frontiers in Bioinformatics

TECHNOLOGY AND CODE
published: 11 February 2022
doi: 10.3389/fbinf.2022.774417

# Naview: A d3.js Based JavaScript Library for Drawing and Annotating Voltage-Gated Sodium Channels Membrane Diagrams

*Marcelo Querino Lima Afonso*[1*†‡], *Néli José da Fonseca Júnior*[2‡], *Thainá Godinho Miranda*[1] *and Lucas Bleicher*[1*]

[1]*Departamento de Bioquímica e Imunologia, Instituto de Ciências Biológicas, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil,* [2]*Cellular Structure and 3D Bioimaging, European Molecular Biology Laboratory, European Bioinformatics Institute, Hinxton, United Kingdom*

OPEN ACCESS

**Edited by:**
*Lydia Gregg,*
*Johns Hopkins University,*
*United States*

**Reviewed by:**
*Daniel Haehn,*
*University of Massachusetts Boston,*
*United States*
*Bjorn Sommer,*
*Royal College of Art, United Kingdom*

***Correspondence:**
*Marcelo Querino Lima Afonso*
*marceloqla@ufmg.br*
*Lucas Bleicher*
*lbleicher@ufmg.br*

**†Present address:**
*Marcelo Querino Lima Afonso, Departamento de Bioquímica e Imunologia, Instituto de Ciências Biológicas, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil*

‡*These authors have contributed equally to this work*

**Specialty section:**
*This article was submitted to Data Visualization, a section of the journal Frontiers in Bioinformatics*

**Received:** *11 September 2021*
**Accepted:** *05 January 2022*
**Published:** *11 February 2022*

**Citation:**
*Afonso MQL, da Fonseca Júnior NJ, Miranda TG and Bleicher L (2022) Naview: A d3.js Based JavaScript Library for Drawing and Annotating Voltage-Gated Sodium Channels Membrane Diagrams. Front. Bioinform. 2:774417. doi: 10.3389/fbinf.2022.774417*

Voltage-gated sodium channels (Nav) are membrane proteins essential to initiating and propagating action potential in neurons and other excitable cells. For a given organism there are often multiple, specialized sodium channels found in different tissues, whose mutations can cause deleterious effects observed in numerous diseases. Consequently, there is high medical and pharmacological interest in these proteins. Scientific literature often uses membrane diagrams to depict important patterns in these channels including the six transmembrane segments (S1–S6) present in four different homologous domains (D1–D4), the S4 voltage sensors, the pore-lining residue segments and the ion selectivity filter residues, glycosylation and phosphorylation residues, toxin binding sites and the inactivation loop, among others. Most of these diagrams are illustrated either digitally or by hand and programs specifically dedicated to the interactive and data-friendly generation of such visualizations are scarce or non-existing. This paper describes Naview, an open-source javascript visualization compatible with modern web browsers for the dynamic drawing and annotation of voltage-gated sodium channels membrane diagrams based on the D3.js library. By using a graphical user interface and combining user-defined annotations with optional UniProt code as inputs, Naview allows the creation and customization of membrane diagrams. In this interface, a user can also map and display important sodium channel properties, residues, regions and their relationships through symbols, colors, and edge connections. Such features can facilitate data exploration and provide fast, high-quality publication-ready graphics for this highly active area of research.

**Keywords: membrane plot, voltage gated sodium channel (NaV), d3.js, data visualization, javascript**

## INTRODUCTION

Voltage-gated sodium (Na+) channels are key signaling membrane proteins responsible for electrical excitability, also involved in biological processes in non-excitable cells, and of considerable physiological and pharmacological interest (Cardoso and Lewis, 2018). Voltage-gated Na + channels (Navs) can generate and propagate action potentials in excitable cells due to channel opening and fast inactivation mechanisms that regulate the permeation of Na + ions across the

membrane (Capes et al., 2012; Xia et al., 2013; Kubota et al., 2017). These channels are present in a large variety of organisms, the domain architecture of human Navs being observed in all animals. Their dysfunction is involved in severe diseases such as epileptic seizures, migraines cardiac arrhythmias, as well as pain-related neuropathies (Xia et al., 2013; Erickson et al., 2018). Sodium channels are involved in multiple physiological roles within a given organism, including the transmission of somatosensory signals, angiogenesis, muscle contraction, and immune cell maturation (Cardoso and Lewis, 2018). In addition, insect sodium channels are potential targets for both natural and synthetic insecticides and are therefore of agricultural interest (Zhang et al., 2016).

Each channel consists of an alpha subunit and auxiliary beta subunits that modify the properties of the first (Widmark et al., 2011). The alpha subunit is composed of a single chain of four sub-units in tandem (Domains I-IV), each formed by a structure of six transmembrane helices (6TM, H1-H6) that associate as tetramers to form a channel. Small extracellular and intracellular loops connect each helix, and the pore loops and large intracellular loops connect each domain (Yu and Catterall, 2003). In mammals, nine isoforms of these channels are found (Gene names SCN1A-SCN11A) possessing different functional roles, properties, and tissue-specific distributions among cells of the central and peripheral nervous systems (Chowdhury and Chanda, 2019). Post-translational modifications such as glycosylations and phosphorylations are part of the cellular modulation repertoire of these channels *in vivo*, being mostly found within the intracellular loop between the first and second domain of these channels (Scheuer, 2011; Laedermann et al., 2015; Cardoso and Lewis, 2018).

Graphical representations of the Nav alpha subunit transmembrane architecture are widely used in the scientific literature, with the earliest examples dated from the late 1980s—(Tanabe et al., 1988; Trimmer et al., 1989; Chiamvimonvat et al., 1996; Marban et al., 1998; Yu and Catterall, 2003; Yamaoka et al., 2006; Wood and Iseppon, 2018; Zybura et al., 2021). In these diagrams, membranes are disposed as either boxes or a bilayer lipid, transmembrane helices are shown as rectangles or cylinders, and loops as curved lines. Features commonly described by such plots include the voltage sensing helix S4, the fast inactivation motif IFM, glycosylation and phosphorylation sites, drug binding sites, important mutation sites, relevant sites for subunit interaction, and toxin binding sites. These features are usually displayed as either text or symbols inside the diagram.

Although sodium channel diagrams have been used for over 30 years, the availability of tools dedicated to an automated generation of such plots has been limited, but options for simpler diagrams with varying features are available. TOPO2 (Johns, 2010) reads an input indicating the number of segments in a protein chain, start/finish residues for transmembrane or partially inserted segments and residues to be colored and generates a simplified color diagram. Topology diagrams can also be drawn by using the output of a topology detection software such as HERA (Hutchinson and Thornton, 1990) and feeding it to topology drawing software such as TopDraw (Bond, 2003). This approach can also be used for globular proteins, but does not allow for individual residue/segment annotation, and includes no information about membrane insertion, being restricted to the secondary structure topology obtained from a PDB file. Membrane diagrams with individual annotations can be created using TMRPres2D (Spyropoulos et al., 2004) using user-provided info or importing information about transmembrane boundaries using public databases. The LaTeX based Protter web application (Omasits et al., 2014) and Textopo (Beitz, 2000) are capable of generating membrane protein diagrams in which each residue is displayed as geometric forms (often as circles). Whereas annotations can be easily included in both programs as symbols, text or specific colors, secondary structures cannot be easily distinguished in the diagrams of Protter and Textopo.

Various commercial and open source alternatives dedicated to drawing chemical compounds such as MarvinSketch, ChemDoodle, BKchem, XDrawChem, JChemPaint, ACD/ChemSketch, and MolView often have modules dedicated to the 3D visualization of proteins, but generating 2D diagrams (Krause et al., 2000; Todsen, 2014; Bergwerf, 2015). ChemDraw is one of the few alternatives including the possibility of drawing such diagrams in a highly dynamic and easy-to-use interface but lacking the possibility of direct inclusion of protein related data (Cousins, 2005).

Sodium channels membrane diagrams remain popular despite the increasing deposition of Nav structures in the last years, especially by cryogenic electron microscopy (Ahuja et al., 2015; Pan et al., 2018; Xu et al., 2019; Jiang et al., 2021), and the vast number of software dedicated to the 3D visualization of protein molecules such as PyMOL (Schrödinger, 2015), UCSF Chimera (Pettersen et al., 2004), VMD (Humphrey et al., 1996), Jmol (Jmol development team, 2016), and JavaScript based tools such as 3Dmol (Rego and Koes 2015), iCn3D (Wang et al., 2020), Litemol (Sehnal et al., 2017), NGL Viewer (Rose and Hildebrand 2015) and Mol* (Sehnal et al., 2021). Often used alongside figures rendered from 3D structures, the persistent usage of Nav diagrams could be attributed to their summarizing capacity. The alpha subunit of Navs often possess a length of more than 1,500 amino acids which can be challenging to depict when their complex topology is taken into account: four domains of six transmembrane helices and a reentrant loop, long and short interdomain loops disposed on either the intra or extracellular faces of the plasma membrane. Due to this the explicit representation of some features could require multiple 3D poses.

This publication describes Naview, an open-source d3.js based JavaScript library for drawing and annotating voltage-gated sodium channels membrane diagrams. Naview can highlight essential Nav features by using custom data provided by the user to modify the text, color, and connecting lines at specific helix/loop elements or residues.

# METHODS

## Implementation

Naview is implemented as an open-source d3.js based JavaScript web component, which can be used by importing its main CDN

**TABLE 1 |** Property table example. First column must be formatted with the "Resid" header followed by digits indicating each residue for a property to be mapped. The following property columns have header strings and are followed by float or integer numbers indicating the value of a property for each residue of a NaV.

| Resid | Property |
|---|---|
| 1 | 0.2871809547 |
| 2 | 0.9835970474 |
| 3 | 0.3891381106 |
| 4 | 0.2391246386 |

**TABLE 2 |** Relationship table example. Four columns are allowed with the following headers: "source", "target", "raw_weight", and "type". First and second columns indicating the interacting residues or elements. The "raw_weight" column contains an edge weight for color or width mapping. The last column "type" can be used to indicate edge types which can be weighted or colored separately.

| Source | Target | Raw_weight | Type |
|---|---|---|---|
| 776 | 660 | 0.6944505517 | Resids |
| 86 | 469 | 0.7383026986 | Resids |
| 1,308 | 318 | 0.4949883823 | Resids |
| 305 | 510 | 0.9651479396 | Resids |
| 1,621 | 123 | 0.3030461658 | Resids |
| DomainI; Helix4 | DomainII; Loop4 | 0.08937180957 | Elements |
| DomainII; Helix4 | DomainIV; Helix4 | 0.9300459795 | Elements |
| InterDomain5; Loop | InterDomain1; Loop | 0.1476849439 | Elements |

file (naview.js) into web pages. The complete documentation of each of the library's 107 functions and eight global variables can be found at: http://bioinfo.icb.ufmg.br/naview/public/docs/index.html. Naview is freely available under the Apache License 2.0. The complete source code and additional information related to library usage can be found at GitHub (https://github.com/marceloqla/NaView/). In addition to the web component, Naview can also be used as a web application (http://bioinfo.icb.ufmg.br/naview), developed in PHP, allowing direct access to any sodium channel available in the UniProtKb. Naview Style Editor is a graphical user interface that allows plot customization, the upload of residue mapped properties and residue/element interactions, and the download of the plot figures as Scalable Vector Graphics (SVG) or Portable Network Graphics (PNG). The styling information can also be exported as a text file that can be reused in new diagrams.

## Data Input and Processing

Two main inputs are generally supplied to Naview for generating a Nav alpha-subunit diagram (**Figure 1**):

1) A mandatory UniProt formatted text string (hereafter named *Raw Text*) containing the required data plotting a Nav alpha subunit. In the web application version, it is automatically fetched from the UniProtKb, requiring only the sequence identifier;
2) An optional JavaScript Object Notation (JSON) object, hereafter named *Style Object*, containing information related to the elements plot disposition such as their drawing types, widths, heights, scales, and colors (http://bioinfo.icb.ufmg.br/naview/public/docs/symbols/style_obj.html on the documentation for further information on the *Style Object*). When not supplied, a default representation of the *Style Object* is automatically applied. Any drawing options of the *Style Object* can be modified by Naview Style Editor (**Figure 2**).

Additionally, other inputs related to plotting text, color, and relationship annotations can be supplied. Each of them is described alongside their specific syntax in their dedicated sections.

The UniProt formatted *Raw Text* supplied by the user is then processed for the definition of drawing areas for three possible element types: membrane, helices, and loops which are further

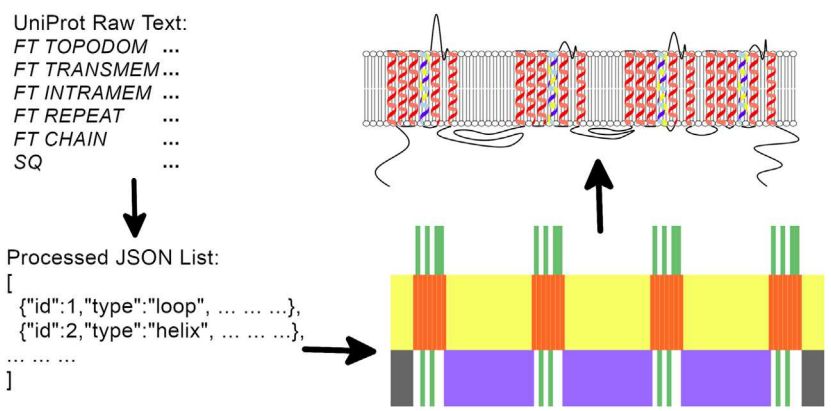


**FIGURE 1 |** Naview's general workflow: UniProt formatted data is initially checked and processed before initial draw areas are pre-calculated from the *Style Object* width, height and border definitions. These areas define the positioning of each drawn element in the following order: Membrane, Helices, Short Loops, Pore Loops, Long Loops, N and C terminal Loops. As each protein element is rendered, anchorage points are defined for proper loop positioning. Properties defined from the *Style Obj* or mapped from user selected settings define colors and drawing modes for each of the drawn features.

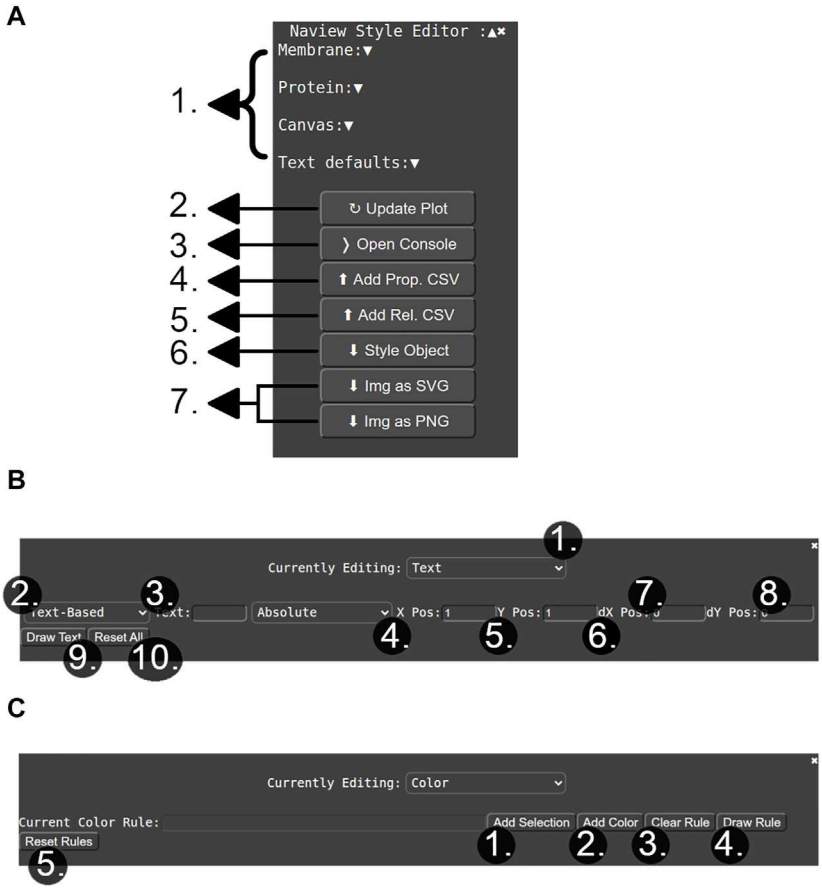


**FIGURE 2 |** Naview's style editor. **(A)** Main options of the styling menu including: 1) Dropdowns for options related to each of the drawn diagrams features such as colors, sizes and proportions. 2) Button for refreshing the currently drawn plot. 3) Button for opening the console that allows entering specific text annotation or color rules. 4) and 5) are buttons for adding property and relationship related data to the plot. 6) button for exporting a *Style Object* with the currently selected configurations. 7) Buttons for exporting the plot image in the SVG and PNG formats. **(B)** Console for adding a text annotation or color rule. 1) Dropdown for selecting between the text annotation or color rule modes. 2) Dropdown for selecting the input of free/property based text. 3) Input box for typing the desired text annotation. 4) Text positioning scheme: "absolute" defines text position by the given "x" and "y" 5) and 6) parameters; "relative" defines text position according to a selected element. "dx" and "dy" 7) and 8) shift the text to be drawn in the informed horizontal ("dx") and vertical direction ("dy"), being especially helpful in the "relative" positioning scheme. 9) Button for appending the currently defined text to the figure. 10) Removes all added text annotations. **(C)** Color rule addition console. 1) Opens a window for allowing specific residue/elements selections. 2) Opens a window for selecting a specific color/property-based color mapping. 3) Clears the currently selected color rule. 4) Updates plot with the currently selected color rule. 5) Removes all previously added color rules.

sub-divided as short loops, long loops, pore loops, N-terminal loop and C-terminal loop.

## Membrane, Helix and Loop Despictions

The Membrane element can be depicted as a "box" (SVG "rect" element) or as a lipid bilayer (multiple SVG "path" elements). The Style Object controls all specifications of coloring and drawing aspects of these two membrane representations, such as their opacity and relative sizes. Likewise, helices elements can be plotted according to three possible *Style Object* draw types: "box", "cylinder" and "cartoon".

Loops can be drawn by different curves whose rendering depends on their classification. Two aspects are considered for the rendering of these curves: their curve type function and their curve scaling method. Curve type functions describe the shape of a given loop by generating points to be interpolated by the d3.curveNatural function. Distances between these points have fixed or user-selected bounded proportions such that each curve type drawing aspect is scaled according to the Curve Scaling methods defined in the *Style Object*. Curve types common to the short and long loops include the "*Simple*", "*Bulb*" and "*Mushroom*" curves. The "*swirl*" curve type is specific to short Loops. Pore loops are generated by the "*pore*" curve type and N- and C- terminal by the "*N Curves*" curve type. The availability of multiple curve customization options allows users to customize plot aspects to their preferred style (**Figure 3**).

Design decisions for the representation of membranes, helices and loops attempted to cover most previously published Nav diagrams (Tanabe et al., 1988; Trimmer et al., 1989; Chiamvimonvat et al., 1996; Marban et al., 1998; Yu and Catterall, 2003; Yamaoka et al., 2006; Wood and Iseppon, 2018; Zybura et al., 2021). The usage of individual elements

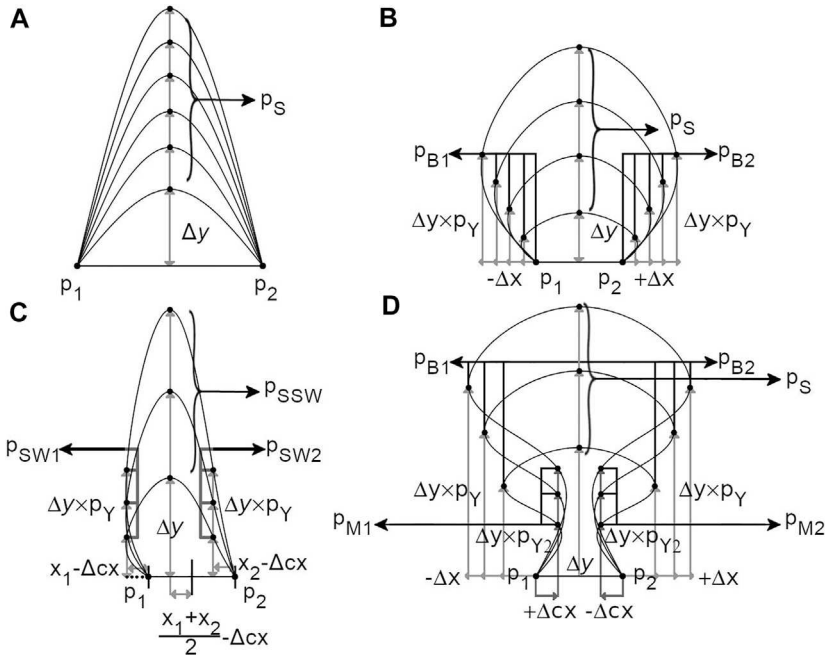


**FIGURE 3 |** Naview's curve drawing logic. **(A)** "*Simple*" curve function: a new point $p_S$ is generated in the center of two anchoring ponts drawing functions and scaled by a Δy parameter according to the selected loop length scales. **(B)** The "*Bulb*" curve function in which two new points are generated in relation to the "*Simple*" curve type: $p_{B1}$ and $p_{B1}$ whose vertical growth is controlled by $p_Y$, a proportion of the total Δy. The horizontal position of these points is given by the Δx parameter in the opposite direction of their closest anchoring points. **(C)** "*Swirl*" curve function is a variation of the "*Bulb*" curve type whose horizontal position is defined in a symetrical direction by a Δcx parameter, defined as a proportion of the distance of the anchoring points to their centroid. **(D)** The "*Mushroom*" curve type includes two new points in relation to the "*Bulb*" curve type: $p_{M1}$ and $p_{M2}$. The vertical position of these points is defined by the $p_{Y2}$ parameter as a proportion of the total Δy, and their horizontal position is defined from the anchoring points positions towards their centroid by the Δcx parameter.

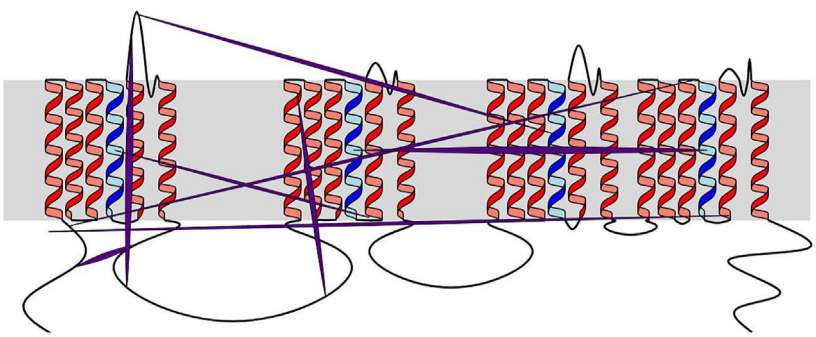

**FIGURE 4 |** Example of Naview's relationship drawing. Edges are colored in purple, with their central widths scaled according to the "*raw_weight*" column weights. This scaling allows the visual perception of stronger (larger width) and weaker (thinner width) relationships within the user inputted data. The membrane is shown as a grey box. All helices are shown as red cartoons except for the voltage-sensing helix 4, colored in blue.

inside a SVG document for each of the single Nav main secondary elements allowed the attribution of precise cartesian coordinates for each individual residue in this document. This enables the proper assignment of any text, color or edge annotations on the plot by the user.

Naview includes four scales to determine the loop length, depending on each loop type:

- "Fixed" in which a box of fixed height (and possibly width for "Bulb" and "Mushroom" curves) is set for determining the interpolating points of all loops of a given type (Short, Long, Pore or N/C terminus Loops).
- "Scaled" in which the height (and possibly width as above) of the boxes set for determining the interpolating points of all loops of a given type (Short, Long or Pore Loops) are set

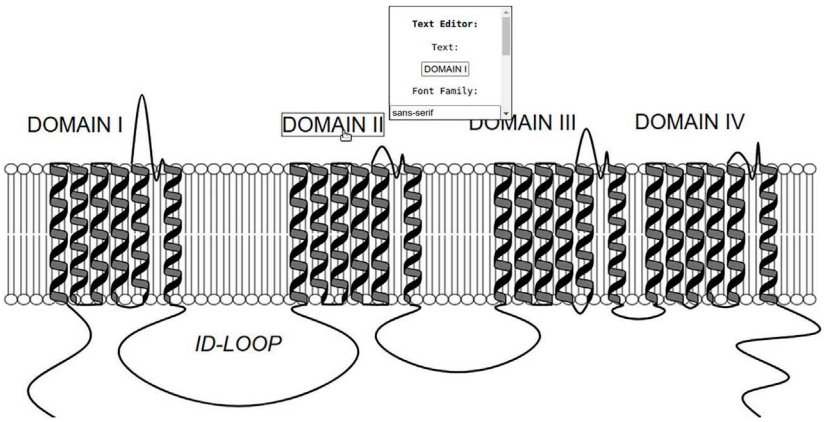


**FIGURE 5 |** Example of Naview's text annotations. All domain-indicating texts were added by using the Naview Style Editor console in the text edition mode. Text position can be adjusted by clicking and dragging any added text element. A single click highlights the selected text annotation and allows the editing of its current text and font characteristics. In this example such annotations were used to indicate specific domains (I-IV) and the first intracellular loop (ID-LOOP). Helices are shown as black cartoons and the membrane as a lipid bilayer. All loop residues are scaled to two pixels.

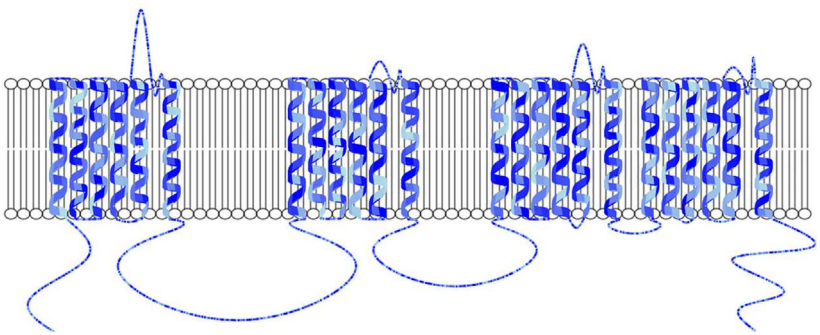

**FIGURE 6 |** Example of Naview's property-based color map from lightblue to blue after loading a CSV containing a randomly valued property named "Conservation" ranging from 0 to 1 for each of the protein's residues. Used color rule: *"ALL, by:Conservation,#ADD8E6;#0000FF, min;max"*. As such residues with a higher "Conservation" value are colored in a darker tone of blue. Helices are shown as cartoons and the membrane as a lipid bilayer. All loop residues are scaled to two pixels.

from a linear, power or logarithmic scale of their amino acid numbers up to a maximum box height (and possibly width).
- "Reslen" in which the height (and possibly width) of each box of a loop-type (Short, Long, Pore or N/C terminus Loops) is defined by a specific pixel value.
- "Custom" in which boxes of fixed specific height (and possibly width for "Bulb" and "Mushroom" curves) are set for determining the interpolating points of each loop of a given type (Short, Long Loops).

All helix and loop coloring, opacity, stroke and scaling settings are controlled by properties of the *Style Obj*.

## Input of Residue/Element Mapped Properties and Relationships

User-inputted residue properties and residue, helix and loop relationships can also be rendered as text annotations, specific coloring rules and edges between residues or elements (**Figures 4–6**). The possibility of including properties and relationships in the plot differentiate Naview from drawing-only methods, by allowing the ability of the direct inclusion and visualization of experimental data. Both types of data can be either preloaded alongside the *Raw Text* (Examples in **Tables 1** and **2**) or included by the Naview Style Editor.

Specific property values mapped for a set of residues can be loaded and used to generate color scales for differential residue coloring or element mapped text annotations. These properties should be loaded as a JSON object in which each Nav alpha subunit residue index (Example *1,2,3 . . . 2005* for a Nav containing 2005 residues) is used as a key for another dictionary, whose keys are strings describing a given property and whose values are those of the given properties for the selected residue (Example: *1:{"Conservation":0.1},2:{"Conservation":0.3},3:{"Conservation":0.5}* and henceforth).

Data representing relationships or interactions between different residues/elements present in the plot can be included

as a list of JSON inputs in the following format. Example: *{"source": 1,"target": DomainI;Helix6, raw_weight:0.5, "type": "Residue Importance"}*.

### Color Rules and Text Annotations

A list containing multiple color-filling text rules can be loaded as an input for generating a property-based residue color map. Accepted strings for color rules are any residue or element string keys followed by a comma-separated hex or string formatted color. Additionally, when properties have been mapped for a given Nav, they can be used for generating property-specific color maps.

Text annotations can be added as a list of JSON objects containing information about where a specific text should be drawn. This information can either be coded as absolute horizontal and vertical coordinates or as relative coordinates according to the positioning of a given residue or helix/loop element.

Alternatively, both color rules and text annotations can be added by the Naview Style Editor graphical interface (**Figures 2**, **5**, **6**).

## RESULTS AND DISCUSSION

The existence of a diagram for displaying the alpha-subunit architecture of Nav for over 30 years highlights their usefulness in depicting important properties of these proteins. The Naview d3.js based JavaScript library described in this publication is the first automated method focused on generating these diagrams. Examples and the full documentation for this library can be found at: http://bioinfo.icb.ufmg.br/naview/use and http://bioinfo.icb.ufmg.br/naview/public/docs/index.html.

The construction of transparent, information-rich and thought-provoking visual narratives is an intrinsic challenge in bioinformatics data visualization which requires the management of different graphical elements for efficient communication (Tao et al., 2004; O'Donoghue, 2021). This challenge is addressed by Naview's through its high customization and data integrative potential and facilitated by the inclusion of a dynamic graphical interface. Since Naview is formatted as a fully documented JavaScript library, its inclusion in web data resources focused on these channels can also be done simply and straightforwardly. By allowing the inclusion of residue mapped properties and relationships, Naview can be used for data exploration and integration purposes beyond the generation of publication-ready Nav figures.

In this publication, we demonstrate Naview and describe the logic of its implementation along with many of its features for plotting text, interactions and color mapped properties of sodium channels. Future updates should be focused on expanding the text annotation syntax to include drawing of polygons, arrows, backgrounds and other symbols, as well as reconfiguring the JavaScript library for drawing schemes and displaying data for any transmembrane/membrane-anchored protein.

## DATA AVAILABILITY STATEMENT

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found in the article/Supplementary Material.

## AUTHOR CONTRIBUTIONS

MA implemented most of the JavaScript code in the library. NdF tested, hosted the online version and wrote some additional JavaScript code and all PHP code. TM and LB tested the code and all authors contributed to the writing of the manuscript and gave ideas regarding the implemented features.

## FUNDING

This work was supported by CNPq (Grant 457851/2014-7) and CAPES (Grant 051/2013 and MA scholarship). LB is a fellow researcher of CNPq.

## ACKNOWLEDGMENTS

The authors would like to thank Lucas Carrijo de Oliveira for his helpful advice and discussions during writing.

## REFERENCES

Ahuja, S., Mukund, S., Deng, L., Khakh, K., Chang, E., Ho, H., et al. (2015). Structural Basis of Nav1.7 Inhibition by an Isoform-Selective Small-Molecule Antagonist. *Science* 350, aac5464. doi:10.1126/science.aac5464

Beitz, E. (2000). T(E)Xtopo: Shaded Membrane Protein Topology Plots in LAT(E)X2epsilon. *Bioinformatics* 16, 1050–1051. doi:10.1093/bioinformatics/16.11.1050

Bergwerf, H. (2015). MolView : an Attempt to Get the Cloud into Chemistry Classrooms. *ACS CHED CCCE Newsl.* 2015, 1–9.

Bond, C. S. (2003). TopDraw: a Sketchpad for Protein Structure Topology Cartoons. *Bioinformatics* 19, 311–312. doi:10.1093/bioinformatics/19.2.311

Capes, D. L., Arcisio-Miranda, M., Jarecki, B. W., French, R. J., and Chanda, B. (2012). Gating Transitions in the Selectivity Filter Region of a Sodium Channel Are Coupled to the Domain IV Voltage Sensor. *Proc. Natl. Acad. Sci. U S A.* 109, 2648–2653. doi:10.1073/pnas.1115575109

Cardoso, F. C., and Lewis, R. J. (2018). Sodium Channels and Pain: from Toxins to Therapies. *Br. J. Pharmacol.* 175, 2138–2157. doi:10.1111/bph.13962

Chiamvimonvat, N., Pérez-García, M. T., Ranjan, R., Marban, E., and Tomaselli, G. F. (1996). Depth Asymmetries of the Pore-Lining Segments of the Na+ Channel Revealed by Cysteine Mutagenesis. *Neuron* 16, 1037–1047. doi:10.1016/S0896-6273(00)80127-0

Chowdhury, S., and Chanda, B. (2019). Sodium Channels Caught in the Act. *Science* 363, 1278–1279. doi:10.1126/science.aaw8645

Cousins, K. R. (2005). ChemDraw Ultra 9.0. CambridgeSoft, 100 CambridgePark Drive, Cambridge, MA 02140. Www.cambridgesoft.Com. See Web Site for Pricing Options. *J. Am. Chem. Soc.* 127, 4115–4116. doi:10.1021/ja0410237

Erickson, A., Deiteren, A., Harrington, A. M., Garcia-Caraballo, S., Castro, J., Caldwell, A., et al. (2018). Voltage-gated Sodium Channels: (NaV )igating the Field to Determine Their Contribution to Visceral Nociception. *J. Physiol.* 596, 785–807. doi:10.1113/JP273461

Humphrey, W., Dalke, A., and Schulten, K. (1996). VMD: Visual Molecular Dynamics. *J. Mol. Graph.* 14, 33–38. doi:10.1016/0263-7855(96)00018-5

Hutchinson, E. G., and Thornton, J. M. (1990). HERA--a Program to Draw Schematic Diagrams of Protein Secondary Structures. *Proteins* 8, 203–212. doi:10.1002/prot.340080303

Jiang, D., Tonggu, L., Gamal El-Din, T. M., Banh, R., Pomès, R., Zheng, N., et al. (2021). Structural Basis for Voltage-Sensor Trapping of the Cardiac Sodium Channel by a Deathstalker Scorpion Toxin. *Nat. Commun.* 12, 128. doi:10.1038/s41467-020-20078-3

Jmol development team (2016). *Jmol*.

Johns, S. J. (2010). TOPO2, Transmembrane Protein Display Software. Available at: http://www.sacs.ucsf.edu/TOPO2/.

Krause, S., Willighagen, E., and Steinbeck, C. (2000). JChemPaint - Using the Collaborative Forces of the Internet to Develop a Free Editor for 2D Chemical Structures. *Molecules* 5, 93–98. doi:10.3390/50100093

Kubota, T., Durek, T., Dang, B., Finol-Urdaneta, R. K., Craik, D. J., Kent, S. B., et al. (2017). Mapping of Voltage Sensor Positions in Resting and Inactivated Mammalian Sodium Channels by LRET. *Proc. Natl. Acad. Sci. U S A.* 114, E1857–E1865. doi:10.1073/pnas.1700453114

Laedermann, C. J., Abriel, H., and Decosterd, I. (2015). Post-translational Modifications of Voltage-Gated Sodium Channels in Chronic Pain Syndromes. *Front. Pharmacol.* 6, 263. doi:10.3389/fphar.2015.00263

Marban, E., Yamagishi, T., and Tomaselli, G. F. (1998). Structure and Function of Voltage-Gated Sodium Channels. *J. Physiol.* 508 (Pt 3), 647–657. doi:10.1111/j.1469-7793.1998.647bp.x

O'Donoghue, S. I. (2021). Grand Challenges in Bioinformatics Data Visualization. *Front. Bioinform.* 1, 669186. doi:10.3389/fbinf.2021.669186

Omasits, U., Ahrens, C. H., Müller, S., and Wollscheid, B. (2014). Protter: Interactive Protein Feature Visualization and Integration with Experimental Proteomic Data. *Bioinformatics* 30, 884–886. doi:10.1093/bioinformatics/btt607

Pan, X., Li, Z., Zhou, Q., Shen, H., Wu, K., Huang, X., et al. (2018). Structure of the Human Voltage-Gated Sodium Channel Nav1.4 in Complex with β1. *Science* 362, 362. doi:10.1126/science.aau2486

Pettersen, E. F., Goddard, T. D., Huang, C. C., Couch, G. S., Greenblatt, D. M., Meng, E. C., et al. (2004). UCSF Chimera--A Visualization System for Exploratory Research and Analysis. *J. Comput. Chem.* 25, 1605–1612. doi:10.1002/jcc.20084

Rego, N., and Koes, D. (2015). 3Dmol.js: Molecular Visualization with WebGL. *Bioinformatics* 31, 1322–1324. doi:10.1093/bioinformatics/btu829

Rose, A. S., and Hildebrand, P. W. (2015). NGL Viewer: a Web Application for Molecular Visualization. *Nucleic Acids Res.* 43, W576–W579. doi:10.1093/nar/gkv402

Scheuer, T. (2011). Regulation of Sodium Channel Activity by Phosphorylation. *Semin. Cel Dev. Biol.* 22, 160–165. doi:10.1016/j.semcdb.2010.10.002

Schrödinger, L. L. C. (2015). *The PyMOL Molecular Graphics System, Version~1.8*.

Sehnal, D., Bittrich, S., Deshpande, M., Svobodová, R., Berka, K., Bazgier, V., et al. (2021). Mol* Viewer: Modern Web App for 3D Visualization and Analysis of Large Biomolecular Structures. *Nucleic Acids Res.* 49, W431–W437. doi:10.1093/nar/gkab314

Sehnal, D., Deshpande, M., Vařeková, R. S., Mir, S., Berka, K., Midlik, A., et al. (2017). LiteMol Suite: Interactive Web-Based Visualization of Large-Scale Macromolecular Structure Data. *Nat. Methods* 14, 1121–1122. doi:10.1038/nmeth.4499

Spyropoulos, I. C., Liakopoulos, T. D., Bagos, P. G., and Hamodrakas, S. J. (2004). TMRPres2D: High Quality Visual Representation of Transmembrane Protein Models. *Bioinformatics* 20, 3258–3260. doi:10.1093/bioinformatics/bth358

Tanabe, T., Takeshima, H., Mikami, A., Flockerzi, V., Takahashi, H., Kangawa, K., et al. (1988). Primary Structure of the Receptor for Calcium Channel Blockers from Skeletal Muscle. *Nature* 328, 313–318. doi:10.1038/328313a0

Tao, Y., Liu, Y., Friedman, C., and Lussier, Y. A. (2004). Information Visualization Techniques in Bioinformatics during the Postgenomic Era. *Drug Discov. Today BIOSILICO* 2, 237–245. doi:10.1016/S1741-8364(04)02423-0

Todsen, W. L. (2014). ChemDoodle 6.0. *J. Chem. Inf. Model.* 54, 2391–2393. doi:10.1021/ci500438j

Trimmer, J. S., Cooperman, S. S., Tomiko, S. A., Zhou, J. Y., Crean, S. M., Boyle, M. B., et al. (1989). Primary Structure and Functional Expression of a Mammalian Skeletal Muscle Sodium Channel. *Neuron* 3, 33–49. doi:10.1016/0896-6273(89)90113-X

Wang, J., Youkharibache, P., Zhang, D., Lanczycki, C. J., Geer, R. C., Madej, T., et al. (2020). iCn3D, a Web-Based 3D Viewer for Sharing 1D/2D/3D Representations of Biomolecular Structures. *Bioinformatics* 36, 131–135. doi:10.1093/bioinformatics/btz502

Widmark, J., Sundström, G., Ocampo Daza, D., and Larhammar, D. (2011). Differential Evolution of Voltage-Gated Sodium Channels in Tetrapods and Teleost Fishes. *Mol. Biol. Evol.* 28, 859–871. doi:10.1093/molbev/msq257

Wood, J. N., and Iseppon, F. (2018). Sodium Channels. *Brain Neurosci. Adv.* 2, 2398212818810684. doi:10.1177/2398212818810684

Xia, M., Liu, H., Li, Y., Yan, N., and Gong, H. (2013). The Mechanism of $Na^+/K^+$ Selectivity in Mammalian Voltage-Gated Sodium Channels Based on Molecular Dynamics Simulation. *Biophys. J.* 104, 2401–2409. doi:10.1016/j.bpj.2013.04.035

Xu, H., Li, T., Rohou, A., Arthur, C. P., Tzakoniati, F., Wong, E., et al. (2019). Structural Basis of Nav1.7 Inhibition by a Gating-Modifier Spider Toxin. *Cell* 176, 702–e14. doi:10.1016/j.cell.2018.12.018

Yamaoka, K., Vogel, S. M., and Seyama, I. (2006). Na+ Channel Pharmacology and Molecular Mechanisms of Gating. *Curr. Pharm. Des.* 12, 429–442. doi:10.2174/138161206775474468

Yu, F. H., and Catterall, W. A. (20032003). Overview of the Voltage-Gated Sodium Channel Family. *Genome Biol.* 4 (4), 207–7. doi:10.1186/GB-2003-4-3-207

Zhang, Y., Du, Y., Jiang, D., Behnke, C., Nomura, Y., Zhorov, B. S., et al. (2016). The Receptor Site and Mechanism of Action of Sodium Channel Blocker Insecticides. *J. Biol. Chem.* 291, 20113–20124. doi:10.1074/jbc.M116.742056

Zybura, A., Hudmon, A., and Cummins, T. R. (2021). Distinctive Properties and Powerful Neuromodulation of Nav1.6 Sodium Channels Regulates Neuronal Excitability. *Cells* 10, 1595. doi:10.3390/cells10071595

**Conflict of Interest:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

**Publisher's Note:** All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.