Review of **Genomic Perl**
From Bioinformatics Basics to Working Code
Rex A. Dwyer
Publisher: Cambridge University Press, 2003
ISBN: 0-521-80177-X

Review by:
Raymond Wan (*rwan@cs.mu.oz.au*)
University of Melbourne, Australia

1 Overview

Genomic Perl by Rex A. Dwyer covers several topics in bioinformatics in the context of the Perl language. According to the preface, the book selects some topics in bioinformatics, and presents them to the intended audience – a student in an "upper year level undergraduate or graduate level course".

Overall, I found the book interesting to read, but I believe it should be read with other resources available for reference.

2 Summary of Contents

The book consists of 17 chapters and appendices. While certain chapters build on the work from previous ones, each is self-contained and follows a fixed formula. The first half of a chapter provides the background in biochemistry that is essential to the chapter. Then, a Perl program is built up during the remainder of the chapter which addresses the problem, with difficult parts explained in the text. Every chapter has one to around ten questions for the student, with no solutions provided. The difficulty of the questions range from drawing a suffix tree for a word to extending the functionality of the Perl program from that chapter. Every chapter concludes with a bibliography. A brief summary of each chapter is as follows.

1. **The Central Dogma.** The book begins with an introduction to biochemistry, with emphasis on DNA, RNA, proteins, and the relationships between them. A Perl program that outputs the proteins that are encoded by a given DNA sequence is developed.

2. **RNA Secondary Structure.** An RNA molecule physically exists in three dimensional space, but is described at various levels. The nucleotides of an RNA sequence are written down as a sequence of characters at the lowest level. At the next level (also called its secondary structure), the molecule folds so that pairs of nonadjacent nucleotides form hydrogen bonds. Two programs examine how the sequence folds, while demonstrating recursion and dynamic programming.

3. **Comparing DNA Sequences.** The alignment of pairs of DNA sequences in the presence of errors is an example of approximate string-matching problems. A Perl program is given which addresses global alignment using the Needleman-Wunsch algorithm, another example of dynamic programming.

4. **Predicting Species: Statistical Models.** Probability, information theory, and entropy form the basis for a simple program which predicts the probability a DNA strand came from a particular species.

5. **Substitution Matrices for Amino Acids.** Substitution matrices extend the alignment techniques of Chapter 3 for proteins which differ due to evolution. The design of a Perl program which computes substitution matrices is shown.

6. **Sequence Databases.** Biological sequences are stored in public databases in several formats. A program using Perl's limited object-oriented facilities is developed which reads and processes the GenBank data format.

7. **Local Alignment and the BLAST Heuristic.** Chapter 3 describes an algorithm for aligning two entire sequences. In contrast, "local alignment" refers to the alignment of substrings of two sequences. The Smith-Waterman algorithm and BLAST heuristic are implemented in Perl for this purpose.

8. **Statistics of BLAST Database Searches.** The previous chapter is extended by looking at the statistics associated with aligning using the BLAST heuristic.

9. **Multiple Sequence Alignment I.** Another extension to Chapter 3 looks at the problem of aligning more than two sequences at a time by first extending the Needleman-Wunsch algorithm and then investigating incremental strategies, with a brief excursion into NP-completeness.

10. **Multiple Sequence Alignment II.** The discussion that began in the previous chapter continues by looking at how the efficiency of the algorithms for multiple sequence alignment can be improved.

11. **Phylogeny Reconstruction.** Phylogenies are trees which depict the course of evolution of several species. A Perl program is developed which creates a tree from a list of protein sequences.

12. **Protein Motifs and PROSITE.** Protein motifs are short sequences, each with a known biochemical function. PROSITE is a database of these motifs. Perl programs are implemented which first process motifs in the PROSITE file format, and then build a suffix tree so that all entries in the database can be efficiently compared with a sequence.

13. **Fragment Assembly.** Fragment assembly is an example of determining the shortest common superstring. That is, given a list of substrings representing fragments of a DNA sequence, what is the longest string that contains every substring in the list? A simplified version of the PHRAP program for addressing this problem is created in this chapter.

14. **Coding Sequence Prediction with Dicodons.** Statistical methods can be employed for finding new genes in a DNA sequence when a training set of known genes is available.

15. **Satellite Identification.** This chapter shows how identifying satellites, or tandem repeats in DNA, can be done in Perl. Both DNA fingerprinting and the reconstructing of phylogenies can benefit from these techniques.

16. **Restriction Mapping.** The restriction map of a DNA sequence is a list of locations where a restriction enzyme is known to cut the sequence. Creating a restriction map is one way of analysing a long sequence of DNA. Two techniques for achieving this in Perl are given, one which assumes the data set to be perfect, and another which allows for imprecise data.

17. **Rearranging Genomes: Gates and Hurdles.** One DNA sequence may contain the same genes as another, but in a different order. The number of steps required to transform the first sequence to the second can be used to determine how they are related, with respect to evolution.

18. **Appendices.** The appendices present a program for drawing the two dimensional RNA secondary structure diagrams (Chapter 2), some ideas for reducing space usage during the aligning of sequences, and a Perl solution to the *disjoint sets problem*.

3 Opinion

I approached this book with a few years of experience with Perl, and only limited knowledge of biochemistry. There were several aspects about the book which I liked. First, the structure of each chapter was simple and easy to follow. Second, most chapters are short and can cover the topic in just over 10 pages. That is, rather than going into detail with only a few topics, someone reading this book is given a taste of 17 topics. Third, while the bibliography of each chapter is short, only the most relevant citations are given.

While the theme of this book is bioinformatics, readers of this column will be interested that some time is spent on the underlying algorithms. Dynamic programming, recursion, recurrence relations, NP-completeness, regular expressions, and suffix trees are sprinkled throughout the book. The time and space complexity of algorithms are also occasionally given.

However, there are some parts of the book that did not appeal to me. Sometimes, the problem descriptions are too brief, and I often found myself asking questions whose answers were not available, even though they were not crucial.

For example, in the first chapter, several definitions form the foundation of the introduction to biochemistry. But, due to the limited amount of space, a clear picture of how DNA, RNA, and proteins relate to each other is difficult to form. While a complete understanding of biochemistry is not needed to code a solution in Perl, the short background given in each chapter may cause some readers to yearn for more. Likewise, if someone does not have sufficient knowledge of Perl, then the latter half of each chapter may be hard to follow. The author has decided to spread coverage of Perl syntax throughout the book, rather than dedicating an introductory chapter or appendix to it. This choice almost forces readers to go through the book in order. Finally, the absence of pseudocode makes it difficult to generalise the solution to other languages. While Perl is used often by bioinformaticians, Perl source code is sometimes difficult to read, and text which explain sections of a program can easily become repetitive and uneventful.

Most of these shortcomings, though, are expected in a book such as this. Both biochemistry and computer science are vast fields which can only be fully understood by books dedicated to them. In fact, bioinformatics is a difficult field to write for since it juggles several seemingly disjoint areas including computer science, mathematics, and biochemistry. And, as the goal of this book is to explain bioinformatics using Perl as a framework, the reliance on Perl is unavoidable.

Because of this, the reader may need several other resources in order to follow this book. While *Genomic Perl* is suitable for an upper year level undergraduate course and above, a lecturer or book is necessary to fill in any missing gaps. These gaps include biochemistry, mathematics, Perl, and algorithms, depending on the individual. That is, reading this book in isolation may prove difficult.

In conclusion, I am pleased to say that this book is a rare example of when one *can* judge a book by its cover! The title "Genomic Perl: From Bioinformatics Basics to Working Code" summarises the structure chosen by the author. Each chapter commences with only the rudimentary basics for each topic, and concludes with working Perl code, which is also included in the accompanying CD ROM. Excluding students, others with an adequate knowledge of bioinformatics may benefit from this book since the accompanying Perl source code can be easily extended and deployed.