Clustergrammer Documentation

Release 1.1.0

Nicolas F. Fernandez

Oct 05, 2018

Contents

Clustergrammer is a web-based tool for visualizing high-dimensional data (e.g. a matrix) as an interactive and shareable hierarchically clustered heatmap. Clustergrammer's front-end (*Clustergrammer-JS*) is built using D3.js and its back-end (*Clustergrammer-PY*) is built using Python. Clustergrammer produces highly interactive visualizations that enable intuitive exploration of high-dimensional data and has several optional biology-specific features (e.g. enrichment analysis, see *Biology-Specific Features*) to facilitate the exploration of gene-level biological data. Press play or interact with the gene-expression demo below to see some of Clustergrammer's interactive features and see *Interacting with the Visualization* for more information:

CHAPTER 1

Using Clustergrammer

The easiest ways to use Clustergrammer to produce an interactive visualization of your data are:

- upload a tab-separated matrix file using the Clustergrammer web app homepage
- or use the *Clustergrammer Jupyter Widget* within a Jupyter notebook and share using nbviewer

The *Clustergrammer Web App* is the quickest way to generate an interactive and shareable visualization (see example visualization and *getting started Web-app*). For users that want to visualize their data within a notebook workflow, the *Clustergrammer Jupyter Widget* enables visualizations to be built within Jupyter notebooks and shared through Jupyter's nbviewer (see example notebook and *getting started Widget*).

Web developers can use Clustergrammer's core libraries (*Clustergrammer-JS* and *Clustergrammer-PY*) or the *Clustergrammer-Web API* to dynamically generate visualizations for their own web applications (see examples in *App Integration Examples*).

Please read the *Getting Started* guide for more information.

CHAPTER 2

Use Cases

Clustergrammer was built to visualize biological data but is applicable for visualizing high-dimensional data in general. Below are links to several example use cases (see *Case Studies and Examples* for more information):

- Cancer Cell Line Encyclopedia Gene Expression Data
- Single Cell RNA-seq Data Visualization
- Zika Virus RNA-seq Data Visualization
- Kinase Substrate Similarity Network
- Iris flower dataset
- MNIST Handwritten Digit Dataset

CHAPTER 3

Contact

Please contact Avi Ma'ayan (avi.maayan@mssm.edu) and Nicolas Fernandez (nicolas.fernandez@mssm.edu) for support, comments, and suggestions.

CHAPTER 4

Contents:

4.1 Getting Started

Clustergrammer is a web-based tool for visualizing high-dimensional data as interactive and shareable hierarchically clustered heatmaps. Clustergrammer can be used in three main ways:

1. *Clustergrammer Web App*
2. *Clustergrammer Jupyter Widget*
3. *Clustergrammer-JS* and *Clustergrammer-PY* libraries

This section will provide instructions on how to quickly visualize your data using the *Clustergrammer Web App* and the *Clustergrammer Jupyter Widget* as well as instructions on how to interact with the visualization. For developers interested in building their own web page using Clustergrammer, please see the *Web-Development with Clustergrammer* section.

4.1.1 Clustergrammer Web-App

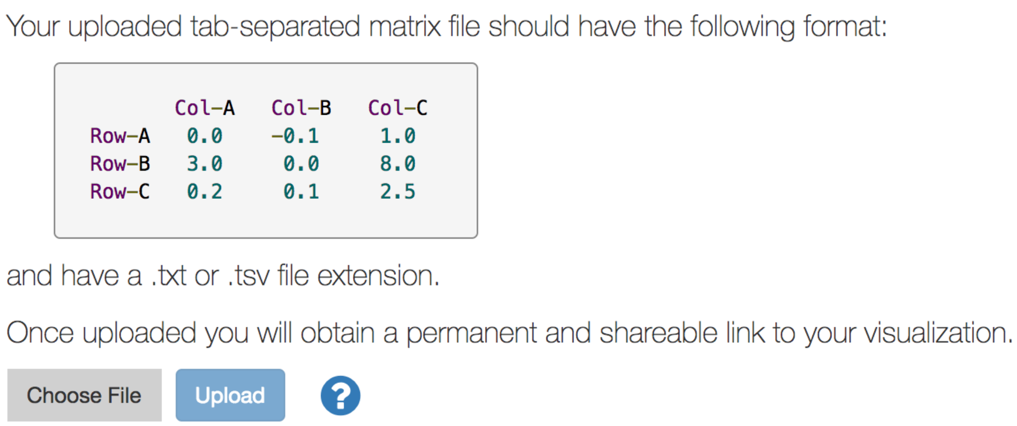
Users can easily generate an interactive and shareable heatmap visualization using the *Clustergrammer Web App*. Simply upload a tab-separated matrix file at the homepage (see screenshot below) to be redirected to a permanent and shareable visualization of your data.

Once you upload your data, the *Clustergrammer Web App* clusters your data and produces three views: a heatmap of your input matrix, a similarity matrix of your columns, and a similarity matrix of your rows. See the screenshots below and the example visualization for an example results page.

Heatmap View

Similarity Matrix View

Users can share their interactive visualizations using the permanent link. See *Interacting with the Visualization* for more information.



Your uploaded tab-separated matrix file should have the following format:

```
       Col-A  Col-B  Col-C
Row-A  0.0    -0.1   1.0
Row-B  3.0     0.0   8.0
Row-C  0.2     0.1   2.5
```

and have a .txt or .tsv file extension.

Once uploaded you will obtain a permanent and shareable link to your visualization.

Fig. 1: Users can upload their data using the web app homepage. Simply choose your file and upload to be redirected to your permanent and shareable visualization.

4.1.2 Clustergrammer-Widget

Jupyter notebooks are ideal for generating reproducible workflows and analysis. They are also the best way to share Clustergrammer's interactive visualizations while providing context, analysis, and the underlying data to enable reproducibility (see *Sharing with nbviewer*). The *Clustergrammer Jupyter Widget* enables users to easily produce interactive visualizations within a Jupyter notebook that can be shared with collaborators (using nbviewer). The *Clustergrammer Jupyter Widget* can be used to visualize a matrix of data from a file or from a Pandas DataFrame (see *Matrix Formats and Input/Output* for more information).

To use the *Clustergrammer Jupyter Widget* users need to install: Python, Jupyter notebook, the widget dependencies (see *Jupyter Widget Dependencies*), and ipywidgets version >6.0.0 (to save the notebook with widgets). The `clustergrammer_widget` can the be installed (with pip) and enabled using the following commands:

```
pip install clustergrammer_widget
jupyter nbextension enable --py --sys-prefix widgetsnbextension
jupyter nbextension enable --py --sys-prefix clustergrammer_widget
```

With the `clustergrammer_widget` installed and enabled users can visualize their data (from a file) using the following Python commands:

```
# import clustergrammer_widgets and initialize network object
from clustergrammer_widget import *
net = Network()

# load matrix file and cluster using default parameters
net.load_file('rc_two_cats.txt')
net.make_clust()

# make interactive widget
clustergrammer_widget(network=net.widget())
```

See the screenshot below for an example widget visualization:

Users can download and reproduce the example notebook, Running_clustergrammer_widget.ipynb, by cloning it's GitHub repo. For more information about using the widget (e.g. loading data from a Pandas DataFrame and sharing using nbviewer) see *Clustergrammer Jupyter Widget*.

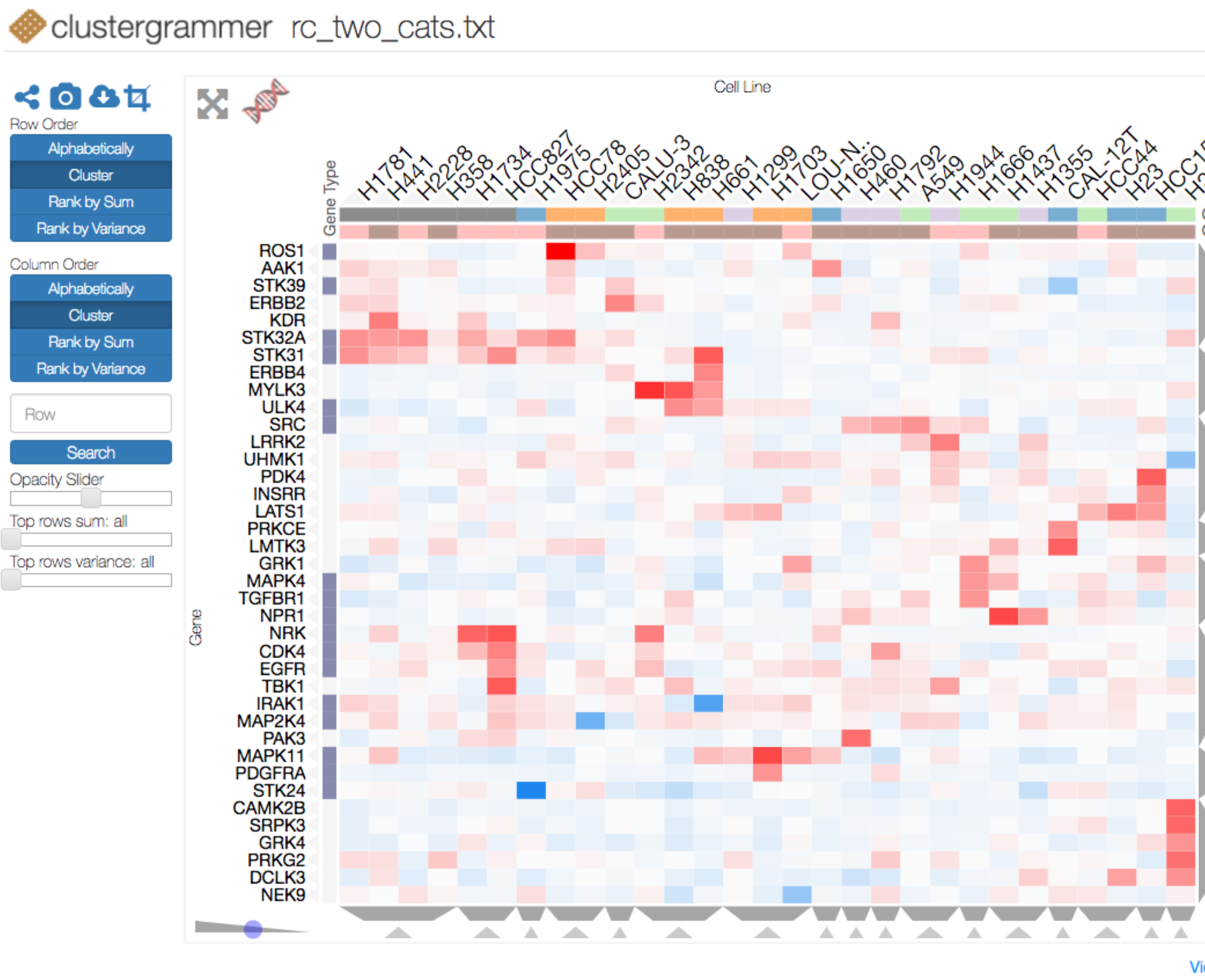


Fig. 2: Above is an example clustergram visualization produced by the *Clustergrammer Web App*. Clustergrammer produces three views of your data and the clustered heatmap view is shown above.

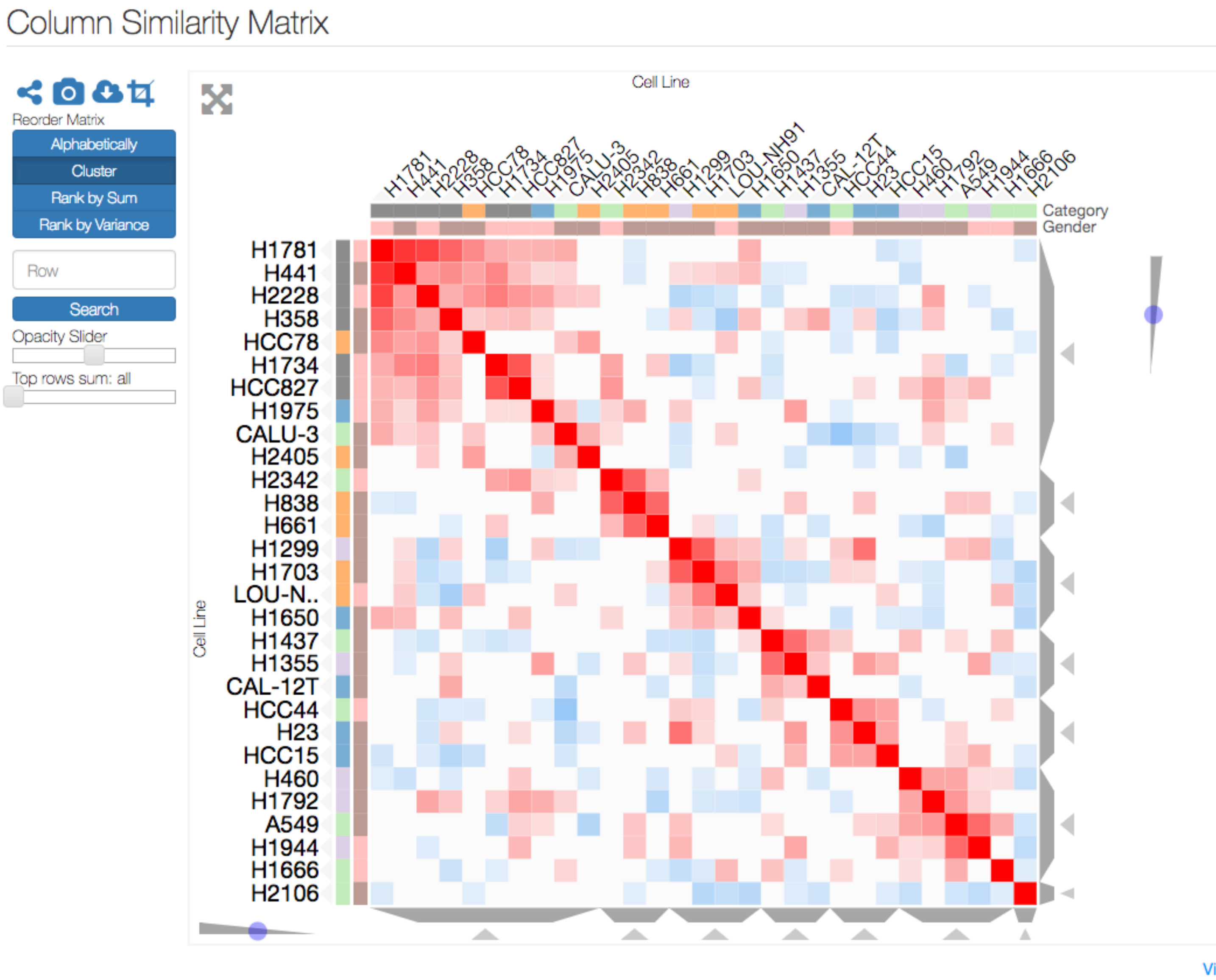
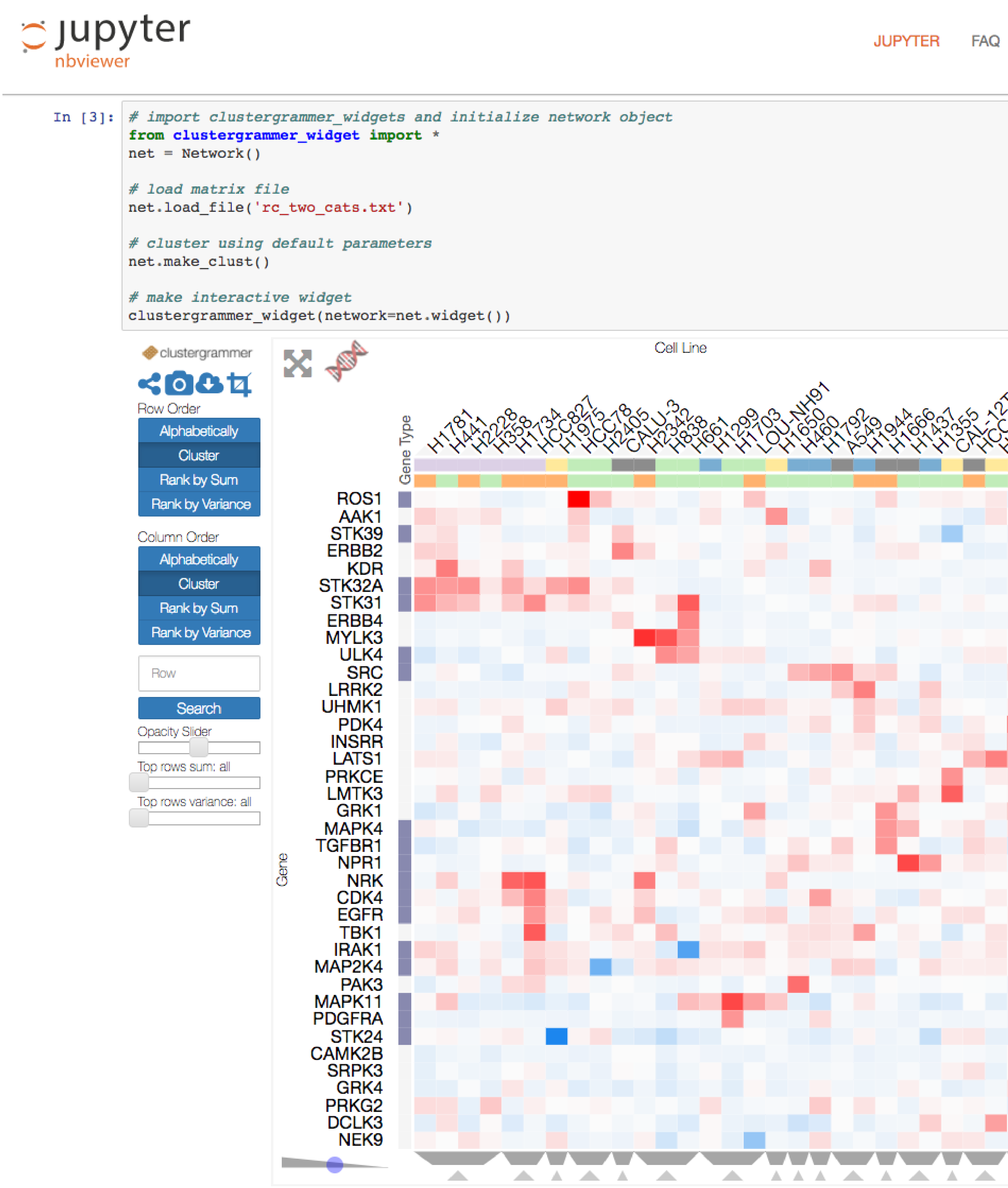


Fig. 3: Clustergrammer produces similarity matrices of rows and columns to provide additional perspectives on a user's data. Above is an example column similarity matrix.



```
In [3]: # import clustergrammer_widgets and initialize network object
        from clustergrammer_widget import *
        net = Network()

        # load matrix file
        net.load_file('rc_two_cats.txt')

        # cluster using default parameters
        net.make_clust()

        # make interactive widget
        clustergrammer_widget(network=net.widget())
```

Fig. 4: Clustergrammer can be used as an interactive widget within a Jupyter notebook and shared using nbviewer (see Running_clustergrammer_widget.ipynb example).

4.1.3 Interacting with Clustergrammer

Clustergrammer produces highly interactive visualizations that enable intuitive exploration of high-dimensional data including:

- *Zooming and Panning*
- *Row and Column Reordering* (e.g. reorder based on sum)
- *Interactive Dendrogram*
- *Interactive Dimensionality Reduction* (e.g. filter rows based on variance)a
- *Interactive Categories*
- *Cropping*
- *Row Searching*

Press play or interact with the gene-expression demo below to see some of Clustergrammer's interactive features and see *Interacting with the Visualization* for more information:

Clustergrammer also has *Biology-Specific Features* for working with gene-level data including:

- mouseover gene names and description look-up (using Harmonizome)
- enrichment analysis to find biological information (e.g. up-stream transcription factors) specific to your set of genes (using Enrichr)

4.2 Clustergrammer Web App

The Web-App enables users to easily generate interactive and shareable heatmap visualizations by uploading their data as a tab-separated file.

4.2.1 Uploading Data using the Web-App Homepage

4.2.2 Clustergrammer-Web Visualization

Uploading a matrix to the Web-App will redirect the user to a new permanent and shareable page with three views of their data:

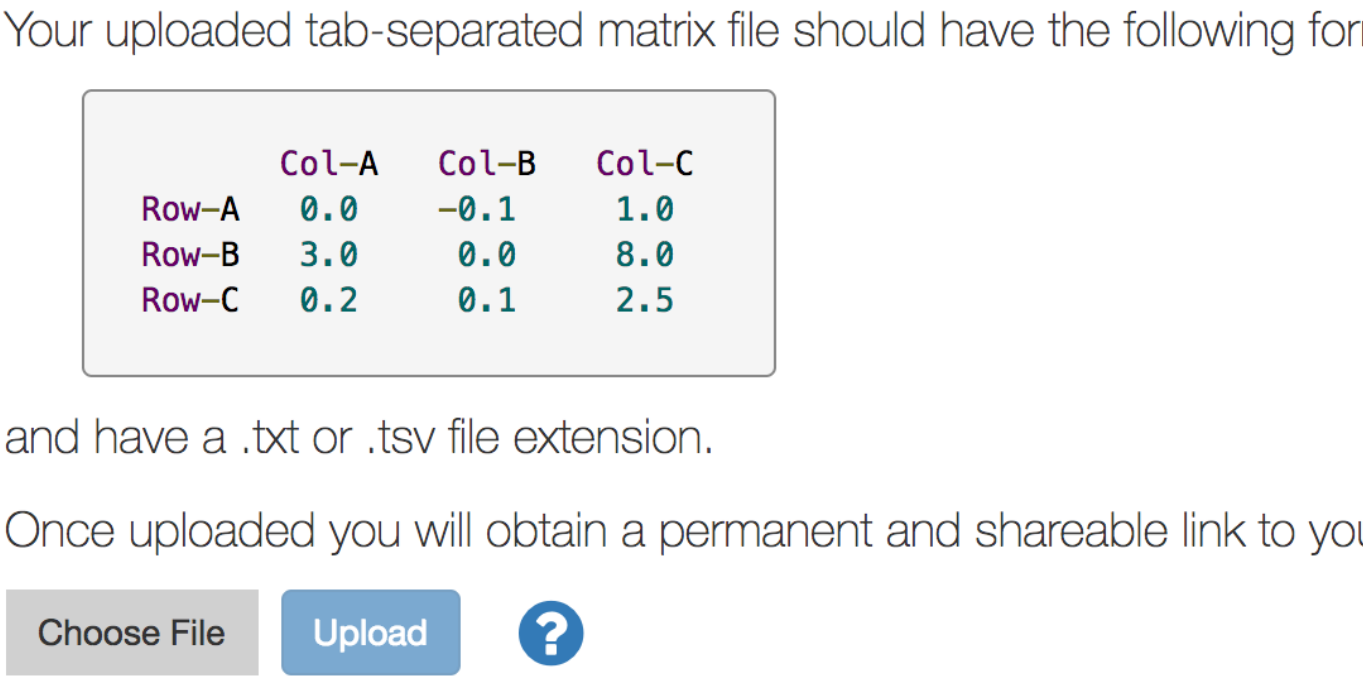
1. clustered heatmap view of their matrix
2. clustered similarity matrix of the columns in their original matrix
3. clustered similarity matrix of the rows in their original matrix

See the screenshots below and the example visualization for an example Web-App visualization page.

Heatmap View

Similarity Matrix View

Users can view the heatmap/similarity-matrices in full screen by clicking the blue link under the visualizations. All visualizations are permanent and shareable, which enables sharing with collaborators. See *Interacting with the Visualization* for more information.



Your uploaded tab-separated matrix file should have the following for

```
        Col-A   Col-B   Col-C
Row-A   0.0    -0.1     1.0
Row-B   3.0     0.0     8.0
Row-C   0.2     0.1     2.5
```

and have a .txt or .tsv file extension.

Once uploaded you will obtain a permanent and shareable link to you

Fig. 5: Users can easily generate an interactive and shareable heatmap visualization using the Web-App. Simply upload a tab-separated matrix file in the homepage to be redirected to a permanent and shareable visualization of your data.

4.2.3 Clustergrammer-Web API

Clustergrammer-Web's RESful API enables users to programmatically generate visualizations. The API can be useful for users that need to generate many clustergrams or developers that need to automatically generate visualizations for their own web application.

Matrix Upload

Users can post a matrix file to Clustergrammer-Web using the endpoint

```
http://amp.pharm.mssm.edu/clustergrammer/matrix_upload/
```

and receive a permanent link to their visualization. Below is an example in Python 2.7 showing the post request and how to obtain the link from the response object:

```python
import requests

filename = 'example_matrix.txt'
upload_url = 'http://amp.pharm.mssm.edu/clustergrammer/matrix_upload/'

r = requests.post(upload_url, files={'file': open(filename, 'rb')})

link = r.text
```

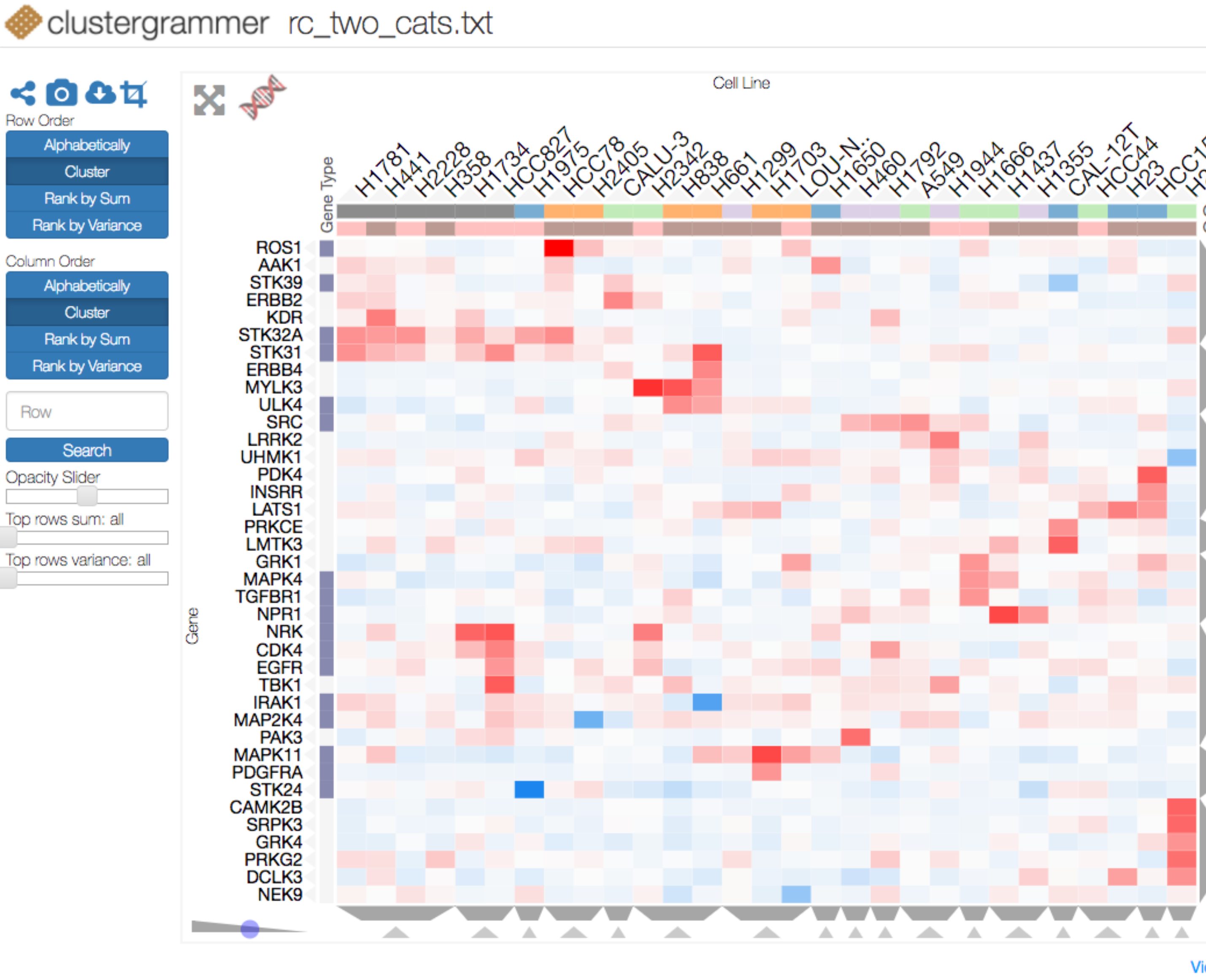


Fig. 6: Above is an example clustergram visualization produced by the Web-App. Clustergrammer produces three views of your data and the clustered heatmap view is shown above.

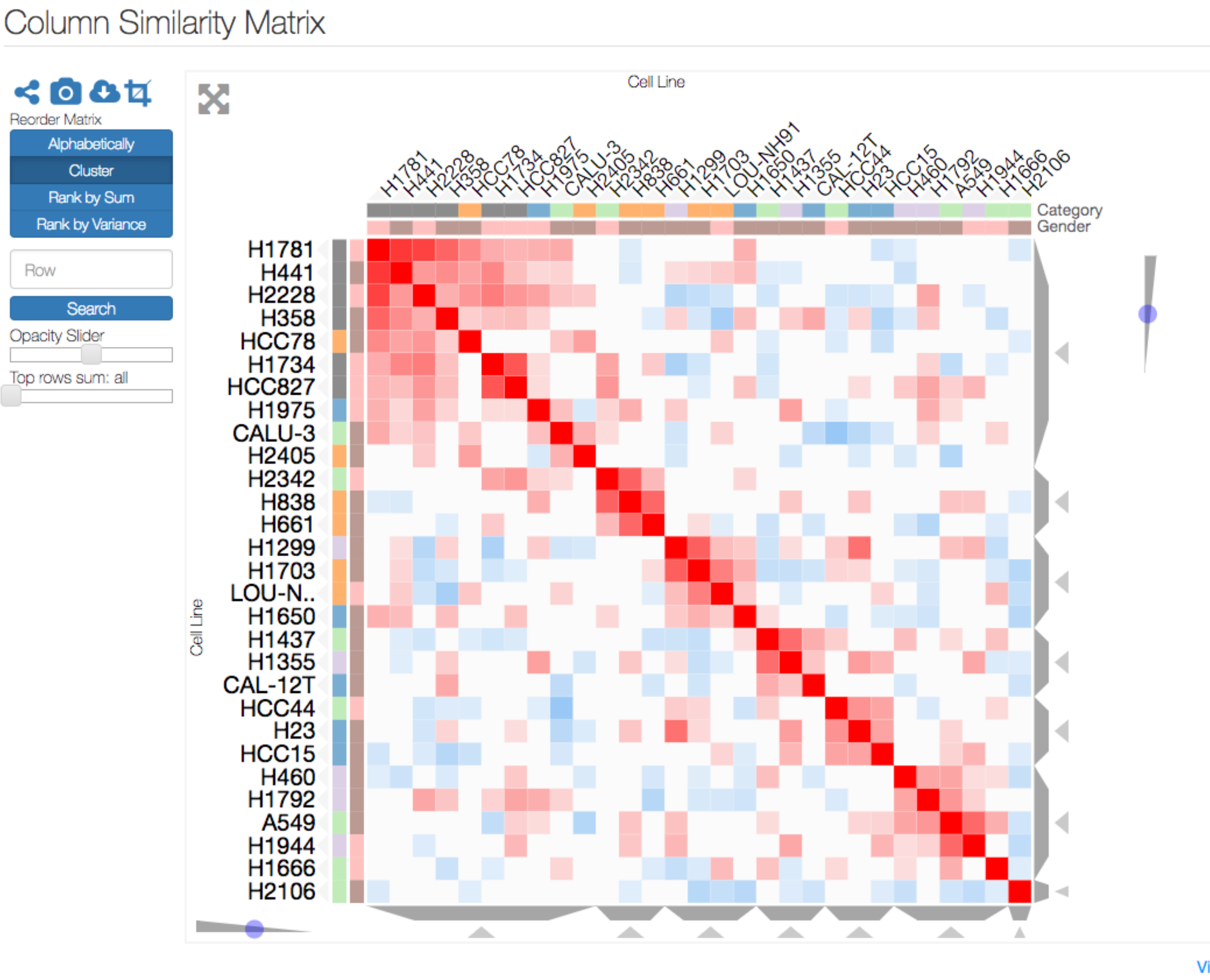


Fig. 7: The Web-App produces similarity matrices of rows and columns to provide additional perspectives on a user's data. Above is an example column similarity matrix.

4.2.4 Clustergrammer-Web Development

Clustergrammer-Web is a dockerized web application built using the Python library Flask and MongoDB database. Clustergrammer-Web uses the *Clustergrammer-JS* and *Clustergrammer-PY* libraries and the source code can be found in the clustergrammer-web GitHub repo.

4.3 Clustergrammer Jupyter Widget

Jupyter notebooks are ideal for generating reproducible workflows and analysis. They are also the best way to share Clustergrammer's interactive visualizations while providing context, analysis, and the underlying data to enable reproducibility (see *Sharing with nbviewer*). The Clustergrammer Widget enables users to easily produce interactive visualizations within a Jupyter notebook that can be shared with collaborators (using nbviewer). Clustergrammer-Widget can be used to visualize a matrix of data from a file or from a Pandas DataFrame (see *Matrix Formats and Input/Output* for more information). See screenshot below for an example visualization:

4.3.1 Jupyter Widget Dependencies

- Numpy
- SciPy
- Pandas
- ipywidgets

Clustergrammer-Widget works with Python 2 and 3.

4.3.2 Installation

Clustergrammer-Widget can be installed (with pip) and enabled using the following commands:

```
pip install clustergrammer_widget
jupyter nbextension enable --py --sys-prefix widgetsnbextension
jupyter nbextension enable --py --sys-prefix clustergrammer_widget
```

4.3.3 Clustergrammer-Widget Workflow Example

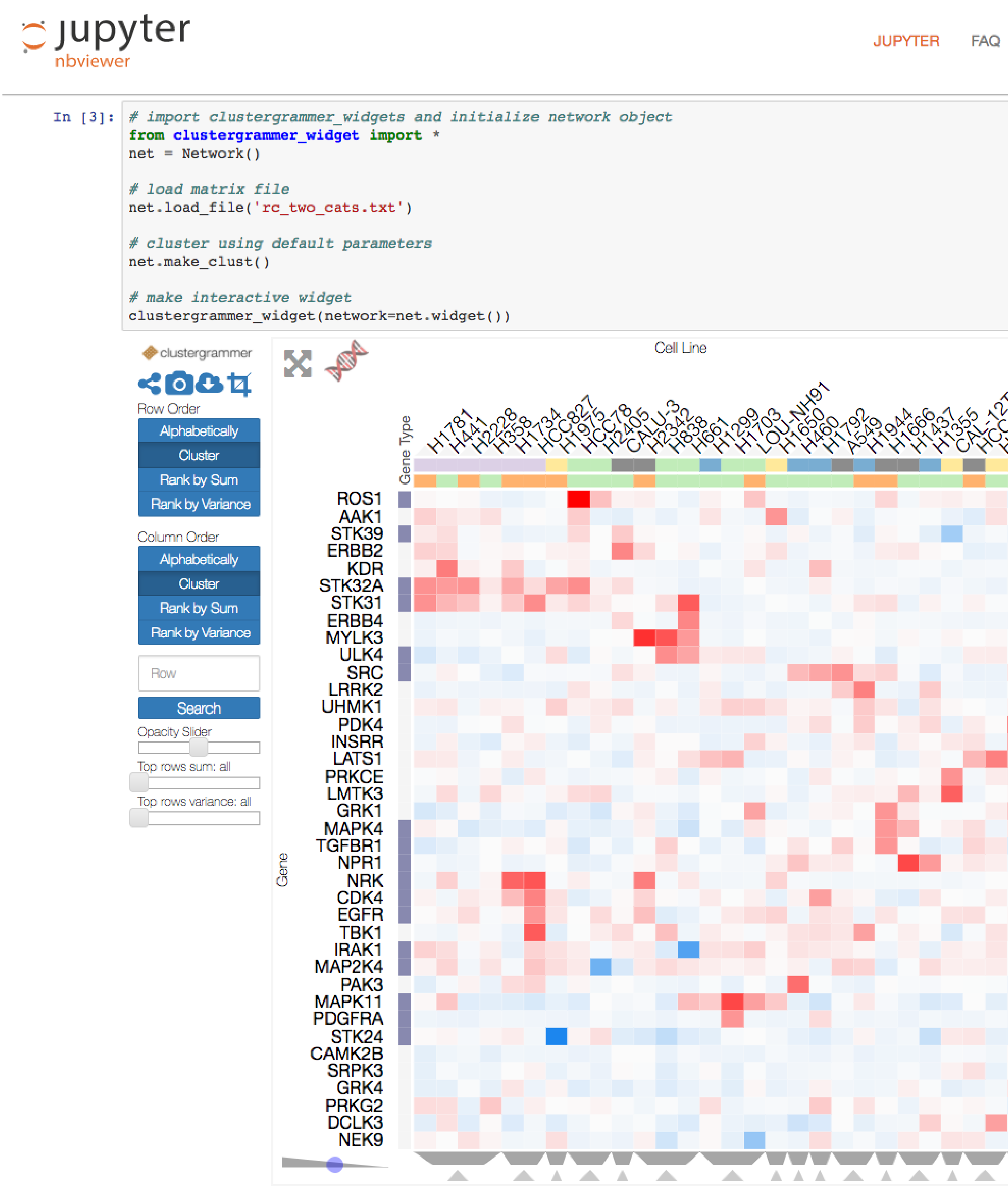
The Jupyter notebook Running_clustergrammer_widget.ipynb (which is rendered using nbviewer) shows how to visualize a matrix from a file and a Pandas DataFrame. The following examples are taken from this notebook.

Here we are visualizing a matrix of data from a file (e.g. `rc_two_cats.txt`). We start by making an instance of the `Network` object, `net`, which is used to load and cluster the data. Then we pass the data to `clustergrammer_widget` to generate the visualization (for more information about the `Network` class, see *Clustergrammer-PY API*):

```
# import clustergrammer_widgets and initialize network object
from clustergrammer_widget import *
net = Network()

# load matrix file
net.load_file('rc_two_cats.txt')
```

(continues on next page)



```
# import clustergrammer_widgets and initialize network object
from clustergrammer_widget import *
net = Network()

# load matrix file
net.load_file('rc_two_cats.txt')

# cluster using default parameters
net.make_clust()

# make interactive widget
clustergrammer_widget(network=net.widget())
```

Fig. 8: Clustergrammer can be used as an interactive widget within a Jupyter notebook and shared using nbviewer (see Running_clustergrammer_widget.ipynb example).

(continued from previous page)

```
# cluster using default parameters
net.make_clust()

# make interactive widget
clustergrammer_widget(network=net.widget())
```

Clustergrammer-Widget can also be used as a general purpose Pandas DataFrame viewer. Below is an example of how to visualize a Pandas DataFrame, `df`, by loading it into the same `net` object from above:

```
# load DataFrame
net.load_df(df)

# cluster using default parameters
net.make_clust()

# make interactive widget
clustergrammer_widget(network=net.widget())
```

Loading new data into `net` clears out the old data, which allows `net` to be easily reused within the same notebook. The `net` object can also be used to filter and normalize your data before visualizing (note that filtering and normalization are permanent and irreversible). The example below performs Z-score normalization on the columns and filters to keep the top 200 rows based on their absolute value sum:

```
# Z-score normalize columns
net.normalize(axis='col', norm_type='zscore', keep_orig=True)

# filter for the top 200 rows based on their absolute value sum
net.filter_N_top('row', 200, 'sum')

# make interactive widget
clustergrammer_widget(network=net.widget())
```

In the examples above, we clustered our matrix using the default parameters. For more information about the `Network` object and additional options see the *Clustergrammer-PY API*.

4.3.4 Sharing with nbviewer

To enable rendering interactive widgets on nbviewer you must have ipywidgets version 6 or later installed and use the "Save Notebook with Widgets" action in the Widgets menu in the Jupyter notebook (see ipywidgets Rendering Interactive Widgets on nbviewer documentation and screenshot below):

4.3.5 Jupyter Notebook Examples

Clustergrammer has been applied to visualize and analyze a wide variety of biological and non-biological data. See the examples Jupyter notebooks below for examples:

- Running_clustergrammer_widget.ipynb
- DataFrame_Example.ipynb
- Single Cell RNA-seq Visualization.ipynb
- Iris Dataset.ipynb

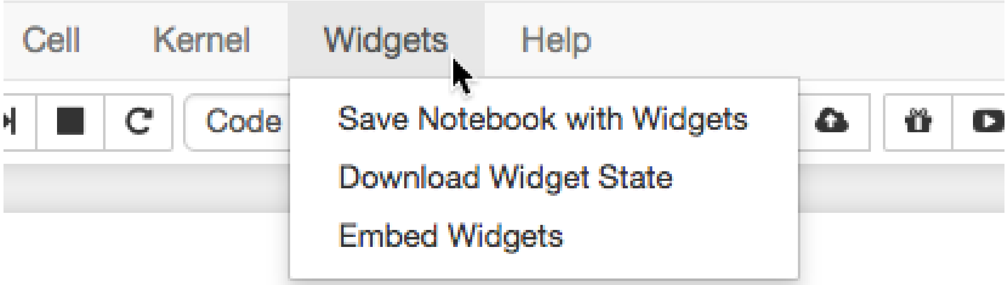


Fig. 9: Users can save notebooks with interactive HTML widgets using the “Save Notebook with Widgets” action in the Jupyter Notebook Widgets menu as shown here. ipywidgets version 6 or later must be installed in order to enable this feature.

4.3.6 Clustergrammer-Widget Development

Clustergrammer-Widget’s source code can be found in the clustergrammer-widget GitHub repo. Clustergrammer-Widget is built using the ipywidgets framework (using the cookie cutter template).

Please *Contact* Nicolas Fernandez or Avi Ma’ayan with questions or use the GitHub issues feature to raise an issue.

4.4 Interacting with the Visualization

Data visualizations benefit enormously from user interactions that allow users to explore their data and interactively generate new views. Clustergrammer produces highly interactive heatmaps that enable users to intuitively explore and perform complex transformations on their data. Clustergrammer visualizations are built using the *Clustergrammer-JS* library and are consistent across the *Clustergrammer Web App* and the *Clustergrammer Jupyter Widget*. This section will overview heatmaps as a visualization tool and cover the types of interactions that are available to the user.

4.4.1 Introduction to Clustergrams/Heatmaps

Clustergrammer visualizes high-dimensional data as a hierarchically clustered matrix with colored tiles (red for positive numbers and blue for negative numbers) and row/column labels, which is commonly referred to as a heatmap or clustergram (this documentation uses the terms ‘heatmap’ and ‘clustergram’ interchangeably; see Eisen et al., 1998 for an early example using biological data). Clustergrams also typically use dendrogram trees to depict the hierarchy of row and column clusters produced by hierarchical clustering.

Heatmaps are powerful visualizations that enable users to directly visualize high-dimensional data without the loss of information and interpretability associated with dimensionality reduction techniques (e.g. t-SNE). For instance, columns can depict data-points (e.g. measured entities) and rows can depict data-dimensions (e.g. measured variables). In this way, heatmaps can visualize thousands of data-points in thousands of dimensions (e.g. in thousand(s)-dimensional space). However, static heatmaps are of limited use for visualizing large datasets (e.g. for large matrices, visualization elements and labels become too small to read). Furthermore, static heatmaps prevent users from interactively exploring their data, e.g. reordering rows/columns. We built Clustergrammer, in part, to address these issues.

4.4.2 Interactive Demo

Press play to take a quick tour of some of Clustergrammer’s interactive features or interact with the demo to explore for yourself:

4.4.3 Zooming and Panning

Clustergrammer allows users to zoom and pan into their heatmap by scrolling and dragging. This is useful for working with large datasets, where labels are not readable without zooming, and for closely investigating regions of interest.

Zooming and Panning Behavior

In general, zooming and panning occur in two stages. First zooming/panning occurs in the direction that matrix-cells have been more compressed (e.g. if there are more more rows than columns, then matrix-cells will be compressed in the vertical direction and the matrix-cells will be wide). Once zooming has decompressed matrix-cells (e.g. matrix-cells height and width are the same) then zooming/panning occurs in both directions. For instance, when visualizing a matrix with many more columns than rows zooming/panning will occur in the horizontal direction first until matrix-cells have equal width and height, then zooming/panning will be allowed in the vertical and horizontal directions. For symmetrical matrices, e.g. adjacency matrices, matrix-cells always have equal width and height and zooming/panning always occurs in both directions.

Large Matrix Zooming and Panning Behavior

Clustergrammer is capable of visualizing matrices with up to ~500,000 to ~750,000 matrix-cells, but is optimized to visualize matrices with more rows than columns. Clustergrammer uses row-downsampling to improve visualization performance for large matrices. If a user visualizes a matrix with a large number of rows (e.g. >1000-2000 rows) such that each matrix-cell is less than 1 pixel tall, then Clustergrammer will perform row downsampling. When zoomed out, the user will see a downsampled (e.g. coarse grained) version of their data. Zooming into the matrix will bring up successively less downsampled views until the original data is shown (e.g. when the original matrix-cells are > 1 pixel tall). Clustergrammer will only display row labels when their font size is at a readable level (above ~5 pixels). Clustergrammer will also hide row/column labels while zooming into large matrices to improve zooming performance.

4.4.4 Mouseover Interactions

Mousing over elements in the heatmap (e.g. row names) brings up additional information using tooltips. For instance, mousing over matrix-cells brings up a tooltip with the row name, column name, and value of the matrix-cell (see below).

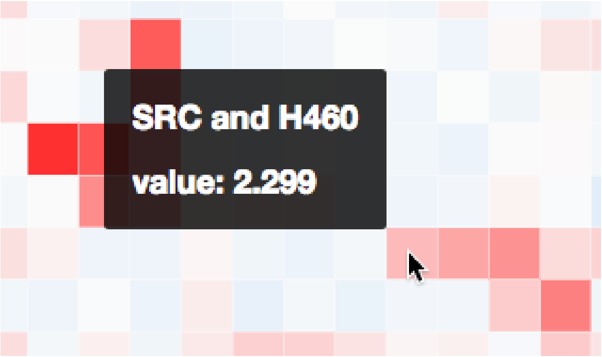


Fig. 10: Mousing over visualization elements (e.g. matrix cell) brings up additional information as a tooltip.

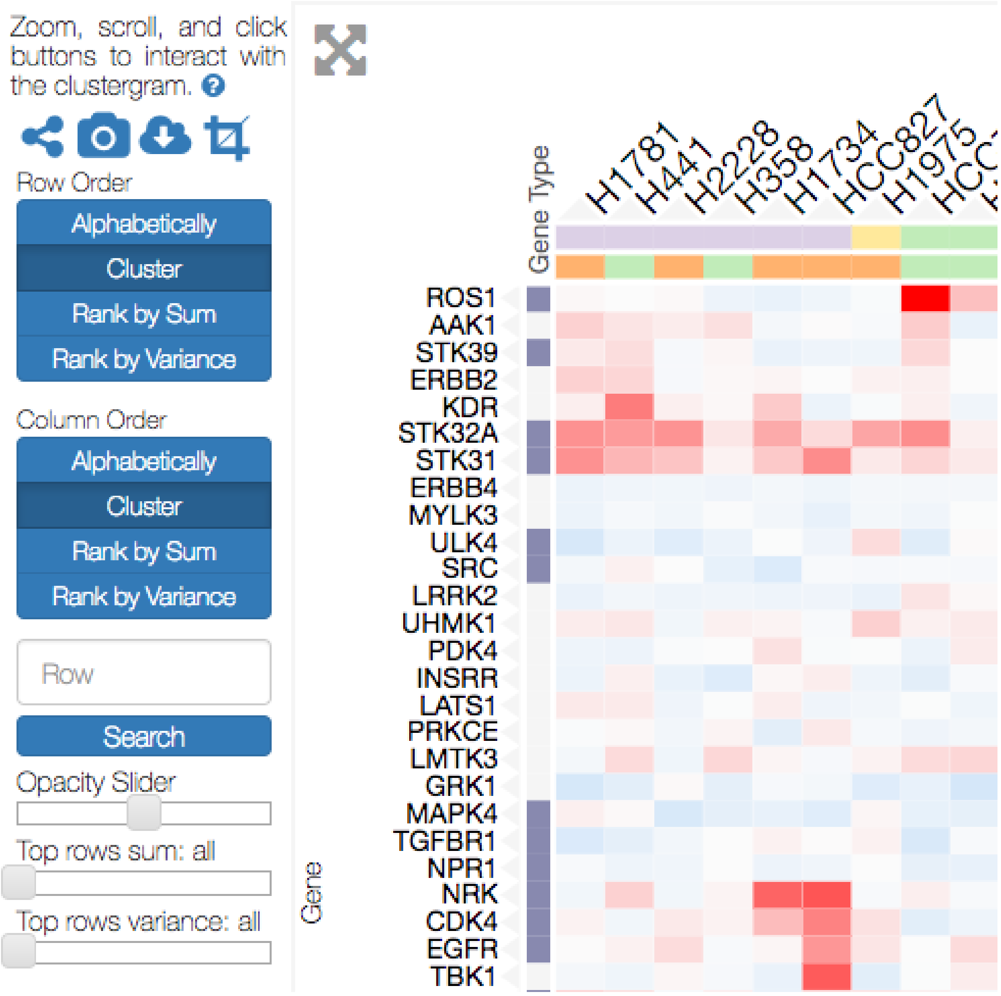
See *Clustergrammer-JS API* for information about adding callback functions to mouseover events and *Mouseover Gene Name and Description* for biology-specific mouseover behavior.

4.4.5 Sidebar Interactions

Clustergrammer visualizations have a sidebar section that contains the following interactive components:

- optional about section (see *Clustergrammer-JS API*)

- Icon-buttons: *share*, *snapshot*, *download*, *crop*
- *Row and Column Reordering Buttons*
- *Row Search Box*
- *Opacity Slider*
- *Row Filter Sliders*



Zoom, scroll, and click buttons to interact with the clustergram.

Fig. 11: The sidebar contains an optional about section and interaction elements (e.g. reordering buttons) and can be hidden by clicking the gray expand buutton (and restored by clicking the menu button).

4.4.6 Row and Column Reordering

Clustergrammer's sidebar reordering-buttons allows users to order rows and columns based on:

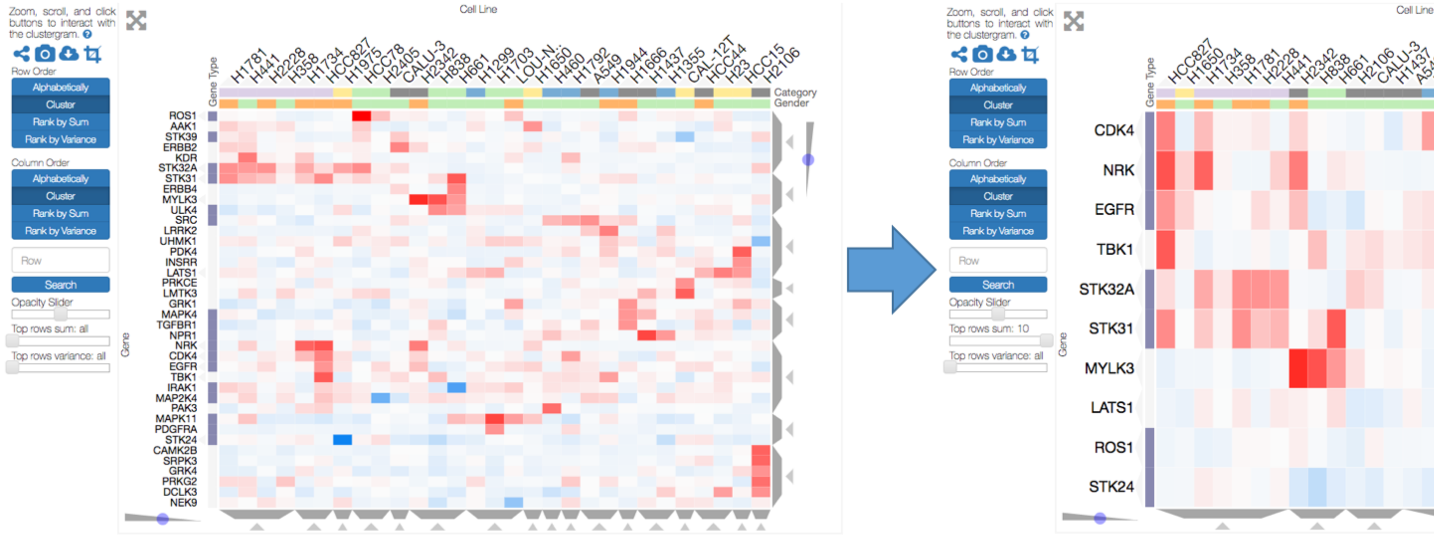
- sum or variance
- hierarchical clustering order
- label order

This can be useful for identifying broad patterns in your data. Users can also reorder their matrix based on the values in a single row/column by double-clicking the row/column labels. Similarly, users can reorder based on categorical information by double-clicking the category labels (see *Interactive Categories*). For small matrices reordering events are animated to help users visually track the effects of this transformation.

4.4.7 Interactive Dimensionality Reduction

Dimensionality reduction is a useful data analysis technique (e.g. PCA , t-SNE) that is often used to reduce the dimensionality of high-dimensional datasets (e.g. hundreds to thousands of dimensions) down to a number that can be easily be visualized (e.g. two or three dimensions). Heatmaps are capable of directly visualizing high-dimensional data, but can also benefit from dimensionality reduction.

Clustergrammer enables users to interactively perform dimensionality reduction, by filtering rows based on sum or variance, and instantaneously observe the effects of this transformation on clustering. Users can filter for the top rows based on sum or variance using the row-filter-sliders in the sidebar and choose to show the top 500, 250, 100, 50, 20, and 10 rows. This can be useful for filtering out dimensions that are not of interest (e.g. dimensions with low absolute value sum) and determining the effect of these dimensions on clustering. For instance, we may see that columns cluster in broadly the same manner when we filter out rows with low variance. Clustered views of the filtered matrices are pre-calculated by *Clustergrammer-PY*.



Zoom, scroll, and click buttons to interact with the clustergram.

Top rows sum: all

Top rows variance: all

Zoom, scroll, and click buttons to interact with the clustergram.

Top rows sum: 10

Top rows variance: all

Fig. 12: The row fitler sliders in the sidebar can be used to perform interactive dimensionality reduction. Here we are filtering for the top 10 rows based on sum.

Visualizing Dimensionality Reduction

For small matrices dimensionality reduction is animated to help the user visualize the effects this transformation. Clustergrammer employs the concept of object constancy by using animations to help the user visually follow changes to their data. Filtering out dimensions (rows) occurs in two steps: 1) filtered rows fade out, then the remaining rows rearrange themselves into their new positions (e.g. clustering order). Adding rows back in occurs in two steps: the current rows rearrange themselves into their new positions, then the new rows fade into view.

4.4.8 Interactive Dendrogram

Clustergrams typically have dendrogram trees (for both rows and columns) to depict the hierarchy of row and column clusters produced by hierarchical clustering. The height of the branches in the dendrogram depict the distance between clusters. *Clustergrammer-PY* calculates hierarchical clustering using SciPy's hierarchy clustering functions (with the default linkage type set to average, see calc_clust.py) and saves ten slices of the dendrogram taken evenly across the height of the tree.

Visualizing Dendrogram Clusters

Rather than visualize the dendrogram as a large branching tree, which uses a lot of visualization-space and is difficult to interact with, Clustergrammer uses a more compact and easy to interact with visual representation. Only a single slice of the dendrogram tree is visualized at a time as a set of non-overlapping adjacent clusters (gray trapezoids, see below). Different slices of the dendrogram can be toggled using the dendrogram-sliders (blue circles that move along a gray triangle). Moving the slider up or down shows slices taken higher or lower in the dendrogram tree, and thereby larger or smaller clusters respectively. This allows users to identify clusters at different scales.

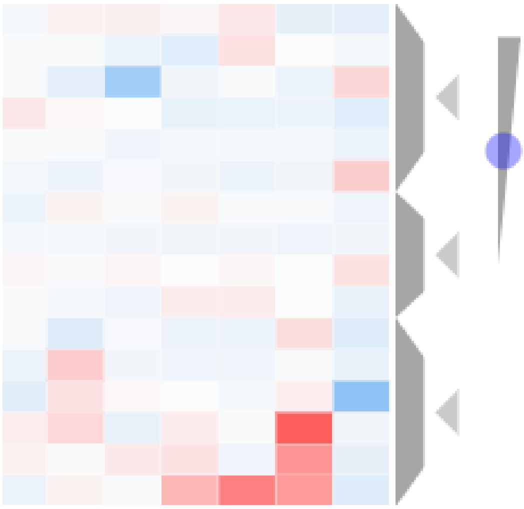


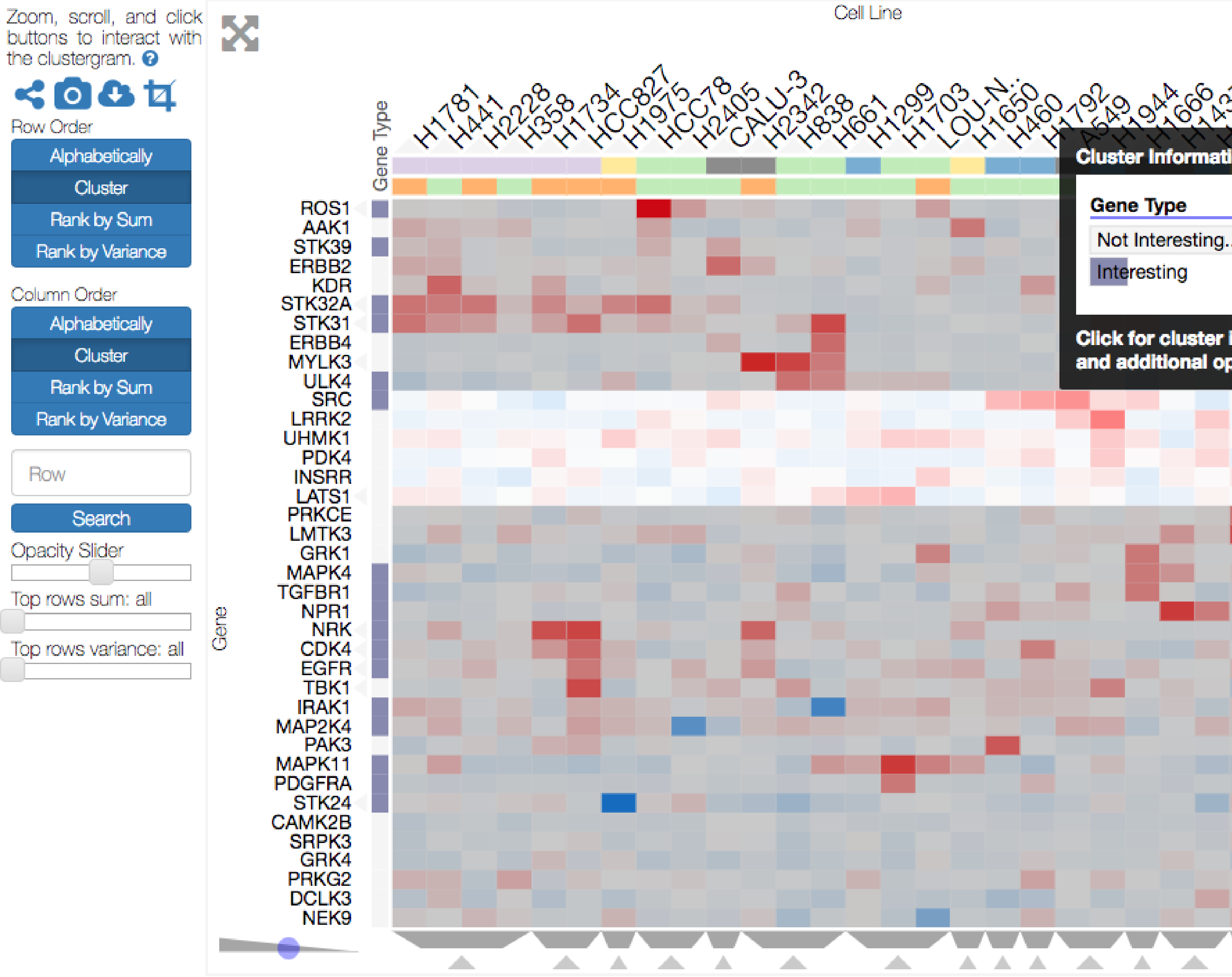
Fig. 13: A subset of the column dendrogram along with the dendrogram slider is shown above. The slider (blue circle and gray triangle) can be used to adjust dendrogram cluster sizes – move up for larger clusters and down for smaller clusters. Each dendrogram cluster has a crop button (gray triangle) above it that can be used to filter the heatmap to only show this cluster.

Interacting with Dendrogram Clusters

Dendrogram clusters are depicted as gray trapezoids, which are easy for a user to interact with (e.g. click). Mousing over a dendrogram cluster (gray trapezoid) highlights the current group of rows or columns (by adding a shadows over the rows or columns not in the cluster) and brings up a tooltip with cluster information (see screenshot below). If the rows or columns have categories, this tooltip will show a breakdown of the rows and columns into their categories, which can be useful for understanding how prior knowledge compares to clusters identified in a data-driven manner (e.g. we can ask, do columns with the same category cluster together based on the data). Clicking a dendrogram cluster brings up the same information in a pop-up window and also allows users to export the names of the rows or columns in the cluster. When a user visualizes biological gene-level data (with genes given as rows), users have the option to export their clustered genes to the enrichment analysis tool, Enrichr (see *Biology-Specific Features* for more information).

Dendrogrm Cropping

Each dendrogram cluster has a small triangular crop button (that points towards the cluster) above it that allows users to crop the matrix to only show the rows or columns in this cluster. Clicking on a dendrogram crop button filters out the rows or columns that not in the cluster, resizes the visualization to show the remaining data, and reverses the orientation of the crop button to point outwards. Clicking on the outward facing crop button undoes the cropping and restores the full matrix. For small matrices, this transformation is animated. Dendrogram cropping can be useful for focusing in on a cluster of interest and when used in combination with *Enrichrgram* to import biological information specific to your cluster of genes from Enrichr (see *Biology-Specific Features* for more information).



Zoom, scroll, and click buttons to interact with the clustergram.

Fig. 14: Mousing over a dendrogram group will highlight the selected cluster and bring up information (e.g. categories) about the cluster.

4.4.9 Interactive Categories

Prior knowledge about our system can be represented as categories in a heatmap. For instance, columns may represent cell lines and our categories may represent their tissue. Overlaying categories on our heatmap can help us understand the relationship between prior knowledge and the structures we find in our data (e.g. clusters). For instance, we may find that columns with the same category (e.g. the same tissue) cluster near each other based on the underlying data (e.g. gene expression) and we can conclude that the prior knowledge agrees with clusters identified in a data-driven manner. Similarly, we can explore how categories are re-distributed when the matrix is *reordered*. We can also use categories to overlay numerical information (e.g. duration of drug treatment of a cell line) and ask similar questions. Please see *Matrix Formats and Input/Output* for more information on how to encode categories into your data.

Row or column categories are represented by an extra column or row, respectively, of colored category-cells underneath the row or column labels (see screenshot below). Categories can be of type string or value (see *Matrix Formats and Input/Output*): each string-type category has a different color while each value-type category ahas a different opacity. The categories also have titles positioned adjacent to the category-cells.

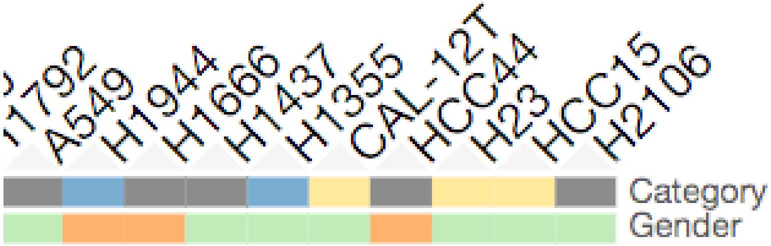


Fig. 15: A subset of column categories are shown above. In this example columns have two categories, ‘Category’ and ‘Gender’, which are depicted as colored cells under the column labels

Interacting with Categories

Mousing over a category will show the category name in a tooltip and highlight the instances of this category (while also dimming the instances of the other categories) to facilitate visualization of a specific category (see screenshot below). Double-clicking a category-title will reorder the matrix based on this category, which can be useful for getting an overview of all categories. Mousing over a dendrogram cluster will also show a breakdown of the rows/columns in a cluster based on their categories.

Updating Categories

Row categories can be updated using the *Clustergrammer-JS API*, which can be used by developers to add dynamic categories. This feature is used by *Enrichrgram* to visualize enrichment analysis results (see *Biology-Specific Features* for more information).

4.4.10 Cropping

Users can use the brush-cropping icon in the sidebar to crop the matrix to a region of interest (see screenshot below). To crop, click the crop icon and then drag the cursor to define your region of interest. Once you stop dragging the matrix will crop to show only your selected region of interest. Cropping can be undone by clicking the undo button in the sidebar (which appears after cropping). This can be useful for focusing in on a small region of your overall matrix. Cropping can be used in combination with the *Download Icon* to export a small region of the matrix or in combination with *Enrichrgram* to perform enrichment analysis on a subset of clustered genes.

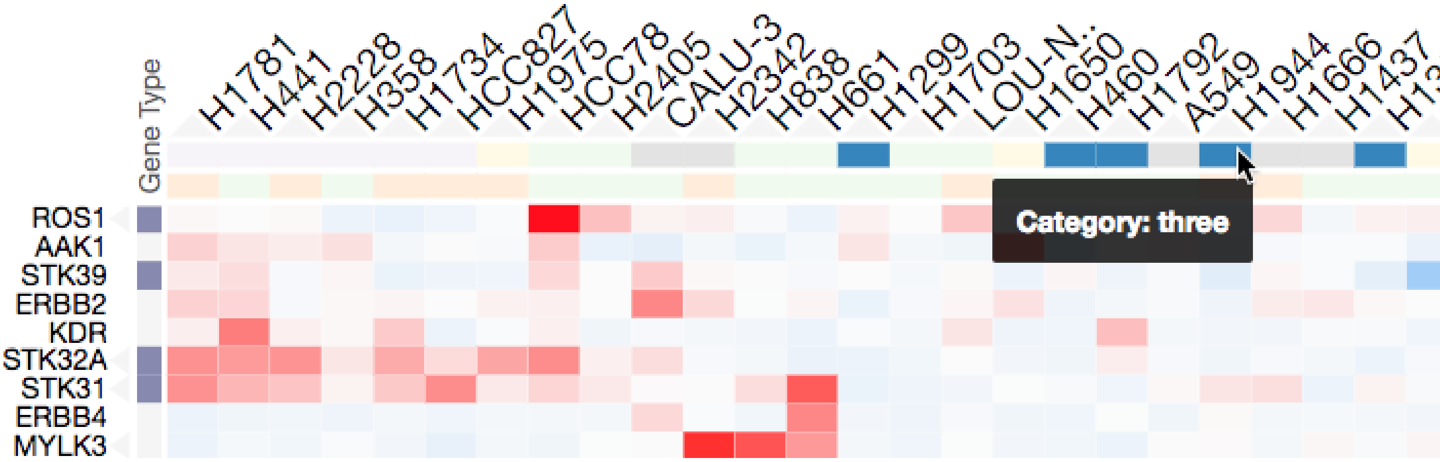

Category: three

Fig. 16: Mousing over a category brings up a tooltip with the category name and highlights instances of thie category. Shown above is an example of mousing over a column category.

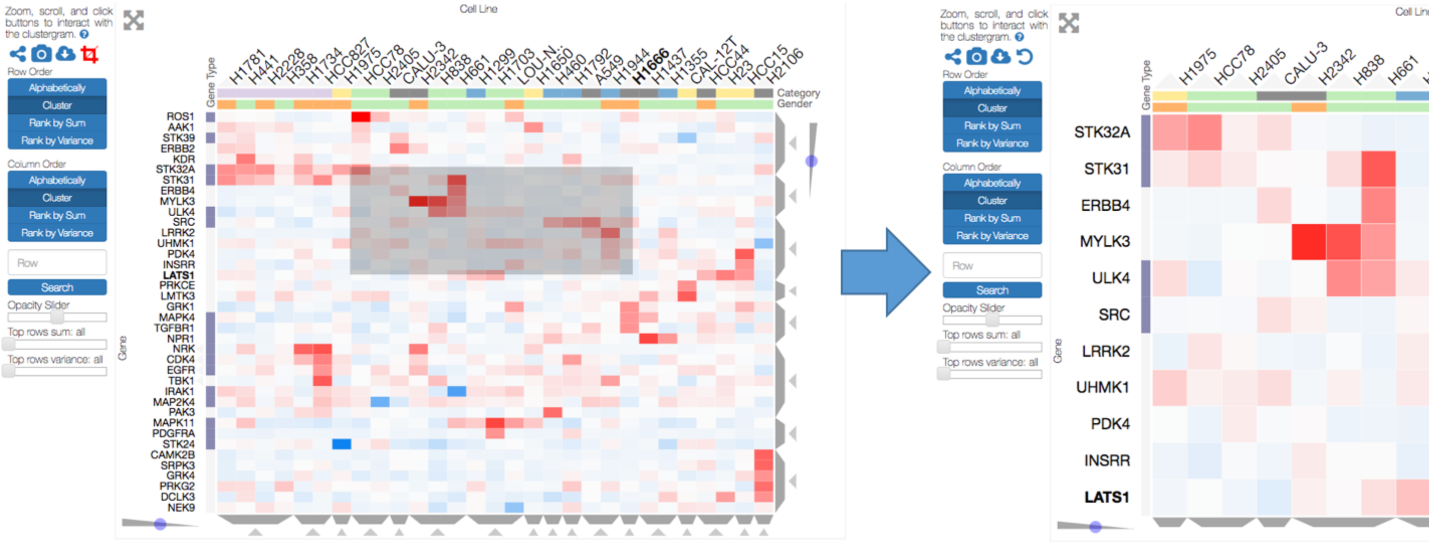

Zoom, scroll, and click buttons to interact with the clustergram.

Fig. 17: The above example shows the result of brush-cropping into a section of the heatmap. To brush crop, click the crop button (the active red icon in the sidebar on the left panel) and drag/brush your cursor over your area of interest. To undo cropping, click the undo button (circular arrow) on the right panel.

4.4.11 Download Icon

Obtaining the underlying data from a visualization for re-use and re-analysis can be a tedious task. To facilitate this common task, Clustergrammer's sidebar has a download icon, shown below, that allows users to download the matrix of data in the visualization. The downloaded data reflects the current state of the matrix, e.g. filtering, cropping, and reordering will be reflected in the downloaded data.

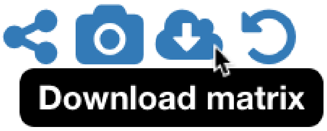


Fig. 18: Click the download icon in the sidebar to download a tab-separated file of the matrix in its current state.

4.4.12 Snapshot Icon

The snapshot icon in the sidebar allows users to take a SVG or PNG snapshot of their visualization. This snapshot will reflect the current state of the visualization (e.g. reordering, etc) as well as zooming and panning.

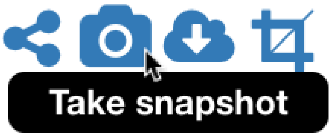


Fig. 19: Click the snapshot icon in the sidebar to take a SVG or PNG snapshot of the matrix in its curent state (including reordering, etc).

4.4.13 Opacity Slider

The opacity slider in the sidebar allows users to toggle the overall opacity levels of the heatmap. Moving the slider to the left reduces the opacity, while moving to the right increases the opacity. This can be useful for working with 'dim' matrices that can occur as a result of outlier values.

4.4.14 Row Searching

Users can search for rows in their matrix using the search box. Row search includes autocomplete and animated zooming into the matrix to display the row of interest.

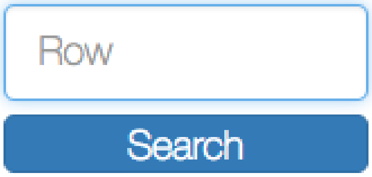


Fig. 20: Users can search for rows using the search box in the sidebar. When a row is found the matrix will zoom into the found row.

4.4.15 Expanding

Users can hide the sidebar *Sidebar Interactions* panel using the expand button to the top left of the matrix. Clicking the menu button, when expanded, returns the sidebar.

4.4.16 Sharing your Interactive Heatmap

Interactive heatmaps produced with the *Clustergrammer Web App* and the *Clustergrammer Jupyter Widget* (when notebooks are rendered through nbviewer) can easily be shared with collaborators by sharing the URL of the visualization on the web app or the notebook. Users can also click the share button on the sidebar (see *Sidebar Interactions*) sidebar to get this shareable URL.

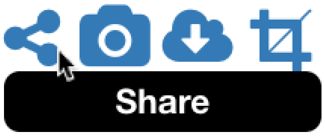


Fig. 21: Interactive heatmaps can be shared by sharing the current URL, which can be obtained from the share icon in the sidebar.

4.4.17 Biology Specific Interactions

Clustergrammer has biology specific features for working with gene-level data including:

- mouseover gene names and description look-up (using Harmonizome)
- enrichment analysis to find biological information (e.g. up-stream transcription factors) specific to your set of genes (using Enrichr)

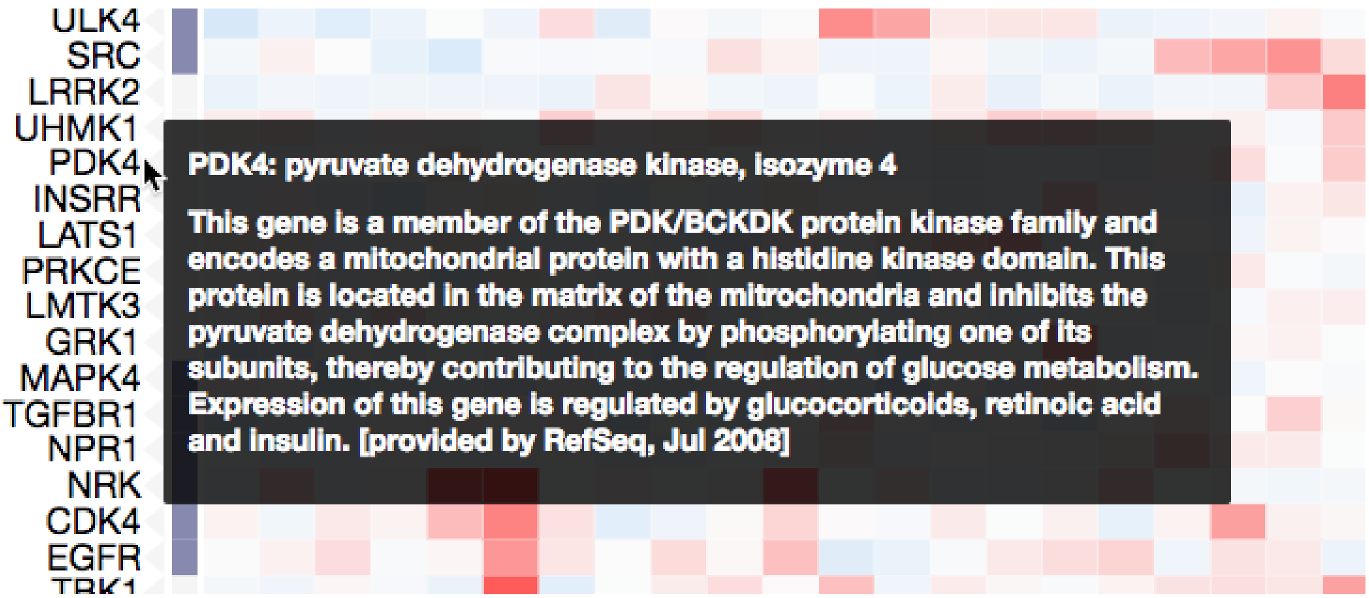
See *Biology-Specific Features* for more information.

4.5 Biology-Specific Features

Clustergrammer was built to visualize high-dimensional biological data (e.g. genome-wide expression data), but can generally be applied to any high-dimensional data (e.g. a matrix). Clustergrammer has several biology-specific features that facilitate the analysis of gene-level biological data, such as: gene-expression data, proteomics-data, etc. To take advantage of these features, genes must be given as rows. See the CCLE Explorer for examples of gene-expression data. These optional biology-specific features are available in the *Clustergrammer Web App* as well as the *Clustergrammer Jupyter Widget* and will activate if the row-names are genes.

4.5.1 Mouseover Gene Name and Description

The human genome consists of over 20,000 genes and modern high-throughput measurements are capable of making measurements across the entire genome (e.g. genome-wide expression studies). Human genes have official gene symbols (e.g. EGFR) that are frequently used to label genes in these datasets. Since no biologist can be knowledgeable about every gene in the genome a common and repetitive task for biologists is looking up the names and descriptions of genes in a dataset or visualization. To streamline this, Clustergrammer automatically gives the full name and description of a gene (provided by data aggregated through the Harmonizome) as a tooltip when a user mouses over a gene label (see screenshot below). This simple feature speeds up analysis of large gene-level datasets.



PDK4: pyruvate dehydrogenase kinase, isozyme 4

This gene is a member of the PDK/BCKDK protein kinase family and encodes a mitochondrial protein with a histidine kinase domain. This protein is located in the matrix of the mitochondria and inhibits the pyruvate dehydrogenase complex by phosphorylating one of its subunits, thereby contributing to the regulation of glucose metabolism. Expression of this gene is regulated by glucocorticoids, retinoic acid and insulin. [provided by RefSeq, Jul 2008]

Fig. 22: Mousing over a gene name row brings up the full gene name and description (provided by data aggregated through the Harmonizome).

The JavaScript file hzome_functions.js provides this functionality by utilizing Harmonizome's RESTful API to obtain gene names and descriptions on mouseover events. hzome_functions.js is passed to *Clustergrammer-JS* as a callback function. See load_clustergram.js for an example use case. Mouseover callback functions can be used by developers to extend functionality for other domain-specific problems.

4.5.2 Enrichment Analysis

The field of biology has amassed an enormous amount of information about the genes in living organisms such as: function, disease-association, up-stream regulators, protein-level binding partners, etc. Integration of this information can help biologists understand patterns in their data. For instance, enrichment analysis a popular method to identify biological information specific to a list of genes – e.g. a biologist may use enrichment analysis to identify up-stream regulatory transcription factors that specifically target their measured set of up-regulated genes and thereby form hypotheses about potential up-stream regulators in their system.

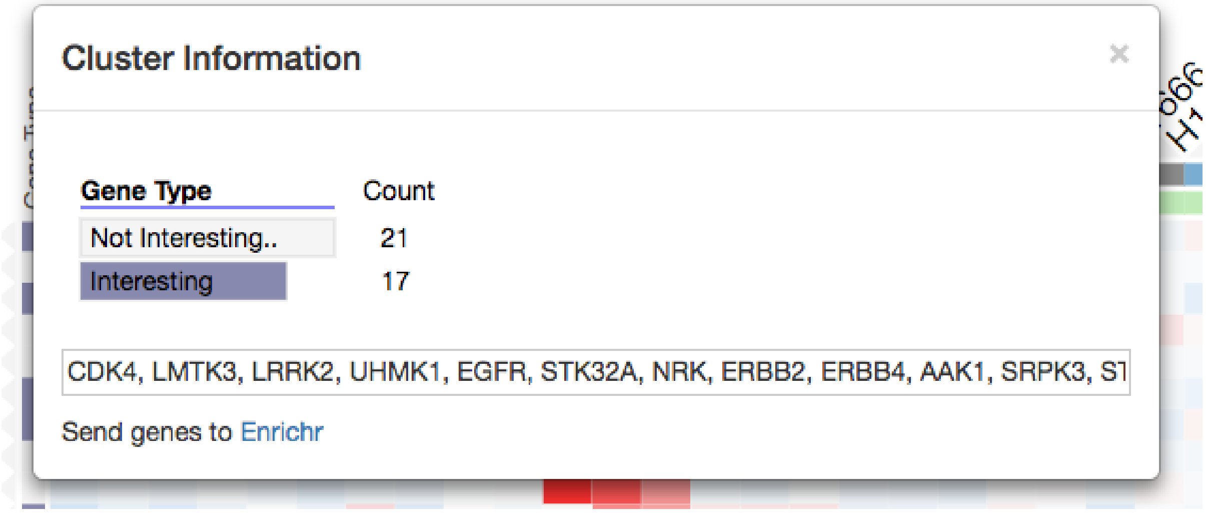
Export to Enrichr

When a user visualizes a matrix with genes as rows, Clustergrammer automatically enables integration with the enrichment analysis tool Enrichr. Users can export a set of clustered genes to Enrichr using the interactive dendrogram (see screenshot) or import enriched terms into the visualization using *Enrichrgram*.

Enrichrgram

Users can also import biological information about their genes directly into the visualization (see screenshot below). Simply click the Enrichr-logo to the top-left of the heatmap to bring up a list of libraries from Enrichr, then click on a library to obtain enriched terms for your genes of interest. For instance, clicking on 'ChEA 2016' will enrich for up-stream transcription factors. The enriched terms are shown as row categories, which enables users to see which genes are associated with each term. The row-category-titles give the enriched term name and the red-bars represent the significance of the enrichment (see Enrichr combined score). Users can run enrichment analysis on specific clusters of genes by filtering the matrix to only show their genes of interest: e.g. use the dendrogram crop buttons or brush-crop to select a subset of genes for analysis.

Enrichrgram.js provides this functionality and works with the *Clustergrammer-JS* API to depict enriched terms and their associated genes as row categories. The update-row-category functionality can be extended by developers for



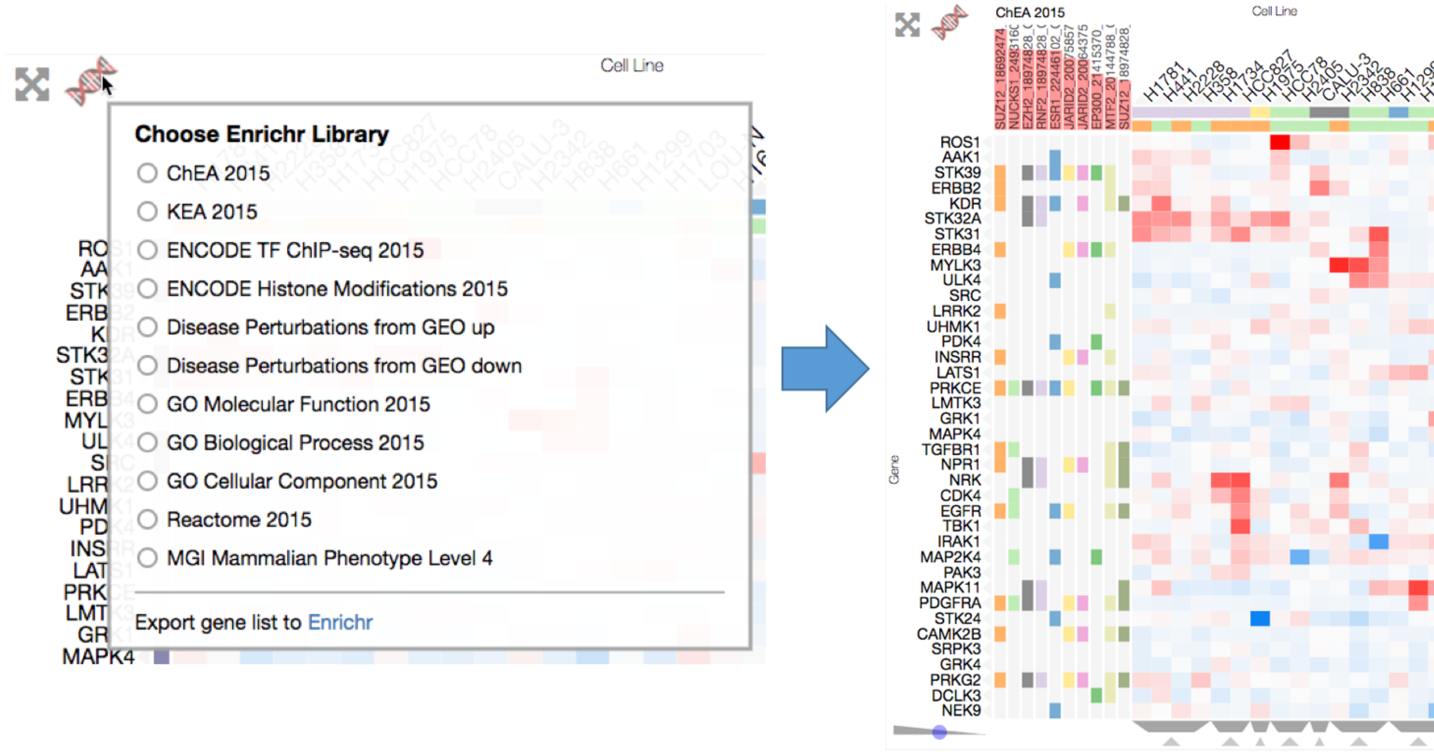
Cluster Information ×

Gene Type	Count
Not Interesting..	21
Interesting	17

CDK4, LMTK3, LRRK2, UHMK1, EGFR, STK32A, NRK, ERBB2, ERBB4, AAK1, SRPK3, S1

Send genes to Enrichr

Fig. 23: Clicking a row dendrogram cluster opens a modal with cluster information, row names, and a 'Send genes to Enrichr' link that allows users to export their gene list (e.g. cluster of row-genes) to Enrichr.



Choose Enrichr Library

- ○ ChEA 2015
- ○ KEA 2015
- ○ ENCODE TF ChIP-seq 2015
- ○ ENCODE Histone Modifications 2015
- ○ Disease Perturbations from GEO up
- ○ Disease Perturbations from GEO down
- ○ GO Molecular Function 2015
- ○ GO Biological Process 2015
- ○ GO Cellular Component 2015
- ○ Reactome 2015
- ○ MGI Mammalian Phenotype Level 4

Export gene list to Enrichr

Fig. 24: Users can perform enrichment analysis to find biological information specific to their genes (e.g. a cluster of genes). Users can select from several enrichment libraries and the top 10 enriched terms will be shown as rows categories. The combined-scores for the enriched terms will be shown as red bars behind the row category titles.

other domain specific problems.

4.6 Case Studies and Examples

Clustergrammer was built to visualize biological data but is generally applicable for visualizing high-dimensional data. Below are links to several example case studies and examples using Clustergrammer:

4.6.1 Cancer Cell Line Encyclopedia Gene Expression Data

The Cancer Cell Line Encyclopedia (CCLE) is a publicly available project that characterized (e.g. genetic characterization) over one thousand cancer cell lines. We used Clustergrammer to re-analyze and visualize CCLE's gene expression data in the CCLE Explorer. The CCLE Explorer allows users to explore the CCLE by tissue type and visualize the most commonly differentially expressed genes for each tissue type as an interactive heatmap.

4.6.2 Zika Virus RNA-seq Data Visualization

Clustergrammer was used to visualize the results of an RNA-Seq data analysis pipeline within a Jupyter notebook: An open RNA-Seq data analysis pipeline tutorial with an example of reprocessing data from a recent Zika virus study (Wang et al.).

4.6.3 Single Cell RNA-seq Data Visualization

Clustergrammer was used to visualize published single-cell gene expression data: Single Cell RNA-seq Data Visualization (Olsson et al.). The visualization was produced using an excel file provided alongside the figures.

4.6.4 Machine Learning Datasets

Clustergrammer was used to visualize two widely used labeled machine learning datasets:

- Iris flower dataset
- MNIST Handwritten Digit Dataset

These examples demonstrate the generality of heatmap visualizations and enable users to interactively explore familiar datasets.

4.7 Matrix Formats and Input/Output

Clustergrammer takes as input either:

- a tab-separated matrix file
- a Pandas DataFrame (using *Clustergrammer-PY*)

The tab-separated matrix file can take several formats shown below, which can include row/column categories and name/category titles. In call cases, row and column names must be unique. Users are encouraged to arrange their matrix with data-points as columns and dimensions as rows, which enables users to take advantage of Clustergrammer's *Interactive Dimensionality Reduction*.

The front-end *Clustergrammer-JS* library can visualize matrices up to ~500,000 to ~1,000,000 cells large and is also optimized to visualize matrices with more rows than columns. However very large matrices may take a long time to cluster using the *Clustergrammer-PY* library.

4.7.1 Simple Matrix Format

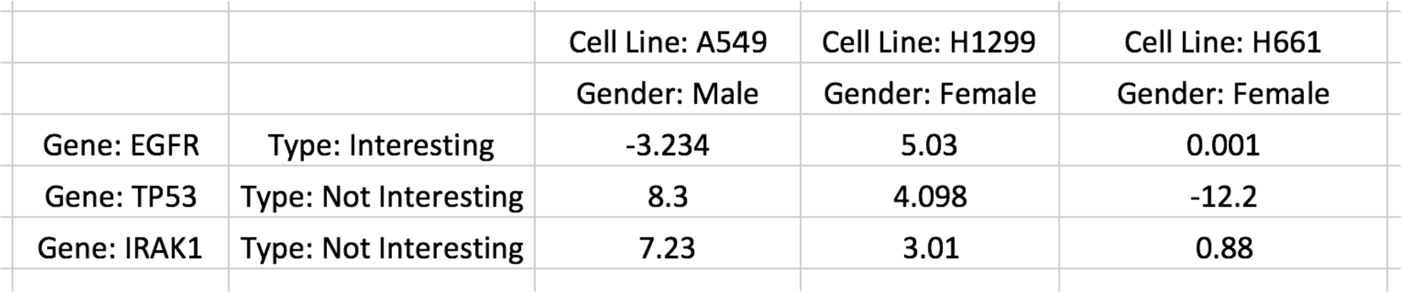
The simplest tab-separated file format is shown here:

```
      Col-A    Col-B    Col-C
Row-A 0.0      -0.1     1.0
Row-B 3.0      0.0      8.0
Row-C 0.2      0.1      2.5
```

The first row gives the column names and starts with a blank tab. All other rows start with the row name followed by the row data. Row and column titles can be added by prefixing each row or column name with `'Title:  '` (not shown in this example). See example_tsv.txt for an example of this matrix format.

4.7.2 Simple-Category Matrix Format

Row and column categories can be included in two ways. The first, simple-category format, is shown below:



		Cell Line: A549	Cell Line: H1299	Cell Line: H661
		Gender: Male	Gender: Female	Gender: Female
Gene: EGFR	Type: Interesting	-3.234	5.03	0.001
Gene: TP53	Type: Not Interesting	8.3	4.098	-12.2
Gene: IRAK1	Type: Not Interesting	7.23	3.01	0.88

Fig. 25: A matrix with row and column categories in 'simple' format.

This simple-category format allows users to encode column categories as a extra rows underneath column labels and row categories as an extra columns next to row labels. The above screenshot of an Excel spreadsheet shows a single row category being added as an additional column of strings (e.g. `Type:  Interesting`) and a single column category being added as an additional row of strings (e.g. `Gender:  Male`). Up to 15 categories can be added in a similar manner. Titles for row or column names or categories can be added by prefixing each string with `'Title: '` (note the space after the colon). For example the title of the column names is `Cell Line` and the title of the row categories is `Gender`. See rc_two_cats.txt for an example of this matrix format. Titles, if given, will be shown as labels above row/column names or adjacent to row/column categories.

4.7.3 Tuple-Category Matrix Format

Row/column names and categories can also be encoded as Python tuples as shown below:

```
        ('Cell Line: A549', 'Gender: Male')      ('Cell Line: H1299', 'Gender: Female
↪')  ('Cell Line: H661', 'Gender: Female')
('Gene: EGFR','Type: Interesting')      -3.234  5.03    0.001
('Gene: TP53','Type: Not Interesting')  8.3     4.098   -12.2
('Gene: IRAK','Type: Not Interesting')  7.23    3.01    0.88
```

This tuple-category format is easier to work with in Python and can be imported/exported into Pandas DataFrames and as tab-separated files. Note that titles have been added to row/column names and categories as discussed above. See tuple_cats.txt for an example of this matrix format.

4.7.4 Category Types: String and Value

Row and column categories can be of type: string or value. If categories are given as strings (e.g. containing letters) then categories will be depicted using colors. If categories are of type value (e.g. all categories contain only numbers) then value-categories will be depicted using color and opacity (gray for positive and orange for negative).

Value-based categories can be useful for adding data to your visualization (e.g. drug-dosage value) that you would like to compare to your other dimensions, but would not like to influence your clustering. Value-based and String-based categories can also be used to reorder your matrix by double-clicking their labels (see *Interactive Categories*).

4.7.5 Matrix File Examples

Several example tab-separated matrix files can be found in example matrix files.

4.7.6 Matrix Input/Output to Clustergrammer.py

Clustergrammer.py can load a matrix directly from a file or from a Pandas DataFrame as well as export to a file or Pandas DataFrame:

```
# initialize Network object
from clustergrammer import Network
net = Network()

# load matrix from file or DataFrame
####################################

# load data from file
net.load_file('your_matrix.txt')

# load data from DataFrame, df
net.load_df(df)

# export matrix
###############

# write matrix to tab separated file
net.write_matrix_to_tsv(filename)

# export data to Pandas DataFrame
df_export = net.export_df()
```

For more information about Clustergrammer.py and its API see *Clustergrammer-PY* section.

4.8 Web-Development with Clustergrammer

Clustergrammer can be used by developers to add interactive heatmap visualizations to their web pages and web applications (see *App Integration Examples*).

4.8.1 Embedding Clustergrammer

The Clustergrammer web app can be used to produce visualizations that are embedded into another page using an IFrame - see below:

```
<iframe id="iframe_preview" src="http://amp.pharm.mssm.edu/clustergrammer/viz/
↪5734a7399fee36034aeb787e/rc_two_cats.txt" frameborder="0"></iframe>
```

Users can obtain a permanent link to their visualization by manually uploading their data using the upload section of Clustergrammer-Web's homepage and copying the URL to the full-screen version of their visualization. Alternatively users can programmatically upload their data using the *Clustergrammer-Web API* and receive their permanent links through the API.

4.8.2 Adding Clustergrammer to a Page

The *Clustergrammer-JS* library can be used to generate an interactive visualization in your webpage. Simply include the *Clustergrammer-JS* script in your page and load the pre-calculated *visualization-JSON* to generate a visualization. Alternatively, *Clustergrammer-JS* can also be included as a node module.

Clustergrammer can be used to generate interactive visualizations for your own web application by: using the *Clustergrammer-JS* library on your site, or embedding a visualization provided by the *Clustergrammer Web App*.

The easiest way to generate a visualize of your own data on a webpage is to:

1. Follow the *Python Workflow Examples* to cluster your matrix and generate the *Visualization-JSON*
2. Then use the *Example Pages* as templates to build a site with your visualization

These examples require Clustergrammer's JavaScript and Python libraries:

1. the front-end *Clustergrammer-JS* JavaScript library makes the interactive visualization
2. the back-end *Clustergrammer-PY* Python library clusters a matrix of data and makes the JSON for the front-end

These libraries can be installed npm, `npm install Clustergrammer`, and pip, `pip install clustergrammer`, respectively.

4.9 Clustergrammer-JS

Clustergrammer-JS is the front-end JavaScript library that builds the interactive heatmap visualization in SVG using the visualization library D3.js

4.9.1 Clustergrammer-JS Dependencies

- D3.js
- JQuery
- Underscore

4.9.2 Installation

Clustergrammer.js can be installed using node package manager (npm package) with the following:

```
npm install clustergrammer
```

or the source code and library, `clustergrammer.js`, can be obtained from the Clustergrammer GitHub repo.

4.9.3 JavaScript Workflow Example

This workflow shows how to make a visualization using a JSON produced by Clustergrammer.py

```
// load visualization JSON to network_data
var args = {
  'root': '#id_of_container',
  'network_data': 'network_data'
}

var cgm = Clustergrammer(args);
```

The `id` of the container where the visualization will be made is passed as `root` (this root container must be made by the user). The *Visualization-JSON* contains the information necessary to make the visualization and is passed as `network_data`. See the *Clustergrammer-JS API* for additional arguments that can be passed to Clustergrammer.js.

4.9.4 Example Pages

The Clustergrammer GitHub repo contains several example pages demonstrating how to make a webpage with a Clustergrammer heatmap. The page index.html and corresponding script load_clustergram.js show how to make a full-screen resizable visualization. The page multiple_clust.html and corresponding script load_multiple_clustergrams.js show how to visualize multiple clustergrams on one page. Note that each heatmap requires its own container.

4.9.5 Clustergrammer-JS API

class Clustergrammer(*args*)

The Clustergrammer JavaScript object takes the `args` object and produces a visualization on the page.

This `args` object has two required arguments, `network_data` and `root`:

`Clustergrammer.args.`**`network_data`**

This required attribute is where the visualization JSON should be passed as a JavaScript object.

`Clustergrammer.args.`**`root`**

This required attribute is the `id` (passed as a string) of the container where Clustergrammer will be built. Each Clustergrammer visualization in a page should be passed a unique `id`.

`Clustergrammer.args.`**`about`**

This attribute is a string (which can include HTML) that will produce a small about section at the top of the sidebar. This can be used to provide a quick description about the data or visualization.

`Clustergrammer.args.`**`row_tip_callback`**

Users can pass a callback function that will run when mousing over row labels.

`Clustergrammer.args.`**`col_tip_callback`**

Users can pass a callback function that will run when mousing over col labels.

Clustergrammer.args.**tile_tip_callback**

Users can pass a callback function that will run when mousing over a matrix-cell (e.g. matrix tile).

Clustergrammer.args.**dendro_callback**

Users can pass a callback function that will run when mousing over a dendrogram cluster (e.g. gray trapezoid)

Clustergrammer.args.**matrix_update_callback**

Users can pass a callback function that will run anytime the matrix has been updated, for instance when filtering/un-filtering, cropping, etc.

Clustergrammer.args.**sidebar_width:**

Users can modify the width of the sidebar by specifying the width of the sidebar in pixels as a number.

Clustergrammer.args.**ini_view**

Users can initialize the 'view' of their matrix, e.g. a initialize the matrix at a particular row filtering level.

Clustergrammer's attributes and functions are listed below:

Clustergrammer.**params**

The Clustergrammer parameters object, which contains all the parameters necessary to generate the visualization.

Clustergrammer.**update_cats**(*row_data*)

Update the visualization row categories.

Arguments

- **row_data** – Row category data.

Clustergrammer.**reset_cats**()

Reset the row categories to their original state.

Clustergrammer.**resize_viz:**()

Call this function to resize the visualization to fit in its resized container (if the user has resized the container).

Clustergrammer.**d3_tip_custom**()

Generate a D3 tooltip for SVG elements.

Clustergrammer.**update_view**(*filter_type*, *inst_state*)

Update the heatmap with a specified row filter 'view'.

Arguments

- **filter_type** – The available filter types sum or variance: e.g. N_row_sum, N_row_var
- **inst_state** – The value of the row filter, e.g. 500

Clustergrammer.**filter_viz_using_names**(*names*)

Update the visualization to show the row and column names specified in the names object.

Arguments

- **names** – Object with `row` and `col` attributes that specify the row and column names that will be visible after updating.

Clustergrammer.**filter_viz_using_names**(*nodes*)

Update the visualization to show the row and column names specified in the nodes object.

Arguments

- **names** – Object with `row` and `col` attributes that specify the row and column nodes that will be visible after updating.

`Clustergrammer.`**`zoom`**`(`*pan_x*, *pan_y*, *zoom*`)`

Zoom and pan into the visualization.

Arguments

- **pan_x** – Panning in the *x* direction
- **pan_y** – Panning in the *y* direction
- **zoom** – The zoom level applied to the visualization.

`Clustergrammer.`**`export_matrix`**`()`

Save the current matrix (e.g. after cropping) as a tab-separated-file.

4.9.6 Visualization-JSON

The visualization-JSON is calculated by *Clustergrammer-PY* and encodes everything needed for the front-end Clustergrammer-JS to produce the visualization. The visualization-JSON format is described here (see clustergrammer_example.json for an example file). An overview of the format is shown below (note that the group arrays are not shown):

```
{
  "row_nodes":[
     {
      "name": "ATF7",
      "clust": 67,
      "rank": 66,
      "rankvar": 10,
      "group": []
    }
  ],
  "col_nodes":[
    {
      "name": "Col-0",
      "clust": 4,
      "rank": 10,
      "rankvar": 120,
      "group": []
    }
  ],
  "links":[
    {
      "source": 0,
      "target": 0,
      "value": 0.023
    }
  ]
}
```

Optional 'views' of the matrix (e.g. row-filtered views) are encoded into the `views` attribute at the base level of the object. These views are used to store filtered version of the matrix. Only the row and column names are stored in these views since all views share the same matrix cells. The view attributes are stored in the view object (e.g. `N_row_sum`):

```
"views":[
  {
```

(continues on next page)

(continued from previous page)

```
    "N_row_sum": "all",
    "dist": "cos",
    "nodes":{
      "row_nodes": [],
      "col_nodes": []
    }
  }
```

There are three required properties for the Visualization-JSON: `row_nodes`, `col_nodes`, and `links`. Each of these properties is an array of objects and these objects are discussed below

Nodes

`row_nodes` and `col_nodes` objects are required to have three properties: `name`, `clust`, `rank`. `name` specifies the name given to the row or column. `clust` and `rank` give the ordering of the row or column in the clustergram. Two optional properties are `group` and `value`. `group` is an array that contains group-membership of the row/column at different dendrogram distance cutoffs and is necessary for displaying a dendrogram. If nodes have the `value` property, then semi-transparent bars will be made behind the labels to represent this value.

Links

`links` have three properties: `source`, `target`, and `value`. `source` and `target` give the integer value of the row and column of the matrix-cell in the visualization. `value` specifies the opacity and color of the matrix-cell, where positive/negative values results in red/blue matrix-cells in the visualization. The optional properties `value_up` and `value_dn` allow the use to have a split matrix-cell that has an up-triangle and down-triangle.

Users can also generate the visualization-JSON using their own scripts as long as they adhere to the above format.

4.9.7 Clustergrammer-JS Development

Clustergrammer-JS' source code can be found in the Clustergrammer GitHub repo. The Clustergrammer-JS library is utilized by the *Clustergrammer Web App* and the *Clustergrammer Jupyter Widget*. Clustergrammer-JS is built with Webpack Module Bundler from the source files in the src directory.

Please *Contact* Nicolas Fernandez or Avi Ma'ayan with questions or use the GitHub issues feature to raise an issue.

4.10 Clustergrammer-PY

Clustergrammer-PY is the back-end Python library that is used to hierarchically cluster the data and generate the *Visualization-JSON* for the front-end *Clustergrammer-JS* visualization library. Clustergrammer-PY is compatible with Python 2 and 3.

4.10.1 Clustergrammer-PY Dependencies

- Numpy
- SciPy
- Pandas

4.10.2 Installation

Clustergrammer-PY can be installed using pip (package index) with the following:

```
pip install clustergrammer
```

or the source code can be obtained from the GitHub repo.

4.10.3 Python Workflow Examples

This workflow shows how to cluster a matrix of data from a file (see *Matrix Formats and Input/Output*) and generate a *Visualization-JSON* (for use by *Clustergrammer-JS*):

```
# make network object and load file
from clustergrammer import Network
net = Network()
net.load_file('your_matrix.txt')

# calculate clustering using default parameters
net.make_clust()

# save visualization JSON to file for use by front-end
net.write_json_to_file('viz', 'mult_view.json')
```

The file `mult_view.json` will be loaded by the front-end and used to build the interactive visualization. See make_clustergrammer.py for an additional example.

Clustergrammer can also load data from a Pandas DataFrame and perform normalization and filtering. In this example we will load data from a DataFrame, normalize the rows, and filter the columns:

```
# make network object and load DataFrame, df
net = Network()
net.load_df(df)

# Z-score normalize the rows
net.normalize(axis='row', norm_type='zscore', keep_orig=True)

# filter for the top 100 columns based on their absolute value sum
net.filter_N_top('col', 100, 'sum')

# cluster using default parameters
net.make_clust()

# save visualization JSON to file for use by front-end
net.write_json_to_file('viz', 'mult_view.json')
```

Note that filtering done on the `Network` object before clustering is permanent, unlike the filtering done within `make_clust` which can be toggled on and off in the front-end visualization. The `keep_orig` parameter in the `normalize` function allows us to show the un-normalized when a user mouses over a matrix-cell in the visualization. See the *Clustergrammer-PY API* documentation below for more information.

4.10.4 Clustergrammer-PY API

Clustergrammer-PY generates a Network object (see Network class definition), which is used to load a matrix (e.g. from a Pandas DataFrame), optionally normalize or filter the matrix, cluster the matrix, and finally generate the visualization JSON for the front-end Clustergrammer.js.

When a matrix is loaded into an instance of `Network` (e.g. `net.load_file('your_file.txt')`) it is stored in the data, `dat`, attribute. Normalization and filtering will permanently modify the `dat` representation of the matrix. When the matrix is clustered (by calling `make_clust`) this produces the *Visualization-JSON*, which is stored in the `viz` attribute. This JSON can then be exported as a string using `net.export_net_json('viz')` or saved to a file using `net.write_json_to_file('viz', filename)`.

The function `make_clust` calculates hierarchical clustering of your data and hierarchical clustering of successive-row-filtered versions of your data. These alternate filtered-views are stored as `views` within the *Visualization-JSON*.

class `clustergrammer_py.`**`Network`**

version 1.2.2

Clustergrammer.py takes a matrix as input (either from a file of a Pandas DataFrame), normalizes/filters, hierarchically clusters, and produces the *Visualization-JSON* for *Clustergrammer-JS*.

Networks have two states:

1. the data state, where they are stored as a matrix and nodes
2. the viz state where they are stored as viz.links, viz.row_nodes, and viz.col_nodes.

The goal is to start in a data-state and produce a viz-state of the network that will be used as input to clustergram.js.

`dat_to_df`()

Export Pandas DataFrams (will be deprecated).

`df_to_dat`(*df*)

Load Pandas DataFrame (will be deprecated).

`enrichr`(*req_type*, *gene_list=None*, *lib=None*, *list_id=None*, *max_terms=None*)

Under development, get enrichment results from Enrichr and add them to clustergram

`export_df`()

Export Pandas DataFrame/

`export_net_json`(*net_type='viz'*, *indent='no-indent'*)

Export dat or viz JSON.

`filter_N_top`(*inst_rc*, *N_top*, *rank_type='sum'*)

Filter the matrix rows or columns based on sum/variance, and only keep the top N.

`filter_sum`(*inst_rc*, *threshold*, *take_abs=True*)

Filter a network's rows or columns based on the sum across rows or columns.

`filter_threshold`(*inst_rc*, *threshold*, *num_occur=1*)

Filter the matrix rows or columns based on num_occur values being above a threshold (in absolute value).

`load_data_file_to_net`(*filename*)

Load Clustergrammer's dat format (saved as json).

`load_df`(*df*)

Load Pandas DataFrame.

`load_file`(*filename*)

Load tsv file.

`load_stdin`()

Load stdin tsv formatted string.

`load_tsv_to_net`(*file_buffer*, *filename=None*)

This will load a tsv matrix file buffer, this is exposed so that it will be possible to load data without having to read from a file.

load_vect_post_to_net(*vect_post*)
Load data in the vector format JSON.

make_clust(*dist_type='cosine', run_clustering=True, dendro=True, views=['N_row_sum', 'N_row_var'], linkage_type='average', sim_mat=False, filter_sim=0.1, calc_cat_pval=False, run_enrichr=None*)
The main function performs hierarchical clustering, optionally generates filtered views (e.g. row filtered views), and generates the :*visualization_json*.

normalize(*df=None, norm_type='zscore', axis='row', keep_orig=False*)
Normalize the matrix rows or columns using Z-score (zscore) or Quantile Normalization (qn).

produce_view(*requested_view=None*)
This function is under development and will produce a single view on demand.

reset()
This re-initializes the Network object.

swap_nan_for_zero()
Swaps all NaN (numpy NaN) instances for zero.

widget()
Export viz json, for use with clustergrammer_widget.

write_json_to_file(*net_type, filename, indent='no-indent'*)
Save dat or viz as a JSON to file.

write_matrix_to_tsv(*filename=None, df=None*)
Export data-matrix to file.

4.10.5 Clustergrammer-PY Development

Clustergrammer-PY's source code can be found in the clustergrammer-py GitHub repo. The Clustergrammer-PY library is utilized by the *Clustergrammer Web App* and the *Clustergrammer Jupyter Widget*.

Please *Contact* Nicolas Fernandez or Avi Ma'ayan with questions or use the GitHub issues feature to raise an issue.

4.11 App Integration Examples

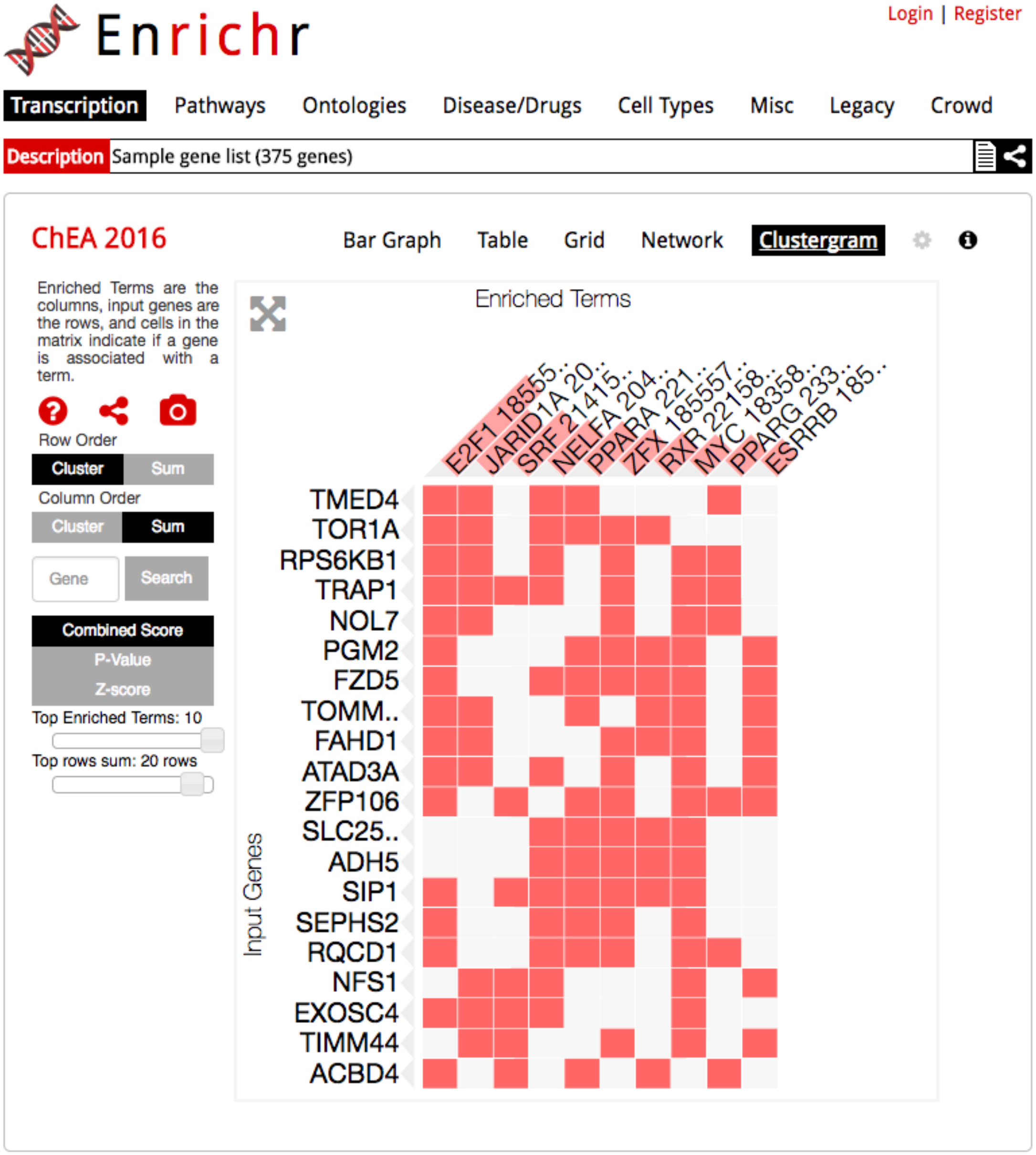
Clustergrammer can be integrated into web applications to dynamically produce interactive visualizations – see *Web-Development with Clustergrammer* for information. Clustergrammer is currently being utilized to visualize data for the following Ma'ayan lab web applications:

4.11.1 Enrichr

The enrichment analysis tool, Enrichr, uses Clustergrammer to produce dynamic heatmaps of enriched terms as columns and user input genes as rows. This helps users understand the relationships between their input genes and the returned enriched terms.

4.11.2 GEN3VA

The gene signature analysis and visualization tool, GEN3VA, uses Clustergrammer's core libraries, *Clustergrammer-JS* and *Clustergrammer-PY*, to dynamically visualize collections of gene expression signatures collected by users from GEO as interactive heatmaps. GEN3VA also uses Clustergrammer to visualize enrichment analysis results (obtained from Enrichr) and perturbations that reverse or mimic gene expression signatures (obtained from L1000CDS2)



Description Sample gene list (375 genes)

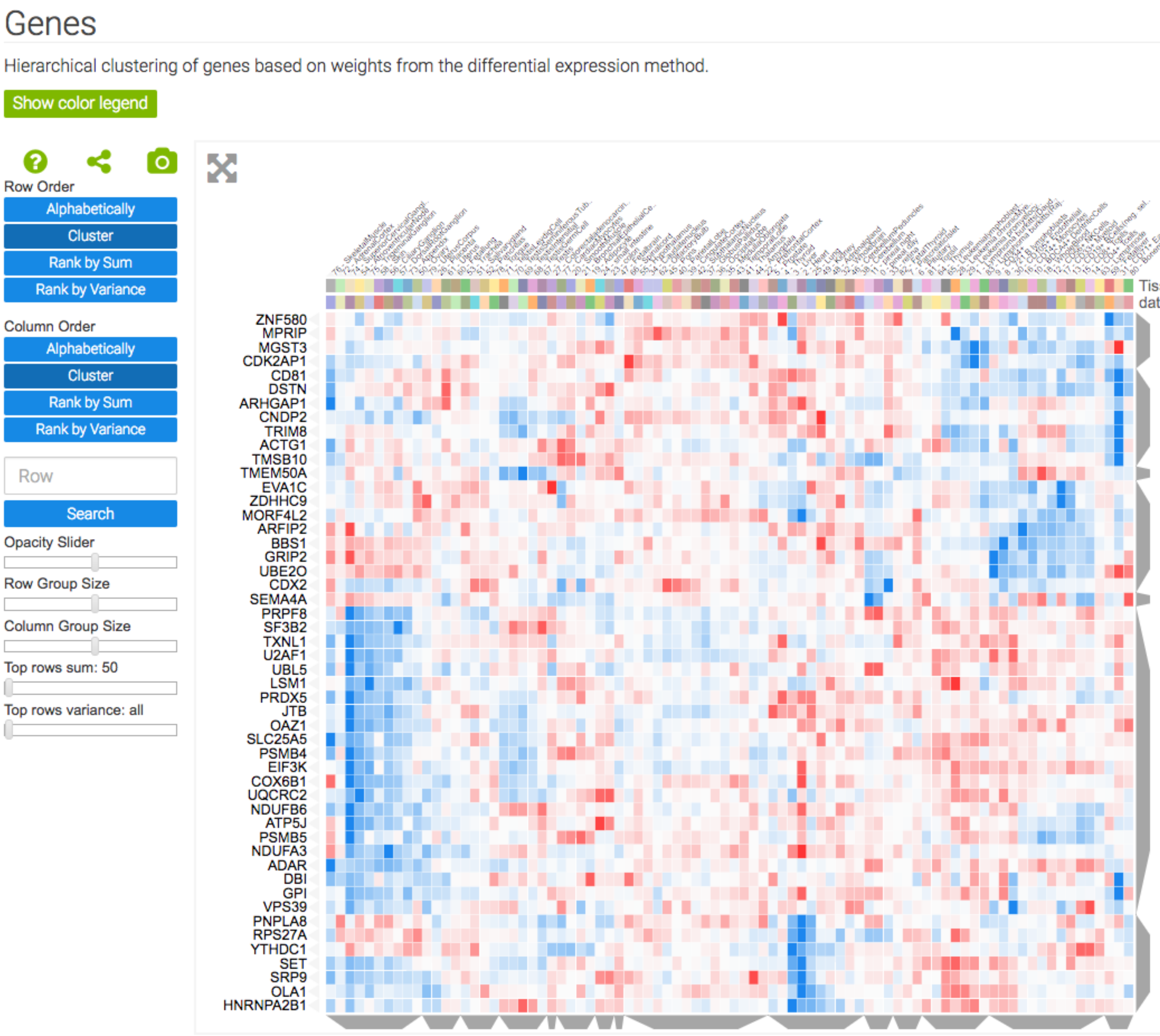
Enriched Terms are the columns, input genes are the rows, and cells in the matrix indicate if a gene is associated with a term.

Top Enriched Terms: 10

Top rows sum: 20 rows

	E2F1 18555..	JARID1A 20..	SRF 21415..	NELFA 204..	PPARA 221..	ZFX 185557..	RXR 22158..	MYC 18358..	PPARG 233..	ESRRB 185..
TMED4	🟥	🟥		🟥	🟥				🟥	
TOR1A	🟥	🟥		🟥	🟥	🟥	🟥			
RPS6KB1	🟥	🟥		🟥		🟥		🟥	🟥	
TRAP1	🟥	🟥	🟥	🟥		🟥		🟥	🟥	
NOL7	🟥	🟥				🟥		🟥	🟥	
PGM2	🟥				🟥	🟥	🟥	🟥		🟥
FZD5	🟥			🟥	🟥	🟥	🟥	🟥		🟥
TOMM..	🟥	🟥			🟥		🟥	🟥		🟥
FAHD1	🟥	🟥				🟥	🟥	🟥		🟥
ATAD3A	🟥	🟥		🟥		🟥		🟥		🟥
ZFP106	🟥		🟥		🟥	🟥		🟥	🟥	🟥
SLC25..				🟥	🟥	🟥	🟥	🟥		
ADH5				🟥	🟥	🟥	🟥	🟥		
SIP1	🟥		🟥	🟥	🟥	🟥	🟥	🟥		
SEPHS2	🟥			🟥	🟥	🟥		🟥		
RQCD1	🟥			🟥	🟥	🟥		🟥	🟥	
NFS1		🟥	🟥	🟥				🟥		🟥
EXOSC4	🟥	🟥	🟥	🟥				🟥		
TIMM44		🟥	🟥			🟥		🟥		🟥
ACBD4	🟥		🟥		🟥		🟥		🟥	

Fig. 26: Enrichr uses *Clustergrammer Web App*'s API to produce interactive heatmaps of enriched terms as columns and user input genes as rows.



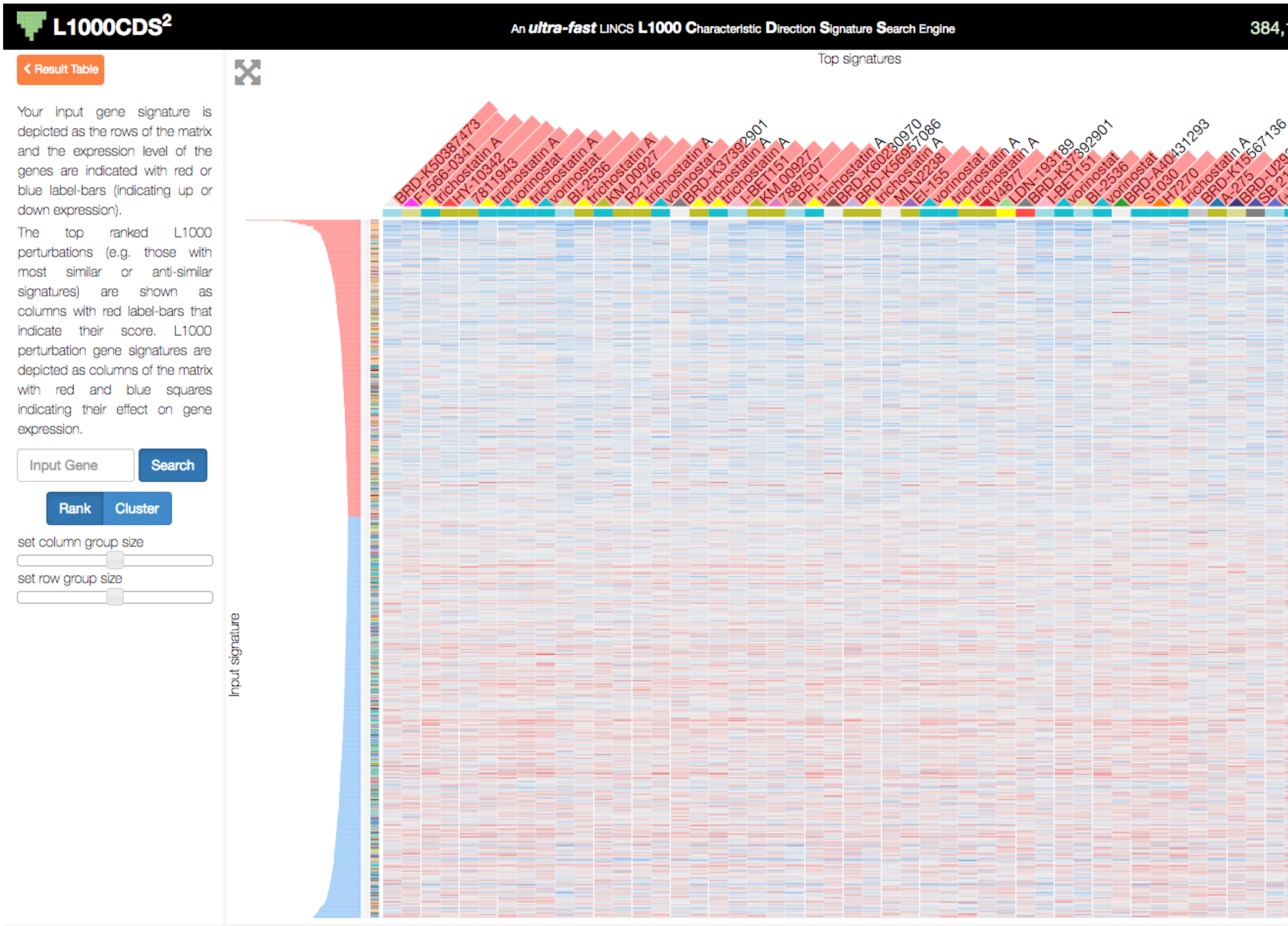
Genes

Hierarchical clustering of genes based on weights from the differential expression method.

Fig. 27: GEN3VA uses Clustergrammer's core libraries, *Clustergrammer-JS* and *Clustergrammer-PY*, to visualie gnee expression signatures and enrichment analysis results.

4.11.3 L1000CDS2

L1000CDS2 uses the *Clustergrammer Web App*'s API to produce interactive heatmaps of perturbagen gene signatures that mimic or reverse an input gene signature. This can be useful for users that are interested in the specific genes that are differentially regulated by the identified perturbagens.



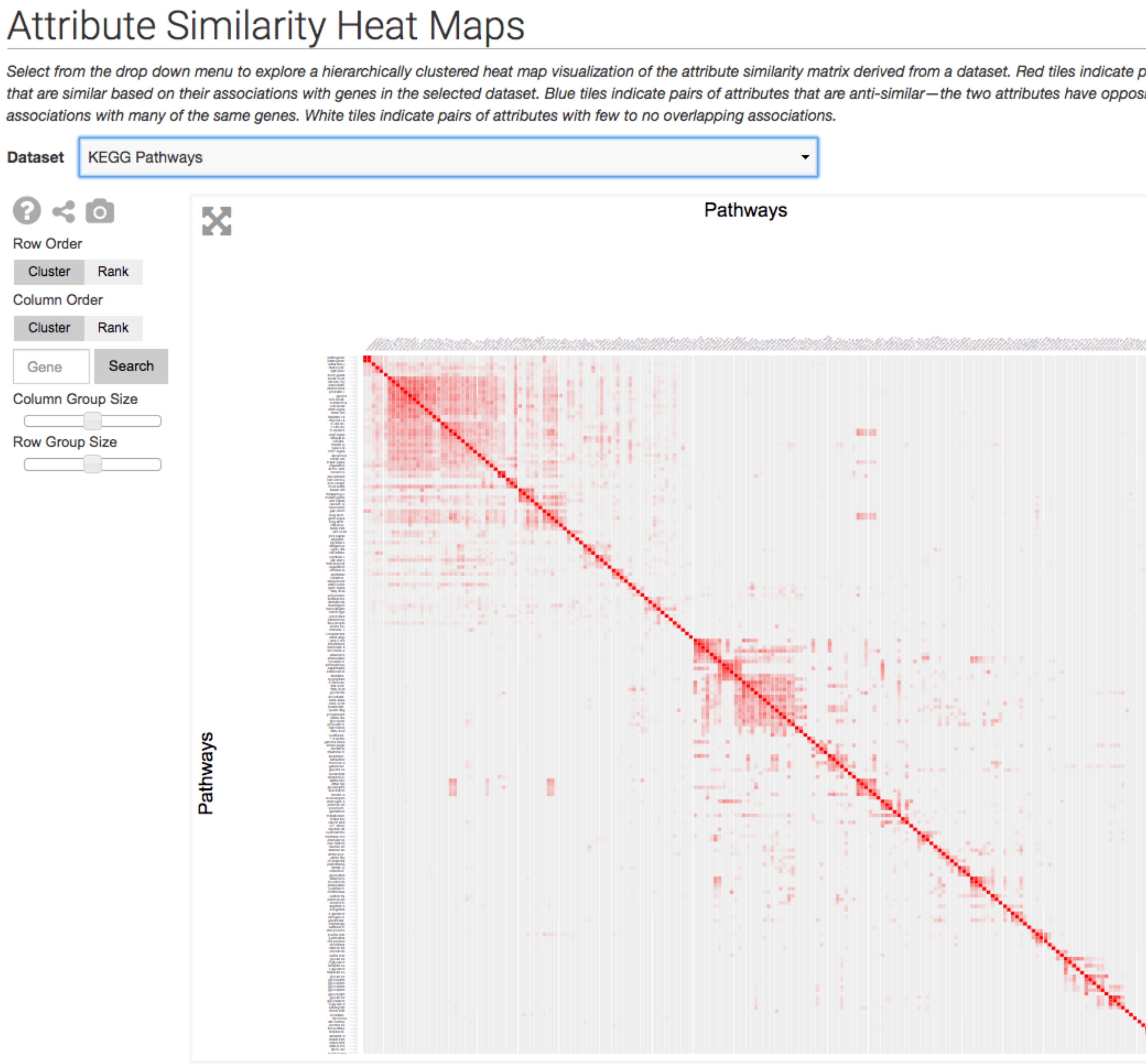
Your input gene signature is depicted as the rows of the matrix and the expression level of the genes are indicated with red or blue label-bars (indicating up or down expression).

The top ranked L1000 perturbations (e.g. those with most similar or anti-similar signatures) are shown as columns with red label-bars that indicate their score. L1000 perturbation gene signatures are depicted as columns of the matrix with red and blue squares indicating their effect on gene expression.

Fig. 28: L1000CDS2 uses Clustergrammer to produce interactive visualizations of input gene signatures and mimicking or reversing perturbation signatures. A users's input signature is shown as rows with gene-expression levels shown as row-bars (red/blue for up/down expression) and perturbations found to mimic/reverse their signature are shown as columns in the heatmap.

4.11.4 Harmonizome

The Harmonizome uses *Clustergrammer Web App*'s API to generate visualizations of curated biological datasets as heatmaps and adjacency matrices (e.g. to depict networks). The Harmonizome also uses the Clustergrammer to visualize the amount of biological information that is available for different families of genes in the Harmonogram



Attribute Similarity Heat Maps

Select from the drop down menu to explore a hierarchically clustered heat map visualization of the attribute similarity matrix derived from a dataset. Red tiles indicate p
that are similar based on their associations with genes in the selected dataset. Blue tiles indicate pairs of attributes that are anti-similar—the two attributes have opposi
associations with many of the same genes. White tiles indicate pairs of attributes with few to no overlapping associations.

Dataset: KEGG Pathways

Fig. 29: The Harmonizome uses Clustergrammer to visualize datasets as heatmaps and similarity matrices (e.g. similarity of attributes based on shared genes). Above is an example similarity matrix of KEGG pathways.

4.12 Developing Clustergrammer

Developers interested in working with the source code or in contributing to the Clustergrammer project can find instructions for the sub-projects here:

1. *Clustergrammer-JS Development*
2. *Clustergrammer-PY Development*
3. *Clustergrammer-Widget Development*
4. *Clustergrammer-Web Development*

Clustergrammer-JS and *Clustergrammer-PY* are the two core libraries which are used to build the *Clustergrammer Jupyter Widget* and the *Clustergrammer Web App* and can be used by developers to build their own web pages and apps.

4.13 License

Clustergrammer was developed by Nicolas F. Fernandez in the Ma'ayan lab at the Icahn School of Medicine at Mount Sinai.

Clustergrammer's license and third-party licenses are in the LICENSES directory.

Python Module Index

Index