and A (0,j) = 0 for $0 \leq j \leq n$.
For all $1 \leq i \leq m$ and $1 \leq j \leq n$,

define $A(i,j) = \max \{ A(i-1, j-1) + s_{ij} , A(i-1, j) - d, A(i, j-1) – d, 0\}$,
where s_{ij} is the 'match reward' if $a_i = b_j$ and the 'mismatch penalty' if $a_i \neq b_j$, and d is the 'insertion/deletion penalty'. I experiment with different values for rewards and penalties.

After I have created matrix A, I will identify $k = \max \{ A(i,j) \}$. Starting at entry k, I will backtrack until I hit an entry that equals 0. My final alignment is determined by the path I trace through the matrix A: a top-down jump indicates a deletion, a left-right jump indicates an insertion, and a diagonal jump indicates that bases a_i and b_j align (although they need not match).

While the Smith-Waterman alignment algorithm gives information about the insertion and deletion of bases, it does not distinguish between matching and non-matching aligned bases. The aim of my project is to study the mismatches, or base substitutions. Hence my last step is to identify the base substitutions from each alignment and record them on an accompanying read. I will compare the corresponding bases in the aligned sequences to determine the substitution errors. Create annotated reads marking these substitution errors. The set of accompanying reads are my error-annotated reads (my deliverable).

Part 2: Determining the most frequent errors as a function of quality scores

For part 2 of the project, I will use the annotated reads that I created in part 1 as well as the quality scores assigned to the original reads.

Let (x_1, x_2, x_3, x_4) be the event that the letter (A, C, G, T) is marked at base *h* in the original sequence. Let (y_1, y_2, y_3, y_4) be the event that the letter (A, C, G,T) is marked at base *h* in the annotated sequence.

For each quality score *q*, I create a 4 x 4 Substitution Error Matrix E_q , with entries $e_q(i,j) = \sum_h y_j \mid x_i$. For example, $e_{q(1,2)}$ is the frequency of an A $\rightarrow$ C error, i.e. A is erroneously read as a C.

I then determine the most dominant error type for each quality score q (my deliverable).

Part 3: Determining the most frequent errors as a function of GC content

I will compute the GC content, that is the percentage of bases whose letter is either a G or C, for each 200-base interval on the finished sequence. There are N-200 such intervals, where N is the number of bases in the finished sequence. I will categorize each interval by its GC content and study the errors for each GC category.

First I will study how the rate of substitution errors varies with GC content. I will plot the frequency of (combined) substitution errors as a function of a region's GC content (my deliverable). Next, I study how individual types of substitution errors vary with GC content by creating a matrix similar to the one above.

Once again, let (x_1, x_2, x_3, x_4) be the event that the letter (A, C, G, T) is marked at base *h* in the original sequence. Let (y_1, y_2, y_3, y_4) be the event that the letter (A, C, G,T) is marked at base *h* in the annotated sequence.

For each available GC percentage g, I create a 4 x 4 Substitution Error Matrix E_g of errors contained in intervals with a GC content equal to *g.* This matrix has entries $e_q(i,j) = \sum_h y_j \mid x_i$. For example, $e_{(1,2)}$ is the frequency of A $\rightarrow$ C errors for all intervals with a GC percentage of g.

I determine the most frequent error type for each GC percentage available (my deliverable).

Possible Problems

- The execution times for the Smith-Waterman algorithm may be a problem. I will compute an alignment for each read, which requires O(*l*^2) operations. I will compute Smith-Waterman alignments in parallel.
- The Smith-Waterman alignment algorithm finds all local alignments. There may be multiple parts of a read aligning well, with non-aligning gaps in between. I will have a gap penalty.

IMPLEMENTATION

My code will be implemented in Perl. I will use outside software, Nucmer, to perform the preliminary alignment of reads onto finished sequence (Delcher *et al.*, 2002). My hardware consists of an AMD Opteron computer with 32 cores and 256 GB of RAM.

Part 1 of the code will be parallelized.

VALIDATION

I will validate each part of my code individually.

Part 1: Annotating the Reads

I will generate a database of faux reads, each k bases long, by reading off subsequences from random locations on the finished sequence. Next, I will introduce random substitution errors to the faux reads and record these errors onto accompanying error–annotated faux reads. Finally, I will run part 1 of the code using the faux reads as input. If my software works correctly, the annotated reads produced by the code should be exactly the same as the error-annotated faux reads.

Part 2: Determining the Most Frequent Errors as a Function of Quality Scores.

I will continue to use the faux reads and faux annotated reads created to validate part 1 as my input data. Additionally, I will assign each base in the finished sequence a quality score $q \in \{0,\ldots, 40\}$ uniformly at random. I will run part 2 of the code with this data. In my output, the expected value for each entry in a given row is equal *in all matrices* since letter substitutions and quality scores were chosen at random.

Part 3: Determining the Most Frequent Errors as a Function of GC Content.

I will again use the faux reads and faux annotated reads created in part 1. This time, however, I will assign each 200-base interval in the finished sequence a GC content uniformly at random. I will run part 3 of the code with this data. The plot created should show no correlation between GC content and frequency of substitution errors. As above, the expected value for each entry in a given row is equal *in all matrices* since letter substitutions and GC content were chosen at random.

TESTING

In order to test my code for these potential weaknesses, I will run my finished software with data from another genome, *Staphylococcus aureus.* This genome is smaller than the *Rhodobacter* sphaeroedes and has a much lower GC content. Moreover, the two genomes were sequenced at different Illumina sequencing centers. I will also compare my results with published studies. (Dohm *et al.*, 2008*; Kelly* et a*l.,* 2010).

PROJECT SCHEDULE/MILESTONES

October: Learn how to use Nucmer software. Use Smith-Waterman algorithm to align reads and identify insertions/deletions.

November: Annotate reads for substitutions. Validate part 1 of the code.

December: Prepare mid-year presentation.

January: Create error matrices as a function of quality scores. Validate part 2 of code.

February: Compute GC contents on intervals. Compute error frequencies. Create error matrices as a function of GC content. Validate part 3 of code.

March: Create all tables and plots. Parallelize part 1 of the code.

April: Write final project document and prepare presentation.

DELIVERABLES

1. Finished code.
2. *Rhodobacter* sphaeroedes reads annotated for insertion, deletion, and substitution errors.
3. Table of the most frequent error type for each quality score.
4. Plot of the frequency of the combined substitution errors as a function of a region's GC content.
5. Table of the most frequent error type for each available GC content.

REFERENCES

Dalloul RA, Long JA, Zimin AV, Aslam L, et al. **Multi-Platform Next-Generation Sequencing of the Domestic Turkey (*Meleagris gallopavo*): Genome Assembly and Analysis.** PLoS Biol. 2010, 8(9):e1000475.

Delcher AL, Phillippy A, Carlton J, Salzberg SL: **Fast algorithms for large-scale genome alignment and comparison**. Nucleic Acids Res. 2002 Jun 1;30(11):2478-83.

Dohm JC, Lottaz C, Borodina T, Himmelbauer H: **Substantial biases in ultra-short read data sets from high-throughput DNA sequencing.** *Nucleic Acids Res* 2008, **36:**e105+.

Genome 10K Community of Scientists*.* **Genome 10K: a proposal to obtain whole-genome sequence for 10 000 vertebrate species.** *J. Heredity* 2009, 100*:*659-674*.*

Kelley DR, Schatz MC, Salzberg SL: **Quake: quality-aware detection and correction of sequencing errors.** *Genome Biol* 2010, **11**(11)**:**R116.

Smith, TF and Waterman, MS: **Identification of Common Molecular Subsequences.** *Journal of Molecular Biology* 1981, **147**: 195–197.