Assignment Social Network Analysis

Meta Infos	
Student	Jonathan Baß
Titel	Hier könnte der Titel ihrer Arbeit stehen!
Kurskontext	Social Network Analysis
Datum	18.11.2021
Dozent	Philipp M. Mendoza, M.Sc.

Wichtig: das ganze sollte in der Form eines Blogpostes geschrieben sein; sprich ein Fließtext! Nachfolgend ein Vorschlag der Strukturierung eurer Arbeit; in jedem Report sollten *zumindest* die hier angeführten Punkte abgedeckt werden.

Executive Summary

Dies ist die einzige Sektzion die in Bullet points angeführt werden soll. * Einleitung * Forschungsfrage * Datensatz * Strategie * Ergebnisse

Einleitung und Fragestellung

Die Analyse und Nachverfolgung von Kontaktdaten ist in der Medizin und Forschung schon lange ein wichtiges Thema, momentan durch die aktuelle Corona-Pandemie jedoch relevanter als je zuvor. Bei engen oder langen Kontaken können Keime und Erreger zwischen den Personen ausgetauscht werden und sich Krankheiten auf diesem Weg verbreiten. Dies stellt im Alltag kein großes Problem dar, da die meisten Erreger harmlos sind und das menschliche Immunsystem den Ausbruch der Krankheit verhindern kann. Dies ist jedoch in Krankenhäusern nicht immer der Fall. Durch die hohe Konzentration an angewendeten Medikamenten wie Antibiotika können sich schnell multiressistente Keime bilden, welche auch Krankenhauskeime genannt werden. Eine detaillierte Beschreibung und Quantifizierung der Kontakte in Krankenhäusern kann deshalb wichtige Informationen für die Epidemiologie von Krankenhausinfektionen sowie für die Konzeption und Validierung von Kontrollmaßnahmen liefern. Wie diese Vorgenommen werden und welche Schlüsse aus der Analyse gezogen werden können wir in dem folgenden Beitrag erläutert.

Thema

In der geriatrigerschen Abteilung eines Krankenhauses in Lyon, Frankreich wurde 2010 ein Experiment gemacht. Dafür wurden allen sich auf der Station befindlichen Personen mit tragbare RFID (Radio Frequency Identification) Sensoren ausgerüstet. Diese haben alle Interaktionen im Nahbereich von etwa 1,5m und einer zeitlichen Auflösung von 20 Sekunden über einem Zeitraum von vier Tagen und vier Nächten gemessen. Die Studie umfasste 46 Mitarbeiter des Gesundheitswesens und 29 Patienten und es wurden insgesamt 14.037 Kontakte erfasst.

Daten

Die Daten liegen als ein igraph graph-Objekt vor, welches die graph-Attribute "name" und "Citation", als vertex-Attribut "Status" und als edge-Attribut "Time" besitzt. "Status" beschreibt dabei die Rolle der Person. Dabei wird in Verwaltungspersonal (ADM), Ärzte (MED), medizinisches Personal wie Krankenschwestern und -pleger oder Hilfspersonal (NUR) und Patienten/Patientinnen (PAT) unterschieden. "Time" ist der Zeitstempel der Sekunde, an dem das 20-Sekunden Intervall ausgelaufen und nicht erneuert wurde.

Forschungsfrage

Die Übertragung von Krankheiten kann unter anderem durch persönlichen Kontakt übertragen werden. Personen, die viele Kontakte mit einer Vielzahl an Personen haben, können dabei als Superspreader fungieren und Krankheiten schnell verteilen. **In der folgenden Ausarbeitung beschäftige ich mich mit der Frage, ob gewisse Rollen in Krankenhäusern ein höheres Risiko haben solch ein Superspreader zu sein und ob die durch organisatorische und räumlich gegebene Bildung von Gemeinschaften einen Einfluss auf eine mögliche Verbereitung hat.**

Relevanz der Forschungsfrage

Die Kontaktnachverfolgung ist durch die aktuelle Corona-Pandemie ein wichtigeres Thema denn je. Es gilt geschwächte und anfällige Personen zu schützen. Gerade in Krankenhäusern kann eine unkontrollierte Ausrbeitung des SarsCov2-Viruses durch einen Superspreader verherende Auswirkungen haben. Die Analyse und Beantwortung der Forschungsfrage kann bei der Entwicklung eines Schutzkonzeptes helfen und das allgemeine Risiko einer Ansteckung während eines Krankenhausaufenthaltes verringern.

Analysestrategie

Um die Forschungsfrage korrekt und vollständig beantworten zu können, reicht es nicht aus die reinen Verbindungen zu zählen. Sollte zum Beispiel ein Patient viel Kontakt mit anderen (infizierten) Patienten haben, würde dadurch keine Übertragung und damit auch keine Ausbreitung der Krankheit erfolgen. Aus diesem Grund muss man die Rollen der einzelnen Personen bei der Bewertung des Kontaktes berücksichtigen. Dafür können zwei Werte für jede Node berechnen: Der "Within-community Degree"-Wert (deutsch: Innergemeinschaftlicher Grad) beschreibt die Vernetzung innerhalb der eigenen Community. Dazu wird der "Participation Coefficient" (deutsch: Teilnahmekoeffizient) betrachtet. Dieser beschreibt die Vernetzung einer Person über die Grenzen der eigenen Community heraus. Für jede Node können beide Werte berechnet und auf einem Diagramm als Punkte ausgeplottet werden. Sollte zum Beispiel eine Person eine sehr gute Vernetzung innerhalb der eigenen Community, aber auch zu anderen Communites haben, ist das Risiko bei einer Infektion als Superspreader zu fungierten stark erhöht. Für die Berechnung dieser Schlüselwerte wird der Algorithmus "Netcarto" verwendet. Rnectcarto bietet schnelle Netzwerkmodularität und Rollenberechnung durch simuliertes Annealing. Die Ergebnisse werden dann in dem zweiten Teil als Edge-Data dem Netzwerk zugeführt und visualisiert.

Charakterisierung des Netzwerks

Der Datensatz bildet die Interaktionen innerhalb der Station über einen Zeitraum von 96 Stunden ab. Die 75 Nodes (Vertecies) bilden die Teilnehmer des Experiments ab. Dabei wird nicht weiter in Patienten oder Angestellte unterschieden. Die Node-Daten beinhalten jeweils den Status bzw. die Rolle der Person, welche im Punkt "Daten" bereits beschrieben sind. Die Edges (Links) repräsentieren die Kontakte der Personen untereinander. Je öfter zwei Personen kontakt miteinander hatten, desto mehr Edges gibt es zwischen ihnen. Da die Links nur den reinen Kontakt symbolisieren sind sie nicht gerichtet (directed) und zeigen nur, dass ein engerer Kontakt stattgefunden hat. Aus diesem Grund findet auch keine gewichtung (weighted) statt.

Aus diesem Grund können für die reine Analyse alle Schleifen entfernt werden, da bereits ein Kontakt für eine potentielle Übertragung ausreicht. Zudem werden nicht benötigte Daten entfernt und die Berechnungen dadurch schneller. Zusammenfassend lässt sich sagen, dass es sich um ein One-Node-Netzwerk handelt, welches weder eine Richtung noch eine Gewichung in den Nodes besitzt.

Um die Forschungsfrage zu beantworten werden mithilfe der Degree-Zentralität die Personen ermittelt, welche die meisten Kontakte mit anderen haben. Dies zeigt, wie viele Freunde die Personen angegeben haben und wie zentral dadurch eine Person im Netzwerk angeordnet ist. Um herauszufinden, welche der Beziehungen auch wirklich beidseitig sind, also von zwei Personen gleichermaßen ausgehen, dient die Visualisierung mit ungerichteten Kanten. Durch die anschließende Berechnung der Anzahl der Kanten kann ermittelt werden, wie viele einseitige Freundschaften dadurch wegfallen. Es ergibt sich ein guter Überblick über die Universität für die Beantwortung der Forschungsfrage. Um vom Allgemeinen, der Universität, etwas spezifischer zu werden, werden die Fakultäten einzeln betrachtet. Damit lässt sich die Anzahl der Personen in einer Fakultät und die Person mit der höchsten Degree-Zentralität ermitteln. Die Forschungsfrage lässt sich weiter beantworten und es lassen sich nun spezifischere Rückschlüsse ziehen.

- Kontext der verwendeten Daten Anzahl nodes and edges ## Umsetzung

Datenmanipulationen

Erklärung der Schritte

Die folgende Auflistung beschreibt die einzelnen Schritte, welche für die Auswertung verwendet wurden.Diese könnnen chronologisch ausgeführt werden, um das gleiche Ergebniss zu erhalten.

1. Umgebung Vorbereiten

Zu Beginn müssen alle benötigten Pakete installiert werden. Darunter befinden sich unter anderem “igraph-data”, aus welchem wir die Daten laden, standart-Pakete wie “ggraph”. Zudem benötigen wir den oben erwähnten Annealing-Algorithmus “rnetcarto”.

```
# install.packages("igraph", dependencies = T)
# install.packages("igraphdata", dependencies = T)
# install.packages("tidygraph", dependencies = T)
# install.packages("tidyverse", dependencies = T)
# install.packages("ggraph", dependencies = T)
# install.packages("GGally", dependencies = T)
# install.packages("ggthemes", dependencies = TRUE)
# install.packages("gganimate", dependencies = TRUE)
# install.packages("gifski", dependencies = TRUE)
# install.packages("rnetcarto", dependencies = TRUE)
```

Nun werden die installierten Pakete werden als Bibliotheken deklariert und damit in der Umgebung auch instanziiert.

```
library(igraph)
library(igraphdata)

library(ggraph)
library(ggthemes)
```

```
library(gganimate)
library(GGally)
library(gifski)

library(tidyverse)
library(tidygraph)

library(RColorBrewer)
library(rnetcarto)
```

Damit keine Konflikte mit alten Daten auftreten, und die Berechnungen möglichst perfomant laufen können, werden alle alten Daten aus dem Environment gelöscht.

```
rm(list = ls())
```

2. Daten für den Algorithmus vorbereiten

Nun instanziieren wir die Daten mit dem folgenden Befehl. Es existiert nun ein igraph graph-Objekt "rfid", welches die benötigten Daten beinhaltet. Dieses Objekt bildet die Grunlage für die folgenden Berechnungen.

```
data("rfid")
is_directed(rfid)
```

```
## [1] FALSE
```

```
is_weighted(rfid)
```

```
## [1] FALSE
```

```
gsize(rfid)
```

```
## [1] 32424
```

```
gorder(rfid)
```

```
## [1] 75
```

Danach werden Schleifen entfernt und direktionale Beziehungen in inderktionale Beziehungen aufgelöst. Dadurch werden Fehler in den Darstellungen vermieden.

```
df <- as.undirected(simplify(rfid))
gsize(df)
```

```
## [1] 1139
```

```
gorder(df)
```

```
## [1] 75
```

Damit der Netrcarto-Algorithmus die Daten verarbeiten kann müssen diese als Adjazenzmatrix vorliegen. Der folgende Befehl konvertiert das Dateframe in dieses Format speichert die neue Matrix als Vairiation der Variable df.

```
df.mat=as_adjacency_matrix(df, sparse = F)
head(df.mat, 1)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
## [1,]    0    1    1    1    1    1    1    1    1     1     1     1     1     1
##      [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
## [1,]     1     1     1     1     1     1     1     1     1     1     1     1
##      [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37] [,38]
## [1,]     1     1     1     1     1     0     1     0     1     1     1     0
##      [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48] [,49] [,50]
## [1,]     0     1     1     1     1     0     1     1     1     1     1     0
##      [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60] [,61] [,62]
## [1,]     1     1     1     1     1     0     0     1     0     1     0     1
##      [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72] [,73] [,74]
## [1,]     1     1     1     0     1     1     1     0     1     1     1     1
##      [,75]
## [1,]     0
```

Wie hier ersichtlich besteht die Matrix aus einem zweidimensionalen Tabelle, wobei die Spalten die verschiedenen Nodes darstellen und die Spalten die Nodes (Hier nur eine Node gezeigt).

3. Simulierter annealing Algorithmus

Nun kann der Netcarto-Algorithmus angewandt werden. Dieser symuliert ein sogenanntes “annealing” oder auf Deutsch “abkühlen”, und sich auf die Art und Weiße bezieht, wie die einzelnen Communities entstehen. Eine gute Analogie ist das Abkühlen von Metall, nachdem es zum glühen gebracht wurden. Zu diesem Zeitpunkt bewegen sich die einzelnen Atome noch sehr schnell - erkennbar an der Hitze- und Lichtemision. Im verlauf des Abkühlprozesses ordnen sich die Atome in einer kristallinen Form an und es wird ein nahezu optimaler Zustand erreicht. Zieht man nun die Verbindungen zu dem Algorithmus würde so wird das Netzwerk in einen ähnlichen Zustand gebracht und es können einzelne Module mit einer Wahrscheinlichkeit von bis zu 90% berechnet werden.

```
rnc=netcarto(df.mat)
head(rnc[[1]], 5)
```

```
##    name module connectivity participation      role
## 67   67      0   -1.8812090     0.6938776 Connector
## 47   47      0   -1.2388450     0.7218935 Connector
## 68   68      0   -1.2388450     0.7448015 Connector
## 50   50      0   -0.5964809     0.6848073 Connector
## 41   41      0   -0.5964809     0.7066116 Connector
```

4. Daten für den ersten Plot vorbereiten

Es liegt nun ein Dataframe mit den zu Beginn definierten Werten vor, welche auf das Modul und die Rolle beinhalten. Leider ist jedoch beim Erstellen der Adjacency-Matrix die Rolle verloren gegangen, da es sich bei der Spalte nicht um nummerische Werte handelt. Daher müssen die beiden Dataframes vereint werden. Dafür wird dem neu erstellten Dataframe eine Spalte mit dem Index hinzugefügt, um im nächsten Schritt einen merge durchführen zu können. Da diese bei dem ursprünglichen DF "name" heißt, wurd hier die gleiche Bezeichnung verwendet.

```
df2 <- df %>% as_tbl_graph() %>% activate(nodes) %>% mutate(name = row_number())
df2
```

```
## # A tbl_graph: 75 nodes and 1139 edges
## #
## # An undirected simple graph with 1 component
## #
## # Node Data: 75 x 2 (active)
##   Status  name
##   <chr>  <int>
## 1 ADM        1
## 2 NUR        2
## 3 NUR        3
## 4 NUR        4
## 5 NUR        5
## 6 NUR        6
## # ... with 69 more rows
## #
## # Edge Data: 1,139 x 2
##    from    to
##   <int> <int>
## 1     1     2
## 2     1     3
## 3     1     4
## # ... with 1,136 more rows
```

Durch das Hinzufügen der Index-Spalte kann nun einen Merge durchgeführt werden. Das Dataframe "dfPlot" beinhaltet jetzt alle benötigten Werten, welche für die Auswertung relevant sind und kann in einem Plot visualisiert werden.

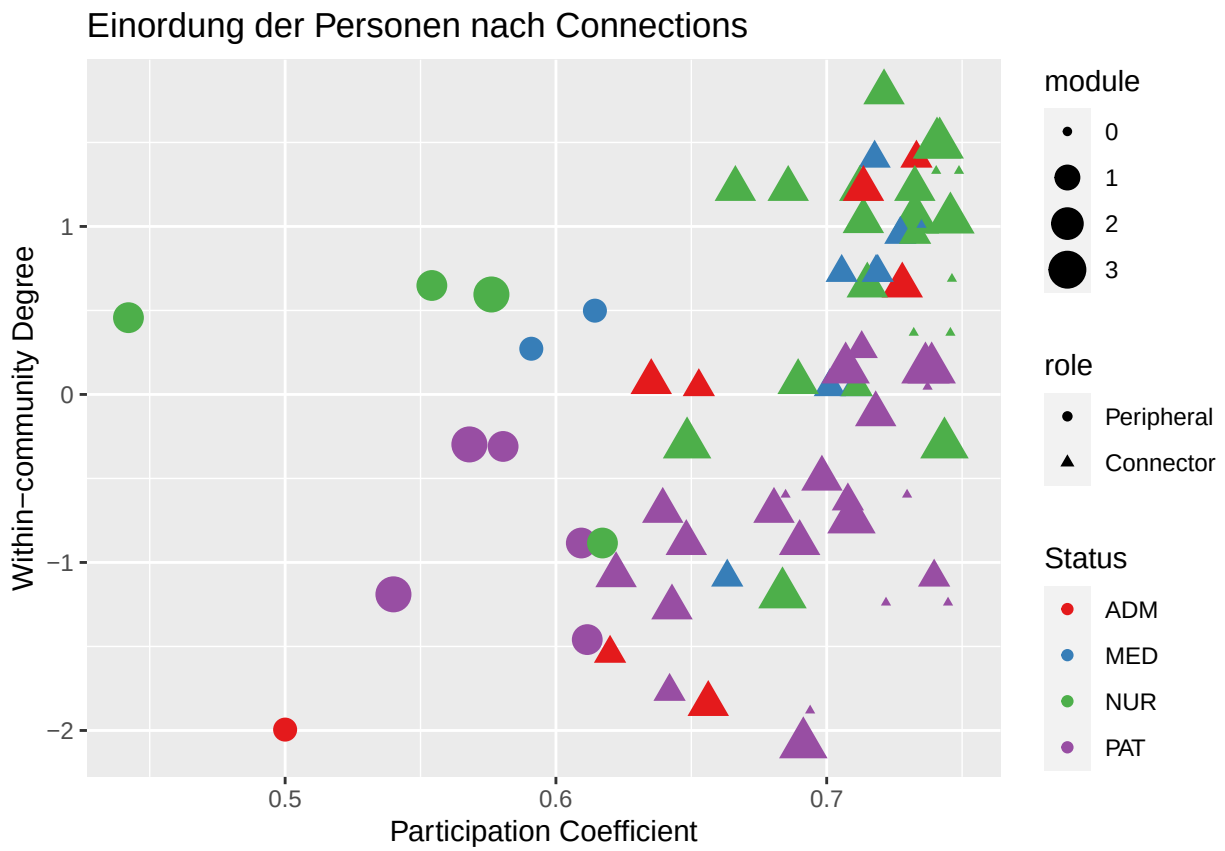
```
dfPlot <- merge(x = df2 , y = rnc[[1]], by = "name", all = TRUE)
head(dfPlot, 10)
```

```
##    name Status module connectivity participation       role
## 1     1    ADM      1   1.40546577     0.7331363  Connector
## 2     2    NUR      0   0.36706517     0.7321429  Connector
## 3     3    NUR      3  -0.29748206     0.7434842  Connector
## 4     4    NUR      0   0.36706517     0.7456747  Connector
## 5     5    NUR      3   1.04118721     0.7325620  Connector
## 6     6    NUR      3   1.48741029     0.7407407  Connector
## 7     7    NUR      0   1.00942923     0.7325331  Connector
## 8     8    NUR      3   0.59496412     0.5761773 Peripheral
## 9     9    MED      1   0.04533761     0.7011719  Connector
## 10   10    NUR      2   1.22336938     0.6662701  Connector
```

5. Plot #1

Der letzte Schritt ist das Plotten mit ggplot. Als y-Wertewird der "Within-community Degree", also den Wert, welcher die Vernetzung innerhalb der eigenen Community darstellt. Auf der x-Achse begindet sich "Participation Coefficient", bzw. die Vernetzung zu anderen Communities. Der Algorithmus unterscheidet dabei in verschiedene Rollen. Menschen, die weder eine hohe Vernetzung innerhalb, noch außerhalb ihrer Community haben werden als *"Peripherals"* bezeichnet. *"Connectors"* spielen keine große Roller innerhalb ihrer Community, sind jedoch sehr gut mit anderen Vernetzt. Des weitern gibt es noch zwei andere Rollen, welche bei diesem Beispiel nicht auftreten. *"Provincial Hubs*" nur in Ihrer Community gut vernetzt und *"Connector Hubs"* vereinen "Connectors" und "Provincial Hubs".

```
ggplot(dfPlot, aes(y = connectivity, x = participation)) +
  ggtitle("Einordung der Personen nach Connections") +
  xlab("Participation Coefficient") +
  ylab("Within-community Degree") +
  geom_point(aes(color=Status, shape=role, size=module)) +
  scale_color_brewer(palette = "Set1")
```



6. Vorbereitung Kombination

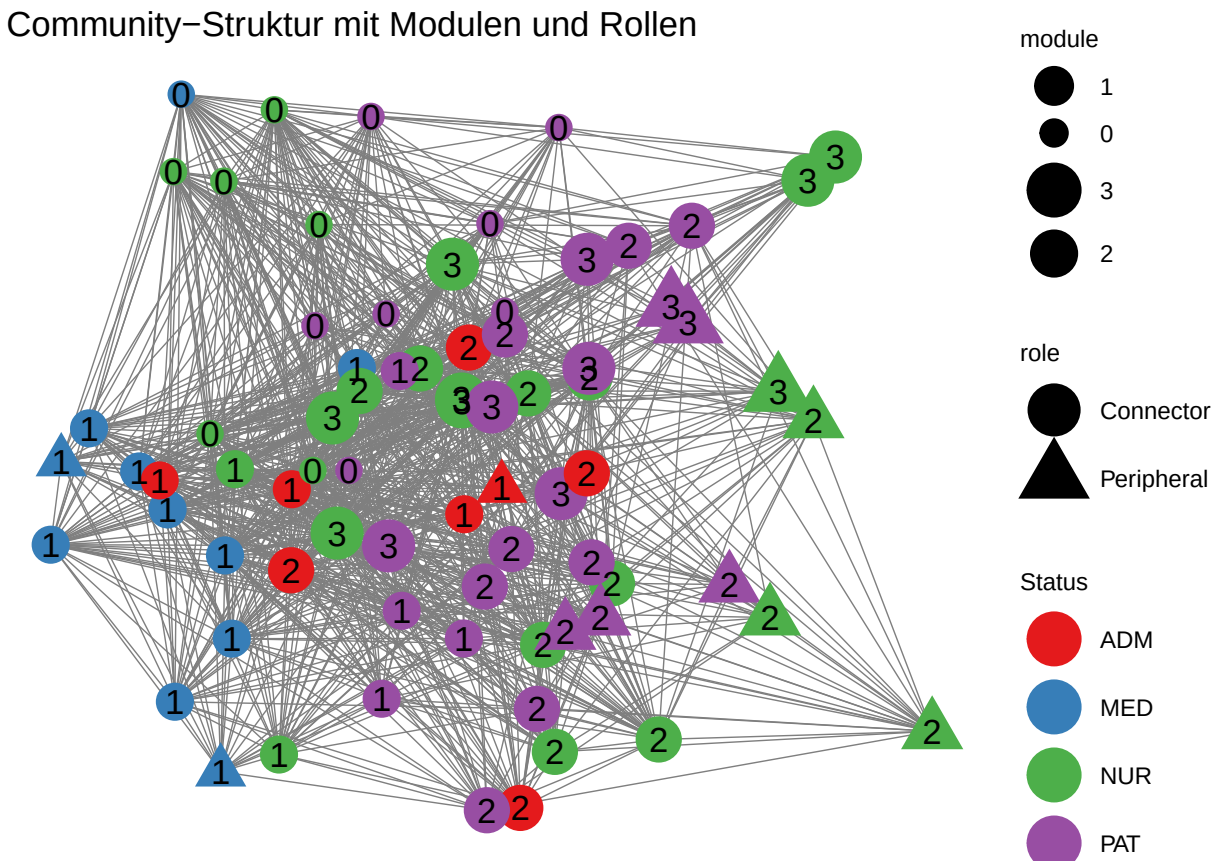
Die vom Algorithmus berechneten Daten können nun mit dem Netzerk kombiniert und in einer Visualisierung dargestellt werden. Dafür werden die gesäuberten und mit der Spalte "name" versehenen Daten aus Schritt 4 verwendet und mit den Edge-Daten kombiniert. Dafür werden die Spalten "modul" und "role" aus dem Algorithmus-Output erstellt und mit Werten befpllt. Da sich die Werte von "role" im "orderd factor"-Format vorliegen müssen wir diese erst zu Characters konvertieren, um in der Visualisierung verwendbar zu sein.

```
netPlot <- df2 %>% activate(nodes) %>% mutate(role <- merge(x = df2 , y = rnc[[1]][ , c("name", "module
netPlot <- netPlot %>% as_tbl_graph() %>% activate(nodes) %>% mutate(role = unlist(lapply(role, as.chara
```

7. Plot #2

Für den Plot des Netzwerks wird ggnet2 verwendet. Aus Gründen der Lesbarkeit und Vergleichbarkeit werden die selben Visualisierungs-Eigenschaften verwendet, wie in dem ersten Plot. Lediglich die module wurden mit Labels zusätzlich visualisiert, da diese im Fokus stehen. Die Verteilung der Edges wird über den modus "adj" realisiert, welcher auf der Adjazenzmatrix basiert.

```
ggnet2(netPlot, mode = "adj", label = "module", node.size = "module", palette = "Set1", node.color = "S
  ggtitle("Community-Struktur mit Modulen und Rollen")
```



Interpretation der Visualisierungen

Betrachtet man zu Beginn die erste Visualisierung, so kann man direkt erkennen, dass es deutlich mehr

Conclusio

Wiederholung der Fragestellung

Zusammenfassung der zentralen Ergebnisse

(***Limitationen, weiterführende Kommentare, etc.***)