BIOINFORMATICS APPLICATIONS NOTE Vol. 29 no. 15 2013, pages 1893–1894
doi:10.1093/bioinformatics/btt312

Genome analysis

Advance Access publication June 4, 2013

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

Alexei Nordell Markovits[1,†], Charles Joly Beauparlant[2,†], Dominique Toupin[3], Shengrui Wang[3], Arnaud Droit[2,*] and Nicolas Gevry[1,*]

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

Associate Editor: Michael Brudno

ABSTRACT

Motivation: The development of computational tools to enable testing and analysis of high-throughput–sequencing data is essential to modern genomics research. However, although multiple frameworks have been developed to facilitate access to these tools, comparatively little effort has been made at implementing low-level programming libraries to increase the speed and ease of their development.

Results: We propose NGS++, a programming library in C++11 specialized in manipulating both next-generation sequencing (NGS) datasets and genomic information files. This library allows easy integration of new formats and rapid prototyping of new functionalities with a focus on the analysis of genomic regions and features. It offers a powerful, yet versatile and easily extensible interface to read, write and manipulate multiple genomic file formats. By standardizing the internal data structures and presenting a common interface to the data parser, NGS++ offers an effective framework for epigenomics tool development.

Availability: NGS++ was written in C++ using the C++11 standard. It requires minimal efforts to build and is well-documented via a complete docXygen guide, online documentation and tutorials. Source code, tests, code examples and documentation are available via the website at http://www.ngsplusplus.ca and the github repository at https://github.com/NGS-lib/NGSplusplus.

Contact: nicolas.gevry@usherbrooke.ca or arnaud.droit@crchuq.ulaval.ca

Received on November 12, 2012; revised on May 23, 2013; accepted on May 27, 2013

1 INTRODUCTION

Previous years have witnessed an explosion in the amount of data produced 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 greatly 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 giving 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 developmental 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 of generic 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.

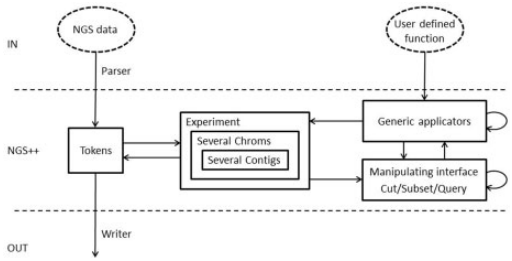

Fig. 1. Typical workflow of the NGS++ library. Data are read by our Parsing interface then filtered as needed. User-defined functions are executed via our operators, and the transformed data are stored via our Writer interface

The internal structure of the library is divided into a three-tiers hierarchy separating a genomic dataset by unique scaffolds with each of them containing any number of contigs. This hierarchy is represented in (Fig. 1), and each level offers a number of functions that can be extended via inheritance for specialized data manipulation. Loading datasets is done through integration with our Parser class and allows the user to easily load data:

```
TagExp.loadWithParser(inputStream,"SAM");
RegionExp.loadWithParser(inputStream,"BED");
```

NGS++ offers a powerful set of generic functions to compare, sort, merge and modify the previously loaded data. These include typical operations such as overlapping, merging and comparing that allows the user to easily filter his data as needed. The following would return tags overlapping a BED file:

```
auto fExp=TagExp.getOverlapping(RegionExp);
```

Additionally, the majority of these operators can be used on any given feature of the data objects, including features added by the user via inheritance. This example sorts and counts a subset based on a new object feature:

```
RegionExp.sortSites([](uRegion item1, uRegion
      item2){return item1.Score < item2.Score}, &
      uRegion::getScore, &uRegion::getScore)
int count = RegionExp.getSubsetCount(0.5,0.8);
```

As shown earlier in the text, this is greatly facilitated by the inclusion of anonymous lambda functions in the C++11 standard. Using these flexible operators allows the experienced developer to implement powerful modifications, whereas the default genomic interval operators are easily usable by all.

Finally, to accommodate specific analysis needs, NGS++ offers an interface to run developer defined functions and transformations on the selected data. This allows the developer to concentrate on the function he wishes to implement without having to spend time on the underlying structure that will support it. Borrowing heavily from the functional programming paradigm, this allows for rapid prototyping and implementation. In the following we define and execute a function that allows us to trivially generate a histogram of contig sizes:

```
map<int,int> sizeOfContigs;
uTagsExp.applyOnSites([&](uTag Elem){sizeOfContigs
      [Elem.getSize()]++;});
```

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 breath 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

Barnett,D.W. *et al.* (2011) BamTools: a C++ API and toolkit for analyzing and managing BAM files. *Bioinformatics*, **27**, 1691–1692.

Dagum,L. and Menon,R. (1998) OpenMP: an industry standard API for shared-memory programming. *IEEE Comput. Sci. Eng.*, **5**, 46–55.

Döring,A. *et al.* (2008) SeqAn an efficient, generic C++ library for sequence analysis. *BMC Bioinformatics*, **9**, 11.

ENCODE Project Consortium *et al.* (2012) An integrated encyclopedia of DNA elements in the human genome. *Nature*, **489**, 57–74.

Giardine,B. *et al.* (2005) Galaxy: a platform for interactive large-scale genome analysis. *Genome Res.*, **15**, 1451–1455.

Genome Space. http://www.genomespace.org (18 June 2013, date last accessed).

Quinlan,A.R. and Hall,I.M. (2010) BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics*, **26**, 841–842.