Visualization | Workflows | Cloud | e-Science

Integrating a new visualization tool in Galaxy

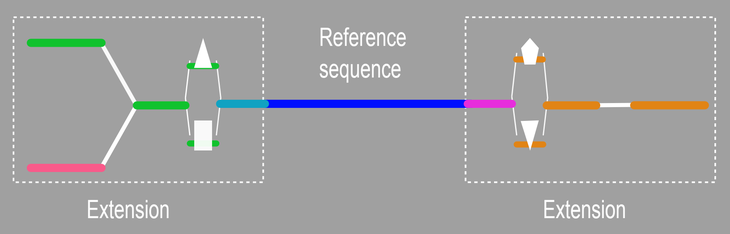
Alexan Andrieux[1], Pierre Peterlongo[1], Yvan Le Bras[2], Cyril Monjeaud[2], Charles Deltel[3]

[1] Genscale, INRIA, Campus de Beaulieu, 35042, Rennes Cedex, France
[2] GenOuest Core Facility, UMR6074 IRISA CNRS/INRIA/Université de Rennes 1, Campus de Beaulieu, 35042, Rennes Cedex, France
[3] SED, INRIA, Campus de Beaulieu, 35042, Rennes Cedex, France

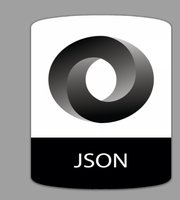
Galaxy with Mapsembler 2 tool integrated

Galaxy + Mapsembler 2

Mapsembler 2 is a target assembly tool that allows to extend reference sequences from each side with one or more sets of reads.

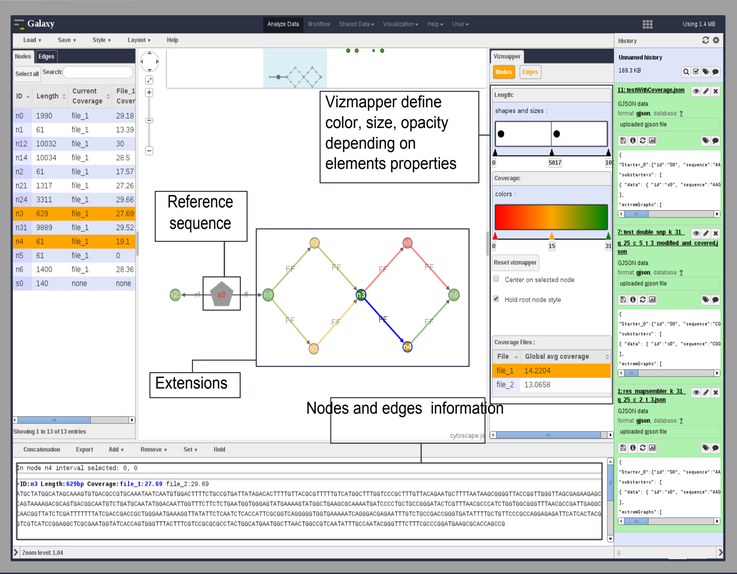


Several extensions are possible and Mapsembler 2 constructs a graph with all possible extensions.



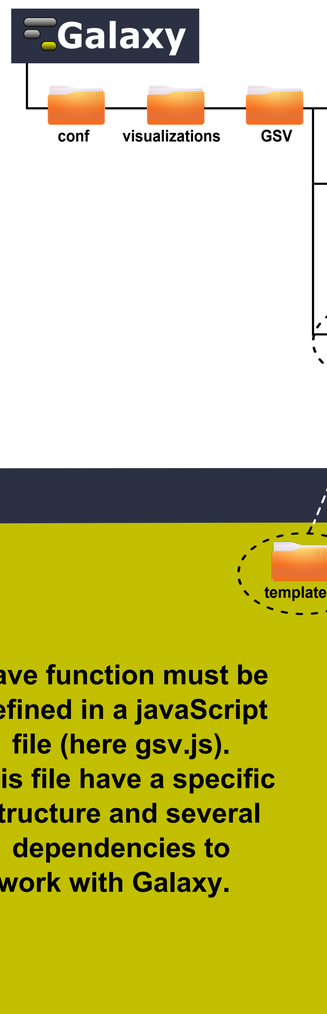
Output is a JSON (JavaScript Object Notation) file with a particular strucutre (gJSON).

Integrating the visualizer GSV in Galaxy



GSV (Graph Sequence Viewer) is a graph viewer in Html5/JavaScript/jQuery for the tool Mapsembler 2.

HTML5 JS CSS3 jQuery jQuery UI



The visualizer source files must be in a directory with a specific structure.

Initialize and restore session functions was defined into javaScript file of GSV.

Save function must be defined in a javaScript file (here gsv.js). This file have a specific structure and several dependencies to work with Galaxy.

GSV.xml

Configuration file for the visualizer.

Structure :

1) Define XML language definition.

2) Import visualization.dtd for doc type definition.

3) Define link to the visualization tool in the history.

4) Test the datatype of the data (here gjson).

5) If the test is passed, generate an id to recognize data in the database.

6) Send the data to the visualization tool with a defined variable.

7) Define the file use to start visualization tool (here GSV.mako).

GSV.mako

Template for python.
Use to generate html and javaScript codes dynamically.

GSV.mako replaces the index.html of a standard web page.

Structure:

1) Import Galaxy libraries needed.

2) Define python global variables for visualization behaviour.

3) Import Css files.

4) Import javaScript files.

5) Define galaxy_config with root path for save function.

6) Import the id given by Galaxy to the data of the output file or uploaded file.

7) Import the file name to have the good name of save in the saved visualization page.

8) Configure require library and use it to import GSV.js.

9) Write the function initGSV(data) that defines a new object (constructor in GSV.js) for implementing methods (here save methods).

10) Test if it's an output file/uploaded file or a restore session call.

11a) If the test is passed, get the data of the file.

12a) Call initGSV(data) to initialize methods and call the function to initialize the visualization tool.

11b) If the test isn't passed, import name and data of file use with this session.

12b) Call initGSV(data) to initialize methods and call the function to restore a saved session.

GSV.js

Implement save method using several javaScript libraries of Galaxy (visualization.js, mvc.js, backbone.js,...).

Structure:

1) Import Galaxy files dependencies.

2) Use the backbone library to initialize an object with data from a file and the methods for the visualization tool.

2.1) Define the name of the object.

2.2) Initialize the object with visualization library.

3) Use visualization library to save data and and define methods.

3.1) Define default attributes.

3.2) Define id and data associated with the methods.

3.3) Define new methods (here save methods).

4) Return the new object.

Funding

This work was funded by the ANR-12-BS02-0008 (Colib'read) and the INRIA ADT Mapsembler.

Download : https://colibread.inria.fr/mapsembler2/

informatiques mathématiques — Biogenouest — Colib'read — GENSCALE