

## S-TRACE Visualization (hivtrace-viz)

Generated by Doxygen 1.9.8

<b>1 Contributor Covenant Code of Conduct</b>	<b>1</b>
1.1 Our Pledge	1
1.2 Our Standards	2
1.3 Enforcement Responsibilities	2
1.4 Scope	2
1.5 Enforcement	3
1.6 Enforcement Guidelines	3
1.6.1 1. Correction	3
1.6.2 2. Warning	3
1.6.3 3. Temporary Ban	3
1.6.4 4. Permanent Ban	3
1.7 Attribution	4
<b>2 Migration for Clusters of Interest (Col) / Priority Sets History Tracking</b>	<b>4</b>
2.1 Requirements	4
2.1.1 Installing ChromeDriver	4
2.2 Usage:	4
2.2.1 Example Usage:	5
<b>3 hivtrace-viz</b>	<b>5</b>
3.1 Dependencies	5
3.2 Development	5
3.3 Deployment	5
3.4 Documentation	5
<b>4 README</b>	<b>6</b>
<b>5 Class Index</b>	<b>6</b>
5.1 Class List	6
<b>6 Class Documentation</b>	<b>6</b>
6.1 HIVTxNetwork Class Reference	6
6.1.1 Detailed Description	9
6.1.2 Member Function Documentation	9
6.1.3 Member Data Documentation	20
<b>Index</b>	<b>29</b>

# 1 Contributor Covenant Code of Conduct

## 1.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender

identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

## 1.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

## 1.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

## 1.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

## 1.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at [sweaver@temple.edu](mailto:sweaver@temple.edu). All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

## 1.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

### 1.6.1 1. Correction

**Community Impact:** Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

**Consequence:** A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

### 1.6.2 2. Warning

**Community Impact:** A violation through a single incident or series of actions.

**Consequence:** A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

### 1.6.3 3. Temporary Ban

**Community Impact:** A serious violation of community standards, including sustained inappropriate behavior.

**Consequence:** A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

### 1.6.4 4. Permanent Ban

**Community Impact:** Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

**Consequence:** A permanent ban from any sort of public interaction within the community.

## 1.7 Attribution

This Code of Conduct is adapted from the [Contributor Covenant](https://www.contributor-covenant.org/version/2/0/code_of_conduct.html), version 2.0, available at [https://www.contributor-covenant.org/version/2/0/code\\_of\\_conduct.html](https://www.contributor-covenant.org/version/2/0/code_of_conduct.html).

Community Impact Guidelines were inspired by [Mozilla's code of conduct enforcement ladder](#).

For answers to common questions about this code of conduct, see the FAQ at <https://www.contributor-covenant.org/faq>. Translations are available at <https://www.contributor-covenant.org/translations>.

## 2 Migration for Clusters of Interest (Col) / Priority Sets History Tracking

`get-cluster-history.py` is a script that uses Selenium to scrape the history of changes to the Clusters of Interest (Col) / Priority Sets in the site. It is intended to be used for migration purposes, to generate the previously untracked history of changes to the Col / Priority Sets.

The script loads the site with a provided network json file and priority set json file, and then downloads the history of changes by clicking the "View history over time" button in the Col / Priority Sets page.

### 2.1 Requirements

- Python3
- argparse
- selenium
- Chrome
- ChromeDriver

#### 2.1.1 Installing ChromeDriver

1. Download the latest version of ChromeDriver from <https://googlechromelabs.github.io/chrome-for-testing/>
2. For Linux, extract the downloaded file and move the chromedriver binary to `/usr/local/bin/chromedriver` (or any other directory in your PATH)

### 2.2 Usage:

Ensure the site is running at <http://127.0.0.1:8080/> with `npm run develop` and then run `python get-cluster-json-data.py`. See `python get-cluster-json-data.py --help` for more options. To insert the data into the MongoDB database, run `python mongodb-insert-cluster-json-data.py`. See `python mongodb-insert-cluster-json-data.py --help` for more options.

### 2.2.1 Example Usage:

```
python get-cluster-json-data.py --network-file "/ui-tests/data/dummy_data_network.json" --priority-sets-file  
"/ui-tests/data/dummy_data_coi.json"
```

**\*\*Note the file URL here is respective to the html/priority-sets-args.html file, with the root directory (/) being the root of the project.\*\***

```
python mongodb-insert-cluster-json-data.py --input-file "output/dummy_data_network/2024-09-10.json"  
--database "datamonkey-dev"
```

## 3 hivtrace-viz

This repository contains the visualization code for HIV-TRACE.

### 3.1 Dependencies

- Yarn or NPM

### 3.2 Development

```
git clone https://github.com/veg/hivtrace-viz.git  
cd hivtrace-viz  
yarn  
yarn develop
```

Navigate your browser to localhost:8273

### 3.3 Deployment

The HIV-Trace webpage was formerly hosted using github-pages at <https://veg.github.io/hivtrace-viz/> but has been migrated to using a pm2 process at [hivtrace-viz.hyphy.org](https://hivtrace-viz.hyphy.org) (see [veg/operations](#) documentation for details).

The master branch of this repo should always be in sync with what is published to NPM and is live on the production website. With the exception of urgent bug fixes, all changes to veg/master should be done via pull requests from veg/develop.

### 3.4 Documentation

A basic JSDoc documentation of hivtrace-viz can be found at `docs/` and can be viewed by cloning the repository and opening `docs/index.html` in a browser (or running a basic web server in the `docs/` directory, e.g. `python -m http.server`).

A Doxygen-based PDF version of the documentation can be found at `docs/hivtrace-viz-doxygen-docs.pdf`.

This documentation is generated automatically using `jsdoc -c jsdoc.json` and ran as part of the CI/CD pipeline (see `.github/workflows/jsdoc-doxygen.yml`).

## 4 README

Some scripts to assist / automate the refactoring process.

`function-usages.py`: Provide a regex expression (optional) and files to search within. The script returns all instances of the regex expression that are used only in one of the files (and is useful since a function that is only used in one file can just be moved to that file).

Default pattern: `self\[a-zA-Z0-9_\]+\ (` (i.e. `self` followed by a function name)

For example, finding `clusternetwork.js` `self` functions that can be refactored from `clusterOI.js`, run: `python3 refactor/function-usages.py src/clusternetwork.js src/clustersOI/clusterOI.js`

Run `python3 refactor/function-usages.py --help` for more details.

## 5 Class Index

### 5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

#### [HIVTxNetwork](#)

**Represents an HIV transmission network with annotations**

**6**

## 6 Class Documentation

### 6.1 HIVTxNetwork Class Reference

Represents an HIV transmission network with annotations.

#### Public Member Functions

- [constructor](#) (`json`, `button_bar_ui`, `primary_key_function`)
- [initialize\\_ui\\_ux\\_elements](#) ()  
*initialize UI/UX elements*
- [tabulate\\_multiple\\_sequences](#) ()  
*Iterate over nodes in the network, identify all those which share the same primary key (i.e., the same individual), tabulate them, and collate node attributes.*
- [cluster\\_display\\_filter](#) (`cluster`)  
*@cluster [dict] : cluster object*
- [get\\_reference\\_date](#) ()
- [lookup\\_option](#) (`key`, `default_value`, `options`)
- [get\\_ui\\_element\\_selector\\_by\\_role](#) (`role`, `not_nested`)  
*retrive the DOM ID for an element given its data-hivtrace-ui-role*
- [process\\_multiple\\_sequences](#) (`reduce_distance_within`, `reduce_distance_between`)  
*Process the network to simplify multiple sequences per individual.*
- [annotate\\_multiple\\_clusters\\_on\\_nodes](#) ()

When MSPP are present, this function will annotate node objects with fields that indicate whether or not the nodes belong to multiple clusters or subclusters.

- [simplify\\_multisequence\\_cluster](#) (filtered\_json)
 

When MSPP are present, this function will reduce the network encoded by .Nodes and .Edges in filtered\_json, and reduce all sequences that represent the same entity into one node.
- [generate\\_cross\\_hatch\\_pattern](#) (color)
 

generate a cross-hatch pattern for filling nodes with a specific color and add it as a definition to the network SVG
- [priority\\_groups\\_pending](#) ()
 

filter the list of Col to return those which have not been reviewed/validated
- [priority\\_groups\\_expanded](#) ()
 

filter the list of Col to return those which have been automatically expanded
- [priority\\_groups\\_automatic](#) ()
 

filter the list of Col to return those which have been created by the system
- [generateClusterOfInterestID](#) (subcluster\_id)
 

generate the name for a cluster of interest
- [map\\_ids\\_to\\_objects](#) ()
 

create a map between node IDs and node objects
- [attribute\\_node\\_value\\_by\\_id](#) (d, id, number)
 

Fetch the value of an attribute from the node.
- [extract\\_single\\_cluster](#) (nodes, filter, no\_clone, given\_json, include\_extra\_edges, edge\_subset)
 

Extract the nodes and edges between them into a separate object.

### Static Public Member Functions

- static [lookup\\_form\\_generator](#) ()
- static [is\\_new\\_node](#) (node)
 

does the node have "new node" attribute
- static [is\\_edge\\_injected](#) (e)
 

Is this node NOT genetic, i.e.

### Public Attributes

- [\\_calc\\_country\\_nodes](#)

this is a function which calculates country node centers for the (experimental) option of rendering networks with topo maps
- [filter\\_by\\_size](#)

@cluster [dict] : cluster object
- [unique\\_entity\\_list](#)

@node\_list [array] : list of nodes
- [unique\\_entity\\_list\\_from\\_ids](#)

@node\_list [array] : list of node IDs
- [unique\\_entity\\_object\\_list](#)

@node\_list [array] : list of nodes
- [filter\\_singletons](#)

@cluster [dict] : cluster object
- [filter\\_if\\_added](#)

@cluster [dict] : cluster object
- [filter\\_time\\_period](#)

@cluster [dict] : cluster object
- [priority\\_groups\\_find\\_by\\_name](#)



- [lookup a Col by name; null if not found](#)
- [priority\\_groups\\_all\\_events](#)  
generate a set of all unique temporal events (when new data were added to ANY Col) return a Set of date strings formatted with `timeDateUtil.DateViewFormatSlider`
- [priority\\_groups\\_compute\\_overlap](#)  
compute the overlap between Col
- [auto\\_expand\\_pg\\_handler](#)  
Grow a Col defined in @pg based on its growth mode.
- [priority\\_groups\\_export](#)  
export Col records for interactions with the external DB @group\_set : custom set or all (if null) @include\_↔  
unvalidated: if true will include Col which did not undergo/pass validation

## : the name of the Col

interact with the remote DB to send updates of Col operations

@operation: what happened ("insert", "delete", "update")

- [priority\\_groups\\_update\\_node\\_sets](#)
- [priority\\_groups\\_edit\\_set\\_description](#)  
A function that updates the "freehand" description of a specific Col.
- [priority\\_groups\\_remove\\_set](#)  
Remove a Col from the list of defined Col.
- [priority\\_groups\\_export\\_nodes](#)  
Export nodes that are members of Col.
- [priority\\_groups\\_export\\_sets](#)  
Export Col summary info.
- [priority\\_groups\\_is\\_new\\_node](#)  
returns true is the node was added by the system during Col definition/expansion
- [check\\_for\\_time\\_series](#)  
Generate a function callback for attribute time series data.
- [parse\\_dates](#) (value)  
parse a date record
- [filter\\_by\\_date](#) (cutoff, date\_field, start\_date, node, count\_newly\_added)  
Check if the date attribute of a node falls within a pre-specified range.
- [priority\\_group\\_entity\\_count](#) (pg)
- [priority\\_groups\\_validate](#) (groups, auto\_extend)  
validate the list of Col
- [display\\_warning](#) (warning\_string, is\_html)  
display a warning string
- [priority\\_groups\\_compute\\_node\\_membership](#) ()  
Compute which Col do various nodes belong to, and define additional attributes for each node.
- [priority\\_group\\_node\\_record](#) (node\_id, date, kind)  
Generate a Col node record.
- [load\\_priority\\_sets](#) (url, is\_writeable)  
read and process JSON files defining COI
- [inject\\_attribute\\_description](#) (key, d)  
add an attribute description
- [populate\\_predefined\\_attribute](#) (computed, key)  
populate\_predefined\_attribute

- [define\\_attribute\\_COI\\_membership](#) (network, date)  
=====
- [define\\_attribute\\_binned\\_vl](#) (field, title)  
*define an attribute generator for binned viral loads*
- [define\\_attribute\\_vl\\_interpretation](#) ()  
*define an attribute generator for Viral load result interpretation*
- [define\\_attribute\\_network\\_update](#) ()  
*define an attribute generator for new network nodes/clusters*
- [define\\_attribute\\_dx\\_years](#) (relative, label)  
*define an attribute generator for dx year*
- [fetch\\_sequences\\_for\\_pid](#) (pid)  
*Retrieve the list of sequences associated with a node.*
- [define\\_attribute\\_sequence\\_count](#) (label)  
*define an attribute generator for the number of sequences associated with this node*
- [define\\_attribute\\_age\\_dx](#) ()
- [annotate\\_cluster\\_changes](#) ()  
*annotate\_cluster\_changes*
- [extract\\_individual\\_level\\_records](#) ()  
*extract\_individual\_level\_records*
- [aggregate\\_individual\\_level\\_records](#) (node\_list)  
*aggregate\_individual\_level\_records*
- [entity\\_id\\_from\\_string](#) (node\_name)  
*generate an entity (primary key) id from string*
- [entity\\_id](#) (node)  
*generate an entity (primary key) id from node*
- [apply\\_to\\_entities](#) (cb)
- [list\\_of\\_aliased\\_sequences](#) (node)  
*generate a list of sequence IDs represented by a node*
- static [inject\\_attribute\\_node\\_value\\_by\\_id](#) (node, id, value)  
*Add an attribute value to the node object.*

### 6.1.1 Detailed Description

Represents an HIV transmission network with annotations.

#### Parameters

<i>{Object}</i>	json - A JSON object containing the network data.
<i>{HTMLElement}</i>	button_bar_ui - A UI element for interacting with the network.
<i>{Object}</i>	cluster_attributes - Attributes related to clusters within the network.

### 6.1.2 Member Function Documentation

#### [aggregate\\_individual\\_level\\_records\(\)](#)

```
HIVTxNetwork::aggregate_individual_level_records (
    node_list ) [inline]
```

## aggregate\_individual\_level\_records

for networks that have multiple sequences per individual, this function will reduce the list of node records to only have one per primary key all attributes where more than one value is present will be shown as ';' separated

## annotate\_cluster\_changes()

```
HIVTxNetwork::annotate_cluster_changes ( ) [inline]
```

### annotate\_cluster\_changes

If the network contains information about cluster changes (new/moved/deleted nodes, etc), this function will annotate cluster objects (in place) with various attributes "delta" : change in the size of the cluster "flag" : a status flag to be used in the cluster display table if set to 2 then TBD if set to 3 then TBD

## attribute\_node\_value\_by\_id()

```
HIVTxNetwork::attribute_node_value_by_id (
    d,
    id,
    number ) [inline]
```

Fetch the value of an attribute from the node.

### Parameters

<i>d</i>	node object
<i>id</i>	[string] the attribute whose value should be fetched
<i>number</i>	[bool] if true, only return numerical values

## cluster\_display\_filter()

```
HIVTxNetwork::cluster_display_filter (
    cluster ) [inline]
```

@cluster [dict] : cluster object

return true if the cluster passes all the currently defined filters see this.cluster\_filtering\_functions

## constructor()

```
HIVTxNetwork::constructor (
    json,
    button_bar_ui,
    primary_key_function ) [inline]
```

SLKP 20241029 this function is used to identify which nodes are duplicates it converts the name of the node (sequence) into a primary key ID (by default, taking the .id string up to the first pipe) all sequences/nodes that map to the same primary key are assumed to represent the same entity / individual

initialize UI/UX elements

the list of defined clusters of interest, format as follows (SLKP, 20240715: may need updating) { 'name' : 'unique name', 'nodes' : [ { 'node\_id' : text, 'added' : date, 'kind' : text }], 'created' : date, 'description' : 'text', 'modified' : date, 'kind' : 'text' }

time filter element for various displays

### define\_attribute\_binned\_vl()

```
HIVTxNetwork::define_attribute_binned_vl (
    field,
    title ) [inline]
```

define an attribute generator for binned viral loads

#### Parameters

<i>field</i>	the node attribute field to use
<i>title</i>	display this title for the attribute

#### Returns

attribute definition dict

### define\_attribute\_COI\_membership()

```
HIVTxNetwork::define_attribute_COI_membership (
    network,
    date ) [inline]
```

=====

attribute callback definitions

The following functions are generators for attribute callbacks. They return dict-like objects that contain fields used to populate and display network node and cluster attributes

The fields in the attribute definition are as follows

depends [optional] : the list of node fields that must be defined in order for this attribute to be computed; null = none

label [required] : the attribute label to display in the dropdown other locations enum [optional] : if provided as an array, specifies the set of allowed values volatile [optional] : if non-null, tag this attribute for re-computation when certain events take place color\_scale[required]: value=>color map for rendering map[required] : a function to compute attribute value from node data color\_stops[optional]: # of color stops for a continuous variable that's binned ===== define an attribute generator for subcluster membership attribute

**Parameters**

<i>network</i>	: the network / cluster object to use
<i>data</i>	reference date to use

**Returns**

attribute definition

**define\_attribute\_dx\_years()**

```
HIVTxNetwork::define_attribute_dx_years (
    relative,
    label ) [inline]
```

define an attribute generator for dx year

**Parameters**

<i>relative</i>	if T, compute dx date relative to the network date in years
<i>label</i>	use this label

**Returns**

attribute definition dict

**define\_attribute\_network\_update()**

```
HIVTxNetwork::define_attribute_network_update ( ) [inline]
```

define an attribute generator for new network nodes/clusters

**Returns**

attribute definition dict

**define\_attribute\_sequence\_count()**

```
HIVTxNetwork::define_attribute_sequence_count (
    label ) [inline]
```

define an attribute generator for the number of sequences associated with this node

**Parameters**

<i>label</i>	use this label
--------------	----------------

**Returns**

attribute definition dict

**define\_attribute\_vl\_interpretaion()**

```
HIVTxNetwork::define_attribute_vl_interpretaion ( ) [inline]
```

define an attribute generator for Viral load result interpretatio

**Returns**

attribute definition dict

**entity\_id()**

```
HIVTxNetwork::entity_id (
    node ) [inline]
```

generate an entity (primary key) id from node

**Parameters**

<i>node</i>	(Object)
-------------	----------

returns [String] entity id

**entity\_id\_from\_string()**

```
HIVTxNetwork::entity_id_from_string (
    node_name ) [inline]
```

generate an entity (primary key) id from string

**Parameters**

<i>node_name</i>	(string)
------------------	----------

returns [String] entity id

**extract\_individual\_level\_records()**

```
HIVTxNetwork::extract_individual_level_records ( ) [inline]
```

extract\_individual\_level\_records

for networks that have multiple sequences per individual, this function will reduce the list of node records to only include those that have attribute data. If more than one node has attribute data, the first one (chosen based on the sorting order when this.primary\_key\_list was initialized) is returned.

**extract\_single\_cluster()**

```
HIVTxNetwork::extract_single_cluster (
    nodes,
    filter,
    no_clone,
    given_json,
    include_extra_edges,
    edge_subset ) [inline]
```

Extract the nodes and edges between them into a separate object.

**Parameters**

<i>nodes</i>	[array] the list of nodes to extract
<i>filter</i>	[function, optional] (edge) -> bool filtering function for deciding which edges will be used to define clusters
<i>no_clone</i>	[bool] if set to T, node objects are <b>not</b> shallow cloned in the return object

**Returns**

[dict] the object representing "Nodes" and "Edges" in the extracted cluster

**fetch\_sequences\_for\_pid()**

```
HIVTxNetwork::fetch_sequences_for_pid (
    pid ) [inline]
```

Retrieve the list of sequences associated with a node.

**Parameters**

<i>pid</i>	use this entity id
------------	--------------------

**Returns**

list of sequence\_ids

**filter\_by\_date()**

```
HIVTxNetwork::filter_by_date (
    cutoff,
    date_field,
    start_date,
    node,
    count_newly_added ) [inline]
```

Check if the date attribute of a node falls within a pre-specified range.

## Parameters

<i>cutoff</i>	
<i>date_file</i>	
<i>start_date</i>	
<i>node</i>	
<i>count_newly_add</i>	[bool]; if true, then a "new node" attribute overrides date checks, so all new (compared to the previous network) nodes pass the check

**get\_reference\_date()**

```
HIVTxNetwork::get_reference_date ( ) [inline]
```

get the reference (creation) date for the network same as "today", unless this is not the primary network (cluster or subcluster view), in which case the reference date for the parent is used

**get\_ui\_element\_selector\_by\_role()**

```
HIVTxNetwork::get_ui_element_selector_by_role (
    role,
    not_nested ) [inline]
```

retriev the DOM ID for an element given its data-hivtrace-ui-role

## Parameters

<i>role</i>	data-hivtrace-ui-role
<i>nested</i>	true if this is being called from a secondary network or element (dialog, cluster view etc), which does not have primary button_ui elements

**initialize\_ui\_ux\_elements()**

```
HIVTxNetwork::initialize_ui_ux_elements ( ) [inline]
```

initialize UI/UX elements

define a D3 behavior to make node labels draggable

default node colorizer

if there is computed support for network edges, use it to highlight possible spurious edges

default node shaper

d3 layout option setting

filters which control which clusters get rendered



**inject\_attribute\_description()**

```
HIVTxNetwork::inject_attribute_description (
    key,
    d ) [inline]
```

add an attribute description

Given an attribute definition (see comments elsewhere), and a key to associate it with do

**inject\_attribute\_node\_value\_by\_id()**

```
static HIVTxNetwork::inject_attribute_node_value_by_id (
    node,
    id,
    value ) [inline], [static]
```

Add an attribute value to the node object.

**Parameters**

<i>node</i>	[object] : node,
<i>id</i>	[string] : attribute id
<i>value</i>	: attribute value

**is\_edge\_injected()**

```
static HIVTxNetwork::is_edge_injected (
    e ) [inline], [static]
```

Is this node NOT genetic, i.e.

added to the network via social or other means

**list\_of\_aliased\_sequences()**

```
HIVTxNetwork::list_of_aliased_sequences (
    node ) [inline]
```

generate a list of sequence IDs represented by a node

**Parameters**

<i>node</i>	(Object)
-------------	----------

returns [array] list of sequence ids

**load\_priority\_sets()**

```
HIVTxNetwork::load_priority_sets (
    url,
    is_writeable ) [inline]
```

read and process JSON files defining COI

**Parameters**

<i>url</i>	[string]: load the data from here
<i>is_writeable</i>	[string]: if "writeable", changes to COI lists will be pushed back to the server

This needs to be called AFTER the clusters/subclusters have been annotated check if the system needs to create/expand Col

**lookup\_option()**

```
HIVTxNetwork::lookup_option (
    key,
    default_value,
    options ) [inline]
```

retrieve an option associated with "key" if not found in Settings or options, return "default value"

**parse\_dates()**

```
HIVTxNetwork::parse_dates (
    value ) [inline]
```

parse a date record

**Parameters**

<i>value</i>	(date object or string)
--------------	-------------------------

**Returns**

date object

**populate\_predefined\_attribute()**

```
HIVTxNetwork::populate_predefined_attribute (
    computed,
    key ) [inline]
```

populate\_predefined\_attribute

Given an attribute definition (see comments elsewhere), and a key to associate it with do

0. Inject the definition of the attribute into the network dictionary

1. Compute the value of the attribute for all nodes
2. Compute unique values

#### Parameters

<i>computed</i>	(dict) : attribute definition
<i>key</i>	(string) : the key to associate with the attribute

### priority\_group\_node\_record()

```
HIVTxNetwork::priority_group_node_record (
    node_id,
    date,
    kind ) [inline]
```

Generate a Col node record.

#### Parameters

<i>node↔ _id</i>	[string] : node name,
<i>date</i>	(optional) : creation date
<i>kind</i>	(optional) : node creation mode

### priority\_groups\_compute\_node\_membership()

```
HIVTxNetwork::priority_groups_compute_node_membership ( ) [inline]
```

Compute which Col do various nodes belong to, and define additional attributes for each node.

define and populate categorical node attributes

### priority\_groups\_validate()

```
HIVTxNetwork::priority_groups_validate (
    groups,
    auto_extend ) [inline]
```

validate the list of Col

#### Parameters

<i>groups</i>	{array} is a list of Col name: unique string description: string, nodes: { { 'id' : node id, 'added' : date, 'kind' : _cdcPrioritySetNodeKind } }, created: date, kind: kGlobals.CDCCOIKind, tracking: kGlobals.CDCCOITrackingOptions createdBy : kGlobals.CDCCOICreatedBySystem,kGlobals.CDCCOICreatedManually
<i>auto_extend</i>	{bool} : if true, automatically expand existing Col

extract the list of clusters meeting national priority criteria, these have been precomputed elsewhere (*priority\_score*)

check for nodes that are in the Col but may be missing from the network

extract network data at 0.015 and subcluster thresholds filter on dates subsequent to the created date

all the network nodes connected to the nodes in the Col at 1.5%; directly or indirectly

all the network nodes connected to the nodes in the subcluster threshold (0.5%); also saves all the edges that have been taken if *auto\_extend* is true

all the network nodes connected to the nodes in the Col at 1.5%; only directly

all the network nodes connected to the nodes in the Col at 1.5%; only directly

extract the 1.5% cluster network object

process the cluster object to extract directly connected subcluster nodes and new nodes

partition all the Col nodes into groups

handle Col growth

check to see the Col meets priority definitions

### **process\_multiple\_sequences()**

```
HIVTxNetwork::process_multiple_sequences (
    reduce_distance_within,
    reduce_distance_between ) [inline]
```

Process the network to simplify multiple sequences per individual.

1. Identify null clusters, i.e., clusters that consist only of sequences with the same primary key (individual) Delete ALL null clusters; remove all nodes and edges associated with them
2. Identify identical sequence sets, i.e., sequences with the same individual that have the same connection patterns, (a) All sequences in the set have the same primary key (b) All sequences in the set are connected to each other (at length  $\leq$  *reduce\_distance\_within*) (c) All sequences in the set are connected to the same set of OTHER sequences (at length  $\leq$  *reduce\_distance\_between*)  
All identical sequence sets are collapsed to a

now iterate over non-trivial clusters, and see if any nodes are collapsible

### **simplify\_multisequence\_cluster()**

```
HIVTxNetwork::simplify_multisequence_cluster (
    filtered_json ) [inline]
```

When MSPP are present, this function will reduce the network encoded by *.Nodes* and *.Edges* in *filtered\_json*, and reduce all sequences that represent the same entity into one node.

Such nodes inherit the union of their links (so at least of the sequences being collapsed link to X, the "joint" node will link to X).

The joint nodes will also receive aggregated attributes; if the nodes being merged have different attributes values for a given key, the merged node will have a ';' separated list of attributes for the same key. 20241030 SLKP Perform a greedy collapse of all the sequences that map to the same primary key For a reduced cluster view

### **tabulate\_multiple\_sequences()**

```
HIVTxNetwork::tabulate_multiple_sequences ( ) [inline]
```

Iterate over nodes in the network, identify all those which share the same primary key (i.e., the same individual), tabulate them, and collate node attributes.

generate a primary key to node ID map [primary key] => [array of IDs]

iterate over all duplicate sequences, synchronize node attributes

### **6.1.3 Member Data Documentation**

#### **\_calc\_country\_nodes**

```
HIVTxNetwork::_calc_country_nodes
```

##### **Initial value:**

```
= (calc_options) => {  
  if (calc_options && "country-centers" in calc_options) {  
    this.mapProjection = d3.geo  
      .mercator()  
      .translate([  
        this.margin.left + this.width / 2,  
        this.margin.top + this.height / 2,  
      ])  
      .scale((150 * this.width) / 960);  
    _.each(this.countryCentersObject, (value) => {  
      value.countryXY = this.mapProjection([value.longt, value.lat]);  
    });  
  }  
}
```

this is a function which calculates country node centers for the (experimental) option of rendering networks with topo maps

#### **auto\_expand\_pg\_handler**

```
HIVTxNetwork::auto_expand_pg_handler
```

Grow a Col defined in @pg based on its growth mode.

##### **Returns**

the set of added nodes (by numeric ID) @nodeID2idx : if provided, maps the name of the node to its index in the nodes array; avoids repeated traversal if provided

#### **check\_for\_time\_series**

```
HIVTxNetwork::check_for_time_series
```

Generate a function callback for attribute time series data.

## Parameters

<i>export_items</i>	if set (and is an array), the function will add the callback to the array otherwise the callback will be invoked on this
---------------------	--

## Returns

noting

**filter\_by\_size**

HIVTxNetwork::filter\_by\_size

**Initial value:**

```
= (cluster) => {
  return cluster.children.length >= this.minimum_cluster_size;
}
```

@cluster [dict] : cluster object

return true if cluster size is at least this.minimum\_cluster\_size

**filter\_if\_added**

HIVTxNetwork::filter\_if\_added

**Initial value:**

```
= (cluster) => {
  return this.cluster_attributes[cluster.cluster_id].type !== "existing";
}
```

@cluster [dict] : cluster object

return true if the cluster is new compared to the previous network

**filter\_singletons**

HIVTxNetwork::filter\_singletons

**Initial value:**

```
= (cluster) => {
  return cluster.children.length > 1;
}
```

@cluster [dict] : cluster object

return true if cluster size is at least 2

## filter\_time\_period

HIVTxNetwork::filter\_time\_period

### Initial value:

```
= (cluster) => {
  return _.some(
    this.nodes_by_cluster[cluster.cluster_id],
    (n) =>
      this.attribute_node_value_by_id(
        n,
        timeDateUtil.getClusterTimeScale()
      ) >= this.using_time_filter
  );
}
```

@cluster [dict] : cluster object

return true if the cluster has nodes newer than this.using\_time\_filter

## priority\_groups\_all\_events

HIVTxNetwork::priority\_groups\_all\_events

### Initial value:

```
= function () {
  const events = new Set();
  if (this.defined_priority_groups) {
    _.each(this.defined_priority_groups, (g) => {
      _.each(g.nodes, (n) => {
        events.add(timeDateUtil.DateViewFormatSlider(n.added));
      });
    });
  }
  return events;
}
```

generate a set of all unique temporal events (when new data were added to ANY Col) return a Set of date strings formatted with timeDateUtil.DateViewFormatSlider

## priority\_groups\_compute\_overlap

HIVTxNetwork::priority\_groups\_compute\_overlap

compute the overlap between Col

@groups: an array with Col objects

1. Populate this.priority\_node\_overlap dictionary which stores, for every node present in AT LEAST ONE Col, the set of all PGs it belongs to, as in "node-id" => set ("PG1", "PG2"...)
2. For each Col, create and populate a member field, .overlaps which is a dictionary that stores { sets : #of Col with which it shares nodes nodes: the # of nodes contained in overlaps }

### priority\_groups\_edit\_set\_description

HIVTxNetwork::priority\_groups\_edit\_set\_description

#### Initial value:

```
= function (
    name,
    description,
    update_table
) {
    let pg_to_update = this.priority_groups_find_by_name(name);
    if (pg_to_update) {
        pg_to_update.description = description;
        this.priority_groups_update_node_sets(name, "update");
        if (update_table) {
            clustersOfInterest.draw_priority_set_table(this);
        }
    }
}
```

A function that updates the "freehand" description of a specific Col.



**Parameters**

<i>name</i>	[string] : the name of the Col
<i>description</i>	[string] : the actual description
<i>update_table</i>	[bool] : if true, trigger Col table update in UI/UX

**Returns**

N/A

**priority\_groups\_export**

HIVTxNetwork::priority\_groups\_export

**Initial value:**

```
= function (group_set, include_unvalidated) {  
  group_set = group_set || this.defined_priority_groups;  
  
  return _.map(  
    _.filter(group_set, (g) => include_unvalidated || g.validated),  
    (g) => ({  
      name: g.name,  
      description: g.description,  
      nodes: g.nodes,  
      modified: timeDateUtil.DateFormats[0] (g.modified),  
      kind: g.kind,  
      created: timeDateUtil.DateFormats[0] (g.created),  
      createdBy: g.createdBy,  
      tracking: g.tracking,  
      autocreated: g.autocreated,  
      autoexpanded: g.autoexpanded,  
      pending: g.pending,  
      history: g.history,  
    })  
  );  
}
```

export Col records for interactions with the external DB @group\_set : custom set or all (if null) @include\_unvalidated: if true will include Col which did not undergo/pass validation

**priority\_groups\_export\_nodes**

HIVTxNetwork::priority\_groups\_export\_nodes

Export nodes that are members of Col.

**Parameters**

<i>name</i>	[array] : set of Col OBJECTS, by default this is defined_priority_groups
<i>include_unvalidated</i>	[bool] : if true, include all Col (validated/not) in the export

**Returns**

an array of node records

**priority\_groups\_export\_sets**

HIVTxNetwork::priority\_groups\_export\_sets

**Initial value:**

```
= function () {
  return _.flatten(
    _.map(
      _.filter(this.defined_priority_groups, (g) => g.validated),
      (g) => ({
        cluster_type: g.createdBy,
        cluster_uid: g.name,
        cluster_modified_dt: timeDateUtil.hivtrace_date_or_na_if_missing(
          g.modified
        ),
        cluster_created_dt: timeDateUtil.hivtrace_date_or_na_if_missing(
          g.created
        ),
        cluster_ident_method: g.kind,
        cluster_growth: kGlobals.CDCCOIConciseTrackingOptions[g.tracking],
        cluster_current_size: this.aggregate_individual_level_records(
          g.node_objects
        ).length,
        national_priority: g.meets_priority_def,
        cluster_dx_recent12_mo: g.cluster_dx_recent12_mo,
        cluster_dx_recent36_mo: g.cluster_dx_recent36_mo,
        cluster_overlap: g.overlap.sets,
      })
    )
  );
}
```

Export Col summary info.

**Returns**

an array of Col records

**priority\_groups\_find\_by\_name**

HIVTxNetwork::priority\_groups\_find\_by\_name

**Initial value:**

```
= function (name) {
  if (this.defined_priority_groups) {
    return _.find(this.defined_priority_groups, (g) => g.name === name);
  }
  return null;
}
```

lookup a Col by name; null if not found

**priority\_groups\_is\_new\_node**

HIVTxNetwork::priority\_groups\_is\_new\_node

**Initial value:**

```
= function (node) {
  return node.autoadded;
}
```

returns true is the node was added by the system during Col definition/expansion

## priority\_groups\_remove\_set

HIVTxNetwork::priority\_groups\_remove\_set

### Initial value:

```
= function (name, update_table) {
  if (this.defined_priority_groups) {
    var idx = _.findIndex(
      this.defined_priority_groups,
      (g) => g.name === name
    );

    if (idx >= 0) {
      this.defined_priority_groups.splice(idx, 1);
      this.priority_groups_update_node_sets(name, "delete");
      if (update_table) {
        clustersOfInterest.draw_priority_set_table(this);
      }
    }
  }
}
```

Remove a Col from the list of defined Col.

### Parameters

<i>name</i>	[string] : the name of the Col
<i>update_table</i>	[bool] : if true, trigger Col table update in UI/UX

### Returns

N/A

## priority\_groups\_update\_node\_sets

HIVTxNetwork::priority\_groups\_update\_node\_sets

### Initial value:

```
= function (name, operation) {
  const coi_to_update = this.priority_groups_find_by_name(name);
  if (coi_to_update) {
    const sets = this.priority_groups_export([coi_to_update]);

    const to_post = {
      operation: operation,
      name: name,
      url: window.location.href,
      sets: JSON.stringify(sets),
    };

    if (this.priority_set_table_write && this.priority_set_table_writeable) {
      d3.text(this.priority_set_table_write)
        .header("Content-Type", "application/json")
        .post(JSON.stringify(to_post), (error, data) => {
          if (error) {
            console.log("received fatal error:", error);
          }
        });
    }
  }
}
```

## unique\_entity\_list

HIVTxNetwork::unique\_entity\_list

### Initial value:

```
= (node_list) => {  
  return _.map(  
    _.groupBy(node_list, (n) => this.primary_key(n)),  
    (d, k) => k  
  );  
}
```

@node\_list [array] : list of nodes

returns the list of unique "individuals", collapsing nodes representing multiple sequences from the same entity into a single blob

## unique\_entity\_list\_from\_ids

HIVTxNetwork::unique\_entity\_list\_from\_ids

### Initial value:

```
= (node_list) => {  
  return this.unique_entity_list(  
    _.map(node_list, (d) => {  
      return { id: d };  
    })  
  );  
}
```

@node\_list [array] : list of node IDs

returns the list of unique "individuals", collapsing nodes representing multiple sequences from the same entity into a single blob

## unique\_entity\_object\_list

HIVTxNetwork::unique\_entity\_object\_list

### Initial value:

```
= (node_list) => {  
  return _.groupBy(node_list, (n) => this.primary_key(n));  
}
```

@node\_list [array] : list of nodes

returns [primary key] => [objects] dict

The documentation for this class was generated from the following file:

- src/hiv\_tx\_network.js



## Index

- [\\_calc\\_country\\_nodes](#)  
[HIVTxNetwork](#), [20](#)
- [aggregate\\_individual\\_level\\_records](#)  
[HIVTxNetwork](#), [9](#)
- [annotate\\_cluster\\_changes](#)  
[HIVTxNetwork](#), [10](#)
- [attribute\\_node\\_value\\_by\\_id](#)  
[HIVTxNetwork](#), [10](#)
- [auto\\_expand\\_pg\\_handler](#)  
[HIVTxNetwork](#), [20](#)
- [check\\_for\\_time\\_series](#)  
[HIVTxNetwork](#), [20](#)
- [cluster\\_display\\_filter](#)  
[HIVTxNetwork](#), [10](#)
- [constructor](#)  
[HIVTxNetwork](#), [10](#)
- [Contributor Covenant Code of Conduct](#), [1](#)
- [define\\_attribute\\_binned\\_vl](#)  
[HIVTxNetwork](#), [11](#)
- [define\\_attribute\\_COI\\_membership](#)  
[HIVTxNetwork](#), [11](#)
- [define\\_attribute\\_dx\\_years](#)  
[HIVTxNetwork](#), [12](#)
- [define\\_attribute\\_network\\_update](#)  
[HIVTxNetwork](#), [12](#)
- [define\\_attribute\\_sequence\\_count](#)  
[HIVTxNetwork](#), [12](#)
- [define\\_attribute\\_vl\\_interpretation](#)  
[HIVTxNetwork](#), [13](#)
- [entity\\_id](#)  
[HIVTxNetwork](#), [13](#)
- [entity\\_id\\_from\\_string](#)  
[HIVTxNetwork](#), [13](#)
- [extract\\_individual\\_level\\_records](#)  
[HIVTxNetwork](#), [13](#)
- [extract\\_single\\_cluster](#)  
[HIVTxNetwork](#), [13](#)
- [fetch\\_sequences\\_for\\_pid](#)  
[HIVTxNetwork](#), [14](#)
- [filter\\_by\\_date](#)  
[HIVTxNetwork](#), [14](#)
- [filter\\_by\\_size](#)  
[HIVTxNetwork](#), [21](#)
- [filter\\_if\\_added](#)  
[HIVTxNetwork](#), [21](#)
- [filter\\_singletons](#)  
[HIVTxNetwork](#), [21](#)
- [filter\\_time\\_period](#)  
[HIVTxNetwork](#), [21](#)
- [get\\_reference\\_date](#)  
[HIVTxNetwork](#), [15](#)
- [get\\_ui\\_element\\_selector\\_by\\_role](#)  
[HIVTxNetwork](#), [15](#)
- [hivtrace-viz](#), [5](#)
- [HIVTxNetwork](#), [6](#)
  - [\\_calc\\_country\\_nodes](#), [20](#)
  - [aggregate\\_individual\\_level\\_records](#), [9](#)
  - [annotate\\_cluster\\_changes](#), [10](#)
  - [attribute\\_node\\_value\\_by\\_id](#), [10](#)
  - [auto\\_expand\\_pg\\_handler](#), [20](#)
  - [check\\_for\\_time\\_series](#), [20](#)
  - [cluster\\_display\\_filter](#), [10](#)
  - [constructor](#), [10](#)
  - [define\\_attribute\\_binned\\_vl](#), [11](#)
  - [define\\_attribute\\_COI\\_membership](#), [11](#)
  - [define\\_attribute\\_dx\\_years](#), [12](#)
  - [define\\_attribute\\_network\\_update](#), [12](#)
  - [define\\_attribute\\_sequence\\_count](#), [12](#)
  - [define\\_attribute\\_vl\\_interpretation](#), [13](#)
  - [entity\\_id](#), [13](#)
  - [entity\\_id\\_from\\_string](#), [13](#)
  - [extract\\_individual\\_level\\_records](#), [13](#)
  - [extract\\_single\\_cluster](#), [13](#)
  - [fetch\\_sequences\\_for\\_pid](#), [14](#)
  - [filter\\_by\\_date](#), [14](#)
  - [filter\\_by\\_size](#), [21](#)
  - [filter\\_if\\_added](#), [21](#)
  - [filter\\_singletons](#), [21](#)
  - [filter\\_time\\_period](#), [21](#)
  - [get\\_reference\\_date](#), [15](#)
  - [get\\_ui\\_element\\_selector\\_by\\_role](#), [15](#)
  - [initialize\\_ui\\_ux\\_elements](#), [15](#)
  - [inject\\_attribute\\_description](#), [15](#)
  - [inject\\_attribute\\_node\\_value\\_by\\_id](#), [16](#)
  - [is\\_edge\\_injected](#), [16](#)
  - [list\\_of\\_aliased\\_sequences](#), [16](#)
  - [load\\_priority\\_sets](#), [16](#)
  - [lookup\\_option](#), [17](#)
  - [parse\\_dates](#), [17](#)
  - [populate\\_predefined\\_attribute](#), [17](#)
  - [priority\\_group\\_node\\_record](#), [18](#)
  - [priority\\_groups\\_all\\_events](#), [22](#)
  - [priority\\_groups\\_compute\\_node\\_membership](#), [18](#)
  - [priority\\_groups\\_compute\\_overlap](#), [22](#)
  - [priority\\_groups\\_edit\\_set\\_description](#), [22](#)
  - [priority\\_groups\\_export](#), [24](#)
  - [priority\\_groups\\_export\\_nodes](#), [24](#)
  - [priority\\_groups\\_export\\_sets](#), [24](#)
  - [priority\\_groups\\_find\\_by\\_name](#), [25](#)
  - [priority\\_groups\\_is\\_new\\_node](#), [25](#)
  - [priority\\_groups\\_remove\\_set](#), [25](#)
  - [priority\\_groups\\_update\\_node\\_sets](#), [26](#)
  - [priority\\_groups\\_validate](#), [18](#)
  - [process\\_multiple\\_sequences](#), [19](#)
  - [simplify\\_multisequence\\_cluster](#), [19](#)

- tabulate\_multiple\_sequences, [19](#)
- unique\_entity\_list, [26](#)
- unique\_entity\_list\_from\_ids, [27](#)
- unique\_entity\_object\_list, [27](#)
- initialize\_ui\_ux\_elements
  - HIVTxNetwork, [15](#)
- inject\_attribute\_description
  - HIVTxNetwork, [15](#)
- inject\_attribute\_node\_value\_by\_id
  - HIVTxNetwork, [16](#)
- is\_edge\_injected
  - HIVTxNetwork, [16](#)
- list\_of\_aliased\_sequences
  - HIVTxNetwork, [16](#)
- load\_priority\_sets
  - HIVTxNetwork, [16](#)
- lookup\_option
  - HIVTxNetwork, [17](#)
- Migration for Clusters of Interest (Col) / Priority Sets History Tracking, [4](#)
- parse\_dates
  - HIVTxNetwork, [17](#)
- populate\_predefined\_attribute
  - HIVTxNetwork, [17](#)
- priority\_group\_node\_record
  - HIVTxNetwork, [18](#)
- priority\_groups\_all\_events
  - HIVTxNetwork, [22](#)
- priority\_groups\_compute\_node\_membership
  - HIVTxNetwork, [18](#)
- priority\_groups\_compute\_overlap
  - HIVTxNetwork, [22](#)
- priority\_groups\_edit\_set\_description
  - HIVTxNetwork, [22](#)
- priority\_groups\_export
  - HIVTxNetwork, [24](#)
- priority\_groups\_export\_nodes
  - HIVTxNetwork, [24](#)
- priority\_groups\_export\_sets
  - HIVTxNetwork, [24](#)
- priority\_groups\_find\_by\_name
  - HIVTxNetwork, [25](#)
- priority\_groups\_is\_new\_node
  - HIVTxNetwork, [25](#)
- priority\_groups\_remove\_set
  - HIVTxNetwork, [25](#)
- priority\_groups\_update\_node\_sets
  - HIVTxNetwork, [26](#)
- priority\_groups\_validate
  - HIVTxNetwork, [18](#)
- process\_multiple\_sequences
  - HIVTxNetwork, [19](#)
- README, [6](#)
- simplify\_multisequence\_cluster
  - HIVTxNetwork, [19](#)
- tabulate\_multiple\_sequences
  - HIVTxNetwork, [19](#)
- unique\_entity\_list
  - HIVTxNetwork, [26](#)
- unique\_entity\_list\_from\_ids
  - HIVTxNetwork, [27](#)
- unique\_entity\_object\_list
  - HIVTxNetwork, [27](#)