Bioinformatics, 2026, **42**(2), btag035
https://doi.org/10.1093/bioinformatics/btag035
Published: 22 January 2026
Applications Note | Genome analysis

OXFORD

Uchimata: a toolkit for visualization of 3D genome structures on the web and in computational notebooks

David Kouřil[1], Trevor Manz[1], Tereza Clarence[2,3,4,5], Nils Gehlenborg[1,*]

[1]Department of Biomedical Informatics, Harvard Medical School, Boston, MA 02115, United States
[2]Center for Disease Neurogenomics, Icahn School of Medicine at Mount Sinai, New York, NY 10029, United States
[3]Friedman Brain Institute, Icahn School of Medicine at Mount Sinai, New York, NY 10029, United States
[4]Department of Psychiatry, Icahn School of Medicine at Mount Sinai, New York, NY 10029, United States
[5]Department of Genetics and Genomic Sciences, Icahn School of Medicine at Mount Sinai, New York, NY 10029, United States
*Corresponding author. Department of Biomedical Informatics, Harvard Medical School, 10 Shattuck Street, Boston, MA 02115, United States.
E-mail: nils@hms.harvard.edu
Associate Editor: Can Alkan

Abstract

Summary: **Uchimata is a toolkit for visualization of 3D structures of genomes. It consists of two packages: a Javascript library facilitating the rendering of 3D models of genomes, and a Python widget for visualization in Jupyter Notebooks. Main features include an expressive way to specify visual encodings, and filtering of 3D genome structures based on genomic semantics and spatial aspects. Uchimata is designed to be highly integratable with biological tooling available in Python.**

Availability and implementation: **Uchimata is released under the MIT License. The Javascript library is available on NPM, while the widget is available as a Python package hosted on PyPI. The source code for both is available publicly on Github (https://github.com/hms-dbmi/uchimata and https://github.com/hms-dbmi/uchimata-py) and Zenodo (https://doi.org/10.5281/zenodo.17831959 and https://doi.org/10.5281/zenodo.17832045). The documentation with examples is hosted at https://hms-dbmi.github.io/uchimata/.**

1 Introduction

Alongside sequencing-based technologies developed to probe the spatial conformation of genomes—such as the influential Hi-C method (Lieberman-Aiden *et al.* 2009)—there are ongoing efforts to produce concrete 3D models that depict genome folding. Although research on macromolecular structures has long benefited from sustained investment in resources such as the Protein Data Bank (Berman *et al.* 2000) and visualization tools like PyMOL (Schrödinger, LLC, 2015) and Mol* (Sehnal *et al.* 2021), comparable tools for analysis and visualization of physical *genome* structures remain limited.

There are two principal approaches to generating structural models of genomes (Imakaev *et al.* 2015): *data-driven* methods, which produce structures that satisfy constraints derived from input data (e.g. Hi-C), and methods that simulate structures *de novo* based on mechanistic principles. Li *et al.* (2023) review reconstruction methods and highlight how resulting 3D structures can lead to novel insights. Portillo-Ledesma *et al.* (2023) further emphasize that the choice of model resolution and simulation technique restricts the scale of the phenomena that can be investigated. Oluwadare *et al.* (2019) summarize approaches for reconstructing chromosome- and genome-level structures from Hi-C data. Notable examples of existing structures include Duan *et al.*'s yeast genome model (Duan *et al.* 2010), Stevens *et al.*'s mammalian (mouse) genome structures constructed from single-cell Hi-C (Stevens *et al.* 2017), and Tan *et al.*'s genome structures of single diploid human cells (Tan *et al.* 2018). Recent studies show promise in using machine learning techniques to infer 3D genome structures (Schuette *et al.* 2025). Beyond individual publications, there have been efforts to develop centralized databases of 3D genome models (Oluwadare *et al.* 2020), although community adoption remains limited.

Visualizing specialized 3D data can be challenging for computational biologists lacking computer graphics expertise, who depend on tools that abstract low-level rendering details. Although several tools have been developed for *visualizing 3D genome* structural data—such as Genome3D (Asbury *et al.* 2010), 3DGB (Butyaev *et al.* 2015), GMOL (Nowotny *et al.* 2016), HiC-3DViewer (Djekidel *et al.* 2017), and CSynth (Todd *et al.* 2021)—most exist as standalone desktop or web applications, making them difficult to integrate into notebook-based analytical workflows, to adapt for case-specific applications, and to customize beyond the visual choices predefined by their developers.

To address the lack of dedicated visualization tools in computational notebooks, scientists often repurpose general plotting

Received: 15 September 2025. **Revised:** 8 December 2025. **Accepted:** 8 January 2026

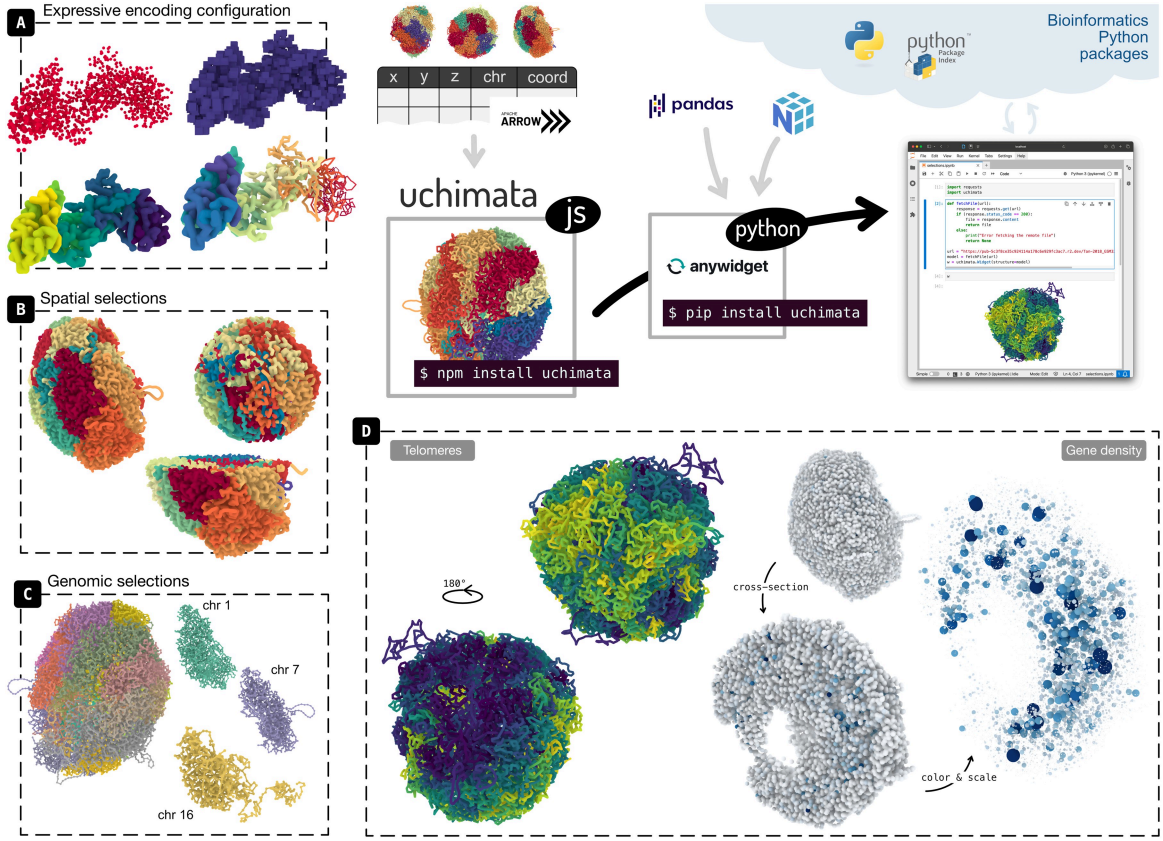

Figure 1 Uchimata consists of two packages, Javascript and Python, intended to cover use cases for visualization of 3D genome models across the web and computational notebooks. Key features include expressive configuration of visual encoding (A), selections based on spatial attributes (B), and selections based on genomic coordinates (C). Two use cases illustrating situations where 3D visualization brings new insights (D): mapping bin coordinates to continuous color scale shows concentration of telomeres on one side of the structures (left), and encoding gene density to color and scale of the bin marks shows concentration of gene-rich regions within the structures, further from the boundary (right).

libraries or tools originally designed for protein structures. For instance, nglutils (https://github.com/mirnylab/nglutils) builds on the molecular viewer nglview (Nguyen *et al.* 2018) to support interactive visualization of genome structures within notebooks. Similarly, some simulation packages for 3D genome structures recommend using general visualization libraries, such as matplotlib (Hunter 2007) or Fresnel (https://github.com/glotzerlab/fresnel). Biologists often reuse molecular file formats (e.g. PDB) to store 3D genome structures, but differing interpretations of these standards for genomic data can cause incompatibilities even within the same format.

We developed **uchimata**, a toolkit for visualizing 3D genome structures across the web and computational notebook environments (Fig. 1). Its unique combination of features exceeds the capabilities of existing solutions: (i) **A declarative specification for visual encoding** provides flexible and expressive customization of the visual depiction and allows mapping genomic data onto the 3D structure; (ii) **An API for spatial and genomics filtering** of 3D genome structures supports operations such as applying cutting planes or selecting genomic ranges. (iii) **Dual JavaScript and Python availability** enables use both during analysis—close to the modeling and analysis environment—and as an integrated component within larger web applications, such as data portals.

At its core, uchimata operates on tabular data that link genomic regions to XYZ coordinates, making it suitable for a broad range of applications, including Hi-C-based simulations, *de novo* modeling, and emerging approaches such as imaging-based methods.

2 Methods

2.1 Design of the uchimata toolkit

The design of uchimata is guided by the **principle of composability** (Manz 2024). To support long-term maintainability and enable the reuse of visualization tools as components in novel scenarios, each tool should focus on a narrow and well-defined set of features. This contrasts with the approach of many other tools in this domain, which combine 3D data with Hi-C matrices and genomic browsers, ultimately reducing reusability. In uchimata, our focus is on visualizing 3D models, while related representations are deliberately left to other specialized tools. This modular approach allows users to select and combine components to construct interfaces and pipelines tailored to their specific requirements, avoiding the pitfalls of monolithic software that becomes progressively difficult to extend and maintain.

The second guiding principle is the **separation of concerns** and leveraging strengths of individual programming environments. Over time, the Javascript and Python ecosystems have developed distinct competencies. The web platform excels at

building user interfaces and offers a wide range of visualization libraries. Its core technologies—HTML, CSS, and Javascript—benefit from standardization and strong backward compatibility. Python, by contrast, is well suited for data wrangling and analysis, and has become a dominant platform in computational biology. Some libraries, such as NumPy (Harris *et al.* 2020) and pandas (McKinney 2010), introduced concepts now considered core to the Python ecosystem, even though they exist outside the language and its standard library. Similarly, we advocate separating the tasks of building the 3D genome models (e.g. Hi-C-based simulations) and visualizing them. This separation must be facilitated by robust mechanisms for exchanging data between tools.

Therefore, uchimata embraces existing **data standards**, some of which reach beyond genomics or bioinformatics. This approach allows it to benefit from mature infrastructure developed by broader communities. For example, uchimata uses Apache Arrow (https://arrow.apache.org) as its in-memory representation of 3D genome structures. Rather than supporting a fixed set of file formats, the uchimata Javascript library accepts only structures as Arrow tables, whereas the Python package also ingests canonical scientific Python structures such as NumPy array and pandas DataFrame, internally converting both to Arrow. This approach addresses incompatibilities in existing file formats for genome structures. Several tools reinterpret the PDB format, originally designed for atomistic models, to store 3D genome structures, but each adopts its own conventions. One concrete issue is that genomic coordinates can span up to nine digits, exceeding the standard column widths of the text-based PDB format and forcing users to use non-standard fields. Representing 3D genome models as Arrow tables leverages a standardized binary format widely supported by analytical tools. Our adoption of Apache Arrow is inspired by other efforts to unify bioinformatics file formats, e.g. Oxbow (Abdennur *et al.* 2025).

The main downside is the need to convert existing structures to Arrow. Due to the subtle differences among file formats, fully automatic conversion is rarely possible. We provide examples of how to perform this conversion in the Supplemental Materials, available as supplementary data at *Bioinformatics* online.

2.2 Implementation

The core web uchimata library (https://github.com/hms-dbmi/uchimata) is implemented using Typescript, transformed into a standard Javascript module, and hosted on NPM. The Jupyter notebook widget (https://github.com/hms-dbmi/uchimata-py) is built using anywidget (Manz *et al.* 2024) and uses the Javascript library to serve the canvas within a notebook's cell. The 3D graphics rendering is accomplished through three.js, currently using the WebGL backend. We use DuckDB (Raasveldt and Mühleisen 2019) to perform a variety of queries to filter the Apache Arrow tables representing the 3D structures and associated data.

3 Results

The development of uchimata was driven by the aim to support a broad range of end-use scenarios. We focused on core functionality that could be reused in web-based applications with case-specific interfaces, without overloading the core library with features relevant only to a narrow subset of users. We also envisioned seamless integration of uchimata into data portals.

A second major group of use cases involves visualization in exploratory stages, such as during simulations and in downstream analyses of simulated data. While the library can be used directly in Javascript-based notebook environments such as Observable Notebooks (https://observablehq.com/platform/notebooks), much computational notebook work today is carried out in Python, leveraging its extensive scientific software ecosystem.

3.1 Integration in web applications and Javascript-based notebooks

In the first scenario, uchimata functions like any other Javascript library in a web environment, enabling visualization of 3D genome structural data. To show uchimata's ability to support novel visualizations, we use it within Observable Notebooks—a literate programming environment, similar to Jupyter Notebooks, that executes Javascript code in interactive cells. A set of examples is available at https://hms-dbmi.github.io/uchimata/, source code for these examples is in the uchimata Github repository, under the "docs" folder. At its core, a 3D genome structure is a list of XYZ coordinates, sometimes associated with genomic coordinates. How these data items are visually represented is left to the user, who specifies a *view config* that maps items to concrete *marks*, such as spheres or cubes, and their visual features such as *color* and *scale*. This manner of declaring visual encodings is inspired by the grammar-of-graphics approach (Wilkinson 2005). A *scene* contains an array of structures, each paired with its corresponding view config, which together define how the structures are rendered. The view config lets users define how each structure is depicted by mapping data columns to visual attributes. For example, mapping a column containing the chromosome information to the color channel assigns a different color to each chromosome. Such annotations can be applied at any level (e.g. compartment, TADS) and depend solely on the data supplied with the structure.

We envision uchimata being integrated into use case-specific web applications and genome browsers. We recently used uchimata to extend the Gosling grammar (L'Yi *et al.* 2022) with support for 3D genome models (Kouřil *et al.* 2026), thereby implementing the previously recognized "spatial" layout for genomic coordinate systems (Nusrat *et al.* 2019) that was missing in existing grammar-based tools. This grammar-of-graphics approach allows for both expressiveness and reproducibility, and by unifying 3D representations of genomes with conventional genomic data views, it opens new avenues for exploring efficient visual linking between distant genomic loci.

3.2 Interoperability with existing bioinformatics workflows in Python

Computational notebooks play an important role in day-to-day analytical work of genomics researchers (van den Brandt *et al.*

2025). In the second scenario, uchimata can be used as a widget for Python-based computational notebooks.

Python is often used for chromatin simulations, where visual inspection is typically the first step in assessing the results. To shorten the iteration loop, a visualization tool should be available directly within the simulation environment, enabling biologists to perform basic checks, adjust parameters, and rerun simulations. The uchimata widget offers multiple ways to input data which aligns with in-memory storage practices in Python-based computation notebooks. Most relevant for Jupyter environments are widely used data structures such as NumPy arrays and pandas DataFrame.

Furthermore, the availability through Python allows for integration with existing bioinformatics tools. For example, we can combine uchimata with bioframe (Open2C *et al.* 2024) to apply genomic range selections on 3D genome structures. Bioframe offers functionality for loading genomic data from typical formats such as BED, GFF, or GTF, and performing a variety of interval operations, representing the ranges as pandas DataFrames. Uchimata then accepts these dataframes as selection queries and outputs corresponding bins of the 3D structure. The user can thus apply the same range as selection across multiple different 3D structures, facilitating comparison across an ensemble of simulated structures. In addition to genomics-based filtering, uchimata supports spatial selections (Fig. 1B), currently implementing cutting-plane (cross-section) operations and selecting spherical neighborhoods of a specified radius.

Figure 1D highlights two examples that demonstrate how visualization of 3D models reveals insights about genome structure. First, we use Stevens *et al*.'s mouse cell structures (Stevens *et al.* 2017) and encode each chromosome's coordinates with a continuous color scale. This results in clear identification of locations where telomeres concentrate (Fig. 1D, left). Second, we aggregate gene annotation loaded from a GTF into bins matching the resolution of the structure. We then map this gene density data to both color and scale of the marks representing the bins (Fig. 1D, right). Scaling the marks can act as a form of removing occlusion and highlighting inner structure. The biological insight in this example is that gene-rich regions concentrate in the center of the genome structure, further from the border.

4 Conclusion

While the heatmap representation (i.e. the contact matrices resulting from Hi-C) remains the most direct experimental way to observe how whole genomes fold in nuclear space, we believe that the 3D structural representation offers a complementary benefit. Its primary strength is that it contextualizes genomic data within 3D space, which can lead to hypotheses related to specific patterns observed in the actual 3D structure. With uchimata, we contribute software that makes it easier to visualize this type of genomics data. With this crucial infrastructure in place, we intend to further investigate means of linking between traditional genome browser views, dense multiscale matrix viewers, and 3D structure visualizations. On the rendering side, we plan to adopt the WebGPU API as it becomes broadly supported across major web browsers.

Author contributions

David Kouřil (Conceptualization [lead], Methodology [equal], Software [equal], Validation [equal], Visualization [equal], Writing—original draft [lead], Writing—review & editing [equal]), Trevor Manz (Methodology [equal], Software [equal], Writing—review & editing [equal]), Tereza Clarence (Conceptualization [equal], Data curation [equal], Validation [equal], Writing—review & editing [equal]), and Nils Gehlenborg (Conceptualization [equal], Funding acquisition [equal], Supervision [equal], Writing—review & editing [equal])

Supplementary material

Supplementary material is available at *Bioinformatics* online.

Conflict of interests

N.G. is a co-founder and equity owner of Datavisyn.

Funding

This work was supported in part by the National Institutes of Health [R01 HG011773 and UM1 HG011536].

Data availability

No new data were generated or analysed in support of this research.

References

Abdennur N, Bzura C, Manz T *et al. abdenlab/oxbow: py-oxbow@v0.5.0*. Zenodo, 2025.

Asbury TM, Mitman M, Tang J *et al.* Genome3D: a viewer-model framework for integrating and visualizing multi-scale epigenomic information within a three-dimensional genome. *BMC Bioinformatics* 2010;**11**:444.

Berman HM, Westbrook J, Feng Z *et al.* The protein data bank. *Nucleic Acids Res* 2000;**28**:235–42.

Butyaev A, Mavlyutov R, Blanchette M *et al.* A low-latency, big database system and browser for storage, querying and visualization of 3D genomic data. *Nucleic Acids Res* 2015;**43**:e103.

Djekidel MN, Wang M, Zhang MQ *et al.* HiC-3DViewer: a new tool to visualize hi-C data in 3D space. *Quant Biol* 2017;**5**:183–90.

Duan Z, Andronescu M, Schutz K *et al.* A three-dimensional model of the yeast genome. *Nature* 2010;**465**:363–7.

Harris CR, Millman KJ, van der Walt SJ *et al.* Array programming with NumPy. *Nature* 2020;**585**:357–62.

Hunter JD. Matplotlib: a 2D graphics environment. *Comput Sci Eng* 2007;**9**:90–5.

Imakaev MV, Fudenberg G, Mirny LA. Modeling chromosomes: beyond pretty pictures. *FEBS Lett* 2015;**589**:3031–6.

Kouřil D, Manz T, L'Yi S *et al.* Design space and declarative grammar for 3D genomic data visualization. *IEEE Trans Visual Comput Graphics* 2026;**32**:1–11.

Li Z, Portillo-Ledesma S, Schlick T. Techniques for and challenges in reconstructing 3D genome structures from 2D chromosome conformation capture data. *Curr Opin Cell Biol* 2023;**83**:102209.

Lieberman-Aiden E, van Berkum NL, Williams L *et al.* Comprehensive mapping of long-range interactions reveals folding principles of the human genome. *Science* 2009;**326**:289–93.

L'Yi S, Wang Q, Lekschas F *et al.* Gosling: a grammar-based toolkit for scalable and interactive genomics data visualization. *IEEE Trans Vis Comput Graph* 2022;**28**:140–50.

Manz T. Composable visualization systems for biological data. PhD thesis, Harvard University, 2024.

Manz T, Abdennur N, Gehlenborg N. Anywidget: reusable widgets for interactive analysis and visualization in computational notebooks. *JOSS* 2024;**9**:6939.

McKinney W. Data structures for statistical computing in python. In: *Proceedings of the Python in Science Conference*. Austin, TX, USA: SciPy, 2010, 56–61.

Nguyen H, Case DA, Rose AS. NGLview-interactive molecular graphics for jupyter notebooks. *Bioinformatics* 2018;**34**:1241–2.

Nowotny J, Wells A, Oluwadare O *et al.* GMOL: an interactive tool for 3D genome structure visualization. *Sci Rep* 2016;**6**:20802.

Nusrat S, Harbig T, Gehlenborg N. Tasks, techniques, and tools for genomic data visualization. *Comput Graph Forum* 2019;**38**:781–805.

Oluwadare O, Highsmith M, Cheng J. An overview of methods for reconstructing 3-D chromosome and genome structures from hi-C data. *Biol Proced Online* 2019;**21**:7.

Oluwadare O, Highsmith M, Turner D *et al.* GSDB: a database of 3D chromosome and genome structures reconstructed from hi-C data. *BMC Mol Cell Biol* 2020;**21**:60.

Open2C, Abdennur N, Fudenberg G, Flyamer IM, *et al.* Bioframe: operations on genomic intervals in pandas dataframes. *Bioinformatics* 2024;**40**:btae088.

Portillo-Ledesma S, Li Z, Schlick T. Genome modeling: from chromatin fibers to genes. *Curr Opin Struct Biol* 2023;**78**:102506.

Raasveldt M, Mühleisen H. DuckDB. In *Proceedings of the 2019 International Conference on Management of Data*, New York, NY, USA: ACM, 2019.

Schrödinger LLC. *The PyMOL Molecular Graphics System, Version 1.8*. 2015. https://www.pymol.org/support.html#citing

Schuette G, Lao Z, Zhang B. ChromoGen: diffusion model predicts single-cell chromatin conformations. *Sci Adv* 2025;**11**:eadr8265.

Sehnal D, Bittrich S, Deshpande M *et al.* Mol viewer: modern web app for 3D visualization and analysis of large biomolecular structures. *Nucleic Acids Res* 2021;**49**:W431–7.

Stevens TJ, Lando D, Basu S *et al.* 3D structures of individual mammalian genomes studied by single-cell hi-C. *Nature* 2017;**544**:59–64.

Tan L, Xing D, Chang C-H *et al.* Three-dimensional genome structures of single diploid human cells. *Science* 2018;**361**:924–8.

Todd S, Todd P, McGowan SJ *et al.* CSynth: an interactive modelling and visualization tool for 3D chromatin structure. *Bioinformatics* 2021;**37**:951–5.

van den Brandt A, L'Yi S, Nguyen HN *et al.* Understanding visualization authoring techniques for genomics data in the context of personas and tasks. *IEEE Trans Vis Comput Graph* 2025;**31**:1180–90.

Wilkinson L. *The Grammar of Graphics. Statistics and Computing*. 2nd edn. New York, NY: Springer, 2005.

Bioinformatics, 2026, 42, 1–5
https://doi.org/10.1093/bioinformatics/btag035
Applications Note