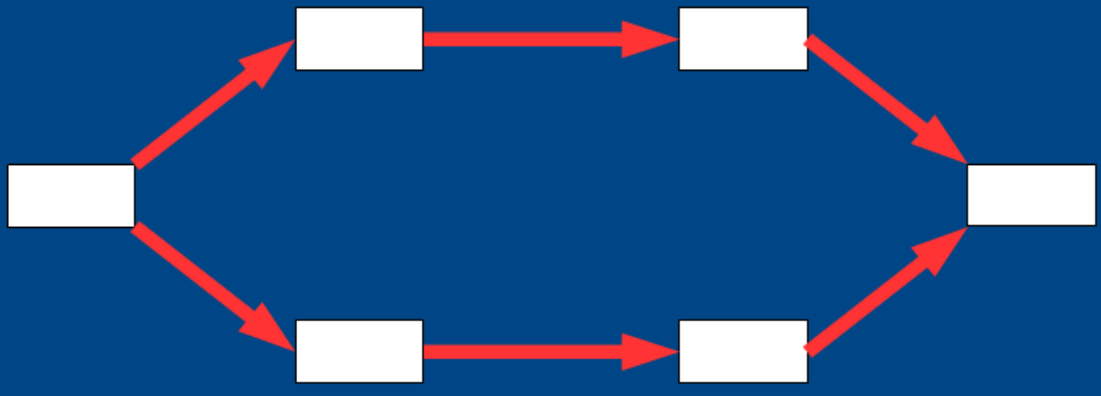
Bioinformatics - Introductory Series

Using de Bruijn Graphs for Short Read Assembly



Homolog.us

Using de Bruijn Graphs for Short-read Assembly

Homolog_us

This book is for sale at http://leanpub.com/NGS-assembly

This version was published on 2015-08-18

Also By Homolog_us

Pandora's Toolbox for Bioinformatics

Contents

1. Introduction

This book is written for thousands of biologists eager to explore living systems using recent technological breakthroughs in nucleotide sequencing, known as next-generation sequencing (NGS) technologies. NGS technologies revolutionized biology by giving access to high-throughput sequencing instruments to large and small research organizations around the globe. Such easy access to sequencing brought a dramatic shift in the way genetics research gets done. Merely a decade back, sequencing of large eukaryotic genomes used to be very long and expensive endeavor, for which hundreds of scientists had to raise funds together by writing joint white papers and petitioning to various government agencies. The tasks of sequencing and assembly were handled by dedicated sequencing centers, of which only a few existed around the globe. Naturally, the capacities at those sequencing centers were significantly constrained from high volume of requests. In contrast, today any scientist can get more nucleotides than the human genome sequenced within a week from a local facility for an insignificant price.

This rapid and somewhat unexpected democratization of sequencing capabilities left the biologists unprepared for resulting 'data deluge'. Every downstream analysis step, including sequence mapping, assembly and even storage of data, has now become a challenge. In previous arrangements, genome assemblies were performed by a few dedicated groups within the large sequencing centers, and those groups sheltered the biologists from various unpleasant aspects of handling large volumes of data. Many of those dedicated groups still remain the leaders in research on sequence assembly, but the scientists from other institutions are venturing into assembly and analysis to make sense of locally acquired short-read libraries. In fact, it is now nearly impossible to do high-quality genetics research in life sciences or medical sciences without proper knowledge of the bioinformatics tools. To make matters worse for the newcomers, the technological frontier in bioinformatics itself is moving fast with introduction of new tools almost every week. This book is written to make the lives of the life scientists easy by providing a simple introduction to the recent and ongoing developments of algorithms related to short read assembly.

Assembly procedure requires an understanding of sequencing technologies, algorithms and statistics

The genome is the primary informational unit of living organisms. It is packaged into multiple chromosomes. Those chromosomes get replicated and passed on from generation to generation to maintain continuity of the species. Natural selection works through random changes in the genome ('mutation') and fixation of advantageous mutations within the population. Therefore, genome sequencing and assembly is an important step toward understanding the phenotypes and their modifications during evolution.

The genome assembly process, described in abstract terms, requires an understanding of three aspects - sequencing technology, algorithms and statistics. Let us see what roles they play.

Sequencing technologies

NGS technologies cover wide range of sequencing instruments, each offering unique benefits and challenges during assembly. The assembly process often requires combining sequences from multiple technologies. Moreover, many researchers are restricted to use certain technologies due to ease of access or cost. For these reasons, this book covers the differences between sequencing technologies and their impact on assembly process in chapter 4.

Algorithms

An algorithm consists of a set of rules to perform a given task. For example, young children learn algorithms to add numbers using carry and subtract using borrow. Conceptually, algorithms are not too different from the experimental protocols used by biologists. Their mastery requires two steps - understanding the concept and memorization through practice. Mechanization, such as using calculators to add and subtract numbers, cannot replace the value added from understanding of the concepts. Therefore, this book focuses on explaining the algorithms involved in assembly process. This point is further explained in the following section.

Statistics

Given that nucleotide sequencing is an experimental process, experimental variations and noise are always part of the data. Those artifacts manifest in the assembly procedure in two ways - (i) use of cutoff parameters to limit the amount of noise and (ii) use of statistical

methods, such as averaging or hidden Markov model. Biologists are well familiar with those procedures from other contexts.

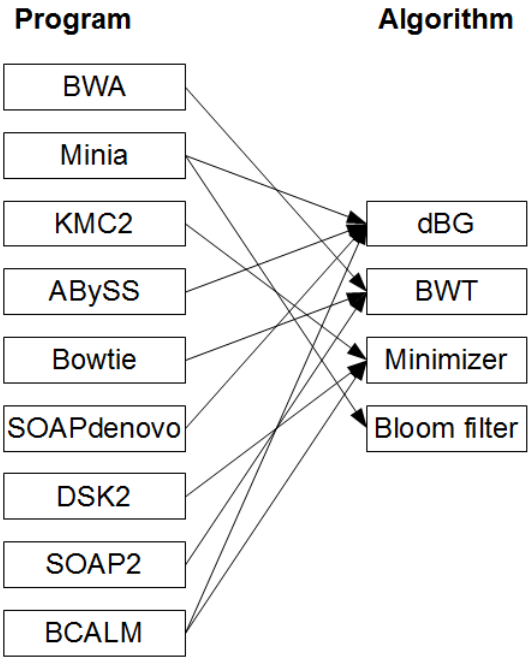
This book explains the statistical aspects by first assuming error-free data and presenting the core assembly algorithm in the absence of noise (Chapter 3). Then the impact of noise and uneven coverage are explained in a later chapter (Chapter 5). By partitioning the assembly problem into three components, a biologist will be able to figure out whether his assembly quality can be improved by moving to different technology, better algorithm or higher coverage. Moreover, this book adds adequate amount of simulations and analysis of real data to allow readers play with various aspects and learn.

Why should biologists learn about assembly algorithms?

This book's approach of focusing on algorithmic concepts departs from the current training practices in bioinformatics. In the current approach, the biologists analyzing NGS sequences typically learn to use and benchmark various software tools, such as Velvet, ABySS, Trinity, Oases, SOAPdenovo, Minia, Ray, IDBA, SPAdes, etc. to assemble the underlying genomes, transcriptomes or metagenomes. Even though all of those software programs use de Bruijn graphs to reconstruct genomes from NGS libraries, proper understanding of assembly algorithms is not needed to run them. Is there any benefit for biologists in learning the inner workings of those assembly programs?

The answer is an emphatic yes. Such an understanding helps in performing every step of the work including designing experiments, purchasing computing hardware, improving assembly quality, interpreting results and, most importantly, staying abreast of technological developments. Furthermore, NGS sequencing and assembly has become routine tools for analyzing transcriptomes (RNA-seq) of non-model organisms, and the same benefits of learning about assembly algorithms apply there. The biggest advantage of understanding the algorithms is likely to come from the ability it provides in developing clever and unconventional experiments using NGS technologies. It appears that the full potential of NGS technologies is not being realized, because a large gap exits between those who think in terms of experiments and those who develop algorithms to analyze related large data. This book intends to close that gap and give an experimentalist power to think through all aspects of data analysis. Each of the above points is discussed in detail below.

Software tools change, but the algorithms remain more stable



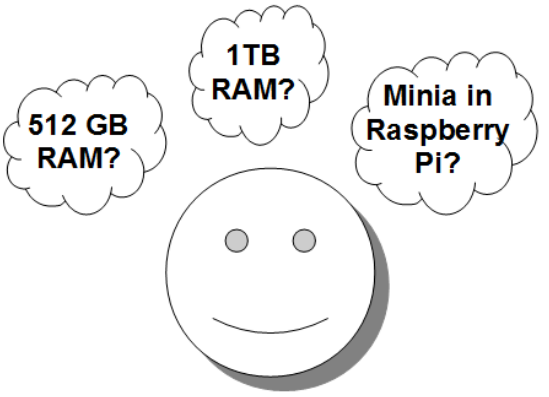
Many software tools are built by reusing a small set of algorithms

Over the last three years, computational researchers came out with many efficient programs to improve various assembly steps, but those programs remain under-utilized until they are fully incorporated into the existing pipelines. Switching from one program to another, or even deciding about proper benchmarking measures to test the effectiveness of new programs, is a major challenge for those trained with software programs. Proper knowledge of algorithms should make the task easier, because the underlying algorithms behind various programs stay more stable than the codes.

Knowledge of algorithms helps in experimental design

Designing of NGS experiments require making decisions regarding the mix of sequencing technologies, mix of mate pair libraries and read coverage. Such decisions are hard to make without a conceptual understanding of how the assembly gets done.

Understanding algorithms helps in purchasing adequate computing hardware



Hardware decisions are difficult to make

Bioinformaticians knowing the assembly algorithms can solve larger problems without buying expensive computer hardware. Exhausting available computer memory (RAM) is the first obstacle faced by new bioinformaticians working on de Bruijn graph-based assemblers. The most obvious and commonly sought solution of finding a computer with large RAM does not scale well with ever-increasing amount of data. A number of elegant algorithms were recently proposed by several bioinformaticians, but their implementation and incorporation into existing workflows require some knowledge of the underlying assembly process.

Improving assembly quality and interpreting results better

When a bioinformatician blindly runs a program to obtain results, it is unclear to him whether the gaps or errors in his assembly are due to shortcoming of technology, lack of coverage, execution of program or noise. Even with a familiar program, the quality of assembly can be greatly improved by searching over the entire parameter space, but some of those parameters can only be explained with reference to the de Bruijn graph structure.

RNA-seq in non-model organisms

De Bruijn graph-based algorithms have become important for a new class of problems, namely transcriptome assembly (RNA-seq). Transcriptomes did not need to get assembled in the older days of Sanger sequencing, because the sizes of ESTs were comparable to genes,

whereas the gene expression was measured by microarray technology. High-throughput NGS can solve both problems together, but shorter NGS reads need to be assembled into genes, especially for organisms lacking underlying genomes. De Bruijn graph-based Trinity and Oases transcriptome assemblers are popular among researchers working on RNA-seq data.

Using NGS technologies to solve novel problems

In 2010, a group from Fred Hutch Research center used NGS technologies to sequence the variable regions of human immune cells, namely T-cells and B-cells. Their method has been commercialized by Adaptive Biotechnologies, a Seattle-based company. In 2012, a Stanford group came up with a method to sequence long reads synthetically using short read sequencing. Their technology was commercialized by Moleculo, a company that was later acquired by Illumina. Those are two of many examples of creative uses of NGS technologies. In all cases, the analysis of large amount of data is a critical component of development of the technology, and often it requires designing new algorithms. It is expected that many other unconventional uses of NGS technology will be discovered in coming years, and an experimentalist with proper knowledge of algorithms will have an upper hand to think about creative applications.

Description of this book

Over the last four years, several introductory articles on de Bruijn graph-based assembly programs were published at homolog.us blog[1] and they remain popular among our readers. For their convenience, the articles were compiled into tutorials and this book is an expansion of those tutorials to explore the same topics in further detail. This book describes how de Bruijn graph-based assembly programs work, explain why so much RAM is needed by the assemblers, show the impact of sequencing error on graph structures, discuss the differences between genome assemblers, transcriptome assemblers and metagenome assemblers and cover many recent algorithmic developments.

Chapters

i) Chapter "The Genome Assembly Problem" explains the concept of shotgun assembly and introduces the readers to various assembly algorithms, such as greedy strategy, overlap-layout-consensus method and de Bruijn graph.

[1] http://homolog.us

ii) Chapter "De Bruijn Graph of a Genome" explains how the de Bruijn graph of an already assembled genome looks like and demonstrates how perfect reads can be assembled using de Bruijn graph approach.

iii) Chapter "Experimental Considerations" covers sequencing technologies.

iv) Chapter "Genome Assembly from Noisy Reads with Uneven Coverage" explains genome assembly procedures for real-life reads containing errors.

v) Chapter "Assembling Transcriptomes, Metagenomes and Highly Polymorphic Genomes" shows how to solve other assembly problems.

vi) Chapter "Faster, Better and Cheaper" discusses various algorithmic improvements.

vii) Chapter "In Depth Discussion of Three de Bruijn Graph-based Assemblers" discusses the algorithms and codes of three assemblers in detail.

Algebraic notations are avoided as much as possible

To make this book more accessible to biologists, preference is given to pictorial style of narration over algebraic notations commonly used in bioinformatics papers. Readers should note that our chosen style is in no way inferior to algebra and can in fact provide better understanding of graph-based algorithms used for assembly. The preference for algebraic notations by computational papers is primarily to make the description more amenable to computer programming.

Living electronic book

This book chooses a publishing style that allows regular updates of the book to keep content up to date with the rapidly advancing NGS bioinformatics. Unlike other technical books, whose contents are static once the book is published, this electronic book will continue to get updated with new information, and readers will be able to download and read the latest version at no extra cost. Leanpub, as an online publisher, allows readers to take advantage of this feature. In addition, the book provides other advantages of electronic publishing, such as using hyperlinks to go between chapters, being able to use search capabilities of electronic reader, etc.

Core concepts are reinforced through repetition

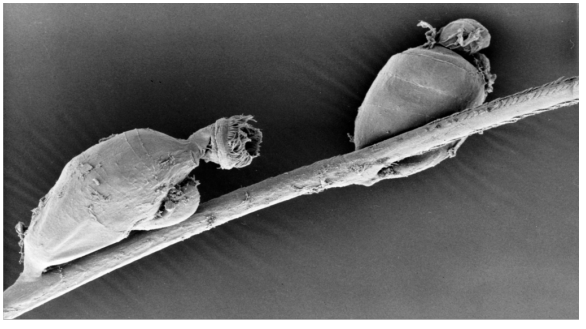
The primary aim of this book is to make sure the readers fully understand and remember the core concepts after reading this book. Therefore, entire chapters in the early part of the book

are dedicated to explaining ideas covered in at most one or two sentences in technical papers, whereas a later chapter "Faster, Better and Cheaper" condenses discoveries presented in many advanced papers. If the readers grasp the core concepts explained in earlier chapters, they will be able to go through the advanced papers themselves with some contextual help from our explanations. Explanations for some of the key concepts are repeated in multiple places to facilitate understanding, and the codes and exercises are expected to further reinforce the learning.

Regarding the exercises

Code - Pandora's Toolbox for Bioinformatics

Many useful bioinformatics programs are scattered all over the internet and it is not easy for a newcomer to locate them without some guidance. Therefore, Pandora's Toolbox for Bioinformatics[2] brings a number of useful codes together. The toolbox is named after Symbion pandora, an unusual animal discovered in 1995.



Symbion pandora (image courtesy Dr. Peter Funch)

Within Pandora's Toolbox, three programs - kmc, bcalm and tip-cutter - will be frequently used in this book. Graphviz, a fourth program, is useful for viewing the de Bruijn graphs.

Program	Use
kmc	k-mer counting
bcalm	de Bruijn graph compaction
tip-cutter	removing tips from the graph
graphviz	viewing de Bruijn graph
bwa	aligner

[2] https://github.com/homologus/Pandoras-Toolbox-for-bioinformatics

Program	Use
samtools	for processing alignments

The program kmc takes one or more read libraries as input and counts the number of k-mers of certain size (k) within them. The following step produces two binary files - kmers.kmc_pre and kmers.kmc_suf. K-mers present only once in the libraries are removed.

```
% mkdir tmp
% kmc -k25 [library] kmers tmp
```

The program kmc_dump_bcalm extracts k-mers present 4 or more times from the binary output of kmc and saves in text file in.dot in format suitable for bcalm.

```
% kmc_dump_bcalm -ci4 kmers in.dot
```

The program bcalm builds a de Bruijn graph from the k-mers in in.dot and compresses their linear portions to generate out.dot. Then tip-cutter removes all tips from out.dot and saves in out-clean.dot.

```
% bcalm in.dot out.dot
% tip-cutter out.dot > out-clean.dot
```

Data - E. coli and Electrophorus

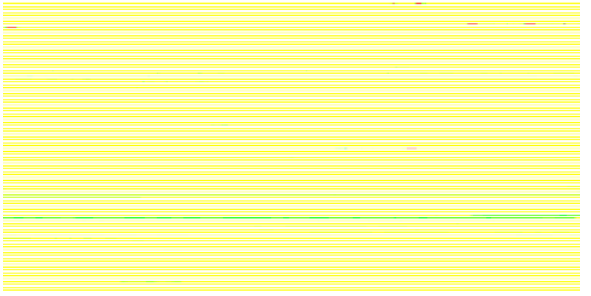
The following data sources will be used in the examples given in the book.

i) 'Hydrogen atom'

This small 1,000nt E. coli sequence along with associated reads has been provided by the authors of SPAdes as a toy set, and it is being used here for demonstration purposes. The genome and reads can be downloaded from here[3].

ii) E. coli genome and reads (SPAdes)

[3] https://github.com/homologus/Hydrogen-atom



E. coli genome

The fully assembled E. coli genome is only 5 megabases is size and can be used to learn about algorithms. A shor read dataset covering the E. coli genome at 1000x can can be downloaded from the SPAdes website[4].

iii) Electric eel (Electrophorus electricus) reads (SRA)

Electrophorus genome is about 700 megabases long. The short read sequences can be downloaded from the NCBI SRA website[5].

Other online resources for learning

Apart from the books within our introductory series, a number of online resources exist for those interested in learning bioinformatics on their own. They are classified below as 'introductory resources', 'intermediate resources' and 'advanced resources'.

Introductory resources

Rosalind and Coursera

Rosalind

[4] http://spades.bioinf.spbau.ru/spades_test_datasets/ecoli_mc/
[5] http://www.ncbi.nlm.nih.gov/sra/SRX554971

Rosalind[6], developed by a team led by Prof. Pavel Pevzner, allows students to learn bioinformatics online through problem solving. Professor Pevzner and his student Phillip Compeau also developed an online bioinformatics course through Coursera that is freely available from youtube[7].

Software Carpentry

Many newcomers to bioinformatics do not have any background in running computer programs, and especially those programs requiring experience with command line and unix operating system pose particular problems. Software Carpentry created a number of useful tutorials[8] to simplify the learning process.

R and Ipython notebook

R[9] is an useful computational tool, where statistical analysis and plotting of data can be easily done without knowledge of programming.

Ipython notebook[10] is a web-based interactive computational environment to help with executing codes, adding plots and writing text and equations.

Seqanswers

In online forum Seqanswers[11], users ask question about various bioinformatics programs or analysis approaches and get answers from the experts.

Biostar

Biostar[12] is an online forum like seqanswers, but it follows a different format. Each answer gets ranked by the members of the community so that the correct or most useful answer rises to the top. It creates an useful knowledge-base for future reference.

Intermediate Resources

The following blogs provide useful information on bioinformatics algorithms.

[6] http://rosalind.info/problems/locations
[7] https://www.youtube.com/user/bioinfalgorithms
[8] http://software-carpentry.org
[9] http://www.r-project.org
[10] http://ipython.org/notebook.html
[11] http://seqanswers.com
[12] http://biostars.org

Homolog.us

Homolog.us[13] blog presents useful information about the latest research in bioinformtics.

Heng Li

Dr. Heng Li of Broad Institute maintains two informative blogs - one at github[14] to cover bioinformatics algorithms and one at Attractive Chaos[15] to cover his lightweight klib library and other computer science-related topics.

Dazzler blog

The Dazzler blog[16] maintained by renowned researcher Gene Myers releases information about his long-read assembly tools.

Advanced resources

Arxiv and biorxiv

The preprint websites arxiv[17] and biorxiv[18] have become invaluable resources for those doing advanced research in bioinformatics. These days, most computational researchers submit their papers at one of those repositories long before they get formally accepted at journals.

Github

Many bioinformaticians doing cutting-edge research do not write blog posts to describe their work, but often maintain very active github sites to publicly post their codes. Others may look into the codes to learn about the problem-solving approaches used by them. The links to a number of useful github pages are provided below.

[13] http://homolog.us/blogs
[14] http://lh3.github.io/
[15] https://attractivechaos.wordpress.com/
[16] https://dazzlerblog.wordpress.com/
[17] http://arxiv.org
[18] http://biorxiv.org

Bioinformaticians and their github pages

GATB[19]
Rayan Chikhi[20]
Heng Li[21]
Jared Simpson[22]
Gene Myers[23]
Alex Bowe[24]
Jason Chin[25]
Dinghua Li[26]

Acknowledgements for this book

The author is thankful to Rayan Chikhi, Jason Chin, Kevin Karplus, Anton Korobeynikov, Heng Li, Ruibang Luo and Jared Simpson for helpful discussions over the years or comments on this manuscript. The author also thanks the readers of the homolog.us blog and Twitter followers for bringing informative links on latest research to his attention.

Further readings

[19] https://github.com/GATB/
[20] https://github.com/rchikhi
[21] https://github.com/lh3
[22] https://github.com/jts/
[23] https://github.com/thegenemyers/
[24] https://github.com/alexbowe/
[25] https://github.com/cschin/
[26] https://github.com/voutcn