Escher Documentation

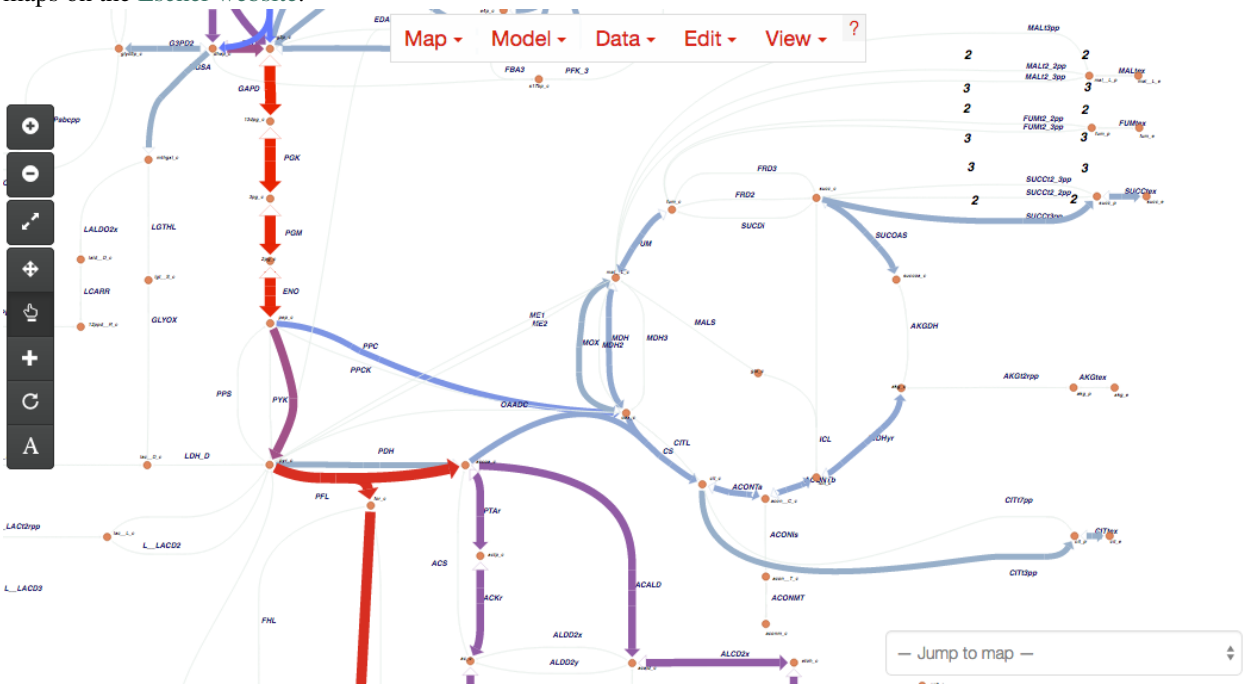
Release 1.0.0

Zachary King

December 11, 2014

Contents

Escher is a web-based tool for building, viewing, and sharing visualizations of metabolic pathways. These 'pathway maps' are a great way to contextualize data about metabolism. The easiest way to use Escher is to browse or build maps on the Escher website.



CHAPTER 1

Features

1. View pathway maps in any modern web browser
2. *Build maps* using the content of genome-scale metabolic models
3. *Visualize data* on reactions, genes, and metabolites
4. Full text search
5. Detailed options for changing colors, sizes, and more, all from the web browser
6. View maps *inside the IPython Notebook*
7. *Embed maps* within any website, with minimal dependencies (escher.js, d3.js, and optionally Twitter Bootstrap)

CHAPTER 2

Supported browsers

We recommend using Google Chrome for optimal performance, but Escher will also run in the latest versions of Firefox, Internet Explorer, and Safari (including mobile Safari).

CHAPTER 3

Installation

Escher can be used without any installation by visiting the Escher website. However, you can install escher if you would like to (1) run Escher offline, (2) include your own maps and models in the launch page, (3) view Escher maps in an IPython Notebook, or (4) modify the source code.

To install the latest stable version of Escher, run:

```
pip install escher
```

For more information, see the documentation on *Escher in the IPython Notebook* and *Developing with Escher*.

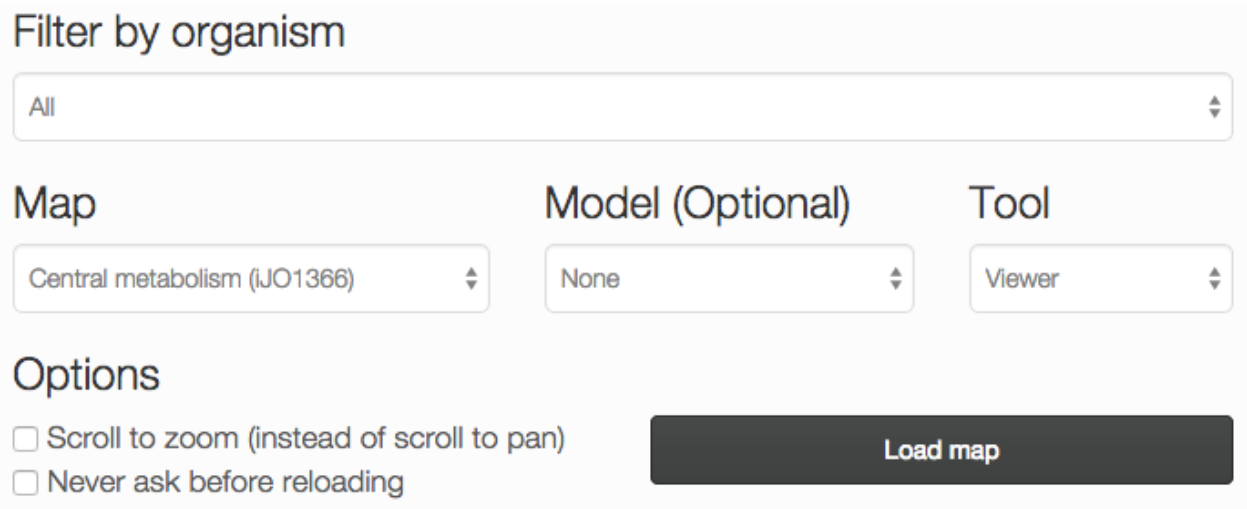
CHAPTER 4

Contents

4.1 Getting Started

4.1.1 The launch page

When you open the Escher launch page, you will see a number of options:

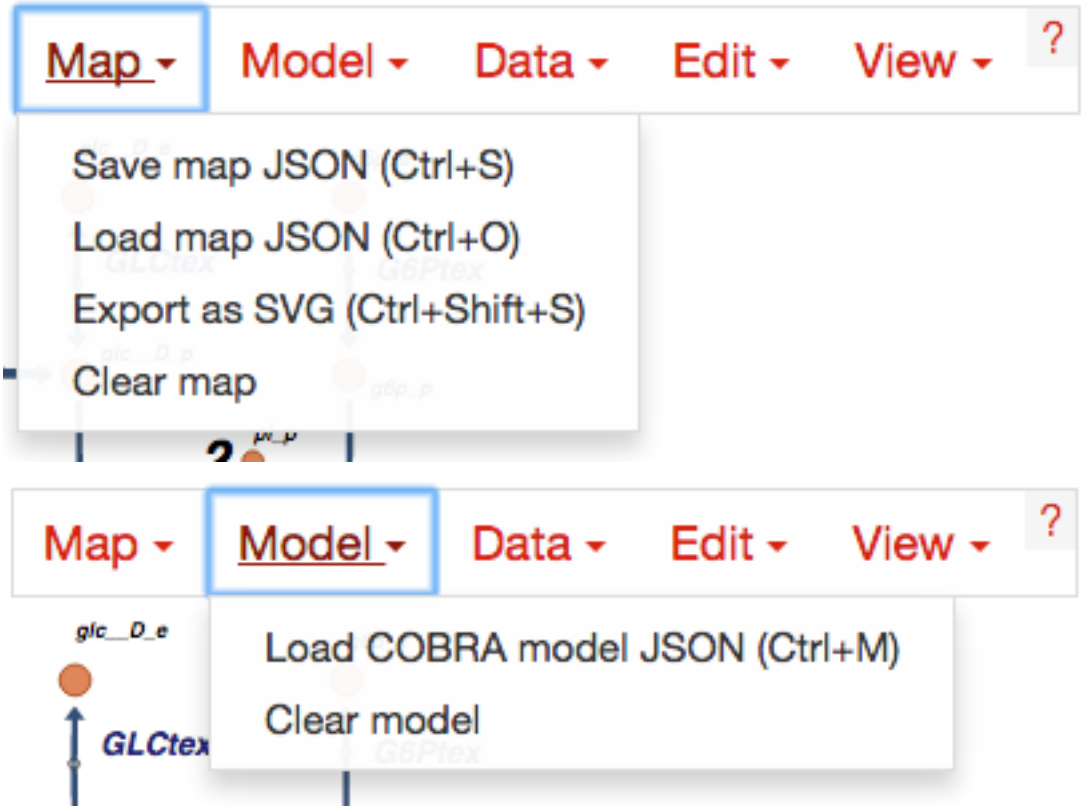


- *Filter by organism*: Choose an organism to filter the Maps and Models.
- *Map*: Choose a pre-built map, or start from scratch with an empty builder by choosing *None*.
- *Model*: (Optional) Choose a COBRA model to load, for building reactions. You can also load your own model after you launch the tool.
- *Tool*:
  - The *Viewer* allows you to pan and zoom the map, and to assign data to reactions, genes, and metabolites.
  - The *Builder*, in addition to the Viewer features, allows you to add reactions, move and rotate existing reactions, and adjust the map canvas.
- *Options*:
  - *Scroll to zoom (instead of scroll to pan)*: Determines the effect of using the mouse's scroll wheel over the map.

- *Never ask before reloading*: If this is checked, then you will not be warned before leaving the page, even if you have unsaved changes.

Choose *Load map* to open the Escher map in a new tab or window.

4.1.2 Loading and saving maps and models



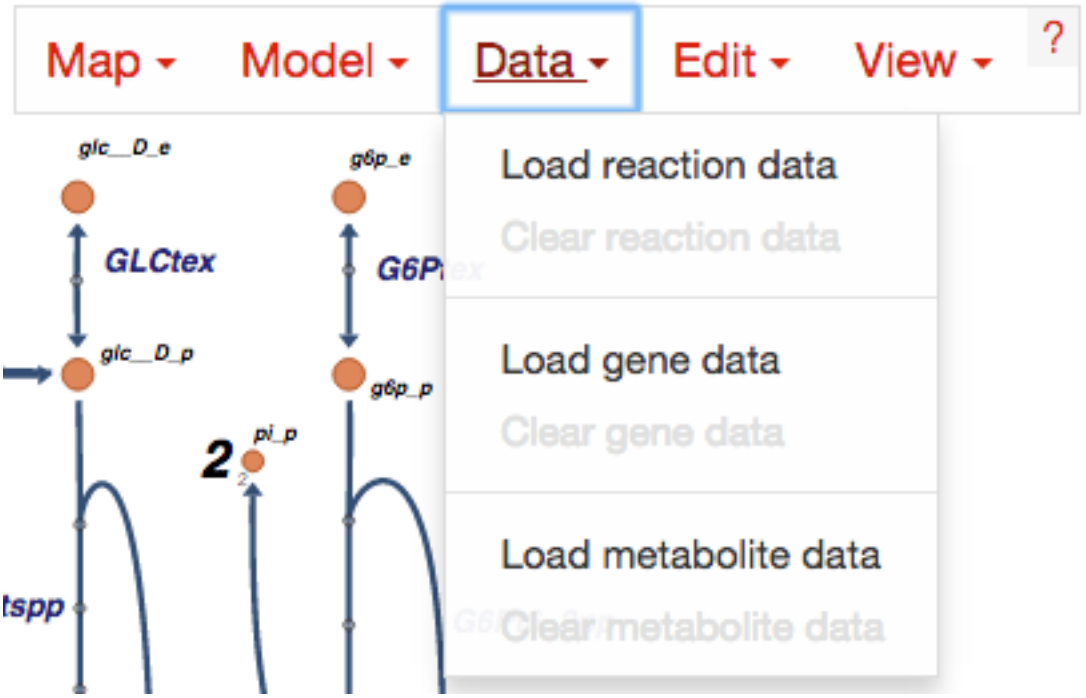
Generating COBRA model JSON files

Generating COBRA models that can be imported into Escher requires the latest beta version of COBRApy which can be found here.

Once you have COBRApy v0.3.0b1 (or later) installed, then you can generate a JSON model by following this example code.

4.1.3 Loading reaction, gene, and metabolite data

Datasets can be loaded as CSV files or JSON files, using the Data Menu.



CSV files should have 1 header row, 1 ID column, and either 1 or 2 columns for data values. Here is an example with a single data value columns:

```
ID,time 0sec
glc__D_c,5.4
g6p__D_c,2.3
```

Which might look like this is Excel:

ID	time 0sec
glc__D_c	5.4
g6p_c	2.3

If two datasets are provided, then the Escher map will display the difference between the datasets. In the Settings menu, the **Comparison** setting allows you to choose between comparison functions (Fold Change, Log2(Fold Change), and Difference). With two datasets, the CSV file looks like this:

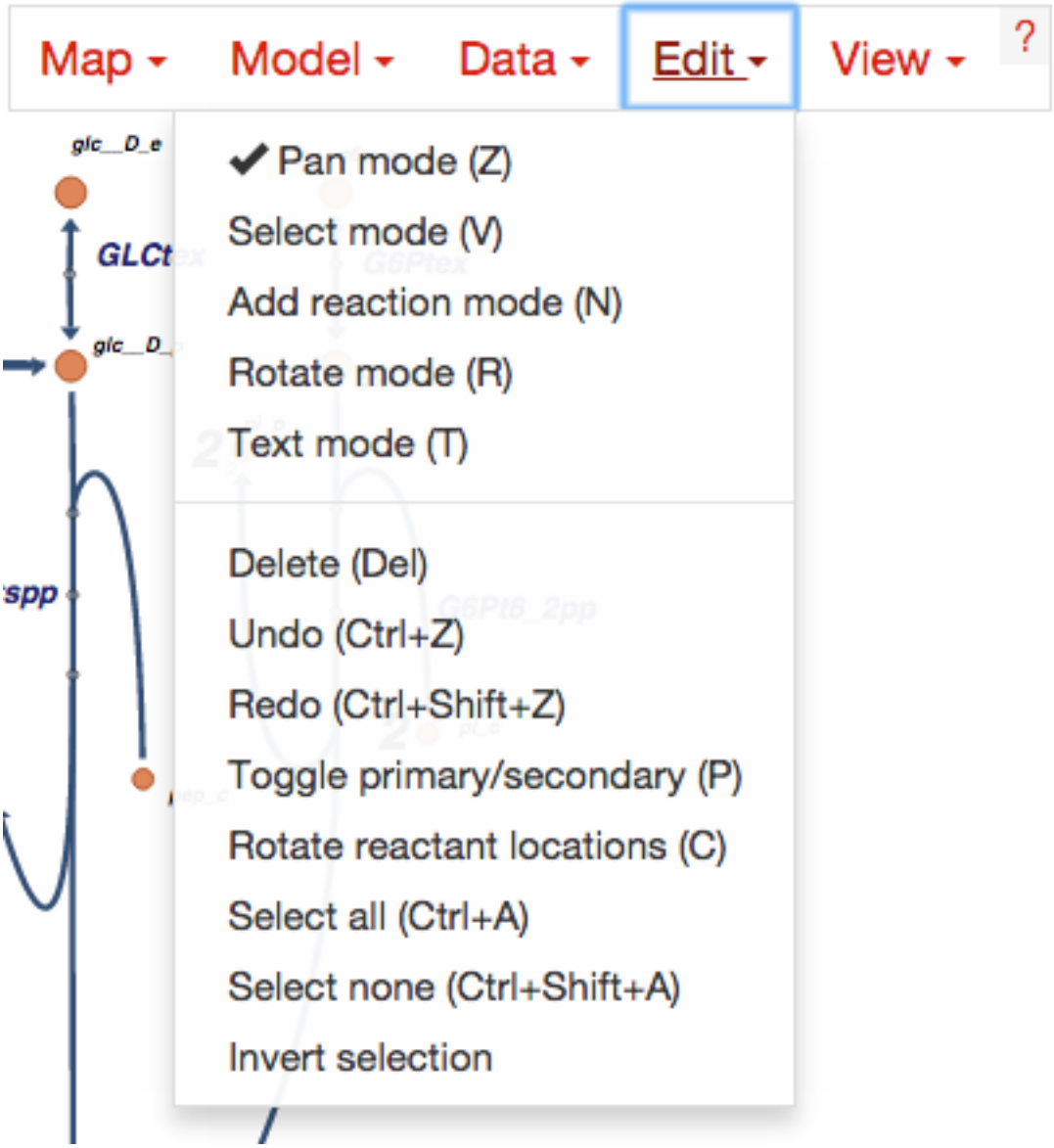
ID	time 0sec	time 5s
glc__D_c	5.4	10.2
g6p_c	2.3	8.1

Data can also be loaded from a JSON file. This Python code snippet provides an example of generating the proper format for single reaction data values and for reaction data comparisons:

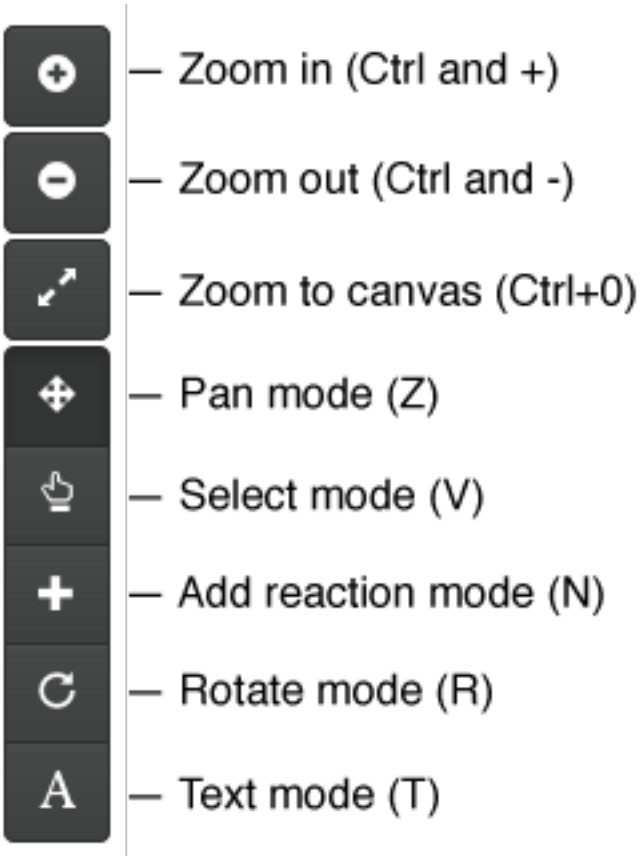
```
import json

# save a single flux vector as JSON
flux_dictionary = {'glc__D_c': 5.4, 'g6p_c': 2.3}
with open('out.json', 'w') as f:
    json.dump(flux_dictionary, f)

# save a flux comparison as JSON
flux_comp = [{'glc__D_c': 5.4, 'g6p_c': 2.3}, {'glc__D_c': 10.2, 'g6p_c': 8.1}]
with open('out_comp.json', 'w') as f:
    json.dump(flux_comp, f)
```

Gene data and gene reaction rules

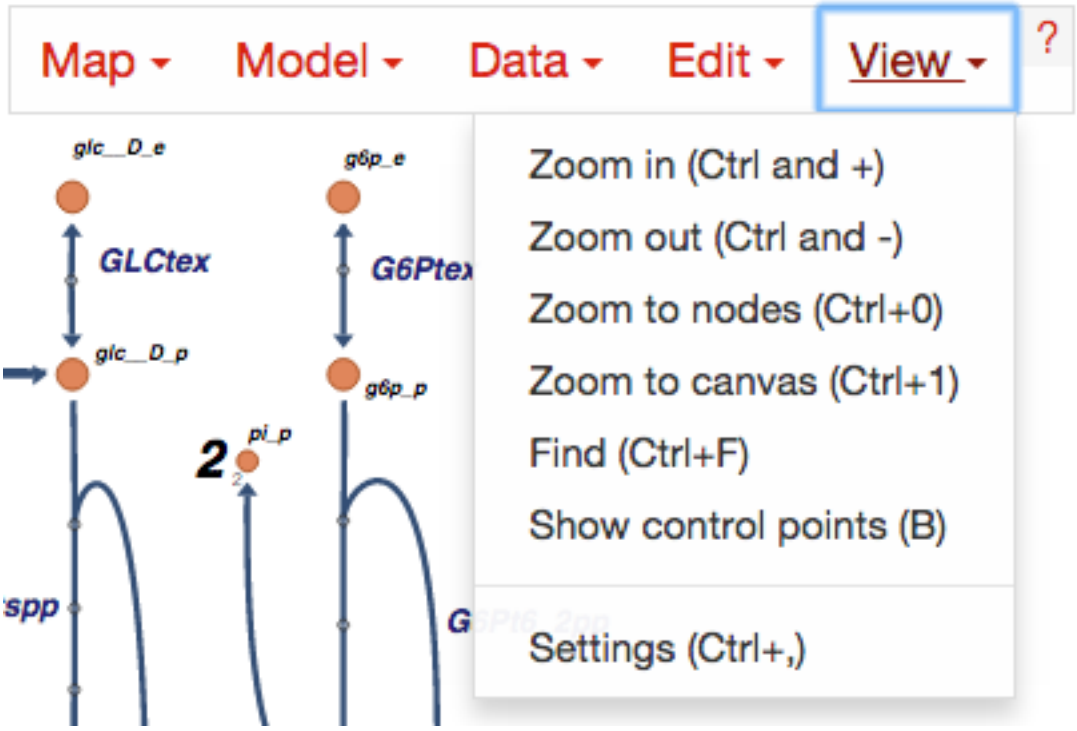
4.1.4 Editing and building



Common functions can be accessed using the buttons in the bar on the left of the screen.



4.1.5 View options



4.2 Escher in the IPython Notebook

The Python package for Escher can be installed using pip:

```
pip install escher --pre
```

Alternatively, one can download the source files and install the package directly:

```
python setup.py install
```

Once you have installed Escher locally, you can interact with Escher maps in an IPython Notebook. This example notebook outlines the basic idea.

Dependencies:

- Jinja2
- Tornado
- COBRApy, 0.3.0b4 or later

4.3 Developing with Escher

4.3.1 Using the static javascript files

You can include the compiled Escher javascript file in any html document. The only dependencies are d3.js, and Twitter Bootstrap if you are using the option menu='all'.

These files can be found in `escher/lib`.

4.3.2 Running the local server

You can run your own local server if you want to modify the Escher code, or use Escher offline. To get started, install the Python package and run from any directory by calling:

```
python -m escher.server
```

This starts a server at localhost:7778. You can also choose another port:

```
python -m escher.server --port=8005
```

4.3.3 Building and testing escher

Build the minified and non-minified javascript files:

```
python setup.py buildjs
```

Test Python and start Jasmine for JavaScript testing:

```
python setup.py test
```

Build the static website:

```
python setup.py buildgh
```

Clear static website files:

```
python setup.py clean
```

4.3.4 The Escher file format

4.4 Contribute Maps

If you would like to contribute maps to Escher, you can make a Pull Request to the GitHub repository escher.github.io. Make sure there is a folder with the name of the organism in `1-0-0/maps`. For example, a new yeast map goes in the folder:

```
1-0-0/maps/Saccharomyces cerevisiae/
```

Then, name your map by concatenating the model ID and the map name, separated by a period. For example, a yeast map built with the genome-scale model iMM904 could be named:

```
iMM904.Amino acid biosynthesis.json
```

Then, add the JSON file for the model to the Pull Request *if that model is not already available*. As before, make a folder for your organism within `1-0-0/models/`. The model filename is just the model ID.

In this example, a correct Pull Request would include the following files:

```
1-0-0/maps/Saccharomyces cerevisiae/iMM904.Amino acid biosynthesis.json
1-0-0/models/Saccharomyces cerevisiae/iMM904.json
```

4.5 JavaScript API

class `escher.`**`Builder`**(*map_data*, *model_data*, *embedded_css*, *options*)

A Builder object contains all the UI and logic to generate a map builder or viewer.

Arguments

- **map_data** (*object*) – The data for a map, to be passed to escher.Map.from_data(). If null, then an empty Builder is initialized
- **model_data** (*object*) – The data for a cobra model, to be passed to escher.CobraModel(). Can be null.
- **embedded_css** (*string*) – The stylesheet for the SVG elements in the Escher map.
- **selection** (*object*) – (Optional, Default: In the body element) The d3 selection of an element to place the Builder into. The selection cannot be inside an SVG element.
- **options** (*object*) –

  (Optional) An object defining any of the following options:

  `options.`**`unique_map_id`**
  A unique ID that will be used to UI elements don't interfere when multiple maps are in the same HTML document.

  `options.`**`primary_metabolite_radius`**
  (Default: 15) The radius of primary metabolites, in px.

  `options.`**`secondary_metabolite_radius`**
  (Default: 10) The radius of secondary metabolites, in px.

options.**marker_radius**
(Default: 5) The radius of marker nodes, in px.

options.**gene_font_size**
(Default: 18) The font size of the gene reaction rules, in px.

options.**hide_secondary_nodes**
(Default: false) If true, then secondary nodes and segments are hidden. This is convenient for generating simplified map figures.

options.**show_gene_reaction_rules**
(Default: false) If true, then show the gene reaction rules, even without gene data.

options.**reaction_data**
An object with reaction ids for keys and reaction data points for values.

options.**reaction_styles**

options.**reaction_compare_style**
(Default: 'diff') How to compare to datasets. Can be either 'fold, 'log2_fold', or 'diff'.

options.**reaction_scale**

options.**reaction_no_data_color**

options.**reaction_no_data_size**

options.**gene_data**
An object with Gene ids for keys and gene data points for values.

options.**and_method_in_gene_reaction_rule**
(Default: mean) When evaluating a gene reaction rule, use this function to evaluate AND rules. Can be 'mean' or 'min'.

options.**metabolite_data**
An object with metabolite ids for keys and metabolite data points for values.

options.**metabolite_compare_style**
(Default: 'diff') How to compare to datasets. Can be either 'fold', 'log2_fold' or 'diff'.

options.**metabolite_scale**

options.**metabolite_no_data_color**

options.**metabolite_no_data_size**

View and build options

options.**identifiers_on_map**
Either 'bigg_id' (default) or 'name'.

options.**highlight_missing**
(Default: false) If true, then highlight reactions that are not in the loaded model in red.

options.**allow_building_duplicate_reactions**
(Default: true) If true, then building duplicate reactions is allowed. If false, then duplicate reactions are hidden in *Add reaction mode*.

Callbacks

options.**first_load_callback**
A function to run after loading the Builder.

Callbacks

```
this.callback_manager.run('view_mode');
this.callback_manager.run('build_mode');
this.callback_manager.run('brush_mode');
this.callback_manager.run('zoom_mode');
this.callback_manager.run('rotate_mode');
this.callback_manager.run('text_mode');
this.callback_manager.run('load_model', null, model_data, should_update_data);
this.callback_manager.run('update_data', null, update_model, update_map, kind, should_draw);
```

load_map (*map_data*[, *should_update_data*])
Load a map for the loaded data. Also reloads most of the Builder content.

Arguments

- **map_data** – The data for a map.
- **should_update_data** (*Boolean*) – (Default: true) Whether data should be applied to the map.

load_model (*model_data*[, *should_update_data*])
Load the cobra model from model data.

Arguments

- **model_data** – The data for a Cobra model. (Parsing in done by escher.CobraModel).
- **should_update_data** (*Boolean*) – (Default: true) Whether data should be applied to the model.

view_mode ()
Enter view mode.

build_mode ()
Enter build mode.

brush_mode ()
Enter brush mode.

zoom_mode ()
Enter zoom mode.

rotate_mode ()
Enter rotate mode.

text_mode ()
Enter text mode.

set_reaction_data (*data*)

Arguments

- **data** (*array*) – An array of 1 or 2 objects, where each object has keys that are reaction ID's and values that are data points (numbers).

set_metabolite_data (*data*)

Arguments

- **data** (*array*) – An array of 1 or 2 objects, where each object has keys that are metabolite ID's and values that are data points (numbers).

set_gene_data (*data*)

Arguments

- **data** (*array*) – An array of 1 or 2 objects, where each object has keys that are gene ID's and values that are data points (numbers).

4.6 Python API

class escher.**Builder**(*map_name=None*, *map_json=None*, *model=None*, *model_name=None*, *model_json=None*, *embedded_css=None*, *reaction_data=None*, *metabolite_data=None*, *gene_data=None*, *local_host=None*, *id=None*, *safe=False*, ***kwargs*)

A metabolic map that can be viewed, edited, and used to visualize data.

This map will also show metabolic fluxes passed in during consruction. It can be viewed as a standalone html inside a browswer. Alternately, the respresentation inside an IPython notebook will also display the map.

Maps are stored in json files and are stored in a cache directory. Maps which are not found will be downloaded from a map repository if found.

Parameters

- **map_name** – A string specifying a map to be downloaded from the Escher web server, or loaded from the cache.
- **map_json** – A JSON string, or a file path to a JSON file, or a URL specifying a JSON file to be downloaded.
- **model** – A Cobra model.
- **model_name** – A string specifying a model to be downloaded from the Escher web server, or loaded from the cache.
- **model_json** – A JSON string, or a file path to a JSON file, or a URL specifying a JSON file to be downloaded.
- **embedded_css** – The CSS (as a string) to be embedded with the Escher SVG.
- **reaction_data** – A dictionary with keys that correspond to reaction ids and values that will be mapped to reaction arrows and labels.
- **metabolite_data** – A dictionary with keys that correspond to metabolite ids and values that will be mapped to metabolite nodes and labels.
- **gene_data** – A dictionary with keys that correspond to gene ids and values that will be mapped to corresponding reactions.
- **local_host** – A hostname that will be used for any local files in dev mode.
- **id** – Specify an id to make the javascript data definitions unique. A random id is chosen by default.
- **safe** – If True, then loading files from the filesytem is not allowed. This is to ensure the safety of using Builder with a web server.

Keyword Arguments

These are defined in the Javascript API:

•identifiers_on_map

•show_gene_reaction_rules

•unique_map_id

•primary_metabolite_radius

- secondary_metabolite_radius
- marker_radius
- hide_secondary_metabolites
- reaction_styles
- reaction_compare_style
- reaction_scale
- reaction_no_data_color
- reaction_no_data_size
- and_method_in_gene_reaction_rule
- metabolite_styles
- metabolite_compare_style
- metabolite_scale
- metabolite_no_data_color
- metabolite_no_data_size
- highlight_missing
- allow_building_duplicate_reactions

All keyword arguments can also be set on an existing Builder object using setter functions, e.g.:

```
my_builder.set_reaction_styles(new_styles)
```

display_in_browser(*ip=u'127.0.0.1'*, *port=7655*, *n_retries=50*, *js_source=u'web'*, *menu=u'all'*, *scroll_behavior=u'pan'*, *enable_editing=True*, *enable_keys=True*, *minified_js=True*, *never_ask_before_quit=False*)

Launch a web browser to view the map.

Parameters

- **ip** – The IP address to serve the map on.
- **port** – The port to serve the map on. If specified the port is occupied, then a random free port will be used.
- **n_retries** (*int*) – The number of times the server will try to find a port before quitting.
- **js_source** (*string*) – Can be one of the following:
  - *web* (Default) - Use JavaScript files from escher.github.io.
  - *local* - Use compiled JavaScript files in the local Escher installation. Works offline.
  - *dev* - Use the local, uncompiled development files. Works offline.
- **menu** (*string*) – Menu bar options include:
  - *none* - No menu or buttons.
  - *zoom* - Just zoom buttons.
  - *all* (Default) - Menu and button bar (requires Bootstrap).
- **scroll_behavior** (*string*) – Scroll behavior options:
  - *pan* - Pan the map.

  - *zoom* - Zoom the map.
  - *none* (Default) - No scroll events.
- **enable_editing** (*Boolean*) – Enable the map editing modes.
- **enable_keys** (*Boolean*) – Enable keyboard shortcuts.
- **minified_js** (*Boolean*) – If True, use the minified version of js files. If js_source is *dev*, then this option is ignored.
- **never_ask_before_quit** (*Boolean*) – Never display an alert asking if you want to leave the page. By default, this message is displayed if enable_editing is True.

display_in_notebook (*js_source=u'web'*, *menu=u'zoom'*, *scroll_behavior=u'none'*, *minified_js=True*, *height=500*)

Embed the Map within the current IPython Notebook.

Parameters

- **js_source** (*string*) – Can be one of the following:
  - *web* (Default) - Use JavaScript files from escher.github.io.
  - *local* - Use compiled JavaScript files in the local Escher installation. Works offline.
  - *dev* - Use the local, uncompiled development files. Works offline.
- **menu** (*string*) – Menu bar options include:
  - *none* - No menu or buttons.
  - *zoom* - Just zoom buttons.
  - Note: The *all* menu option does not work in an IPython notebook.
- **scroll_behavior** (*string*) – Scroll behavior options:
  - *pan* - Pan the map.
  - *zoom* - Zoom the map.
  - *none* - (Default) No scroll events.
- **minified_js** (*Boolean*) – If True, use the minified version of js files. If js_source is *dev*, then this option is ignored.
- **height** – Height of the HTML container.

save_html (*filepath=None*, *js_source=u'web'*, *menu=u'all'*, *scroll_behavior=u'pan'*, *enable_editing=True*, *enable_keys=True*, *minified_js=True*, *never_ask_before_quit=False*, *static_site_index_json=None*)

Save an HTML file containing the map.

Parameters

- **filepath** (*string*) – The HTML file will be saved to this location.
- **js_source** (*string*) – Can be one of the following:
  - *web* (Default) - Use JavaScript files from escher.github.io.
  - *local* - Use compiled JavaScript files in the local Escher installation. Works offline.
  - *dev* - Use the local, uncompiled development files. Works offline.
- **menu** (*string*) – Menu bar options include:
  - *none* - No menu or buttons.

  - *zoom* - Just zoom buttons.
  - *all* (Default) - Menu and button bar (requires Bootstrap).
- **scroll_behavior** (*string*) – Scroll behavior options:
  - *pan* - Pan the map.
  - *zoom* - Zoom the map.
  - *none* (Default) - No scroll events.
- **enable_editing** (*Boolean*) – Enable the map editing modes.
- **enable_keys** (*Boolean*) – Enable keyboard shortcuts.
- **minified_js** (*Boolean*) – If True, use the minified version of js files. If js_source is *dev*, then this option is ignored.
- **height** (*number*) – Height of the HTML container.
- **never_ask_before_quit** (*Boolean*) – Never display an alert asking if you want to leave the page. By default, this message is displayed if enable_editing is True.
- **static_site_index_json** (*string*) – The index, as a JSON string, for the static site. Use javascript to parse the URL options. Used for generating static pages (see static_site.py).

4.6.1 Cache

escher.**get_cache_dir**(*name=None*)
Get the cache dir as a string.

Parameters **name** (*string*) – An optional subdirectory within the cache

escher.**clear_cache**(*different_cache_dir=None*)
Empty the contents of the cache directory.

Parameters **different_cache_dir** (*string*) – (Optional) The directory of another cache. This is mainly for testing.

escher.**list_cached_maps**()
Return a list of all cached maps.

escher.**list_cached_models**()
Return a list of all cached models.

escher.**list_available_maps**()
Return a list of all maps available on the server

escher.**list_available_models**()
Return a list of all models available on the server

- *genindex*

CHAPTER 5

License

Escher is licensed under the MIT license.

e

B

C

D

E

G

L

O

R

S

T

V

Z