**Documentation NeTlas code**

Python

parsingBiogrid() = This function performs various operations on the Biogrid genetic dataset. Initially, the tuples are ordered in such a way as to always have the two genes of the interaction sorted in ascending order (useful for avoiding duplicated but opposite tuples). Then the tuples related to the same interaction are grouped. Then various labels and useless information are eliminated from the single fields of the tuples. Then unnecessary tuples are eliminated (with ids missing or not related to the human genome). Finally, the missing complementary ids are added so as to have a triplet (uniprotId, entrezId, ensemblId). This sequence of operations is repeated for all the other datasets with some variations depending on the shape of the single dataset.

geneAggregation() = This function is used to join the various datasets taking into account the order of the tuples and reporting, in a new column of the aggregate dataset, a reference to the single starting datasets containing this tuple. The same process is carried out with drugs and diseases through the related aggregation functions.

writeGenes() = Instances the various Neo4j dbs which will then be populated by the tuples of the datasets obtained in csv version. Then it defines 4 "interaction" functions that perform the writing of the tuple on the Neo4j database (there are 4 different functions depending on the structure of the tuple to be written). The same approach also applies to writeDrugs() and writeDiseases().

writeCluster () = Through this function the ineto dataset "genes" is initially written in a Neo4j db, after which the Neo4j graph is repeatedly modified by eliminating single genes step by step and creating clusters of genes. At each turn, the single node with the highest centrality is taken into consideration, then all the edges of the nodes connected to the chosen node are saved. Subsequently, all the nodes connected to the chosen node are deleted and a "cluster" node is created showing the IDs of the individual nodes just deleted. Finally, the set of edges initially saved is taken and it is reduced by considering only edges of which only one node belongs to the cluster while the other is a single node still present in the graph. These edges are added to the graph, after which we proceed to a new iteration of the process until it is possible to find a single node with centrality at least 15.

Node.js server

loadEdges() = This function is called at the end of the loading of the nodes of the clustered graph on the graph object of cytoscape. Through this function all the edges of the graph are also loaded.

loadCytosnap() = This function is called at the end of the loadEdges() function. Through this function the layout of the graph is calculated according to the various settings (node repulsion, etc.). Returns the graph itself and the position assigned to the individual nodes.

loadEdgesCen() = This function is called at the end of loading the nodes of the non-clustered graph on the graph object of cytoscape. All the edges of the graph are also loaded with this function.

loadCyoscape() = This function is called at the end of the loadEdgesCen() function. Through this function the centrality of all nodes of the non-clustered graph is calculated. At the end of this operation, the 2 to 2 similarity of all drugs and diseases present in the aggregated "drugs" and "diseases" datasets is also calculated.

startapi () = This function is called at the end of the loadCytosnap() function and instantiates an API through which the user can request the download of the obtained datasets. Furthermore, the user can also request the csv version of the current system view, a version that will be calculated on the server side and sent via the API instantiated by this function.

Javascript

* **graph.js**

mouseclickclusternode() = When the user clicks on a cluster node, the area around the node is expanded and the cluster is broken down into the individual nodes belonging to it. All the nodes and edges of the cluster are created, and depending on the options, you can view these elements in the general context of the network.

mouseclicksinglenode() = When the user clicks on a single node, if this has not already been clicked, all the edges connected to it are displayed.

mouseovernodehandler() = Creates two different tooltips depending on the type of node.

sortTableTooltip() = Sorts the list of nodes displayed in the cluster tooltip according to the selected column.

writedistr() = Creates the linear distribution of nodes within the cluster tooltip.

writedistrmin() = Creates the miniature of the linear distribution of nodes within the cluster tooltip.

* **sidebar.js**

writelist(ddswitch,order) = It creates and updates the list of sidebar items. "ddswitch" indicates the type of elements to be displayed (drugs or diseases), "order" indicates the display order of the table tuples (asc or des in relation to one of the columns of the table).

clicksideelem(name,type,numGenes,intedges,extedges) = Based on the selected element (drug or disease), it obtains the list of nodes highlighted by the element and various information. If the nodes are all present in the network it calls the displaySidebarElem() function, otherwise it saves all the nodes that are in the clusters in a set and calls the extractnode() function.

extractNode(subcluster, name, type, numGenes, intedges, extedges, elemgenes) = Takes as input the set of genes to be extracted from the various clusters and extracts the first element of the set, after which the function is called recursively on the current set (after having deleted the item you just checked out). When all the elements have been extracted from the various clusters, the displaySidebarElem() function is called.

displaySidebarElem(name, type, numGenes, intedges, extedges, elemgenes) = the function adds those targeted by the selected element to the set of highlighted genes and updates the display of the entire set of genes (color and size). Then it updates the hover and click events on nodes and edges and adds the drug or disease just selected to the list of displayed items.

searchside() = Filter the list of sidebar items based on user input.

resetStyle() = Returns the display of the entire system to its initial state

checkboxsets() = show or hide some types of genes depending on the selection

* **livestats.js**

edgegraph() = Receives as input an array of 3 values (number of low/medium/high reliability interactions) relating to the genetic interactions between genes highlighted by the selections made on the sidebar. It shows an interactive barplot of the 3 intervals with the possibility to visualize the single groups through hover event.

* **filters.js**

resizenodes () = According to the selection of the Nodes/Clusters slider and the "Size X 2" checkbox, it updates the size of the network nodes avoiding to modify the nodes highlighted by the drugs and diseases selections.

nodevisibility () = Show or hide nodes and edges based on the selections of the checkboxes related to the 6 genes datasets.

colornodes () = Updates the colors of the nodes based on the selections of the single genes datasets avoiding to modify genes highlighted by selections of drugs and diseases through the sidebar.

* **drugdiseasedivs.js**

addDrug (name, numgenes, intedges, extedges, elemgenes) = Adds the selected drug to the displayed drugs menu, reporting various information on the specific element (the same operation applies to the addDisease() function).