Genome analysis

NGS+++: a library for rapid prototyping of epigenomics software tools

Alexei Nordell Markovits¹,†, Charles Joly Beauparlant²,†, Dominique Toupin³, Shengrui Wang³, Arnaud Droit²,* and Nicolas Gevry¹,*

¹Department of Biology, Université de Sherbrooke, Sherbrooke, Quebec J1K 2R1, ²Department of Molecular Medicine, Centre de Recherche du CHU de Quebec, Université Laval, Quebec, Quebec G1V 4G2 and ³Department of Computer Science, Université de Sherbrooke, Quebec J1K 2R1, Canada

ABSTRACT

Motivation: The development of biological databases enables researchers to store, analyze and manipulate genomic information. However, the huge amount of data generated during genomic experiments requires the development of tools that are both flexible and efficient. Previous years have witnessed an explosion in the amount of genomic data generated using next-generation sequencing (NGS) technologies, as exemplified by the ENCODE project (ENCODE Project Consortium et al., 2012). However, analysis of these enormous datasets (easily >100 GB) requires the use of a new generation of computational tools. As the quantity of data produced by NGS machines increases, so will the time spent on developing new tools. Although substantial efforts have been made at integrating them into user-friendly frameworks such as Galaxy (Giardine et al., 2005) or GeneSpace (Genome Space), relatively little effort has gone into providing the groundwork needed to increase the productivity of NGS developers, such as libraries and using standardized formats. Improvement in these areas would allow developers to greatlly accelerate the speed at which they design and deploy new analysis software.

Although certain tool suites, such as BEDtools (Quinlan and Hall, 2010) and BAMTools (Barnett et al., 2011), offer a library or API to assist developers, these are generally aimed at providing access to the existing tool functionality rather than facilitating development of new ones. As such they are highly specialized. The SeqAn library (Döring et al., 2008) offers functionality for the development of future tools, but it specializes in sequence analysis rather than genomic regions and features. Our proposed library, NGS+++, aims to fill this gap by offering a powerful set and flexible options to accelerate development and prototyping of epigenomics analysis tools.

2 APPROACH

It is impossible to predict the entirety of future needs for NGS data analysis. As such, NGS++ focuses on being a customizable and generic library that facilitates the prototyping and implementation of new functionalities via a transparent data interface. In this section, we summarize the three main components of NGS+++: (i) file format management, (ii) data manipulation and (iii) functional operators.

Dealing with the wealth of existing file formats is a time consuming task. NGS++ offers a simple interface to parse and write in many frequently used genomics file formats (BED, GFF/GTF, Sam, Wig, bedGraph) using a generic data structure named Tokens that contain a number of standard features of genomic data entries (eg: Start/End positions, position value and mapping quality). Additionally, the user can define ‘on-the-fly’ custom formats to deal with the plethora of datasets that do not respect format specifications, and BAM format is supported via integration of the BamTools API. The conversion between most supported formats is a trivial task:

uParser parser("filename.sam", "SAM");
uWriter writer("filename.bed", "BED");

while (!parser.eof())
{
    writer.writeToken(parser.getNextEntry());
}

*To whom correspondence should be addressed.
†The authors wish it to be known that, in their opinion, the first two authors should be regarded as joint First Authors.
Additional functional operators exist, allowing a variety of different operations on the dataset. This interface wraps many of the STL algorithms, enabling rapid parallelism via the OpenMP standard (Dagum and Menon, 1998).

3 IMPLEMENTATION

NGS++ is written in C++ using the C++11 standard. It offers a complete exception handling interface using the Boost exception class. A complete test suite is implemented using the Google test platform. It has been designed for a Linux environment using a C++11 compatible gcc compiler. Complete user guide, tutorial and discussion are available on the web page. Source code is hosted on GitHub.

4 CONCLUSION

Progress in the development of advanced bioinformatics analysis tools has undoubtedly been hindered by the lack of available programming frameworks. Our library aims to assist in filling this gap for the community of C++ epigenomic developers by giving them access to robust building blocks, thus reducing the time spent on development significantly. Our efforts are now focused on including additional genomic formats and on increasing the breadth of our tutorials. Future developments will include the integration of mid-level reusable functions, such as similarity functions and normalization methods. The website provides a list of tutorials and commented working code examples, to assist developers in getting started with the library. Finally, the GitHub should facilitate the integration of suggestions and feedback from the community.

Funding: Natural Sciences and Engineering Research Council of Canada (NSERC) (to S.W.); Canadian Institutes of Health Research (CIHR) (to N.G.); Ministère du Développement Économique, Innovation et Exportation (MDEIE) (to A.D.). N.G. holds a Chercheur boursier (junior 1) award from the Fond de Recherche en Santé du Québec (FRSQ). A.D. holds a Réseau de médecine génétique appliquée (RMGA) salary award. C.J.B. holds a Centre de Recherche en Endocrinologie et Génomique Humaine (CREMOGH) award.

Conflict of Interest: none declared.

REFERENCES


