Bioinformatics, 39(1), 2023, btad032
https://doi.org/10.1093/bioinformatics/btad032
Advance Access Publication Date: 17 January 2023
Applications Note

Genome analysis

JBrowse Jupyter: a Python interface to JBrowse 2

Teresa De Jesus Martinez [1], Elliot A. Hershberg [1], Emma Guo[1], Garrett J Stevens [1], Colin Diesh [1], Peter Xie[1], Caroline Bridge[2], Scott Cain[2], Robin Haw[2], Robert M. Buels[1], Lincoln D. Stein[2] and Ian H. Holmes[1,*]

[1]Department of Bioengineering, Stanley Hall, University of California, Berkeley, CA 94720, USA and [2]Adaptive Oncology, Ontario Institute for Cancer Research, MaRS Centre, 661 University Avenue, Suite 510, Toronto, ON M5G 0A3, Canada

*To whom correspondence should be addressed.

Associate Editor: Tobias Marschall

Received on May 17, 2022; revised on December 10, 2022; editorial decision on January 5, 2023; accepted on January 16, 2023

Abstract

Motivation: JBrowse Jupyter is a package that aims to close the gap between Python programming and genomic visualization. Web-based genome browsers are routinely used for publishing and inspecting genome annotations. Historically they have been deployed at the end of bioinformatics pipelines, typically decoupled from the analysis itself. However, emerging technologies such as Jupyter notebooks enable a more rapid iterative cycle of development, analysis and visualization.

Results: We have developed a package that provides a Python interface to JBrowse 2's suite of embeddable components, including the primary Linear Genome View. The package enables users to quickly set up, launch and customize JBrowse views from Jupyter notebooks. In addition, users can share their data via Google's Colab notebooks, providing reproducible interactive views.

Availability and implementation: JBrowse Jupyter is released under the Apache License and is available for download on PyPI. Source code and demos are available on GitHub at https://github.com/GMOD/jbrowse-jupyter.

Contact: ihh@berkeley.edu

1 Introduction

Genome browsers are popular tools for displaying genome annotations (Kent *et al.*, 2002). Web-based browsers, particularly JavaScript tools such as IGV.js (Robinson *et al.*, 2023) and JBrowse 2 (Diesh *et al.*, 2022), allow for a more responsive user experience by performing computation on the client, reducing server and network load. These genome browsers are versatile and widely adopted (Wang *et al.*, 2013).

Although JBrowse is straightforward to deploy, the need to administer a web server could present a barrier to entry for some users, particularly those more comfortable working within Python or R. Motivated by this, we recently released JBrowseR, a version of JBrowse designed to run in an R environment or a Shiny app (Hershberg *et al.*, 2021). This was enabled by the React-based architecture of JBrowse 2, which allows JBrowse components to be re-used programmatically.

Python, similarly to R, is one of the most widely used programming languages in computational biology. The Jupyter notebook, a web-based interactive environment for Python (and other languages), is one of the leading open-source platforms for data analysis (Cock *et al.*, 2009; Shen, 2014). Several tools exist for interacting with genome browsers in Python. D3GB and IGV.js's integration with Dash provide interactive linear displays of the genome whereas pygbrowse (https://github.com/phageghost/python-genome-browser) provides static snapshots (Barrios and Prieto, 2017; Robinson *et al.*, 2023). Mango, Gosling's Python library Gos and wrappers to IGV.js including ipyigv (https://github.com/QuantStack/ipyigv) and igv-notebook (https://github.com/igvteam/igv-notebook) also exist and provide integration to Jupyter notebooks (L'Yi *et al.*, 2022; Manz *et al.*, 2022; Morrow *et al.*, 2018). However, some of these tools either have limited extensibility and or customization capabilities including limited color theming, text indexing for text searching and support of additional views such as the Circular Genome View.

To broaden the options for users of genome browsers in Python environments, we created JBrowse Jupyter, a Python package for configuring, creating and launching JBrowse views from Jupyter notebooks and other Python applications. JBrowse Jupyter uses the Dash framework, a bridge from React components to Python code (Hossain, 2019). The JBrowse Jupyter package currently allows users to embed two of the most popular JBrowse 2 views: the Linear Genome View and the Circular Genome View. The package aims to make it easier for users to configure genome browsers in Jupyter notebooks and to enable interactive exploration and visualization of genomic data from within Python.

2 Materials and methods

JBrowse Jupyter makes use of Dash JBrowse, a collection of Dash components for JBrowse 2 views, to render React components inside

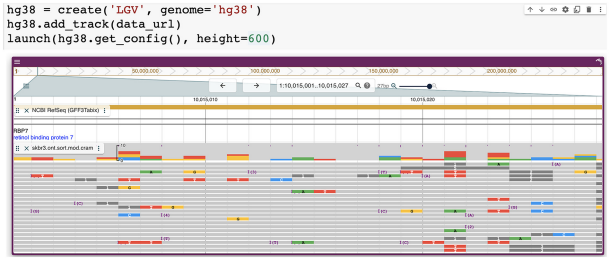

Fig. 1. A close-up of the human genome is displayed by JBrowse 2 in a Colab notebook

Jupyter notebooks. Dash wraps JBrowse's React embeddable components (such as the Linear Genome View) for use in any Python Dash environment. JBrowse Jupyter's integration with Dash and Dash callbacks makes it possible to control the state of the JBrowse views from the controlling Python application. Python code can control a JBrowse instance by using the JBrowseConfig API. Examples of supported operations include updating an assembly for viewing, adding tracks, deleting tracks, enabling text searching and changing default view settings. JBrowse Jupyter supports loading track data from files or from Pandas DataFrames. Most genomic data file formats are supported including FASTA, BAM, BigWig, CRAM, GFF3, VCF and more.

Users can take advantage of the existing tools available for Jupyter notebooks. Visual Studio Code provides a Jupyter extension that allows users to create and run cells with the embedded genome browser in their development environments. With Binder and Colab, the community is able to share reproducible notebooks with configured JBrowse views (Bisong, 2019; Project Jupyter *et al.*, 2018).

The package is distributed by the Python Package Index and is compatible with Python 3.6 and higher. The project follows continuous integration and continuous delivery practices that enable automated deployments. Flake8 and Pytest support automatic testing and linting to ensure code quality, and Sphinx automatically generates documentation.

3 Discussion

JBrowse Jupyter is a flexible tool that reduces the effort required to embed an interactive genome browser in a Jupyter notebook: a user can launch a JBrowse View with fewer than 10 lines of code (Fig. 1).

We provide several Python Dash applications and Jupyter notebooks along with source code to show the versatility of JBrowse Jupyter. The browser.ipynb notebook demonstrates how to add a track from a Pandas DataFrame, add tracks in bulk from distinct data files and specify initial location in the configuration of the Linear Genome View. The local_support.ipynb notebook demonstrates how to utilize local data to configure JBrowse views by leveraging the development http server provided by JBrowse Jupyter.

To illustrate application to real data, we provide the skbr3.ipynb Jupyter notebook that visualizes genome sequencing and annotation data from the SK-BR-3 cell line (Nattestad *et al.*, 2017). This demo shows how to create an application that allows users to navigate around genomic locations involved in gene fusions.

Funding

This work was supported by the National Institutes of Health (NIH) [HG004483 and GM080203].

Conflict of Interest: none declared.

Data availability

The package is available for download on PyPI. Source code and demos can be found at https://github.com/GMOD/jbrowse-jupyter.

References

Barrios,D. and Prieto,C. (2017) D3GB: an interactive genome browser for R, Python, and WordPress. *J. Comput. Biol.*, **24**, 447–449.

Bisong,E. (2019) Google colaboratory. *In: Building Machine Learning and Deep Learning Models on Google Cloud Platform*. Apress, Berkeley, CA. pp. 59–64. https://doi.org/10.1007/978-1-4842-4470-8_7.

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

Diesh,C. *et al.* (2022) *JBrowse 2: a modular genome browser with views of synteny and structural variation*. BioRxiv, Cold Spring Harbor Laboratory, https://doi.org/10.1101/2022.07.28.501447.

Hershberg,E.A. *et al.* (2021) JBrowseR: an R interface to the JBrowse 2 genome browser. *Bioinformatics*, **37**, 3914–3915.

Hossain,S. (2019) Visualization of bioinformatics data with dash bio. In: *Proceedings of the 18th Python in Science Conference*. SciPy. https://doi.org/10.25080/Majora-7ddc1dd1-012.

Kent,W.J. *et al.* (2002) The human genome browser at UCSC. *Genome Res.*, **12**, 996–1006.

L'Yi,S. *et al.* (2022) Gosling: a grammar-based toolkit for scalable and interactive genomics data visualization. *IEEE Trans. Vis. Comput. Graph*, **28**, 140–150.

Manz,T. *et al.* (2022) GOS: a declarative library for interactive genomics visualization in Python. *OSF Preprints*. https://doi.org/10.31219/osf.io/yn3ce.

Morrow,A.K. *et al.* (2018) *Mango: distributed visualization for genomic analysis*. BioRxiv, Cold Spring Harbor Laboratory, https://doi.org/10.1101/360842.

Nattestad,M. *et al.* (2018) Complex rearrangements and oncogene amplifications revealed by long-read DNA and RNA sequencing of a breast cancer cell line. *Genome Res.*, **28**, 1126–1135.

Project Jupyter *et al.* (2018) Binder 2.0 - reproducible, interactive, sharable environments for science at scale. In: *Proceedings of the 17th Python in Science Conference*, pp. 113–120. https://doi.org/10.25080/Majora-4af1f417-011.

Robinson,J.T. *et al.* (2023) igv.js: an embeddable JavaScript implementation of the integrative genomics viewer (IGV). *Bioinformatics*, **39**(1), btac830.

Shen,H. (2014) Interactive notebooks: sharing the code. *Nature*, **515**, 151–152.

Wang,J. *et al.* (2013) A brief introduction to web-based genome browsers. *Brief. Bioinform.*, **14**, 131–143.