plyranges: a grammar for manipulating genomics data

Stuart Lee [1] , Michael Lawrence [2] , Di Cook [1]

1 Department of Econometrics and Business Statistics, Clayton, Victoria, Australia
2 Bioinformatics and Computational Biology, Genentech, Inc., South San Francisco, California, United States of America

Abstract

The Bioconductor project has created many powerful abstractions for reasoning about genomics data, such as the *Ranges* data structures for representing genomic intervals. By recognising that these data structures follow 'tidy' data principles we have created a grammar of genomic data manipulation that defines verbs for performing actions on and between genomic interval data. This grammar simplifies performing common genomic data analysis tasks via method chaining, type consistency and results in creating human readable pipelines. We have implemented this grammar as an Bioconductor/R package called plyranges.

Introduction

Genomic data may be naturally represented as sets of pairs of integers corresponding to the start and end points of sequences. Further information such as strandedness and chromosome name may be added to these sets to provide biological context. Because of the flexibility of this representation supplemental information such as measurements obtained from an experimental assay or annotations from a genome database can be joined to their relevant sequences. In the Bioconductor/R packages `IRanges` and `GenomicRanges` these representations have been implemented as a suite of data structures called *Ranges* [1]. These data structures cover many common data types encountered in bioinformatics analyses - a gene can be represented with its coordinates, along with its identifier and the identifiers of its exons; or an RNA-seq experiment may be represented as sets of genes with a matching count column.

The Bioconductor infrastructure for computing with genomic ranges are highly effecient and powerful, however the application programming interface (API) for performing analysis tasks with *Ranges* is complex due to its large number of methods and classes. It also makes resulting scripts written difficult for a non-programmer to read and reason about. An alternative approach would be to implement a domain specific language (DSL) as a fluent interface built on top *Ranges*. The goal of fluent interface is to enable users to write human-readable code via method chaining and consistent function returns. Fluent interfaces fit naturally in the context of Bioinformatics workflows because they enable writing succinct pipelines.

Several other DSLs have been implemented to reason about genomics data. Broadly, these are either implemented as query languages or as command line tools embedded in the unix environment.[1] An example of the former is the Genome Query Language (GQL) and its distributed implementation GenAp which use an SQL-like syntax for fast

[1] other ideas to mention GROK [2]

retrieval of information from genomic databases and BAM files [3]; [4]. Another example is the Genometric Query Language (GMQL) which implements a relational algebra for combining big genomic datasets [5]. The command line application BEDtools develops an extensive algebra for performing arithmetic between two or more sets of genomic regions [6]. It also has a python interface which simplifies construcing scripts for performing analyses based on BEDTools [7].[2]

The abstraction provided by the *Ranges* data structures aligns with the concept of tidy data [8]. The tidy data pattern is useful because it allows us to see how the data relates to the design of an experiment and the variables measured. Consequently, it makes both the modelling and manipulation of data more systematic. The *Ranges* data structure follows this abstraction: it is a rectanglular table corresponding to a single biological context. Each row contains a single observation and each column a variable about that observation.

The tidy data abstraction has motivated the development of `plyranges` a grammar of genomic data manipulation based on the *Ranges* data structures. It implements and extends the grammar defined by the R package `dplyr` [9]. The grammar provides a consistent way of interacting with and analysing genomic data via methods for constructing, grouping, mutating, filtering, and summarising *Ranges* and an algebra for reasoning about actions on *Ranges* and relationships between *Ranges*.

Design and Implementation

Discussion of the API and its key features

- every action on a Range is a verb (functional composition, method chaining)
- fluent (human readable, code describes what to do rather than 'how')
- every function returns a class familiar to the user
- an expressive algebra for performing arithmetic (anchoring functions)
- split-apply-combine strategies (group_by + reduce/summarise/mutate/filter)
- recasting overlapping/nearest etc as joins

References

1. Lawrence M, Huber W, Pagès H, Aboyoun P, Carlson M, Gentleman R, et al. Software for computing and annotating genomic ranges. PLoS Comput Biol. 2013;9. doi:10.1371/journal.pcbi.1003118

2. Ovaska K, Lyly L, Sahu B, Jänne OA, Hautaniemi S. Genomic region operation kit for flexible processing of deep sequencing data. IEEE/ACM Trans Comput Biol Bioinform. 2013;10: 200–206. doi:10.1109/TCBB.2012.170

3. Kozanitis C, Heiberg A, Varghese G, Bafna V. Using genome query language to uncover genetic variation. Bioinformatics. 2014;30: 1–8. doi:10.1093/bioinformatics/btt250

4. Kozanitis C, Patterson DA. GenAp: A distributed SQL interface for genomic data. BMC Bioinformatics. 2016;17: 63. doi:10.1186/s12859-016-0904-1

5. Kaitoua A, Pinoli P, Bertoni M, Ceri S. Framework for supporting genomic operations. IEEE Trans Comput. 2017;66: 443–457. doi:10.1109/TC.2016.2603980

6. Quinlan AR, Hall IM. BEDTools: A flexible suite of utilities for comparing genomic features. Bioinformatics. 2010;26: 841–842. doi:10.1093/bioinformatics/btq033

7. Dale RK, Pedersen BS, Quinlan AR. Pybedtools: A flexible python library for manipulating genomic datasets and annotations. Bioinformatics. 2011;27: 3423–3424.

[2] probably should also mention something about their cons, and more detaiil

doi:10.1093/bioinformatics/btr539

8. Wickham H. Tidy data. Journal of Statistical Software, Articles. 2014;59: 1–23. doi:10.18637/jss.v059.i10

9. Wickham H, Francois R, Henry L, Müller K. Dplyr: A grammar of data manipulation [Internet]. 2017. Available: https://CRAN.R-project.org/package=dplyr