Julia Genomics Package

Final Project 18.337

Isha Jain

Table of Contents

Motivation

As economist Thomas Friedman once stated, "I firmly believe that the next great breakthrough in biosciences could come from a 15-year-old who downloads the human genome in Egypt". The sequencing of the human genome in 2003 has transformed the world of the biosciences and blurred the lines between experimentalists and computational biologists. As we experience this transition, it is essential to adapt our training and resources.

From everyday molecular biology experiments to more large-scale, high-throughput datasets, computation has become an everyday aspect of biology. However, the current state of computational tools is limited. A more user-friendly and centralized computational toolset would empower experimentalists and computational biologists alike.

Introduction

The human genome is comprised of three billion "base pairs" or building blocks. Each position in the genome has one of four types of building blocks. Together, base pairs make up individual genes. Many genes and regulatory regions together create a chromosome. Every cell in our body has 23 pairs of chromosomes, one from our mother and one from our father.

The primary functional components of cells are proteins or enzymes. Different cell types express different subsets of genes. Expressed genes are *transcribed* from DNA to RNA. RNA fragments are strings made of building blocks (with four types of building blocks). The RNA is then *translated* into protein, another string made of 20 types of building blocks.

As is apparent from the above description, molecular biology is essentially a collection of carefully regulated string operations. The human body computes a countless number of such operations every second. Biologists must also perform such computations to enable their research. Below I describe my attempt at creating a user-friendly genomics package in the Julia language.

Problem Definition

The functions required in a genomics setting can be placed in three categories: daily use, intermediate applications and large-scale analysis. Daily use functions are primarily required for cloning (DNA manipulations) or to convert reference sequences from DNA to RNA to protein (and in the reverse direction). Intermediate applications are also used for the aforementioned purposes, however require more extensive computations based on biophysical properties. Traditionally, experiments are designed to study a specific pathway, specific disease or specific condition. Over the last decade, it has become possible to more holistically survey *all* genes, *all* pathways, etc. Recently developed techniques create large datasets which must be processed and mined computationally. Such applications require more advanced computing. Julia serves as an ideal language in which to create a genomics package spanning these different levels of computational expertise. It is more user friendly than other options, yet can be used to

parallelize some of the more advanced functions. Currently available software is discussed below, for comparison.

Related Software

A. Every-day Tasks

Day-to-day manipulations of DNA and RNA are essentially a collection of string operations performed by various enzymes (proteins). In order to harness the power of such naturally-occurring operations, we must be able to predict the outcome of various enzymatic reactions.

The operations needed to convert between DNA, RNA and protein are relatively simply mappings between dictionaries. However, the operations can become somewhat more complex when applied to longer and longer sequences (or collections of sequences) on the order of the genome (3e9 base pairs).

Currently, molecular biologists turn to a collection of disjointed websites to perform such simple operations. Often, such websites are maintained by companies which sell the enzymes needed to perform such reactions in the laboratory setting. As such, there is little motivation to have an efficiently written or maintained back-end.

B. Intermediate Tasks

Certain stages of experimental design require a more nuanced understanding of the biophysical properties of DNA, RNA or proteins. The theory behind such biophysical properties has been elucidated over the last few decades.

Matlab provides several functions that can used to predict such biophysical properties. However, the algorithms they apply are not optimal when applied to experimental settings. Additionally, most biologists are not willing to fund or install Matlab to perform such tasks. Websites hosted by individual laboratories offer an alternative option. However, such websites are not maintained properly and are not centrally accessible.

C. Advanced Tasks

The advent of high-throughput technology has generated terabytes of data over the last decade or so. While generating such datasets is relatively straightforward, the analysis and interpretation of such datasets requires a substantial amount of computational resources and computational savvy. Experimental biology labs that rely heavily on such datasets will often hire a computational scientist for the sole purpose of parsing such data. Alternatively,

the analysis will be performed in an ad-hoc manner, leaving little hope of reproducibility.

The R Bioconductor package is the current state-of-the-art resource for analyzing high-throughput datasets. It has some very well-written packages for interpreting typical datasets such as genome sequencing, exome sequencing, ChIP-Seq, Microarray data, etc. However, since these packages are largely open source, the number of options and lack of documentation can be overwhelming for a new user.

Molecular biologists could greatly benefit from a more centralized computational toolset which spans the realm of daily-use through advanced computations.

Library Description

A. Daily-Use Library

1. Reverse Complement

DNA polymers are sequences of nucleotides. The order of the sequence determines the structure and functionality of the resulting protein. DNA is double-stranded, with each strand having directionality. By the nature of base-pairing, information from one strand can be used to determine the sequence and directionality of the opposite strand (referred to as the reverse complement).

I have implemented a Julia function which determines the reverse complement of any given sequence of arbitrary length. Generally, such operations are only performed on relatively short strings (few thousand base pairs at most). However, a parallel version of this function could be used for larger scale applications, such as genome-scale reverse complementation.

2. Translator

RNA is converted to protein in three base-pairs units. In order to predict a protein sequence that will result from a RNA sequence, you must first determine the “start site” specified by a particular triplicate. From there, three base pairs units are converted into protein sequences until reaching a “stop site”. Proteins are sequences of amino acids which are mappings from triplicate sequences of RNA. I have implemented a Julia function which translates an RNA or DNA sequence into the corresponding amino acid or protein sequence.

B. Intermediate Toolbox

1. Primer Melting Temperature Calculator

A typical experiment in molecular biology requires the amplification of a stretch of DNA from the genome or a plasmid. Amplification requires the binding of short pieces of DNA flanking the region of interest. For amplification to occur, we must design fragments or "primers" with a specific melting temperature and then adjust the experimental protocol to match such conditions. The melting temperature of DNA strands can be predicted based on the subsequences and sequence context.

I have implemented a well-established algorithm which uses the relative representation of G/C nucleotides (building blocks) and A/T nucleotides to predict the melting temperature of a primer or longer DNA sequence.

2. Restriction Enzyme Cutter

Restriction enzymes are natural-occurring proteins which bind to and cut specific sequences of DNA. Such enzymes recognize 8-12 base pair sequences and can cut DNA strands symmetrically or asymmetrically. Hundreds of such enzymes exist, corresponding to different recognition sequences. We can harness the power of such enzymes to perform "cut and paste"-like operations on DNA. This empowers us to clone different regions of the genome and study their function.

I use a list of such enzymes and their corresponding recognition sequences to predict all cutting sites for a piece of DNA. The user can input a gene or locus they are trying to clone and the function outputs the enzymes which cut the sequence.

Currently, New England Biosciences provides an analogous tool. However, due to high user-load, it is often very slow or does not load properly. This Julia function provides a faster alternative.

3. Protein Atomic Composition

While amino acid composition is useful to understand a protein's functionality, a higher resolution understanding is needed to discern biochemical and biophysical properties of a structure. For example, the atomic structure of a protein will determine the net charge at a given pH. The number of disulfide bridges (a type of secondary structure) in a protein will be determined by the number of sulfur atoms present. The molecular weight, isoelectric point, etc. are all based on atomic composition. Therefore, I have created a function that takes a protein sequence as an input and outputs the predicted atomic structure. An

added functionality would be to include the pH and environmental dependence of such features. This option will be included in future versions of the library.

C. Advanced Toolbox

1. Motif Finder

DNA and RNA-binding proteins have a great specificity for binding to subsequences. Generally, such binding exists at regulatory regions (ex. upstream of a gene). In this scenario, a gene may turn on or off if such a sequence exists upstream of the gene's start site. While such a paradigm has been well-established, the specific sequences that dictate such binding are unknown. In order to discern such motifs, experimentalists can define the regions to which a protein is bound and then look for motif enrichment to determine the sequences the protein recognizes.

I have implemented a motif finder in Julia. The basic principle relies on finding enrichment of subsequences in one collection of sequences compared to another collection of "background" sequences. This approach was first proposed in 1998 and works especially well for smaller genomes such as the yeast genome.

I first create a list of sequences present in intergenic (or less functional areas). This can then be used to create a matrix in which each subsequence is stored as a base 4 representation. The relative frequencies of all subsequences in this "intergenic list" are used to create a background distribution of subsequences. It is plausible that certain subsequences exist at a higher or lower frequency across the genome. We can use this background distribution to normalize for such differences.

The algorithm then creates a similar matrix for a sequence list that is thought to have an enrichment of a particular binding motif (based on experimental design). The test and background distribution are compared using a binomial distribution and the user receives an output of enriched motifs. Currently, this function is designed for the yeast genome. However, extending this to other organisms is simply a matter of creating files with different intergenic genome sequences corresponding to the given organism of interest.

2. Sequence Alignment

DNA sequences can duplicate over time and over the diversification of different species through evolution. It is often of interest to determine which genes are related and/or emerge from the same evolutionary

ancestor. Such information can be used to predict how close different species are in evolutionary time. It can also allow us to study genes or proteins of interest in a model organism and then map the results back to higher organisms.

Sequence alignment has been a problem in molecular and evolutionary biology for many decades. As such, various algorithms have been developed to optimize both local and global alignment of pairs and groups of sequences.

In the Julia genomics package, I implemented a variation of the Needleman-Wunsch Algorithm. This dynamic programming algorithm involves a sequential alignment of larger and larger subsequences, until the entirety of both sequences has been aligned (Fig 1).

First, a penalty is assigned for mismatches and gaps in sequences. This can be further developed to have context-specific penalties. A matrix is constructed with each of the two sequences on different axes. The first column and row is filled with gap penalties. Every remaining cell is filled by considering the alignment of the previous step and then choosing the option (gap in sequence A, gap in sequence B, match or mismatch), which gives the lowest overall penalty. This matrix is iteratively filled. The most optimal path is reconstructed from the bottom right corner of the matrix by tracing the path that leads to the ultimate penalty score. This algorithm is currently implemented in serial, however a parallel version would be useful for multiple sequence alignment. Additional algorithms should be implemented for different types of alignments (ex. Extremely divergent or similar).

		C	O	E	L	A	C	A	N	T	H
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
P	-1	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
E	-2	-2	-2	-1	-0	-3	-4	-5	-6	-7	-8
L	-3	-3	-3	-2	-2	-1	-2	-3	-4	-5	-6
I	-4	-4	-4	-3	-1	-1	-2	-1	-4	-5	-6
C	-5	-3	-4	-4	-2	-2	-0	-1	-2	-3	-4
A	-6	-4	-4	-5	-3	-1	-1	-1	-0	-1	-2
N	-7	-5	-5	-5	-4	-2	-2	-0	-2	-1	-0

Fig 1. Schematic of Needleman-Wunsch Algorithm. Taken from etutorials.org

3. Co-expression Analysis

Microarrays can be used to profile gene expression in a given tissue or sample type. Microarray datasets are now available for a plethora of different experimental conditions. Such data can be used to determine novel genes in a given pathway, organelle, etc. Genes that are functionally related tend to be co-expressed across different experimental conditions.

The Julia genomics package contains a co-expression analysis function. For each gene, a ranked list is created of genes which are co-expressed through different conditions. A gene list is created corresponding to the biological function of interest. The intersection of top-correlating genes and the comparison gene list is determined for each gene. The genes with the largest number of neighbors, or genes in the intersection are likely to be related to the original function being probed. This can be used as the basis for experimental confirmation of a novel gene's function. Such approaches have previously been used to determine organelle proteomes, functional pathways, etc.

Future Work

The current genomics library provides examples of computational functions that are needed for a wide-range of applications of different experimental and computational scales. The advanced toolbox can be expanded to include a larger variety of data analyses. For example, it would be useful to have functions to analyze mass spectrometry data, construct evolutionary trees, create gene expression networks/graphs, etc. Furthermore, the functions in the daily-use and intermediate toolboxes can be optimized by trying several different algorithms and testing them on larger sequences.

In order to make this even more user-friendly, a user interface could be written to allow ease of access.

Conclusion

The biological sciences are generating an ever-expanding need for computational tools and scientists. As this transformation is in its infancy, solutions to such problems are rather ad-hoc and disorganized. Currently, biologists use an amalgam of websites and software to perform computations related to their work. This leads to a lack of standardization and efficiency. This Julia Genomics Package is an attempt to provide a universal, easy-to-use toolset for computational (and increasingly for experimental) biology. Further expansion and

refinement of this library has the potential to attract many users from the biological and genomic sciences.

Acknowledgements

Thank you to Jeff and Prof. Edelman for teaching such a useful and informative class. It was a pleasure being a part of it.