WEB TOOL

REVISED *FeatureViewer*, a BioJS component for visualization of position-based annotations in protein sequences [v2; ref status: indexed, http://f1000r.es/38v]

Leyla Garcia[1], Guy Yachdav[2], Maria-Jesus Martin[1]

[1]European Bioinformatics Institute EMBL-EBI, Hinxton, Cambridge, CB10 1SD, UK
[2]TUM, Department of Informatics, Bioinformatics and Computational Biology-I12, 85748 Garching, Germany

v2 **First published:** 13 Feb 2014, **3**:47 (doi: 10.12688/f1000research.3-47.v1)
Latest published: 09 Apr 2014, **3**:47 (doi: 10.12688/f1000research.3-47.v2)

Abstract

Summary: FeatureViewer is a BioJS component that lays out, maps, orients, and renders position-based annotations for protein sequences. This component is highly flexible and customizable, allowing the presentation of annotations by rows, all centered, or distributed in non-overlapping tracks. It uses either lines or shapes for sites and rectangles for regions. The result is a powerful visualization tool that can be easily integrated into web applications as well as documents as it provides an export-to-image functionality.
Availability: https://github.com/biojs/biojs/blob/master/src/main/javascript/Biojs.FeatureViewer.js ; http://dx.doi.org/10.5281/zenodo.7719

Article Status Summary

Referee Responses

Referees	1	2
v1 published 13 Feb 2014	✓ report 1	✓ report 1
v2 published 09 Apr 2014 REVISED		

1 **Jim Procter**, University of Dundee UK

2 **Andreas Prlic**, University of California, San Diego USA

Latest Comments

No Comments Yet

Corresponding author: Leyla Garcia (ljgarcia@ebi.ac.uk)

How to cite this article: Garcia L, Yachdav G and Martin MJ (2014) ***FeatureViewer*, a BioJS component for visualization of position-based annotations in protein sequences [v2; ref status: indexed, http://f1000r.es/38v]** *F1000Research* 2014, **3**:47 (doi: 10.12688/f1000research.3-47.v2)

Grant information: GY was supported by the Alexander von Humboldt Foundation. UniProt EMBL-EBI is funded by National Institutes of Health (NIH) [1U41HG006104-03].
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: No competing interests were disclosed.

First published: 13 Feb 2014, **3**:47 (doi: 10.12688/f1000research.3-47.v1)
First indexed: 18 Mar 2014, **3**:47 (doi: 10.12688/f1000research.3-47.v1)

REVISED **Amendments from Version 1**

To the reviewers. We are grateful for your detailed reviews, they have been helpful in order to improve both our work and our manuscript. Based on the reviewers' comments, we have introduced some modifications to the original version. The summary of those modifications is included here: (i) we have simplified the abstract; (ii) we have included a new paragraph and figure in the Introduction section summarizing other approaches related to web-based visualization of protein sequence annotations; (iii) we have emphasized the novelty of this work at the end of the Introduction section; (iv) we have added a figure showing the relation across the main component and its extensions at the beginning of the Extensions section; and (v) we have corrected the typos and grammar errors from the first version.

See referee reports

Introduction

Position-based annotation is one of the cornerstones of bioinformatics. A great number of databases, analysis and prediction methods are geared towards providing data mapped to specific sequence coordinates. In the case of proteins, the Pfam[1] database identifies, marks-up, and characterizes different functional regions within a given protein. The coordinates of these domains are often given in terms of the start and end position within the protein. The largest pool of reviewed and automatically annotated proteins is provided by the UniProt Consortium[2]. It contains position-based annotations for structural regions, modified residues, and functional sites among others. Finally, protein feature prediction methods such as those integrated into PredictProtein[3] provide position-based annotations such as secondary structure states, buried and exposed residues, coiled-coil stretches, and disordered regions. PredictProtein also maps functional regions such as protein-protein binding sites and protein-DNA binding sites onto positions within the sequence.

Visualization of protein sequence features has already been used in different projects, some of which are shown in Figure 1. For intance, Pfam renders Pfam domains as well as some sites of interest, such as metals, active binding sites, and also disulphide bonds. It supports uncertainty for the start and end positions of the features by means of variations of rectangular-based shapes. Dasty[4] displays protein features from different sources as well as sequences and 3D structures, provindg an overview of the visualized protein. The Research Collaboratory for Structural Bioinformatics Protein Data Bank (RCSB-PDB)[5] mainly focuses on 3D structures for proteins but also includes feature visualization, showing the relationship between UniProt and PDB coordinates.

BioJS[6] is an open source JavaScript collection of components for visualization of biological data on the web. Here, we present *FeatureViewer* and its current extensions: *SimpleFeatureViewer* that simplifies the input data format, and *DasProteinFeatureViewer* that retrieves the input data from a web service. The *FeatureViewer* is a standard, portable BioJS component designed to easily render position-based annotations, a.k.a. features. The *FeatureViewer* component can be easily integrated into and controlled from other applications. As the *FeatureViewer* and its extensions have been developed within the BioJS framework, they result in a set of modular visual components displaying position-based annotations that can be integrated with other web applications in a standard manner. Modularity and easy integration differentiate these components from previous protein feature web based visualization efforts.

The *FeatureViewer* component

The *FeatureViewer* component extensively uses the Raphaël javascript library[7] that renders Scalable Vector Graphics (SVG) objects in modern web browsers. The use of SVG allows the graphics to scale to any requested resolution and is portable across different computing platforms and viewing software.

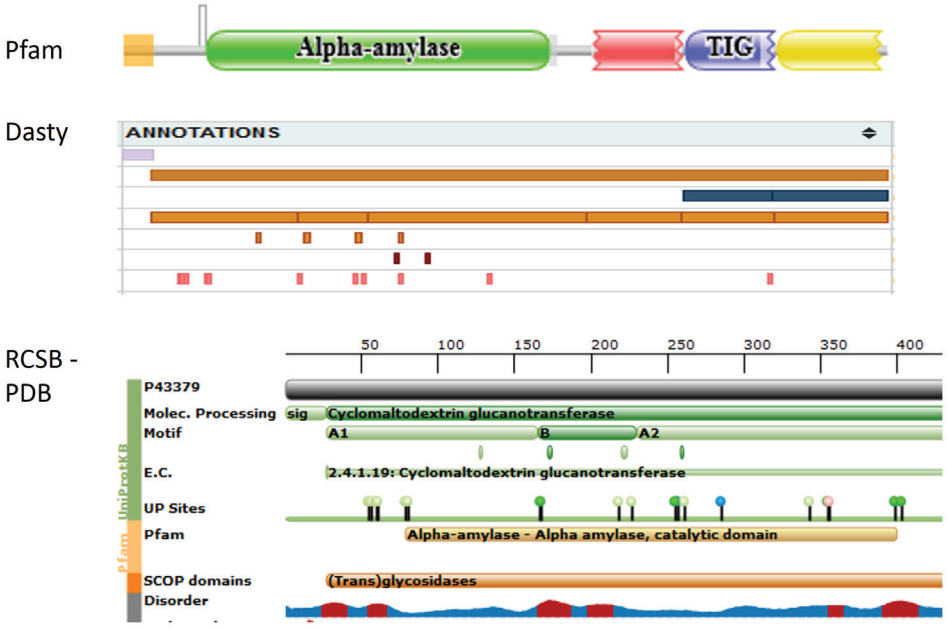

Figure 1. Visualization of a portion of the protein P43379–UniProt accession, in Pfam, Dasty and RCSB-PDB.

The *FeatureViewer* component can be easily integrated into any web application by including its dependencies in the head section, e.g., jQuery[8] and Raphaël, and then instantiating the component within a JavaScript section. A special dependency for some images is required as they are used for the pop-up dialogue controls. The code below shows how to instantiate the component to create the visualization shown in Figure 2. A complete example and more information can be found at http://www.ebi.ac.uk/Tools/biojs/registry/Biojs.FeatureViewer.html. The *FeatureViewer* component has been tested with modern browsers such as Mozilla, Chrome, and Internet Explorer (IE); however, the image export option is not available in IE.

```
var json = {
 featuresArray:[{
  // configuration for each style
  nonOverlappingStyle:{heightOrRadius:10, y:56}
  ,centeredStyle:{heightOrRadius:40, y:75}
  ,rowsStyle:{heightOrRadius:10, y:157}
  // feature information
  ,featureLabel:'Elicitor peptide 3'
  ,evidenceText:'UniProt'
  ,typeCode:'SO:0001064'
  ,typeCategory:'Molecule processing'
  ,featureId:'UPKB_Q8LAX3_PEPTIDE_74_96'
  ,featureTypeLabel:'active_peptide'
  // Display information
  ,type:'rect', fillOpacity:0.5, stroke:'#7DBAA4'
  ,height:10, r:10, y:56, x:529, cy:56, cx:529
  ,strokeWidth:1, fill:'#7DBAA4', width:151
 }]
 ,segment:Q8LAx3
 ,legend:{/*omitted ...*/}
 ,configuration:{
  // centered style
  sizeYCentered:160, sequenceLineYCentered:95,
  ,verticalGridLineLengthCentered:172
  ,horizontalGridNumLinesCentered:6
  // similar non-overlaping & rows (omitted)
  ,style:'nonOverlapping', nonOverlapping:true
  // ruler
  ,requestedStart:1, requestedStop:96
  ,rulerY:20, rulerLength:660, belowRuler:30
  ,pixelsDivision:50, aboveRuler:10
  // others
  ,sizeY:76, sizeX:700, rightMargin:20
  ,leftMargin:20, sequenceLineY:54
  ,sequenceLength:96, unitsize:6.875
  ,sizeYKey:210, verticalGridLineLength:66
  ,horizontalGridNumLines:2
  ,gridLineHeight:12
  ,verticalGrid:false, horizontalGrid:false
  ,verticalGridLineLengthRows:284
  ,horizontalGridNumLinesRows:8
  ,dasSources:'http://www.ebi.ac.uk/das
  -srv/uniprot/das/uniprot'
  ,dasReference:'http://www.ebi.ac.uk/das
  -srv/uniprot/das/uniprot'
  }
};
var myPainter = new Biojs.FeatureViewer ({
 target: "YourOwnDivId",
 json: json
});
```

Options and data

In order to instantiate the *FeatureViewer* component, some options should be defined. The mandatory options correspond to (i) a place holder named *target*, i.e., a DIV element in the web page where the annotations will be rendered, and (ii) a JSON object, named *json*, with the configuration, the protein identifier, the annotations, and the legend. *FeatureViewer* is a dummy component in the sense that it does not make any calculations about where to render the annotations, not even when the rendering style is changed; all the rendering information is provided in the *json* option. A comprehensive list of the elements in the *json* option is available at http://www.ebi.ac.uk/Tools/biojs/registry/Biojs.FeatureViewer.html. The *FeatureViewer* component includes three different layouts to display the features: all features centered, features grouped by type, and features organized in non-overlapping tracks, as shown in Figure 3.

User controls

Additional options can also be specified in order to customize the user controls as well as interaction with the features. User controls include the zooming slider and the export-to-image button, as shown in the Figure 3a. The zooming slider allows users to hone in on a region of interest and view it in greater detail, making it possible to move from an overview aspect into a detailed one without navigating to a different page. The export-to-image button allows users to export the rendered features into an image that can embedded into a paper or presentation.

Different kinds of interaction are also possible. Events bound to rendered annotations include a mouse-over action that highlights

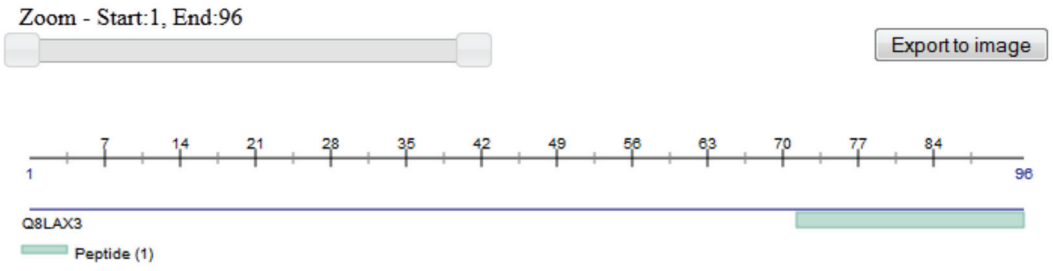

Figure 2. Visualization of a peptide using the *FeatureViewer* component.

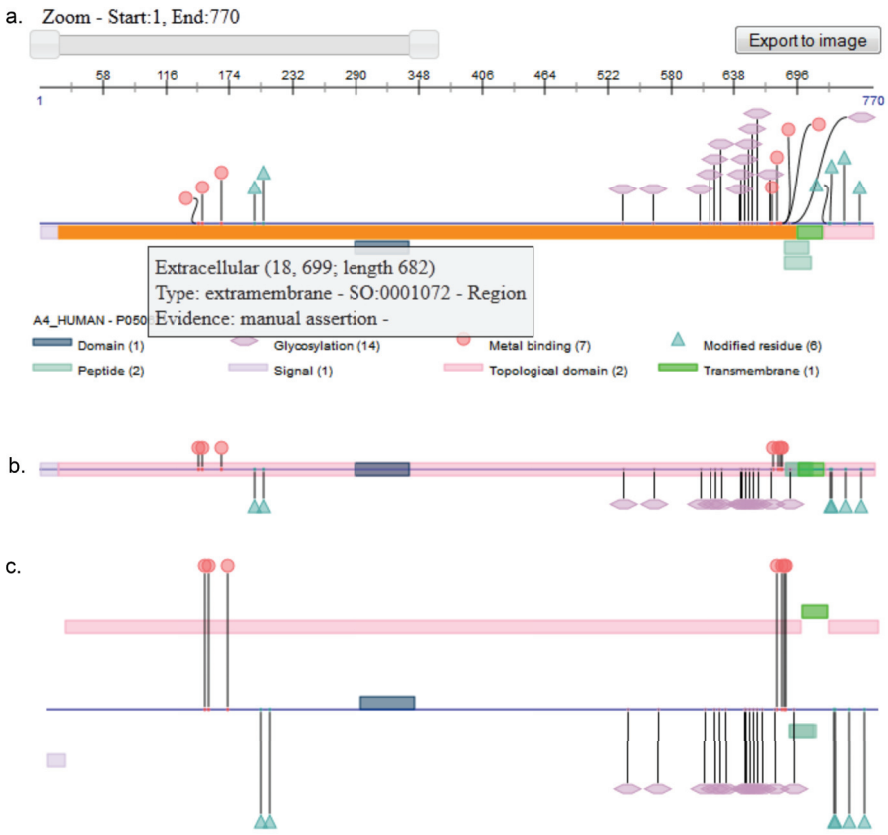

Figure 3. This visualization corresponds to the UniProt protein "Amyloid beta A4 protein" in the non-overlapping style; interactions such as shape dragging, tooltip, and selection as well as user controls such as zooming and image exporting are illustrated in this figure; **b**. shows the centralized view, while **c**. shows the by-rows view.

and colors the "focused" feature. Click action is also supported. Clicking on a feature selects it so it will remain highlighted until another feature is selected; clicking an already selected feature will deselect it. Tooltips tied to each shape pop up and reveal additional information about the rendered annotations. Either shapes or lines can be used to display features covering one single amino acid; currently metal bindings can be rendered as circles, active sites as diamonds, lipidation as waves, glycosylation as hexagons, and other post translational modifications, i.e., modified residues, as triangles. When shapes are used, it is possible to drag them making it easier to distinguish one from another when they are clustered.

Extensions

In order to make it easier for both developers and users to work wiht the *FeatureViewer*, two extensions are also provided. *SimpleFeatureViewer* simplifies the required features data while *DasProteinFeatureViewer* uses a web service to retrieve the features data. Figure 4 shows the three components in the Feature Viewer family.

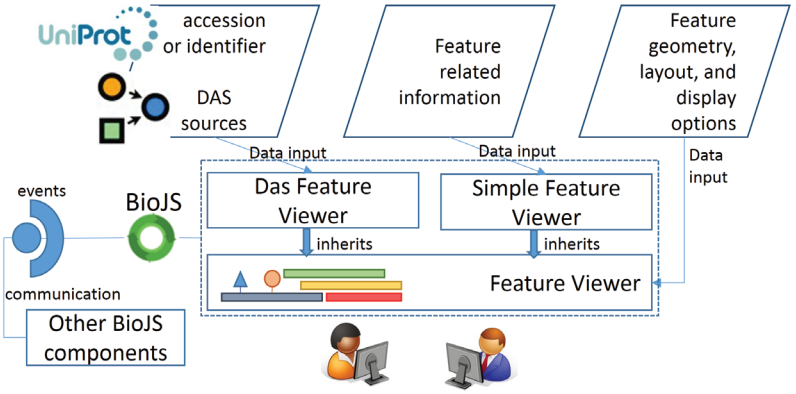

Figure 4. This graphic shows the *FeatureViewer* component and its extensions as well as the variations on the required input data. It also illustrates how BioJS handles the interaction with other components by means of events.

As *FeatureViewer* requires highly detailed information in order to display the features, a simpler version, the *SimpleFeatureViewer* component, builds on top of it. This simplified version takes care of calculating the configuration options as well as the localization of the features; thus, developers using this version can focus on the actual data rather than on intricate details regarding styles, pixels, and coordinates. However, only the non-overlapping tracks style is supported by this component. The main advantage of this component is that its data structure is much simpler than the one required for *FeatureViewer*, as observed in the following code excerpt.

```
var myFT = [
  {
   featureId:'UNIPROTKB_Q8LAX3_PEPTIDE_54_96',
   featureStart:54, featureEnd:96,
   typeLabel:'Peptide', typeCode:'SO:0001064',
   featureLabel:'Elicitor peptide 3',
   typeCategory:'Molecule processing',
   evidenceText:'UniProt', evidenceCode:' ',
   color:'blue'
  },
  {
   featureId:'UNIPROTKB_Q8LAX3_PEPTIDE_74_96',
   featureStart:74, featureEnd:96,
   typeLabel:'Active Site',
   typeCode:'SO:0001064'
   featureLabel:'Elicitor peptide 3',
   typeCategory:'Molecule processing',
   evidenceText:'UniProt', evidenceCode:' ',
   type:'diamond'
  },
  {
   featureId:'UPKB_Q8LAX3_DISULFID_75_96',
   featureStart:75, featureEnd:96,
   typeLabel:'Active Site',
   typeCode:'SO:0001064'
   featureLabel:'Elicitor peptide 3',
   typeCategory:'Molecule processing',
   evidenceText:'UniProt', evidenceCode:' ',
   color:'#33FF66',
   type:'bridge'
   }];
var myPainter =
   new Biojs.SimpleFeatureViewer ({
   target:'YourOwnDivId',
   sequenceId:'a4_human',
   sequenceLength:770,
   features:myFT
});
```

This component requires a place holder, a sequence identifier, a sequence length, and a features array; the width in pixels to be used to rendered the protein features can also be defined by using the option *imageWidth*. The features array contains information for each feature to be displayed including, for instance, identifier, start and end positions in the sequence, label, and color among others. More information is available at http://www.ebi.ac.uk/Tools/biojs/registry/Biojs.SimpleFeatureViewer.html.

A second extension, the *DasProteinFeatureViewer*, makes use of a web service that retrieves data from Distributed Annotation System (DAS) sources. DAS defines a communication protocol used to exchange annotations on gene or protein sequences[9]. Multiple protein databases provide their data following the DAS principles, for instance UniProt and InterPro[10]. For this extension, no information about the features themselves is required as such details will be retrieved from the web service, as shown in the code below.

```
var myPainter =
  new Biojs.DasProteinFeatureViewer({
   target: "YourOwnDivId",
   segment: "a4_human"
});
```

Additional options allow developers to specify the protein identifier, the DAS sources, the feature types – e.g., domain, chain, variant, etc., the rendering style, the image width, and some others. In order to avoid cross-domain problems, a proxy can also be specified. The feature types used by this component are those defined by UniProt, which is also used as the reference DAS source, i.e., the one providing the protein sequence. More information available at http://www.ebi.ac.uk/Tools/biojs/registry/Biojs.DasProteinFeatureViewer.html.

Use case

The PredictProtein service[3] integrates multiple algorithms that either retrieve from curated databases or automatically predict aspects of protein structure and function. Many of the predictions provided by the methods are mapped to positions within the protein. In order to easily highlight patterns, compare predictions, and cross-validate results, the PredictProtein interface lays out the predicted annotations in data tracks, i.e., in separate rows, each row presenting different predicted features. Data tracks are laid one under the other and enable the quick overview of some of the prominent features of the protein e.g., a cluster of binding sites close to the N-terminal or the count of trans-membrane regions. Figure 5 shows the implementation of the *FeatureViewer* component used for the PredictProtein service.

Conclusions

The *FeatureViewer* component and its extensions, *SimpleFeatureViewer* and *DasProteinFeatureViewer*, provide a platform to visualize position-based biological data easily and efficiently. *FeatureViewer*, like any other BioJS component, can be easily integrated with other web components or extended to have greater functionality than the one shown here. We expect this component to be particularly useful to developers and users alike, requiring little technical knowledge for its full functioning.

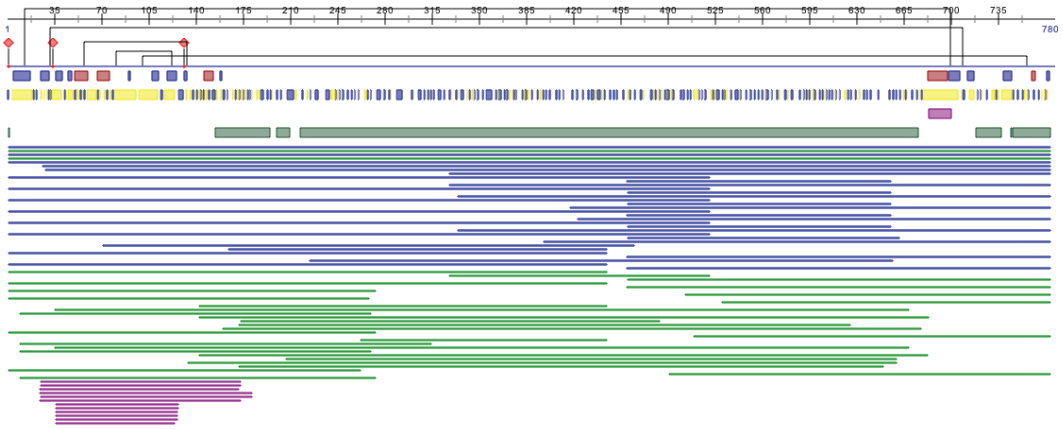

Figure 5. *FeatureViewer* is used by the PredictProtein service to show a stack of predicted structure and function features.

Software availability

Zenodo: BioJS Feature Viewer, doi: 10.5281/zenodo.7719[11]

GitHub: BioJS, https://github.com/biojs/biojs/

Author contributions

LG is the original author and current lead developer of the *FeatureViewer*, *SimpleFeatureViewer*, and *DasProteinFeatureViewer* components. GY extended *FeatureViewer* in order to support bridges, and provided the use case presented in this manuscript. MM conceived the original concept and contributed to the design of *FeatureViewer*. All authors have revised and agreed to the content in this manuscript.

Competing interests

No competing interests were disclosed.

Grant information

GY was supported by the Alexander von Humboldt Foundation. UniProt EMBL-EBI is funded by National Institutes of Health (NIH) [1U41HG006104-03].

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgements

GY thanks Burkhard Rost who helped fund this work and contributed ideas and feedback. GY also acknowledges Edda Kloppman and Tatyana Goldberg for helpful discussions and invaluable feedback and insight.

LG thanks Pablo Moreno who spotted a bug related to multiple instantiation and contributed to fixing it, as well as Mark Bingley, Claire O'Donovan, Sangya Pundir, and Xavier Watkins in the UniProt EMBL-EBI team and Rafael Jiménez and John Gómez in the IntAct EMBL-EBI team for their comments and suggestions about the *FeatureViewer* component and its extensions.

The authors thank all those who funded our research as well as researchers who deposited data into publicly available datasets and programmers who provided their work under a free license: our work stands upon their shoulders and would not have been possible without them.

References

1. Punta M, Coggill PC, Eberhardt RY, *et al.*: **The Pfam protein families database.** *Nucleic Acids Res.* 2012; **40**(Database issue): D290–301.
PubMed Abstract | Publisher Full Text | Free Full Text

2. UniProt Consortium. **Update on activities at the universal Protein Resource (uniprot) in 2013.** *Nucleic Acids Res.* 2013; **41**(Database issue): D43–7.
PubMed Abstract | Publisher Full Text | Free Full Text

3. Rost B, Yachdav G, Liu J: **The PredictProtein server.** *Nucleic Acids Res.* 2004; **32**(Web Server issue): W321–6.
PubMed Abstract | Publisher Full Text | Free Full Text

4. Villaveces JM, Jimenez RC, Garcia LJ, *et al.*: **Dasty3, a WEB framework for DAS.** *Bioinformatics.* 2011; **27**(18): 2616–2617.
PubMed Abstract | Publisher Full Text | Free Full Text

5. Rose PW, Bi C, Bluhm WF, *et al.*: **The RCSB Protein Data Bank: new resources for research and education.** *Nucleic Acids Res.* 2013; **41**(D1): D475–D482.
PubMed Abstract | Publisher Full Text | Free Full Text

6. Gómez J, García LJ, Salazar GA, *et al.*: **BioJs: an open source Javascript framework for biological data visualization.** *Bioinformatics.* 2013; **29**(8): 1103–4.
PubMed Abstract | Publisher Full Text | Free Full Text

7. **Raphaël—javascript library**. 2013.
Reference Source

8. **jquery—javascript library**. 2014.
Reference Source

9. Jenkinson AM, Albrecht M, Birney E, *et al.*: **Integrating biological data--the Distributed Annotation System.** *BMC Bioinformatics.* 2008; **9**(Suppl 8): S3.
PubMed Abstract | Publisher Full Text | Free Full Text

10. Hunter S, Jones P, Mitchell A, *et al.*: **InterPro in 2011: new developments in the family and domain prediction database.** *Nucleic Acids Res.* 2012; **40**(Database issue): D306–D312.
PubMed Abstract | Publisher Full Text | Free Full Text

11. Leyla G, Yachdav G, Moreno P: **Biojs feature viewer.** *Zenodo.* 2014.
Data Source

Current Referee Status:

Referee Responses for Version 1

Andreas Prlic
San Diego Supercomputer Center, University of California, San Diego, CA, USA

Approved: 18 March 2014

Referee Report: 18 March 2014
There are currently only a few approaches available for visualising protein features on the web. The *BioJS FeatureViewer* is a welcome addition to the set of tools available to web developers. The fact that it is available as a BioJS component should make it interesting as a re-usable web site element. However, I find that some of the already existing approaches provide a representation of features that I (subjectively) find graphically more refined and that go beyond simple boxes, circles, and triangles to represent sequence annotations.

Minor modifications:

I feel that the current version of this manuscript does not adequately summarise the already available approaches for visualising protein features. To list a few already available protein feature viewers:

- **Pfam** (is already being mentioned in the manuscript), shows the location of Pfam domains on protein sequences. It has currently one of the graphically most appealing views and can show more than one sequence per page (e.g. http://pfam.sanger.ac.uk/family/Piwi#tabview=tab1)

- **PDBsum** provides a graphical summary of secondary structure and pfam domains for protein sequences (e.g. https://www.ebi.ac.uk/pdbsum/1cdg and in large: https://www.ebi.ac.uk/thornton-srv/databases/cgi-bin/pdbsum/GetPage.pl?pdbcode=1cdg&template=wirp)

- **DASty** has been developed for several years at the EBI (in fact there seems to be some shared co-authorship) and provides a feature view that is based on DAS and can also map annotations to 3D (e.g. https://www.ebi.ac.uk/dasty/client/index.html?q=P05067)

- **RCSB Protein Feature View**: shows an SVG image for a full-length protein sequence from UniProt and how it corresponds to PDB entries. It loads annotations from external databases (e.g. http://www.rcsb.org/pdb/protein/P43379) and is available as open source from https://github.com/rcsb/proteinfeatureview

- **RCSB Sequence Page**: can show the relationship between UniProt and PDB coordinates on a per-residue level. Also can map annotations to 3D (e.g. http://www.rcsb.org/pdb/explore/remediatedSequence.do?structureId=1CDG).

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Competing Interests: I am the developer of the RCSB Protein Feature View.

1 Comment

Author Response

L. Garcia, EMBL-EBI, UK
Posted: 03 Apr 2014

Dear Andreas,

Thanks for your review, it has been useful to improve our work. Please see below our responses.

Response: We have included a new paragraph in the Introduction in order to cover other efforts on web visualizations for protein sequence annotations. We have included those related to protein sequence features in general rather than specialized on secondary or 3D structures.
Competing Interests: No competing interests were disclosed.

Jim Procter
Computational Biology, College of Life Sciences, University of Dundee, Dundee, UK

Approved: 25 February 2014

Referee Report: 25 February 2014
Summary
This *F1000Research* article describes an ensemble of JavaScript components designed for bioinformatics web developers that allow the retrieval, layout and display of positional annotation on a 1D coordinate system, such as a protein or nucleotide sequence. Special support is provided for the display of protein positional annotation retrieved from the Distributed Annotation System (requires some server-side configuration), and the system provides standard glyphs and shading styles to allow active sites and common types of post-translational modifications to be effectively displayed. Importantly, the components employ the BioJS framework, which allows them to exchange messages with other BioJS biological data visualization components (as well as any jquery based module) to facilitate the creation of rich, interactive web interfaces.

Using the component
The feature viewer component example page demonstrates that it provides a clean static visualization of the positional annotation as it appears on a coordinate system (complete with annotation legend). Online documentation supports the authors' statement that the plugin is highly configurable: attributes are provided to control practically every aspect of an annotation's appearance, and the surrounding coordinate space and additional decorators'. A few minutes spent interacting with the demonstration, however, shows that there is room for improvement:

(1) Suggested improvement. Once the zoom control is adjusted to focus in on a smaller region, it doesn't allow the user to sweep across the coordinate space (i.e. by click-dragging the visible range). From a developer perspective, I would also expect the component to raise BioJS events to inform other

components about the change in 'Region Of Interest' to allow them to respond in kind.

(2) Suggested improvement. The authors mention that feature glyphs can be dragged to facilitate inspection in crowded regions. It is unclear how useful this capability is – since once dragged, any change in view results in the feature glyph's shape being reset, and in crowded regions, the user must – by definition – move many features to examine the precise location of annotation. User modified glyph layout should – at the very least – be preserved on changes of scale. There may also be wisdom in including a force-directed layout algorithm to better optimize placement of nearby glyphs in response to manual adjustment of any particular one.

Developing with the component(s)

Apart from choosing appropriate values for the vast array of attributes and annotation display settings that *FeatureViewer* supports (more on that later), the most onerous aspect of deploying *FeatureViewer* is keeping track of the array of dependencies it requires. However, JavaScript module dependency management is a moving target, and I make the recommendation below with that in mind:

(3) Suggested improvement. I **strongly recommend** that the authors provide examples that employ a client-side JavaScript package management system such as 'RequireJS' (see http://jquerysbestfriends.com for some other tips about more sanitary methods of deploying jquery). Dependency management is important, and demonstrating good practice will allow these tools to be more widely adopted, and more effectively maintained.
One aspect of this paper that was not immediately clear on first reading was the functional relationship between *FeatureViewer* – which renders annotation layouts into SVG, and the helper components SimpleFeatureRenderer and DASFeatureRenderer.

(4) Suggested revision/reviewer contribution
A diagram such as the one I uploaded to FigShare here:
http://figshare.com/articles/The_BioJS_FeatureViewer_components_for_sequence_annotation_retrieval_

may help readers of this article more effectively grasp the different functionalities provided by the components in this system.

Specific revisions/corrections

(5) Required revision
In the abstract, the authors claim: "To our knowledge, this is the first client-side modular component to visualize position-based annotations that can be integrated into other web applications in a standard manner."

This is not quite correct. There have been other client-side modular web components that have been created for the visualisation of positional annotation, although some of the co-authors may have, for modesty reasons omitted mentioning the fact. In this case, Dasty3 and its forerunner, Dasty2 both provided a self-contained (ie modular) component that could be controlled through javascript and embedded in a more complex web page.

Here, I recommend the authors emphasise more strongly the key novelty in this work. For example: " **because these components are built with the BioJS framework**, they are the first modular visualization components for the display of position-based annotation that can be integrated with other web applications in a standard manner."

(6) Please revise the abstract for clarity

"This component is highly flexible and customizable, allowing the presentation of annotations by rows, all centered, or distributed in non-overlapping tracks. It uses either lines or shapes for sites and rectangles for regions."

Point 1. I'm not sure what presenting annotations "all centered" actually means!
Point 2. (optional). The authors might consider providing a biological example. This would help the lay-reader understand what the tool does without already having deep knowledge of protein sequence annotation visualization.

(7) Please revise the following sentence for clarity

The authors state that DasFeatureViewer can employ different glyphs for particular types of protein sequence annotation, including post translational modifications (PTMs).

Point 1. Suggested revision in ***bold italics***
“Either shapes or lines can be used to display features covering one single amino acid; currently metal bindings can be rendered as circles, active sites as diamonds, lipidation as waves, glycosylation as hexagons, and [***other***] post translational modifications as triangles.“

The reason for this revision is that Lipidation and glycosylation are, of course, both PTMs. The authors could also mention here that triangles are used as the default for other types of ‘MOD_RES’ type PTM (glycation, hydroxylation, phosphorylation, etc).

(8) Minor typos/grammar

Point 1. In the Introduction section "Particularly in the case of proteins, the Pfam database identifies, marks-up, and characterizes different functional regions within a given protein." THe word 'Particularly' is unnecessary.

Point 2. In the following sentence (in the Introduction section): "The largest pool of reviewed and automatically annotated proteins provided by the UniProt Consortium also contains position-based annotations for structural regions, modified residues, and functional sites among others." The authors probably mean to make the following statement (revision in ***bold italics***):

"The largest pool of reviewed and automatically annotated proteins ***is*** provided by the UniProt Consortium ***. It*** also contains position-based annotations .."

Point 3. Again, revising for clarity (in the Introduction section) (revision in ***bold italics***):

"Finally, ***protein feature prediction*** methods such as those integrated into PredictProtein provide ***position-based*** feature.."

Point 4. In the next paragraph: "and*DasProteinFeatureViewer* that retrieves the input data from a web service." There is a missing space between 'and' and 'DasProteinFeatureViewer'

Point 5. In the second paragraph of “The *FeatureViewer* component" section “The code below shows how

to instantiate the component; the corresponding visualization is shown in Figure 1." This sentence is perhaps more cleanly stated as:

"The code below shows how to instantiate the component to create the visualization shown in Figure 1".

(9) Typos in Web Resources
Please fix the missing '***e***' on this page:
http://www.ebi.ac.uk/Tools/biojs/registry/Biojs.DasProteinFeatureViewer.html

"This component uses a DASProtein web service that builds the JSON data object used by F***e*** atureViewer component. Version 1.0.0."

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Competing Interests: No competing interests were disclosed.

1 Comment

Author Response

L. Garcia, EMBL-EBI, UK
Posted: 03 Apr 2014

Dear James,

Thanks for your review, it has have been useful to improve our work. We have tried to addressed all you comments, however those related to the component itself, i.e., JavaScript code, will be taken into account for a new version of the software, and those related to BioJS in general will be sent to BioJS core developers. Please see our responses below.

Response to (1) and (2): We are currently working on an improved component to visualize protein sequence annotations. We have made notes about this suggestions and will take them into account for the new visualization. Unfortunately, such improvements are not yet ready to be integrated into the public BioJS GitHub repository.

Response to (3): We agree with the reviewer on the convenience of using dependency management for BioJS components. As we think it will impact not just this component but any other currently in BioJS, we have initiated a discussion on how to improve this aspect in BioJS, but no decision has been made yet. We will take it into account for the new visualization we are working on.

Response to (4): We have included a figure similar to the proposed one by the reviewer at the beginning of the 'Extensions' section. It introduces the extensions and indeed, as mentioned by the reviewer, helps the reader to understand the relation across the three components.

Response to (5): We have included a new paragraph in the Introduction in order to cover other efforts on web visualization for protein sequence annotations, covering not only Dasty but others as

well. We have also emphasized the novelty of this work as suggested by the reviewer.

Response to (6): We have simplified the abstract.

Response to (7 and 8): We have followed the reviewer's suggestions.

Response to (9): We have added the missing 'e' and it will be included in the next BioJS registry update.

Regards
Competing Interests: No competing interests were disclosed.