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
[2]Adaptive Oncology, Ontario Institute for Cancer Research,Toronto, ON M5G 0A3, Canada

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: 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

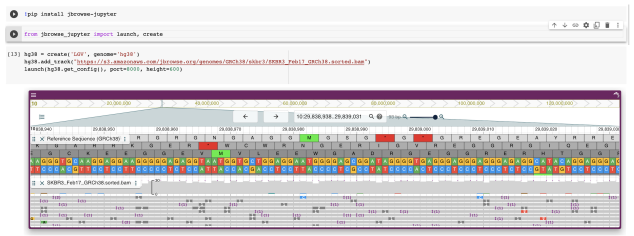
1 Introduction

Genome browsers are popular tools for displaying genome annotations [1]. Web-based browsers, particularly JavaScript tools such as IGV.js [2] and JBrowse [3], 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 [4].

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 [5]. This was enabled by the React-based architecture of JBrowse 2, which makes possible the development of embeddable components for distinct JBrowse views.

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 [6, 7]. 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 provides static snapshots [2, 8, 9]. However, these tools have limited extensibility or interactivity.

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 [10]. 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.



```
!pip install jbrowse-jupyter
```

```
from jbrowse_jupyter import launch, create
```

```
[13] hg38 = create('LGV', genome='hg38')
     hg38.add_track("https://s3.amazonaws.com/jbrowse.org/genomes/GRCh38/skbr3/SKBR3_Feb17_GRCh38.sorted.bam")
     launch(hg38.get_config(), port=8000, height=600)
```

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

2 Methods

JBrowse Jupyter makes use of Dash JBrowse, a collection of Dash components for JBrowse 2 views, to render React components inside 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, enabling text searching, and changing default view settings. JBrowse Jupyter supports loading track data from files or from Python data frames. Most genomic data file formats are supported.

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 [11, 12].

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 ten lines of code (Fig1).

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 Data frame, add tracks in bulk from distinct data files, and specify initial location in the configuration of the Linear Genome View.

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 [13]. This demo shows how to create an application that allows users to navigate around genomic locations involved in gene fusions.

4 Availability

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

Funding

This work was funded by NIH grants HG004483 and GM080203.

References

[1] W James Kent, Charles W Sugnet, Terrence S Furey, Krishna M Roskin, Tom H Pringle, Alan M Zahler, and David Haussler. The human genome browser at UCSC. *Genome Res.*, 12(6):996–1006, June 2002.

[2] James T Robinson, Helga Thorvaldsdóttir, Douglass Turner, and Jill P Mesirov. igv.js: an embeddable JavaScript implementation of the integrative genomics viewer (IGV). May 2020.

[3] Robert Buels, Eric Yao, Colin M Diesh, Richard D Hayes, Monica Munoz-Torres, Gregg Helt, David M Goodstein, Christine G Elsik, Suzanna E Lewis, Lincoln Stein, and Ian H Holmes. JBrowse: a dynamic web platform for genome visualization and analysis. *Genome Biol.*, 17(1):1–12, April 2016.

[4] Jun Wang, Lei Kong, Ge Gao, and Jingchu Luo. A brief introduction to web-based genome browsers. *Brief. Bioinform.*, 14(2):131–143, July 2012.

[5] Elliot A Hershberg, Garrett Stevens, Colin Diesh, Peter Xie, Teresa De Jesus Martinez, Robert Buels, Lincoln Stein, and Ian Holmes. JBrowseR: an R interface to the JBrowse 2 genome browser. *Bioinformatics*, 37(21):3914–3915, July 2021.

[6] Helen Shen. Interactive notebooks: Sharing the code. *Nature*, 515(7525):151–152, November 2014.

[7] Peter J A Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J L de Hoon. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, March 2009.

[8] David Barrios and Carlos Prieto. D3GB: An interactive genome browser for R, Python, and WordPress. *J. Comput. Biol.*, 24(5):447–449, May 2017.

[9] phageghost. GitHub - phageghost/python-genome-browser: Tools for making plots of genomic datasets in a genome-browser-like format. `https://github.com/phageghost/python-genome-browser`, 2018. Accessed: 2022-3-27.

[10] Shammamah Hossain. Visualization of bioinformatics data with dash bio. In *Proceedings of the 18th Python in Science Conference*. SciPy, 2019.

[11] Project Jupyter, Matthias Bussonnier, Jessica Forde, Jeremy Freeman, Brian Granger, Tim Head, Chris Holdgraf, Kyle Kelley, Gladys Nalvarte, Andrew Osheroff, M Pacer, Yuvi Panda, Fernando Perez, Benjamin Ragan-Kelley, and Carol Willing. Binder 2.0 - reproducible, interactive, sharable environments for science at scale. In *Proceedings of the 17th Python in Science Conference*, pages 113–120, 2018.

[12] Ekaba Bisong. Google Colaboratory. *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pages 59–64, 2019.

[13] Maria Nattestad, Sara Goodwin, Karen Ng, Timour Baslan, Fritz J Sedlazeck, Philipp Rescheneder, Tyler Garvin, Han Fang, James Gurtowski, Elizabeth Hutton, Elizabeth Tseng, Chen-Shan Chin, Timothy Beck, Yogi Sundaravadanam, Melissa Kramer, Eric Antoniou, John D McPherson, James Hicks, W Richard McCombie, and Michael C Schatz. Complex rearrangements and oncogene amplifications revealed by long-read DNA and RNA sequencing of a breast cancer cell line. August 2017.