SOFTWARE

jvenn: an interactive Venn diagram viewer.

Philippe Bardou[??†], Jérôme Mariette[??*†], Frederic Escudie[??], Christophe Djemiel[??] and Christophe Klopp[??,??]

[*]Correspondence:
Jerome.Mariette@toulouse.inra.fr
[??]Plate-forme bio-informatique Genotoul / MIA-T, INRA, Borde Rouge, 31326 Castanet-Tolosan, France
Full list of author information is available at the end of the article
[†]Equal contributor

Abstract

Background: The amount of rich Web applications allowing scientists to store, share and analyze data on-line is increasing. This enhances the need of embeddable visualization tools. Scientists often produce lists of known identifiers corresponding to different experimental conditions. The Venn diagram is one of the most popular chart types used to present list comparison results.

Results: jvenn is a JavaScript library providing lists processing and Venn diagram displaying functions. It is able to handle up to six input lists presenting results as classical or Edwards-Venn diagrams. Using it, developers can easily embed dynamic Venn diagrams in Web pages. jvenn is fully configurable and allows to control and customize user interactions.

Conclusions: We introduce jvenn, an open source component for Web environments helping scientists to analyze their data. The library package, which comes with full documentation and an integration example, is freely available at http://bioinfo.genotoul.fr/jvenn.

Keywords: Venn; Edward-Venn; vizualisation; jquery; JavaScript

Background

List comparison results are often presented as Venn diagrams [**?**]. In a Venn diagram each list is figured by a transparent shape, differents shapes ovelap when elements are shared between the lists. The intersection counts are usually presented in the corresponding shapes overlap. In proportionnal Venn diagram the size of a shape depends on the number of elements in the corresponding list. Venn diagram with up to four lists are easy to read and understand, but are difficult to interpret with more lists. To overcome this issue, the Edwards-Venn [**?**] representation introduce new shapes that provides a clearer view for diagrams having more than four lists (Fig. 2).

Many Venn diagram software packages are already available. They can be classified using their type : stand-alone, library, Web applications, JavaScript library and their diagram layout: classical or Edwards. VENNTURE [**?**] is a stand-alone application able to generate Edwards-Venn outputs for up to six lists. VennDiagram [**?**] is an R package including functions to draw classical diagrams from two to five lists. Venn diagrams are nowadays also often included in Web pages. BioVenn [**?**] and venny [**?**] are Web applications with identifier input text areas. The first one offers a three circles area-proportional diagram, the second one outputs a non-proportional four lists diagram. Canvasxpress [**?**] and Google Chart API [**?**] are JavaScript libraries including Venn diagram features which can easily be embedded in any Web page.

They respectively produce diagrams with up to four and three lists. Both generate graphical outputs given the intersection counts but cannot calculate them.

Hereafter we introduce jvenn, a JavaScript library helping scientists to present their data, in the same spirit as already existing tools such as jbrowse [**?**], Cytoscape-Web [**?**], and jHeatmap [**?**]. The library has already been cited in two scientific publications [**?**, **?**] and is embedded within nG6 [**?**], RNAbrowse [**?**] and WallProtDB [**?**] Web applications.

Implementation

jvenn is based on venny regarding the algorithm computing the lists intersections. The algorithm has been wrapped in a jQuery plug-in [**?**], extended with new features presented hereunder and the ability to use up to six lists. It can be embedded in an HTML page referencing the JavaScript file. For researchers who want to produce a Venn diagram from their identifier lists, jvenn is also available as a Web application at http://bioinfo.genotoul.fr/jvenn/example.html. The installation documentation is included in the software package which can be downloaded from http://bioinfo.genotoul.fr/jvenn.

The library provides an option to define the input type : *series*. It accepts three different formats : lists, intersection counts and count lists. All are JSON objects. "Lists" contain for each input label and an identifiers table. "Intersection counts" contain a correspondence table between labels and letters [A..F] and a table linking the intersection names formed by the successions of letters and the counts. "Count lists" are organized as "Lists" in which identifiers are replaced by their unique occurrence and their count. Examples are presented in Table 1. Venn diagrams show intersections between a collection of sets. However, in some cases it can be interested to display not just the sets but also their counts. As example, an OTU represents a species or a group of species given by a cluster leader of DNA sequences. Displaying such data with the "lists" format leads to produce intersections of shared and unshared OTUs between samples. However, using "count lists" allows to define the amount of sequences constituting each OTUs. Thus, the produced diagram represents the species abondance between samples. For "lists" and "count lists" jvenn will first execute a function to compute the intersections and display the chart. For "intersection counts", the plug-in only displays graphic. The display is based on a JavaScript canvas object allowing to export it as a PNG file. The intersection table can also be downloaded as a CSV file. It contains a header line with the diagram area labels and, in column, the identifiers of the elements contained in the area. These features can be disabled by setting the *exporting* option to *false* hiding the exporting button.

The Web application developer can also define the diagram display setting the *displayType* option to *edwards* or *classic*. Setting the *shortNumber* option to *false* will disable the default behaviour of the plugin to substitute the intersection count by a question mark if this one overflow its area. The callback function defining the click on an intersection count can be overloaded by defining the *fnClickCallback* parameter. This function gives access to the *this.listnames* and *this.list* variables allowing the developer to control the user interactions. This feature can be disabled by setting the *disableClick* option to *true*. To customize the diagram display, the developer can also set the *colors* option.

Results and Discussion

Venn diagrams are commonly used to display list intersections because they are simple to read and understand. This is true up to four lists but scientists are interested in using it with more. This type of diagram is able to present up to six lists in its classical representation. Reaching this number, the diagram can not a priori be proportional to the list counts and the intersection areas are often too small to display the figures.

To present in a user-friendly manner five or six list diagrams, jvenn implements (Fig. 1) several functionalities. First, the display can be switched to Edwards-Venn (Fig. 2) what gives a clearer graphical representation for six list diagrams. To enhance the figure's readability on the classical six lists Venn graphic, it was decided not to present all the values and to link some areas to their figures using lines. This still did not permit to show all figures, therefore a switch button panel (Fig. 1) was added. This panel enables to switch on and off the different lists and display the corresponding intersection counts. For all diagrams, when the intersection count size exceeds the allowed space, the value is substituted by a question mark. The real value pops-up on mouse over. Last, to show the lists taking part in an intersection, jvenn highlights the corresponding areas on mouse over, fading the others out.

Scientists are usually interested in extracting identifier lists from some of the intersections, therefore, jvenn implements an one-click function which retrieves the names of the corresponding samples and the identifiers. Seeking an identifier can also be done using the plugin. The intersection with the matching identifier is highlighted as well as the lists containing it.

Having an overview of the list size and comparing multiple diagrams can be difficult using a Venn diagram. Thus, jvenn provides two extra charts (Fig. 1) bellow the Venn. The first one represents the input lists size histogram. It allows users to check the homogeneity of the lists size. The second one displays the number of elements located in intersections of a certain size. This feature can be used to compare the compactness of multiple Venn diagrams.

jvenn's performance depends on the client browser. Using the running version on a standard Linux computer (1 cpu, 4GB of RAM), it displays a six lists diagram of 10 000 identifiers in two seconds.

As examples, we produced two Venn diagrams representing six samples SRR068049, SRR06805, SRR068051, SRR068052, SRR068053 and SRR068054 corresponding to sets of Operational Taxonomic Units observed under different conditions. Fig. 1 shows intersections between all of them using the classical Venn diagram display. In Fig. 2, jvenn highlights the intersection between three samples out of six from an Edwards-Venn diagram.

Conclusions

jvenn is an easy-to-use library which generates Venn and Edwards-Venn diagrams from lists of identifiers or from computed intersection counts. Its implementation as a library allows whoever has some JavaScript programming skills to embed it in a Web page without any dependencies.

Availability and requirements

jvenn is freely available under the GNU General Public License (GPL) and can be downloaded with an example and the full documentation at http://bioinfo.genotoul.fr/jvenn website. A running version is accessible at http://bioinfo.genotoul.fr/jvenn/example.html.

Competing interests

The authors declare that they have no competing interests.

Author's contributions

JM conceived and designed the project. JM, PB, FE and CD implemented the project. CK evaluated software capabilities, and provided feedback on implementation. JM and CK wrote the manuscript. All authors read and approved the final manuscript.

Acknowledgements

We would like to acknowledge all our users for providing us useful feedback on the system and for pointing out features worth developing. We thank the reviewers for their insightful and constructive comments.

Figures

Figure 1 A six lists classic Venn diagram. This Venn diagram displays overlaps between six different biological samples. When the user clicks on a figure, it calls the developer defined function. The icon located on the top-right, allows users to download the diagram as a PNG file. On the bottom-right of the chart, a switch button panel allowing to activate or dis-activate lists to access a specific intersection count. The charts showing the lists size repartition and the number of common and specific elements are located underneath the diagram.

Figure 2 A six lists Edwards-Venn diagram. On mouse over a figure, the shape corresponding to the list involved in the intersection are highlighted and the other ones faded out. In this example, the user points the intersection between samples SRR068049, SRR068051 and SRR068052 which contains eight different items.

Tables

Table 1 Available formats and example for the *series* option.

format	example
lists	`series: [{ name: 'sample1', data: ["Otu1", "Otu2", "Otu3", "Otu4", "Otu5", "Otu6", "Otu7"] }, { name: 'sample2', data: ["Otu1", "Otu2", "Otu5", "Otu7", "Otu8", "Otu9"] }]`
intersection counts	`series: [{ name: {A: 'sample 1', B: 'sample 2', C: 'sample 3'}, data: {A: 340, B: 562, C: 620, AB: 639, AC: 456, BC: 915, ABC: 552} }]`
count lists	`series: [{ name: 'sample1', data: ["Otu1", "Otu2", "Otu3", "Otu4", "Otu5", "Otu6", "Otu7"], values: [5, 15, 250, 20, 23, 58, 89] }, { name: 'sample2', data: ["Otu1", "Otu2", "Otu5", "Otu7", "Otu8", "Otu9"], values: [90, 300, 10, 2, 45, 9] }]`