U.PORTO

FEP FACULDADE DE ECONOMIA UNIVERSIDADE DO PORTO

Social Network Analysis: GitHub for Organizational Management

Joana Catarina Santos de Gusmão Guedes

Dissertation

Master in Data Analytics

Supervised by

Professor Doutor João Manuel Portela da Gama

2019

Short biography

Joana Catarina Santos de Gusmão Guedes was born in Porto, Portugal. She earned her Graduate Degree in Biology at Faculdade de Ciências de Universidade do Porto in 2008 and her Masters de Degree in Molecular Oncology at Instituto de Ciências Biomédicas Abel Salazar da Universidade do Porto.

She integrated several research projects in the biological sciences field and eventually her research career crossed paths with bioinformatics and data analytics. During this period, she gained a great interest in data science, machine learning and programming and to deepen her knowledge in these fields, she enrolled in the Masters in Data Analytics at Faculdade de Economia da Universidade do Porto in 2017.

Acknowledgements

I acknowledge my supervisor, Professor João Gama, for sharing his experience and allowing me to choose this project. Thank you also for motivating me into concluding this work and making things seem much simpler.

I would also like to mention my MADSAD colleagues, Marta and Diogo, with whom I did most course projects. We were fortunate to have formed such a balanced team right from the start!

I am very lucky to have had the chance to develop this project during a Data Science internship at an outstanding company. I enjoyed the time spent there, and would like to send a big thank you to all my colleagues in Malta. Particularly my mentor, Kris, who was very dedicated and committed in supervising our project.

I thank my mother, Teresa, father, Zé Pedro, Mafalda, and my grandparents, Luísa and Manuel, for their supporting my choice to enroll in MADSAD and for their confidence in me. I believe this was the best decision I could have made.

A special appreciation to my brother, João, who was always available to help me out and share his expertise.

Last but certainly not least, I'm grateful for my boyfriend's commitment to making sure things ran as smoothly as possible so I could focus on my studies. João, thank you for your patience!

Thank you all!

Resumo

Na economia do conhecimento em que vivemos, e no sector tecnológico em particular, as empresas actuam em ambientes de grande dinamismo e onde a partilha de conhecimento e uma estratégica eficaz de comunicação são importantíssimos. A aplicação da teoria dos grafos ao estudo de estruturas sociais, designada de Análise de Redes Sociais (ARS), pode ser utilizada para estudar padrões de comunicação entre funcionários e compará-lo à organização formal.

Com este trabalho pretende-se modular a comunicação entre colaboradores de uma empresa usando seu histórico do GitHub. A empresa em questão é internacional de médio tamanho e opera no sector do jogo online. Cresceu rapidamente e, por via disso, necessitou de se reestruturar. O objetivo proposto foi o de usar ARS para avaliar a eficácia desta mudança.

O GitHub fornece um histórico de dados particularmente interessante já que é uma plataforma colaborativa onde são registadas todas as tarefas planeadas e realizadas. É também o local onde são discutidos assuntos do dia-a-dia relacionados com trabalho e onde os desenvolvedores de software partilham o código em que estão a trabalhar.

Todo o histórico do repositório "home" desta empresa no GitHub, que totaliza perto de sete anos de dados gerados, foi extraído e guardado numa base de dados de grafos (Neo4j). Esta base de dados foi depois consultada para modelar a comunicação entre os utilizadores antes e depois do evento de reestruturação. Foram criados modelos de redes pesadas e não pesadas e o melhor modelo para o propósito foi selecionado. De seguida, analisaram-se medidas de centralidade e foi feita a detecção de comunidades.

Os resultados mostram que há de facto diferenças nas redes estudadas, e que coincidem com os propósitos da reestruturação, contribuindo para afirmar que a ARS é uma ferramenta poderosa e válida para avaliar o desempenho deste tipo de decisões estratégicas.

Palavras-chave: Análise de Redes Sociais, Análise de Redes Organizacionais, GitHub, redes de comunicação

Abstract

In the knowledge economy, and in the tech sector in particular, companies operate in highly dynamic business environments where knowledge sharing and effective communication are critical to allow a fast response. By applying graph theory to the study of social structures, Social Network Analysis (SNA) can be used by companies to study employee communication patterns and compare them to the formal organizational structure.

The goal of this work is to use GitHub data and SNA tools to evaluate the effect of organizational structure changes made within a mid-size, fast-growing international company. GitHub is a very interesting source of information given that it is the platform where all tasks and projects are tracked and centralized, and where discussions are held and collaborative coding takes place.

The entire GitHub "home" repository history, generated over nearly seven years, was extracted and stored in a graph database (Neo4j). It was then queried to model employee communication before and after the company suffered a structural reorganization. Weighted and unweighted user-user communication networks were derived from the data and the model that best suited the problem under study was chosen. Centrality measures and communities were analyzed to evaluate the effect the company's internal reorganization had in the way employees work and communicate.

Results show that the recent organizational changes did have an impact in the way people communicate and that SNA is a valid and powerful tool to evaluate performance and success of such strategic decisions.

Keywords: Social Network Analysis, Organizational Network Analysis, GitHub, communication networks

Table of Contents

CHAPTER 1 INTRODUCTION 1

1.1 Motivation 1

1.2 Problem definition 1

1.3 Contribution 3

1.4 Organization 3

CHAPTER 2 STATE OF THE ART 4

2.1 An Overview of Social Network Analysis 4

2.1.1 Network Analysis Applications 4

2.1.2 Graph Theory 6

2.1.3 Analyzing Social Networks 11

2.1.4 Community Detection 20

2.2 Mining GitHub 21

CHAPTER 3 METHODOLOGY 24

3.1 Problem Description 24

3.2 Data description 25

3.3 Modelling GitHub "home" repository in Neo4j 25

3.4 Modelling user-user interaction 27

CHAPTER 4 RESULTS AND DISCUSSION 29

4.1 Data Collection and Pre-Processing 29

4.2 Computation of Network Measures 31

4.3 Statistical Analysis 31

4.4 Model Comparison and Selection 32

4.4.1 Network Measures 32

4.4.2 Community Detection 38

4.4.3 Concluding on the Chosen Model 39

4.5 ABCDE Unweighted Model Analysis 40

4.5.1 Network Measures 40

4.5.2 Comparing Business Clusters and Communities 44

4.6 Shiny App 47

CHAPTER 5 CONCLUSIONS 49

5.1 Closing remarks 49

5.2 Limitations and Future Work 49

BIBLIOGRAPHY 51

APPENDIX I 57

I. R Scripts 57

i. Querying Neo4j 57

ii. Creating Relationship dataframes 60

II. Network Measure Plots 65
i. Weighted Networks 65
ii. Unweighted Networks 67
III. Relevant network nodes 68
IV. Network plots 70

List of Figures

Figure 1: Internal restructuring of the organization from missions to team clusters. 2
Figure 2: A simple network composed of five vertices and edges. 4
Figure 3: Gene co-expression network. 5
Figure 4: Commonly used graph data structures. 7
Figure 5: Graph types. 8
Figure 6: Adjacency matrix representations. 8
Figure 7: Simple networks. 10
Figure 8: Graph with three distinct paths from A to B. 11
Figure 9: Set of hubs and the authorities they point to. 19
Figure 10: Overview of a GitHub issue structure. 25
Figure 11: Graph model of the GitHub "home" repository. 26
Figure 12: Relationships accounted for when modelling the interaction between users via GitHub. 27
Figure 13: Illustration of the eight user-user interaction sub-networks generated. 28
Figure 14: Sub-network obtained by querying Neo4j. 30
Figure 15: Sub-network obtained by querying Neo4j. 30
Figure 16: Node closeness distribution in weighted models. 34
Figure 17: Node local clustering coefficient in weighted models. 34
Figure 18: Distribution of edge weights in the weighted networks. 36
Figure 19: Node closeness distribution in unweighted relationship models. 37
Figure 20: Node degree distribution in unweighted models. 38
Figure 21: User-user communication networks created with the ABCDE unweighted model. 45
Figure 22: Networks highlighting the known games business cluster collaborators. 46
Figure 23: Networks highlighting the known player services business cluster collaborators. 46
Figure 24: Screenshot of the first app panel. 47
Figure 25: Screenshot of the second app panel. 48
Figure 26: Screenshot of the third app panel. 48
Figure 27: Node degree distribution in weighted models. 65
Figure 28: Node strength distribution in weighted models. 65
Figure 29: Node betweenness distribution in weighted models. 66
Figure 30: Shortest path length distribution in weighted models. 66
Figure 31 Shortest path length distribution in unweighted models. 67
Figure 32: Node betweenness distribution in unweighted models. 67
Figure 33: Node local clustering coefficient distribution in unweighted models. 68
Figure 34: Networks highlighting the known employee experience business cluster collaborators. 70
Figure 35: Networks highlighting the known marketing business cluster collaborators. 70
Figure 36: Networks highlighting collaborators for which the business cluster is not known. 71

List of Tables

Table 1: Length of all paths between nodes A and B, calculated by different distance metrics. 11
Table 2: Network node types and their count. 29
Table 3: Network edge types and their count. 29
Table 4: Indicators describing the weighted sub-networks. 33
Table 5: Ratio of annotated nodes in the weighted networks. 33
Table 6: Weight distribution percentiles in the weighted sub-networks. 36
Table 7: Indicators describing the unweighted sub-networks. 37
Table 8: Ratio of annotated nodes in the unweighted networks. 37
Table 9: Network modularity after applying each community detection algorithm. 39
Table 10: Network modularity and number of communities detected in weighted networks using the cluster fast greedy algorithm. 39
Table 11: Network modularity and number of communities detected in unweighted networks using the cluster spin-glass algorithm. 39
Table 12: Weight distribution percentiles in the ABCDE unweighted sub-networks. 40
Table 13: Nodes in 2017 ABCDE unweighted network with the five highest betweenness centrality scores. 41
Table 14: Nodes in 2018 ABCDE unweighted network with the five highest betweenness centrality scores. 41
Table 15: Nodes in 2017 ABCDE unweighted network with the five highest closeness centrality scores. 41
Table 16: Nodes in 2018 ABCDE unweighted network with the five highest closeness centrality scores. 41
Table 17: Nodes in 2017 ABCDE unweighted network with the five highest degree centrality scores. 42
Table 18: Nodes in 2018 ABCDE unweighted network with the five highest degree centrality scores. 42
Table 19: Nodes in 2017 ABCDE unweighted network with degree centrality of one. 43
Table 20: Nodes in 2018 ABCDE unweighted network with degree centrality of one. 43
Table 21: Nodes in 2017 ABCDE unweighted network with the five highest eigenvector centrality scores. 44
Table 22: Nodes in 2018 ABCDE unweighted network with the five highest eigenvector centrality scores. 44
Table 23: Nodes in 2017 ABCDE unweighted network with clustering coefficient of one. 68
Table 24: Nodes in 2018 ABCDE unweighted network with clustering coefficient of one. 69

Chapter 1 Introduction

In this chapter the problem under study, its contributions and motivations behind it are explained.

1.1 Motivation

In the knowledge economy, and in the tech sector in particular, companies operate in highly dynamic business environments where knowledge sharing and effective communication are critical to allow a fast response. Many tech startups adopt a flatter, simpler and less formal organizational structure, rather than the traditional highly hierarchical one. However, as companies become successful and grow these structures are often unsustainable. Social Network Analysis (SNA) can contribute to the conception of a new structure within a company, by pinpointing the right people for leadership roles, and it can be helpful in evaluating its implementation.

SNA uses graph theory to study social structures. It can be used by companies as a mean to study employee communication patterns and assess how they compare to the formal organizational structure (Zack, 2000). SNA produces visual and quantitative descriptions of organizational structures and contributes to the identification of key-persons, isolated actors and the presence of information flow bottlenecks in organizations. Using SNA tools, it is possible to depict who is at the center or at the periphery of the network, an information that is valuable not only for the management team but also for the employees themselves. Furthermore, it allows an evolutionary perspective of the network by providing snapshots over different time periods.

1.2 Problem definition

The goal of this work is to use SNA to study the organizational structure of a mid-size, fast-growing international online gambling company. It was proposed to me as a starting point for a three-month data science internship, after their business experienced some growing pains.

The institution in question, whose name will not be disclosed for privacy reasons, was initially organized in large cross-functional missions composed of people with diverse backgrounds and that may work on-site or remotely. As the company grew, the number of

missions surpassed thirty, and the whole structure became unsustainable. It was too slow at reacting to changes in market, gambling and privacy regulations. The success of missions became hard to track and the company was experiencing communication and collaboration problems. There were also leadership issues and an overall lack of common goals and strategy. Aiming at reducing the complexity of the organization and expedite the adaptation to changes, there was a shift from a flat to a highly networked organization. The idea was to shorten lines of communication, increase transparency and collaboration, enable knowledge sharing and make it easier to take the right decisions. They moved from missions to smaller teams, aggregated in six clusters (Figure 1). Clusters are fluid organizational structures, built around business areas that have a distinct concern and can be seen as autonomous units, each with a defined strategy and roadmap. They can consist of cross-functional teams, teams of a single profession, or both. Teams are dynamic structures, whilst the scope of each cluster is more permanent. Each team is managed by team lead and each cluster by a cluster lead. In this structure, communication lines between people are meant to reflect the true organizational chart.

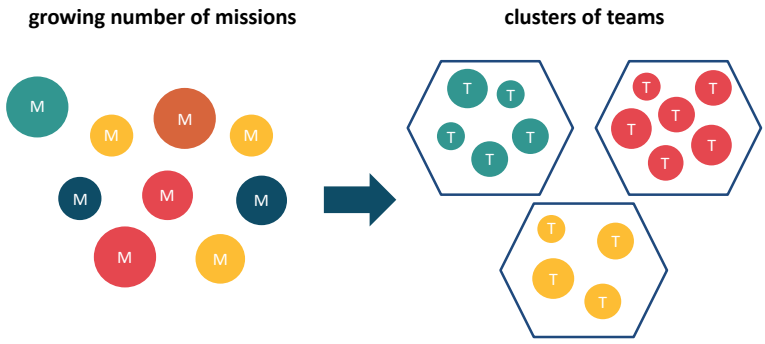


Figure 1: Internal restructuring of the organization from missions to team clusters. M: missions; T: teams.

The code repository service GitHub[1] is used within the organization as a vital tool for enabling cooperation. It serves as a platform for idea sharing, it is where problems are identified and solved. The company's GitHub "home" repository is utilized by all employees to map and keep track of issues, new ideas, proposed changes or changes already agreed upon. It is a place to foster discussions and internal collaboration.

In this work, an intra-organizational social network was built from data on the company's GitHub "home" repository, to answer following questions:

[1] https://github.com/, accessed on 26th December 2018

- How does the overall topology of the network look like?
- Who are the key-persons in the network?
- Are there any silos?
- Are there identifiable clusters of employees? How do they evolve over time?
- Are the recent organizational changes having an impact in how people communicate?

1.3 Contribution

The goal of this work is to develop a model that is able assess organizational performance by capturing the way employees communicate and collaborate regarding their work functions.

1.4 Organization

This Dissertation is organized in the following chapters:

Chapter 2 describes the state of the art and fundamental information about SNA, centrality measures and community detection. Relevant publications using SNA and GitHub data are also reviewed.

Chapter 3 contains a description of the problem and data under study. It includes an overview of the methodology used and an outline of GitHub and Neo4j.

Chapter 4 presents methodological details about data collection, pre-processing, network model selection and analysis. Results are shown and discussed.

Chapter 5 depicts the main conclusions and limitations of this work, as well as ideas on how to improve in future work.

Chapter 2 State of the Art

This chapter briefly overviews the myriad of network analysis applications in different fields, such as computer engineering or biological and social sciences. Then graph theory is introduced, explaining what graphs represent, the different graph types and how to analyze them using node and network measures. Traditional community detection algorithms are also described. Finally, relevant publications where GitHub data is mined are referred.

2.1 An Overview of Social Network Analysis

2.1.1 Network Analysis Applications

Network analysis has its basis in network theory, a branch of graph theory, whose origins have been traced to the 18th century, when the Swiss mathematician and physicist Leonhard Euler used graph concepts to solve "The Königsberg Bridge Problem" (Euler, 1736). In network theory, objects (vertices) and the connections they share (edges) are represented as graphs (Figure 2), and their properties analyzed using specific techniques. It is possible to identify communities of vertices more similar to each other than to other vertices (Fortunato & Hric, 2016) and study how the network structure evolves over time (Doreian, 2002). Network theory has contributed to the development of fields spanning from natural to social sciences, since many real-world situations have an underlying network structure. A graph approach can be followed to simplify the representation of many problems whose structure can be thought of as interactions between entities.

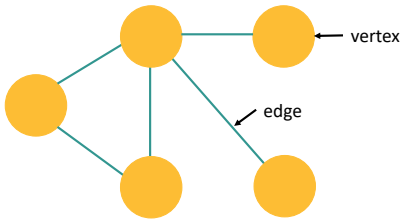


Figure 2: A simple network composed of five vertices (circles) and edges (lines).

Biologists study a variety of biological networks: metabolic networks (Kanehisa, Furumichi, Tanabe, Sato, & Morishima, 2017), where metabolites in the same pathway are connected by enzymatic reactions; complex gene regulatory networks (Peter & Davidson, 2015), depicting known interactions between regulatory and regulated genes; protein-protein

interaction networks (Chatr-aryamontri et al., 2017; Ghadie, Coulombe-Huntington, & Xia, 2018); gene co-expression networks (Saelens, Cannoodt, & Saeys, 2018) (Figure 3), that identify genes sharing identical expression patterns in different tissues or experimental conditions; and ecological networks (Woodward & Benstead, 2010), that represent interactions between species in ecosystems.

In epidemiology, the transmission mechanisms of contagious diseases, such as HIV (Campbell et al., 2017) or the Ebola virus (Mate et al., 2015), are studied using Transmission Network Analysis (TNA). An interesting study was also done on the spread of obesity through a large network of people over the course of 35 years (Christakis & Fowler, 2007), having reached relevant conclusions for public health interventions in obesity management.

In computer engineering, computer networks and the Word Wide Web (WWW) can be thought of as graphs. In the latter, vertices represent pages and edges the hyperlinks pointing from one page to another (Huberman, 2001). Very recently, network analysis gave a crucial aid in understanding the intricate ties in the Panama and Paradise Papers financial scandals, being helpful at the same time for journalists, criminologists and economists.

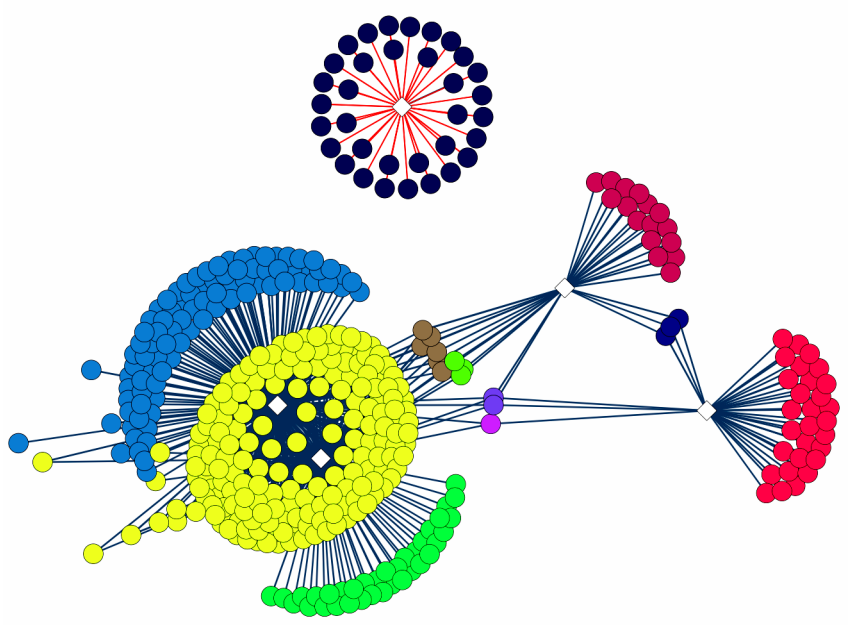


Figure 3: Gene co-expression network. Circular vertices represent genes that share a pattern of expression across samples with bait-genes, which are represented as white diamond-shaped vertices. Vertices have the same color if they co-express with the same baits. Edge color represents a positive (blue) or negative (red) expression correlation with baits.

In social sciences, SNA studies social structures formed by relations between individuals or collectives. It helps determining meaningful patterns, depict central actors and describe

their influence on the network. In the 1970's and 80's, surveys, field work and interviews on how individuals interact with their colleagues were used infer social networks (Burt, 1984; Wellman, 1979). More recently, e-mail exchange data has been used to create an e-mail network and make inferences on the propagation rate of viruses via e-mail (Ebel, Mielsch, & Bornholdt, 2002). Taking advantage of the boom in internet accessibility and social network platforms, it is now possible to generate large-scale networks representing social connections between LinkedIn[2] (Russell, 2013), Facebook[3] (Lewis, Kaufman, Gonzalez, Wimmer, & Christakis, 2008) or Twitter[4] (Ediger et al., 2010) users.

Network analytics tools are used in 21st-century corporations (Bryan & Joyce, 2005) to map and exploit the informal social structures, though which information flows better than by the hierarchical chain. Mapping collaborations, with associated costs and benefits, between employees or between them and costumers, can contribute to the development of strategies that increase productivity and revenue (Bryan & Joyce, 2005).

2.1.2 Graph Theory

2.1.2.1 Graph Representation

A social network can be represented a graph, G, that according to (Bondy & Murty, 1976) is formally defined as pair $G = (V, E, \psi_g)$, where V is a nonempty set of vertices, E a set of edges disjoint from V, and ψ_g an incidence function mapping each edge to a pair of vertices. Graph nomenclature varies across disciplines with vertices being known also as nodes, actors or agents; and edges named as links, connections, ties or bonds. The terms network and graph can be considered synonymous.

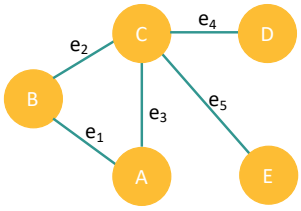
Given two vertices *u* and *v* connected by and edge *e*, e is represented by the pair (*u*, *v*) and *u* and *v* are said to be **adjacent** or neighbors. Both nodes *u* and *v* are **incident** on *e*, and vice-versa, *u* and *v* are also endpoints of *e;* u is the tail of *e* and *v* is the head.

List and matrices are the two types of data structures used for graph storage and commonly contain information on node adjacency or incidence (Figure 4). Adjacency lists keep information on all neighbors of each node in the network. In adjacency matrices, all nodes are represented as rows and columns and edge information fills in the matrix. These matrices are symmetric and their dimension is given by the number of nodes in the network.

---

[2] https://www.linkedin.com/, accessed on 26th December 2018
[3] https://www.facebook.com/, accessed on 26th December 2018
[4] https://twitter.com/, accessed on 26th December 2018

Incidence lists, store information on edges incident to each node in the network. In incidence matrices, rows represent nodes and columns represent edges, thus they are not necessarily symmetric. If an edge connects two nodes, information on that link is stored in the respective edge column and node rows. Adjacency lists are very compact space efficient and are particularly suited for sparse graphs. Matrices are advisable for representing dense graphs.



adjacency list:

A: B, C
B: A, C
C: B, D
D: C
E: C

adjacency matrix:

	A	B	C	D	E
A	0	1	1	0	0
B	1	0	1	0	0
C	1	1	0	1	1
D	0	0	1	0	0
E	0	0	1	0	0

incidence list:

A: {A, B}, {A, C}
B: {B, A}, {B, C}
C: {C, A}, {C, B}, {C, D}, {C, E}
D:{D, C}
E: {E, C}

incidence matrix:

	e_1	e_2	e_3	e_4	e_5
A	1	0	1	0	0
B	1	1	0	0	0
C	0	1	1	1	1
D	0	0	0	1	0
E	0	0	0	0	1

Figure 4: Commonly used graph data structures. Adjacency (left) and incidence (right) lists and matrices representing an unweighted and undirected graph (middle).

2.1.2.2 Graph Types

The undirected and unweighted graphs we've seen so far, where nodes are connected by edges representing bidirectional relationships that are even across the network, are the simplest type of graphs. Networks of higher complexity can be created where various information is stored in both edges and nodes (Figure 5). Regarding the edges, networks can be **unweighted** (Figure 5 A, C, D) if they all have the same representation across the graph, or **weighted** (Figure 5 B) if they have different weights. Weights can represent a variety of things, such as the intensity or size of a measure, they can for instance indicate the number of exchanged e-mails between people, or the length of a path between cities. In weighted networks, edges vary in their thickness and/or contain a number representing their weight. Graphs can be **directed** (Figure 5 A), also called digraphs, where or relationships may not be reciprocal, or **undirected** (Figure 5 B, C, D) if they are always mutual between nodes. In directed graphs, edges are defined by an ordered pair of nodes and end in an arrow pointing the relationship direction. More than one type of relationship can be represented in the same graph (Figure 5 D). Nodes can represent different objects and share different relationships (Figure 5 C). There are **bipartite** graphs (Figure 5 E), containing two independent sets of

nodes, where edges establish connections only between different node types (Newman, Strogatz, & Watts, 2001). Nodes of the same type are connected if they share a connection with the same node. The gene co-expression network in Figure 3 is an example of a bipartite graph.

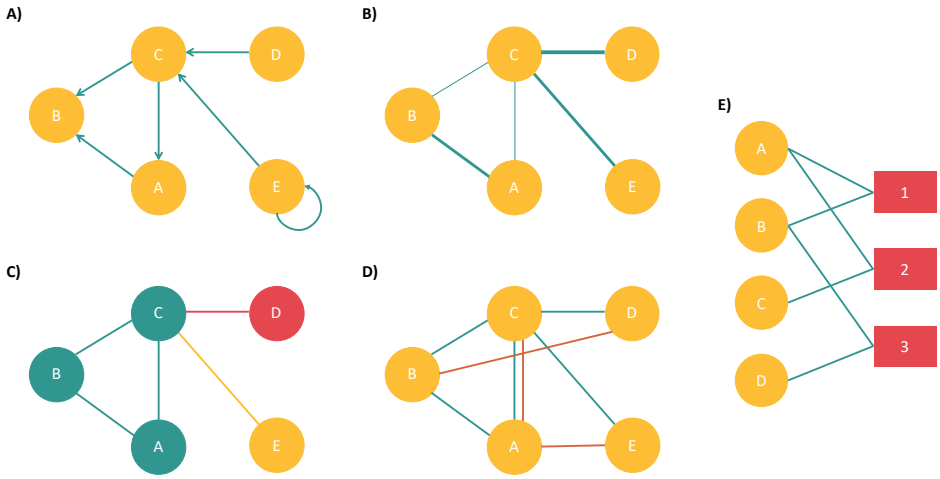
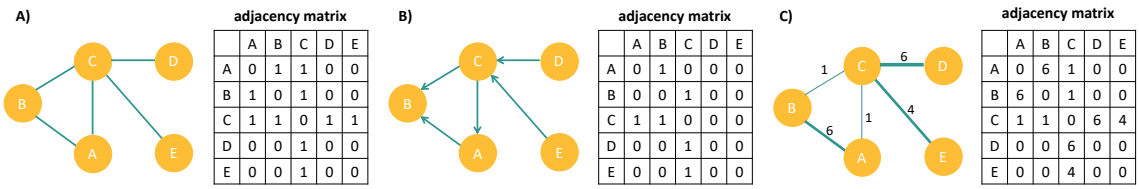


Figure 5: Graph types. A) unweighted, directed graph with arrows pointing edge direction; B) weighted, undirected graph with edge thickness representing weight; C) unweighted, undirected graph containing nodes of different types, as evidenced by their color; D) unweighted, undirected graph containing more than one relationship type, as indicated by their color; E) bipartite graph with two distinct node types (circle and rectangle).

Matrices and lists can contain information representing edge direction and weight, and are able to describe the mentioned types of graph structures (Figure 6). For undirected graphs (Figure 6 A, C), there is symmetry along the adjacency matrix diagonal, since to a tie from node A to B there is a corresponding tie from node B to A. This does not happen for directed graphs (Figure 6 B), where unidirectionality exists. If a tie goes in the direction from A to B, but not from B to A, information for that tie is indicated in the matrix only in the correct direction. For weighted graphs (Figure 6 C), if an edge exists its weight can be indicated at the corresponding row and column intersection. For unweighted (Figure 6 A, B) graphs, the edge information is Boolean, taking the value of 1 if the edge exists and 0 if it does not.



	A	B	C	D	E
A	0	1	1	0	0
B	1	0	1	0	0
C	1	1	0	1	1
D	0	0	1	0	0
E	0	0	1	0	0

	A	B	C	D	E
A	0	1	0	0	0
B	0	0	1	0	0
C	1	1	0	0	0
D	0	0	1	0	0
E	0	0	1	0	0

	A	B	C	D	E
A	0	6	1	0	0
B	6	0	1	0	0
C	1	1	0	6	4
D	0	0	6	0	0
E	0	0	4	0	0

Figure 6: Adjacency matrix representations of A) an undirected unweighted graph; B) a directed unweighted graph and C) an undirected weighted graph.

Networks can be **sociocentric** (whole networks), built from a group perspective, or **egocentric** (personal networks) if they focus on relationships of a single individual, called ego, with other nodes, called alters, and between the alters themselves (Tabassum & Gama, 2016). A person's (ego) Facebook friendship network is an example of an ego network, where each node (alter) in the network is friends with the ego and edges connecting alters show they are also friends between them.

2.1.2.3 General Graph Concepts

Simple graphs are those without loops and whose edges all join different pairs of vertices (Bondy & Murty, 1976). **Loops** designate edges whose endpoints are the same, as an example, node E in Figure 5 A has a loop, and **links** designate edges whose endpoints are distinct (Bondy & Murty, 1976). A graph is **complete** if every two distinct vertices are adjacent. A graph H is a **subgraph** of G if all vertices and edges of H are subsets of the vertices and edges of G, in which case G is a supergraph of H. A complete subgraph is called a **clique**.

The **order** of a graph, *n*, is given by the number of vertices it comprises, and its **size**, *m*, is equal to the count of edges. The maximum number of links a network of *n* nodes can have is given by: $m_{max} = n(n-1)$ for directed, or $m_{max} = \frac{n(n-1)}{2}$ for undirected networks. For **complete** networks, the total number of links is equal to m_{max}.

A **walk**, W, of length *k*, in a graph is a sequence of adjacent vertices, *v*, $W = v_0 v_1 v_2 \dots v_k$ that starts in an origin node (v_o) and ends, proceeds through internal nodes, and ends in a terminus node (v_k) (Figure 7 A). A closed walk has its origin and terminus are at the same node. If all nodes in a walk are different it is called a **path**, and a closed path constitutes a cycle.

A **triplet** is a set of three nodes connected by either two edges (open triplet) – e.g. {E, C, D} in Figure 6 A – or three edges (closed triplet) – e.g. {A, B, C} in Figure 6 A. A **triangle** is formed by three closed triplets, each centered on one node – e.g. the three closed triplets {A, B, C}, {B, C, A} and {C, A, B} in Figure 6 A form a triangle.

A graph is **connected** if any two nodes are connected by a path (Figure 7 A), and **disconnected** (Figure 7 B) if they are not. Connected graphs contain a single maximal connected subgraph (**component**) and disconnected graphs can be separated into multiple connected components. As an example, in Figure 7 B the graph can be split into three connected components: {A, B, C}, {E, F} and {G, H, I, J, K}. The distance between nodes

of different components is said to be infinite. An acyclic connected graph is designated a **tree**, and is exemplified in Figure 7 B by the connected component {G, H, I, J, K}. If the removal of an edge in a connected graph results in its disconnection, that edge is called a **bridge** (Figure 7 C).

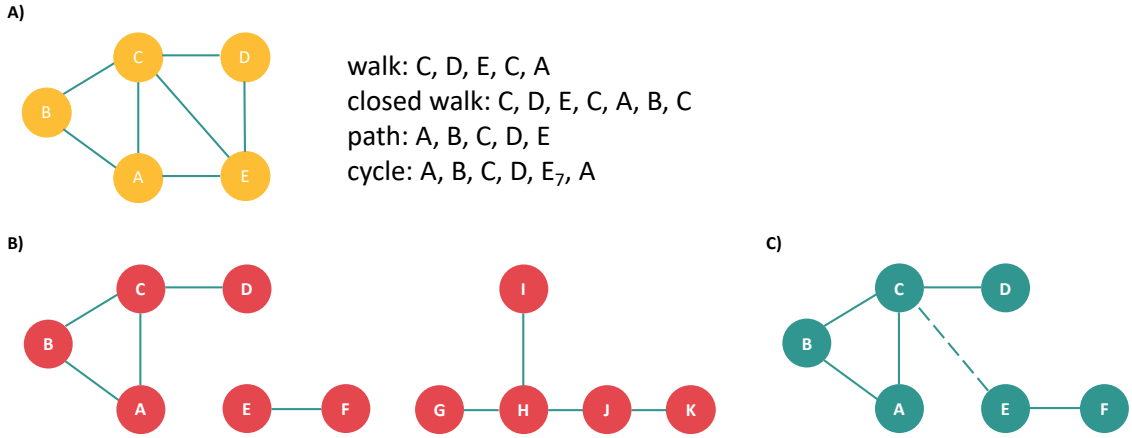


Figure 7: Simple networks with A) connected graph structure, where a walk, closed walk, path and cycle are exemplified; B) disconnected graph structure. C) exemplifies a connected graph with a bridge (dashed line), whose removal would result in two separate connected components.

The **shortest path**, geodesic distance, or simply distance, d, between two nodes *i* and *j* through internal nodes *h*, in an unweighted network is the length of the minimal path separating them (eq. (2.1):

(2.1)

$$d = \min (x_{ih} + \cdots + x_{hj})$$

where each x is the distance between neighbor nodes and always equal to 1. There might be more than one shortest path between the same two nodes, as exemplified in the network in Figure 7 A, where two shortest paths separate nodes A and D: {A, C, D} and {A, E, D}.

For weighted networks, if ties represent a cost (e.g. distance), the shortest path, d^w, between *i* and *j* is the one with the minimum sum of edge weights, w (Dijkstra, 1959):

(2.2)

$$d^w = \min (w_{ih} + \cdots + w_{hj})$$

However, if ties represent an ease of communication (e.g. degree of interaction between people), eq. (2.3 can be modified so that the inverse of the weights is used to calculate the shortest path (Brandes, 2001; Newman, 2001), giving stronger ties a lower cost that weaker ties (2.3):

(2.3)

$$d^w = \min \left(\frac{1}{w_{ih}} + \cdots + \frac{1}{w_{hj}}\right)$$

This distance gives a higher importance to paths with larger number of higher weight connections than to weak direct links. Furthermore, we might wish to consider a tradeoff between the number of ties in a path and their weigh (Opsahl, Agneessens, & Skvoretz, 2010). This can be done by introducing in a positive parameter, α, to be tuned by the user (2.4):

(2.4)

$$d^{w\alpha} = \min\left(\frac{1}{(w_{ih})^{\alpha}} + \cdots + \frac{1}{(w_{hj})^{\alpha}}\right)$$

Using this equation, if $\alpha = 0$ the distance measure is the same as for unweighted networks (2.1) and if $\alpha = 1$ the distance is the same as calculated by (2.3). For other values of α, and for paths of the same distance as calculated by (2.3, a higher importance is given to a lower number of ties ($\alpha < 1$), or to higher tie weights ($\alpha > 1$). This is exemplified in Figure 8 and Table 1, where all paths between nodes A and B: {A, B}, {A, C, B} and {A, D, E, B}, have the same length, as defined by (2.3). When using (2.4) with $\alpha = 0.5$, the shortest path is the one with fewest links; setting $\alpha = 1.5$ assigns the shortest path to the one containing links with higher weight.

From node distance derives another concept which is node **eccentricity**, *i.e.* the maximum of all distances (longest shortest path) from that node to any other node in the network.

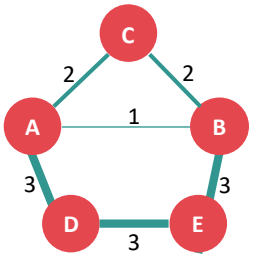


Figure 8: Graph with three distinct paths from A to B. From (Opsahl et al., 2010) .

Table 1: Length of all paths between nodes A and B in Figure 8, calculated by different distance metrics. d:(2.1) considering an unweighted network; d^w: (2.3); $d^{w\alpha}$: (2.4) with different α parameter values. Shortest paths are highlighted in yellow.

	d	d^w	$d^{w\alpha}$			
			$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 1.5$
{A, B}	1	1	1	1	1	1
{A, C, B}	2	1	2	1.4	1	0.7
{A, D, E, B}	3	1	3	1.8	1	0.5

2.1.3 Analyzing Social Networks

2.1.3.1 Node Measures

Several node metrics exist to aid in the characterization of the relative importance of a node in a network and understand its role and behavior in that context. Centrality measures,

such as degree, closeness, betweenness and eigenvector centrality, ascertain the position of a node in a graph. The local clustering coefficient measures cohesion in a node's neighborhood.

Degree Centrality

The degree, k, of a focal node, i, is equal to the sum of incident edges connecting it to other nodes, which is the same to say it is equal to the number of neighbors it has (Freeman, 1978). It is calculated for each node and measures their individual involvement in a graph, their importance or influence.

In unweighted undirected graphs it is calculated as follows:

(2.5)

$$k_i = \sum_{j}^{N} x_{ij}$$

where N represents all nodes in the network, and x the adjacency matrix where the cell (i, j) takes a value of 1 if nodes i and j are connected and 0 if they are not. A node has the maximum possible degree, $k_i = N - 1$, if it is connected to all other nodes in the network.

When extending this measure to unweighted directed graphs, each node has an in-degree, k_i^+, i.e. the number of edges pointing to it, and an out-degree, k_i^-, i.e. is the number of edges pointing from it to other nodes. In-degree measures the support a node has in a network, and out-degree its influence.

(2.6)

$$k_i^+ = \sum_{j}^{N} x_{ji}$$

(2.7)

$$k_i^- = \sum_{j}^{N} x_{ij}$$

In unweighted undirected graphs, the degree is calculated as the sum of weights, w, of edges leaving a node (Barrat, Barthélemy, Pastor-Satorras, & Vespignani, 2004; Newman, 2004; Opsahl, Colizza, Panzarasa, & Ramasco, 2008), and is designated as the node's strength, k_i^w:

(2.8)

$$k_i^w = \sum_{j}^{N} w_{ij}$$

where w is the weighed adjacency matrix and the cell (i, j) takes the value of the edge weight (≥ 1) connecting the nodes, or 0 if they are not neighbors. However, this definition of

strength does not take into account the number of edges leaving a node, only their weight. Following the previously explained reasoning for distance calculation in weighted networks, Opsahl *et al.* (Opsahl et al., 2010) propose a way to combine degree, k_i, and strength, k_i^w, in the same measure. By including a positive tuning parameter, α, it is possible to determine the relative importance of edge number and weight:

(2.9)

$$k_i^{w\alpha} = k_i \times \left(\frac{k_i^w}{k_i}\right)^{\alpha} = k_i^{(1-\alpha)} \times (k_i^w)^{\alpha}$$

The parameter α is set by the user according to the following: if $\alpha = 0$ the measure is the same as for unweighted networks (2.5) and if $\alpha = 1$ it is the same as calculated by (2.8. For other values of α, and for nodes of the same strength as calculated by (2.8, the measure takes higher values if the weight is distributed by more edges ($\alpha < 1$), or less edges ($\alpha > 1$). Furthermore, the authors also propose an extension of $k_i^{w\alpha}$ to weighted directed graphs.

Betweenness Centrality

Betweenness, b, measures the control a node, i, has over the network's information flow. It quantifies the number of times a node lies in the paths of a graph (Freeman, 1978), and is calculated for paths between all node pairs in the network:

(2.10)

$$b_i = \sum_{j,k \in V(G)} \frac{g_{jk}(i)}{g_{jk}}$$

where g_{jk} is the number of geodesic paths between nodes j and k, and $g_{jk}(i)$ the number of those paths that go through i. The higher the betweenness of a node the more central role it occupies in the network as information preferentially runs by it on the way to other nodes.

Opsahl *et al.* (Opsahl et al., 2010) propose the use of (2.4 to calculate the distances in undirected weighed networks, resulting in the following betweenness centrality measure:

(2.11)

$$b_i^{w\alpha} = \sum_{j,k \in V(G)} \frac{g_{jk}^{w\alpha}(i)}{g_{jk}^{w\alpha}}$$

Closeness Centrality

Closeness, Cl, measures the extent to which a node, i, is near all other nodes in the network, thus capturing its ability to access information. It is the inverse of the sum of all distances between i and all other nodes in the network. Smaller closeness values represent larger

distances from a node to all other nodes, thus less chances of receiving or effectively transmitting information.

(2.12)

$$Cl_i = \left[\sum_j^N d(i,j)\right]^{-1}$$

For undirected unweighted networks, when using (2.4) to calculate distances as proposed by Opsahl *et al.* (Opsahl et al., 2010), closeness centrality is calculated as follows:

(2.13)

$$Cl_i^{w\alpha} = \left[\sum_j^N d^{w\alpha}(i,j)\right]^{-1}$$

Eigenvector Centrality

Eigenvector centrality, C, measures how well a node, i, is connected to other well-connected nodes in the network. Thus, it is proportional to the sum of the neighbor's centralities. Eigenvector centrality is an extension of node degree in the sense that quantifies node importance considering not only the quantity but also the quality of a node's neighbors. In this metric, the existence of links to high-scoring nodes recursively increases the score of the node in question; a computation made using the eigenvector centrality formula:

(2.14)

$$C_i \frac{1}{\lambda} \sum_j^N x_{ij} C_j$$

where x_{ij} is the adjacency matrix, taking values 1 or 0 if nodes are connected or not, respectively. The largest adjacency matrix eigenvalue is represented by λ.

Transitivity and Local Clustering Coefficient

Transitivity is a property that describes the likelihood of node's neighbors to also connected between themselves. It is normally high in social networks, where an individual's friends are likely to also be each other's friends. The clustering coefficient measures transitivity at the node level (local clustering coefficient) and at the network level (global clustering coefficient).

The local clustering coefficient, *CC*, of a node, i, is the ratio of number of edges that connect i's neighbors over the total number of possible edges between them (Watts & Strogatz, 1998). For an unweighted undirected graph:

(2.15)

$$CC_i = \frac{\left|\{e_{jk}: v_j v_k \in N_i, e_{jk} \in E\}\right|}{\frac{k_i(k_i-1)}{2}}$$

where k_i is the number of neighbors in the neighborhood, N_i, of i and $\left|e_{jk}\right|$ is the number of edges between nodes in that neighborhood, which vertices j and k are a part of. E is the set of vertices in the graph.

2.1.3.2 Edge measures

Most research has been done on node centrality measures. However, it is equally important to characterize the importance of edges. Edge betweenness was proposed by Girvan and Newman (Girvan & Newman, 2002) to identify edges separating communities. It quantifies the number of shortest paths between node pairs that go through a particular edge. The idea is that edges standing between communities are part of the shortest paths separating members of different communities, and thus have higher edge betweenness than edges connecting nodes of the same community. It should be noted that this metric disregards paths of length slightly bigger than the shortest one, that might be equally relevant for information flow.

Other edge metrics exist, such as the spanning edge betweenness (Teixeira, Monteiro, Carriço, Ramirez, & Francisco, 2013), or current flow centrality (Brandes & Fleischer, 2005) and Kirchhoff edge centrality (Li & Zhang, 2018), the last two based on the resistance distance in electrical networks (Klein & Randić, 1993) and the latter applicable to undirected weighted networks.

2.1.3.3 Network Measures

Network measures are calculated on the network as a whole. They are global measures that tell us something about the overall graph structure.

Diameter and Radius

The diameter and radius of a network are the maximum and minimum of all node eccentricities, respectively. The diameter of a graph tells us how far apart are the two most distant nodes, always considering the shortest path.

Nodes whose eccentricity equals the network diameter are designated peripheral vertices and constitute the network periphery. Those with an eccentricity equal to the radius are central vertices and compose its center.

Average Geodesic Distance

The average geodesic distance, *l*, of a network tells us how closely connected nodes are in the network. In unweighted graphs, it is the average number of edges in the shortest paths between all possible node pairs in the network:

(2.16)

$$l = \frac{1}{\frac{1}{2}n(n-1)} \sum_{i \neq j} d(i,j)$$

considering an undirected graph, where $\frac{1}{2}n(n-1)$ is the maximum possible number of edges, and $d(i,j)$ is the distance between nodes *i* and *j*.

Average Degree and Degree Distribution

The network's average degree, $\bar{k}$, tells us, on average, how many nodes any node in the network is connected to. It gives a picture of the overall network connectivity. Formally, for n nodes:

(2.17)

$$\bar{k} = \frac{1}{n} \sum_{i=0}^{n} k_i$$

The degree distribution of a graph, p_k, is the fraction of nodes that have degree *k*, n_k:

(2.18)

$$p_k = \frac{n_k}{n}$$

For real-world networks, including social networks, it has been described that only a minority of nodes have a degree much higher than the average network degree, resulting in a highly rightly-skewed p_k histogram (Barabasi & Albert, 1999).

Density

The density, ρ, is another way of looking at the global connectivity of a graph, *G*. It is the proportion of edges a network has, $m(G)$, relative to the maximum possible edges it could have if it was a complete network, $m_{max}(G)$:

(2.19)

$$\rho(G) = \frac{m(G)}{m_{max}(G)}$$

It can take values between 0, if no nodes are connected, and 1 if the graph is fully connected. Graphs with higher ρ values are said to be denser than those with lower ρ values, which are sparser.

Reciprocity

Reciprocity, *r*, is used in directed networks to quantify the propensity of nodes to be mutually connected. It is simply the ratio of ties in the network that are bidirectional:

(2.20)

$$\mathrm{r} = \frac{\#\ \text{bidirectional edges}}{\#\ \text{total edges}}$$

A directed network where all edges are bidirectional has an $r = 1$; one where all edges are unidirectional has an $r = 0$.

Global Clustering Coefficient

The graph's global clustering coefficient (Watts & Strogatz, 1998), *CC*, estimates its overall level of clustering. It is the average of all local clustering coefficients, CC_n, in the network over all network vertices, *n*:

(2.21)

$$\mathrm{CC} = \frac{1}{\mathrm{n}}\sum_{\mathrm{n}} \mathrm{CC_n}$$

Another way to measure global clustering coefficient is using the ratio of closed triplets which, as seen previously, is the same as 3 triangles, over the total number of triplets in an undirected network (Luce & Perry, 1949):

(2.22)

$$CC = \frac{3 \times \#triangles}{\#triplets} = \frac{\sum \tau_\Delta}{\sum \tau}$$

where $\sum \tau$ is the total number of triplets (open and closed) in the network, and $\sum \tau_\Delta$ is the subset of those triplets that are closed. Opsah and Panzaraza (Opsahl & Panzarasa, 2009) generalize this definition for weighted networks by taking the weight, w, of edges into account:

(2.23)

$$CC_w = \frac{total\ weight\ of\ closed\ triplets}{total\ weight\ of\ triplets} = \frac{\sum_{\tau_\Delta} w}{\sum_{\tau} w}$$

2.1.3.4 Link Analysis

Link analysis explores the relationships between all entities in a graph to find and rank the most relevant node(s) in a network. It has been employed in several fields, from fraud detection and terrorist network analysis, to internet search engine optimization. Two renowned algorithms in link analysis are PageRank and HITS (Hyperlink-Induced Topic Search), both developed to rank the importance of webpages despite their application not being restricted to the WWW.

PageRank

The PageRank algorithm is based on eigenvector centrality and was the first known algorithm developed by Google™ for its web page automated search engine (Page, Brin, Motwani, & Winograd, 1998), with the purpose of selecting the best pages to appear at the top of a search list when googling any topic. It ranks page importance using the web digraph, where nodes are pages and directed edges are the hyperlinks pointing from one page to the other. A node, u, can have a set of back-links, B_u, pointing to it from other pages, and a set of forward-links, F_u, pointing from it to other pages. $N_u = |F_u|$ is the number of forward-links from u, and the parameter d is a damping factor ($d < 1$) (Brin & Page, 1998). The PageRank, $R(u)$, of node u, is defined by:

(2.24)

$$\mathrm{R(u)} = 1 - \mathrm{d} + \mathrm{d} \sum_{\mathrm{v} \in \mathrm{B_u}} \frac{\mathrm{R(v)}}{\mathrm{N_v}}$$

In very broad terms, the algorithm recursively computes and updates $R(u)$ for k iterations until conversion, starting with the assignment of an initial rank to all nodes equally. Thus, the PageRank of a website considers not only the number but also the rank of pages pointing to it. This way, a page has a higher rank if other pages that hyperlink it are highly ranked themselves.

HITS: Hubs and Authorities

Although not specific to WWW application, the HITS algorithm was developed to find the "authoritative", more relevant, web pages when querying any broad search topic (Kleinberg, 1999). Considering the WWW digraph described in the previous section, for a given query, HITS's objective is to select a focused subgraph of the WWW, whose node set, S_σ, must obey the following criteria: be of relatively small number, be rich in relevant pages and contain most of the strongest "authoritative" pages on that subject. To do so, it starts with the first t (~200) results outputted from any search engine. This small starting set of pages, designated as root set, R_σ, contains the highest-ranked pages for the search in question, as calculated by the search engine's algorithm. In principle, R_σ already satisfies the first two criteria for the base set, S_σ. However, it still needs to be enriched in "authoritative" nodes, which are expected to be hyperlinked by at least one page in R_σ. Thus, pages referenced by at least one node in R_σ, as well as other pages, external to R_σ, that point to any page in it up to a fixed limit number, d, of external references per page, are used to

expand R_σ. It is expected that most authorities in the queried subject now belong to R_σ, which is then designated S_σ and the subgraph induced by it called $G[S_\sigma]$.

All links pointing between webpages of the same domain are removed from $G[S_\sigma]$, as they are irrelevant, resulting in a subgraph G_σ. Authors suggest the application of other heuristics at this point may further reduce the number of unwanted links or nodes.

The link structure of G_σ is now the ground for detection of the relevant authorities and hubs. To identify authorities, HITS assumes they not only have a high in-degree but also that different authorities will share many of the same neighbors pointing to them. These neighbor nodes that point to many different authorities are called hubs.

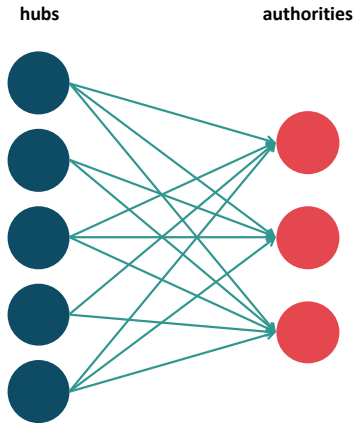


Figure 9: Set of hubs and the authorities they point to.

To untangle the mutual reinforcement relationship between good hubs, that reference good authorities, and good authorities, that are referenced by good hubs, HITS employs a recursive algorithm to find these nodes. In general terms, the algorithm starts with the assignment of an equal non-negative authority weight, $x^{\langle p \rangle}$, and hub weight, $y^{\langle p \rangle}$, to each node in the network. Being the square sum of all node's authority weights equal to one, and the same for hub weights.

If a node, p, is pointed to by many pages with large hub weights, its authority weight is increased, according to (2.25. On the other hand, if p points to many pages with large authority weights, its hub weight is increased according to (2.26. Then, for k iterations, all node's $x^{\langle p \rangle}$ and $y^{\langle p \rangle}$ are alternately updated until conversion. The pages with larger $x^{\langle p \rangle}$ or $y^{\langle p \rangle}$ are considered the best authorities and hubs, respectively.

(2.25)

$$x^{\langle p \rangle} \leftarrow \sum_{q:(p,q)\epsilon G_\sigma} y^{\langle q \rangle}$$

(2.26)

$$y^{\langle p \rangle} \leftarrow \sum_{q:(p,q)\epsilon G_\sigma} y^{\langle q \rangle}$$

2.1.4 Community Detection

Real-world networks have high clustering coefficients and an underlying community structure. Social networks are a particularly intuitive example of this, as people tend to form groups of friends, of people with shared interests, of co-workers, etc. Communities, also called clusters or modules, contain groups of nodes that are densely connected between them – by intracommunity edges –, but loosely connected to other groups – by intercommunity edges (Fortunato & Castellano, 2009). They are dense subgraphs that match functional groups of some sort. Moreover, nodes of the same community share more similarities between them than with nodes belonging to other communities in the network. Thus, identifying the community structure of a network may provide invaluable information to characterize nodes of a particular cluster.

Several community detection algorithms have been described, two of which are detailed in the following sections.

2.1.4.1 Hierarchical Clustering

Hierarchical clustering is a traditional technique for network community detection that has the advantage of not requiring any prior assumptions regarding cluster composition. The algorithm starts with the choice of a similarity metric between nodes (e.g. Euclidean distance), which is then computed for all node pairs, irrespective of whether they are connected or not, and stored in the form of a square similarity matrix. Next, the nodes are grouped via an agglomerative or a divisive approach, using one of several possible cluster similarity measures.

Agglomerative methods start with a number of clusters equal to the number of nodes in the network. Afterwards, the two most similar clusters are iteratively grouped into clusters of larger dimensions, until all nodes are joined in the same cluster. On the other hand, divisive procedures start with a cluster composed of all nodes in a graph, which are then successively divided into smaller ones. There are distinct ways to define cluster similarity. One approach is to use the minimal distance separating any two nodes of different clusters (single linkage), other is to use the maximal distance (complete linkage), another yet is to use the average distance (average linkage). The output of hierarchical clustering is a dendrogram, that visually represents the cluster hierarchy and the distance separating them.

2.1.4.2 Girvan-Newman Algorithm

The Girvan-Newman algorithm (Girvan & Newman, 2002) is a divisive hierarchical clustering algorithm that uses edge betweenness to form communities. Having the whole network as a starting point, edges with the highest betweenness, those standing between different communities, are progressively removed from the network until each node in the network is a cluster. The general form of the algorithm is the following: (1) determine edge betweenness for all edges in the network; (2) remove from the network the edge with highest betweenness score. If there is a tie, remove one of them at random; (3) recompute betweenness for all edges left; (4) iterate from step 2 until no edges are left.

To choose the best stopping point of the algorithm, so as not to end with as many clusters as the number of nodes, it is possible to use a modularity score to select the best number of divisions (Newman & Girvan, 2004). Modularity, Q, measures how good a network division is based on the number of edges in the original network that connect nodes belonging to different communities:

(2.27)

$$Q = \sum_{i} e_{ii} - \sum_{ijk} e_{ij} e_{ki}$$

where e is the square matrix, $g \times g$, whose dimensions are the number of communities, g, and whose values are the number of edges in the original network connecting nodes of those communities. The best cluster number is the one that maximizes modularity score.

2.2 Mining GitHub

GitHub is the most widely used open source collaborative coding platform. It uses git version control system and hosts code repositories to which different users contribute after their work is reviewed by the repository owner(s). As of November 2018, GitHub hosts 100 million public and private code repositories and is utilized by 2.1 million businesses and organizations worldwide for software development and project/team management. GitHub also incorporates several social features in its design. For instance, developers share their skills and accomplishments via a public profile that describes their collaboration activities. Users can follow other users and projects; they can assign a star to their favorite repositories and see which topics are trending at the moment. Developers have access to a feed with updates on the public activity of users and repositories they are interested in; they can discuss within repositories and use a community forum for idea sharing.

The widespread adoption of collaborative coding, coupled with the fact that GitHub openly shares data on its usage, has led to an interest in understanding how this teamwork takes place. Repositories and main features in GitHub (Kalliamvakou et al., 2014) have been characterized, and research done on language complexity (Kavaler, Sirovica, Hellendoorn, Aranovich, & Filkov, 2017) and sentiment analysis in issue text (Jurado & Rodriguez, 2015).

A fair amount of studies was carried out to understand how developers collaborate by interviewing users or mining public GitHub large-scale data. Social behavior patterns of users have been explored (Y. Yu, Yin, Wang, & Wang, 2014) and a study on the evolution over time of a population of contributors has been done following a biological approach (Loyola & Ko, 2014). Others investigated the influence and popularity of users (Badashian, Esteki, Gholipour, Hindle, & Stroulia, 2014; Blincoe, Sheoran, Goggins, Petakovic, & Damian, 2016; Lima, Rossi, & Musolesi, 2014; Thung, Bissyande, Lo, & Jiang, 2013) and repositories (Aggarwal, Hindle, & Stroulia, 2014; Bissyande, Thung, Lo, Jiang, & Reveillere, 2013; Borges, Hora, & Valente, 2016; Hu, Zhang, Bai, Yu, & Yang, 2016; Thung et al., 2013). Interestingly, Hu *et al.* estimate repository influence by applying a weighted version of HITS to a bipartite graph containing user and repository nodes, that are linked whenever a user assigns a repository with a star.

The impact programming languages have on software quality (Ray, Posnett, Filkov, & Devanbu, 2014) and how commercial organizations use GitHub (Kalliamvakou, Damian, Blincoe, Singer, & German, 2015) have also been object of study. Further research has looked for insights on what motivates users to choose the repositories they want to contribute to (Allaho & Lee, 2014), the developers they collaborate with (Celińska, 2018), how they infer each other's social and technical capabilities (Dabbish, Stuart, & Herbsleb, 2012; Tsay, Dabbish, & Herbsleb, 2014) and how those aspects contribute to productivity (Casalnuovo, Vasilescu, Devanbu, & Filkov, 2015) and issue closure rates (Jarczyk, Jaroszewicz, Wierzbicki, Pawlak, & Jankowski-Lorek, 2018). Particularly, Jarczyk *et al.* created a SN with developers and projects as nodes, containing diverse link types: users share a "cooperation" link if they have cooperated and/or a "discussion" link if they have participated in discussions in the same project. Users can also be tied by a "Skill similarity between developers" if they are fluent in at least one common coding language. One edge type, "skill similarity between developers and repositories", connects users and projects if a user has participated in that project and is fluent in at least on of the project's required coding

languages. This network, alongside other analyses, was used to ascertain characteristics that predict issue closure rate and can be used by project managers to increase performance.

All studies here mentioned take advantage of the extensive data generated with the worldwide massive adoption of GitHub. As far as I am aware, no work has been published that uses GitHub for organizational management as proposed in this work plan.

Chapter 3 Methodology

This chapter contains a definition of the problem under study, with an explanation of how GitHub is used within the company. It provides a description of the data collected and of its storage in a Neo4j database. Then the reasoning behind the queries made to the database is clarified, as are the four user-user communication models generated.

3.1 Problem Description

This work aims to model and study the user-user communication network in a company where GitHub is the main platform used for work-related discussions and task tracking. To answer questions regarding communication between employees, a first network model of the entire GitHub "home" repository was generated and stored in a graph database, which was then queried to extract user-user communication sub-networks.

Despite GitHub's main use being for collaborative coding, the "home" repository under analysis is an issue repository rather than a code repository. Issues can be events of diverse nature, e.g. tasks, bugs, enhancements, etc. Some are related to human resource management events, such as recruitment, onboarding or leaving of employees; others may refer to customer support or marketing assignments. Bug reporting and discussions on actionable subjects, that originate new coding repositories or strategic decisions, are also done by means of creating issues in the "home" repository. Furthermore, all code repositories within the company are associated to an issue in the "home" repository. This is where all tasks employees are or have been working on thus far are tracked and centralized.

The structure of a typical GitHub issue (Figure 10) contains a short title, an issue ID number and information on the issue owner. It comprises a text body that can have tick boxes, code snippets, URLs and a list of pings. Pings are mentions made to specific people or teams to raise awareness about that issue. The issue owner can assign people to actively work on it (assignees), and label the issue with tags regarding its importance or other information useful for categorization. Each issue contains a comment feed where people involved can discuss relevant matters.

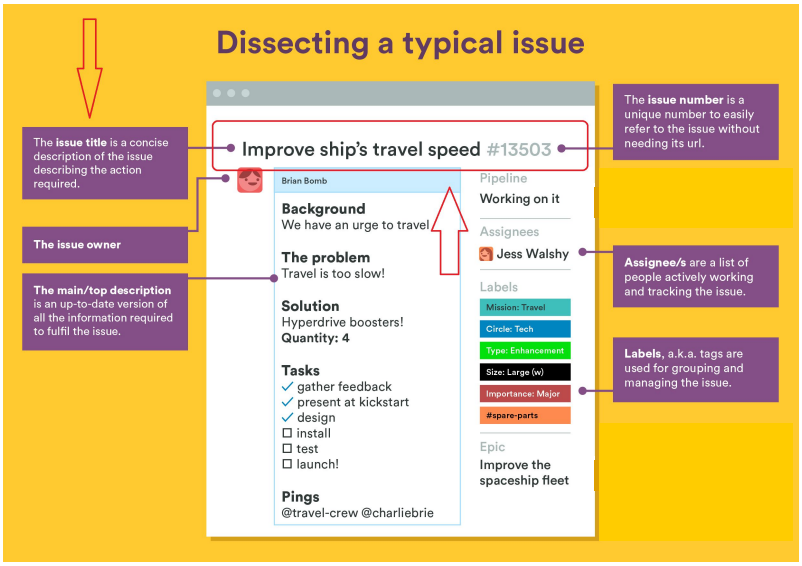


Figure 10: Overview of a GitHub issue structure. This picture is part of a presentation given internally within the company to introduce new employees to GitHub and advise on best practices.

3.2 Data description

The entire GitHub "home" repository data history of a medium sized company was collected, starting from its founding days in November 2011until July 2018.

Using information retrieved directly via Application Programming Interface (API) queries, current users were characterized with a unique user ID number and username. Each issue was described with a unique issue ID number, its title, full text body, issue owner (user ID), labels, creation and closure dates. Comments contain a unique comment ID number, full text body, creation date, username and user ID of its creator, as well as the corresponding issue ID number.

Text attributes were further parsed to extract other relevant features, such as presence of code snippets, URLs or checkboxes in the issue text body. The number of users and teams referred (i.e. pinged) in the text as well as its word count and issue title length were also collected.

3.3 Modelling GitHub "home" repository in Neo4j

Neo4j is a graph database that stores data relationships and is used to traverse, visualize and analyze connected data (Robinson, Webber, & Eifrem, 2015). It was developed to deal with connected data in a flexible way, allowing the addition of nodes and relationship types at any point during database development.

A graph model may contain different node and edge types, as well as store properties associated them, making it easy to model, store and query relationships. Unlike traditional relational databases, graph databases deliver constant query performance irrespective of overall database size. Queries are performed on a data subset so there is no need to use foreign keys and perform complex table joins.

The Neo4j graph database can be queried directly via its built-in web application or via API. It has its own declarative query language, Cypher QL, which is similar to SQL but optimized for graph traversing. Cypher QL is a very intuitive language that uses short and easy to read queries.

The GitHub "home" repository model developed in this work contains four different node types: Issue, Label, Comment and User, connected by one or more directed relationships, and each having associated properties (Figure 11).

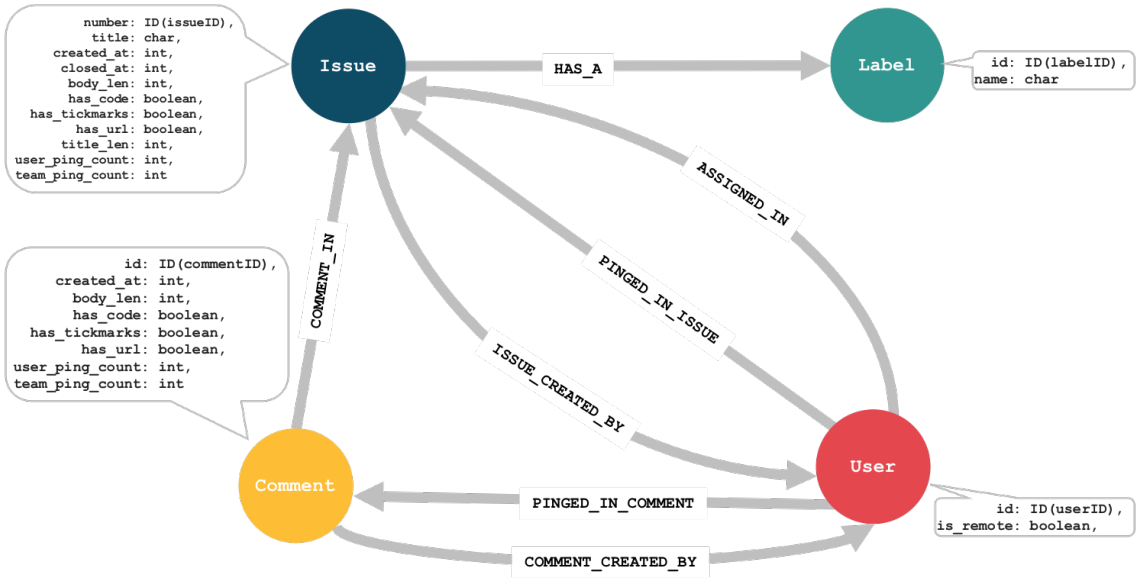


Figure 11: Graph model of the GitHub "home" repository, composed of four node and seven edge types. Each node contains several attributes, show in the respective text boxes.

The model is self-explanatory and very easy to read. It defines the "home" repository as containing issues, that are created by a user and may have labels. On their hand, users can be assigned in issues and pinged in either issues or comments. Comments are created by users in an issue.

3.4 Modelling user-user interaction

To model interaction between different users (u1 and u2), five rules were established and used to derive queries made to the graph database in order to extract the user-user communication sub-networks. The first two relationships describe interactions occurring when a user is assigned to (Figure 12 A) or pinged in (Figure 12Figure 11 B) an issue created by another user. Whenever a user is pinged in another user's comment, that event may also considered as communication (Figure 12 C). Relationships D and E are more obvious, and result from a user commenting directly an issue created by another user (Figure 12 D), or both users commenting on the same issue (Figure 12 E).

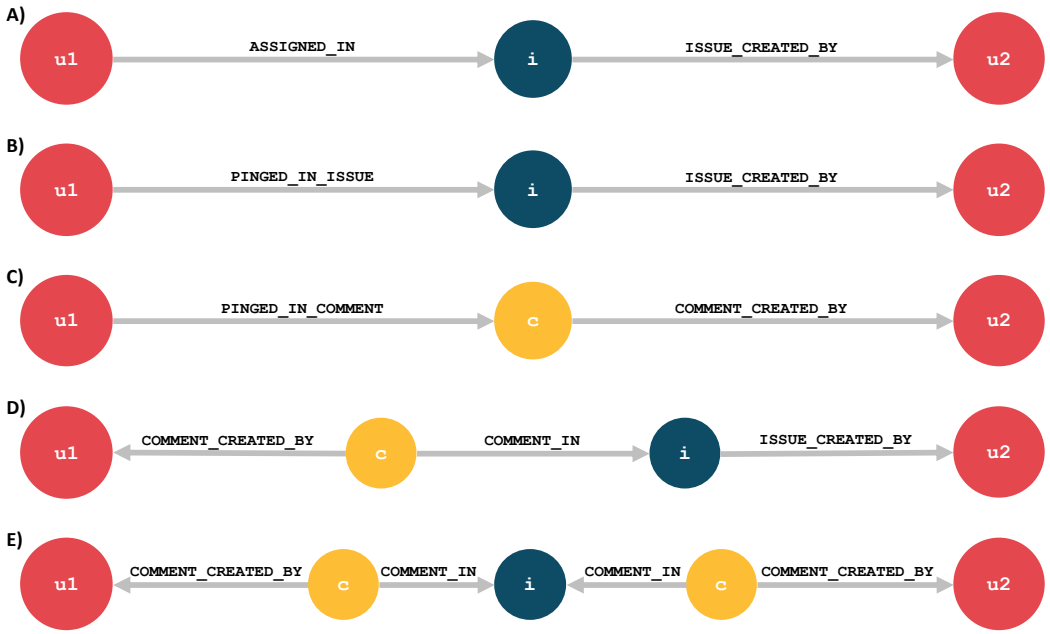


Figure 12: Relationships accounted for when modelling the interaction between users via GitHub. u1: User1, u2: User2, c: Comment; i: Issue.

To provide a better picture of the network, when available, additional employee information was added regarding gender, job role and most recent work cluster within the company.

Given that the objective of this work is to compare user-user interaction before and after a restructuring event that took place in the last months of 2017, sub-networks were generated containing information collected for homologous periods of 2017 and 2018. Specifically, months April, May and June of 2017 and 2018.

To understand the best way to model user-user interaction via GitHub, a total of four networks, two weighted and two unweighted, were generated for each time period. First off, the most complex network was generated, where all relationships are considered (A, B, C, D, and E) and their sum used for edge weighting. Then, a simplified weighted version where only relationships D and E are counted. Furthermore, unweighted versions of each network were generated. After assessing the distribution of edge weights in each network, a weight cutoff was chosen and utilized to generate unweighted networks. Only edges with a weight higher than the selected cutoff were kept (Figure 13). All networks were then compared to figure out the best way to model user-user interaction in this context.

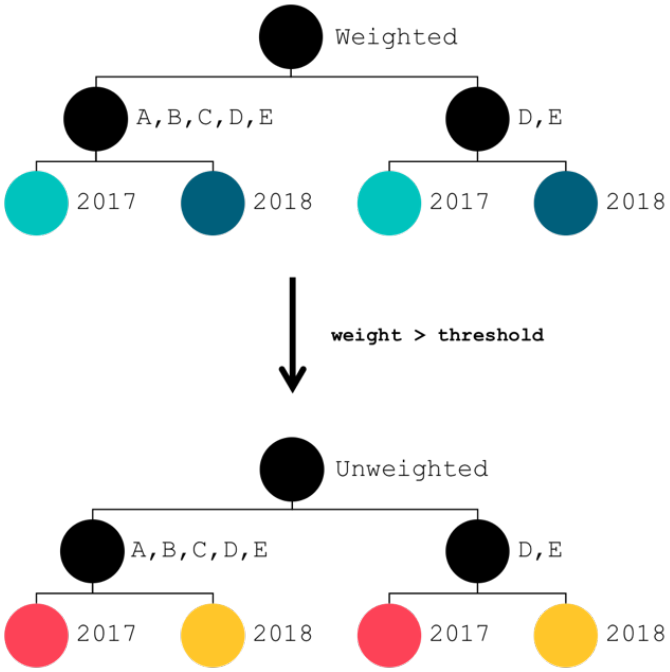


Figure 13: Illustration of the eight user-user interaction sub-networks generated. Networks for years 2017 and 2018 contain data pretraining to April, May and June.

Chapter 4 Results and Discussion

In this chapter, data collection and pre-processing are described. The graph database and its querying to generate user-user networks is then explored. A comparative evaluation of weighted and unweighted models is performed, using network measures and community detection algorithms. Reasoning behind the best model choice for the task at hand is explained. Finally, the chosen model is analyzed to compare differences in the two time periods analyzed and depict the most influential nodes.

4.1 Data Collection and Pre-Processing

Nearly 7 years of data were obtained using GitHub's REST API within R (v 3.5.0) with the RStudio IDE (v 1.1.442). Queries retrieved information on current users (n = 313), issues (n = 21563) and the first 30 comments (n = 122018) of every issue.

Issue body text and comments were parsed for the presence of URLs, tick marks and code snippets. Their number of words was counted (excluding URLs and code snippets), as were the total number of teams and users mentioned. Usernames mentioned in comments or body text were also stored. Because GitHub's API does not give access to a user history, to obtain the final user list (n = 430), users who were no longer with the company but had been mentioned, pinged or had commented an issue at any point in time, were added to the list. Given that a significant proportion (7.7%) of employees work remotely, information regarding their remote status was added.

The entire GitHub "home" repository history was modeled in Neo4j desktop (v 1.1.10), according to the previously described graph model (Figure 11). A summary of the nodes and edge types composing the network, as well as their counts, is given in Tables 2 and 3.

Table 2: Network node types and their count.

Node type	Number
"Issue"	21.506
"Comment"	122.018
"User"	430
"Label"	235

Table 3: Network edge types and their count.

Edge type	Number
"ASSIGNED_IN"	24.794
"COMMENT_CREATED_BY"	244.036
"COMMENT_IN"	243.828
"HAS_LABEL"	146.306
"ISSUE_CREATED_BY"	43.012
"PINGED_IN_COMMENT"	118.660
"PINGED_IN_ISSUE"	46.300

A first inspection of the network was done by querying the database directly in Neo4j's browser, and immediately visualizing the resultant sub-networks. Two query examples and their outputs are presented below. Figure 14 shows 10 relationships following a pattern where a user is linked to a comment in an issue to which a second, distinct, user is also linked to. Figure 15 exhibits 10 relationships where two distinct users are related to different comments on the same issue.

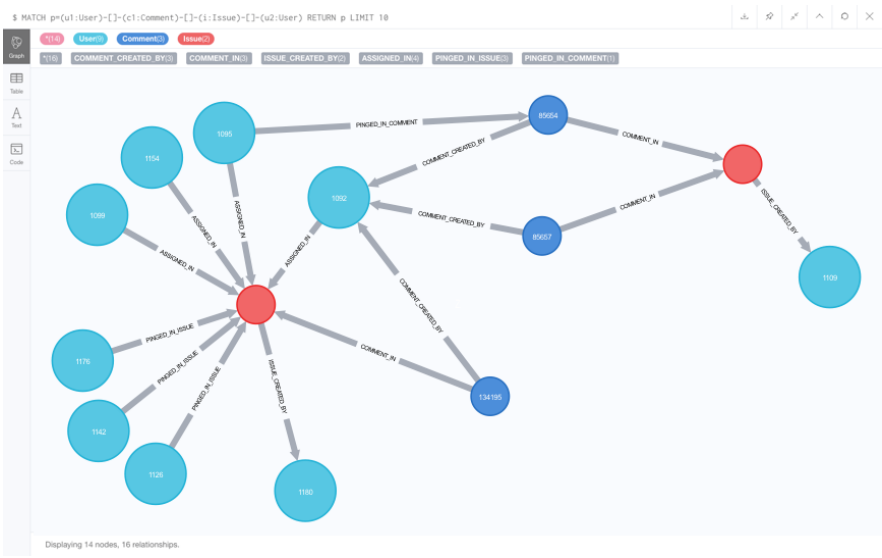


Figure 14: Sub-network obtained by querying Neo4j with the query **MATCH p= (u1:User)-[]-(c1:Comment)-[]-(i:Issue)- []-(u2:User) RETURN p LIMIT 10**. Users: light blue nodes; comments: dark blue nodes; issues: red nodes; relationships: grey arrows.

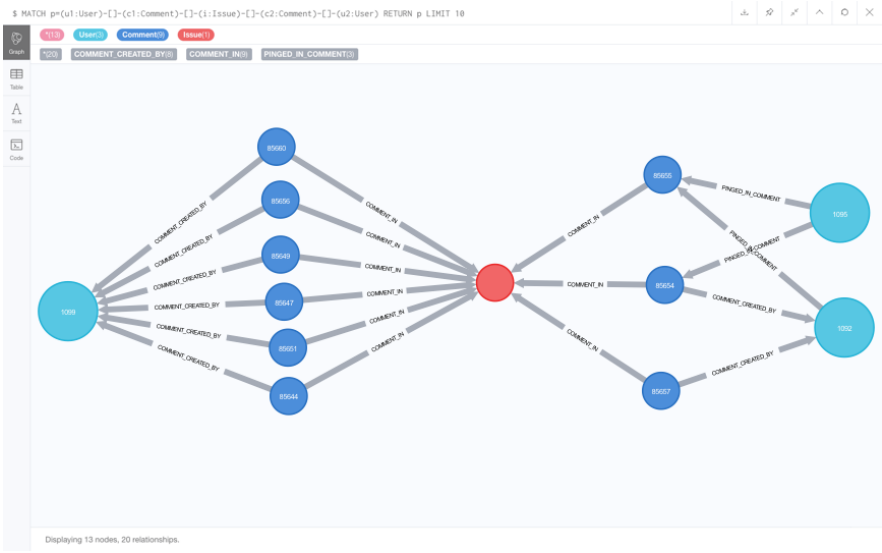


Figure 15: Sub-network obtained by querying Neo4j with the query **MATCH p=(u1:User)-[]-(c1:Comment)-[]-(i:Issue)-[]-(c2:Comment)-[]-(u2:User) RETURN p LIMIT 10**. Users: light blue nodes; comments: dark blue nodes; issues: red nodes; relationships: grey arrows.

The Neo4j graph database was then queried using the "RNeo4j" (v1.6.4) R package (White, n.d.) to find user pairs sharing the defined relationships. The sum of user interactions was used as tie weight to build undirected weighted networks with "tnet" (v3.0.14) (Opsahl, 2009) R package. Relevant extracts of the R scripts that include queries made to the Neo4j database and their post-processing can be found in Annex I.

When available, user node information was updated to contain gender, role and company cluster information. During this step, five users were identified as bots and removed from the sub-networks.

4.2 Computation of Network Measures

All network measures (betweenness, closeness, degree, shortest path length, local and global clustering coefficients) were computed with "tnet". For weighted networks, betweenness, closeness and degree were calculated with the alpha parameter set to 1.5. Closeness was divided by the number of nodes in the network minus one (N-1), and is denoted as normalized closeness.

4.3 Statistical Analysis

To compare network statistics between sub-networks generated with data from 2017 with homologous data from 2018, that is, before and after the company's internal restructuring, the normality of sample distributions was first assessed with quantile-quantile (Q-Q) plots and Shapiro-Wilk tests. Afterwards, appropriate statistical tests were performed for each case. All tests were done for a 95% confidence interval, using the "stats" (v3.5.0) R package.

Quantile-quantile (Q-Q) plots allow a graphical assessment of the goodness-of-fit of a sample distribution to a theoretical distribution, in this case the normal distribution. By plotting the quantiles of both and visually comparing them, it is possible to visualize if the sample is normally distributed according to how well it fits the theoretical distribution quantiles.

After inspection of the Q-Q plots, the Shapiro-Wilk test was used to confirm the results. This test assesses the normality of a distribution with following working hypotheses:

H_0: the sample came from a normal distribution.

H_1: the sample did not come from a normal distribution.

For non-normally distributed or ordinal samples, the non-parametric bilateral Mann–Whitney–Wilcoxon U test with continuity correction was executed to evaluate if the equality of their distribution, using the hypotheses:

H_0: the distributions of the two samples are the same.

H_1: the sample distributions differ by some location shift.

When enough evidence existed to reject the null hypothesis, specific unilateral U tests were performed.

For normally distributed variables, Bartlett test was first used to evaluate the equality of sample variances, having as hypotheses:

H_0: the variances of the two samples are the same.

H_1: the sample variances are different.

Given that for all cases the compared sample variances were different, the parametric Welch two sample t-test was performed to compare their means, using the hypotheses:

H_0: the two samples have equal means.

H_1: the two samples have different means.

When enough evidence existed to reject the null hypothesis, specific unilateral tests were performed.

4.4 Model Comparison and Selection

4.4.1 Network Measures

4.4.1.1 Weighted Networks

Comparing the network model based on the sum of all possible relationships (ABCDE) with the simplified version that counts only relationships D and E, a decrease in the number of nodes and edges is evident. Decreasing the number of relationships used in the model resulted in a 15% node and 16% edge removal in 2018. In 2017, 12% of nodes and 13% of edges were eliminated (Table 4). These percentages are identical, and tell us that this simplification affected both networks in a similar way. Maximum network weights were only slightly affected by the removal or relationships ABC from the model. Average path length is similar in both time periods and models. However, the network diameter of the 2017 network was much more affected by the relationship removal than was the 2018 one.

Not all nodes were annotated regarding their role, location and work cluster (Table 5), but there are no differences in the ratios of annotated nodes between models.

Table 4: Indicators describing the weighted sub-networks: count of nodes and edges, maximum edge weight, average and global clustering coefficients, network diameter and average path length.

	nodesCount	edgesCount	maxWeight	avg cc	global cc	diameter	avg path len
ABCDE.2017	187	1450	137	0.507	0.381	25.82	3.31
ABCDE.2018	268	1744	184	0.529	0.427	17.46	3.71
DE.2017	164	1258	126	0.574	0.394	14.6	2.33
DE.2018	228	1463	165	0.649	0.451	14.65	2.98

Table 5: Ratio of nodes in the weighted networks that are annotated with job role, remote status, location and work cluster.

	hasRole	hasRemote	hasLocation	hasCluster
ABCDE.2017	0.54	1	0.58	0.54
ABCDE.2018	0.67	1	0.71	0.67
DE.2017	0.57	1	0.61	0.57
DE.2018	0.68	1	0.73	0.68

When analyzing the distribution of network measures between 2017 and 2018, for both the ABCDE and DE models, shortest path lengths, degree, strength and closeness distributions are all significantly different. In both models, the global clustering coefficient (Table 4), calculated taking the weight of edges into account (2.23) is greater in 2018 than in 2019. The biggest differences between models seem to lay in the distribution of closeness and local clustering coefficient, the latter being statistically different between years only in the DE model.

It is in node closeness that the models differ the most (Figure 16). There is evidence to reject the null hypothesis and affirm that closeness distribution is shifted to the left in 2018 compared to 2017 in both ABCDE and DE networks (U test; ABCDE p-value $< 2.2 \times 10^{-16}$; DE p-value $< 2.2 \times 10^{-16}$). In both cases, there is a great decrease in mean closeness, normalized by the number of nodes in the network, from 5.7×10^{-6} to 2.4×10^{-6} in the ABCDE model and from 1.4×10^{-5} to 5.0×10^{-6} in the DE model. Furthermore, in the DE model closeness values are higher than those observed in the ABCDE one.

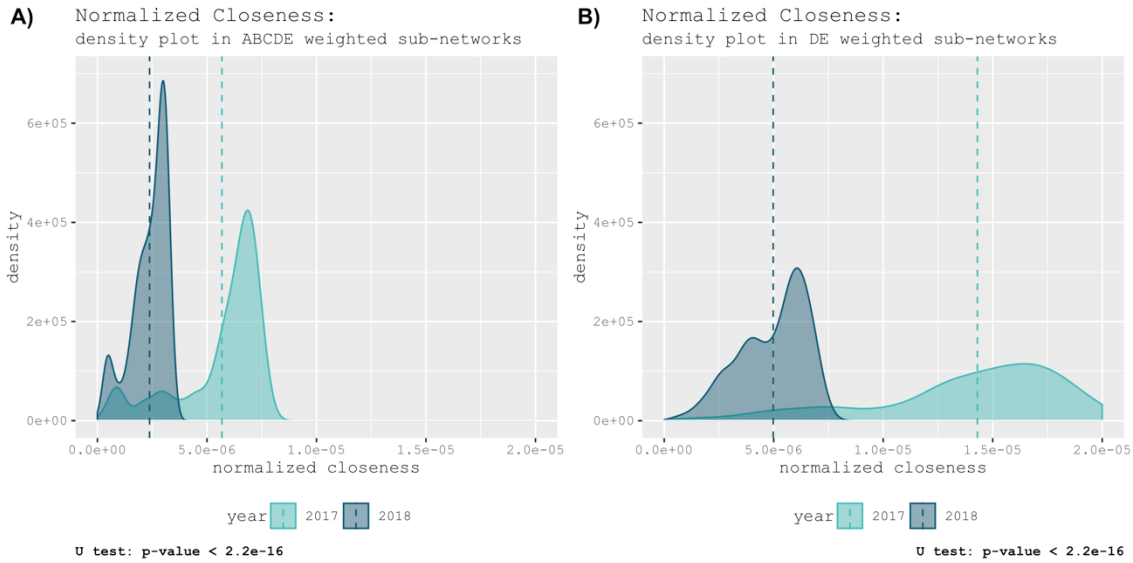


Figure 16: Node closeness distribution, divided by N-1, in weighted A) ABCDE and B) DE relationship models. Dotted lines (---) indicate mean values.

Regarding local clustering coefficient, only in the DE model is there evidence to state that right-shift occurs in 2018 data (U test: DE p-value = 0.005948) (Figure 17). The number of nodes with local clustering coefficient equal to one is higher in the DE model, accounting for 15.9% (26/164) of all nodes in 2017 and 25% (57/228) in 2018. In the ABCDE networks, a lower number of nodes have this characteristic: 9.6% (18/187) in 2017 and 13.1% (35/268) in 2018.

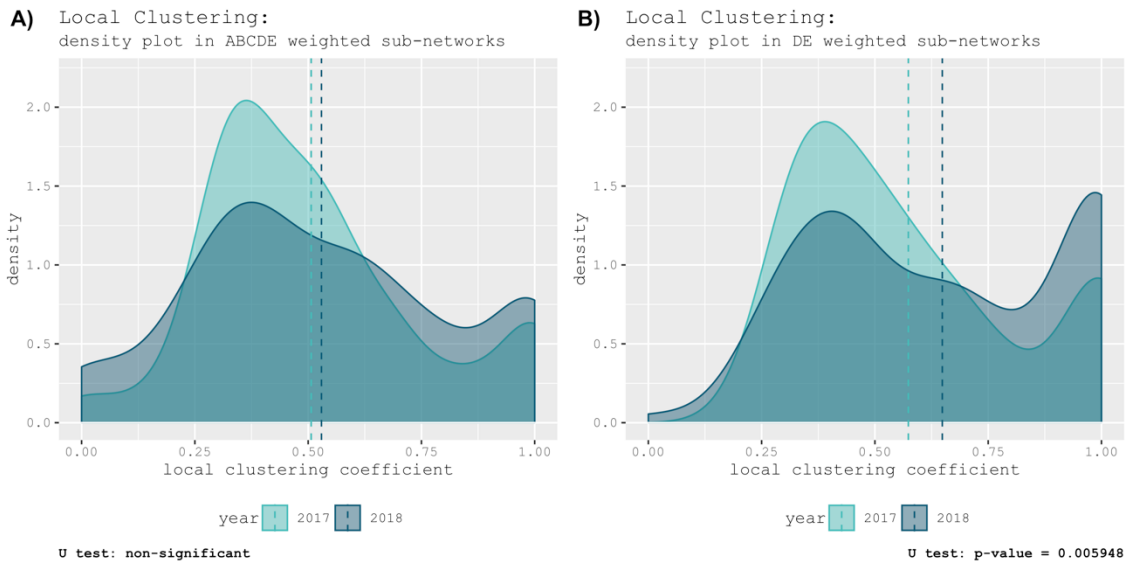


Figure 17: Node local clustering coefficient in weighted A) ABCDE and B) DE relationship models. Dotted lines (---) indicate mean values.

Taking a closer look at degree distribution (Figure 27, Annex I) for the both ABCDE and DE models, there is enough evidence to support the hypothesis that the 2018 distribution is shifted to the left (U test: ABCDE p-value = 0.01289; DE p-value = 0.0009582). Even though statistically significant, the shift is slight. Mean degree for ABCDE was 15.5 in 2017, compared to 13.0 in 2018. For the DE model, the mean decreased from 15.3 to 12.8 in 2018. Regarding this measure, results are similar in both models, with the distribution looking smoother in the simpler one (Figure 27 B, Annex I).

Node strength is also statistically different in both models (Figure 28, Annex I), with its distribution being displaced to the left in 2018 (U test: ABCDE p-value = 0.001939; DE p-value = 0.002515). Between years, mean node strength decreases slightly from 116.3 to 87.1 in the ABCDE network and from 105.1 to 81.7 in the DE network. Overall, results for this metric do not differ in both models.

According to the statistical test performed, there is evidence that betweenness does not differ between 2018 and 2017 in both models (two-sided U test: ABCDE p-value = 0.3405; DE p-value = 0.337) (Figure 29, Annex I), with a mean node betweenness of 328.2 in 2017 and 538.6 in 2018 for ABCDE, and 271.1 and 422.6 for DE networks.

For both models, the distribution of shortest path length is deviated to the right in 2018 compared to 2019 (U test: ABCDE p-value < 2.2×10^{-16}; DE p-value < 2.2×10^{-16}) (Figure 30, Annex I). This shift is greater in the DE model with the mean increasing by 0.7, from 2.3 to 3, as compared to the ABCDE model where mean shortest path increases just 0.4, from 3.3 to 3.7. As previously discussed, both network diameters are closer in the DE model (Table 4) and with a much smoother distribution.

In sum, the greatest difference between these models when looking at network measures is in node closeness and local clustering coefficients. The DE model is able to provide a better separation for 2017 and 2018 and, given that it is the simplest one, it seems to be the best weighted network model for the data under study.

4.4.1.2 Unweighted Networks

To generate unweighted versions of each of the previously discussed networks, distribution of their edge weights was analyzed to select a cutoff value. As a first approach, the 40% percentile was chosen by visual inspection of weight distribution plots (Figure 18). This cutoff provided a good compromise between discarding a large number of less informative edges while keeping the most informative node pairs. Choosing lower of higher

percentiles seemed like a more unbalanced choice (Table 6), where too few low weight edges or too many putatively relevant edges would be discarded.

When generating the 2018 unweighted DE network, two nodes were disconnected from the principal component (userIDs: 9280378, 12448176) and were thus removed from the analysis.

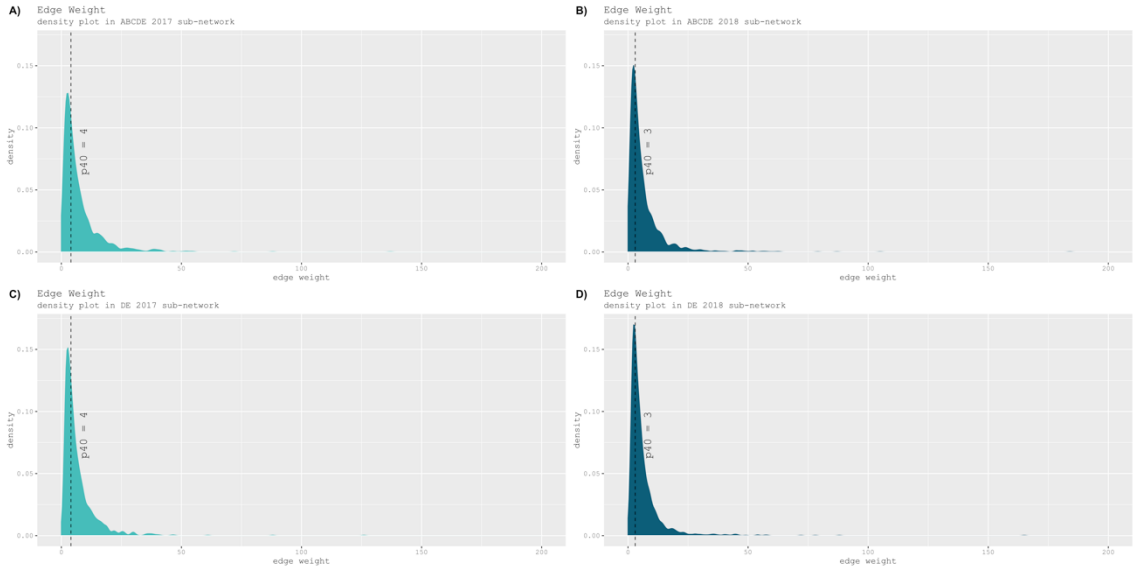


Figure 18: Distribution of edge weights in the ABCDE network model generated with A) 2017 and B) 2018 data; and DE network model for C) 2017 and D) 2018. Dotted line (---) marks the 40% percentile, chosen as cutoff to generate the unweighted networks.

Table 6: Weight distribution percentiles in the weighted sub-networks.

	5%	10%	20%	30%	40%	50%
ABCDE.2017	1	2	2	3	4	5
ABCDE.2018	1	1	2	2	3	4
DE.2017	2	2	2	3	4	5
DE.2018	2	2	2	3	3	4

For unweighted models, the most noticeable change when using just the DE relationships lays in the number of edges, that decreases by 117 and 121 for 2017 and 2018 networks, respectively. Network diameters remain the same for 2017 but decreases from 9 to 7 in the 2018 networks by removing ABC relationships. When comparing ABCDE and DE unweighted models, the number of nodes varies only slightly and there isn't much difference regarding average path lengths and clustering coefficients (Table 7). The ratio of annotated nodes remains identical irrespective of the model applied (Table 8) and is slightly higher than what was observed in weighted network.

Table 7: Indicators describing the unweighted sub-networks: count of nodes and edges, maximum edge weight, average and global clustering coefficients, network diameter and average path length.

	nodesCount	edgesCount	avg cc	global cc	diameter	avg path len
unweighted ABCDE.2017	153	760	0.446	0.268	5	2.58
unweighted ABCDE.2018	231	968	0.519	0.314	9	3.17
unweighted DE.2017	144	643	0.472	0.259	5	2.65
unweighted DE.2018	203	847	0.551	0.325	7	3.03

Table 8: Ratio of nodes in the unweighted networks that are annotated with job role, remote status, location and work cluster.

	hasRole	hasRemote	hasLocation	hasCluster
unweighted ABCDE.2017	0.58	1	0.62	0.58
unweighted ABCDE.2018	0.68	1	0.72	0.68
unweighted DE.2017	0.59	1	0.63	0.59
unweighted DE.2018	0.7	1	0.75	0.7

Regarding closeness distribution, the unweighted models are not as dissimilar as the weighted ones. Normalized closeness is much similar between models but still significantly lower in 2018 (t-test: ABCDE p-value < $2.2x10^{-16}$; DE p-value < $2.2x10^{-16}$) (Figure 19).

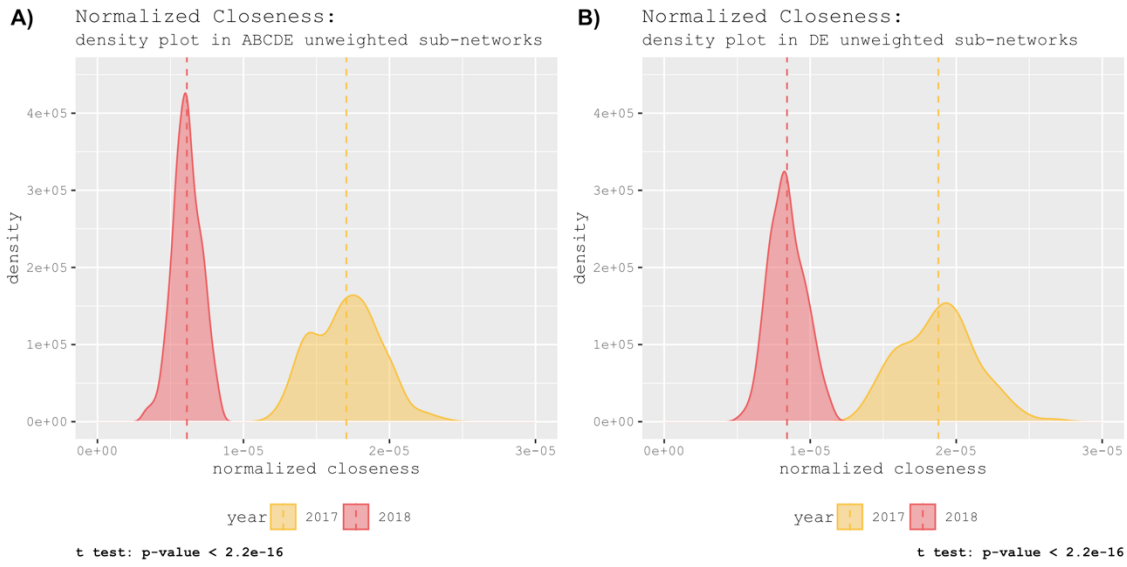


Figure 19: Node closeness distribution, divided by N-1, in unweighted A) ABCDE and B) DE relationship models. Dotted lines (---) indicate mean values.

Node degree distribution in the ABCDE model is consistent with results obtained with weighted models and maintains the difference observed between time periods, with a distribution left-shift in 2018 (U test: ABCDE p-value = 0.003448) (Figure 20). This difference however is lost in the perhaps over-simplified DE model.

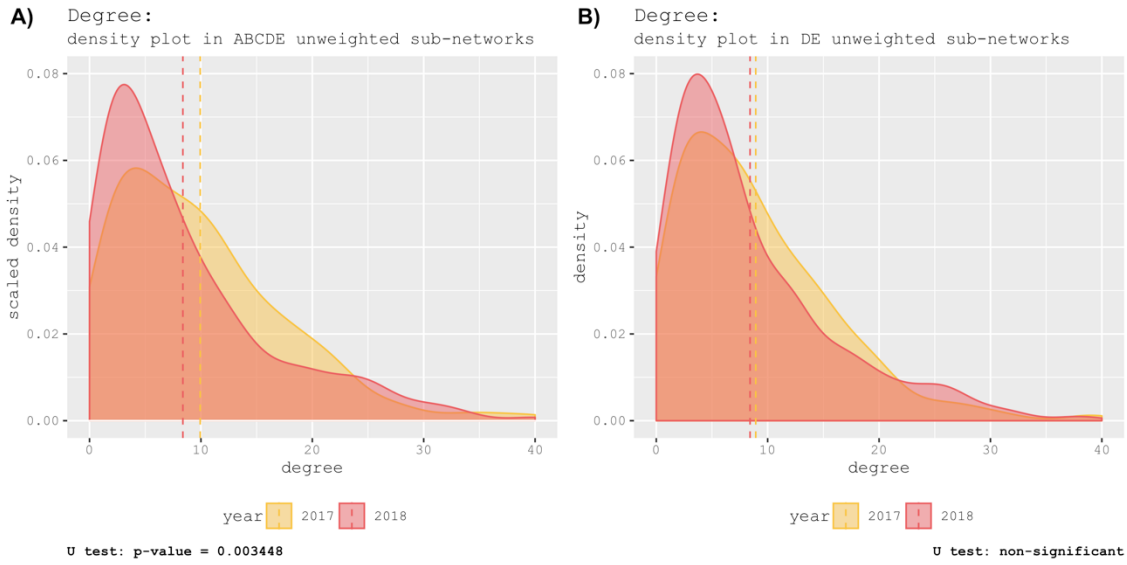


Figure 20: Node degree distribution in unweighted A) ABCDE and B) DE relationship models. Dotted lines (---) indicate mean values.

Shortest path length distribution shifts to the right in 2018 in both ABCDE and DE models (U test: ABCDE p-value < 2.2x10-16; DE p-value < 2.2x10-16) (Figure 31, Annex I).

There is no statistical evidence to affirm that betweenness changes between the two time periods in either model (Figure 32, Annex I), an equality that is consistent with results obtained with weighted versions.

Local clustering coefficient is identical in both models (Figure 33, Annex I) and U test results provide evidence to support the affirmation that the distribution suffers a right-shift in 2018 (U test: ABCDE p-value = 0.01936; DE p-value = 0.008567).

Overall, the ABCDE seems to be the best unweighted model since it retains all the statistically significant differences between time periods evident in the DE weighted model. Possibly due to an over-simplification of the data, the same is not verified for the DE unweighted model.

4.4.2 Community Detection

To discriminate communities, several algorithms implemented in the "igraph" (v1.2.2) R package (Csardi & Nepusz, 2006) were applied to the networks using default parameters. The algorithm that resulted in higher modularity was chosen for each case (Table 9).

For weighted networks, the selected algorithm was cluster fast greedy. This algorithm detects communities with a greedy optimization of modularity. For unweighted networks,

the preferred algorithm was cluster spin-glass, that uses the spin-glass model and simulated annealing to find communities.

Table 9: Network modularity after applying each community detection algorithm. Underlined and highlighted in bold are the algorithms chosen to detect communities in weighted and unweighted models.

	weighted				unweighted			
	ABCDE		DE		ABCDE		DE	
	2017	2018	2017	2018	2017	2018	2017	2018
cluster_fast_greedy	**0.432**	**0.594**	**0.427**	**0.587**	0.336	0.476	0.353	0.488
cluster_walktrap	0.398	0.583	0.402	0.579	0.339	0.494	0.382	0.507
cluster_infomap	0.425	0.584	0.423	0.594	0.354	0.512	0.403	0.525
cluster_leading_eigen	0.369	0.526	0.388	0.556	0.341	0.477	0.348	0.486
cluster_spinglass	0.057	0.089	0.070	0.098	**0.394**	**0.530**	**0.413**	**0.540**
cluster_edge_betweenness	0.223	0.443	0.278	0.462	0.258	0.436	0.322	0.462
cluster_label_prop	0.004	0.544	0.006	0.572	0.000	0.380	0.000	0.372

In weighted networks, the change in number of communities detected from 2017 to 2018 was the smallest. This change represented only one community in the ABCDE model and two in the DE model (Table 10). The unweighted ABCDE model is the one where the biggest difference in the number of detected communities exist between, doubling from five in 2017 to ten in 2018 (Table 11).

Table 10: Network modularity and number of communities detected in weighted networks using the cluster fast greedy algorithm.

	cluster_fast_greedy : modularity	nr communities
weighted.ABCDE.2017	0.432	7
weighted.ABCDE.2018	0.594	8
weighted.DE.2017	0.427	6
weighted.DE.2018	0.587	8

Table 11: Network modularity and number of communities detected in unweighted networks using the cluster spin-glass algorithm.

	cluster_spinglass : modularity	nr communities
unweighted.ABCDE.2017	0.394	5
unweighted.ABCDE.2018	0.530	10
unweighted.DE.2017	0.413	6
unweighted.DE.2018	0.540	10

4.4.3 Concluding on the Chosen Model

After analyzing the network measures, the DE weighted and ABCDE unweighted models are the ones that capture the most differences in network centrality measures between time periods. The ABCDE weighted may be too complex and the DE unweighted too simplified to capture them. Furthermore, when detecting communities, the DE unweighted model provides the greatest difference in terms of community numbers. Unweighted models are

also easier to interpret, so the ABCDE unweighted model was selected as the best approach to model the data under study.

4.5 ABCDE Unweighted Model Analysis

4.5.1 Network Measures

Having chosen the ABCDE unweighted model as a reference, and as previously stated, in the 2018 network, closeness and degree increase and there is a decrease in local clustering coefficients and shortest path lengths, when comparing with the 2017 network. Average node closeness decreased from $1.7x10^{-5}$ to $6.1x10^{-6}$. If more nodes have a lower closeness, it means that, on average, the shortest path lengths connecting them to the other nodes in the network are longer. In 2017 50% of nodes had 9 connections whereas in 2018 only 6 (Table 12), and the average node degree in 2018 decreased from 9.9 to 8.4. Consistently, mean shortest path increases, from 2.6 to 3.2. Network diameter increased from 5 to 9 and while network density decreased from 0.07 to 0.04. On the other hand, mean clustering coefficient increased from 0.4 to 0.5, with more nodes forming closed triplets.

Table 12: Weight distribution percentiles in the ABCDE unweighted sub-networks.

	5%	10%	20%	30%	40%	50%
unweighted ABCDE.2017	1	2	3	5	6	9
unweighted ABCDE.2018	1	1	2	3	4	6

All these metrics indicate that globally in 2018, users communicate with fewer colleagues but they do so in more interconnected, cohesive groups.

To understand which employees take more relevant positions in the network, nodes with the five highest scores in each measure were selected from both networks.

4.5.1.1 Highest Betweenness

Betweenness Centrality measures the number of times a node lies in the paths of a graph. Nodes acting as bridges between communities in the network have high betweenness centrality. They connect different network groups and are of extreme importance because they form bottlenecks in information flow, having a higher level of control over it.

In the 2017 network no particular job role seems to stand out (Table 13). Interestingly, in the 2018 network nodes with the highest betweenness are producers (Table 14), which is the

same to say product managers. They are the people responsible for planning and executing projects.

Table 13: Nodes in 2017 ABCDE unweighted network with the five highest betweenness centrality scores.

userID	betweeness	role	cluster
20163620	1303	Chief Marketing Officer	Senior Leadership
6143467	1097	Producer	Casino Product Cluster
16758472	944		
12133416	658	Systems Engineer	Platform Cluster
1101238	640	Software Engineer - Backend	Games Cluster

Table 14: Nodes in 2018 ABCDE unweighted network with the five highest betweenness centrality scores.

userID	betweeness	role	cluster
6450911	2858	Producer	Player Services Cluster
34676288	2841	Producer	Casino Product Cluster
13198411	2424		
6143467	1887	Producer	Casino Product Cluster
25950012	1826	Producer	Games Cluster

4.5.1.2 Highest Closeness Centrality

Closeness Centrality is a measure of reachability that gives an idea about how long it will take to reach other nodes from a given starting node. Lower closeness centrality values represent bigger distances separating a node from all other nodes in the network and thus a lower chance of receiving information.

In the 2017 network, the nodes with highest closeness centrality are diverse, ranging from collaborators in management and leadership positions to more technical jobs such as a systems engineer (Table 17). Once more, in the 2018 network producers stand out as central nodes in the network (Table 18).

Table 15: Nodes in 2017 ABCDE unweighted network with the five highest closeness centrality scores.

userID	norm.closeness	role	cluster
6143467	2.35e-05	Producer	Casino Product Cluster
20163620	2.23e-05	Chief Marketing Officer	Senior Leadership
12133416	2.23e-05	Systems Engineer	Platform Cluster
16758472	2.20e-05		
5127360	2.08e-05	Project Manager	Casino Product Cluster
19772430	2.08e-05	Producer	Casino Product Cluster

Table 16: Nodes in 2018 ABCDE unweighted network with the five highest closeness centrality scores.

userID	norm.closeness	role	cluster
37797854	8.35e-06	UKGC Compliance Manager	Business Cluster
6450911	8.30e-06	Producer	Player Services Cluster
34676288	8.17e-06	Producer	Casino Product Cluster
25004713	8.11e-06	Producer	Business Cluster
16758472	8.07e-06		

4.5.1.3 Highest Degree Centrality

Degree centrality statistic measures general network connectivity. It is a count of the number of edges that stem from a particular node. In both networks, people in managing positions seem to be the most highly connected ones, with no great apparent differences (Table 17, Table 18).

Table 17: Nodes in 2017 ABCDE unweighted network with the five highest degree centrality scores.

userID	degree	role	cluster
6143467	40	Producer	Casino Product Cluster
20163620	36	Chief Marketing Officer	Senior Leadership
16758472	33		
19772430	28	Producer	Casino Product Cluster
6554540	27		
12133416	27	Systems Engineer	Platform Cluster

Table 18: Nodes in 2018 ABCDE unweighted network with the five highest degree centrality scores.

userID	degree	role	cluster
6450911	40	Producer	Player Services Cluster
6143467	33	Producer	Casino Product Cluster
25004713	33	Producer	Business Cluster
28704367	31		
34023263	30	Creative Talent Lead	Marketing Cluster
37797854	30	UKGC Compliance Manager	Business Cluster
22749254	28	Market Manager	Marketing Cluster

4.5.1.4 Lowest Degree

Both networks are formed by a single component, meaning there are no isolated nodes. However, in 2017 6.5% (10/153) of nodes had only one connection, whereas this number nearly doubled to 12.6% (29/231) in 2018. These nodes represent employees of diverse roles in the 2017 network (Table 19). On the other hand, in 2018 there is a predominance of technical roles and of people with customer support functions (i.e., ambassadors) (Table 20).

Table 19: Nodes in 2017 ABCDE unweighted network with degree centrality of one.

userID	degree	role	cluster
18258437	1	Ambassador	Player Services Cluster
29193619	1	Infosec Engineer	Platform Cluster
20107341	1	Content Creative	Marketing Cluster
25640071	1	Social Media Strategist	Marketing Cluster
6780979	1	Office Manager	Employee Experience Cluster
20041438	1		
13731644	1		
19647185	1		
23096062	1		
20576982	1		

Table 20: Nodes in 2018 ABCDE unweighted network with degree centrality of one.

userID	degree	role	cluster
1454079	1	Chief Operations officer	Senior Leadership
2466081	1	Chief Executive Officer	Senior Leadership
38035747	1	Ambassador	Player Services Cluster
33895262	1	Ambassador	Player Services Cluster
16882247	1	Talent Lead	Player Services Cluster
20967490	1	Ambassador	Player Services Cluster
38035728	1	Ambassador	Player Services Cluster
38985769	1	Ambassador	Player Services Cluster
9280378	1	Systems Engineer	Platform Cluster
31063966	1	Systems Engineer	Platform Cluster
17699574	1	Affiliates Manager	Marketing Cluster
6574641	1	People Operations Administrator	Employee Experience Cluster
39375632	1	Office Coordinator	Employee Experience Cluster
27953053	1	People Operations Specialist	Employee Experience Cluster
25954914	1	Software Engineer - Backend	Casino Product Cluster
73201	1	Software Engineer - Frontend	Casino Product Cluster
709418	1	Software Engineer - Frontend	Casino Product Cluster
2766	1		
5855477	1		
1931919	1		
18718683	1		
20041438	1		
18261928	1		
13149921	1		
38035790	1		
39045390	1		
38034050	1		
36402747	1		
32838105	1		

4.5.1.5 Highest Eigenvector Centrality

Eigenvector centrality is a more elaborated version of the degree. It quantifies the importance of a node based on how influential and important its neighbors are. It assumes that not all connections have the same importance by considering not only the quantity, but especially the quality of connections.

Individuals with high eigenvector scores are leaders of the network in the sense that their connections have many connections themselves. In both networks, individuals in leadership and management positions assume the greatest eigenvector values (Table 21, Table 22).

Table 21: Nodes in 2017 ABCDE unweighted network with the five highest eigenvector centrality scores.

userID	eigenValue	role	cluster
6143467	1.00	Producer	Casino Product Cluster
20163620	0.84	Chief Marketing Officer	Senior Leadership
19772430	0.78	Producer	Casino Product Cluster
6554540	0.74		
16758472	0.67		

Table 22: Nodes in 2018 ABCDE unweighted network with the five highest eigenvector centrality scores.

userID	eigenValue	role	cluster
37797854	1.00	UKGC Compliance Manager	Business Cluster
6450911	0.87	Producer	Player Services Cluster
22749254	0.87	Market Manager	Marketing Cluster
6143467	0.83	Producer	Casino Product Cluster
16758472	0.81		

4.5.1.6 Clustering coefficient

Nodes with a clustering coefficient of one form the cohesive groups highest possible density of ties, where all nodes are connected between them. In 2017, 13.1% (20/153) of nodes have a clustering coefficient of one and 20.8% (48/231) in 2019 (Table 23, Table 24 Annex I). Developers, analysts, ambassadors and senior leaders are represented in these tables, with analysts greatly increasing their expression in 2018.

4.5.2 Comparing Business Clusters and Communities

The observations above translate very well to the network plots. The network generated with 2017 data appears more loosely connected and with larger communities, whereas in the 2018 network several smaller clusters stand out, with some interesting high betweenness nodes connecting them (Figure 21). Particularly, the light-green and red clusters seem to be contributing to the increase in node clustering coefficients.

Bearing in mind that after the company's organizational restructuring people were grouped into business-unit clusters, it is possible to observe how well they fit the communities detected with cluster spinglass, and where users belonging to those cluster stood prior to the reorganization.

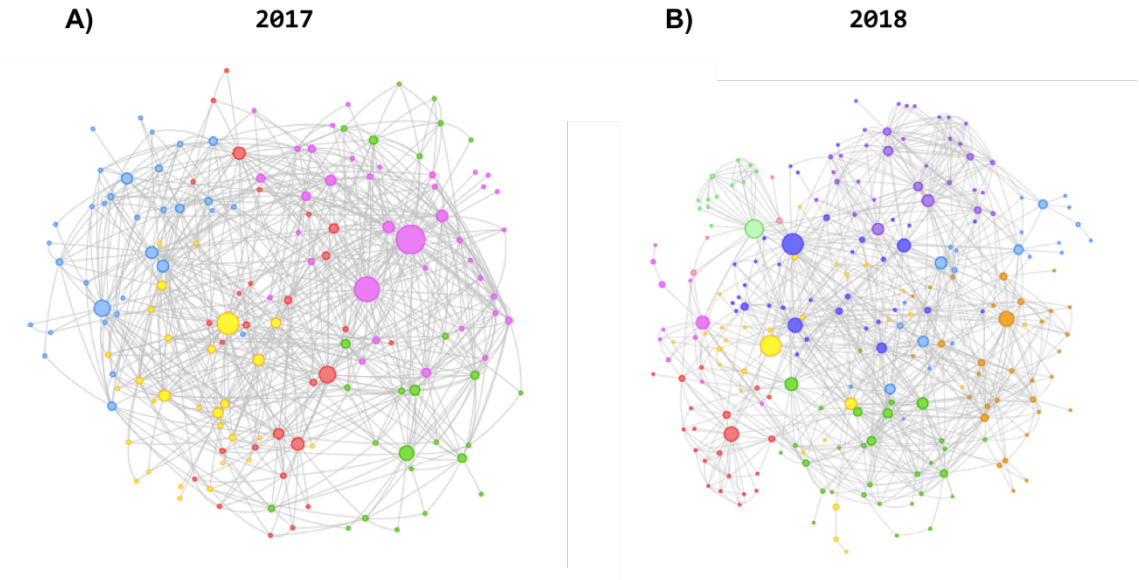


Figure 21: User-user communication network, created with the ABCDE unweighted model, for A) 2017 and B) 2018. Colors represent the cluster membership of said nodes, as computed by the clustering algorithm. Node size represents betweenness.

The games cluster stands out as being particularly tight and well defined in 2018, perfectly identified by the community detection algorithm (Figure 22). The player services cluster is another interesting case (Figure 23), with the vast majority of its known nodes sitting in the blue, purple and light-green communities in 2018 and one node with the highest betweenness (userid: 6450911). This node represents a producer, who scores high in all 2018 network measures except, obviously, clustering coefficient. Employee experience and Marketing clusters, are also very well confined by detected communities (Figure 34, Figure 35, Annex I).

In conclusion, the 2018 network structure seems to reflect the structural reorganization. Unlike in the 2017 network, where communication was more disperse, employees are communicating in smaller, betted defined groups.

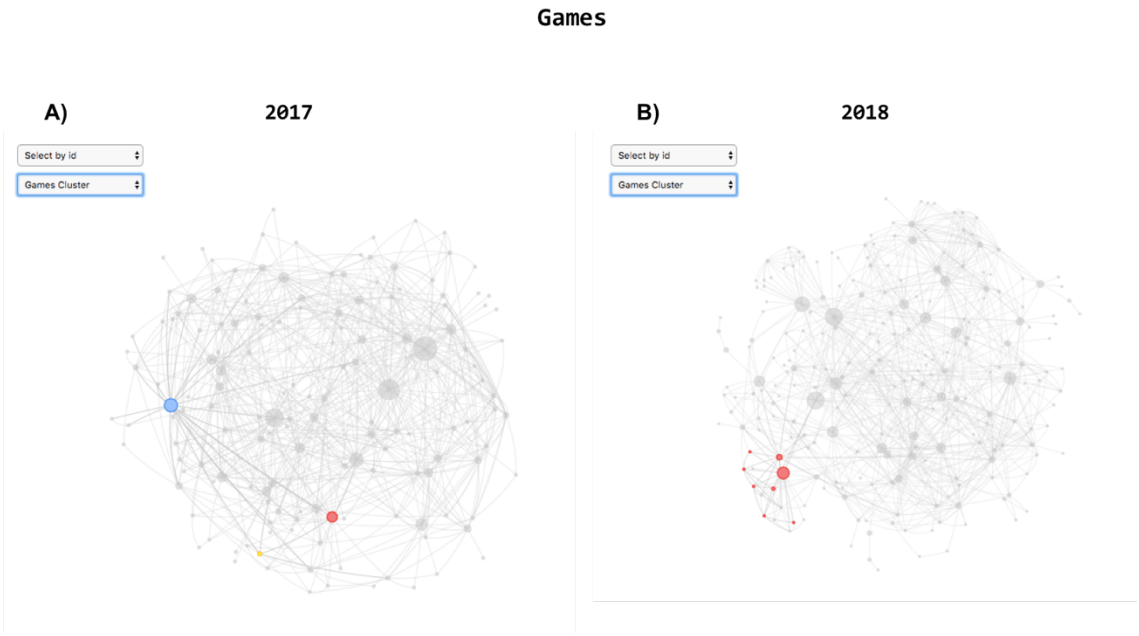


Figure 22: Networks highlighting the known games business cluster collaborators. Colors represent the cluster membership of said nodes, as computed by the clustering algorithm. Node size represents betweenness.

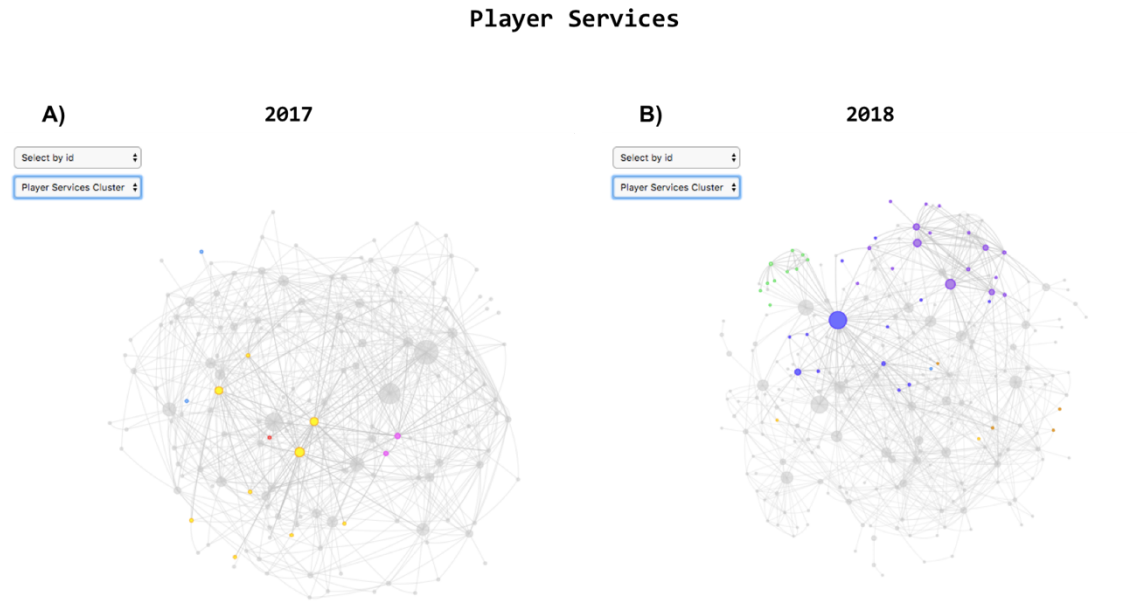


Figure 23: Networks highlighting the known player services business cluster collaborators. Colors represent the cluster membership of said nodes, as computed by the clustering algorithm. Node size represents betweenness.

4.6 Shiny App

To allow employees to interactively visualize the network and raise awareness about this project, a shiny app was created using the "shiny" (v1.2.0) R package (Chang, Cheng, Allaire, Xie, & McPherson, 2018).

The app contains three panels, showing the results available at the time of its creation which were all based on the weighted ABCDE networks:

1. the first one, with a short description of how the network was created and what it represents (Figure 24);
2. a second, containing a summary of each node measure and allowing a dynamic visualization of the networks. Choosing a measure (strength, degree, closeness, betweenness or local clustering coefficient) changes node size to represent the selected option. Furthermore, users can highlight nodes by username to visualize their own and their colleague's position in the network (Figure 25);
3. the third panel contains different tabs for each network measure, where their distribution is show, as are the results of statistical tests and tables with usernames scoring higher in those measures (Figure 26).

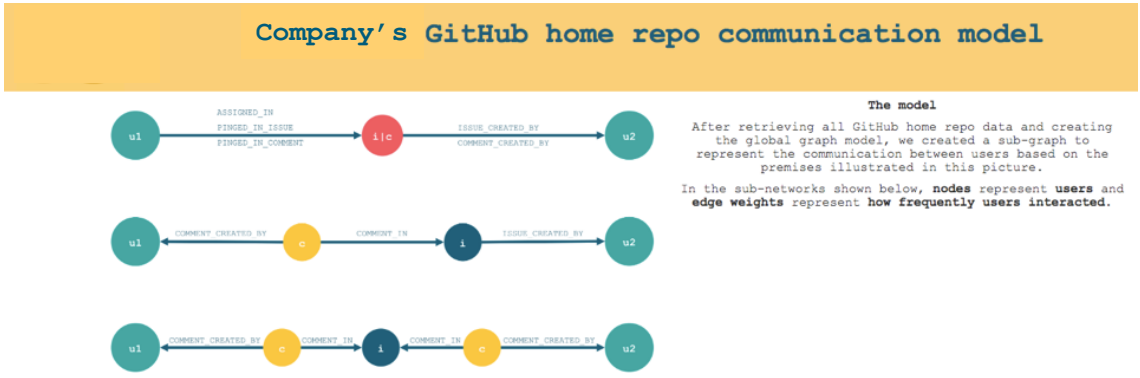


Figure 24: Screenshot of the first app panel.

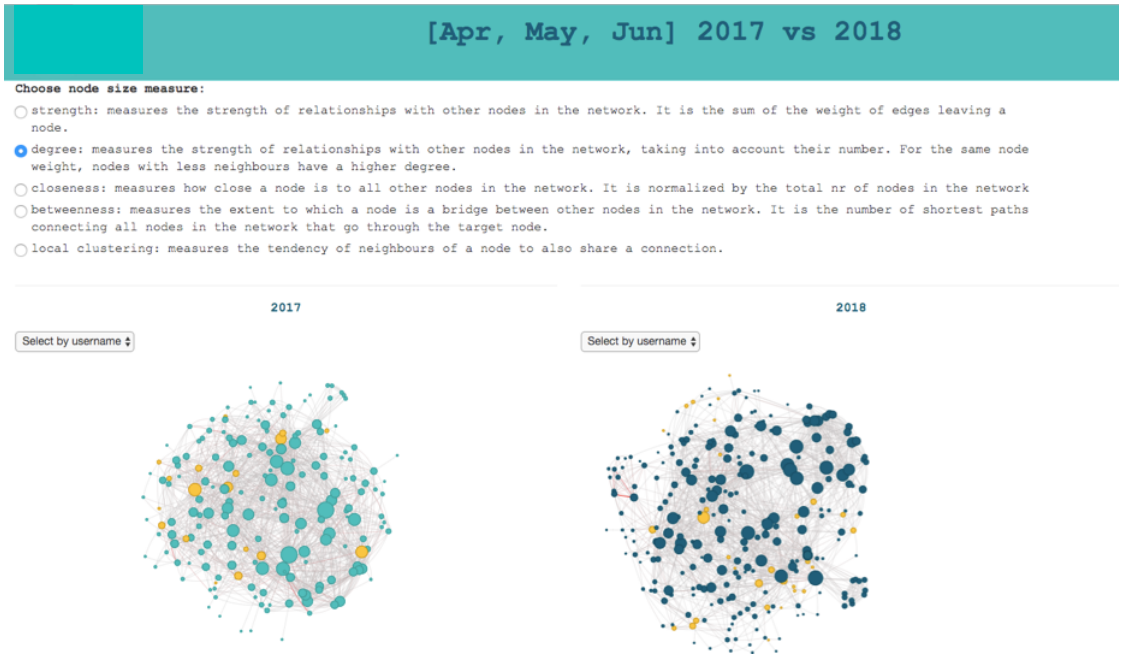


Figure 25: Screenshot of the second app panel. Yellow nodes correspond collaborators working remotely.

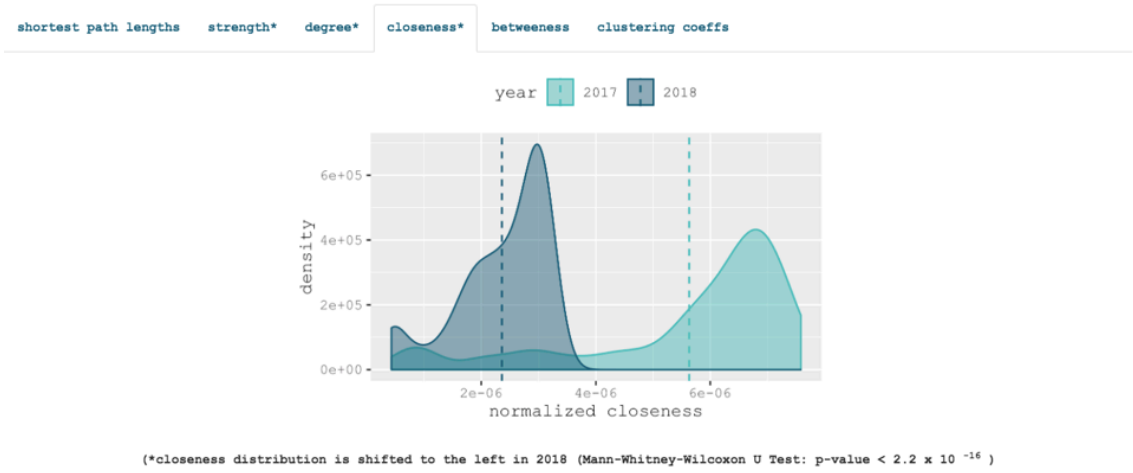


Figure 26: Screenshot of the third app panel. Tables containing username rankings are not shown for anonymity reasons.

Chapter 5 Conclusions

The main conclusions derived from this work are summarized in this chapter. Its limitations and considerations on work that could be done in the future are pinpointed.

5.1 Closing remarks

In this work, user-user communication networks were modeled from a company's GitHub data to analyze the impact a reorganizational change had on the way employees interact. Homologous data generated before (2017) and after (2018) this strategic move was used to derive and compare user communication patterns.

Both 2017 and 2018 networks reflect the internal organization model at the time. The overall topology of the user-user communication changed substantially after the change into smaller teams. Nodes are closer and have fewer connections but within more cohesive groups, showing a decrease in organization complexity.

Producers stand out as the most influential people in the network, able to promote knowledge sharing between communities. In both time periods no silos were observed, that is, no employees were found to be communicating between themselves and disconnected from the rest of the network. This work shows that the recent organizational changes did have an impact in the way people communicate.

Overall, the outcomes of this project add to the point that organizational network analysis is a powerful tool to evaluate performance and success of strategic changes, showing who employees are working with on their day-to-day.

5.2 Limitations and Future Work

One limitation of this work lays in node annotations, given that many nodes lacked job role and cluster affiliation (Figure 36, Annex I). When existent, this information referred to a single, most recent, timepoint that does not exactly match time when GitHub data was generated. So, the 2017 networks are not accurate regarding these annotations. This however is not critical, given that is unlikely that many people changed their job roles in that period.

Degree cutoff selection to generate unweighted networks could also be optimized, as this first approach was based solely on a visually informed guess.

As future work, besides the improvements above, it would be interesting to follow up with data for 2019.

Bibliography

Aggarwal, Karan, Hindle, Abram and Stroulia, Eleni (2014), Co-Evolution of Project Documentation and Popularity Within Github, 360–363, in: Hyderabad, India: ACM Press.

Allaho, Mohammad Y and Lee, Wang-Chien (2014), Trends and Behavior of Developers in Open Collaborative Software Projects. *2014 International Conference on Behavioral, Economic, and Socio-Cultural Computing (BESC2014)*: 1–7.

Badashian, A S, Esteki, A, Gholipour, A, Hindle, A and Stroulia, E (2014), Involvement, Contribution and Influence in Github and Stack Overflow, 1–15, in: Ontario, Canada.

Barabasi, AL and Albert, R (1999), Emergence of Scaling in Random Networks. *Science*, 286(5439): 509–512.

Barrat, A, Barthélemy, M, Pastor-Satorras, R and Vespignani, A (2004), The Architecture of Complex Weighted Networks. *Proceedings of the National Academy of Sciences*, 101(11): 3747–3752.

Bissyande, Tegawende F, Thung, Ferdian, Lo, David, Jiang, Lingxiao and Reveillere, Laurent (2013), Popularity, Interoperability, and Impact of Programming Languages in 100,000 Open Source Projects, 303–312, in: Kyoto, Japan: IEEE.

Blincoe, Kelly, Sheoran, Jyoti, Goggins, Sean, Petakovic, Eva and Damian, Daniela (2016), Understanding the Popular Users: Following, Affiliation Influence and Leadership on GitHub. *Information and Software Technology*, 70: 30–39.

Bondy, J A and Murty, USR (1976), *Graph Theory with Applications*. New York: Elsevier Science Publisching Co. Inc.

Borges, Hudson, Hora, Andre and Valente, Marco Tulio (2016), Understanding the Factors That Impact the Popularity of GitHub Repositories. *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*: 334–344.

Brandes, U (2001), A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, 25(2): 163–177.

Brandes, U and Fleischer, D (2005), Centrality Measures Based on Current Flow, 533–544

Brin, S and Page, L (1998), The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer networks and ISDN Systems*, 30: 107–117.

Bryan, L L and Joyce, C (2005), The 21st-Century Organization. *The McKinsey Quarterly*, 3: 1–75.

Burt, Ronald S (1984), Network Items and the General Social Survey. *Computer networks and ISDN Systems*, 6: 293–339.

Campbell, Ellsworth M, Jia, Hongwei, Shankar, Anupama, Hanson, Debra, Luo, Wei, Masciotra, Silvina, Owen, S Michele, Oster, Alexandra M, Galang, Romeo R, Spiller, Michael W, Blosser, Sara J, Chapman, Erika, Roseberry, Jeremy C, Gentry, Jessica, Pontones, Pamela, Duwve, Joan, Peyrani, Paula, Kagan, Ron M, Whitcomb, Jeannette M, Peters, Philip J, Heneine, Walid, Brooks, John T and Switzer, William M (2017), Detailed Transmission Network Analysis of a Large Opiate-Driven Outbreak of HIV Infection in the United States. *The Journal of Infectious Diseases*, 216(9): 1053–1062.

Casalnuovo, Casey, Vasilescu, Bogdan, Devanbu, Premkumar and Filkov, Vladimir (2015), Developer Onboarding in GitHub: the Role of Prior Social Links and Language Experience, 817–828, in: Bergamo, Italy: ACM.

Celińska, Dorota (2018), Coding Together in a Social Network: Collaboration Among GitHub Users, 31–40, in: Copenhagen, Denmark: ACM Press.

Chatr-aryamontri, Andrew, Oughtred, Rose, Boucher, Lorrie, Rust, Jennifer, Chang, Christie, Kolas, Nadine K, O'Donnell, Lara, Oster, Sara, Theesfeld, Chandra, Sellam, Adnane, Stark, Chris, Breitkreutz, Bobby-Joe, Dolinski, Kara and Tyers, Mike (2017), The BioGRID Interaction Database: 2017 Update. *Nucleic Acids Research*, 45(D1): D369–D379.

Christakis, Nicholas A and Fowler, James H (2007), The Spread of Obesity in a Large Social Network Over 32 Years. *New England Journal of Medicine*, 357(4): 370–379.

Csardi, G and Nepusz, T (2006), The Igraph Software Package for Complex Network Research. *Inter. J. Comp. Syst.*, 1695: 1–9. available at http://igraph.org

Dabbish, Laura, Stuart, Colleen and Herbsleb, Jim (2012), Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository, 1277–1286, in: Seattle, Washington, USA.

Dijkstra, EW (1959), A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1: 269–271.

Doreian, P (2002), Event Sequences as Generators of Social Network Evolution. *Social Networks*, 24(2): 93–119.

Ebel, Holger, Mielsch, Lutz-Ingo and Bornholdt, Stefan (2002), Scale-Free Topology of E-Mail Networks. *Physical Review E*, 66(3): 440–4.

Ediger, D, Jiang, K, Riedy, J, Bader, DA, Corley, C, Farber, R and Reynolds, W N (2010), Massive Social Network Analysis: Mining Twitter for Social Good,

Euler, Leonhard (1736), Solutio Problematis Ad Geometriam Situs Pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8: 128–140.

Fortunato, Santo and Castellano, Claudio (2009), Community Structure in Graphs, 490–512, in: *Encyclopedia of Complexity and Systems Science*. New York, NY: Springer, New York, NY.

Fortunato, Santo and Hric, Darko (2016), Community Detection in Networks: a User Guide. *Physics Reports*, 659: 1–44.

Freeman, L C (1978), Centrality in Social Networks Conceptual Clarification. *Social Networks*, 1: 215–239.

Ghadie, Mohamed A, Coulombe-Huntington, Jasmin and Xia, Yu (2018), Interactome Evolution: Insights From Genome-Wide Analyses of Protein–Protein Interactions. *Current Opinion in Structural Biology*, 50: 42–48.

Girvan, M and Newman, M E J (2002), Community Structure in Social and Biological Networks. *Proceedings of the National Academy of Sciences*, 99(12): 7821–7826.

Hu, Yan, Zhang, Jun, Bai, Xiaomei, Yu, Shuo and Yang, Zhuo (2016), Influence Analysis of Github Repositories. *SpringerPlus*: 1–19.

Huberman, Bernardo A (2001), *The Laws of the Web*. MIT Press.

Jarczyk, Oskar, Jaroszewicz, Szymon, Wierzbicki, Adam, Pawlak, Kamil and Jankowski-Lorek, Michal (2018), Surgical Teams on GitHub: Modeling Performance of GitHub Project Development Processes. *Information and Software Technology*, 100: 32–46.

Jurado, Francisco and Rodriguez, Pilar (2015), Sentiment Analysis in Monitoring Software Development Processes: an Exploratory Case Study on GitHub•S Project Issues. *The Journal of Systems & Software*, 104(C): 82–89.

Kalliamvakou, Eirini, Damian, Daniela, Blincoe, Kelly, Singer, Leif and German, Daniel M (2015), Open Source-Style Collaborative Development Practices in Commercial Projects Using GitHub. *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 1: 574–585.

Kalliamvakou, Eirini, Gousios, Georgios, Blincoe, Kelly, Singer, Leif, German, Daniel M and Damian, Daniela (2014), The Promises and Perils of Mining GitHub, 92–101, in: Hyderabad, India: ACM.

Kanehisa, Minoru, Furumichi, Miho, Tanabe, Mao, Sato, Yoko and Morishima, Kanae (2017), KEGG: New Perspectives on Genomes, Pathways, Diseases and Drugs. *Nucleic Acids Research*, 45(D1): D353–D361.

Kavaler, David, Sirovica, Sasha, Hellendoorn, Vincent, Aranovich, Raul and Filkov, Vladimir (2017), Perceived Language Complexity in GitHub Issue Discussions and Their Effect on Issue Resolution, 1–12, in: Urbana, IL, USA.

Klein, D J and Randić, M (1993), Resistance Distance. *Journal of Mathematical Chemistry*, 12(1): 81–95.

Kleinberg, Jon M (1999), Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM (JACM)*, 46(5): 604–632.

Lewis, Kevin, Kaufman, Jason, Gonzalez, Marco, Wimmer, Andreas and Christakis, Nicholas (2008), Tastes, Ties, and Time: a New Social Network Dataset Using Facebook.com. *Social Networks*, 30(4): 330–342.

Li, Huan and Zhang, Zhongzhi (2018), Kirchhoff Index as a Measure of Edge Centrality in Weighted Networks: Nearly Linear Time Algorithms,, in: New Orleans, Louisiana: Society for Industrial and Applied Mathematics.

Lima, A, Rossi, L and Musolesi, M (2014), Coding Together at Scale: GitHub as a Collaborative Social Network. *the 8th AAAI International Conference on Weblogs and Social Media*.

Loyola, Pablo and Ko, In-Young (2014), Population Dynamics in Open Source Communities: an Ecological Approach Applied to GitHub, 993–998, in: New York, New York, USA: ACM Press.

Luce, R Duncan and Perry, Albert D (1949), A Method of Matrix Analysis of Group Structure. *Psychometrika*, 14(1): 95–116.

Mate, Suzanne E, Kugelman, Jeffrey R, Nyenswah, Tolbert G, Ladner, Jason T, Wiley, Michael R, Cordier-Lassalle, Thierry, Christie, Athalia, Schroth, Gary P, Gross, Stephen M, Davies-Wayne, Gloria J, Shinde, Shivam A, Murugan, Ratnesh, Sieh, Sonpon B, Badio, Moses, Fakoli, Lawrence, Taweh, Fahn, de Wit, Emmie, van Doremalen, Neeltje, Munster, Vincent J, Pettitt, James, Prieto, Karla, Humrighouse, Ben W, Ströher, Ute, DiClaro, Joseph W, Hensley, Lisa E, Schoepp, Randal J, Safronetz, David, Fair, Joseph, Kuhn, Jens H, Blackley, David J, Laney, A Scott, Williams, Desmond E, Lo, Terrence, Gasasira, Alex, Nichol, Stuart T, Formenty, Pierre, Kateh, Francis N, De Cock, Kevin M, Bolay, Fatorma, Sanchez-Lockhart, Mariano and Palacios, Gustavo (2015), Molecular Evidence of Sexual Transmission of Ebola Virus. *New England Journal of Medicine*, 373(25): 2448–2454.

Newman, M E J (2001), Scientific Collaboration Networks. II. Shortest Paths, Weighted Networks, and Centrality. *Physical Review E*, 64(1): 0161321–0161327.

Newman, M E J (2004), Analysis of Weighted Networks. *Physical review. E*, 70(056131).

Newman, M E J and Girvan, M (2004), Finding and Evaluating Community Structure in Networks. *Physical review. E*, 69(2): 268–15.

Newman, M E J, Strogatz, S H and Watts, D J (2001), Random Graphs with Arbitrary Degree Distributions and Their Applications. *Physical Review E*, 64(2): 290–17.

Opsahl, Tore (2009), Structure and Evolution of Weighted Networks. : 1–151.

Opsahl, Tore and Panzarasa, Pietro (2009), Clustering in Weighted Networks. *Social Networks*, 31(2): 155–163.

Opsahl, Tore, Agneessens, Filip and Skvoretz, John (2010), Node Centrality in Weighted Networks: Generalizing Degree and Shortest Paths. *Social Networks*, 32(3): 245–251.

Opsahl, Tore, Colizza, Vittoria, Panzarasa, Pietro and Ramasco, José J (2008), Prominence and Control: the Weighted Rich-Club Effect. *Physical Review Letters*, 101(16): 161–4.

Page, L, Brin, S, Motwani, R and Winograd, T (1998), The PageRank Citation Ranking: Bringing Order to the Web, 161–172, in: Brisbane, Australia.

Peter, Isabelle and Davidson, Eric H (2015), *Genomic Control Process*. Academic Press.

Ray, Baishakhi, Posnett, Daryl, Filkov, Vladimir and Devanbu, Premkumar (2014), A Large Scale Study of Programming Languages and Code Quality in Github, 155–165, in: HongKong, China: ACM.

Robinson, Ian, Webber, Jim and Eifrem, Emil (2015), *Graph Databases* (M. Beaugureau, Ed.).

Russell, Matthew A (2013), *Mining the Social Web*. O'Reilly.

Saelens, Wouter, Cannoodt, Robrecht and Saeys, Yvan (2018), A Comprehensive Evaluation of Module Detection Methods for Gene Expression Data. *Nature Communications*, 9(1): 1090.

Tabassum, Shazia and Gama, João (2016), Evolution Analysis of Call Ego-Networks, 213–225, in: *Discovery Science*. Lecture Notes in Computer Science. Cham: Springer, Cham.

Teixeira, A S, Monteiro, P T, Carriço, JA, Ramirez, M and Francisco, A P (2013), Spanning Edge Betweenness, 27–31

Thung, Ferdian, Bissyande, Tegawende F, Lo, David and Jiang, Lingxiao (2013), Network Structure of Social Coding in GitHub. *the 17th European Conference on Software Maintenance and Reengineering*: 323–326.

Tsay, Jason, Dabbish, Laura and Herbsleb, James (2014), Influence of Social and Technical Factors for Evaluating Contribution in GitHub, 356–366, in: Hyderabad, India: ACM Press.

Watts, D J and Strogatz, S H (1998), Collective Dynamics of 'Small-World' Networks. *Nature*, 393(6684): 440–442.

Wellman, Barry (1979), The Community Question: the Intimate Networks of East Yorkers'. *American Journal of Sociology*, 84(5): 1201–1231.

White, Nicole RNeo4j. *https://github.com/nicolewhite/RNeo4j*. available at https://github.com/nicolewhite/RNeo4j [1 August 2018].

Woodward, G and Benstead, J P (2010), Ecological Networks in a Changing Climate. *Advances in Ecological Research*, 42: 71–138.

Yu, Yue, Yin, Gang, Wang, Huaimin and Wang, Tao (2014), Exploring the Patterns of Social Behavior in GitHub, 31–36, in: Hong Kong, China: ACM.

Zack, Michael H (2000), Researching Organizational Systems Using Social Network Analysis. *the 33rd Hawaii International Conference on System Sciences*: 7.

Appendix I

I. R Scripts

i. Querying Neo4j

Imports

```
library(RNeo4j)
```

Establishing connection to Neo4j

```
graph <- startGraph("http://localhost:7474/db/data")
```

Setting start and end dates for sub-networks

From April to June 2017

```
start <- as.numeric(as.POSIXct("2017-04-01", format="%Y-%m-%d", tz="UCT"))
end <- as.numeric(as.POSIXct("2017-06-30", format="%Y-%m-%d", tz="UCT"))
```

From April to June 2018

```
start <- as.numeric(as.POSIXct("2018-04-01", format="%Y-%m-%d", tz="UCT"))
end <- as.numeric(as.POSIXct("2018-06-30", format="%Y-%m-%d", tz="UCT"))
```

Defining queries

The following queries were executed to the Neo4j database, to retrieve 2017 and 2018 sub-networks. For each sub-network, "created_at" and "closed_at" parameters were set with the relevant timestamps above described. For simplicity, only the queries for sub-network 2017 are shown:

Relationship A: query q1

```
q1 <- "
MATCH p1=(u1:User)-[:ASSIGNED_IN]-(i:Issue)-[:ISSUE_CREATED_BY]-(u2:User)
USING SCAN u1:User
USING SCAN u2:User
WHERE u1.id < u2.id AND i.created_at >= 1491004800 AND i.closed_at <= 149878080
0
RETURN u1.username AS u1, u2.username AS u2, i.id AS issue, count(*) AS p1
"
```

Relationship B: query q2

```
q2 <- "
MATCH p2=(u1:User)-[:PINGED_IN_ISSUE]-(i:Issue)-[:ISSUE_CREATED_BY]-(u2:U
ser)
USING SCAN u1:User
```

```
USING SCAN u2:User
WHERE u1.id < u2.id AND i.created_at >= 1491004800 AND i.closed_at <= 149878080
0
RETURN u1.username AS u1, u2.username AS u2, i.id AS issue, count(*) AS p2
"
```

Relationship C: query q3

```
q3 <- "
MATCH p3=(u1:User)-[:PINGED_IN_COMMENT]-(c:Comment)-[:COMMENT_CRE
ATED_BY]-(u2:User)
USING SCAN u1:User
USING SCAN u2:User
WHERE u1.id < u2.id AND c.created_at >= 1491004800 AND c.created_at <= 1498780
800
RETURN u1.username AS u1, u2.username AS u2, c.id AS comment, count(*) AS p3
"
```

Relationship D: query q4

```
q4 <- "
MATCH p4=(u1:User)-[:COMMENT_CREATED_BY]-(c:Comment)-[:COMMENT_IN]
-(i:Issue)-[:ISSUE_CREATED_BY]-(u2:User)
USING SCAN u1:User
USING SCAN u2:User
WHERE u1.id < u2.id AND i.created_at >= 1491004800 AND i.closed_at <= 149878080
0
RETURN u1.username AS u1, u2.username AS u2, i.id AS issue, count(*) AS p4
"
```

Relationship E: query q5

```
q5 <- "
MATCH p5=(u1:User)-[:COMMENT_CREATED_BY]-(c1:Comment)-[:COMMENT_I
N]-(i:Issue)-[:COMMENT_IN]-(c2:Comment)-[:COMMENT_CREATED_BY]-(u2:User)
USING SCAN u1:User
USING SCAN u2:User
WHERE u1.id < u2.id AND i.created_at >= 1491004800 AND i.closed_at <= 149878080
0
RETURN u1.username AS u1, u2.username AS u2, i.id AS issue, count(DISTINCT c1) +
count(DISTINCT c2) AS p5
"
```

Database querying

```
edges1 <- cypher(graph, q1)
edges2 <- cypher(graph, q2)
edges3 <- cypher(graph, q3)
edges4 <- cypher(graph, q4)
edges5 <- cypher(graph, q5)
```

Save output files

```
write.csv(edges1, "queryOutputs/q1_2017.csv", row.names = F)
write.csv(edges2, "queryOutputs/q2_2017.csv", row.names = F)
write.csv(edges3, "queryOutputs/q3_2017.csv", row.names = F)
write.csv(edges4, "queryOutputs/q4_2017.csv", row.names = F)
write.csv(edges5, "queryOutputs/q5_2017.csv", row.names = F)
```

ii. Creating Relationship dataframes

Imports

```
library(dplyr)
```

Function sumRelationships.ABCDE

Reads the results of the separate relationship queries (A, B, C, D and E in the relations diagram) made to the neo4j database and compiles all the info into one single weight per relationship

@ args: edges1, edges2, edges3, edges4, edges5 - dataframes containing the output of the 5 relationship queries

@ return: rel2.df - dataframe with nodes pairs and edge weights

```
sumRelationships.ABCDE <- function(edges1, edges2, edges3, edges4, edges5){

  # get unique issues present in edges1, edges2, edges4 and edges5
  issues <- sort(unique(c(edges1$issue, edges2$issue, edges4$issue, edges5$issue)))
  nr.issues <- length(issues)

  # data frame to store the count of relationships for each user pair
  rel.df <- data.frame(stringsAsFactors=F)
  print("#### 1 ####")
  for (i in 1:nr.issues) {
    print(i/nr.issues)
    # for each issue, subset all data frames to get all relationships in that issue
    target.issue <- issues[i]  # i
    df1 <- subset(edges1, issue == target.issue)
    df2 <- subset(edges2, issue == target.issue)
    df4 <- subset(edges4, issue == target.issue)
    df5 <- subset(edges5, issue == target.issue)

    # get all pairs of users present in all relationships for that issue
    user.pairs <- unique(rbind(df1[,1:2],df2[,1:2],df4[,1:2],df5[,1:2]))
    nr.user.pairs <- nrow(user.pairs)

    for (j in 1:nr.user.pairs) {
      # for each pair of users in a relationship in that issue
      target.user.pair <- user.pairs[j,] # j
      # subset all edges dataframes for that pair of users and sum the nr of relationships
      rel1 <- df1[(df1$u1 == target.user.pair$u1 & df1$u2 == target.user.pair$u2),]
      rel2 <- df2[(df2$u1 == target.user.pair$u1 & df2$u2 == target.user.pair$u2),]
      rel4 <- df4[(df4$u1 == target.user.pair$u1 & df4$u2 == target.user.pair$u2),]
      rel5 <- df5[(df5$u1 == target.user.pair$u1 & df5$u2 == target.user.pair$u2),]
```

```
# to account for redundancies in rel4 and rel5
    if (dim(rel4)[1] != 0 & dim(rel5)[1] == 0) { # if rel4 is present and rel5 is absent, count only
rel4
      cnt <- sum(rel1$p1, rel2$p2, rel4$p4)
    } else if (dim(rel5)[1] != 0 & dim(rel4)[1] == 0){ # if rel5 is present and rel4 is absent, cou
nt only rel5
      cnt <- sum(rel1$p1, rel2$p2, rel5$p5)
    } else if (dim(rel4)[1] != 0 & dim(rel5)[1] != 0) { # if both rel4 and rel5 are present, count r
el5 + 1
      cnt <- sum(rel1$p1, rel2$p2, rel5$p5) + 1
    } else { # count only rel1 + rel2
      cnt <- sum(rel1$p1, rel2$p2)
    }

    temp <- cbind(target.user.pair, cnt)
    rel.df <- rbind(rel.df, temp)
  }
 }

 # for each pair of users, sum all their interactions and save them in a new data frame
 unique.pairs <- unique(rel.df[,1:2])
 rel2.df <- data.frame(stringsAsFactors=F)
 print("#### 2 ####")
 for (i in 1:nrow(unique.pairs)) {
  print(i/nrow(unique.pairs))
  target.user.pair <- unique.pairs[i,] # i
  temp <- rel.df[(rel.df$u1 == target.user.pair$u1 & rel.df$u2 == target.user.pair$u2),]
  cnt <- sum(temp$cnt)
  temp <- cbind(target.user.pair, cnt)
  rel2.df <- rbind(rel2.df, temp)
 }
 row.names(rel2.df) <- 1:nrow(rel2.df)
 nrow(rel2.df) == nrow(unique(rel2.df[,1:2]))
 print("#### 3 ####")
 for (k in 1:nrow(rel2.df)) {
  print(k/nrow(rel2.df))
  # subset the edges3 by user pair in the rel2.df and add the count of each user pair to rel2.df
  u1 <- rel2.df[k,1] # k
  u2 <- rel2.df[k,2] # k
  rel3 <- edges3[(edges3$u1 == u1 & edges3$u2 == u2),]
  rel2.df[k,3] <- rel2.df[k,3] + sum(rel3$p3) # k
 }
 # add to rel2.df the user pairs present in edges3 that are absent in rel2.df
 missing.pairs <- subset(edges3, !(edges3$u1 %in% rel2.df$u1 & edges3$u2 %in% rel2.df
$u2))
 missing.pairs <- missing.pairs[,c(1,2,4)]
 row.names(missing.pairs) <- 1:nrow(missing.pairs)
 colnames(missing.pairs) <- c("u1","u2","cnt")
```

```
  # for each pair of users, sum all their interactions and save them in a new data frame
  unique.pairs <- unique(missing.pairs[,1:2])
  pairs.to.add <- data.frame(stringsAsFactors=F)
  print("#### 4 ####")
  for (i in 1:nrow(unique.pairs)) {
    print(i/nrow(unique.pairs))
    target.user.pair <- unique.pairs[i,] # i
    temp <- missing.pairs[(missing.pairs$u1 == target.user.pair$u1 & missing.pairs$u2 == t
arget.user.pair$u2),]
    cnt <- sum(temp$cnt)
    temp <- cbind(target.user.pair, cnt)
    pairs.to.add <- rbind(pairs.to.add, temp)
  }
  row.names(pairs.to.add) <- 1:nrow(pairs.to.add)
  str(pairs.to.add)
  colnames(pairs.to.add) <- c("u1","u2","cnt")
  rel2.df <- rbind(rel2.df, pairs.to.add)

  return(rel2.df)
}
```

Function: sumRelationships.DE

Reads the results of the separate relationship queries (D and E in the relations diagram) made to the neo4j database and compiles all the info into one single weight per relationship

@ args: edges4, edges5 - dataframes containing the output of the 2 relationship queries

@ return: rel2.df – dataframe with nodes pairs and edge weights

```
sumRelationships.DE <- function(edges4, edges5){

  # get all issues present in edges1, edges3, edges4 and edges5
  issues <- sort(unique(c(edges4$issue, edges5$issue)))
  nr.issues <- length(issues)

  # data frame to store the count of relationships for each user pair
  rel.df <- data.frame(stringsAsFactors=F)
  print("#### 1 ####")
  for (i in 1:nr.issues) {
    print(i/nr.issues)
    # for each issue, subset all data frames to get all relationships in that issue
    target.issue <- issues[i]  # i
    df4 <- subset(edges4, issue == target.issue)
    df5 <- subset(edges5, issue == target.issue)

    # get all pairs of users present in all relationships for that issue
    user.pairs <- unique(rbind(df4[,1:2],df5[,1:2]))
    nr.user.pairs <- nrow(user.pairs)
```

```
for (j in 1:nr.user.pairs) {
    # for each pair of users in a relationship in that issue
    target.user.pair <- user.pairs[j,] # j
    # subset all edges dataframes for that pair of users and sum the nr of relationships
    rel4 <- df4[(df4$u1 == target.user.pair$u1 & df4$u2 == target.user.pair$u2),]
    rel5 <- df5[(df5$u1 == target.user.pair$u1 & df5$u2 == target.user.pair$u2),]

    # to account for redundancies in rel4 and rel5
    if (dim(rel4)[1] != 0 & dim(rel5)[1] == 0) { # if rel4 is present and rel5 is absent, count only
rel4
      cnt <- sum(rel4$p4)
    } else if (dim(rel5)[1] != 0 & dim(rel4)[1] == 0){ # if rel5 is present and rel4 is absent, cou
nt only rel5
      cnt <- sum(rel5$p5)
    } else if (dim(rel4)[1] != 0 & dim(rel5)[1] != 0) { # if both rel4 and rel5 are present, count r
el5 + 1
      cnt <- sum(rel5$p5) + 1
    }

    temp <- cbind(target.user.pair, cnt)
    rel.df <- rbind(rel.df, temp)
  }
 }

 # for each pair of users, sum all their interactions and save them in a new data frame
 unique.pairs <- unique(rel.df[,1:2])
 rel2.df <- data.frame(stringsAsFactors=F)
 print("#### 2 ####")
 for (i in 1:nrow(unique.pairs)) {
  print(i/nrow(unique.pairs))
  target.user.pair <- unique.pairs[i,] # i
  temp <- rel.df[(rel.df$u1 == target.user.pair$u1 & rel.df$u2 == target.user.pair$u2),]
  cnt <- sum(temp$cnt)

  temp <- cbind(target.user.pair, cnt)

  rel2.df <- rbind(rel2.df, temp)
 }
 row.names(rel2.df) <- 1:nrow(rel2.df)
 nrow(rel2.df) == nrow(unique(rel2.df[,1:2]))

 return(rel2.df)
}
```

Read edge dataframes

Read the edge dataframes created with script 4_queryNeo4j

```
edges1 <- read.csv("queryOutputs/q1_2017.csv", header = T, stringsAsFactors = F)
edges2 <- read.csv("queryOutputs/q2_2017.csv", header = T, stringsAsFactors = F)
```

```
edges3 <- read.csv("queryOutputs/q3_2017.csv", header = T, stringsAsFactors = F)
edges4 <- read.csv("queryOutputs/q4_2017.csv", header = T, stringsAsFactors = F)
edges5 <- read.csv("queryOutputs/q5_2017.csv", header = T, stringsAsFactors = F)
```

Run functions and save output

```
## run sumRelationships.ABCDE ----
relABCDE.df <- sumRelationships.ABCDE(edges1, edges2, edges3, edges4, edges5)
relABCDE.df$cnt.norm <- relABCDE.df$cnt/max(relABCDE.df$cnt) # add normalized we
ight column
nrow(relABCDE.df) == nrow(unique(relABCDE.df[,1:2])) # check all node pairs are unique
write.csv(relABCDE.df,"queryOutputs/subNets/2017_RelationshipsABCDE.csv", row.n
ames = F)


## run sumRelationships.DE ----
relDE.df <- sumRelationships.DE(edges4, edges5)
relDE.df$cnt.norm <- relDE.df$cnt/max(relDE.df$cnt) # add normalized weight column
nrow(relDE.df) == nrow(unique(relDE.df[,1:2])) # check all node pairs are unique
write.csv(relDE.df,"queryOutputs/subNets/2017_RelationshipsDE.csv", row.names = F)
```

II. Network Measure Plots

i. Weighted Networks

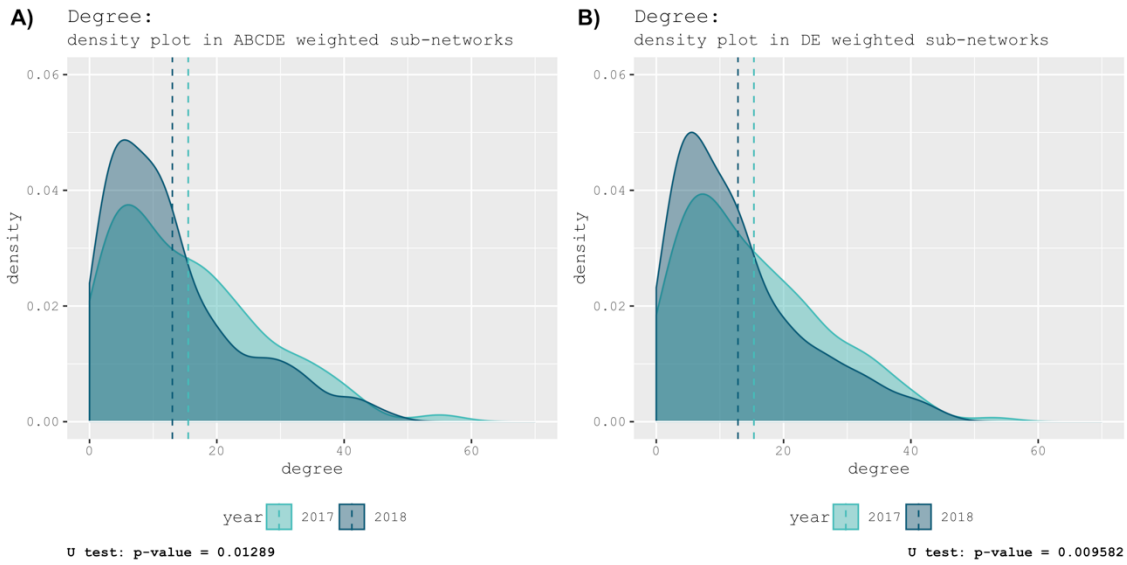


Figure 27: Node degree distribution in weighted A) ABCDE and B) DE relationship models.
Dotted lines (---) indicate mean values.

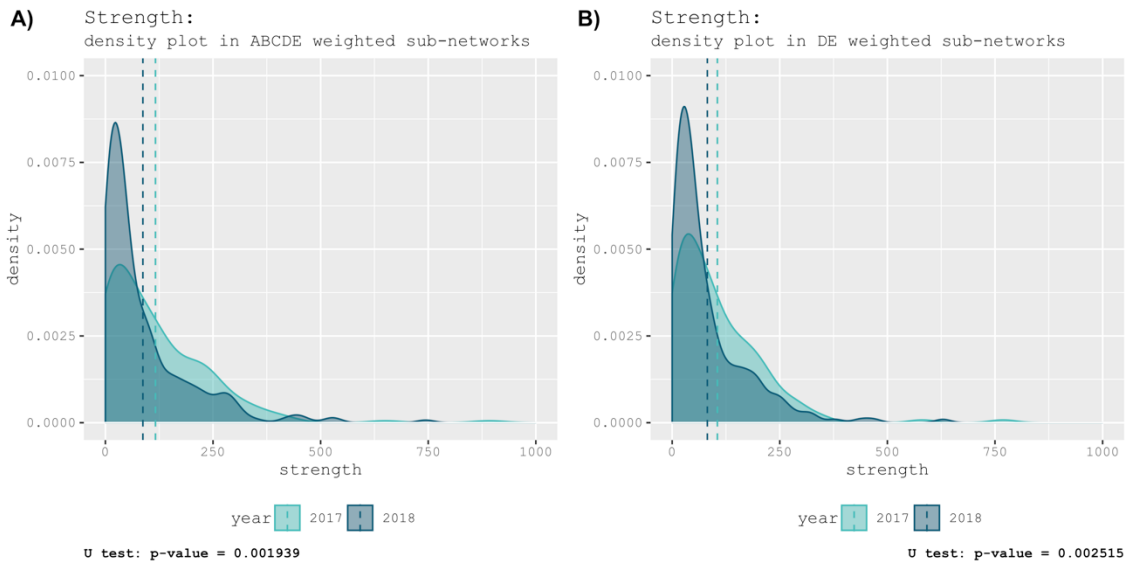


Figure 28: Node strength distribution in weighted A) ABCDE and B) DE relationship models.
Dotted lines (---) indicate mean values.

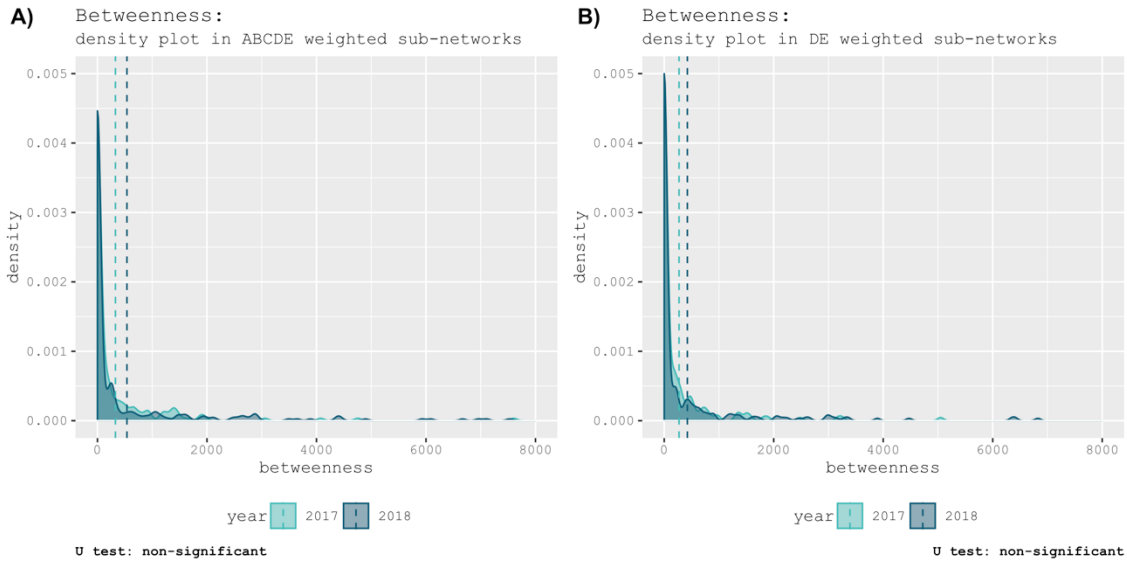


Figure 29: Node betweenness distribution in weighted A) ABCDE and B) DE relationship models. Dotted lines (---) indicate mean values.

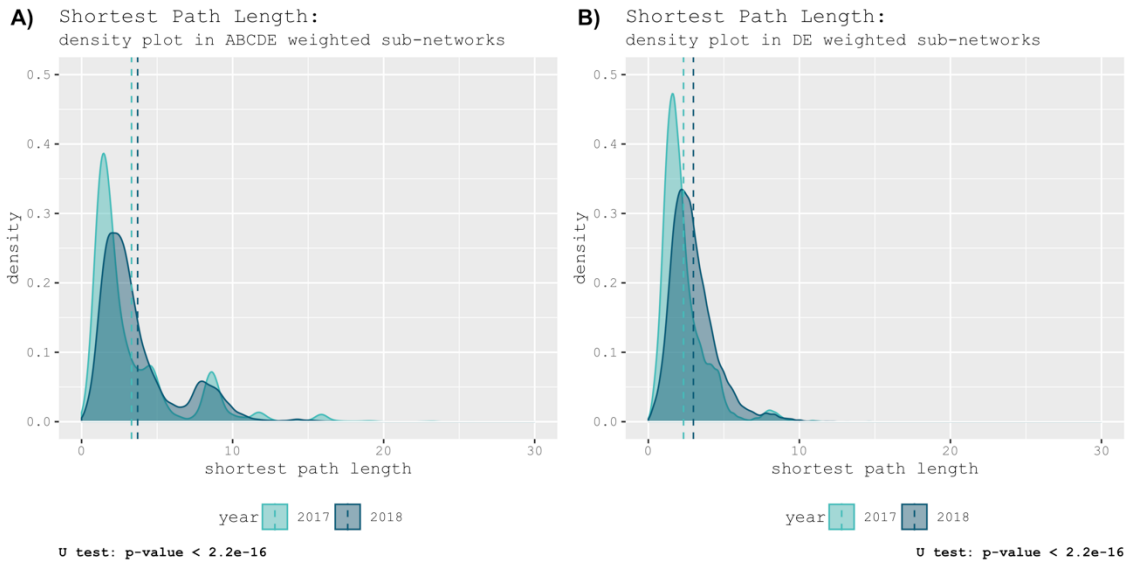


Figure 30: Shortest path length distribution in weighted A) ABCDE and B) DE relationship models. Dotted lines (---) indicate mean values.

ii. Unweighted Networks

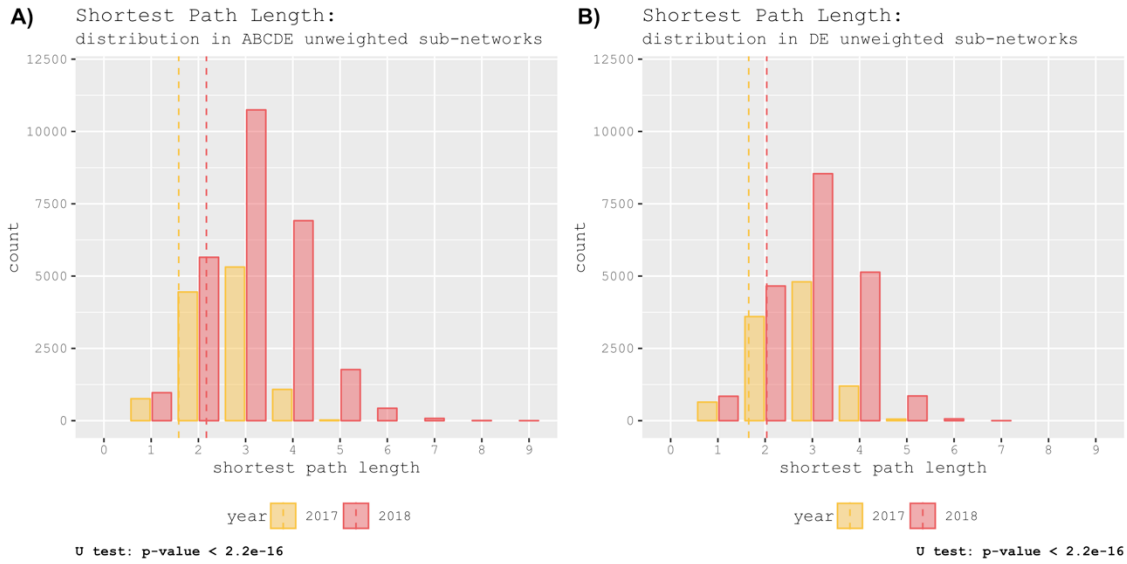


Figure 31: Shortest path length distribution in unweighted A) ABCDE and B) DE relationship models. Dotted lines (---) indicate mean values.

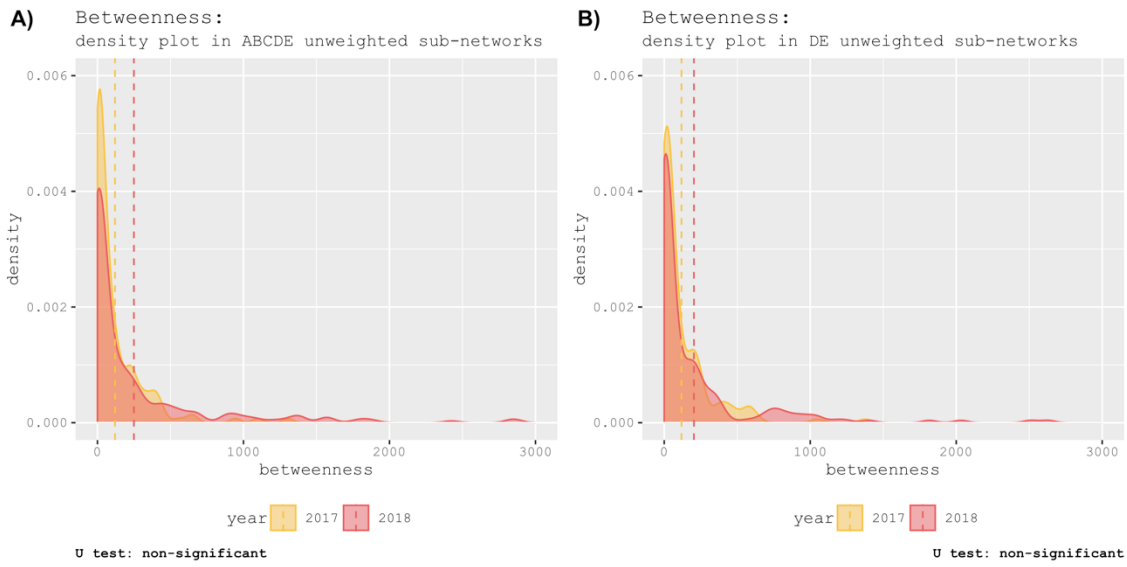


Figure 32: Node betweenness distribution in unweighted A) ABCDE and B) DE relationship models. Dotted lines (---) indicate mean values.

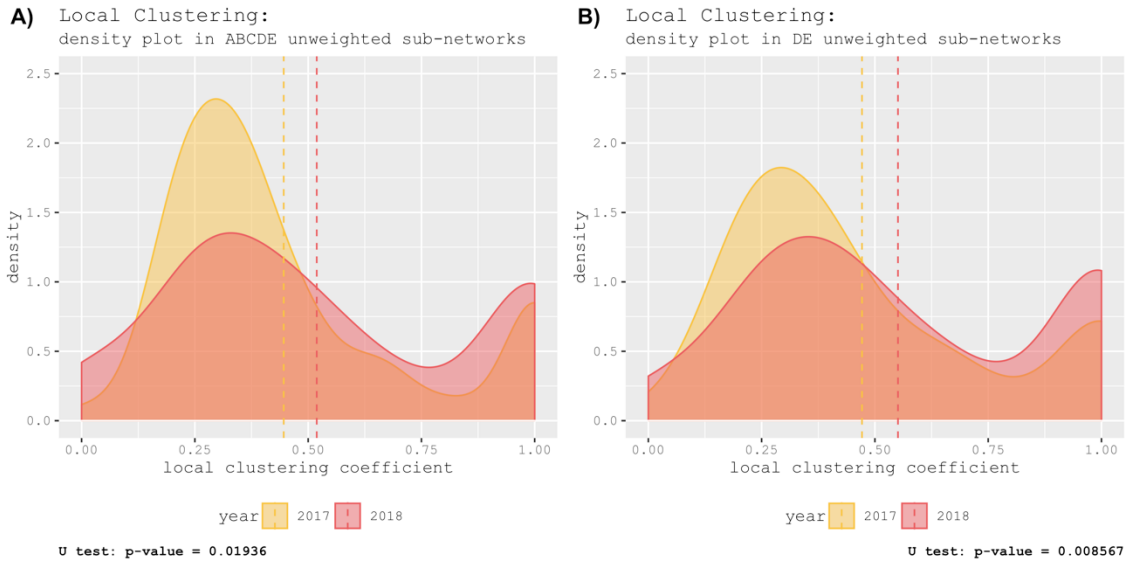


Figure 33: Node local clustering coefficient distribution in unweighted A) ABCDE and B) DE relationship models. Dotted lines (---) indicate mean values.

III. Relevant network nodes

Table 23: Nodes in 2017 ABCDE unweighted network with clustering coefficient of one.

userID	local.cc	role	cluster
2466081	1.00	Chief Executive Officer	Senior Leadership
5712887	1.00		
2032603	1.00		
1589650	1.00		
12230715	1.00	Backend Lead	Casino Product Cluster
42919	1.00		
22445508	1.00		
73201	1.00	Software Engineer - Frontend	Casino Product Cluster
7986682	1.00		
1449282	1.00	Founder	Senior Leadership
2591677	1.00		
14107554	1.00		
20901033	1.00	Deputy MLRO Fraud and AML Coach	Player Services Cluster
9280378	1.00	Systems Engineer	Platform Cluster
26111720	1.00		
25659099	1.00	Payments and Risk Analyst	Player Services Cluster
25148998	1.00		
26866981	1.00	Ambassador	Player Services Cluster
26735702	1.00	Content Creative	Marketing Cluster
15886074	1.00		
26273591	0.83		
25583966	0.83	Finance Officer	Business Cluster
16539576	0.80		
7534114	0.71	Content Creative	Employee Experience Cluster
265696	0.70		

Table 24: Nodes in 2018 ABCDE unweighted network with clustering coefficient of one.

userID	local.cc	role	cluster
8300431	1.00		
499393	1.00		
8088699	1.00		
5417763	1.00	Developer	Casino Product Cluster
2535897	1.00		
18258556	1.00	People Lead	Employee Experience Cluster
31698349	1.00		
19647185	1.00		
34026936	1.00		
29751649	1.00	Ambassador	Player Services Cluster
20901033	1.00	Deputy MLRO Fraud and AML Coach	Player Services Cluster
26273591	1.00		
13994611	1.00	Business Strategist	Business Cluster
26735702	1.00	Content Creative	Marketing Cluster
4668225	1.00		
6874488	1.00	Data Analyst	Casino Product Cluster
29477667	1.00	Responsible Gaming Specialist	Player Services Cluster
4331581	1.00	Software Engineer - Frontend	Casino Product Cluster
7986682	1.00		
25957005	1.00	Payments and Risk Analyst	Player Services Cluster
30112680	1.00	Fraud & AML Analyst	Player Services Cluster
25659099	1.00	Payments and Risk Analyst	Player Services Cluster
36295610	1.00	Payment Coach	Player Services Cluster
38035749	1.00	Ambassador	Player Services Cluster
37145525	1.00		
35920724	1.00	Games Artist	Games Cluster
28043486	1.00		
38034530	1.00		
26866981	1.00	Ambassador	Player Services Cluster
31698352	1.00	Ambassador	Player Services Cluster
38315501	1.00	Loyalty Officer	Player Services Cluster
37145567	1.00	Ambassador	Player Services Cluster
36196397	1.00		
36196358	1.00		
37145728	1.00	Ambassador	Player Services Cluster
37074035	1.00	Chief People Officer	Senior Leadership
35033967	1.00	Chef	Employee Experience Cluster
30406780	1.00	Loyalty Officer	Player Services Cluster
38423475	1.00	Designer	Casino Product Cluster
39982653	1.00		
35454250	1.00	BI Analyst	Business Cluster
36952711	1.00	Payments and Risk Analyst	Player Services Cluster
36952709	1.00	Payments and Risk Analyst	Player Services Cluster
30112721	1.00		
25957026	1.00	Payments and Risk Analyst	Player Services Cluster
38347260	1.00	Payments and Risk Analyst	Player Services Cluster
23311312	1.00		
39056832	1.00		
1449282	0.90	Founder	Senior Leadership
32569158	0.90	Cloud Engineer	Games Cluster
5712887	0.86		
25640042	0.86		
12693760	0.83	Payment Product Lead	Player Services Cluster
8812990	0.83	Software Engineer - Backend	Casino Product Cluster
33220402	0.80	PPC Analyst	Marketing Cluster

IV. Network plots

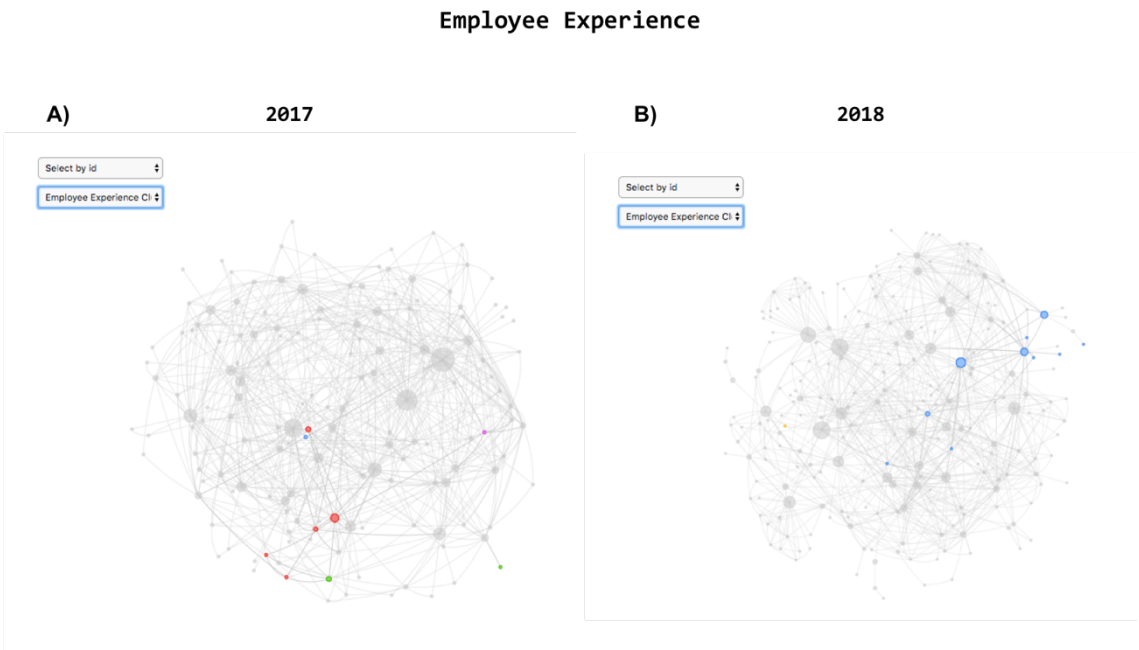


Figure 34: Networks highlighting the known employee experience business cluster collaborators. Colors represent the cluster membership of said nodes, as computed by the clustering algorithm. Node size represents betweenness.

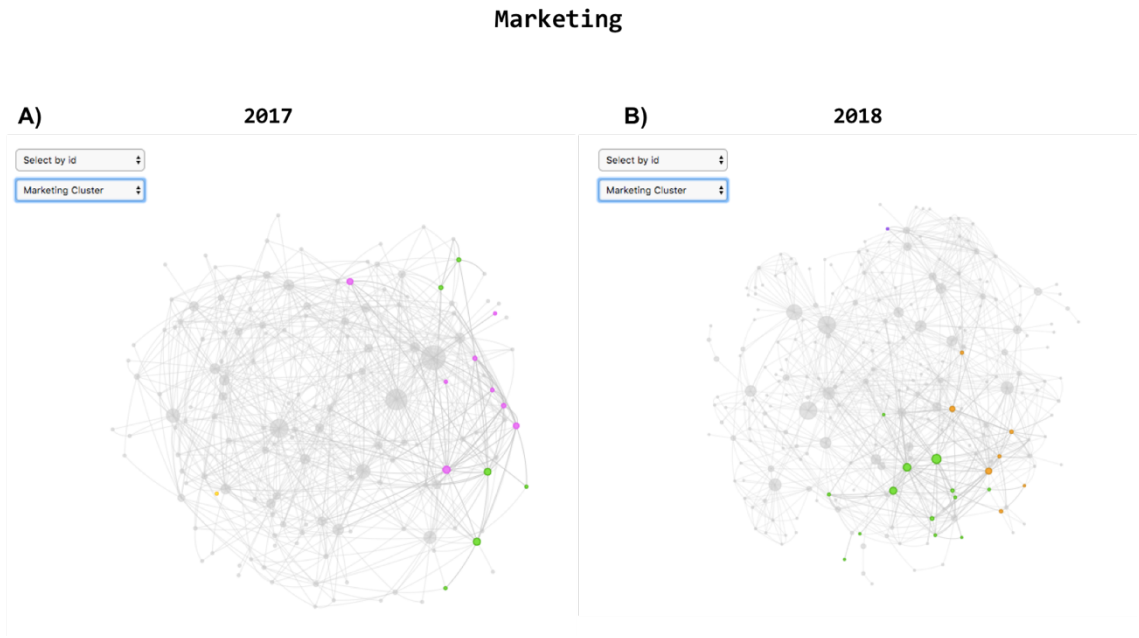


Figure 35: Networks highlighting the known marketing business cluster collaborators. Colors represent the cluster membership of said nodes, as computed by the clustering algorithm. Node size represents betweenness.

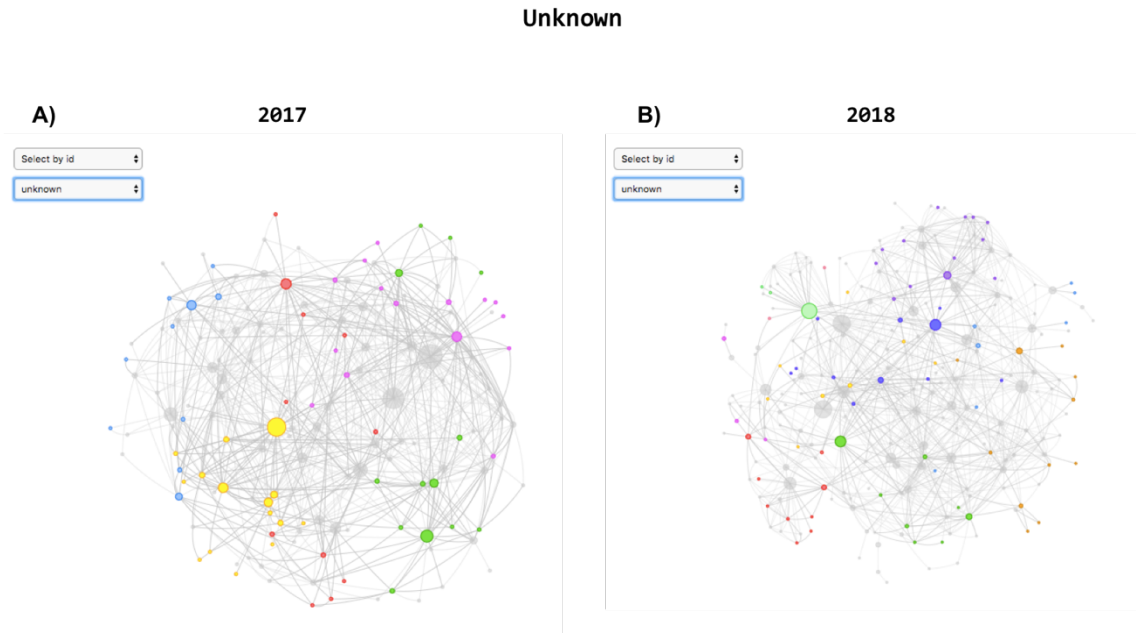


Figure 36: Networks highlighting collaborators for which the business cluster is not known. Colors represent the cluster membership of said nodes, as computed by the clustering algorithm. Node size represents betweenness.