FOSS
IIT Bombay

Symposium on
FREE & OPEN SOURCE SOFTWARE
Usage in Teaching & Research

Jupyter Notebook: Software Development for Computational Genomics

Dhananjay Raman

Department of Computer Science and Engineering, IIT Bombay

Research can begin from the smallest of sparks

Presentation Outline

Introduction

Our Requirements

Our Approach

Our Results

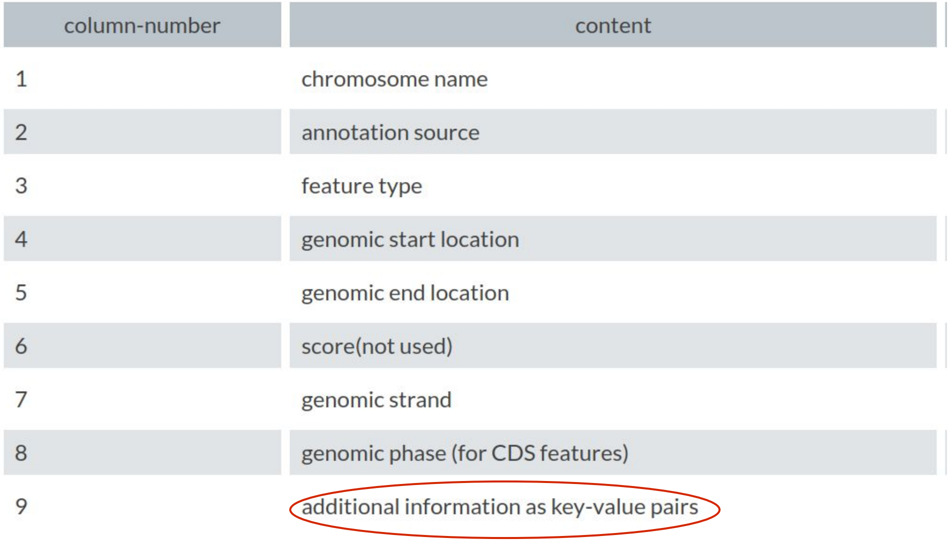
Conclusions and Challenges

Appendix

Introduction

Genomic data is stored in GTF file format:

- Very unintuitive
- Difficult to process
- Space inefficient



column-number	content
1	chromosome name
2	annotation source
3	feature type
4	genomic start location
5	genomic end location
6	score(not used)
7	genomic strand
8	genomic phase (for CDS features)
9	additional information as key-value pairs

Introduction

Existing Software:

- Pyranges (Python)
- Gffutils (Python)
- Plyranges (R)

Problems:

- Long loading times
- Long query times (for certain types of queries)

Stovner, E. B., Sætrom, P. (2019). PyRanges: efficient comparison of genomic intervals in Python. Bioinformatics, 36(3), 918-919.
Dale, R. (2011). gffutils: GFF and GTF file manipulation and interconversion. Retrieved from https://github.com/daler/gffutils.
Lee, S., Cook, D., Lawrence, M. (2019). Plyranges: A grammar of genomic data transformation. Genome Biology, 20, 1-8.

Requirements

Space Efficient

Fast

(easy to query)

Standardized

(well known)

Approach

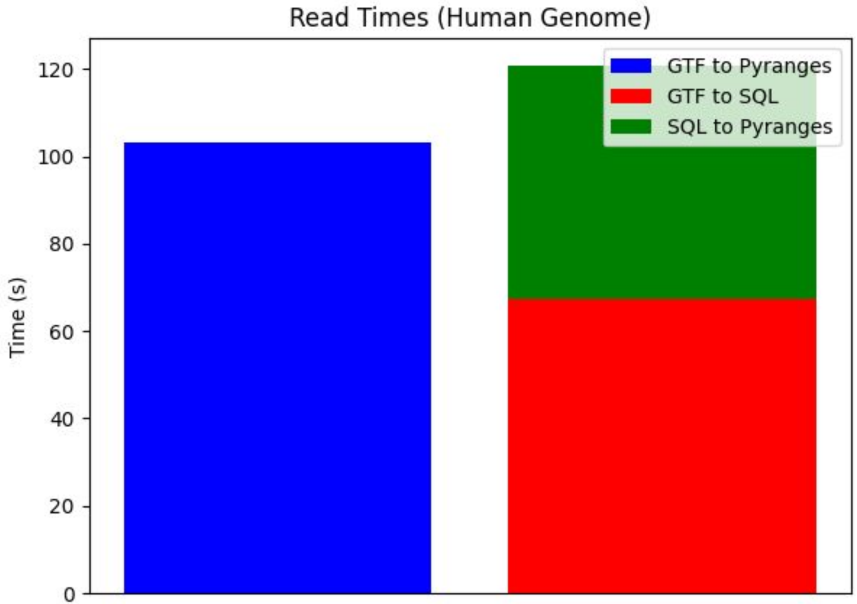
- One of the tools, Pyranges (written in Python), is very good at handling interval queries
- We can add more features to it!
- Specifically, we use a SQL database to store data and handle aggregate queries

Approach

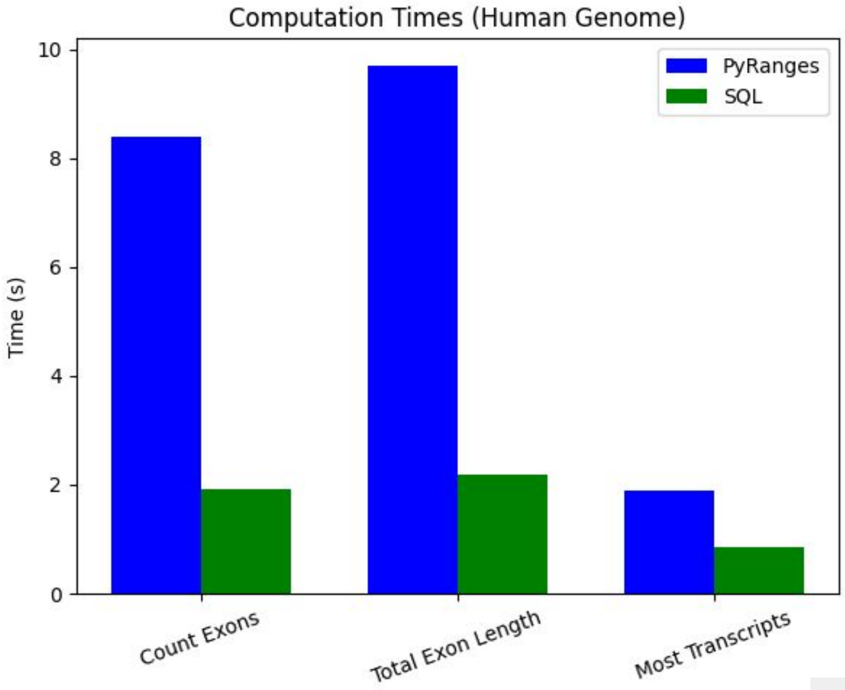
End users are not Software Engineers - abstract out all complicated details:

- Python wrapper around C++ code to convert GTF file to SQL database
- Multithreaded execution for faster processing time
- SQL queries wrapped by self-explanatory Python functions

Results



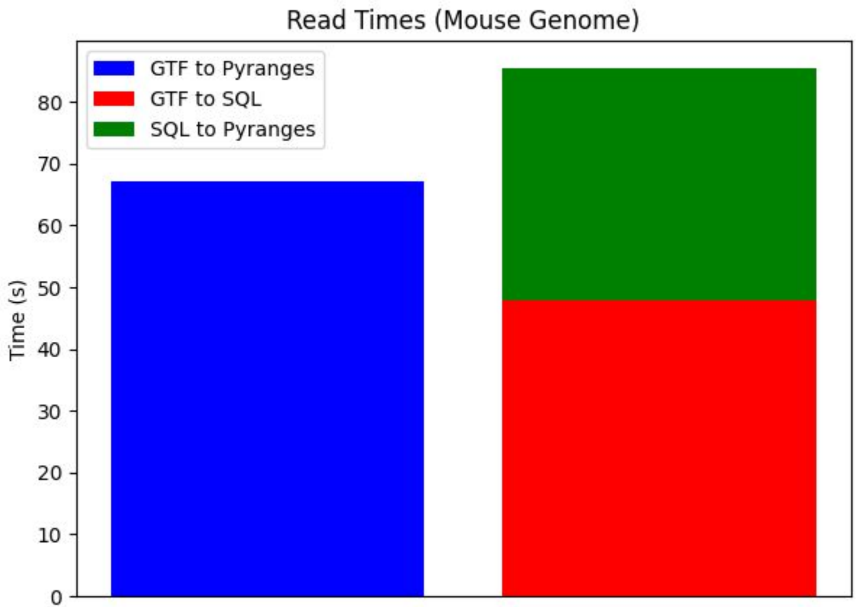
Read Times (Human Genome)

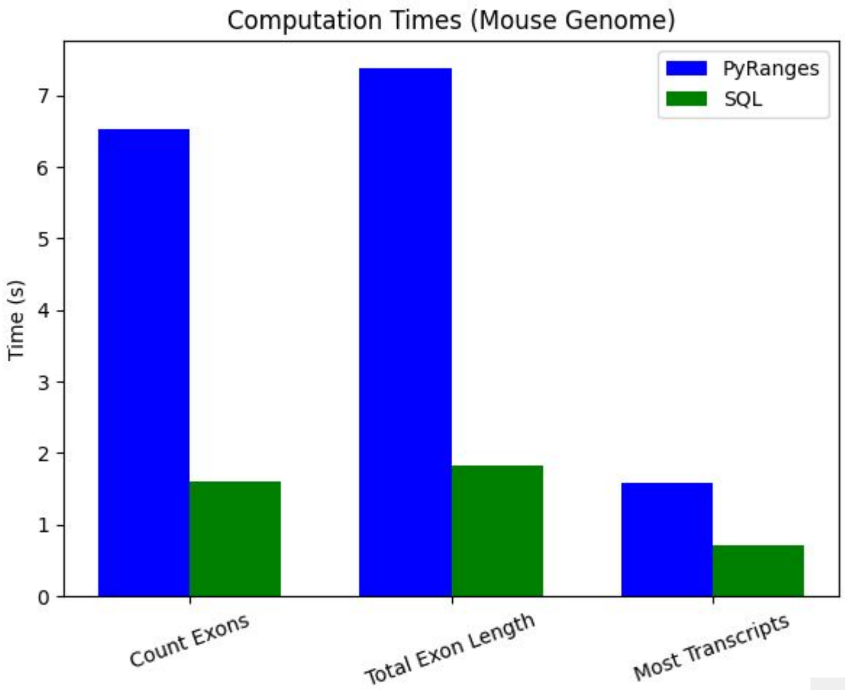


Computation Times (Human Genome)

Storage: GTF - 1.5 GB, SQL: 686 MB

Results





Storage: GTF - 1.1 GB, SQL: 581 MB

Conclusions

- SQL is space efficient
- Good at aggregate queries
- Faster to load than Pyranges object
- Takes slightly longer loading time if Pyranges object also required (say for interval queries)

Challenges

- Optimize multithreading hyperparameters
  - Number of producer and consumer threads
  - Batch Size for reading and writing
  - Choice of synchronization primitive
- Write Python wrappers for a lot of different queries
  - or find a way to automate it
- Merge with Pyranges or create an entirely different tool

Thank you!

Appendix

Email: d.raman@iitb.ac.in
Website: cse.iitb.ac.in/~dhano

Code used: https://github.com/DhanoHacks/Genomics-Tools-Analysis

Software Used: Python, C++, Jupyter, SQLite

Special Thanks: Prof. Saket Choudhary, KCDH