Article

Not peer-reviewed version

Nightingale: Web components for protein feature visualization

Gustavo A Salazar [*] , Aurelien Luciani , Xavier Watkins , Swaathi Kandasaamy , Daniel L Rice , Matthias Blum , Alex Bateman , Maria Martin

Posted Date: 25 May 2023

doi: 10.20944/preprints202305.1797.v1

Keywords: Visualization, Bioinformatics, Proteins, Software, Web services

Article

Nightingale: Web Components for Protein Feature Visualization

Gustavo A Salazar *, Aurelien Luciani, Xavier Watkins, Swaathi Kandasaamy, Daniel L Rice, Matthias Blum, Alex Bateman and Maria Martin

European Bioinformatics Institute, European Molecular Biology Laboratory (EMBL-EBI), Wellcome Genome Campus, Hinxton CB10 1SD, UK

* Correspondence: email:gsalazar@ebi.ac.uk or gsalazar@ebi.ac.uk

Abstract: Motivation: The visualization of biological data is a fundamental technique that enables researchers to understand and explain biology. Some of these visualizations have become iconic, for instance: tree views for taxonomy, cartoon rendering of 3D protein structures, or tracks to represent features in a gene or protein, for instance in a genome browser. Nightingale provides visualizations in the context of proteins and protein features. **Results:** Nightingale is a library of re-usable data visualization web components that are currently used by UniProt and InterPro, among other projects. The components can be used to display protein sequence features, variants, interaction data, 3D structure, etc. These components are flexible, allowing users to easily view multiple data sources within the same context, as well as compose these components to create a customized view. **Availability:** Nightingale examples and documentation are freely available at https://ebi-webcomponents.github.io/nightingale/. It is distributed under the MIT license, and its source code can be found at https://github.com/ebi-webcomponents/nightingale. **Contact:** gsalazar@ebi.ac.uk

Keywords: visualization; bioinformatics; proteins; software; web services

Introduction

The advances in the understanding of life sciences have often been accompanied by engaging visualizations: from Darwin, with his tree of life, to AlphaFold structural predictions with its striking coloring of confidence. Visualizations have been used for teaching the fundamentals of biology (e.g in school textbooks), and they have also been used to explain the most cutting-edge developments, as you can appreciate by browsing any recent issue of a biology journal.

Visualization is an active field of research and techniques have improved over time, with new strategies on how to represent data in the best way possible for each different case. A driving factor in this evolution has been technological. Thanks to this, visualizations are now not limited to paper and can be found in all types of displays, from the small screens of a cell phone to fully immersive virtual reality. Besides the medium, the biggest improvement is interactivity. These days visualizations can allow the audience to engage with the data directly and control some elements: filtering data, focusing on particular areas, coloring or marking pieces of the visualization, etc. Another factor to consider is the ubiquity of the web, which makes it the perfect environment to host visualizations: the audience is now global, and there are technical standards for how to render graphics, and manipulate their parts; to create interactive, performant, and beautiful visualizations.

The purpose of Nightingale is to take advantage of these standards, to create a library of biological web components that can be reused by multiple teams and projects, streamlining some visual aspects and allowing the users to focus on their data and research questions.

Unsurprisingly, Nightingale is not the only project working in this scope. There is an increasing number of web visualizations available for bioinformatics projects looking to explore, understand and explain complex biological systems ([6]).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 May 2023 doi:10.20944/preprints202305.1797.v1

However, Nightingale is not intended as a solution for all biological visualization needs, but rather its main biological focus is on the visualization of proteins, their features, domains, and other regions of interest.

ProtVista ([7]) is Nightingale's predecessor. It was developed and maintained by UniProt ([5]) between 2013 and 2018. ProtVista was a visualization tool for the graphical representation of protein sequence features in the UniProt Knowledgebase, and it was shared with the community as a BioJS component. BIoJS is a project that takes bioinformatics visualization components published in npm and presents them in a searchable interface, together with their documentation and examples ([1]).

In 2018, during the development of a new version of the InterPro website ([2]), it became clear that InterPro required a tool similar to ProtVista, however, it was also evident that it was going to require a lot of work to adapt it to the particular needs of the InterPro project. We then started a collaboration between the two teams, to modularize ProtVista and use web standards to create a set of reusable components, able to answer the needs of both projects, but purposely generic enough for other projects to adopt or extend. The outcome of this collaboration became what is currently referred to as Nightingale components.

Both UniProt and InterPro use Nightingale components in their websites, and other projects have adopted and extended them. Additionally, the scope of the components has grown, and now there are Nightingale components to visualize heatmaps, taxonomy, multiple sequence alignments, and more.

The main difference between ProtVista and Nightingale is the modularity of the latter. While ProtVista was a single visualisation unit, the components in Nightingale can be added or removed in order to customize the protein viewer for the particular needs of the use case.

Methods

Nightingale components are modular, so the user can display a single track, or combine multiple components to create their own personalized visualization. All components are published in npm (https://www.npmjs.com/search?q=keywords%3Anightingale%20webcomponents).

Nightingale follows web standards, and all its components are developed as custom elements following the HTML specification authored by W3C and WHATWG (https://html.spec.whatwg.org/).

Nightingale is open source, and our code resides in GitHub at https://github.com/ebi-webcomponents/nightingale, and although the main developers are from UniProt and InterPro, we have received contributions from software developers from several institutions.

Nightingale is a monorepo: a single repository containing all the components as packages. In this way the common tasks, such as bundling, deployment, and testing can be centralized, while the details of each component are isolated per package. It uses Lerna (https://lerna.js.org/) to coordinate this approach.

Nightingale is well documented and provides code examples, the reference for which can be found in the `README.md` file that is part of each package. This makes it available from GitHub, but also it is used to generate the documentation area in Nightingale's website.

Nightingale version 4.0.0 was released in March 2023. It is a complete re-factor of the common functionality. The new version has been rewritten in TypeScript (https://www.typescriptlang.org/), which helps with the early detection of bugs, and improves the developer experience when used in combination with a good integrated development environment (IDE). In this version, we used Lit (https://lit.dev/) as a minimalistic framework for the components. This reduces the need for some boilerplate code associated with web components and offers some tools to better organize the code. In particular, we use Lit's implementation of mixins to create small functionalities that each component can opt into.

Following modern practices of web development, Nightingale components are now deployed as es-modules, a feature that is currently supported on all major browsers, and that allows the creation of web applications that combine multiple components without the need for a bundler that pre-processes the files.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 May 2023 doi:10.20944/preprints202305.1797.v1

The new packages are published in npm using the namespace "@nightingale-elements": https://www.npmjs.com/org/nightingale-elements which makes it easier to identify them as an official nightingale element.

Results

Published components

Figure 1, on the left side, shows several Nightingale track components visualising UniProt data for the Amyloid-beta precursor protein (UniProt accession: P05067), and on its right, one of the structures associated with this protein (PDB accession: 1AAP) using the `nightingale-structure` component.

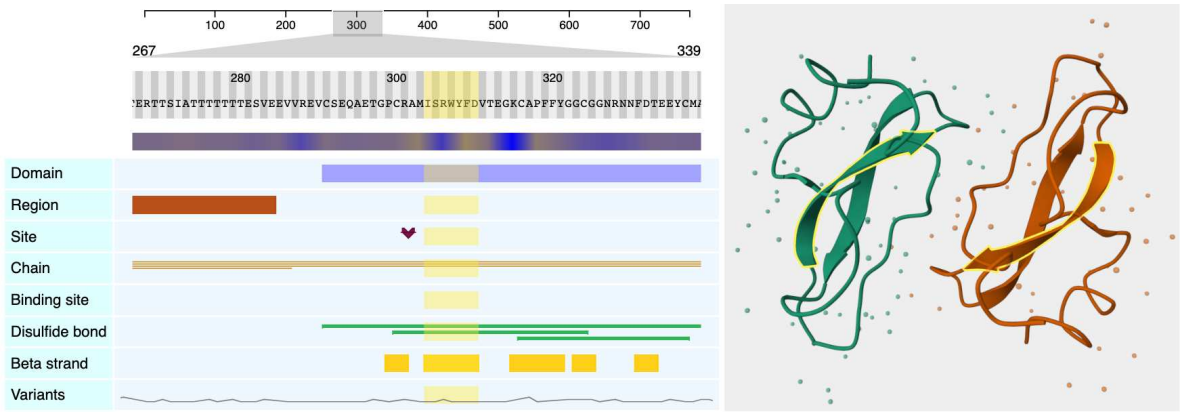

Figure 1. Snapshot of a composition of several nightingale components to display available information in UniProt for the Amyloid-beta precursor protein (UniProtKB: P05067, PDB: 1AAP).

The current release of Nightingale includes the following track components, all of which have common attributes to select a region on a protein of a given length:

`sequence`†: Displays a sequence of amino acids.
`colored-sequence`: Paints a color depending on each residue in the sequence.
`track`†: Renders protein features as simple shapes given their coordinates.
`interpro-track`: Extends `track` in order to represent the domain composition intrinsic to InterPro entries.
`linegraph-track`: Displays line graphs whose values correspond to positions in a protein sequence.
`links`: A node-link representation to visualize points of interaction between bases in a sequence.
`msa`: Multiple Sequence Alignment viewer, using the same coordinate system that other tracks.
`structure`: A wrapper over Mol* ([3]) to display protein structures. Although, it is not displayed as a track, it shares the same logic, in order to support interactivity with other tracks.

Nightingale also includes a set of utility components, supporting different functionalities in the protein feature viewer:

`manager`†: Container for all the tracks, and facilitates communication between them.
`navigation`†: Toolbar to zoom and navigate along the sequence.
`saver`: Takes snapshots of the current viewer as a PNG image.
`overlay`: Creates a layer over a given HTML element. Used to inform the user they need to press CTRL to zoom.

There are two components that don't fit in the context of the protein viewer: `sunburst` to display taxonomy data, and `heatmap` currently used to display confidence levels of structure models.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 May 2023 doi:10.20944/preprints202305.1797.v1

Additionally, we created a meta-component called `protvista-uniprot`. This is a single package that covers all the features integrated into the original ProtVista component, but in a more configurable way. It combines the essential components[2] and some of the other utilities. This has also served as a template for projects that want to include a customized version of ProtVista such as Pharos ([4]).

Components Adoption

Multiple projects are currently using Nightingale components and Table 1 lists some of these.

Table 1. Projects using Nightingale components.

Project	Example URL	Components Used
InterPro	https://www.ebi.ac.uk/interpro/protein/UniProt/P05067/	All components, except `structure`
UniProt	https://www.uniprot.org/uniprotkb/P05067/feature-viewer	All components, except `interpro-track` and `links`
PDBe	https://www.ebi.ac.uk/pdbe/entry/pdb/1aap/protein/1	Extends `protvista-uniprot` using Essential components† and develop its own components
Enzyme Portal	https://www.ebi.ac.uk/enzymeportal/search/P31153/enzyme	Uses `protvista-uniprot`
Pharos	https://pharos.ncats.nih.gov/targets/ULK4#sequence	Uses `protvista-uniprot` and develop its own components
GlyGen	https://glygen.org/protVista/P46782	Essential components†

Nightingale components have proven to be beneficial not only for the projects leading its development, but also for the community that shares similar visualization needs. It is important to highlight that members of this community have contributed back to the project with bug fixes, adding features, reporting errors, etc. Thanks to these contributions we have been able to improve the components over time, making them more generic and robust.

Funding: This work was supported by the National Human Genome Research Institute (NHGRI), Office of Director (OD/DPCPSI/ODSS), National Institute of Allergy and Infectious Diseases (NIAID), National Institute on Aging (NIA), National Institute of General Medical Sciences (NIGMS), National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK), National Eye Institute (NEI), National Cancer Institute (NCI), National Heart, Lung, and Blood Institute (NHLBI) of the National Institutes of Health under Award Number [U24HG007822], core EMBL funding and the Wellcome Trust [221320/Z/20/Z] (the content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health).

Acknowledgments: This work was partially done during the BioHackathon Europe 2022 organized by ELIXIR in November 2022. We thank the organizers and fellow participants. We also want to thank all the contributors in GitHub: your code, questions, and bug reports have helped us create this library.

Conflicts of Interest: A.B. is Editor-in-Chief of Bioinformatics Advances, but was not involved in the editorial process of this manuscript. The remaining authors have no conflicts of interest to declare.

References

1. Corpas, M., Jimenez, R., Carbon, S., Garcia, A., Garcia, L., Goldberg, T., Gomez, J., Kalderimis, A., Lewis, S., Mulvany, I., Pawlik, A., Rowland, F., Salazar, G., Schreiber, F., Sillitoe, I., Spooner, W., Thanki, A., Villaveces, J., Yachdav, G., and Hermjakob, H. (2014). Biojs: an open source standard for biological visualisation ? its status in 2014 [version 1; peer review: 2 approved]. *F1000Research*, 3(55).

[2] Indicates an essential component for the protein viewer.

2. Paysan-Lafosse, T., Blum, M., Chuguransky, S., Grego, T., Pinto, B. L., Salazar, G., Bileschi, M., Bork, P., Bridge, A., Colwell, L., Gough, J., Haft, D., Letunić, I., Marchler-Bauer, A., Mi, H., Natale, D., Orengo, C., Pandurangan, A., Rivoire, C., Sigrist, C. J. A., Sillitoe, I., Thanki, N., Thomas, P. D., Tosatto, S. C. E., Wu, C., and Bateman, A. (2022). InterPro in 2022. *Nucleic Acids Research*, 51(D1):D418–D427.
3. Sehnal, D., Bittrich, S., Deshpande, M., Svobodová, R., Berka, K., Bazgier, V., Velankar, S., Burley, S. K., Koča, J., and Rose, A. S. (2021). Mol* Viewer: modern web app for 3D visualization and analysis of large biomolecular structures. *Nucleic Acids Research*, 49(W1):W431–W437.
4. Sheils, T. K., Mathias, S. L., Kelleher, K. J., Siramshetty, V. B., Nguyen, D.-T., Bologa, C. G., Jensen, L. J., Vidović, D., Koleti, A., Schürer, S. C., Waller, A., Yang, J. J., Holmes, J., Bocci, G., Southall, N., Dharkar, P., Mathé, E., Simeonov, A., and Oprea, T. I. (2020). TCRD and Pharos 2021: mining the human proteome for disease biology. *Nucleic Acids Research*, 49(D1):D1334–D1346.
5. The UniProt Consortium (2022). UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research*, 51(D1):D523–D531.
6. Wang, R., Perez-Riverol, Y., Hermjakob, H., and Vizcaíno, J. A. (2015). Open source libraries and frameworks for biological data visualisation: A guide for developers. *PROTEOMICS*, 15(8):1356–1374.
7. Watkins, X., Garcia, L. J., Pundir, S., Martin, M. J., and Consortium, U. (2017). ProtVista: visualization of protein sequence annotations. *Bioinformatics*, 33(13):2040–2041.