저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer
MONGKIE: Integrated network analysis and visualization platform for multi-omics data

Multi-omics 데이터 분석을 위한 통합 네트워크 분석 및 가시화 프로그램 개발

2016년 2월

서울대학교 대학원
자연과학대학 협동과정: 생물정보학전공
장 영 준
Abstract

MONGKIE: Integrated network analysis and visualization platform for multi-omics data

Yeongjun Jang
Interdisciplinary Program in Bioinformatics
The Graduate School
Seoul National University

Network-based integrative analysis is a powerful technique for extracting biological insights from multi-layered omics data such as somatic mutations, copy number variations, and gene expression data. However, integrated analysis of multi-omics data is quite complicated and can hardly be done in an automated way. Thus, a powerful interactive visual mining tool supporting diverse analysis algorithms for identification of driver genes and regulatory modules is much needed. Here, we present a software platform that integrates network visualization with omics data analysis tools seamlessly. The visualization unit supports various options for displaying multi-omics data as well as unique network models for describing sophisticated biological networks such as complex biomolecular reactions. In addition, we implemented diverse in-house algorithms for network analysis including network clustering and over-representation analysis. Novel functions include facile definition and optimized visualization of subgroups, comparison of a series of data sets in an identical network by data-to-visual mapping and
subsequent overlaying function, and management of custom interaction networks. Utility of MONGKIE for network-based visual data mining of multi-omics data was demonstrated by analysis of the TCGA glioblastoma data. The result of case study showed that network-based analysis in MONGKIE can identify and extract biologically relevant genes and pathways that play important roles in GBM tumorigenesis. MONGKIE is a java-based open-source application built on top of plug-in architecture, thus being platform-independent and easily extensible with additional functionalities. It is available under GNU AGPL v3 license at http://yijiang.github.io/mongkie.

............................................................

keywords : Network visualization, Network modeling, Graph clustering, Omics data analysis, Over-representation analysis 

Student Number : 2014-20341
CONTENTS

1. INTRODUCTION --------------------------------------------------------------- 1

2. RESULTS -------------------------------------------------------------------- 3
   2.1. Integration of Interaction Networks ---------------------------------------- 3
       2.1.1. hiPathDB ------------------------------------------------------------- 3
       2.1.2. Interaction Manager --------------------------------------------------- 4
   2.2. Network Visualization ---------------------------------------------------- 6
       2.2.1. Visual representation of biological entities and interactions ------- 6
       2.2.2. Pathway visualization ------------------------------------------------- 8
       2.2.3. Visual editor UI and Data-to-Visual mapping ------------------------ 10
       2.2.4. Graph layouts -------------------------------------------------------- 12
       2.2.5. Exploring network ----------------------------------------------------- 14
   2.3. Network Analysis ---------------------------------------------------------- 15
       2.3.1. Network clustering ---------------------------------------------------- 16
       2.3.2. Gene expression overlay --------------------------------------------- 18
       2.3.3. Over-representation analysis ------------------------------------------- 20
   2.4. User Interface ------------------------------------------------------------- 22
   2.5. Import and Export Data ----------------------------------------------------- 24

3. CASE STUDY ------------------------------------------------------------------ 26
   3.1. Cancer Omics Data -------------------------------------------------------- 26
   3.2. Methods ------------------------------------------------------------------ 27
       3.2.1. Extraction of GBM-altered sub-network and network clustering -- 27
       3.2.2. Statistical test for significant linker genes ------------------------ 29
       3.2.3. Network visualization and enrichment analysis ----------------------- 29
   3.3. Results ------------------------------------------------------------------ 29
4. IMPLEMENTATION .................................................. 30
  4.1. Software Architecture ......................................... 31
  4.2. RESTFul Web service API .................................... 34
  4.3. Graph Database ................................................ 35
5. CONCLUSION ......................................................... 35
  5.1. Availability ..................................................... 36

REFERENCES ............................................................ 37

국문초록 ................................................................. 41
INTRODUCTION

In the recent years, high-throughput studies, including genomics, transcriptomics, and proteomics, have been resulting in a greatly increased volume of complex, interconnected data [1]. One of the common but powerful approaches to gain insights into the underlying complexity and dynamics of these data is to analyze them on the context of various biological networks - e.g. protein-protein interactions, biological pathways. Such analyses essentially require integrated network analytics together with insightful visualization for both the reasoning and understanding of multi-omics data from the perspective of biological networks, without being threatened by the complexity of data [2,3].

There are many network analysis and visualization tools for systems biology [4,5], however it still remains considerable challenges that are not adequately addressed [6]. One example of such challenges is the scalability problem of huge networks. In order to interactively manipulate and interpret such huge networks, it is essential to divide the complete network into many small sub-networks by manual selection or algorithmic module detection, automatically adjust their layout, and change the visual representations associated with nodes in subsets [7]. Another example is the overlay of temporal or dynamic contextual attributes onto the nodes and edges. To focus this challenge, an animated visualization or 'small multiples' [1] can be an effective approach to capture the actual complexity of network dynamics, for example, changes of gene expression levels over multiple treatments or time points [3].

To fulfill these requirements, we present MONGKIE, Modular Network Generation and Visualization Platform with Knowledge Integration Environment, which is an integrated network visualization and analysis platform, allowing researchers to explore and analyze inter-connected biological data in an interactive manner with various knowledge integration
environment.

Our new visualization platform provides the sophisticated data model of visualization to represent different types of biological entities and relationships between them. All visual properties of them can be customized through the integrated visual editor UI, or set continuously or discretely based on their contextual information by data-to-visual mapping. MONGKIE also guides the exploration of networks in many interactive ways including selection, zooming, panning, highlighting, dragging, grouping or partitioning, search, filtering, and dynamic network expansion and deletion.

MONGKIE is designed for both the visualization of biological networks and the analysis of these networks with a seamless integration between the two procedures. It incorporates various software modules in the platform for the knowledge integration and network analyses, including import and export data, extraction and build of context-specific sub-network, interaction manager, network clustering, GO over-representation analysis, gene expression overlay, and pathway data integration and visualization.

As a case study, we performed an integrated network analysis to identify frequently altered network modules and candidate driver mutations in the TCGA study of glioblastoma mutiforme [8], utilizing some of major functionalities of MONGKIE. As a result, two highly ranked modules identified by network analysis corresponded very closely to critical signaling pathways prior known to GBM biology, such as EGFR/PI3K signaling pathway, and TP53/RB1 DNA damage response and Cell Cycle, which shows that network-based omics data analysis in MONGKIE can facilitate the study of structural pattern of altered genes and aberrant pathways in cancer.

MONGKIE is a java-based application built on top of the NetBeans platform [9] that supports modular (plug-in) architecture, thus being platform-independent and easily extensible with additional functionalities with little programming effort. Download, documentation, and source code are available at http://yjjang.github.io/mongkie.
RESULTS

Integration of Interaction Networks

Systems biology aims to study the relationships between molecular or functional components to gain insights into the underlying complexity and dynamics of biological processes from the properties or structure of the interactions [10]. This generally requires interaction data from diverse sources - e.g. protein-protein interactions, signaling or metabolic pathways - to be integrated, and analyzed together with data derived from high-throughput experiments, such as genomics, transcriptomics, and proteomics. One example of this integrative approach is to analyze altered genes in specific disease samples on the context of biological networks. By projecting the list of mutated, amplified, or deleted genes onto biological networks, one will find statistically significant subsets of related genes that are closely clustered as network modules [11,12] or biological pathways affected by such genes [13].

Network-based multi-omics analyses can thus provide important insights into complex biological mechanisms and processes - e.g. the biology underlying disease etiology, or progression of several cancer types [14] - and reliable pathway databases as well as high-coverage protein interaction dataset are essential for such an analysis. MONGKIE is integrated with hiPathDB [15] which is the Human Integrated Pathway Database described below, MiMI [16], STRING [17], and ReactomeFI [18] interaction data sources, in order to provide built-in background interactions for such a network analysis. Furthermore, it provides an interaction manager that allows users to incorporate external interaction data from multiple sources into the process of network analysis and visualization, and to manage them through an integrated UI.

hiPathDB
With heterogeneous biological pathway data sets in the diversity of potential data formats available, the integration of pathway resources has become an important issue in utilizing these resources systematically and efficiently. In order to utilize these resources to interpret and analyze genomic data using network-based analysis methods, the information stored in dispersed data repositories needs to be linked and combined in efficient ways, and strongly required to unify heterogeneous interactions in different pathway data sources into the one general network model.

Here, MONGKIE provides a built-in software module for highly interactive pathway visualization and exploration, tightly integrated with hiPathDB \[^{15}\]. hiPathDB is an integrated pathway database that combines curated human pathway databases of NCI-Nature PID \[^{19}\], Reactome \[^{20}\], BioCarta \[^{21}\] and KEGG \[^{22}\]. hiPathDB provides two different types of integration. One is the pathway-level integration - shown in Figure 3 of \[^{15}\] - which is conceptually a simple collection of individual pathways, was achieved by devising an elaborate model that takes distinct features of four databases into account, and subsequently reformatting all pathways in accordance with our model - shown in Figure 2 of \[^{15}\] - while maintaining molecular details of signaling processes. Another is the entity-level integration - shown in Figure 4 of \[^{15}\] - creates a single unified pathway, super-pathway, that encompasses all pathways by merging common components. Even though the detailed molecular-level information such as complex formation or post-translational modifications tends to be lost in the entity-level integration, such integration makes it possible to investigate signaling network over the entire pathways and allows identification of pathway cross-talks. Therefore, the unified super-pathway achieved by entity level integration facilitates the network-based analysis and navigation from the perspective of biological pathways.

**Interaction manager**

In addition to biological pathways, there are other types of biological
interactions that are well modeled by simple binary, pair-wise graphs, such as PPIs, TF-target, miRNA-target and genetic interactions, and they are also essential for the network analysis of genomic data. MONGKIE provides a elegant way to use these types of network as background networks for the exploratory network-based analysis and visualization, through the Interaction Manager UI, shown in Figure 1A.

**Figure 1.** (A) Interaction Manager UI. (B) demonstrates the procedure of dynamic network construction starting with a small part of genes of interest (orange nodes).

Imported and managed interaction data sources can be utilized for various exploratory and analytical purposes, such as network generation from a scratch by search query, dynamic network exploration, expansion and filtering of interactions by their sources. For example, there are cases when the size of a interaction network is so huge that it is impossible to handle and visualize the complete network at once. It is recommended for such situations to follow the approach of a classical top-down exploration, in which rather than display the entire network in one display, initially start with a small part of nodes in the network, e.g. deferentially expressed genes, and then iteratively build a larger network by allowing the user to successively expand particular nodes of interest with their further interactions and neighbors.
The procedure for this strategy of top-down exploration provided by the Interaction Manager is shown in Figure 1B. It allows the user to expand interactions through the context-sensitive right-click menus on a selected nodes of interest (orange nodes), as well as just to connect existing nodes using the interaction dataset from a data source by selecting the check-box of that source in the interaction manager UI. Later, the user may delete neighbors out of the interest to reduce complexities of the network such that he/she focuses on the region of interest. Furthermore, each newly added node resulting from a expansion action is placed in an appropriate position with an animation, well incorporated into the force-directed layout algorithm \([23,24]\), and this allows the user to easily preserve the so-called mental map \([25]\) during exploration of the network.

MONGKIE is distributed with a built-in interaction sources of MiMI \([16]\), STRING \([17]\), and ReactomeFI \([18]\), and each source provides access to the knowledge and data integrated from numerous protein-protein interactions and pathway databases. Users also can import more their own binary interaction dataset into the interaction manager either from files using standard formats - e.g. GraphML \([44]\) and CSV files - or from remote databases.

Additionally, visual styles of each interaction source can be fully customized through the Visual Style editor, and they will be persisted, the latter means that customized visual properties of edges from the interaction source as well as imported interaction dataset themselves is locally stored and will be available for styling edges on the next run of the application.

**Network Visualization**

**Visual representation of biological entities and interactions**

The graphical representation of biological networks includes how to intuitively visualize a set of connected nodes (or vertex) corresponding to biological entities, including genes, gene products (protein, transcript factor, miRNA,
etc.), small molecules (compound, metabolite etc.), protein family and complex, and their links (or edges), such as physical or genetic interactions, regulatory events (transcriptional and translational activation or inhibition, phosphorylation, etc.), co-expression, shared protein domain, complex formation, trans-location and other biochemical reactions.

**Figure 2.** illustrates how MONGKIE visually represents biological entities and relationships between them, by visualizing an example network, Glucocorticoid receptor regulatory network, which is a signaling pathway curated by the NCI-Nature PID [19].

MONGKIE provides the sophisticated data model for visualization of biological networks with advanced graph drawing techniques, and therefore can represent different types of biological entities and interactions between them with out-of-the-box visual styles, shown in Figure 2. Both nodes and edges differ in their style according to their biological meaning. The style of nodes - e.g. label, font, shape, color, size and icon image - shows the type and state of biological components, and edges linking a relation participant with the information about the role also differ in their style - e.g. shape or
thickness or color of lines, shape or color of arrows as well as label and font.

**Pathway visualization**

MONGKIE provides a built-in software module for pathway visualization, which supports visual analytics and exploratory studies of metabolic or signaling pathways in an interactive fashion tightly integrated with the human integrated pathway database, hiPathDB [15].

Like other types of biological interactions that can be modeled as regular binary-graphs, where interactions are between exactly two interactors, biological pathways also can be represented as a graph consisting of nodes and edges. Although these simple models have been shown to yield biological insights, biological pathways can be complex multi-modal or hyper-graphs [2], in which an edge could connect an arbitrary number of nodes or might connect a node to another edge. Therefore basic graph representations are incapable of modeling more complicated and interwoven biochemical reactions that involve more than two sub-components, e.g. the formation of protein complexes, and interactions that controlled by external controllers.

In order to visually capture such complex biochemical events, MONGKIE presents an optimized pathway visualization based on the sophisticated visualization model taking into consideration these important domain-specific knowledges.

Figure 3 illustrates how diverse types of complex components and their relationships in biological pathways are visually represented in MONGKIE. A node representing a biological molecule in a pathway visualization may be either a protein, family, complex, dimer, enzyme, other small molecules (e.g. compound, metabolites), or especially a super-node which is a hierarchically decomposed composite node representing adjacency and inclusion relationships, e.g. the hierarchical modeling of protein complex assembly can be flattened.
by splitting nodes representing complexes into their own individual members (see Figure 3A). An edge in a pathway visualization represents a relationship or some form of interaction between nodes. The interaction could be one of many types: complex formation, activation, inhibition, aggregation, trans-location, catalysis, chemical modification, etc. In most cases, single-line connections are insufficient to capture the whole range of information contained in a biological pathway, because biological entities are often linked by more than one type of relationship. In such cases, multi-edge networks offer the possibility to link two entities by multiple edges, in which every edge having a different meaning. Also, hyper-edge connects a node to another
edge, e.g. an inhibitory interaction (edge) actually indicates a biological process by which one molecule (node) might prevent some other interaction (edge) from occurring \[^2\] (see Figure 3B). All these different types of nodes and edges in the pathway visualization are visually represented with their distinct visual styles and additional informations, including their sub-cellular location, cellular state - e.g. activated, inhibited, phosphorylated, etc. Each component of pathways have informations about the originating data source, and hyper-links to the corresponding web pages is presented at the table view to link relevant databases - eg. NCBI, PubChem, etc.

The force-directed layout algorithm \[^{23}\] is optimized for the virtually automatic placement of components in the pathways.

Additionally, MONGKIE provides an interactive way for the dynamic and step-wise exploration of pathway interactions by utilizing the integrated super-pathway in hiPathDB like the procedure of dynamic network expansion implemented in Interaction Manager UI (see Figure 1B). The exploration may start at particular genes or proteins of interest, initially loaded by search query (nodes with yellow stroke), and the network may then be extended step by step with their additional linked nodes in the super-pathway.

**Visual editor UI and Data-to-Visual mapping**

Every component - nodes, edges, and groups - of the visualization model has a set of visual properties, including label text, text font, stroke, shape, size, color, line shape and width etc. These visual styles of them can be fully customized individually through the integrated Visual Editor UI which allows the user to edit them in any way the user desires with numerous predefined palettes. Actual UIs for editing visual properties of nodes, edges, and groups are shown in Figure 4. These properties also can be changed globally with the Display Options UI, then one can save current graphical representations of each component as a visual style, later can reload and apply the saved one to the new visualization.
Each component in the network can have associated data attributes possibly describing visual properties of them. In addition to selective and manual editing of visual styles, MONKGIE also provides a very useful way to

**Figure 4.** Visual editor UIs for (A) nodes, (B) edges, and (C) groups with their members.

**Figure 5.** Using an example network modeling cross-talks between biological pathways, (A) the number of genes in each pathway was continuously mapped to visual properties (size and color) of the corresponding pathway node. (B) data sources (KEGG or BioCarta) of each pathway were discretely mapped to node colors.
automatically set visual aspects of components based on their data attributes. This continuous or discrete Data-to-Visual attribute mapping allows researchers to synoptically view multiple types of data in a network context (see Figure 5). Data-to-Visual mapping lets the user, for example, load omics data from various high-throughput experiments, e.g. expression profiles, and visually project them into the network by automatically transforming data to some graphical attributes, e.g. color, size, visibility etc.

Finally, all data sets and visual properties in the network visualization can be saved using the MONGKIE specific visual graph file format (VLG), later one can reopen the saved one in order to go on the previous analysis.

**Graph layouts**

Graph layout algorithms are used to place graph nodes and edges in various geometric distribution for the clarity and readability of networks such that the number of edges crossing minimized and that the layout represents the overall structure of the network legibly. A variety of state-of-the-art layout algorithms are implemented in MONGKIE, including Circular, Grid, Fruchterman-Reingold [26], Radial Tree [27], and Force-Directed [23] layouts, both for efficiency and quality (see Figure 6).

Like most of other typical network visualization softwares, we try a force-directed layout first because this layout can usually well organize most biological networks based on the non-deterministic algorithm that lets forces between nodes influence the position of the node in the network. All nodes exert repulsive force on the others whereas connected nodes are attracted to each other. After several iterations in which the positions are adjusted according to the calculated force, the layout stabilizes, keeping edge-crossings to a minimum [28,24]. It also visually animate the process for laying out the network so that one can watch nodes in the network incrementally being placed in optimum positions and can terminate the algorithm when a good layout is obtained. However, this layout quickly becomes inadequate if the
size and complexity of network are too larger to handle and interact. For such cases, MONGKIE provides an opportunity to go without animations or to choose other faster but simpler one - e.g. circular, grid etc.

Each layout algorithm can be easily started, canceled, and customized through the unified layout control UI. Highly configurable layout algorithms also allow the user to change layout settings in real-time, and therefore dramatically increase user feedback and experience. For instance, settings of the force-directed layout, including gravity, spring and forces, can be configured and immediately applied even while the algorithm is running.

In addition to automatic layout algorithms, MONGKIE offers another way to interactively change the layout of the network by manually dragging each node or user-defined groups into any positions, and this is very useful in fine-tuning the automatic layout or emphasizing important nodes or biologically significant regions in the network by geometrically separating

Figure 6. A variety of state-of-the-art layout algorithms are implemented in MONGKIE, including Circular, Grid, Fruchterman-Reingold, Radial Tree, and Force-Directed.
them from other parts.

**Exploring network**

In order to substantially facilitate network navigation and information extraction, MONGKIE provides sophisticated methods to explore networks in highly interactive ways, including searching, filtering, grouping, manual or automatic node selection, highlight, dragging, zoom and panning the display, overview of the complete network, and lastly data table view (see Figure 10) displaying attributes of nodes and edges in a tabular format.

MONGKIE provides easy-to-use search functions for the loaded network. One can enter any keyword or regular expression to search all data attributes held in the nodes and edges. The matching nodes or edges immediately highlighted in the visualization and selection one or more ones in the visualization propagates to the selection of the relevant rows in the data table scrolling to them, and vice versa. Furthermore, the network can be filtered down to interactions meeting a given filtering constraint according to their attributes. For instance, the network may be filtered to show only proteins occurring in particular locations, thus reducing the network complexity and restricting the one's attention only to interactions within a given sub-cellular location, and this could greatly improve the visual perception of complex biological networks.

Given the large and complex networks, one common approach to interpret and visualize such networks is trying to display the complete network on the screen and providing functionalities to zoom, pan, and overview of the network for exploration. Like many other network visualization tools, MONGKIE provides these basic techniques for network navigation too. However, as the size and complexity of interactions grow, it is increasingly impossible to understand underlying structures and extract biological insights from such huge networks just using those basic navigation techniques. Another improved strategy is to dissect the complete network into smaller
sub-networks that are manageable, biologically significant regions that can be understandable [1]. These sub-networks are typically defined as, for example, sets of proteins that are occurring at the same sub-cellular location, or that belong to similar functional GO [29] terms, or that are members of a densely connected cluster identified through the established clustering methods, e.g. MCODE [30], MCL [31], and clique percolation [32] etc. The resultant sub-networks are typically of a size that is more amenable to visualization and analysis.

In order to specifically support of these processes of dissection, MONGKIE provides a variety of ways to define groups of functionally or topologically related nodes, including enrichment analysis for functional modules (Figure 9), clustering analysis for topological clusters (Figure 7), grouping by manual selection, or automatic partitioning of nodes according to their attributes. The defined groups or sub-networks are visually represented with distinct styles and importantly laid out separately from other parts in a way that automatically attracts each other nodes in a group while repelling other groups (see Figure 7), and this geometric separation is essential to focus only on particular groups without being disturbed by unnecessary or noisy interactions. Additionally, one can create a new visualization of each sub-network, then analysis it independently from original one from which it derived.

MONGKIE is developed specially with these grouping functionalities in mind as one of main development goals so that it can be used to help dissect large interactions, thus reducing the overall complexity, and focusing on smaller but biologically interesting parts to gain biological insights without being overwhelmed by complexity and noises in the network.

**Network Analysis**

MONGKIE is designed for the both visualization of biological networks and
also analysis of these networks with a seamless integration between two procedures. MONGKIE includes several of network analytical methods, such as network clustering, overlay of expression profiles, and over-representation analysis, to recognize putative functional or structural patterns, and uncover interesting biological meanings from biological interaction networks.

**Network clustering**

Fundamentally network-based approaches in systems biology are based on the hypothesis that biological entities rarely act alone in the cell, instead they interact spatiotemporally with others, forming modules, in order to perform specific cellular functions \([11,33]\). Graph-based clustering algorithms that are mainly developed in graph theory and recently computational systems biology have been successfully applied to the study of detection of protein complexes \([34]\) or families \([35]\), or identification of functions of uncharacterized proteins \([36]\), or extraction of co-expressed clusters from co-expression networks \([37]\), etc., and shown to obtain good performances for extracting such modules from a variety of biological interactions.

These so-called 'network clustering' methods are also known to be less susceptible to inherent false-positives and more accurately predict modules made up of functionally relate nodes rather than conventional iterative pair-wise clustering, where individual relationship - e.g. sequence similarity, co-expression - between two biological entities is investigated without considering structural patterns in their interactions with neighbors \([38]\).

MONGKIE currently incorporates two popular structure-based network clustering algorithms, including MCODE (Molecular COmplex DEtection algorithm), MCL (Markov CLustering algorithm), and these make it easy to find densely inter-connected, thus functionally related nodes in biological interactions, exploit both local and also global structural patterns, and visually map them onto the network. MCODE is a graph theoretic clustering algorithm for finding molecular complex in large protein interaction networks.
The MCODE plugin, which is implemented by porting from the pre-existing plug-in in Cytoscape [39], identifies clusters by finding regions of significant local density. MCL is a fast and scalable unsupervised clustering algorithm for graphs based on simulation of the flow in the graph [31]. Because MCL is a robust state-of-the-art general purpose clustering algorithm for large graphs, it can be applied to any complex biological networks, e.g. protein functional relationship network to look for candidate cancer driver mutations and relevant functional modules they belong to [18].

One can define resultant network clusters (or modules) as groups, therefore they, as described in 'Exploring network', can be visually organized and laid out onto the network with distinct visual schemes, as well as displayed in a tabular format. As shown in Figure 7, each cluster and its members are visualized with a distinct color and shape according to their cluster membership, also laid out using the optimized force-directed layout algorithm that automatically attracts each member in a cluster while repelling other

![Figure 7](image_url)

**Figure 7.** Demonstration of the procedure of identification network clusters, organization them onto the network with distinct visual schemes, and how nodes in the same cluster are laid out coherently.
clusters. This helps users to visually interpret the coherence of clusters in the context of the network, that cannot be easily obtained by simply examining lists of clusters or their membership. Cluster nodes in the network can then be manipulated just like other general nodes for any exploratory purposes, e.g. one can select, then drag them in order to place in desired positions. One also can create a new visualization for a sub-network made up of nodes and edges in a cluster.

These features of network clustering and grouping in MONGKIE can facilitate to analyze or visualize the large data set of biological interactions in a more modular way that can provide biological insight into both local and also global structures in networks between biologically related nodes - e.g. a pathway affected by mutated genes, proteins belonging to a same complex or family, or cross-talk among biological pathways etc.

Additionally, MONGKIE provides out-of-the-box APIs, SPIs, and UIs for plug-in developments (see Figure 2), hence any new clustering algorithm can be easily added with little programming effort.

**Gene expression overlay**

Gene expression data obtained by micro-array experiments or RNA-seq techniques can provide powerful insights into underlying cellular states and dynamics when they are well integrated into the context of biological networks. Therefore, overlaying expression profile data onto a visualized network is an essential way in identifying a set of genes or proteins that share a related pattern of expression under a particular condition - e.g. co-regulated gene sets and their interactions in a certain disease sample - or capture dynamic changes of their expression levels over a range of time points or different conditions [1]. For these analyses, we require tools that allow users to visually represent expression profiles of the nodes in the network according to their expression level, thus better perceive the dynamic mechanism of a underlying biological system being guided by visualized
expression patterns or changes.

MONGKIE provides powerful functionalities for visual analyses of high-throughput omics data in the context of networks, in particular for the gene expression data analysis of time series or multiple conditions. It supports dynamic visual representations, including a color gradient, size, and label of relevant nodes, that are easy to separate and interpret independently in order to depict the corresponding expression profile ratio of gene or protein nodes. Once expression profiles are imported as data attributes of nodes using the CSV file format, then a heat map visualization, which is used in a wide range of tools for the process of gene expression visualization [1], appears to display those expression data as a ratio-based graphical matrix, as well as incorporated within the graphical representation of nodes in the network (see Figure 8).

![Figure 8](image)

**Figure 8.** Demonstration of the procedure of overlaying of gene expression data from multiple experiments onto genes in the network and the UI for capturing dynamic changes of their expression levels over a range of different experiments.

An important challenge for expression data analysis is to interpret gene
expression data produced from more than one condition, for example, time series experiments, or multiple perturbation studies. Therefore, it is necessary to consider all time points or conditions in order to detect temporal patterns and their changes in gene expression profiles whose values vary over time or different condition. This requires a selective or sequential visualization of multiple expression levels in the network context. MONGKIE allows users to incorporate such dynamics of gene expression profiles into the loaded network visualization by offering a way to change the visual mapping - e.g. color, size - of nodes to reflect the expression levels of a particular time point or condition according to the user selection.

Figure 8 illustrates an example of this process, where gene expression profiles from multiple micro-array experiments in six cell lines were loaded into the heat map display, and also mapped on corresponding nodes in the network.

A expression level of a particular experiment can be navigated using a sliding bar UI on the bottom of the window, and also the navigation process can be animated, as introduced in some tools \cite{40,41}, by automatically switch to visualization of the next experiment with a predefined time interval. This is well suited to investigating by eye changes of expression levels within a group of interesting genes - e.g. genes that share a same functional term, members of a network cluster, or deferentially expressed genes - over given experiments. Furthermore, by arranging visualization windows in a grid (see 'User Interface'), one can in parallel compare multiple visualizations of the same network, where each visualizes for its navigating experiment - known as the 'small multiples' approach \cite{1}.

**Over-representation analysis**

A group of interesting genes in the biological network, e.g. a module of potentially related genes has been found using the above clustering analysis, may be investigated to find biological pathways or other functional categories
like GO \cite{29} terms, where they are significantly over-represented. This approach, so-called 'enrichment analysis', is widely used in order to study the gene set for their over-representation in certain annotation classes that are usually related to biological functions of those genes \cite{42}.

**Figure 9.** Demonstration of the procedure of enrichment analysis, and visual annotation of relevant nodes or regions (e.g. clusters) in the network with significantly over-represented functional terms.

MONGKIE provides a pipeline for this analysis, shown in Figure 9, where researchers perform a statistical test for enrichment or depletion using the provided annotation database, including GO or pathway, in order to identify over-represented functional terms with statistically significance from the selected set of interesting genes in the certain network region. The result of analysis is displayed in table views with both a list of resultant functional terms and also statistics of the enrichment analysis, including number of enriched terms, number of population genes, number of query genes, and detailed information about each annotation term. In the result table, each row of resultant term has a background color that is mapped to its corresponding p-value - that was calculated using the hyper-geometric testing, and might be
adjusted by one of the following multiple testing correction methods: Bonferroni, Bonferroni-Holm, Benjamini-Hochberg, or Benjamini-Yekutieli - and the set of resultant term can be reduced based on user's cut-offs and ranked by their statistical p-values.

Another optimized way to show the result of enriched functional terms in the hierarchical structure, e.g. Gene Ontology, is to display them in a tree table. In the tree table view, intermediate terms for tree hierarchy are automatically included, and users can expand or collapse any sub-tree in the view. Therefore it allows the user to interactively investigate terms according to specific biological complexity within the hierarchical structure as well as their significance.

Once a list of resultant term is displayed in a tabular format, users can select significant terms of interest, then visualize the group of relevant nodes or regions annotated with those terms in the network context by mapping distinguishable visual aspects to them. This visual mapping of functional terms in the context of the network has the distinct advantage of allowing the user to quickly identify by eye both biological functions of certain parts and also higher-order interactions between those parts in the network that would not be obvious without this type of visual representation.

Like other parts of MONGKIE, it is also enable to easily add a new annotation database or statistical analysis methods for it by developing a plug-in that implements service provider interfaces (SPIs) for the enrichment analysis (see Figure 2).

**User Interface**

Figure 10 illustrates the main graphical user interface of MONGKIE. The main application window is made up of several dynamic views using the NetBeans [⁹] window system that lets the user maximize and minimize, dock and undock, auto-hide and sliding, and drag-and-drop windows for well
organizing views inside the main window.

Figure 10. Graphical User Interface consisting of a main visualization display with other windows, including analysis windows (network clustering, expression overlay, and enrichment analysis), visual editor, data-to-visual mapping window, data tables, statistics view, and an overview window.

The network visualization window is placed in the center with many context-sensitive menu items which allow users to easily communicate with other windows in an interactive way, access to currently important functionalities, especially to the interaction manager for the dynamic network expansion and deletion (see Figure 1B). Views in the left side of the main window includes GUIs for those functionalities that require user's input or control actions, such as visual editor, data-to-visual mapping, network clustering, enrichment analysis, and graph layout. Those in the right side display a variety of contextual informations, including overview of the complete network, properties of selected nodes or edges, heat map visualization for gene expression data, and graphical charts that show various statistics in the visualization - e.g. groups or clusters in the network, node visibility after filtering etc. Those views that need to be organized in a tabular format together with search and filtering functionalities, such as the
list of nodes or edges in the network, the result of enrichment analysis, are
placed in the bottom of the main window.

While other views exists only once, the network visualization window can
have multiple instances for different visualizations. This allows users to in
parallel compare any number of visualized networks from different conditions,
e.g. gene expression levels from multiple experiments, by tiling multiple
visualization windows of the same network in a grid, where each one
visualizes the information for its own condition.

Settings of the windows in the application, such as the size, position, and
arrangement, are fully customizable by resizing or drag-and-drop. The
flexibility of window management facilitates coherently working with multiple
windows or views for the process of network visualization, navigation, or
analysis. Also, these window settings are persisted across restart of the
application, and later one can restore them to default settings.

Many parts of UI components and UX (User eXperiences) in MONGKIE
are strongly inspired by Gephi [43]. We also use the prefuse [24] java library
for the graph data structure and interactive visualization.

**Import and Export Data**

Interaction data sets can be loaded and stored using different file formats,
including GraphML (Graph Markup Language) [44] and CSV
(Comma-Separated Values). GraphML (Graph Markup Language) is a
comprehensive and easy-to-use file format for graphs. Its main features
include supports of storing directed or undirected or mixed graphs,
hyper-graphs, hierarchical graphs, graphical representations, references to
external data, and application-specific attributes. CSV (Comma-Separated
Values) stores tabular data sets in a plain-text form, and is a basic file
format that is widely supported by a wide range of scientific applications for
loading or saving their data sets.
MONGKIE comes with built-in parsers for those files to read a graph or attributes of its nodes and edges from the given data file. GraphML file fully stores the structural information of a graph, hence it is quite straightforward to import a graph using that. CSV file, however, requires some extra steps to import a graph. As a first step, one need to prepare two files - one containing nodes and their attributes, and another containing an edge list and attributes. The CSV file containing nodes needs to include a column containing unique node IDs. The edge list CSV should include columns for 'source' and 'target', containing node IDs of the start and end node for each edge. By simply going to File -> Import from CSV Files, a dialog window for importing both files will be appear, then the wizard UI will guide you to remaining steps. In following steps, one can make several modifications to properties of the importing graph - e.g. one can rename column names, specify if edges are 'directed' or 'undirected', select columns of 'source' and 'target' for edges etc. Also, the type of each column in the node or edge table will be appropriately inferred according to the literal expression of their values - decimal data as 'Integer', floating points as 'Double', comma-separated text fields as 'String Array'. When all steps are gone, the report dialog finally shows the summary of the imported graph, including number of nodes and edges, type of graph, issues occurred during the importing process etc. Furthermore, MONGKIE also use the CSV file as a input format in order to import expression profiles, or additional attributes of nodes or edges through the similar UI concepts applied to importing graphs.

MONGKIE is capable to export the graph in the visualization to 1) the GraphML file that contains all graph elements in one formatted file, or 2) two CSV files - each file stores the list of nodes and edges in a tabular format respectively. Exported graph serializations can be later used in not only MONGKIE itself, but also external graph visualization softwares [39,43]. Once satisfied with the network visualization, one can save it in one of multiple image formats - vector graphics, like SVG (Scalable Vector
Graphics), EPS (Encapsulated PostScript), or bitmap images, like PNG, JPG, GIF, and BMP files.

MONGKIE also provides its own visual graph file format (VLG) that stores all visualization-wide properties - e.g. visual representations of each element, graph layout, node positions or visibilities, display scale, etc. - as well as data attributes of the whole graph in the network visualization, therefore one can save the current analysis in the VLG file, then later reopen it for continuing the analysis.

**CASE STUDY**

High-throughput studies of tumor biology at multiple levels, including genome, transcriptome, and proteome, have been resulting in a greatly increased volume of cancer omics data. Given the huge amount of cancer multi-omics data, it is a major challenge to get comprehensive understanding of molecular mechanism in tumor, distinguish driver mutations from passengers, and reveal functional relationships between them. One powerful approach to the challenge is to analyze data on the context of biological network. For example, integration of cancer omics profiles with biological interaction networks has been proposed as an approach to identify drivers, relying on the assumption that they will cluster on the network [45].

In this section, we demonstrate that how MONGKIE can facilitates the study of structural pattern of GBM altered genes on the PPI network to identify functional driver mutations and core gene modules perturbed by them.

**Cancer Omics Data**

For a case study, somatic mutations and DNA copy number alterations level 3 data for all TCGA GBM cases were obtained from the UCSC Cancer
**Browser**. An alteration frequency score for each gene was calculated based on the 273 GBM cases with both sequence mutation and copy number information. Each gene was considered altered if modified by a validated non-synonymous somatic nucleotide substitution, a homozygous deletion, or a multi-copy amplification; all other copy number events were ignored, as was originally done in the original TCGA paper [8]. These somatic SNVs, indels, and called CNAs are combined to produce the patient-mutation matrix M, where M(i;j) indicates whether gene i is altered or not in patient j.

For RNA-seq expression data for TCGA GBM cases, we downloaded level 3 data sets, processed to calculate gene-level expression abundance using the MapSplice [46] genome alignment and RSEM [47] quantification pipeline, for 167 tumor and 5 solid tissue normal samples using the Broad GDAC firehose_get [48]. After normalizing RSEM estimated counts using upper quartile normalization, for each tumor sample, calculate log2FC of all genes against their mean expression of 5 normal samples, and then produced the patient-expression matrix G, where G(i;j) represents the expression change of tumor vs. normal conditions for gene i in patient j.

**Methods**

**Extraction of GBM-altered sub-network and network clustering**
Staring with the TCGA GBM data set, we selected recurrently altered genes with somatic mutations in 6 or more patients, or CNAs in 9 or more patients from the alterations matrix (See above section). A total of 380 genes passed the frequency threshold. For each pair of those genes, we found all shortest paths in the STRING database (Confidence score > 900) with distance threshold 2, resulting in 175 altered genes and 815 linkers. To retain significant linkers only, we applied the hyper-geometric distribution test for local enrichment against the global degree of each linker within the background network (see next section for details). After Benjamini &
Hochberg (aka FDR) multiple testing correction (p-value < 0.01), we built a GBM-specific sub-network with 119 altered genes, 72 linkers, and 861 interactions. The visualization of that network in MONGKIE is shown in Figure 11A.

**Figure 11.** (A) GBM-altered network with total 191 genes (altered genes = 119, linkers = 72). (B) Core gene modules in the GBM-altered network. In (A) and (B), altered genes represented by circles, and linkers by diamonds; alteration frequencies were mapped to node sizes. In (B), mean expression levels of each gene over cases of the *Mesenchymal* subtype were mapped to node colors; expression correlations in tumor cases were mapped to edge thicknesses. Two critical modules (see Results) are represented by different colors (one is blue, another is red; the others are gray)

To give weights to the sub-network, we calculated Pearson Correlation Coefficients of expression log2FC in the patient-expression matrix (see above section) among all pair-wise interactions between genes in the extracted network, and then assigned the PCCs to weights of edges in the network.

Next, we used a highly efficient network clustering algorithm, MCL, to cluster the weighted network into a set of gene modules. Each module consists of gene set that are both topologically close in the PPI network, and
highly correlated by expression abundance change in normal vs. tumor conditions. The visualization of the result is shown in Figure 11B.

**Statistical test for significant linker genes**

It is necessary to assess the probability that linker genes, which are not altered but extracted guilt by association, would connect to the observed number of altered genes by chance alone. The simplest and most widely used statistical test for such purpose is hyper-geometric distribution test, where *successes in sample*: number of edges connecting the linker to altered genes in the sub-network (local degree), *successes in background*: global degree of the linker in the background network, *population size*: total number of genes in the background network, *sample size*: number of altered genes in the sub-network.

**Network visualization and enrichment analysis**

Networks and clustered gene modules were visualized in a way, where gene alteration frequencies in all GBM cases were visually mapped to node sizes, mean expression log2FC of each gene over all cases was mapped to color of corresponding node, and Pearson Correlation Coefficient of expression log2FC between a pair of genes connected to each other in the network was mapped to thickness of corresponding edge, shown in Figure 12. KEGG pathway enrichment analysis was performed using DAVID Web service API [49], then we visually annotated the clustered gene modules with significantly over-represented KEGG pathways in corresponding modules.

**Results**

Among the top five largest modules that we discovered by network-based multi-omics (somatic mutations, copy number variations, and RNA expressions) analysis of GBM cases in MONGKIE, two corresponded very
closely to critical pathways prior known to GBM biology. First one corresponds to the components of the EGFR/PI3K signaling pathway including EGFR, PDGFRA, PIK3CA, and PIK3R1 (see Figure 12A), and second one to the components of DNA damage response and Cell Cycle including TP53, CDKN2A/B, CDK4, MDM2/4, and RB1 (see Figure 12B).

**Figure 12.** Weighted network clustering analysis identified two largest gene modules from the GBM-altered network. The node color shows the log2(FoldChange) between tumor vs. normal condition in all GBM patients.

In summary, we performed an integrated network analysis to identify core network modules in the TCGA study of glioblastoma mutiforme, and showed that network-based integrated approach can automatically identify and extract biologically relevant gene modules that play an important role in GBM tumorigenesis, given cancer multi-omics data.

**IMPLEMENTATION**

MONGKIE is a desktop application implemented in Java 1.6+ based on the NetBeans RCP [9], thus it is executable on all major operating systems such as Windows, Linux or Mac, and provides robust ways to extend
functionalities of the application with ease. In this section, we first describe the main software design and modular architecture focusing on its robustness and extensibility, then the multi-tiered system powered by RESTful Web Service APIs for abstracting data and separating them from the business logics and presentation layers, finally how we used graph database to build integrated built-in interaction data sources.

**Software Architecture**

MONGKIE is a java-based application built on top of the NetBeans Rich Client Platform that supports modular programming architecture, thus it is easy to implement various new plug-ins with additional functionalities. An overview of its modular architecture is given in Figure 13.

Based on its extensible architecture, MONGKIE provides not only core APIs (Application Programming Interfaces), SPIs (Service Provider Interfaces), and UI widgets for the base functionalities, such as graph visualization, network analysis, data integration, or interaction management, but also portable toolkit APIs which can be utilized out of the platform, and many out-of-the-box supports that enable to build your own plug-ins onto the platform. For example, The Graph Layout software module provides well-defined APIs, SPIs and UI components that can be utilized by plug-in developers (see Figure 14A). Therefore, if you want to add a new layout algorithm into the MONGKIE, you only need to implement the logic of the layout algorithm without having to worry about other things like UI components, program states, window management, event handling, and data persistence and so on. All of these fundamental features for developing plug-ins are provided out of the box. This approach can allow a great deal of flexibility in the building various improvements of existing modules as well as the introducing of new functionalities or tools.
MONGKIE also provides a GUI (Graphical User Interface) shown in Figure 14B, Plug-in Manager, in order to facilitate the management of different plug-ins. Hence, users can install, update, remove, activate, or deactivate
individual plug-ins through the integrated UI, this allows the customization of the application functionalities according to their needs.

We adapted the Model–View–Controller design pattern, which is a well-known architectural pattern [50], in order to guarantee the independence between data or logics with their views. Model, where data and their states - e.g. information about clusters of the graph - are stored, is not writable but only readable through the exposed APIs. Controller manages the creation or deletion of models, and exposes the APIs to be used to write and modify models - e.g. finding clusters of the graph and storing them into the model. View is the user interface to listen to model changes, and display model's data and states - e.g. display of the clusters. All components of the MONGKIE are designed in a such way, and this approach allows the separation of data and their views, thus all data views can be newly added or replaced by the plug-ins that implement better features.

Additionally MONGKIE is built on a multi-tasks model, therefore several tasks can be run in the same time in separate threads without blocking user interactions. For example, while loading and mapping expression profiles onto
the network, you can run a layout algorithm on the same network simultaneously.

**RESTful Web service API**

In addition to MVC design pattern, multi-tiered system is applied for abstracting remote data sources and separating control logics from the data and presentation layers to improve data integrity and accessibility. We put retrieval logics for the remote data at the middle-tier, which enables to control database connections and provide unified and maintainable data access. RESTful Web Service APIs, which are at the middle-tier, are implemented using the JAX-RS technology - the Java API for RESTful (Representational State Transfer) Web Services [51]. Based on these web APIs, MONGKIE provides the functionality that allow users to access our integrated data sources and services from outside of the platform in a programmatic fashion through any REST clients, including Java Applet, Python or Ruby scripts (see Figure 15).

**Figure 15.** Overview of 3-tier system implemented in hiPathDB [15]: (1) REST client written for pathway visualization (blue colors), (2) RESTful Web service at the middle tier (orange colors), (3) Relational database backend.
As an example, the implementation and usage of the hiPathDB [15] RESTful Web Service API is shown in Figure 15. hiPathDB APIs allow researchers to retrieve data from hiPathDB database by offering methods to get pathways given their names, or get member genes given a list of pathways, and some others. We also provides the GO enrichment analysis service which is publicly exposed as a RESTful Web Service API.

**Graph Database**

A graph database is a type of NoSQL database that uses graph theory to store, map and query relationships between nodes. Graph databases are well-suited for storing and analyzing the biological interaction networks involving complex relationships and dynamic schema. For the built-in interaction data sources, we imported PPI databases (MiMI [16], STRING [17]) and ReactomeFI [18] human functional interaction database into the Neo4J [52] graph database.

![Figure 16](image)

**Figure 16.** An example query to find shortest paths between two proteins using Cypher Query Language.

Figure 16 shows an example query to find all shortest paths between given two proteins from the STRING database using Cypher Graph Query Language [53], which is a way of using internally in MONGKIE to extract sub-networks from the PPI data sources given the protein names of interest.

**CONCLUSION**

To fulfill requirements that are not adequately addressed in network biology, such as the scalability problem of huge networks, and the overlay of temporal or dynamic contextual informations onto networks, we present a new network analysis and visualization platform especially for multi-omics data
analyses. Our new platform provides not only the sophisticated and
generalized data model for visualization of biological networks, but also
various network analytical processes for integration, management, and de novo
construction of interaction networks, network clustering analysis, gene
expression analysis, and over-representation analysis.

We have applied some of major functionalities of MONGKIE to the
analysis of TCGA Glioblastoma Multiforme multi-omics data set [8] and
revealed that our tool can be used to automatically identify functional driver
mutations and core gene modules sharing structural pattern with those genes
in biological interaction networks, thus capture critical pathways that play
important roles in cancer. We believe that MONGKIE would be a valuable
addition to network analysis software by supporting many unique features and
visualization options, especially for analysing multi-omics data sets in cancer
and other diseases.

Availability

MONGKIE is an open-source and platform-independent software distributed
under the GNU AGPL (Affero General Public License) V3 license [54] with
exception of some external libraries that are available under their own
licenses. Documentation including tutorials and videos, source code, and
installable packages are available at http://yijang.github.io/mongkie. In addition
to installable packages, we also provide zip-packaged versions for Windows,
Mac OS X, and Linux, that are containing an executable for starting the
application without an installation.
REFERENCES


50. Trygve Reenskaug. **MODELS - VIEWS - CONTROLLERS.** Xerox PARC technical note December 1979


52. Neo4j Graph Database. http://neo4j.com/


국문초록

NGS(WES, WGS, RNA-seq) 기술과 분석 도구의 발전으로 암 등의 질병 유전자 변이(SNV, copy number variation, gene fusion 등)들을 빠른 시간 안에 파악 가능하게 되었지만, 변이의 숫자가 많고 상당수가 false positive이거나 passenger이기 때문에 그 중에 driver 변이와 그들과 연결된 세포내 기능 모듈을 찾아내는 것이 중요 도전 과제이다. 세포내의 엔티티(유전자, TF, miRNA 등)들은 각각 독자적으로 기능하지 않고, 복잡하게 상호작용함으로써 기능적 모듈 단위로 세포내에서 특정 기능을 하게 된다. 따라서 다양한 상호작용 네트워크(패스웨이, PPI, TF-target miRNA-target 조절 네트워크 등) 기반의 multi-omics 데이터 분석이 요구된다. 본 논문에서는 이러한 분석을 위해 다양한 생물학 네트워크를 제공, 생성 및 관리할 수 있고, 다양한 그래프 분석 알고리즘을 구현하며, 세포 돌연변이, copy number alteration, 유전자 발현정보 등 여러 omics 데이터를 통합해서 네트워크 기반의 분석을 할 수 있는 파이프라인 제공 및 분석 결과 가시화를 목적으로 개발한 프로그램 MONGKIE에 대해 기술한다. 또한 MONGKIE가 제공하는 네트워크 기반의 분석 방법들이 실제로 암 유전자 분석에 있어 얼마나 효과적인지 보이기 위해, TCGA 뇌종양(GBM) 유전자(somatic mutations, copy number alterations) 및 전사체 데이터로부터 driver 돌연변이와 그로 인해 비정상화되는 세포내 신호전달 패스웨이를 찾는 분석을 수행하였다. 그 결과 뇌종양을 유발하고 암세포 활성화 및 전이에 중요 역할을 한다고 이미 알려진 주요 돌연변이 유전자들 및 핵심 패스웨이들을 찾아낼 수 있음을 확인하였다. MONGKIE는 자바 프로그래밍 언어로 작성되어 운영체제에 독립적으로 실행 가능하며, 플러그인 구조로 개발되어 추가 기능 확장에 용이하다. 또한, 오픈소스 프로그램으로 GNU AGPL v3 라이선스 하에 http://yjjang.github.io/mongkie 사이트에서 배포된다.
주요어 : 네트워크 가시화, 네트워크 모델링, 그래프 클러스터링, Omics 데이터 분석, Over-representation 분석
학번 : 2014-20341