Universidad Rey Juan Carlos

Carbonic User Manual

Cristian Rodríguez Bernal
cristian.rodriguez@urjc.es

Cristian Rodríguez Bernal
23/11/2021

http://vg-lab.es

Content

1. Introduction

Carbonic is a web framework for interactive visualization, analysis, creation and editing of general compound graphs (graphs composed by both a hierarchy and a network).

This framework provides an interactive representation based on a Sunburst chart to display the hierarchy. To work with this representation, Carbonic has different operations to handle complexity: zooming, node-collapsing and filtering. Besides, Carbonic allows editing the hierarchy with both “Bottom-Up” and “Top-Down” approaches allowing creating a new parent or new children to existing elements.

On the other hand, the network is shown using lines that connect the leaf nodes of the hierarchy, in such a way that when nodes are collapsed, the aggregated connections are depicted. Because of line crossing can produce visual cluttering, Hierarchical Edge Bundling has been used. This technique creates bundles with curved trajectories that follow the structure of the tree.

2. Hierarchy and connection visualization

Carbonic visualization is based on two representations known in the literature. The hierarchy is represented by the Sunburst technique. Sunburst diagrams represent containment relationships following a Space Filling approach with a radial layout. Although Sunburst diagrams don't make an optimal use of the screen space, specially in the corners, they provide an intuitive and effective view of a hierarchical structure and, consequently, of the abstraction levels encoded in the hierarchy. On the other hand, the network visualization is based on a node-link representation using a Hierarchical Edge Bundling technique to avoid visual cluttering.

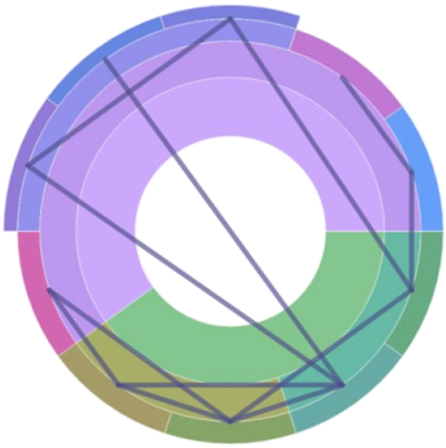

Figure 1: Carbonic representation: the radial subdivisions of the Sunburst indicate the different levels of the hierarchy, while the angular subdivisions in the lower radius indicate the leaf nodes.

3. Hierarchy and connection creation and edition

3.1. Hierarchy creation

As discussed above, Carbonic allows creating a hierarchy using both "*Bottom-Up*" and "*Top-Down*" approaches.

3.1.1. Top-down

Starting with an empty canvas, the user will see a large grey circle (root of the Sunburst), on which he has to right-click to open a context menu and select "*Add children nodes*" (*Figure 2*).

Figure 2: Adding new children nodes to root

This will open a modal window where different parameters can be defined:

- **Generator mode:** Used to select the creation mode of the new children for the selected node.
  - **Suffix Mode:** Generates a list of nodes based on [*NodeName*]_[*NodeNumber*] In *Figure 3.a* and *Figure 3.b*, we create five nodes: *Node_0*, *Node_1*, *Node_2*, *Node_3*, *Node_4* and *Node_5*.
  - **CSV Mode:** A list of names separated with ";". In *Figure 4.a* and *Figure 4.b* we create two nodes: *node1* and *node2*.
- **Node name:** Name of the node to create. If you use "*CSV Mode*", you can create a list of nodes names separated with ";".
- **Number of nodes:** Number of elements to create. Only used on "*Suffix mode*".

New children can be created repeating this action right-clicking on any other node of the hierarchy.

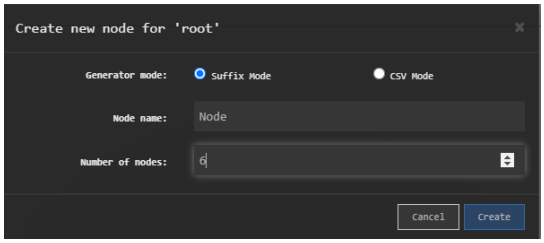

Figure 3.a: Form for creating children with suffix mode.

Figure 3.b: Six new children created with suffix mode.

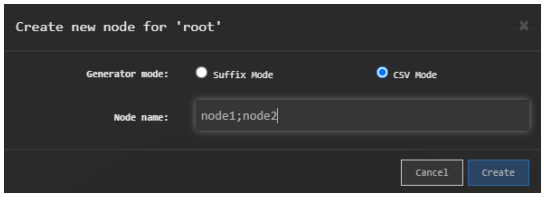

Figure 4.a: Form for creating children with CSV mode.

Figure 4.b: Two new children created with CSV mode.

This approach must begin with a hierarchy that contains at least one level (without counting the root), adding a new parent to an existing node.

3.1.2. Bottom-up

When right clicking on an existing node (*Figure 5.a*), a context menu is displayed, where "Add parent node" must be selected. This option will open a form to select the name of the new parent node (*Figure 5.b*).

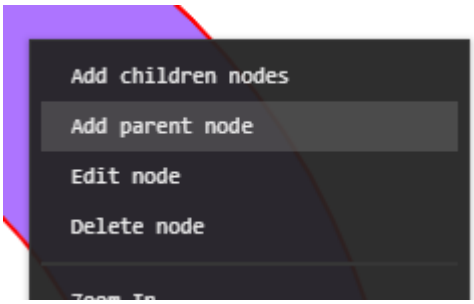

Figure 5.a: Adding a parent to selected node.

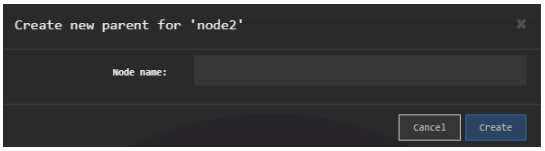

Figure 5.b: Form for creating new parent node to another node.

3.2. Node edition

To modify the properties of a node, right click on the node and select "*Edit Node*" (*Figure 6.a*). This will show a form (*Figure 6.b*) where the properties of the selected node can be edited. The properties of the nodes to be showed, as it has been previously explained, can be defined beforehand depending on the specific domain.

Figure 6.a: Edit node option.

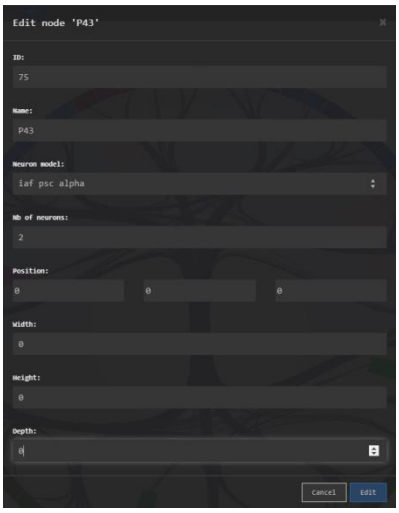

Figure 6.b: Form for edit a node.

3.3. Connection creation

Carbonic allows creating connections in a simple way. To create connections, press one of the Control keys of the keyboard and, while holding it, click with the mouse or trackpad on the source node and target nodes.

The current implementation assigns the first node marked as the source node while the rest of the elements will be the target nodes (*Figure 7*).

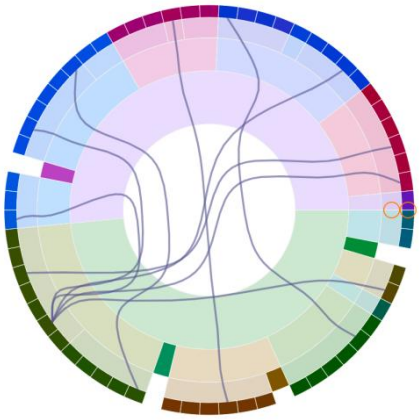

Figure 7: Carbonic view with hierarchy and connections.

Please note that all connections generated with this functionality contain properties with default values, being necessary to edit these connections one by one to set the desired values of the different properties

3.4. Connection edition

The way to edit a connection will be pressing the *Control* key and clicking the two nodes that are connected. If the connection exists, a form will be shown as the image below (*Figure 8*). If there is not any connection, it will create a new connection.

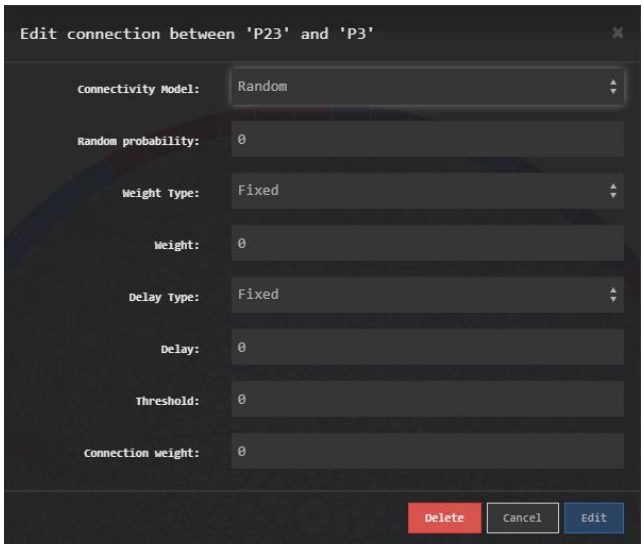

Figure 8: Form for edit connection between leaf node "P23" and leaf node "P3".

Please note that selecting more than two nodes when a connection exists will only affect to the first two nodes, and the rest of them will be ignored.

4. Hierarchy and network interaction

4.1. Hierarchy navigation

Carbonic provides zooming capabilities to allow the user to focus on a specific sub-tree of the hierarchy. In order to show how zooming works a complex scene containing four levels of hierarchy, will be used as an example.

To focus on a specific node right click on it and select "Zoom in". As it can be seen in *Figure 9*, the yellow node of the first level becomes the new root after zooming. This operation can be done with any node of the hierarchy except for the leaf nodes.

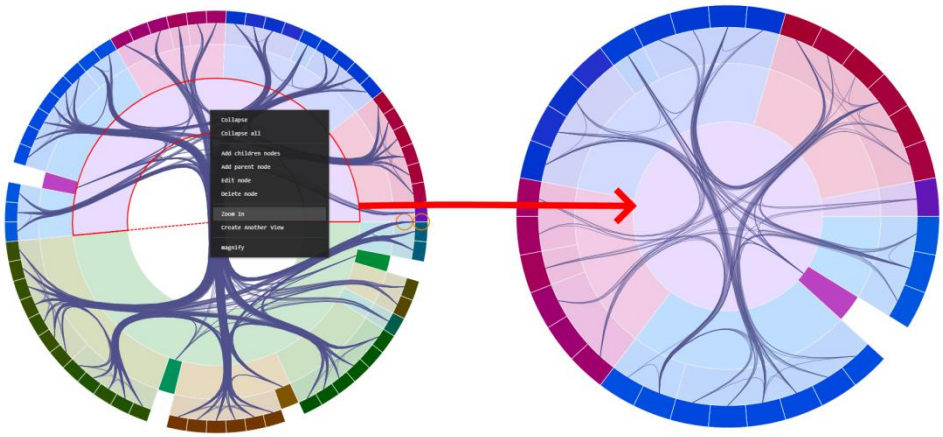

Figure 9: Zoom in example. The selected sector becomes the root of the sunburst diagram.

To finish zooming in change the root by the parent of the current root or return to the original root by right clicking in the new root node (*Figure 10*) and then selecting the "Zoom out" or "Zoom out to root" options, respectively.

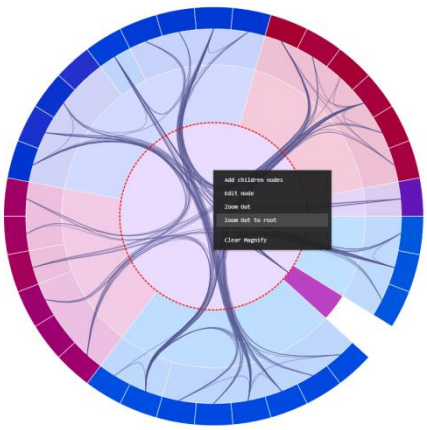

Figure 10: Zoom out/out to root options.

Please note: That when navigating through the hierarchy only the connections between visible nodes in the current view are displayed. In other words, the connections with nodes that don't belong to the descendants of the current root will not be displayed.

4.2. Collapsing and filtering

Navigation over very large hierarchies is sometimes quite complicated. For this reason, Carbonic incorporates techniques that try to solve this problem, such as collapsing nodes or filtering nodes based on certain properties.

4.2.1. Collapsing

The first way to hide complexity from the hierarchy is collapsing nodes. This operation will also allow viewing aggregated connections.

Carbonic has two modes of collapse/uncollapse:

- **Simple:** Simple collapsing hides all the descendant nodes but keeps the state of these nodes when uncollapsing the collapsed node.
- **Recursive:** The recursive collapsing collapses all the nodes up to the leaves (except for the leaves themselves). Therefore, this mode doesn't keep the state of the descendant nodes because of we apply the collapse to all children.

To execute these operations right click in a node and select the options "*Collapse*" or "*Collapse all*" from the menu. If a node is already in a collapsed state, the "*Uncollapse*" or "*Uncollapse recursive*" options will be displayed (*Figure 11.b*).

Besides, collapsed nodes can also be visually identified because they are depicted with a white circle inside (as can be seen in *Figure 11.a* marked by a red circle). For these collapsed nodes this circle will be the beginning and end of the aggregated connections.

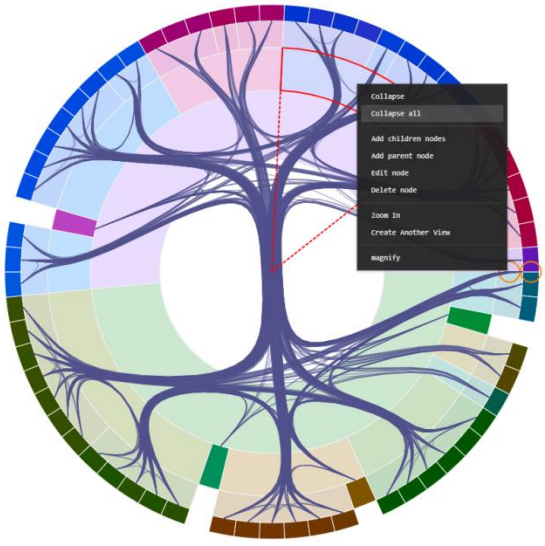

Figure 11.a: Red circle marked a collapsed node.

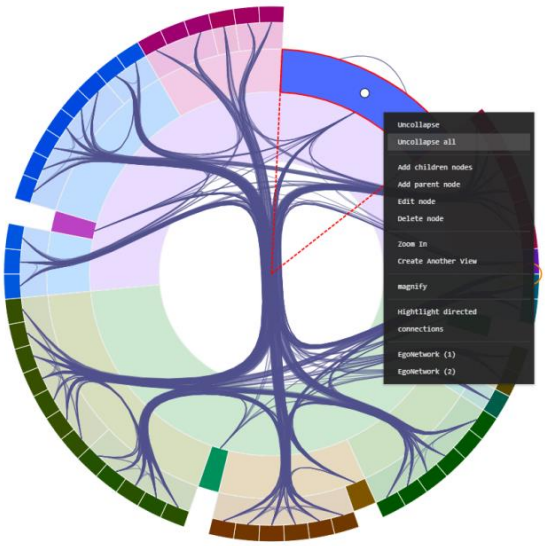

Figure 11.b: Uncollapsing selected node.

4.2.2. Filtering

Carbonic allows filtering nodes based on their properties. The properties of nodes can be defined beforehand depending on the specific domain under analysis. In the specific case of *ConGen* domain, these filters are only executed on the leaf nodes based on the "*Neuron Type*" property.

Carbonic provides two ways of filtering, which depend on the checkbox labelled as “Hide filtered elements” (as shown in the image below).

- **Checkbox marked:** Hides the filtered nodes from the hierarchy.
- **Unmarked checkbox:** Change the fill color of the filtered nodes.

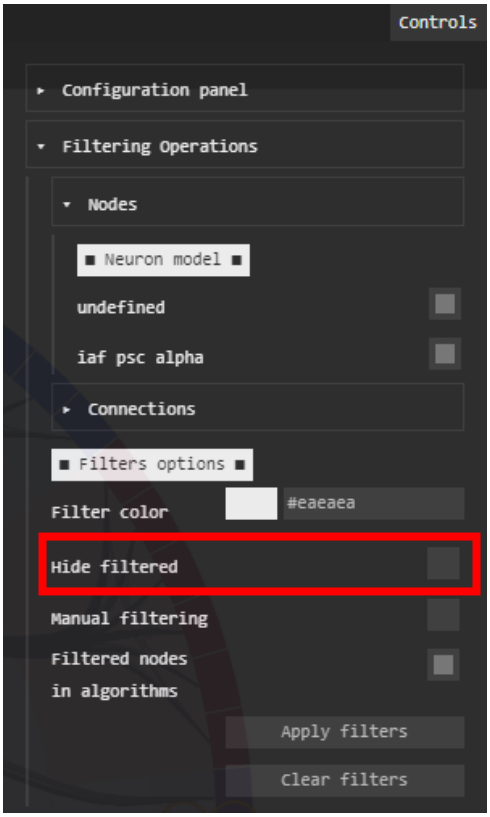

Figure 12: Filtering panel.

To execute the filtering operation, press the “**Apply Filters**” button. Besides, filters can be reseted by clicking on “**Clear Filters**”.

Please note that updating the status of the checkbox (*Figure 12*) means the button “Apply filters” must be pressed again.

Known bug: The “Clear Filters” button does not restore the original status of the “*Neuron Model*” checkboxes.

In *Figure 13*, we can see in the image on the left the original view (*Figure 13.a*) and in the image on the right (Figure 13.b), the first filtering mode (filtered nodes filled with grey color).

In *Figure 14*, we can see in the image on the left the original view (*Figure 14.a*) and in the image on the right (*Figure 14.b*), the second filtering mode (filtered nodes removed).

Figure 13.a: Original view

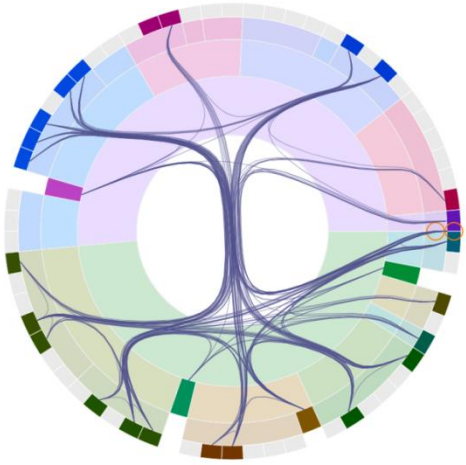

Figure 13.b: Filtered view with "Hide filtered" checkbox unchecked. Grey nodes as filtered nodes.

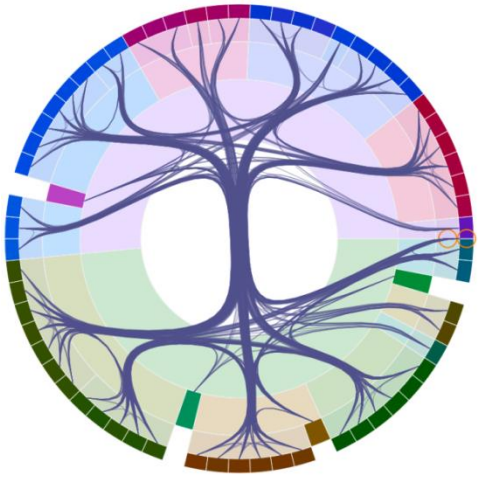

Figure 14.a: Original view

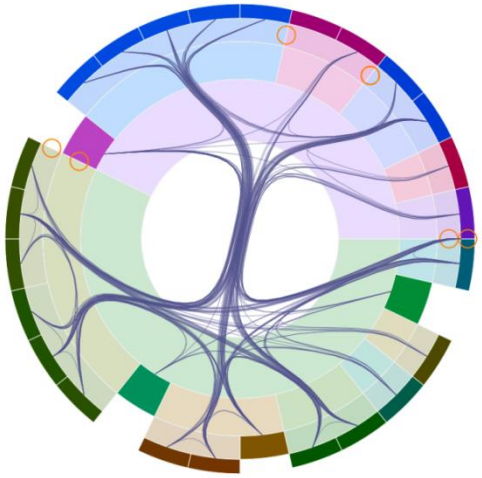

Figure 14.b: Filtered view with "Hide Filtered" checkbox checked. Nodes filtered disappear from visualization.

4.3. Connection display properties

Thirteen visual properties of the connections can be edited (*Figure 15*):

- **Alpha (1):** Global transparency.
- **Alpha Mode (2):** Alpha blending mask
- **Tension or beta (3):** Force used to generate bundles.
- **Conns. Separation (4):** Multiconnection threshold.
- **Max Weight (5):** Maximum thickness.
- **Global normalization (6):** Indicates whether global or local normalization is performed with respect to connection aggregation.
- **Show (7):** Indicate what kind of connections we want to represent.
    - All: Show all connections (direct and aggregated).
    - Only aggregations: Show only aggregated connections.
    - Only direct connections: Show only direct connections.
    - None: Don' show connections.

- **Connection colour (8):** Colour used to display connections.
- **Source colour (9):** Colour used to display source connections on highlighting operations.
- **Target colour (10):** Colour used to display target connections on highlighting operations.
- **External Outside (11):** Display brothers' connections outside.
- **Self-connections (12):** Display self-connections.
- **Show arrows (13):** Display arrows in directed connections.

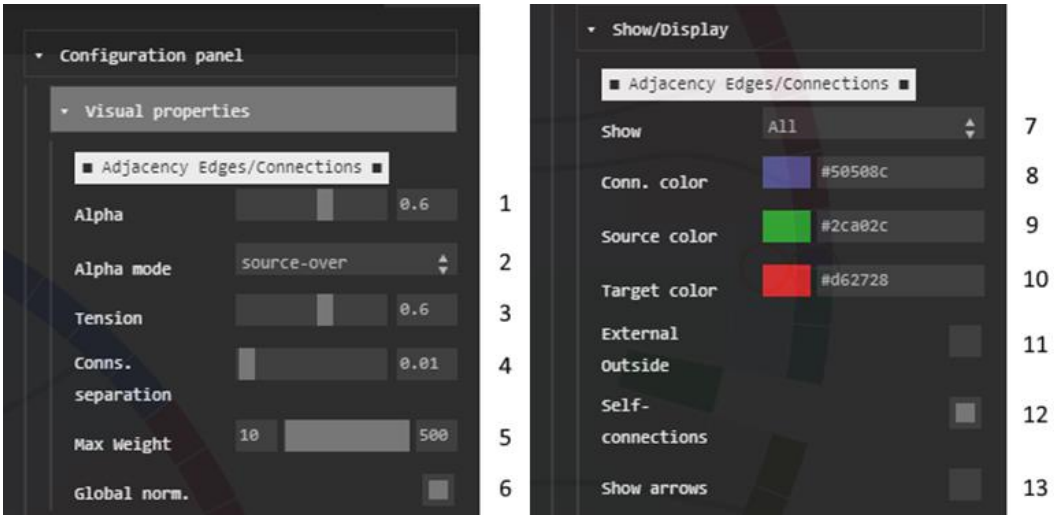

Figure 15: Visual properties panel

4.4. Highlight in and out connections

The input and output connections of a leaf node (or a Stimulator) can be highlighted. To do so, right click on the node or Stimulator and select "*Highlight directed connection*" (*Figure 16.a*)

The green colour is used for connections where the selected node is the source of the connection, while red is used when the selected node is the target (*Figure 16.b*).

Figure 16.a: Highlight directed connections option.

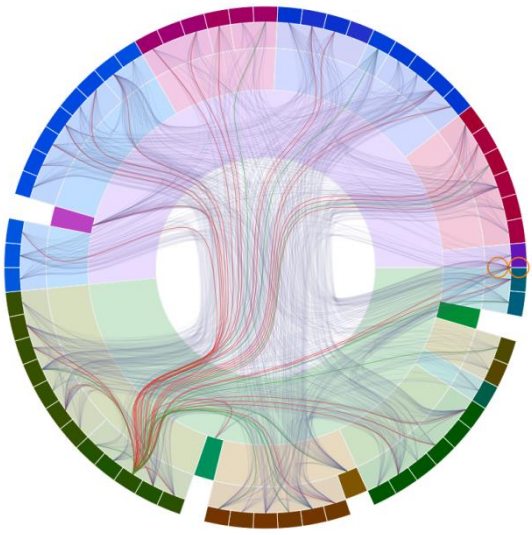

Figure 16.b: Highlight directed connections from a leaf node

5. Scene

Carbonic supports several ways of importing and exporting the current scene.

The supported formats are the following:

- **.backup extension:** This type of files stores the current hierarchy, connections, Stimulators and collapsing states of the nodes.

5.1. Importing

There are several ways to import files to Carbonic:

- Reading a CSV files (nodes and edges) via JavaScript and use the Carbonic API to load the data.
- Using the drag and drop technique. Just drag the file (of any of the types previously enumerated) to be loaded from the file explorer into the canvas and Carbonic will handle the file according to its extension.

5.2. Exporting

Within the "Configuration Panel" panel and the "Scene" subpanel, there are four buttons that allow exporting the current scene to different formats (*Figure 17*).

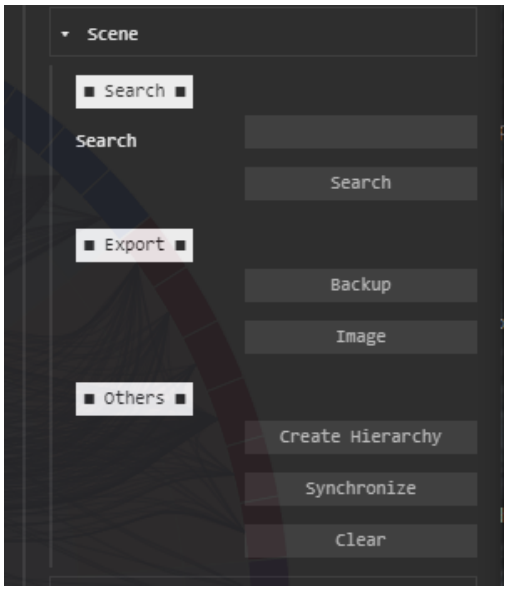

Figure 17: Export panel

The use of the button to export "**Backup**" will generate a file with a predefined name and, depending on the browser, will automatically download it or ask the user to select the desired option.

Alternatively, the "**Image**" button captures the image and stores it at the resolution defined by the user (without hover layer).

6. Appendix

6.1. Domain creation

Creating your own domain allows you to add new properties to Carbonic. The definition of properties forms the main core of the domain specification, since it will provide the visualization with information about the type of property or its possible values. As we can see in Figure X, it shows the definition of node and connection properties, which can be used to perform operations such as assigning properties to the visual channels of hierarchy and connection angle and size, respectively, as well as sorting or hierarchy restructuring operations based on data properties.

```
{
  "name": "airports_world",
  "properties": [ {
    "type": "enum",
    "key": "type",
    "name": "Type",
    "values": [ "airport", "heliport" ],
    "filterable": true,
    "flags": [ "EDITABLE" ]
  }, {
    "type": "scalar",
    "key": "Num Airports",
    "name": "Num Airports",
    "values": [ 0 ],
    "filterable": false,
    "auto": {
      "op": "COUNT",
      "source": {
        "relName": "isParentOf",
        "property": "id",
        "entities": [ "airport" ]
      }
    }
  }, {
    "type": "object",
    "key": "Airport Freq",
    "name": "Airport Freq",
    "values": [ 0 ],
    "filterable": false,
    "auto": {
      "op": "FREQ",
      "source": {
        "relName": "isParentOf",
        "property": "type",
        "entities": [ "airport" ]
      }
    }
  }, {
    "type": "scalar",
    "key": "numConnections",
    "name": "Num Connections",
    "values": [ 0 ],
    "filterable": false,
    "auto": {
      "op": "COUNT",
      "source": { "relName": "connectsWith" }
    }
  }, {
    "type": "enum",
    "key": "stops",
    "name": "Stops",
    "values": [ "0", "1" ],
    "filterable": true
  } ],
  "entities": [ {
    "entity": "root",
    "depth": 0,
    "props": [ "name" ]
  }, {
    "entity": "continent",
    "depth": 1,
    "props": [ "name", "Num Airports",
               "Airport Freq" ]
  }, {
    "entity": "country",
    "depth": 2,
    "props": [ "name", "Airport Freq" ]
  }, {
    "entity": "region",
    "depth": 3,
    "props": [ "name", "Airport Freq" ]
  }, {
    "entity": "municipality",
    "depth": 4,
    "props": [ "name", "Airport Freq" ]
  }, {
    "entity": "airport",
    "depth": 5,
    "props": [ "name", "type" ]
  } ],
  "network": {
    "properties": [ "stops" ],
    "autovalue": true,
    "mode": "undirected"
  },
  "hierarchy": {
    "autovalue": true
  }
}
```

Ontology formalization can be defined through different properties:

- **Type:** Type of the data of this property. Currently *scalar/uint*, *enum*, *boolean*, *vector{2,3,4}*, *date*, *datetime*, *datetime_local* and *object*.
- **Key:** The name of the property (only for non-auto-generated properties).
- **Name:** Refers to the name that will appear in the different entity editing forms.
- **Values:** Possible values (enumerated) or the range in which the value can be moved.
- **Filterable (optional):** Indicates whether filtering can be applied to this property.
- **Flags (optional):** Indicates some usage properties of this property. The possible values are:
  - *UNIQUE:* It doesn't allow to have repeated values among all the entities that have that property in their data.
  - *HIDDEN:* The value exists in the data but is not visible in the edit forms.
  - *EDITABLE:* The value of the property can be edited.
  - *[]:* Allows having a repeatable and visible value, but it cannot be edited.
- **Auto (optional):** Allows you to define the behaviour of the auto-calculated data.
  - **Op:** Type of operation to be applied on the data set to be aggregated. Some of the most used operations are *COUNT*, *MEAN* o *FREQ*.
  - **Source:** This tag indicates how the aggregation operation will be applied:
    - **RelName:** Type of relationship on which the operation is applied:
      - *isParentOf:* Values to be added are part of an entity in the hierarchy.
      - *connectsWith:* Values to be added are part of the connectivity.
    - **Property:** Indicates on which property the aggregation will be performed.
    - **Entities:** Defines on which level(s) of the hierarchy the aggregation will be performed.

The definition of entities or hierarchical nodes allows to indicate the number of levels that make up the hierarchy, as well as the properties (self-calculated or not) that are available for each type of entity. Each entity is defined through the different parameters:

- **Entity:** This tag is like the *"key"* tag of the properties. In some types of aggregations, the value of this property is used to know on which type of entities the aggregation is applied.
- **Depth:** Indicates the depth level. A value of -1 allows infinite depth levels to be created from the previous level containing a positive integer value (greater than 0).
- **Constraints (optional):** Allows defining if this property has certain restrictions, such as having a maximum number of children or if the property is only editable if another property meets certain criteria. There are several types of restrictions, such as *max*, which is used to indicate the maximum number that an entity in the hierarchy can contain; or *subproperty*, which is used so that a certain property only exists (and is editable) if another property has a certain value.
- **Props:** This tag is a list of the different properties contained in an entity. By default, it must contain the value *"name"*.

If we focus on the definition of connectivity, the formalization allows different fields to be configured:

- **Mode (optional):** Allows you to indicate the type of network we are working with. By default, the network will be undirected.
  - *Directed:* Directional network.
  - *Undirected:* Unidirectional network.
- **Properties (optional):** As in the rest of the entities, it defines the different properties that the connections have.