SCHOOL OF ECONOMICS AND MANAGEMENT

Department of Economics

Data Analytics and Business Economics

DABN01: Master Thesis

May 2022

# Interactive Network Visualization of Insurance Portfolios

**Authors:**

Gustavo Lemos Borba

Stella Sofia Sologaistoa

**Supervisors:**

Jonas Wallin

Johan Larsson

## Abstract

Companies are seeking new ways to analyse data, and network visualizations are a great tool to enhance the ability to understand relational data. The objective of this thesis project was to build an interactive network visualization of an insurance company's portfolio. The resulting visualization helps users catch sight of patterns and outliers that might, otherwise, be overlooked. The results were achieved through the combination of R packages shiny and visNetwork and include both text and slider inputs for precise data filtering, as well as tools for highlighting, zooming, and interacting with nodes for more detailed information.

Keywords: Networks, Interactive Visualizations, Insurance Cases, Data, Analytics

## Contents

# 1. Introduction

Insurance cases are complicated, involving many different parts, such as insurance policies, parties, claims, and money. In addition, there might be fraudulent activities that are hard to discover, which might lead to considerable losses to the business, increased premiums, and customer losses. Investigations of insurance fraud in Sweden amount to over 10,500 cases a year, which results in 505 million SEK worth of claims denied after the investigation (Larmtjänst, 2021). Therefore, there is a need for tools that aid in the discovery of such fraudulent and illegal activity. One way to detect this activity is by using visual elements to understand the data, more specifically, via an interactive network visualization that can show relationships between policies, parties, and claims.

Trygg-Hansa is an insurance company founded in 1928 in Sweden. It offers a wide range of insurances such as car, home, personal, other vehicles, pregnancy, and business insurance. Trygg-Hansa, however, currently has no general solution to visualize networks of insurance cases interactively. The insurance company proposed that a graphical user interface (GUI) for visualizing insurance portfolio data as a network could be the best solution for their need. Consequently, the aim of this work is to both create a visualization of the relationship between insurance claims and parties, as well as to provide filter functionalities to facilitate the visualization of specific data points. This will ensure that end users, such as underwriters and claim handlers, will be able to gain intuition and an overview of the impact that the claims and their relationships might produce on their objectives. By developing a dashboard with this visualization, we will ensure that all the teams interested in this information will be able to understand it better and consequently make more informed decisions.

This thesis is divided into four parts. In Methodology, we cover what type of data we use in the project, theory regarding networks, and software packages we used to construct the visualization. Then, in Results (**section 3**), we outline how we built the interactive network visualization and the dashboard. In the Discussion (**section 4**), we discuss our contributions and limitations of the result and provide directions for future work.

## 2. Methodology

This section focuses on the approach we took in designing the visualization of the insurance portfolio. Having the data provided by the insurance company, the interactive network graph was built and later the dashboard was created. This section includes theory on network visualizations, followed by a brief overview of the data used, and lastly a description of the two main R packages used in this project.

### 2.1. Networks

A network is a set of nodes connected by a set of edges (Newman, 2010). The nodes are the elements or characteristics defined, which in our case are claims or parties. And the edges show the connection between each node; therefore, each edge consists of two endpoints (Hsu & Lin, 2008). In **Figure 1**, we can see a simple network and its components. Networks occur in many different disciplines: biology, social sciences, communication, chemistry, economics, logistics, economics, etcetera. For example, they can easily show airline transportation, where the airports act as nodes and the edges are the flights (Steen, 2010).

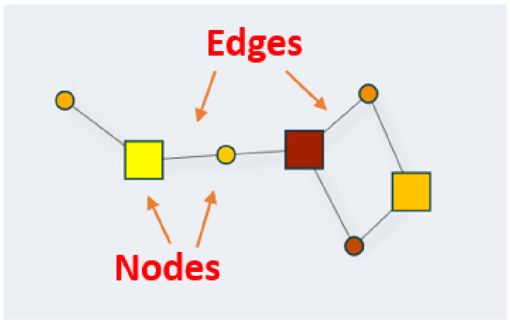

**Figure 1** A simple network to understand how the nodes and edges are visually represented

Edges can be directed and undirected. Directed edges define the flow of one node to another, such that switching the direction of the edges would change the structure of the network. Undirected edges just connect two nodes without specifying a direction (Tyner, Briatte & Hofmann, 2017). In our project, we use undirected edges.

Network visualizations are useful because they enable the user to see connections between elements that might have not been apparent by just observing the data. For example, in **Figure 2**, we can see the data to the left and we can see the corresponding network to the right. If we had seen only the table, probably we would take a lot of time to interpret that there are connections in the data. In contrast, when we see the visualization, we know instantly which nodes are connected and by which edges. Therefore, observing data in a graphical way instead of relying on tables is useful.

| Node ID | Target ID |
|---|---|
| 1 | A |
| 1 | B |
| 1 | C |
| 2 | B |
| 2 | D |
| 2 | C |
| 3 | C |
| 3 | B |
| 3 | D |
| 4 | C |

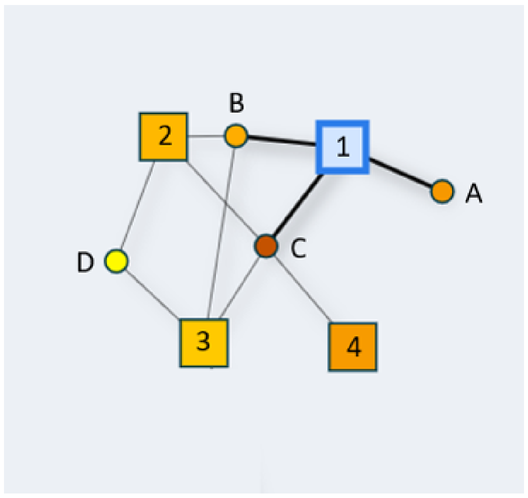

**Figure 2** Comparison between observing the data in a table and observing a network visualization

In addition, adding color to the network graph is of great importance because it can help the end user to better understand the meaning of the graph and because it can give further information that might otherwise be lost (Freeman, 2016). But, at the same time, network visualizations may be difficult to decipher if the data contains thousands of observations. Like any other type of visualization, it might lose interpretability when it becomes too cluttered or when the labels overlap and become unreadable. For that reason, there are techniques to overcome these challenges, such as designing an interactive network visualization instead of a static network visualization. Interactive network visualizations allow the user to choose via an input box, such as a drop-down menu, buttons, or tabs, customizable features that show different subsets of the graph, avoiding conglomerate visualizations (Namata et al. 2014). These features can be, for example, zooming in or out, filtering out nodes or edges, and hovering over a specific point on the graph. Another technique is to animate networks so that it makes transitions between time steps, resulting in a dynamic visualization where nodes appear, disappear, or move to produce a different layout each time (Ma & Muelder, 2013). In this project, the dates where each node connects to another node correspond to such time steps.

## 2.2. Data

Trygg-Hansa provided a data set specifically for use in the project, consisting of real insurance portfolio data that has been modified in two ways. First, the data was anonymized so that it couldn't be traced back to any particular person, in order to avoid having the privacy and security of the people entrusting their data to Trygg-Hansa be compromised. And secondly, they wanted to have a model that could single out specific characteristics and metrics by adapting the input data whilst still getting an interactive network as an output.

The data used for this visualization contains two data sets: a node and an edges data set. The node data set shows the type of insurance characteristics, such as parties, policies, or claims. The edges data set shows the connection paths that indicate the relationship between nodes. The nodes data set contains three columns with the node ID, the value, and the type of node (claim, party, policy, etc.). The edges data set also contains three columns, but it shows the relationship between nodes, meaning that it shows the node ID in its source, the node ID at its target, and the date that this relationship started. In **Table 1**, we can see the description of the data given by Trygg-Hansa.

**Table 1** Data provided by Trygg-Hansa to make an interactive network visualization

| Name | Description |
|---|---|
| Node's ID | Insurance claim or party ID |
| Value | Value related to the node, could be customer value, customer retention, or likelihood to purchase a product |
| Type of node | Tells if the node is a claim, party, policy, etc. |
| Source ID | Node ID at its source |
| Target ID | Node ID at its target |
| Date | Date when source and target nodes connected |

## 2.3. Main R Packages Used

### Shiny

The R package shiny (Chang et al. 2021) has been successful in simplifying the creation of interactive web applications by providing a set of user interface (UI) functions and introducing what is known as reactive programming, which automatically checks dependencies within a code (Wickham, 2020). It allows R users to build web applications without requiring HTML, CSS, or javascript knowledge, and on top of that, it is compatible with many of the facilities that R has to offer. Applications in shiny require two components: a UI function and a server function (Shiny, 2017). In simple terms, the server function defines how the app is going to work (Wickham, 2020) while the UI function translates and presents the results to the interface. Shiny is a form of server-side application, which means that all the information processing occurs on a web server, and shiny itself builds a web server designed to host shiny applications (Allen, 2020). These servers can be published on the internet or privately shared within a company or a team.

To create an interactive user interface, packages such as shiny, shinyWidgets (Perrier, Meyer & Granjon, 2022), and shinydashboard (Chang & Borges, 2021) can be used. Shiny allows users to create a user interface by defining the header, sidebar, and body of a dashboard page. The header can include the title and drop-down menus. The sidebar may include menu items or shiny inputs that will affect the visualization output (Shiny Dashboard Structure, 2014). The interface's main part, the body, is where the output is displayed. The server function requires an input and output object; in this project, for example, there are two different inputs and outputs:

- text input: text that should be typed by the user
- text output: text returned depending on the text input
- slider input: slider with dates in chronological order that can be changed by the user
- visNetwork output: the visNetwork output filtered by the text and slider inputs

## visNetwork

The main R package used in the project was the visNetwork package, which is used to build network visualizations in R (Almende, Thieurmel & Titouan, 2021). The main function in the package is `visNetwork()`, which requires two different data sets: one containing nodes and another containing edges. These data sets are expected to have specific column names for the function to work properly. The nodes data set should at least have an `id` column containing the unique nodes' identification code, while the edges data set must have a `from` and `to` column that creates connections from one ID to another (McNulty, 2022).

Furthermore, a simple way to modify and add features to the network is by appending columns with information that can be read and understood by `visNetwork()`. In this work, the following columns were incorporated into the nodes data set:

- `title`: displayed as a tooltip when a user hovers the mouse over a node or edge
- `color`: used to define specific colors for the nodes. In this project we used two different color specifications:
    - `color.background`: determines the background color of each node
    - `color.border`: determines the border color of each node
- `shape`: used to define specific shapes for the nodes
- `x`: used to define x-coordinates to the nodes
- `y`: used to define y-coordinates to the nodes

The visNetwork package can automatically recognize columns with the specific names listed above, but it is not limited to those customizations. Different customization options are available for use in `visInteraction()`, `visOptions()`, `visLegend()`, and others.

- `visInteraction()` allows users to enable zoom, drag, and select options in the visualization.
- `visOptions()` allows users to highlight the nearest nodes and edges by hovering or clicking on a node, create interactive selection options for specific grouping variables, and set default colors for each group (Luke, 2015).
- `visLegend()` allows users to include custom legends in the visualization.

These additional features were important for increasing the interactivity of the visualization. A section of the R code exemplifying their use in this project can be observed in **Listing 1**.

**Listing 1** R code example of visNewtork, visSettings, visInteraction, and visLegend

```
visNetwork(nodes = nodes.net,
           edges = edges.net,
           main = "Network Visualization",
           background = NA) %>%
visNodes(color = list(highlight = NA),
         borderWidthSelected = 3) %>%
visEdges(color = "#0D0D0D",
         hoverWidth = 5,
         selectionWidth = 3.5,
         shadow = TRUE)%>%
visInteraction(dragNodes = TRUE,
               multiselect = TRUE,
               navigationButtons = FALSE,
               zoomView = TRUE) %>%
visOptions(highlightNearest = list(enabled = TRUE,
                                   degree = nrow(edges.net),
                                   hover = TRUE),
           nodesIdSelection = list(enabled = TRUE,
                                   useLabels = FALSE,
                                   selected = node,
                                   style = 'width: 0px;
                                            height: 0px;
                                            background: #f8f8f8;
                                            color: black')) %>%
visLegend(useGroups = FALSE,
          addNodes = legend,
          width = 0.08)
```

## 3. Results

In this section, we explain how we constructed the interactive network visualization, as well as the choices behind some of its features. We also explain how the dashboard works and what its properties are.

### 3.1. The Network

We took a series of steps in designing this project, from choosing the network nodes' shape format to fixing the nodes' coordinates `x` and `y`. Here we are describing each of the steps towards the final network visualization,

#### Shapes

The original nodes data set used in this project contained a column for the nodes' types. The types are represented by the letters `s`, `c`, `e`, `p`, and `h`. Type `s` nodes were known to be insurance parties, while nodes of all other types could represent a variety of things, such as claims or policies. For simplicity, we ignored all possible types, and categorized nodes of type `s` as `Party`, and `Claim` otherwise.

The `visNetwork()` function recognizes that a column named `shape` will contain the necessary information to define the shape of each node. Having that in mind, an `ifelse()` function was used to create a new column, evaluating to `square` if a node has type `s` (`Party`) and `circle` otherwise (`Claim`), and the list was then added as the `shape` column to the data set. The resulting network can be observed in **Figure 3**, in which squares and circles are representing insurance parties and claims, respectively.

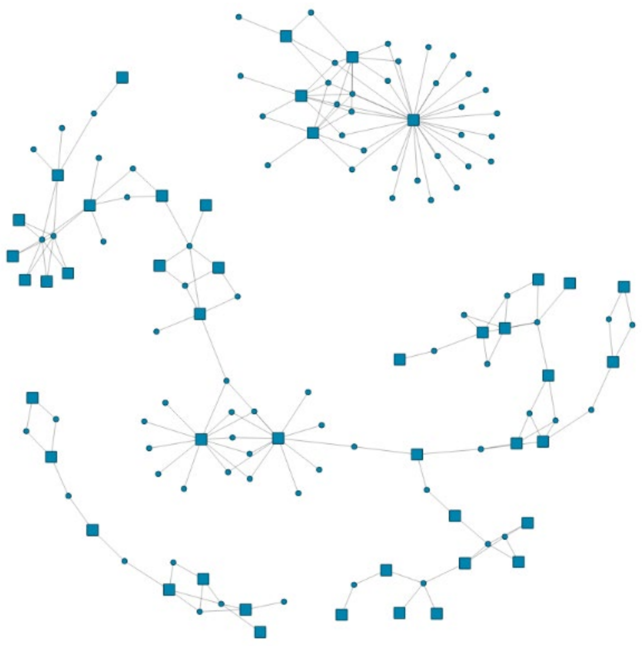

**Figure 3** Network visualization of nodes (parties and claims) with shapes dependent on the node's type. Here, each square represents an insurance party, while a circle represents a specific claim.

## Colors

`visNetwork()` has different settings that allow choosing the network color. For our network, we created two columns, `color.background` and `color.border`, which contained the information required by the function.

The nodes' background color was extracted from a color palette that goes from yellow to dark red, as shown in **Figure 4**, and is dependent on the `value` column that is included in the original nodes data set. Consequently, higher node values result in darker red background colors. The colors were chosen to give a sense of which value is higher or lower because, despite the inability of humans to judge the distance between nearly similar colors, we can still quickly distinguish what is higher or lower (Grant, 2018).

**Figure 4** Palette of colors used for the nodes' background color

All nodes are configured to have a very dark cyan border color in order to add contrast between the nodes and the background.

The resulting network can be seen in **Figure 5**, including the shapes previously assigned to nodes' types, and the colors assigned to nodes' respective values.

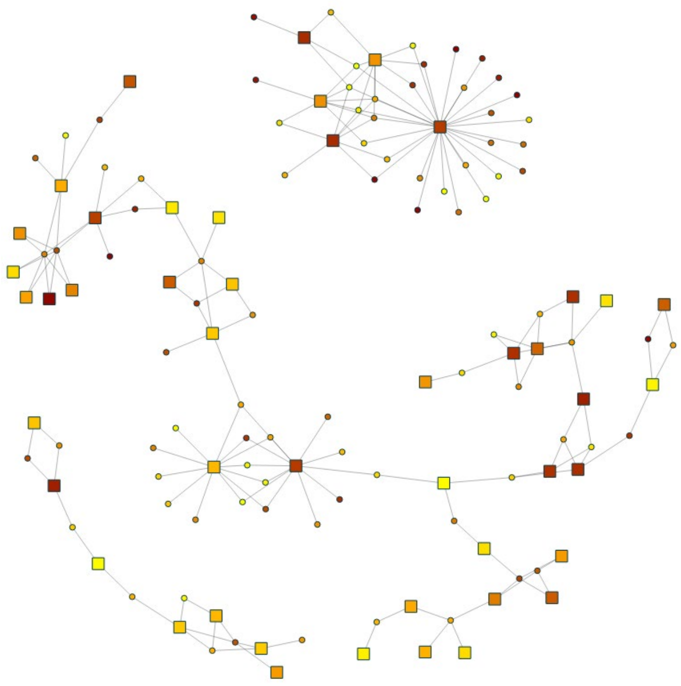

**Figure 5** Network visualization of the nodes, combining the shapes determined in **Figure 3**, and the colors defined in **Colors**

## Legend

The legend used in the visualization explains the meaning of the two shapes used and how the color changes depending on the node's value. The legend was added through `visLegend()`, which allows the developer to tweak multiple settings strictly related to the network's legend, and the result can be seen in **Figure 6**.

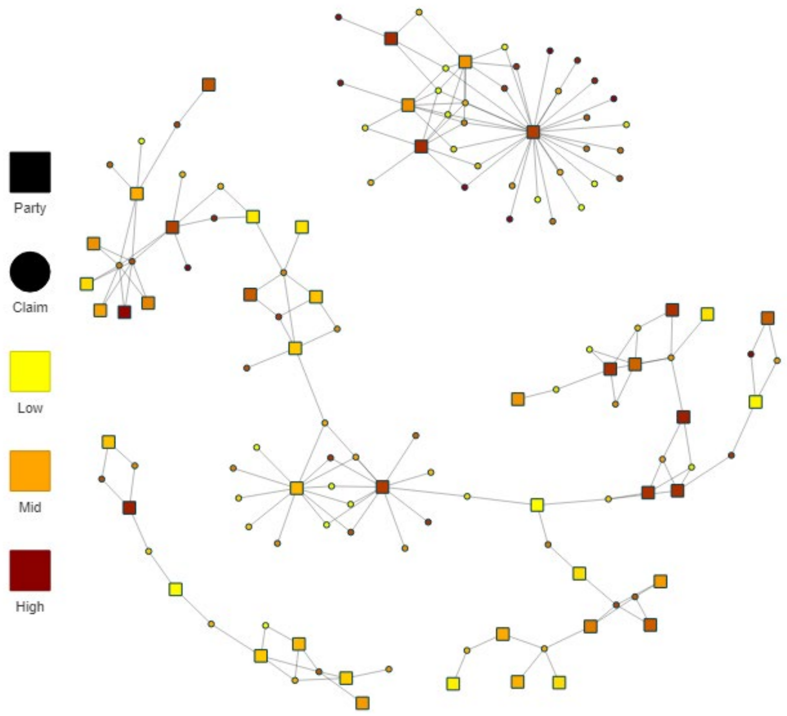

**Figure 6** Network including legends on the left side of the visualization indicating the shapes and colors

## Titles

`visNetwork()` also allows the use of column inputs named `title`. The titles can be added to the nodes data set and to the edges data set and will be shown when a user hovers the mouse over a specific node or edge in the network.

When a user hovers the mouse over a specific node, the ID and Value of that node are displayed in a box (**Figure 7**).

When a user hovers the mouse over a specific edge, the two IDs connected by that edge and the date in which the connection was made are displayed in a box (**Figure 8**).

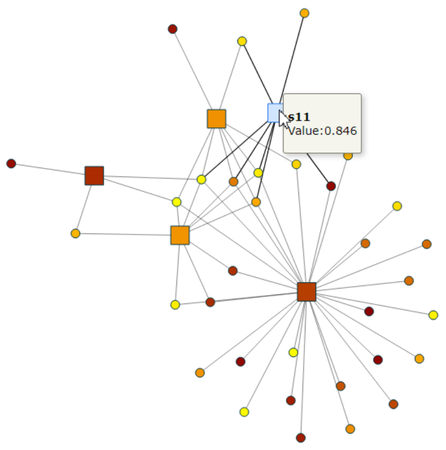

**Figure 7** Example of how a user can access a node's informational box, containing ID and value, by hovering over a specific node

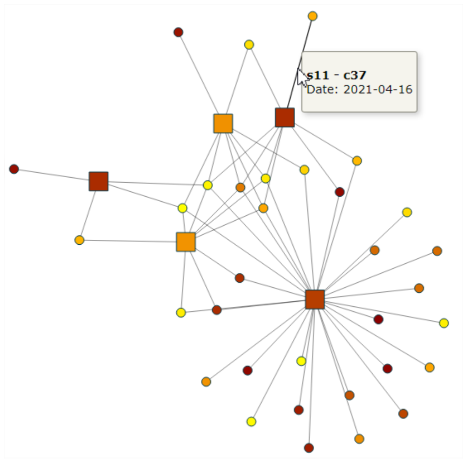

**Figure 8** Example of how a user can access an edge's informational box, containing the IDs of the nodes connected and the date of connection, by hovering over a specific edge

## Node Coordinates

By the end of the project, a user will be changing inputs that will directly impact the network. This results in a new random layout with every change. To avoid this from happening, it is crucial to determine the coordinates of each displayed node.

When new columns named `x` and `y` are added to the nodes data set, `visNetwork()` recognizes that those will be the nodes' positions when the network is displayed. It can be challenging to manually determine specific nodes' coordinates that will be optimal for different data sizes, so a function from the igraph package (Csardi & Nepusz, 2006), `layout_nicely()`, was used. This function uses a graph object created with the same data used in the `visNetwork()` and attempts to choose an appropriate layout algorithm for the network visualization. Therefore, by using `layout_nicely()`, it was possible to extract the coordinates to create the new `x` and `y` columns.

## 3.2. The Dashboard

With the visual aspects of the network in place, the next step was to start building the user interface, including filtering options that are going to allow the company to visualize specific data points. The aim here was that the user would have filtering options available for the ID and date inputs. These were designed as text and slider inputs, respectively, and their effects were presented as two different outputs: a text and a visualization.

### Text Input

The main input of the user interface, the text input, allows the user to search for a specific ID by typing it in the designated space. The text input can be used as a filter for what the user wants to observe, as only the parties and claims connected to the ID typed will be presented on the screen.

For instance, let us imagine that a Trygg-Hansa team wants to visualize the claims in which a specific third party, `X`, is involved. Instead of looking at a long and complicated table, the team can open the application developed in this work and type the ID assigned to `X`. The application will then automatically filter all the claims connected to `X` in any form.

### Text Output

To help the user understand what is selected by the typed input, a red-colored text is displayed between the input box and the visualization output, containing either the ID selected by the program or, in case the user's input is not found in the data set, an "ID NOT FOUND" message.

An example of how the application reacts to the text input can be seen in **Figure 9**. The hypothetical party, `s5`, is selected, and the network was filtered to show the connected nodes, as well as a text confirmation of the ID selected.

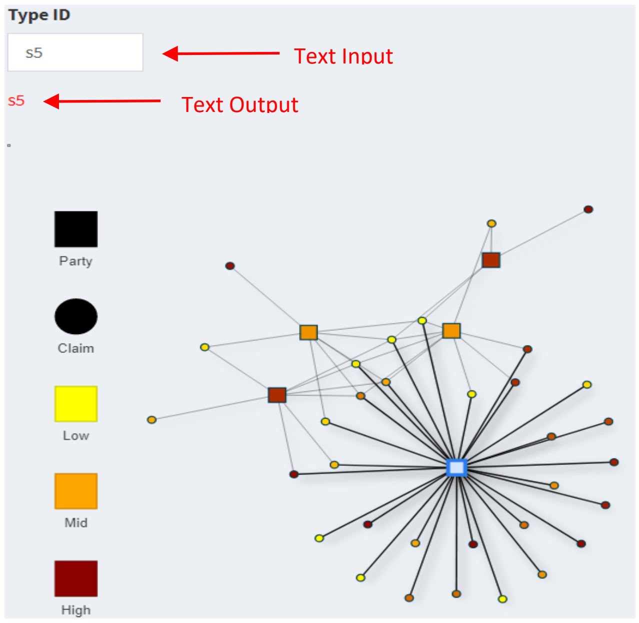

**Figure 9** Shiny application outputs when the user provides a text input. Here, `s5` was provided as the ID, and the network was filtered to show only nodes related to `s5`.

### Slider Input

The slider input serves as a date filter, allowing the user to specify a date range during which the nodes should be presented. The date slider is useful for investigating the changes in the network over time. The dates included in the slider will vary according to the ID typed in the text input, such that the minimum date choice is the date in which the typed ID was created. This dependability avoids errors that would otherwise occur when a user chooses a date for which the ID does not exist.

In practice, when the user types an ID the program automatically shows that ID's network at the maximum date range possible and, from there, the user can manually move the slider to a desired date. **Figure 10** shows an example of how it is possible to visualize the evolution of the *s*5 network over time. Together, the text input filters out all unconnected claims and the slider input only allows claims that were made in between the dates chosen by the user to be visualized.

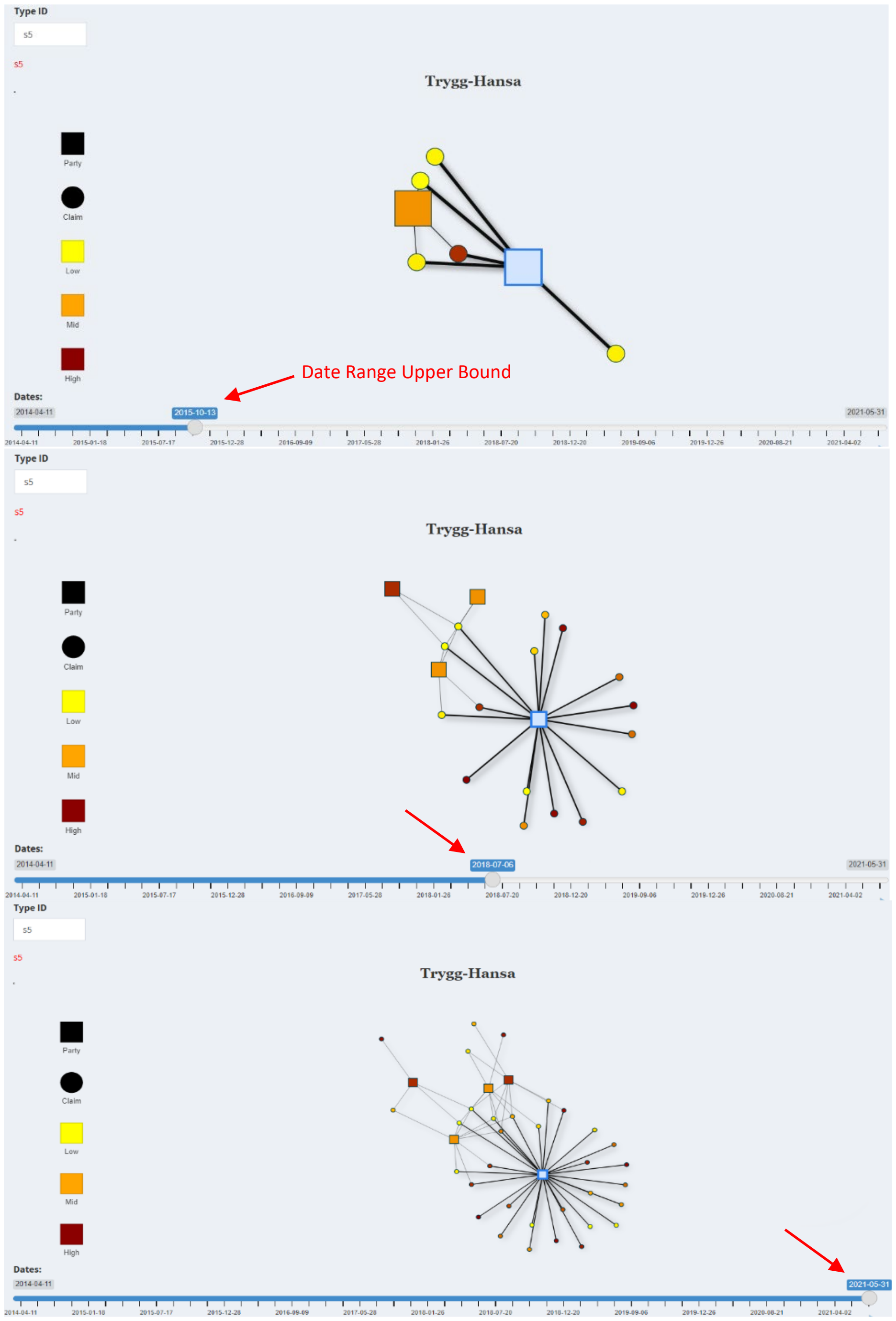

**Figure 10** Date slider input in action

### Sidebar and Main Panel

As mentioned in the description of the shiny package (**section 2.3**), the sidebar in shiny can include different menu items and inputs. In this project, the only item displayed presents the user with the network visualization of the data. In this case, the dashboard sidebar seems to be useless and could be left out of the dashboard. However, the Trygg-Hansa team that will receive this project may want to add more items to the dashboard, so the intention here is to set an example that is going to facilitate their work in the future.

The dashboard main (right) panel displays the information related to the menu item selected by the user, such as the text input and output, visualization output, and slider input.

## 3.3. Final Interface

By combining the work done in **3.1. The Network** and in **3.2. The Dashboard**, we get to the final result of this project, which can be observed in **Figure 11**. The figure shows what a user from Trygg-Hansa would observe when the application is opened.

The application automatically selects the node of the earliest date from the data set. The node ID, in this case, is `s15`. This was done to avoid throwing errors related to the slider input, which requires an initial ID to be selected. From there, the user has complete autonomy: being able to choose the input ID, change the date range with the slider, zoom in and out, hover over nodes for specific information, and drag or select multiple nodes.

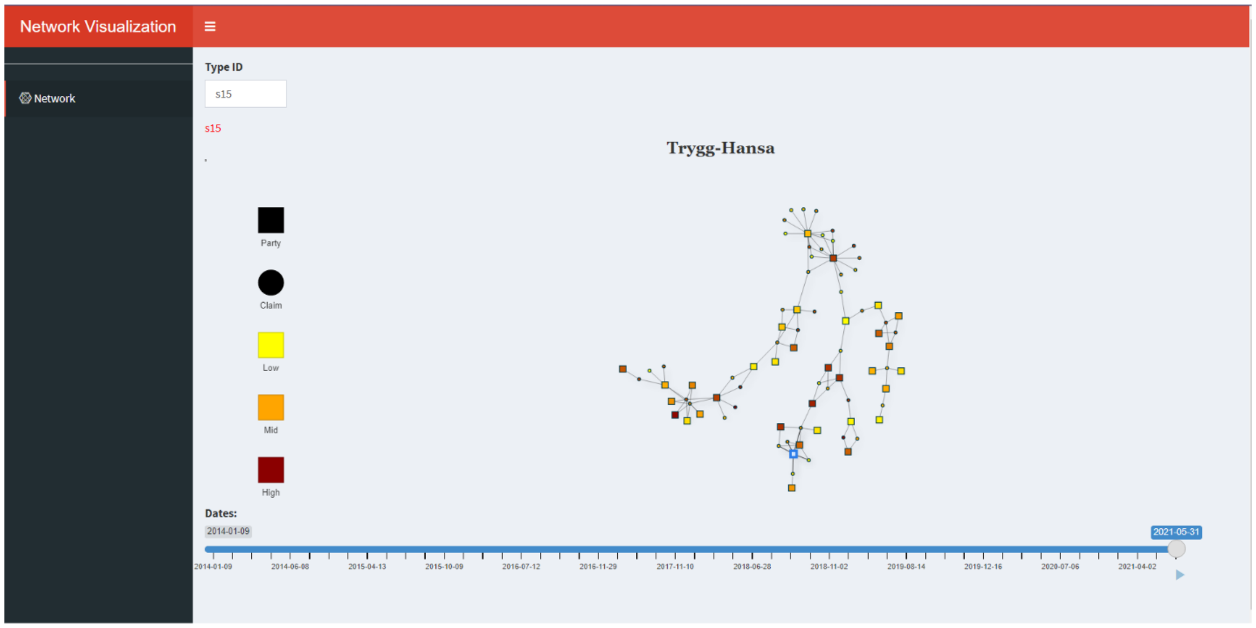

**Figure 11** Final shiny application

The application is available for testing at gustavo-borba.shinyapps.io/Project/. We encourage users to try out the different features described throughout this thesis, as well as different inputs. Some alternative input IDs for use can be found in **Table 2**.

**Table 2** Sample from the nodes dataset containing all columns added during the thesis project

| id | value | type | shape | color.background | color.border | title | x | y |
|---|---|---|---|---|---|---|---|---|
| s2 | 0.39475 | s | square | #FFB800 | #013848 | <p><b>s2</b><br>Value:0.395 | 3.49218 | 11.11478 |
| s5 | 0.79297 | s | square | #B63E00 | #013848 | <p><b>s5</b><br>Value:0.793 | 9.18752 | 1.34352 |
| c10 | 0.57615 | c | circle | #EB8900 | #013848 | <p><b>c10</b><br>Value:0.576 | 5.17543 | 10.44504 |
| c34 | 0.96161 | c | circle | #8E0400 | #013848 | <p><b>c34</b><br>Value:0.962 | -0.47263 | 4.27801 |
| e3 | 0.50760 | e | circle | #FBA000 | #013848 | <p><b>e3</b><br>Value:0.508 | 2.77042 | 9.62593 |
| e6 | 0.94903 | e | circle | #920900 | #013848 | <p><b>e6</b><br>Value:0.949 | 9.63476 | -1.07371 |
| p5 | 0.44923 | p | circle | #FFAD00 | #013848 | <p><b>p5</b><br>Value:0.449 | 10.15191 | 8.40810 |
| p12 | 0.82031 | p | circle | #AF3400 | #013848 | <p><b>p12</b><br>Value:0.82 | -1.84540 | 12.52456 |
| h10 | 0.59257 | h | circle | #E88400 | #013848 | <p><b>h10</b><br>Value:0.593 | -1.71315 | 10.62542 |

# 4. Discussion

The outcome of this project thesis was aligned with the expectations and goals set out by Trygg-Hansa. Therefore, we can say that the results were more than satisfactory. This work focused on building a user interface that could be applied to visualize Trygg-Hansa's insurance portfolio and went beyond the primary objectives by including additional useful features along the way. In this section, we summarize our contributions, discuss possible limitations of the work, and outline opportunities for future work.

## 4.1. Contributions and Limitations

Trygg-Hansa's implementation of the network visualization could be adapted to benefit many companies in different segments. Because there is a scarcity of public studies regarding network visualizations for portfolios, we believe that this project could serve as a template for future work in related areas. We organized the project code base such that other data sets could be applied to it, avoiding hard-coded paths or specific components names, thus making it easier for future users to adapt the code to their own data sets. Because shiny creates server-side applications, however, depending on the expected number of simultaneous users, it will require a powerful server in order to preserve the quality of the application. On the other hand, it is generally quicker and simpler to be designed than client-side applications, which require more advanced programming skills.

Nevertheless, there are certain limitations that companies would need to overcome as means to make good use of such visualization form, which are presented as followed. Firstly and foremost, the data set used to build the model is limited to 133 observations and 188 edges, which is unrealistic for companies that could have thousands of data points. Because of the data size, some issues that could appear in a bigger data set may have been overlooked in this project, such as the number of nodes to be displayed and the quantity of information to be presented by a selected node.

In addition, due to the shortage of time to build and deliver this project, some shiny and visNetwork settings may not have been used in the most efficient way, such as reactive functions and `visNetworkProxy()`. These settings could benefit the computational efficiency and allow for different settings that were not explored in this work.

## 4.2. Future Work

Because this project has a vast scope for use in many different industries beyond the insurance sector, future work is dependent on the end user and their visualization needs. The inclusion of more data may require some adaptations to the visualization and query settings. Future development opportunities arising from this project include limiting the number of nodes presented, adding a table with all data associated related to selected nodes, and assigning icons as the nodes' shapes to ease user understanding.

### Limit the Number of Nodes Presented

Networks can technically grow arbitrarily large and it is infeasible to visualize thousands of components at once. Untested solutions for this could be to create a threshold limiting the number of nodes that can be displayed and to make the terminal nodes expandable, so that the user of the application could manually display information that is out of the limits. Another option is to aggregate terminal nodes that are in the same group, allowing the user to manually trigger extra information. These settings could be achieved by utilizing the `visCollapse()` and `visUncollapse()` functions with a `visNetoworkProxy` object.

### Table with Associated Data

Companies will most likely have much more data information associated with the nodes and edges, so the **Titles** may not fit it all. Having a table that covers all necessary information linked to a node could bring more meaningful insights for the user.

### Icons as Shapes

Companies with multiple types of related nodes may prefer using self-explanatory icons to represent each node type. There are different sources of icons compatible with R, such as Font Awesome and Ionicons that can be used in the network.

## 5. Conclusion

Visualizing data can be useful to find patterns, trends, and outliers with data sets that contain thousands of observations. In this thesis project, we used relational data to demonstrate how the claims and parties of an insurance portfolio can be visualized. We used the programming language R and various of its packages to create an interactive network visualization that is adaptable, such that the company can modify the input data while still getting an interactive network as a result. Furthermore, we created a dashboard – a graphical user interface – to allow the end user to filter inputs and select data ranges. By interacting with the dashboard, the relational data can be explored in an intuitive and simple way.

## References

Allen, J. (2020). Introduction to Shiny Server, Available online: https://shiny.rstudio.com/articles/shiny-server.html#:~:text=Shiny%20Server%20is%20an%20open,will%20never%20leave%20your%20control [Accessed 03 May 2022]

Almende, B.V., Thieurmel, B., & Titouan, R. (2021). visNetwork: Network Visualization using 'vis.js' Library. R package version 2.1.0, Available online: https://CRAN.R-project.org/package=visNetwork [Accessed 23 April 2022]

Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., & Borges, B. (2021). shiny: Web Application Framework for R. R package version 1.7.1. Available online: https://CRAN.R-project.org/package=shiny [Accessed 14 April 2022]

Chang, W., & Borges, B. (2021). shinydashboard: Create Dashboards with 'Shiny'. R package version 0.7.2. Available online: https://CRAN.R-project.org/package=shinydashboard [Accessed 14 May 2022]

Csardi, G., & Nepusz, T. (2006). The Igraph Software Package for Complex Network Research, Available online: https://igraph.org [Accessed 24 April 2022]

Freeman, L. C. (2016). Visualizing Social Groups, *Irvine,* Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.90.6381&rep=rep1&type=pdf [Accessed 11 May 2022]

Grant, R. (2018). Data Visualization: Charts, Maps, and Interactive Graphics, New York: Chapman and Hall/CRC

Hsu, L.-H., & Lin, C.-K. (2008). Graph Theory and Interconnection Networks, Boca Raton: CRC Press

Larmtjänst AB. (2021). Insurance Fraud in Sweden. Available online: https://www.larmtjanst.se/Snabbmeny/In-English/Insurance-fraud/ [Accessed 11 May 2022]

Luke, D.A. (2015). A User's Guide to Network Analysis in R, New York: Springer

Ma, K.-L., & Muelder C. W. (2013). Large-Scale Graph Visualization and Analytics, California: IEEE Computer Society

McNulty, K. (2022). Handbook of Graphs and Networks in People Analytics. Available online: https://ona-book.org/index.html [Accessed 24 April 2022]

Namata, G. M., Staats, B., Getoor, L., & Shneiderman, B. (2007). A Dual-View Approach to Interactive Network Visualization, *Association for Computing Machinery,* Available online: https://www.cs.umd.edu/~namatag/dualnet/cikm2007-Full.pdf [Accessed 11 May 2022]

Newman, E. J. (2010). Networks: An Introduction, New York: Oxford University Press

Perrier, V., Meyer, F., & Granjon, D. (2022). shinyWidgets: Custom Inputs Widgets for Shiny. R package version 0.6.4. Available online: https://CRAN.R-project.org/package=shinyWidgets [Accessed May 14 2022]

Shiny. (2017). The Basic Parts of a Shiny App, Available online: https://shiny.rstudio.com/articles/basics.html [Accessed 23 April 2022]

Shiny Dashboard Structure. (2014). Shiny dashboard, Available online: https://rstudio.github.io/shinydashboard/structure.html [Accessed 23 April 2022]

Steen, M. (2010). An introduction to Game Theory and Complex Networks, Amsterdam: Maarten van Steen

Tyner, S., Braitte, F., & Hofmann, H. (2017). Network Visualization with ggplot2, *The R Journal,* Available online: https://hal.archives-ouvertes.fr/hal-01722543/document [Accessed 15 May 2022]

Wickham, H. (2020). Mastering Shiny, [e-book] O'Reily Media, Available at: https://mastering-shiny.org/index.html [Accessed 07 May 2022]