Bioinformatics, 35(18), 2019, 3487–3488
doi: 10.1093/bioinformatics/btz116
Advance Access Publication Date: 15 February 2019
Applications Note

OXFORD

Genome analysis

Explore, edit and leverage genomic annotations using Python GTF toolkit

F. Lopez[1], G. Charbonnier[1], Y. Kermezli[1,2], M. Belhocine[3], Q. Ferré[1], N. Zweig[4], M. Aribi[2], A. Gonzalez[1], S. Spicuglia[1,5] and D. Puthier[1,*]

[1]Aix Marseille Univ, INSERM, TAGC, UMR U1090, Marseille, France, [2]The Laboratory of Applied Molecular Biology and Immunology, Tlemcen University, Algeria, [3]Molecular Biology and Genetics Laboratory, Dubai, United Arab Emirates, [4]Aix Marseille Univ and [5]Equipe Labellisée LIGUE contre le Cancer

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on November 12, 2018; revised on January 15, 2019; editorial decision on February 8, 2019; accepted on February 13, 2019

Abstract

Motivation: While Python has become very popular in bioinformatics, a limited number of libraries exist for fast manipulation of gene coordinates in Ensembl GTF format.

Results: We have developed the GTF toolkit Python package (pygtftk), which aims at providing easy and powerful manipulation of gene coordinates in GTF format. For optimal performances, the core engine of pygtftk is a C dynamic library (libgtftk) while the Python API provides usability and readability for developing scripts. Based on this Python package, we have developed the gtftk command line interface that contains 57 sub-commands (v0.9.10) to ease handling of GTF files. These commands may be used to (i) perform basic tasks (e.g. selections, insertions, updates or deletions of features/keys), (ii) select genes/transcripts based on various criteria (e.g. size, exon number, transcription start site location, intron length, GO terms) or (iii) carry out more advanced operations such as coverage analyses of genomic features using bigWig files to create faceted read-coverage diagrams. In conclusion, the pygtftk package greatly simplifies the annotation of GTF files with external information while providing advance tools to perform gene analyses.

Availability and implementation: pygtftk and gtftk have been tested on Linux and MacOSX and are available from https://github.com/dputhier/pygtftk under the MIT license. The libgtftk dynamic library written in C is available from https://github.com/dputhier/libgtftk.

Contact: denis.puthier@univ-amu.fr

1 Introduction

Several formats exist to store genomic features. The standard BED format stores basic information (chromosome, start, end, name, score and strand) related to generic genomic features (BED6) or composite genomic features (BED12). The GTF/GFF2 format (thereafter referred as GTF) can describe more exhaustively defined genomic features (genes, transcripts, exons, etc.) by taking advantage of the 'attributes' column which contains a set of keys/values to store various kinds of annotations. Some composition relationships are implicitly declared in the GTF file making it possible to describe, for instance, the exons of the transcripts corresponding to a gene. This relationship is more explicit in the GFF3 format that can be viewed as a directed acyclic graph with nodes corresponding to features (gene, transcript, exon, etc.) and edges corresponding to part-of relationships. Only few libraries are specifically dedicated to GTFs and most of them propose very focused tasks. The GenomeTools suite is a collection of bioinformatic tools based on the libgenometools C library that handle GTF and GFF3 formats (Gremme *et al.*, 2013). However, this library extends well beyond these annotation formats and the developing framework may appear rather complicated for naive developers as it requires deep knowledge of C programming language. Regarding R/Bioconductor, the rtracklayer provides fast access to the GTF/GFF by providing the user with a GRanges object (Lawrence *et al.*, 2009).

While Python language has gained lot of popularity among bioinformaticians, only a handful of tools are available for manipulating GTF files. The gffutils package can parse and store GTF/GFF files into SQLite databases. The creation of a subsequent hierarchical models of genomic features while highly useful can be relatively time consuming. We developed the pygtftk package with the objective to provide a fast and readable way to load and manipulate GTF files within Python scripts. This package comes with the gtftk command line interface (CLI) that provide various operations to write workflows based on GTF files.

2 Implementation

2.1 The core libgtftk C library

The core of the package is written in C and exposed through a dynamic library called libgtftk. The GTF format is represented without hierarchical relationships to maximize performances. More complex operations are carried out by the libgtftk Python client.

2.2 The pygtftk Python package

The GTF class of pygtftk comes with a large number of methods. Most of these methods return a new GTF object so that they can be chained intuitively. This object can also produce two additional objects from the gtftk library including: a TAB object (representation of a matrix) and a FASTA object (representation of a FASTA file). The GTF object is integrated within the scientific Python ecosystem and can produce *pybedtools.BedTool* objects, *Bio.SeqRecord* generators or a *pandas.DataFrame* (Cock *et al.*, 2009; McKinney, 2010; Quinlan, 2014). A typical use case is proposed in Figure 1, where the transcription start site (TSS) coordinates of lincRNAs are extracted with the conditions that (i) the transcript size is above 200 nt, (ii) the number of exons is greater than 2 (iii) and the coding potential (imported from a separated file) is lower than 0.2. The TSSs are then obtained using the *get_tss()* method returning a *pybedtools.BedTool* object that can be used to extend coordinates by 1000 nucleotides in the 5′ and 3′ directions. Regarding performances, the human genome annotation in GTF format from Ensembl release 92 (~$2.7.10^6$ lines) is loaded in about 30 s while the creation of a hierarchical model using gffutils takes about 11 min [performed on Intel(R) Xeon(R) CPU E5-2640 v3, 2.60GHz]. In addition, the search engine is also highly optimized since it takes 0.6 s to select all lincRNAs from the human genome.

```
from pygtftk.gtf_interface import  GTF
from pygtftk.utils import get_example_file

# Create a GTF instance
cod_pot = get_example_file('mini_real', 'tab')[0]
gn_info = get_example_file('mini_real', 'genome')[0]
gtf = GTF(get_example_file('mini_real', 'gtf.gz'))

# Get a BedTool object containing the TSSs
# of the selected transcripts
tss = gtf.select_by_key('gene_biotype', 'lincRNA'
).select_by_transcript_size(min=200
).select_by_number_of_exons(min=2
).add_attr_from_file(feat='transcript',key='transcript_id',
                              has_header=True, new_key='coding_pot',
                              inputfile=cod_pot
).eval_numeric('coding_pot < 0.2', na_omit='.,?'
).get_tss(name=['transcript_id', 'gene_name', 'gene_id']
).slop(s=True, l=1000,r=1000, g=gn_info)
```

Fig. 1. Use case for the pygtftk package. These few lines of codes are used to extract the promoter region [(−1000, 1000) around the TSS] of LincRNAs, with the conditions that the transcripts have size greater than 200 nt, at least two exons and a coding potential (assessed by CPAT and joined from an external file) below 0.2 (Wang *et al.*, 2013)

2.3 The gtftk CLI

The pygtftk package provides a gtftk CLI with 57 subcommands. These subcommands can be used to: (i) download GTF files, (ii) edit them, (iii) mine the GTF files in various ways (select transcripts by genomic/exonic/intronic size, number of exons, associated GO term, etc.), (iv) annotate the GTF files (flagging divergent/convergent/overlapping transcripts, etc.), (v) convert them to other formats or (vi) perform epigenomic analyses by producing faceted coverage diagrams through the plotnine Python package (i.e. the recently developed Python port of ggplot2).

3 Conclusion

The pygtftk package and the associated gtftk CLI provides a new way to easily handle gene coordinates with Python. They are regularly updated and users familiar with Python and/or command-line programmes should quickly get comfortable and productive with (py)gtftk. As the GTF/GFF format is also now used for storing regulatory features and variants, this paves the way for future developments of (py)gtftk that could be an interesting framework for the integration of heterogeneous genomic data (Reese *et al.*, 2010; Zerbino *et al.*, 2018).

Acknowledgements

We thank Jacques van Helden for helpful discussion.

Funding

G.C. was supported by a fellowship from the "Fondation pour la Recherche Médicale" (FRM). S.S. and D.P. were supported by recurrent funding from INSERM and Aix Marseille Univ and by the Foundation for Cancer Research ARC [ARC PJA 20151203149] and A*MIDEX [ANR-11-IDEX-0001-02], Plan Cancer 2015 [C15076AS] and Ligue contre le Cancer Equipe Labellisée. Y.K. was supported by the Franco-Algerian partenariat Hubert Curien (PHC) Tassili [15MDU935].

Conflict of Interest: none declared.

References

Cock,P.J. *et al.* (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, **25**, 1422–1423.

Gremme,G. *et al.* (2013) GenomeTools: a comprehensive software library for efficient processing of structured genome annotations. *IEEE/ACM Trans. Comput. Biol. Bioinform.*, **10**, 645–656.

Lawrence,M. *et al.* (2009) rtracklayer: an R package for interfacing with genome browsers. *Bioinformatics*, **25**, 1841–1842.

McKinney,W. (2010). Data structures for statistical computing in python. In: van der Walt,S. and Millman,J. (eds) *Proceedings of the 9th Python in Science Conference*, pp. 51–56.

Quinlan,A.R. (2014) BEDTools: the Swiss-Army Tool for Genome Feature Analysis. *Curr. Protoc. Bioinformatics*, **47**, 1–34.

Reese,M.G. *et al.* (2010) A standard variation file format for human genome sequences. *Genome Biol.*, **11**, R88.

Wang,L. *et al.* (2013) CPAT: Coding-Potential Assessment Tool using an alignment-free logistic regression model. *Nucleic Acids Res.*, **41**, e74.

Zerbino,D.R. *et al.* (2018) Ensembl 2018. *Nucleic Acids Res.*, **46**, D754–761.