Review of the book

"Combinatorial Pattern Matching Algorithms in Computational Biology using Perl and R"

by Gabriel Valiente

CRC Press, Taylor & Francis Group, 2009

Jannik Pewny
Horst Görtz Institute, Ruhr University Bochum, Germany

1 What the book is about

The book holds what its cover promises: It is a well-sorted collection of pattern matching algorithms that are used to work with problems in computational biology. Pattern matching is term for finding a given pattern in some data or finding common patterns in two different sets of data.

The algorithms handle the following data-types, which model abstract representations of occurring problems:

- **Sequences**
  Nucleotides in DNA/RNA forming the genome of all organisms or amino-acids forming proteins, are basically defined by the order in which they appear. The resulting structure is called a sequence. They are simple and common, not just in computational biology.
  Finding for example a characteristic subsequence in a DNA-string can help you to find the start of single genes. Common patterns (or even: the degree of similarity) can give you hints about whether two creatures are related.
  The algorithms in this section cover things like simple search for a subsequence, usage of suffix arrays and different metrics (Hamming distance, Levenshtein distance, edit distance) of alikeness.

- **Trees**
  Descriptions of the way sequences like RNA/DNA/proteins fold up themselves to 2 or 3 dimensions or phylogenetic systems showing which organisms developed from other organisms, can be modeled using trees.
  E.g. if two different genealogies, constructed from different knowledge, share a similar part, it is likely that this part is correct. These algorithms can help to find and quantify such parts.
  The author explains things like Newick-strings defining trees, finding paths, partition distances, finding subtrees (simple and by quartets defining a tree) and similarity measurement of trees.

- **Graphs**
  Metabolic pathways are the way a group of enzymes interact with their products and each other. Usually you have a large group of enzymes reacting with some byproducts of other enzymes, leading to a byproduct that is used by maybe another enzyme. This gets pretty chaotic and unmanageable as the number of participants increases. Luckily it can be modeled with a graph, where each node is an enzyme or a byproduct and each node is the relation "A is used by B".
  If you can find a pattern in such graphs, it might help you to extract certain "sub-procedures", which you might already know, simplifying the problem and maybe allowing you to understand the whole structure.
  Finding paths, tree, common subgraphs, computing distances, lowest semi-strict ancestors, are only some of the algorithms of these chapters.

Therefore the coverage of the topic seems somewhat complete.

The languages this author has chosen are Perl (a popular scripting-language, which was originally designed for processing log-files) and R (a script-language with run-time interpreter, which can be used to type and execute commands while you type them; it is a free development/enhancement of S, and is meant to allow easy access to statistical and graphical methods to analyze and process scientific data). Additionally, the author used pseudo-code.

It is divided into three main-parts, dealing with above mentioned data-structures. Each of these parts holds chapters introducing the handled type of text/pattern, including their representation in the two languages. It is followed by chapters about simple pattern matching and general pattern matching.

2 What the book is like

Even when looking at the directory of the book, you can see that it is very well structured. The author shows a strict, scientific style of writing, without embellishment or over-shortening content.

Algorithms are usually presented in the bottom-up-way, showing a problem first, an idea to solve it, some pseudo code, then Perl-, then R-code. This representation is logical and explains the algorithms pretty good. Since the chapters are mainly independent, the book also allows reading only these chapters interesting to the reader rather then the whole book cover to cover. The author somewhat prefers R over Perl — not that the Perl-Code is bad or incomplete, but the R-code contains the input of the handled data and the output, more often than the Perl-code does. At least my personal sensation says so. In addition, the author does not reinvent the wheel: He uses libraries like CPAN, CRAN or the BioPerl-Project.

The author also made excessive use of copy-and-paste. I don't blame him, since it is not worth the effort that sentences introducing scripts vary a lot. But be prepared to be confronted with one of the following patterns when reading this book: "Combinatorial pattern matching is the search for exact or approximate occurrences of a given pattern in a given text. When it comes to computational biology, both the pattern and the text are (sequences/trees/graphs) and the pattern matching problem becomes one of..." at the start of the bigger chapters and "The representation of (trees/graphs/phylogenetic tree...) in (Perl/R/BioPerl) does (not) include any method to compute (algorithm or problem-solution). However (that algorithm) can easily be (implemented/designed) using the (function previously-written or from a library)-function to (solve the problem), as illustrated by the following (Perl/R) script" before almost each script.

On the other hand, the author gives a lot of additional references appended to each chapter on where to find more information about the processed structures, algorithms and the general topics.

One also has to mention two other important features of this book: You can find all of the sources on the author's website, which comes pretty handy, when you actually want to use them, since you do not have to retype them. And there is an introduction to Perl as well as to R, showing how to decode DNA/RNA-triplets to amino-acids and giving some basic overview over standard functions. This is — of course — not an exhausting description of the languages and does not always guarantee to understand each script of the book, since this overview does not include library-description of e.g. CPAN, CRAN or the BioPerl-Project.

In my opinion the biggest shortcoming of this book is the missing runtime-analysis of the algorithms presented. Not that I am a fan of Landau-notation, but apart from the adjective "fast", there is no analysis on the runtime whatsoever, not even benchmark-like absolute runtimes over given problems. So the algorithms are described, constructed, illustrated by examples and implemented in different languages—but there is stops. So there is no way to compare the speed of the algorithms shown, not even among each other. In addition, some algorithms presented haven an *exponential* running time and are hence of no practical relevance. This only becomes clear when looking at their speed—and not from a glance at the code alone.

3 Recommendation

Actually, I do recommend this book. Not as an introduction to (pattern matching in) computational biology and not as a compendium of algorithms (of pattern matching) in computational biology, which may be the author's aim. But I certainly recommend this as an introduction and reference to some algorithms of pattern matching in computational biology. You do actually learn, how algorithms over the most important data-types are designed, in a pretty straightforward, logic way. This may be not sufficient for practical uses, since the algorithms may not exactly match your needs and are not optimized for speed.

This book may be of more use for a student of bio-informatics (which is quite intended; it is likely, that they have a direct need for the given algorithms) than to students of biology (they will understand the possible applications of the algorithms, but will certainly lack knowledge or even interest about programming languages and the algorithms themselves) or informatics (they will understand the code, but will not have a use for the algorithms and will miss the analysis and comparison of them).

I cannot state though that you can use this book without having at least a fundamental understanding of programming. If you did not write or read code before, you will have trouble with the scripts, not when it is about understanding what it does or retyping them, but understanding what it does exactly and extending or customizing it, even after having read the whole book.

ISBN: 1-4200-6973-X, 978-1-4200-6973-0

The reviewer is a student of IT-Security at the Ruhr-University of Bochum.