An interactive graphical method for community detection in network data

Andee Kaplan, Heike Hofmann, and Daniel Nordman

September 19, 2014

Abstract

The analysis of graphs, in particular the detection of community structures within network data, is of increasing interest across a broad range of disciplines. Such communities represent clusters of nodes in a network which exhibit strong intra-connections or relationships among nodes in the cluster. Current methodology for community detection often involves an algorithmic approach, and commonly partitions a graph into node clusters in an iterative manner before some stopping criterion. Other statistical approaches for community detection often require model choices and prior selection in Bayesian analyses, which are difficult without some amount of data inspection and pre-processing. Because communities are often fuzzily-defined human concepts, an alternative approach is to leverage human visualization to identify communities. The work presents a new data visualization method (a web application), called *gravicom*, for facilitating the detection of community structures through direct user interaction. In the process of detecting communities, the gravicom application may serve as a standalone tool or as an initialization step for use with another community detection algorithm. In this paper we discuss the design of gravicom and demonstrate its use for community detection using several network data sets. An appendix describes details in the technical formulation of this web application built on the R package Shiny and the JavaScript library D3.

1 Introduction

Many different relationships are often easily conceptualized as a graph or network, where a graph is defined as a collection of nodes (entities or vertices) and edges (relationships or links) [34]. Examples of such relationships include social networks (sociology), the world wide web (computer science), and protein networks (biology). Community detection in particular is often one important goal in the analysis of real world networks, with the aim of identifying members or entities in the network that share a similar characteristic of interest. Communities, also known as clusters or modules, are roughly defined as a group of nodes in a graph that share properties [14]. Commonly, a community structure can be characterized in a graph as a collection of nodes which share many edges internally, but exhibit relatively fewer edges to other nodes outside the collection.

Community detection has largely been studied in the computer science and physics literature ([17, 20, 25, 27, 21]), with increasing interest in statistical investigations as well ([2, 5]). Current methodology for community detection often involves an algorithmic approach for node clustering and commonly partitions a graph into node clusters in an iterative manner before some stopping criterion are met. These algorithmic approaches typically involve first defining an objective function that defines a community in terms of internal connectivity versus external and then optimizing this objective function. One prevalent example of an objective function is Girvan & Newman's [35] modularity measure. Modularity is defined as

$$Q = \sum_r (e_{rr} - a_r^2)$$

where e_{rr} is the fraction of links that connect two nodes inside the community r, and a_r is the fraction of links that have one or both vertices inside the community r. The search for an optimal modularity value is an NP-hard problem because the number of possible partitions of the network requires 2^n complexity (where n is the number of nodes in a network). In fact, the optimization of an objective function is typically an NP-hard problem, which is why heuristic or approximate approaches to find nodes that optimize the objective function are necessary ([26], [11]). Heuristic-based clustering is indeed useful because this offers an automated way to perform community detection. However, determining the best algorithm for community detection is intractable because, as noted by Fortunato [14], "The main elements of the problem themselves [graph clustering], i.e. the concepts of community and partition, are not rigorously defined, and require some degree of arbitrariness and/or common sense". Parametric statistical approaches to community detection have also received increasingly greater attention with many of these methods relying heavily on stochastic block models ([36, 3, 21]), often combined with some form of Bayesian analysis ([22, 1, 18]). However, these statistical analyses are difficult to automate, and presuppose some knowledge about the graph data structure for formulating models and prior distributions. Similar to algorithmic approaches, implementation of these statistical methods can become complicated when the nature of communities is unclear.

Since communities are often fuzzily-defined human concepts, an alternative approach is to leverage the human visual system, which is capable of incorporating nuance, to identify communities. However, in the mathematical mechanics of current algorithmic and statistical approaches to community detection, user visual inspection of graph data does not inherently guide how community structures are obtained. Tools exist for displaying or visualizing graph data but there is a disconnect between the capability to display such data and the ability of a user to visually guide steps of community detection. For example, tools such as igraph [9], Gephi [4], and Graphviz [16] for graph visualization offer only algorithmic approaches for community detection, separate from any input or interface on the part of a user, and NodeXL [19] offers similar algorithms for a related concept called cliques. Recently, there have been advances in the production of open source packages to simplify creation and visualization of graphs for the web. Tools such as JSNetworkX [23] and the R [37] package d3Network [15] are both built upon the D3 [7] framework and aim to help users create graph visualizations from their own data. However, these tools currently do not have built in functionality to deal with community detection as a process of statistical analysis through data visualization.

Our goal in this paper is to introduce a novel visualization-based community detection tool, called gravicom, which allows users to visually direct and interact with the steps of community detection. Additionally, as described in the following, gravicom is equipped with several functionalities that permit users to visually cluster nodes and assess the resulting clusters through visual features and statistical quantitative summaries in the process of finding community structures. Unlike the tools mentioned above for graph visualization, gravicom allows human interaction as the vehicle for community detection. We incorporate three key graphical devices to formulate gravicom. The first is visualization of a graph using a node-link diagram; the second is using a force-directed graph layout in our visualization; and the third is the ability to simplify a graph by grouping subsets of nodes into a representation based on the user's definition of a community. gravicom can be used as a standalone exploratory tool for graph data or to generate an initial state to be passed to a community detection algorithm or model-based statistical procedure in order to reduce the complexity of optimization or implementation.

To visualize a graph, a node-link diagram is used in which a node is represented as a single point and a connecting edge is represented as a line connecting two points. Creation of this diagram involves assigning each node a Cartesian coordinate, which is not an inherent property of a graph. This assignment is called a graph layout. McGrath, Blythe, and Krackhardt [33] found that the layout of a graph significantly affected the number of communities that users detected within a graph. Thus, when humans are the mechanism used to detect communities, special attention needs to be paid to the layout being used. The same study also found that location of a node spatially relative to other nodes in a cluster has a significant effect on user ability to detect the community. The authors suggest a simple principle that will lead to clear depiction of a network: "adjacent nodes must be placed near to each other if possible" [33]. One layout that adheres to this principle is a force-directed layout. This layout algorithm implements edges as fixed-distance geometric constraints, meaning that groups of nodes sharing multiple edges are pulled in closer proximity. Dwyer, et. al. [13] found the force-directed layout to be the best algorithmic layout as a platform to detect cliques, which are one type of community. Further, they concluded that user-generated layouts similar to a force-based layout, but with symmetric layout within the cliques, allowed an even more accurate cluster recognition by users. Adaptation of the force-directed layout to incorporate a symmetric within-cluster layout "may be a difficult task for automatic algorithms, since maximal clique detection is NP-hard" [13]. As cliques are similar to communities, in fact a clique is a community with perfect membership, we infer that adaptation of the force-directed layout to have symmetry within a community would be equivalently difficult. Thus, a force-directed layout is a starting point for the user to be able to "tweak" a layout into a more easily interpretable structure. We incorporated this crucial aspect into the mechanics of gravicom.

In complex or large graphs it can be difficult to glean meaning from a graphical representation, even while using a force-directed layout. Dunne and Shneiderman [12] introduce the idea of motifs, or repeated patterns in a graph, to simplify a network. One type of motif is a clique. By replacing the cliques with representations, like a large circle, and removing the corresponding extra nodes and edges, the visualization will be more effective at revealing relationships. With fewer nodes and edges to display, visual complexity of the graph visualization is greatly reduced, allowing the user to analyze the network structure more accurately.

Having introduced the three key visual devices informing gravicom, we next describe the user interface and functionality in Section 2. In Section 3, we then illustrate the use of gravicom for community detection with two network data sets, one, a well known data set of college football and the other, a network of political book co-purchasing. Section 4 then provides concluding remarks as well as a look forward at further network inference problems that can be addressed with extensions and modifications of gravicom. gravicom is available at `https://andeek.shinyapps.io/gravicom/`. A technical appendix details three important pieces of technology that we used to create gravicom and describes the technical aspects of how these components are implemented together.

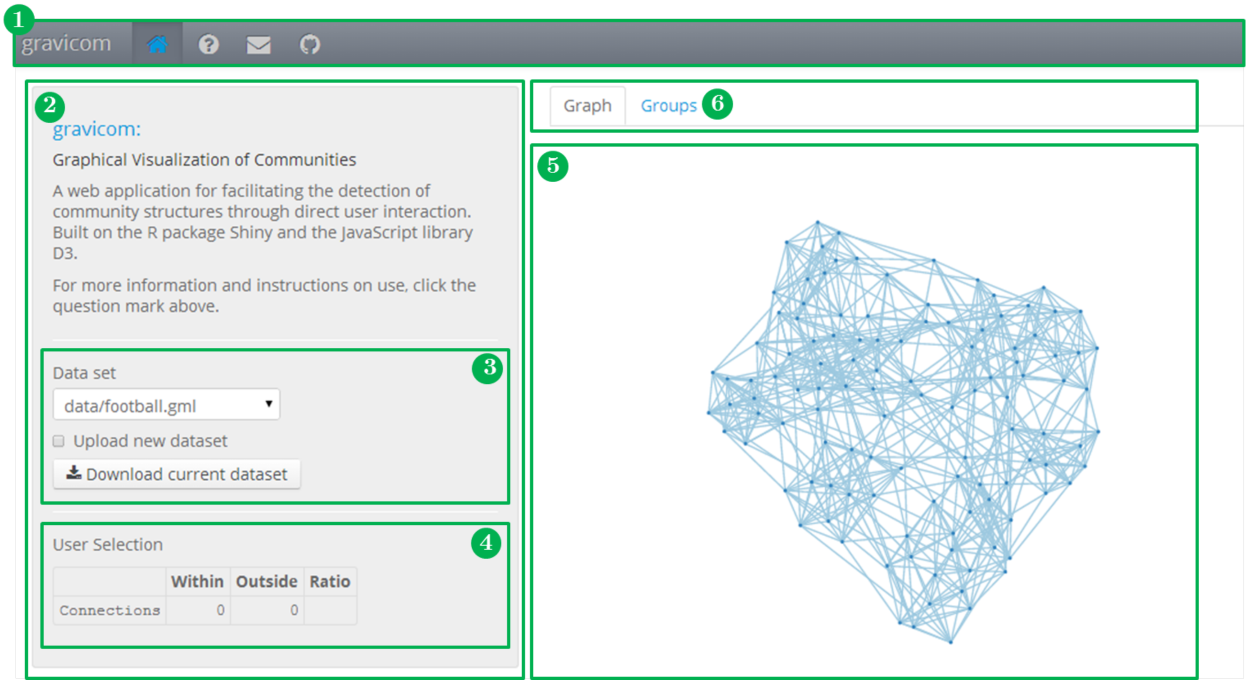
2 User Interface

Before discussing gravicom's performance and use on example data sets, we give a brief overview of the components and functionality that make up the tool. The gravicom interface is comprised of six main parts,

1. Navbar
2. Control panel
3. Data management
4. Connection table

5. Graph display

6. Tabset.



gravicom:

Graphical Visualization of Communities

A web application for facilitating the detection of community structures through direct user interaction. Built on the R package Shiny and the JavaScript library D3.

For more information and instructions on use, click the question mark above.

Data set

data/football.gml

□ Upload new dataset

Download current dataset

User Selection

	Within	Outside	Ratio
Connections	0	0	

Figure 1: The components that make up gravicom, (1) Navbar, (2) Control panel, (3) Data management, (4) Connection table, (5) Graph display, and (6) Tabset.

Each part provides a means for the user to interact with gravicom, either through controls that allow user input to gravicom or through direct interaction with diagnostics and visualization of a graph. Their placement on the gravicom interface can be seen in Figure 1.

Navbar The top navigation bar includes informational buttons. The first is a link to gravicom. The second is a link to this documentation page. The third is a link to the GitHub repository where the code for gravicom is housed. The final button is a link to an author's website, which contains contact information to address questions or comments.

Control Panel The control panel serves as the starting point for a user's session in gravicom. It contains instructions for the user, as well as the means for a user to select a data set and numerical summaries of the graph (such as a diagnostic connection table explained below).

Data Management The data management component is made up of two main parts, data selection and data download. The data selection can be accomplished in two ways, the first being a drop down to select pre-loaded data sets to display. Currently there are two well-known network data sets provided in gravicom, a college football data set [17] and a karate club data set [39]. From the dropdown the user can change the data set to display in the graph. The second approach to data selection gives the user the ability to upload his own data set. Upon clicking the "Upload new data set" checkbox, a file selection control appears which gives the user the ability to upload his own graph data to explore with gravicom. This is shown in Figure 2.

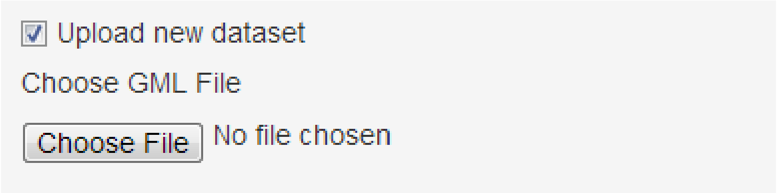

☑ Upload new dataset

Figure 2: The data selection area upon clicking the “Upload new data set” checkbox.

The work performed by a user in visualizing graphs and community structure can also be downloaded as a data set from gravicom with current communities stored. This feature can be used as a save point in processing a graph or as a means to export changes made in gravicom to another tool.

Connection Table The connection table is a quantitative diagnostic tool for the user in assessing the strength of a community structure in a graph. The idea behind the connection table is that a community of nodes will have proportionally more edge connections within the node cluster compared to edge connections to nodes outside the community. The table displays the number of edge connections within a user’s selection of nodes in the graph and the number of connections from nodes in a user’s selection to nodes not in the selection. The comparison of these two numbers can give the user a rough idea of if the plausibility or extend to which the node selection constitutes a community. To aid in the comparison, there is also a column that displays the ratio of number of connections within a node selection to the number of connections outside the selection.

Graph Display The graph display shows an interactive graphical representation of the selected (or uploaded) graph data. Upon load, the graph displays all nodes and edges in the data set using a force-directed layout algorithm. The user has several ways to interact with the graph: drag, select, and group. A user can drag a node at any time. Upon dragging, the force-directed layout is rerun, giving an altered view of the graph. Figure 3 shows a graph in the process of being dragged.

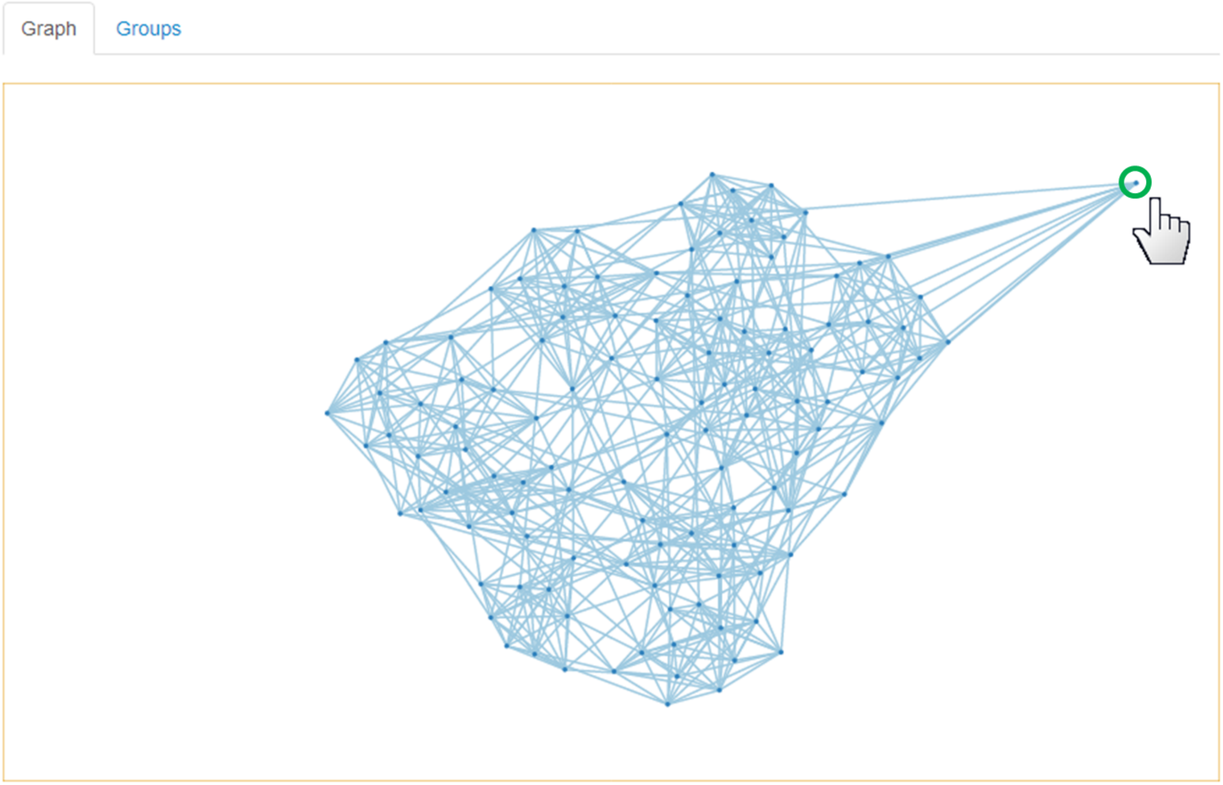
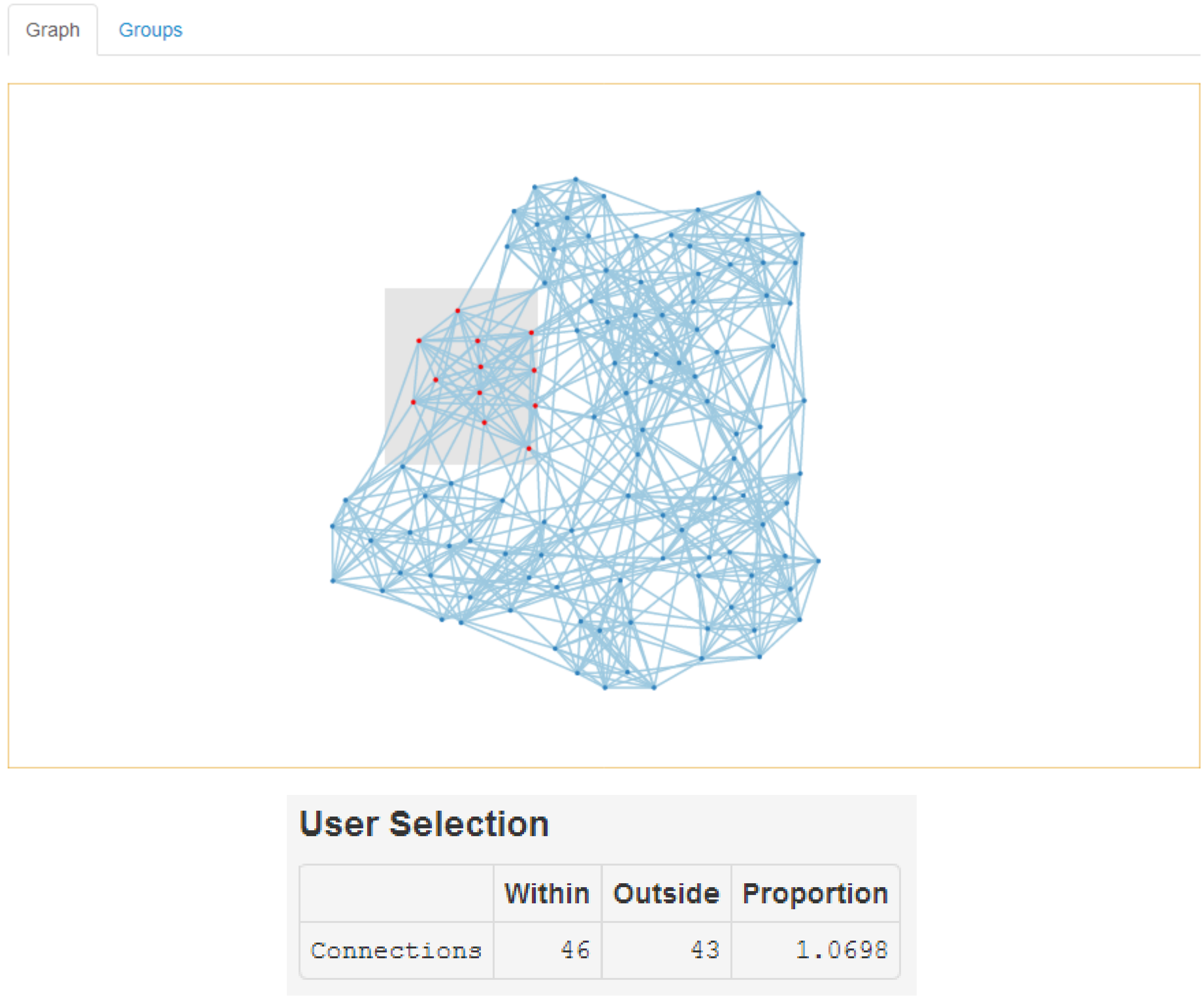


Figure 3: A graph in the process of being dragged. The node being dragged is marked by a green circle.

Selection and grouping of nodes are actions intended to work together. In order to group nodes, a user first determines a node cluster or potential community based on a visual appraisal of the graph. To select nodes the user clicks and drags a selection box around nodes. See Figure 4 for the results of selection in the interface. The shift key can also be used for multiple selections. Upon selection, the connection table is updated and the user can evaluate the selection as a community and alter the selection if need be (the shift key selection is useful in this step).



User Selection

	Within	Outside	Proportion
Connections	46	43	1.0698

Figure 4: A graph in the process of nodes being selected. Upon selection of nodes, the connection table updates to display within and outside edges.

Selected nodes can be grouped together into one consolidated “super-node” or grouped-node. Once grouped, the size and edges of the super-node represent the number of nodes in the potential community and the number of edges to the grouped community nodes, respectively. The force-directed layout is again run, showing the new graph with previous nodes grouped. This is illustrated in Figure 5. This process of node grouping can be repeated until all nodes have been grouped or until the user is satisfied that all communities have been selected. Additionally, grouped nodes can be ungrouped by clicking on a grouped node.

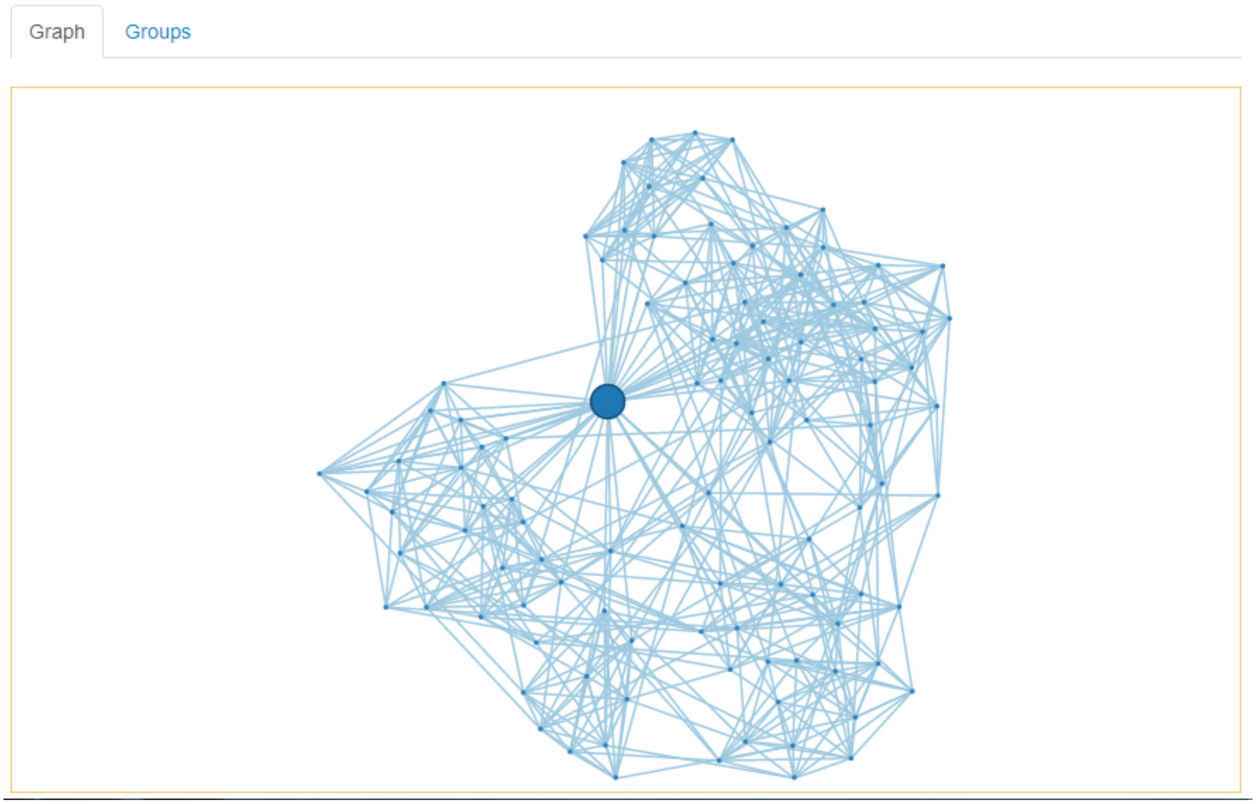
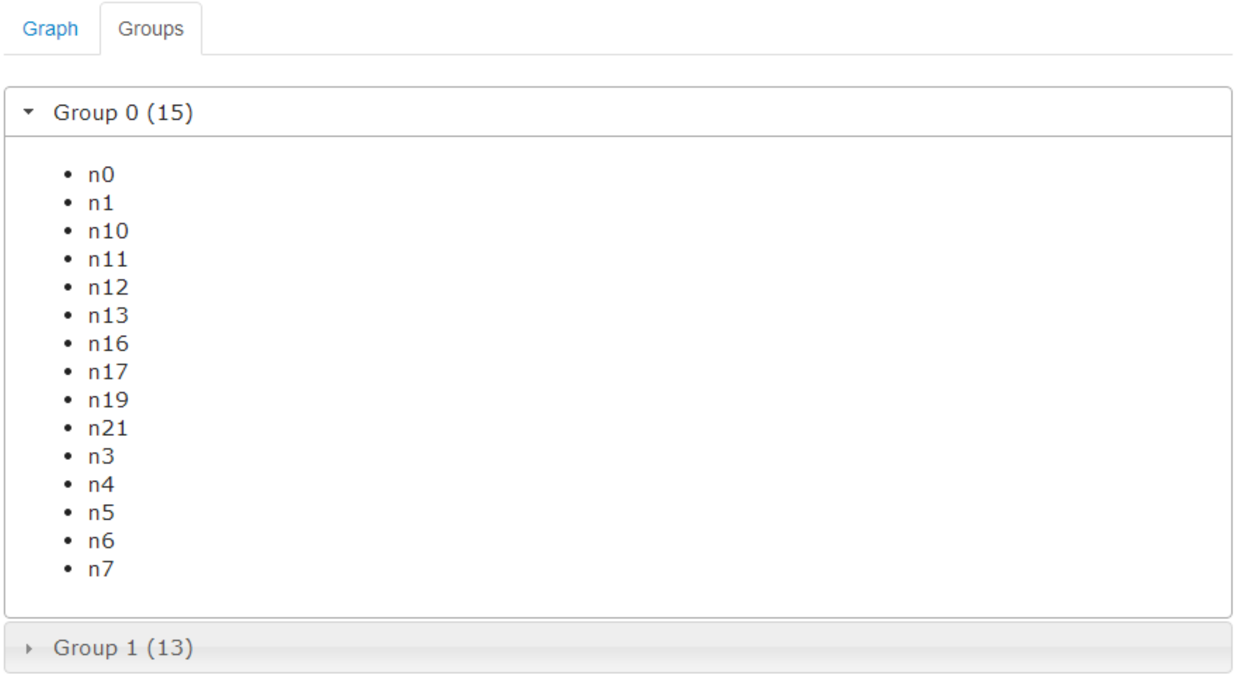


Figure 5: A graph after nodes have been grouped and the force-directed algorithm has been re-run.

Tabset The tabset allows the user to switch between two tabs on the screen. The first (and default tab) shows the graph display. The second tab shows the groups that a user has created in the graph. Within this tab, each node group or community summarizes the number of nodes in the group and also provides the ability to drop down and view the node IDs for that group. If the data are equipped with node labels, these will be displayed. If there are no node labels provided, node IDs will be shown as node numbers. For an example, see Figure 6.



Group 0 (15)

- n0
- n1
- n10
- n11
- n12
- n13
- n16
- n17
- n19
- n21
- n3
- n4
- n5
- n6
- n7

Group 1 (13)

Figure 6: The groups tabset displaying which nodes have been grouped in Group 0, for example. The groups tabset also shows that Group 0 has 15 nodes, while Group 1 has 13 nodes.

3 Examples

To demonstrate the use of gravicom, we present two real-world network data sets and explore their community structure.

3.1 College Football

The first data set is a representation of U.S. College Football Division 1 games from the 2000 season [17]. This is a default data set available in gravicom. In this network, nodes represent teams and an edge represents a regular-season game played between the two connected teams. The distances between the nodes are based on the number of games played between the teams. The network as it appears upon load in gravicom is presented in Figure 7.

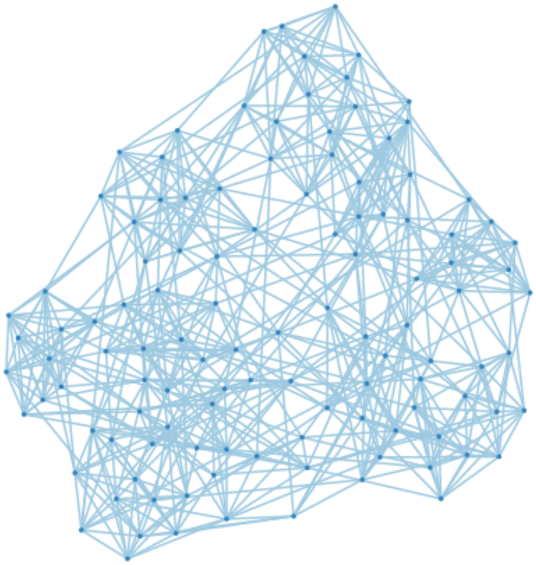
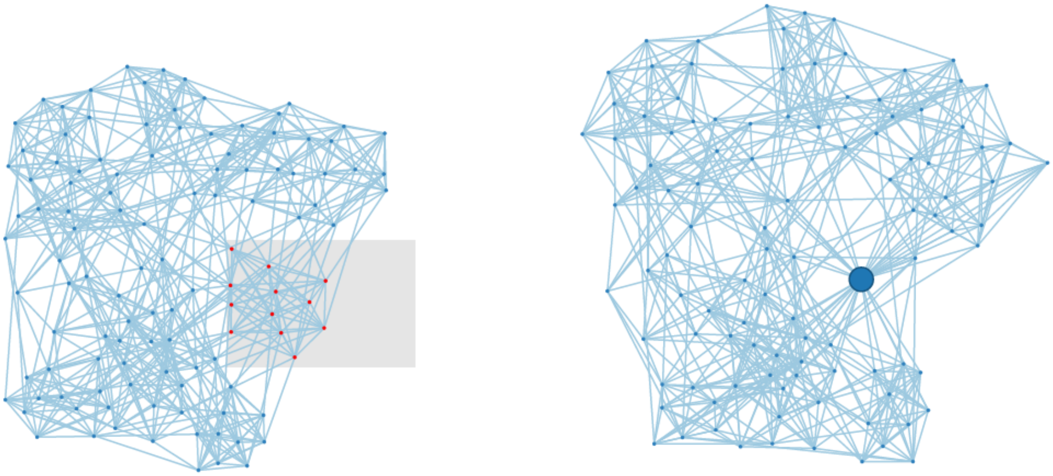
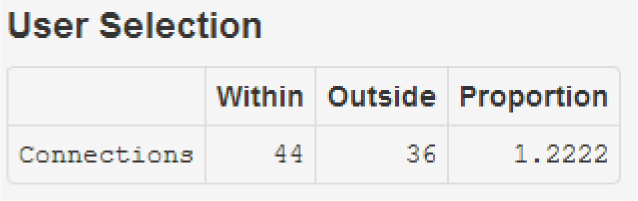


Figure 7: College Football network represented in gravicom. Communities are visually evident and will become more identifiable as other are grouped.

Colleges within the same football conference will play members of their conference more frequently than teams outside of their conference, making this an interesting data set for attempting to visually detect community structures. This is also an ideal illustrative example due to the relatively small number of nodes and edges present, making a graphical representation particularly feasible. One challenge with this data set, however is the existence of independent teams, like Notre Dame, which do not belong to any conference and so may complicate efforts of community detection. Another complication is that small conference schools typically play large conference schools at the beginning of a season in order to help fund their athletic programs, which can potentially cause more edges than perhaps expected between distinct communities, particularly between small conferences and large conferences.

Upon viewing the network in gravicom, some likely communities visually emerge, and by selecting nodes to examine within and outside edges, we can classify colleges into conferences, as seen in Figure 8.





User Selection

	Within	Outside	Proportion
Connections	44	36	1.2222

Figure 8: Selecting a potential community in gravicom and assessing the number of within verses outside selection edges. After the first community is detected, more communities become apparent in the network.

Once the user is satisfied that he has selected a viable community, he clicks to group those nodes and the graph will update to reflect this. In particular, the grouped nodes for the suspected community will be collapsed into one super-node in the updated graph. See Section 2 for details. The resulting graph update allows new community structures to potentially become more easily apparent as seen in Figure 8.

Additionally, the user can check which nodes were grouped in each community as described in 2. In this example, the first community detected contains the following schools, which corresponds to the Big 10: *a*) Illinois; *b*) Indiana; *c*) Iowa; *d*) Michigan; *e*) Michigan State; *f*) Minnesota; *g*) Northwestern; *h*) Ohio State; *i*) Penn State; *j*) Purdue; and *k*) Wisconsin.

After the first community has been selected, another potential community at the top of the graph has been revealed. The user can once again select and group this node cluster into a community and subsequently examine the nodes in the resulting group as seen in Figure 9. The second conference grouped corresponds exactly to the SEC Conference, another large college football conference. Here, the first two communities to become evident correspond to large Division 1 conferences that play the majority of their games within conference, matching our earlier assertion that small conferences should, as expected, be more difficult to detect.

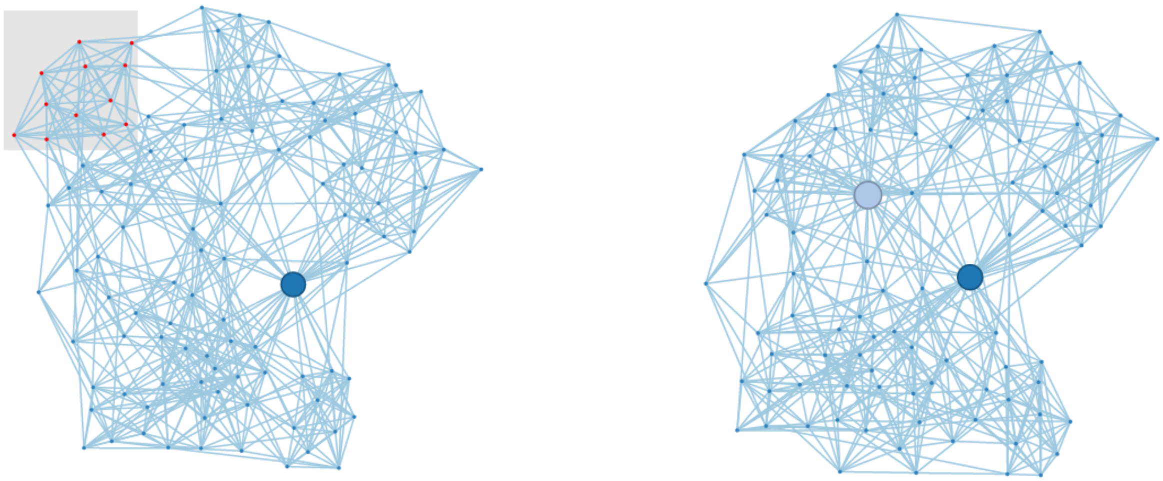


Figure 9: A second community is grouped which corresponds to the SEC Conference.

This process of grouping nodes into suspected communities can be repeated until the user is satisfied with the communities selected. The entirety of this process is seen in Figure 10 and the resulting communities in Table 1. It should be noted that this is a fundamentally a subjective process and that another user with the same data may find slightly different communities. However, by leveraging the human visual system to find structure in a data set, gravicom has the ability to provide a starting point for an objective algorithm to then adopt the detection of communities.

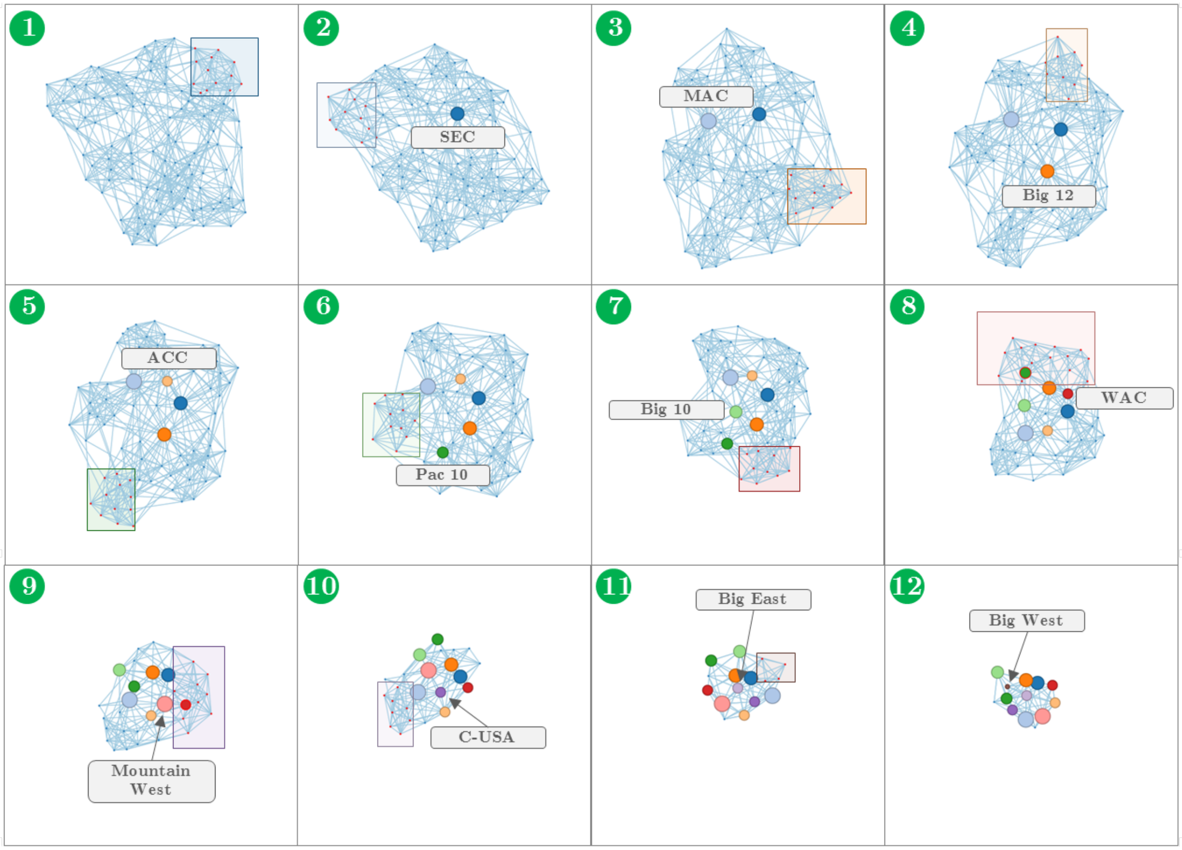


Figure 10: The full process of selecting and grouping communities in gravicom using the football data set.

Conference	Teams Identified	Ratio	Accuracy
SEC	Alabama, Arkansas, Auburn, Florida, Georgia, Kentucky, Louisiana State, Mississippi, Mississippi State, South Carolina, Tennessee, Vanderbilt	1.50	100%
MAC	Akron, Ball State, Bowling Green State, Buffalo, Central Michigan, Eastern Michigan, *Central Florida*, Kent, Marshall, Miami Ohio, Northern Illinois, Ohio, Toledo, Western Michigan	1.46	92.9%
Big 12	Baylor, Colorado, Iowa State, Kansas, Kansas State, Missouri, Nebraska, Oklahoma, Oklahoma State, Texas, Texas A& M, Texas Tech	1.44	100%
ACC	Clemson, Duke, Florida State, Georgia Tech, Maryland, North Carolina, North Carolina State, Virginia, Wake Forest	1.44	100%
Pac-10	Arizona, Arizona State, California, Oregon, Oregon State, Southern California, Stanford, UC LA, Washington, Washington State	1.33	100%
Big 10	Illinois, Indiana, Iowa, Michigan, Michigan State, Minnesota, Northwestern, Ohio State, Penn State, Purdue, Wisconsin	1.22	100%
WAC	Fresno State, Hawaii, *Texas Christian*, Nevada, Rice, San Jose State, Southern Methodist, Texas El Paso, Tulsa	1.20	88.9%
Mountain West	Air Force, Brigham Young, Colorado State, *Arkansas State*, *Boise State*, *Idaho*, *New Mexico State*, *North Texas*, *Utah State*, Nevada Las Vegas, New Mexico, San Diego State, Utah, Wyoming	0.96	57.1%
C-USA	Alabama Birmingham, Army, Cincinnati, East Carolina, Houston, Louisville, Memphis, Southern Mississippi, Tulane	0.91	100%
Big East	Boston College, *Connecticut*, Miami Florida, Pittsburgh, Rutgers, Syracuse, Temple, Virginia Tech, West Virginia	0.83	88.9%
Big West	*Louisiana Tech*, Louisiana Lafayette, Louisiana Monroe, Middle Tennessee State	0.26	75%
Independent	Navy, Notre Dame	0.00	100%

Table 1: Resulting communities detected using gravicom and the corresponding conference. Teams that have been incorrectly classified are italicized.

Using the visual approach detailed above, we were able to detect 11 community structures in the graph. There were 11 conferences and 5 independent schools in the data set. Through manual specification of conferences, we were able to correctly classify 91.3% of the football teams into their conferences.

3.2 Political Books Sold

For illustration, we next consider a second example data set consisting of a network of political books purchased close to the 2004 United States presidential election and sold on Amazon.com [6]. Each node represents a book and each edge represents frequent co-purchasing of two books by the same buyers. The books are classified as being conservative, liberal, or neutral by the author of the data set [24] and we can see a clear partition in Figure 11 between two main groups (i.e. roughly conservative and liberal) with a smaller group between, when looking at the network in gravicom. This data set is interesting because it highlights a great divide in the political books sold via Amazon that can be ascribed to the two party political system.

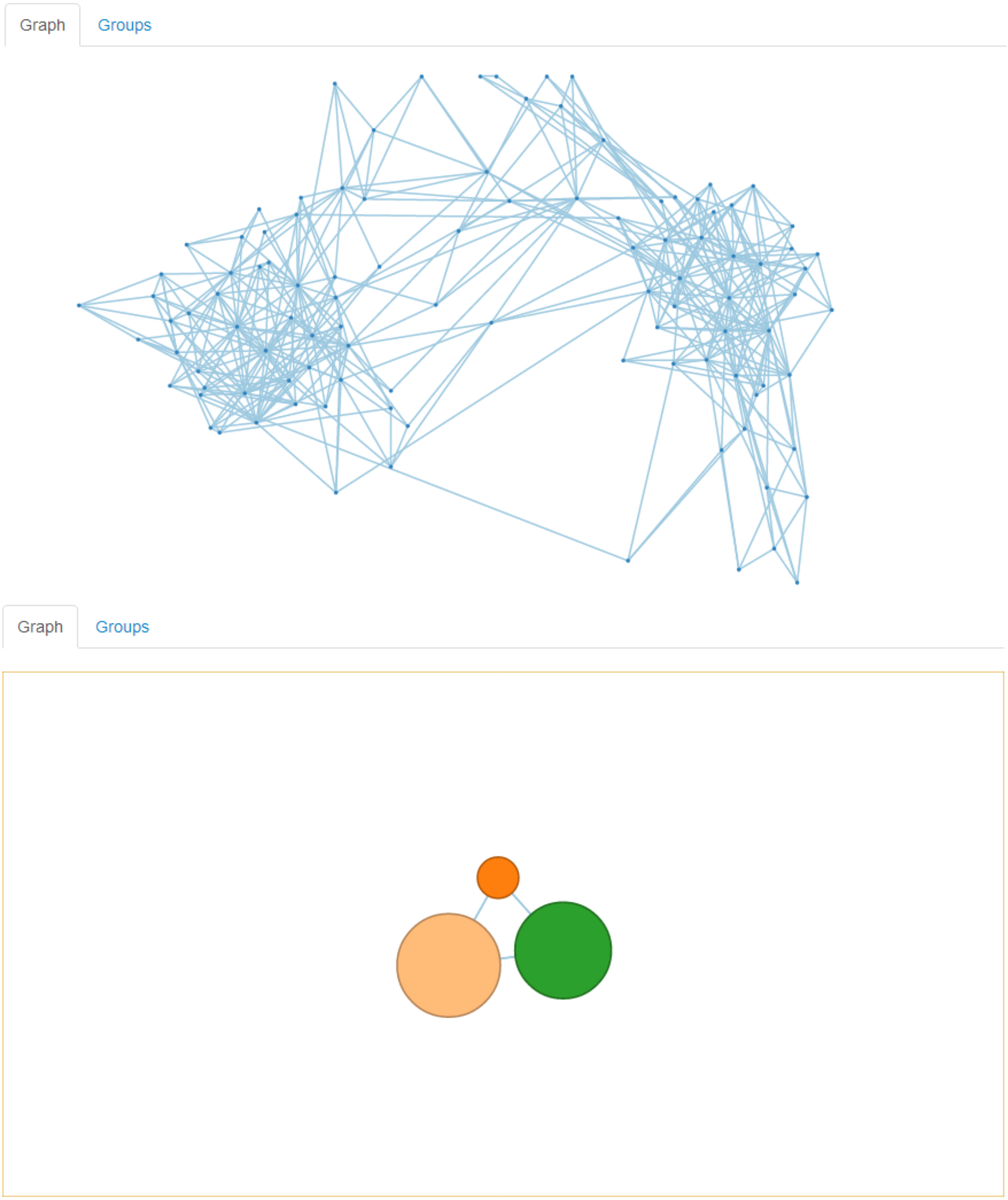


Figure 11: Political books data set as seen in gravicom prior and post community detection.

Classification	Books Identified	Ratio	Accuracy
Conservative	A National Party No More, Arrogance, Betrayal, Bias, Breakdown, Bush Country, Deliver Us from Evil, Dereliction of Duty, Endgame, Fighting Back, Give Me a Break, Hating America, Hillary's Scheme, Hollywood Interrupted, *Meant To Be*, *Power Plays*, *The Perfect Wife*, Legacy, Let Freedom Ring, Losing Bin Laden, Off with Their Heads, Persecution, Rumsfeld's War, Shut Up and Sing, Slander, Spin Sisters, Tales from the Left Coast, Ten Minutes from Normal, The Bushes, The Death of Right and Wrong, The Enemy Within, The Faith of George W Bush, The French Betrayal of America, The Official Handbook Vast Right Wing Conspiracy, The O'Reilly Factor, The Real America, The Right Man, The Savage Nation, The Third Terrorist, Things Worth Fighting For, Those Who Trespass, Useful Idiots, We Will Prevail, Who's Looking Out for You?, Why Courage Matters	8.04	93.3%
Liberal	Against All Enemies, American Dynasty, Big Lies, Buck Up Suck Up, Bushwhacked, Bushwomen, Disarming Iraq, Downsize This!, Dude, Where's My Country?, Fanatics and Fools, Freethinkers, Had Enough?, Hegemony or Survival, House of Bush, House of Saud, *Bush at War*, *Plan of Attack*, It's Still the Economy, Stupid!, Lies and the Lying Liars Who Tell Them, Living History, MoveOn's 50 Ways to Love Your Country, Perfectly Legal, Rogue Nation, Rush Limbaugh Is a Big Fat Idiot, Shrub, Stupid White Men, The Best Democracy Money Can Buy, The Bubble of American Supremacy, The Buying of the President 2004, The Clinton Wars, The Culture of Fear, The Exception to the Rulers, The Great Unraveling, The Lies of George W. Bush, The New Pearl Harbor, The Politics of Truth, The Price of Loyalty, The Sorrows of Empire, Thieves in High Places, Weapons of Mass Deception, We're Right They're Wrong, What Liberal Media?, Worse Than Watergate	7.42	95.2%
Neutral	1000 Years for Revenge, All the Shah's Men, Colossus, Empire, Ghost Wars, *Allies*, *America Unbound*, *Bush vs. the Beltway*, *Charlie Wilson's War*, *Dangerous Dimplomacy*, *Rise of the Vulcans*, *Soft Power*, *The Choice*, *The Man Who Warned America*, Sleeping With the Devil, Surprise, Security, the American Experience, The Future of Freedom, Why America Slept	1.06	50%

Table 2: Resulting communities detected using gravicom and the corresponding type of book. Books that have been incorrectly classified are italicized.

We detected visually 3 types of books in the data set using the process of selection and grouping detailed in section 3.1. Through subsequent manual verification of the classification of books, we found that we were able to correctly classify 86.67% of the books into the categories created by the author of the data set. See Table 2 for the final groups of books found using gravicom.

4 Further Work

In this section, we will briefly present two ideas for extending our work on gravicom. The first, integrated algorithmic community detection, improves community detection, and the second, dynamic temporal graph visualization, will allow for the analysis of changing graphs over time in gravicom.

4.1 Integrated Algorithmic Community Detection

The idea behind integrated algorithmic community detection is to combine the benefits of human detection of communities with algorithmic detection. Human detection using gravicom as described in this paper can serve as an initialization phase of community detection. From there, an iterative algorithm would be available to run in the background. The algorithm might ideally run step-by-step and send results back to the user, which serves two purposes. First, the user can visually track how the algorithm is detecting communities by showing what groups are created at each iteration of the clustering algorithm. This transition could help the user to track progression of the algorithm. Being able to visually track changes might better assist the user to understand what is happening within the algorithm, rather than a black box method.

Second, in between iterations of the algorithm, the user could potentially choose to accept or reject changes made by the algorithm. The user could make adjustments and send the updated results back to the algorithm. This checkpoint would in effect create flexible stopping criteria. For example, suppose the algorithm, running step-by-step, still has remaining steps before meeting the stopping criteria. A user could decide at each point whether to continue through the remaining steps or stop at a specific point. Similarly, suppose the algorithm has met the stopping criteria but the user identifies the need to perform additional iterations. The integration of human and algorithmic detection would allow the user to stop or continue as necessary.

Because we built gravicom on the shiny framework, it is not a major step to add this feature. We already have the ability to send a current status of a graph clustering to R running on the server, perform manipulations, and send the processed graph back. The next step would be to implement the algorithm in R, using, for example, the igraph package.

4.2 Dynamic Temporal Graph Visualization

The idea for dynamic temporal graph visualization is that a user would be able to view a dynamic graph across time, specifically how the edges change between nodes. With this, a user could find time-dependent communities.

An example that illustrates this point is the series of novels, A Song of Ice and Fire [31, 28, 32, 30, 29]. This data was compiled by Jerome Cukier using two fan-maintained resources on the series [10]. This data set contains events that occur for each character for each chapter in the book as well as information on the chapters, books, and characters themselves. From this relational structure we are able to create a graph where each node is a character and each edge is the occurrence of an "important event" happening between the two characters in the same chapter. We categorize an "important event" as any event that is not simply a mention of either character name. Looking at important events in the same chapter, one can examine how relationships between characters change from one chapter to the next. We performed this analysis for each book separately and we present the results in Figure 12. The first striking feature from this visualization is the stark difference in how characters interact from book two to book three. In book two, there are many medium sized clusters in evidence, indicating many disparate events affecting a mid-size amount of characters. To the contrary, book 3 shows one very large event occurring and then many other events between a small amount of characters. In fact, the third book in the series, A Storm of Swords, houses a brutal massacre in which over 10 major characters are killed.

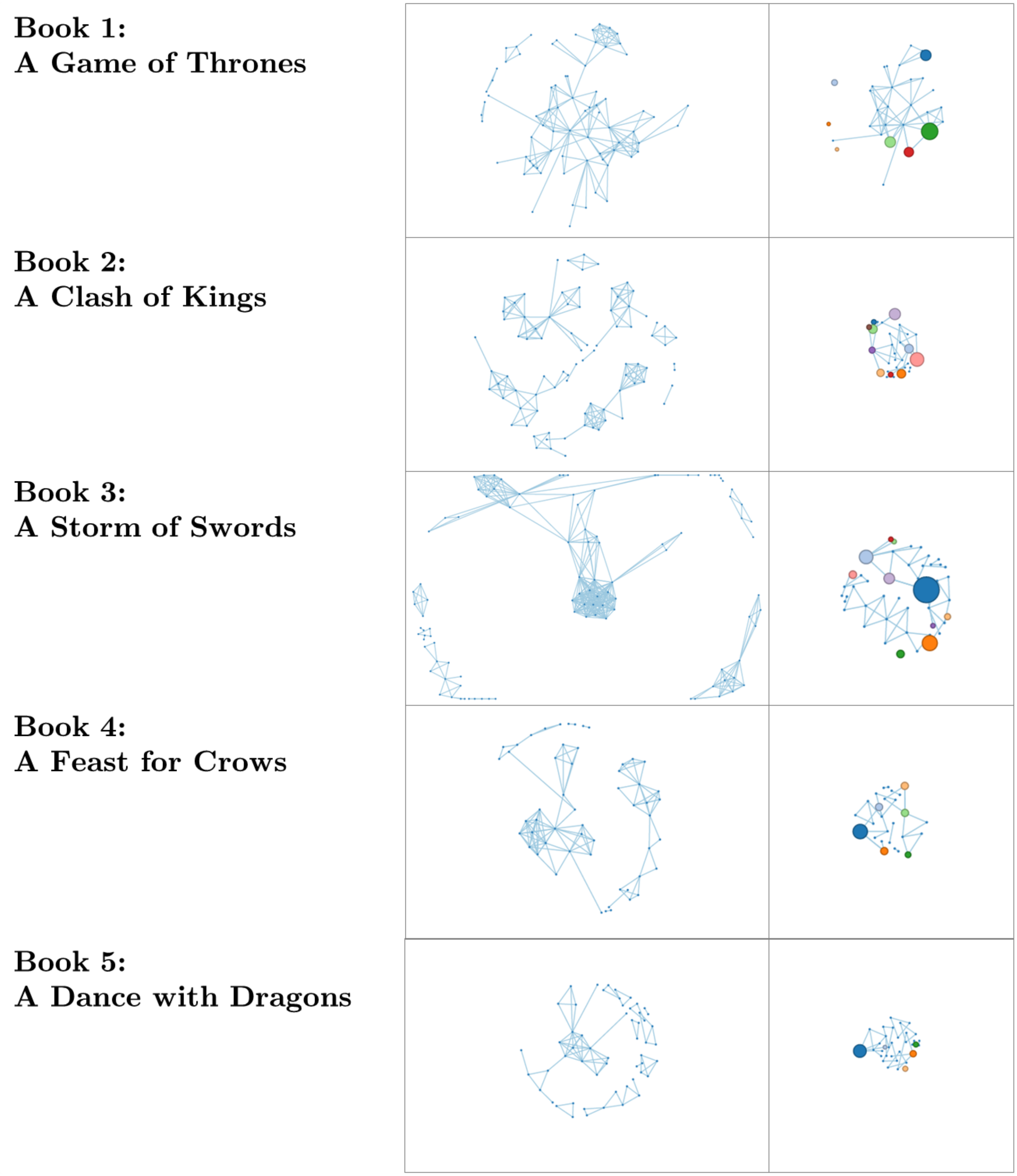


Figure 12: A look at the relationships between characters over time for all five books in A Song of Ice and Fire by George R. R. Martin in terms of mutual events.

The piece missing with this analysis is the ability to track the change of character relationships from one book to the next. A feature that would aid this process is the addition of optional labels on some of the characters in the clusters.

References

[1] Edoardo M Airoldi et al. "Mixed membership stochastic blockmodels". In: *Advances in Neural Information Processing Systems*. 2009, pp. 33–40.

[2] Arash A Amini et al. "Pseudo-likelihood methods for community detection in large sparse networks". In: *The Annals of Statistics* 41.4 (2013), pp. 2097–2122.

[3] Brian Ball, Brian Karrer, and M. E. J. Newman. "Efficient and principled method for detecting communities in networks". In: *Phys. Rev. E* 84 (3 2011), p. 036103. DOI: `10.1103/PhysRevE.84.036103`. URL: `http://link.aps.org/doi/10.1103/PhysRevE.84.036103`.

[4] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. *Gephi: An Open Source Software for Exploring and Manipulating Networks*. 2009. URL: `http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154`.

[5] Peter Bickel et al. "Asymptotic normality of maximum likelihood and its variational approximation for stochastic blockmodels". In: *The Annals of Statistics* 41.4 (2013), pp. 1922–1943.

[6] *Books about US Politics*. `http://networkdata.ics.uci.edu/data.php?d=polbooks`.

[7] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. "D3: Data-Driven Documents". In: *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011). URL: `http://vis.stanford.edu/files/2011-D3-InfoVis.pdf`.

[8] Alex Couture-Beil. *rjson: JSON for R*. R package version 0.2.12. 2013. URL: `http://CRAN.R-project.org/package=rjson`.

[9] Gabor Csardi and Tamas Nepusz. "The igraph software package for complex network research". In: *InterJournal* Complex Systems (2006), p. 1695. URL: `http://igraph.sf.net`.

[10] Jerome Cukier. *Events in the Game of Thrones*. Dec. 2012. URL: `http://www.jeromecukier.net/projects/agot/events.html`.

[11] Jordi Duch and Alex Arenas. "Community detection in complex networks using extremal optimization". In: *Physical review E* 72.2 (2005), p. 027104.

[12] Cody Dunne and Ben Shneiderman. "Motif simplification: improving network visualization readability with fan, connector, and clique glyphs". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2013, pp. 3247–3256.

[13] Tim Dwyer et al. "A comparison of user-generated and automatic graph layouts". In: *Visualization and Computer Graphics, IEEE Transactions on* 15.6 (2009), pp. 961–968.

[14] Santo Fortunato. "Community detection in graphs". In: *Physics Reports* 486.3 (2010), pp. 75–174.

[15] Christopher Gandrud. *d3Network: Tools for creating D3 JavaScript network, tree, dendrogram, and Sankey graphs from R*. R package version 0.5.1. 2014. URL: `http://CRAN.R-project.org/package=d3Network`.

[16] Emden R. Gansner and Stephen C. North. "An open graph visualization system and its applications to software engineering". In: *SOFTWARE - PRACTICE AND EXPERIENCE* 30.11 (2000), pp. 1203–1233.

[17] M. Girvan and M. E. J. Newman. "Community structure in social and biological networks". In: *Proceedings of the National Academy of Sciences* 99.12 (2002), pp. 7821–7826. DOI: `10.1073/pnas.122653799`. eprint: `http://www.pnas.org/content/99/12/7821.full.pdf+html`. URL: `http://www.pnas.org/content/99/12/7821.abstract`.

[18] Jiqiang Guo, Alyson G Wilson, and Daniel J Nordman. "Bayesian Nonparametric Models for Community Detection". In: *Technometrics* 55.4 (2013), pp. 390–402.

[19] Derek Hansen, Ben Shneiderman, and Marc A Smith. *Analyzing social media networks with NodeXL: Insights from a connected world*. Morgan Kaufmann, 2010.

[20] Jake M Hofman and Chris H Wiggins. "Bayesian approach to network modularity". In: *Physical review letters* 100.25 (2008), p. 258701.

[21] Brian Karrer and Mark EJ Newman. "Stochastic blockmodels and community structure in networks". In: *Physical Review E* 83.1 (2011), p. 016107.

[22] Charles Kemp et al. "Learning systems of concepts with an infinite relational model". In: *AAAI*. Vol. 3. 2006, p. 5.

[23] Felix Kling. *JSNetworkX: A JavaScript port of the NetworkX graph library*. `http://felix-kling.de/JSNetworkX/`. 2014.

[24] Valdis Krebs. *Divided We Stand... Still*. `http://www.orgnet.com/divided2.html`. 2004.

[25] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. "Benchmark graphs for testing community detection algorithms". In: *Physical Review E* 78.4 (2008), p. 046110.

[26] Jure Leskovec, Kevin J Lang, and Michael Mahoney. "Empirical comparison of algorithms for network community detection". In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 631–640.

[27] Jure Leskovec et al. "Statistical properties of community structure in large social and information networks". In: *Proceedings of the 17th international conference on World Wide Web*. ACM. 2008, pp. 695–704.

[28] George RR Martin. *A Clash of Kings*. Random House Digital, Inc., 1998.

[29] George RR Martin. *A Dance with Dragons*. Random House Digital, Inc., 2011.

[30] George RR Martin. *A Feast for Crows*. Bantam, 2005.

[31] George RR Martin. *A Game of Thrones*. Random House Digital, Inc., 1996.

[32] George RR Martin. *A Storm of Swords*. 2000.

[33] Cathleen McGrath, Jim Blythe, and David Krackhardt. "Seeing groups in graph layouts1". In: *Connections* 19.2 (1996), pp. 22–29.

[34] Mark EJ Newman. "The structure and function of complex networks". In: *SIAM review* 45.2 (2003), pp. 167–256.

[35] Mark EJ Newman and Michelle Girvan. "Finding and evaluating community structure in networks". In: *Physical review E* 69.2 (2004), p. 026113.

[36] Krzysztof Nowicki and Tom A B Snijders. "Estimation and prediction for stochastic blockstructures". In: *Journal of the American Statistical Association* 96.455 (2001), pp. 1077–1087.

[37] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2014. URL: `http://www.R-project.org`.

[38] RStudio Inc. *shiny: Web Application Framework for R*. R package version 0.4.0. 2013. URL: `http://CRAN.R-project.org/package=shiny`.

[39] Wayne W Zachary. "An information flow model for conflict and fission in small groups". In: *Journal of anthropological research* (1977), pp. 452–473.

A Technical Appendix

gravicom utilizes three main pieces of software to establish interactive user control of a random graph as sketched out in Figure 13, which are Shiny, D3, and igraph. These are used, respectively, for server/client interaction management, user interface and graph layout, and data formatting, respectively. In the following subsections, we describe the purposes of these three components in more detail.

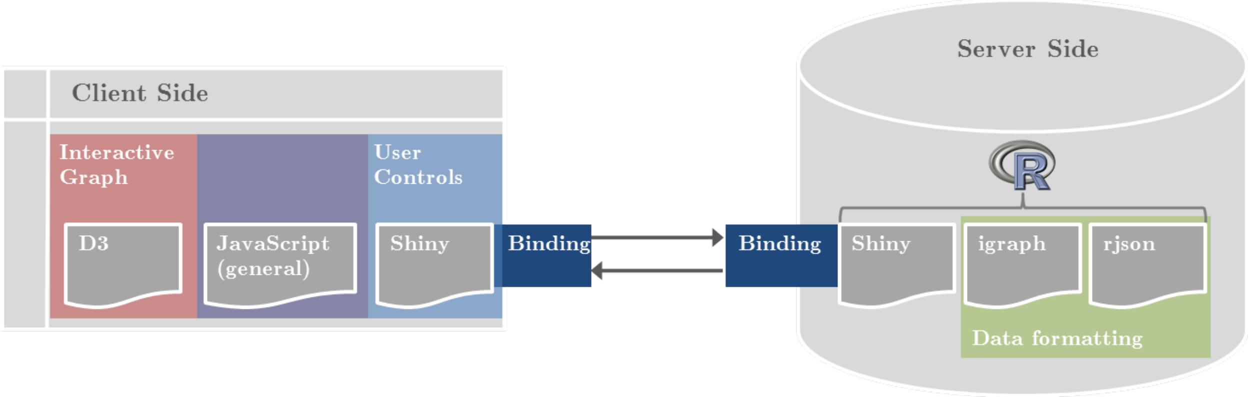


Figure 13: Relationship between client and server, specifically focusing on how data travels between the two.

There are very minimal software requirements for a user of gravicom. The client simply needs to have a JavaScript enabled internet browser with HTML5 compatibility, something which almost any modern browser fulfills (an exception is IE8 and below).

The server side requirements are more extensive, but this does not affect the user of gravicom, only those wanting to host their own instance of the application. To host gravicom, a Linux server is required, with the following installed:

- Node.js (0.8.16 or later)
- R (2.15 or later)
- Shiny R package, installed into the machine-wide site library.
- Shiny Server

A.1 Shiny

Shiny [38] is an R package created by RStudio that enables R users to create an interactive web application that utilizes R as the background engine. Through default methods to build user interface elements in HTML and a handle to the server side code, Shiny is a simple way to turn R code into a website.

gravicom uses the Shiny functionality to create user controls, pass correctly formatted data to the client, and as a means to display summary information regarding the user's interactions with a graph at any point in time. In this context, Shiny serves as the translator between the formatted data and what the user sees and interacts with on his screen.

A.2 D3

D3 [7] stands for "Data Driven Documents" and is a JavaScript library developed and maintained by Mike Bostock with the purpose of visualizing and interacting with data in a web-based interface. It is freely available from `http://www.d3js.org`. The library facilitates manipulation of HTML elements, SVG (scalable vector graphics), and CSS (cascading style sheets) with the end goal of rendering animations and providing user interactions that are tied to the underlying data. The key idea behind the library is that Document Object Model elements are completely determined by the data. The Document Object Model (DOM) is a convention for representing and interacting with objects in HTML, XHTML and XML. So, rather than adding elements to a web page to be viewed by users, D3 allows users to see and interact with graphical representations of their data in a web framework.

gravicom uses D3 to handle all graphical displays and user interactions with the graph. The data is passed to the client and able to be used through Shiny's input bindings. It is crucial that the data has been formatted correctly at this point for the JavaScript to properly function. For this reason, we limit the file types being passed into the tool to a robust graph-specific type.

At this point in the page lifecycle, the graph nodes are tied to circles and the edges are tied to paths on the page. User manipulations such as selecting, dragging, and grouping are handled by D3 and data is passed back to the server via Shiny's output bindings to allow for communication between user and the R engine underneath. This is illustrated in Figure 14. What this means is that all visualization and user interaction with the graph are accomplished using JavaScript, more specifically the library D3. Shiny and R serve as the framework on which the data sits, but when the user touches the data they are doing so through the JavaScript elements.

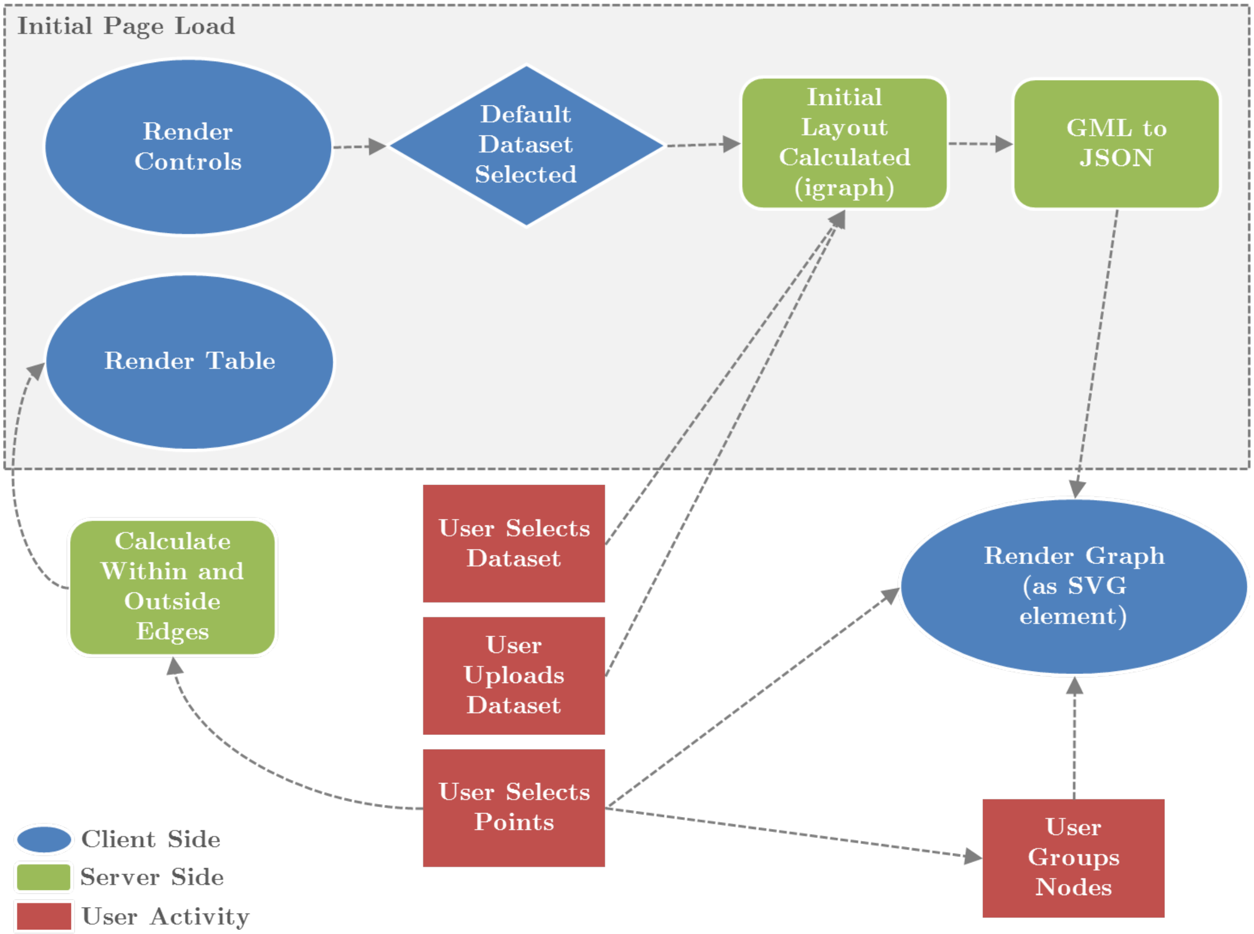


Figure 14: Page lifecycle beginning from on load. User actions are highlighted in red, server actions in green, and actions completed on the client side are highlighted in blue.

A.3 igraph

igraph [9] is a software package used for creating and manipulating undirected and directed graphs. It is a cross-language package available for C, R, python, and Ruby. igraph also supports multiple graph file formats and visualization of graph structures.

gravicom utilizes two parts of igraph, first is the conversion from a gml file to an XML file. The gml file format, short for Graph Modelling Language, is a hierarchical ASCII-based file format for describing graphs. Below is an example gml file of an undirected graph consisting of two nodes linked by a single edge. The important points to note are that node identifiers (id) have to be numeric. An edge consists only of source and target ids of the nodes it connects, while nodes can have other attributes, e.g. `value` in the example. For a directed graph, the parameter `directed` has to be set to 1, which will result in the edge information on target and source being evaluated accordingly.

```
## graph
## [
##   directed 0
##   node
##   [
##     id 0
##     label "Node 1"
##     value 100
##   ]
##   node
##   [
##     id 1
##     label "Node 2"
##     value 200
##   ]
##   edge
##   [
##     source 1
##     target 0
##   ]
## ]
```

For the conversion from an XML file to a JSON file we make user of the R package `rjson` [8]. JSON is the native data format used in D3, which makes working with data in the D3 library incredibly straightforward. Here is our example in the finalized JSON format:

```
## {
##   "nodes":
##   [{"id":"n0","v_id":"0","v_label":"Node 1","v_value":"100"},
##    {"id":"n1","v_id":"1","v_label":"Node 2","v_value":"200"}],
##  "edges":
##   [{"source":0, "target":1}]
## }
```

Our example data will yield the graph in Figure 15.

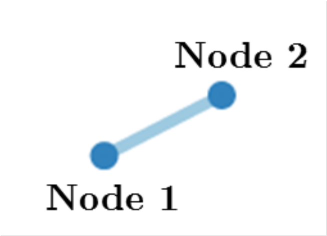


Figure 15: Graph created from sample gml file.

The second use of igraph within gravicom is to compute initial x and y coordinates for the nodes of the graph using a force-driven layout. This provides the initialization for the force-layout algorithm in D3. This reduces the computational load on the clients' side and helps minimize unnecessary movement by the nodes. This is critical as the extra movement at the loading of the pages creates an unnecessarily chaotic start to the user's experience.