Assignment Social Network Analysis

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

Für weitere Anleitungen und Tipps siehe den Syllabus und den Eintrag zur Portfolioprüfung.

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. * Operationalisierung eurer Forschungsfrage (welche Maße verwendet ihr um eure Forschungsfrage zu beantworten und warum?)

??? * Charakterisierung des Netzwerks

- 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. Pakete installieren

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)
```

2. Bibliotheken deklarieren

Die im letzen Schritt installierten Pakete werden nun als Bibliotheken deklariert.

```
library(igraph)
library(igraphdata)

library(ggraph)
library(ggthemes)
library(gganimate)
library(GGally)
library(gifski)

library(tidyverse)
library(tidygraph)

library(RColorBrewer)
library(rnetcarto)
```

3. Alte Daten aus dem Environment enfernen

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())
```

4. Daten von igraphdata instanziieren

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")
```

5. Schleifen und direktionale Beziehungen entfernen und in Variable speichern

In Schritt 5 werden Schleifen entfernt und direktionale Beziehungen in inderktionale Beziehungen aufgelöst. Dadurch werden Fehler in der Darstellung vermieden.

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

6. igraph-Objekt in adjacency matrix convertieren

Damit der Netrcarto-Algorithmus die Daten verarbeiten kann müssen diese als Adjazenzmatrix vorliegen. Der folgende Befehl konvertiert das Dateframe in dieses Format legt 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).

7. 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
## 58   58      0    -1.994855     0.5000000 Peripheral
## 46   46      0    -1.768167     0.6419753  Connector
## 71   71      0    -1.541479     0.6200000  Connector
## 14   14      0    -1.088103     0.6632653  Connector
## 39   39      0    -1.088103     0.7396694  Connector
```

8. Neues Dataframe mit Index als Spalte erstellen

Uns 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 wir die beiden Dataframes vereinen. Dafür fügen wir dem neu erstellten Dataframe eine Spalte mit dem Index hinzu, um im nächsten Schritt einen merge durchführen zu können. Da diese bei dem ursprünglichen DF "name" heißt, verwenden wir hier die gleiche Bezeichnung.

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

9. Status mergen

Durch das Hinzufügen der Index-Spalte im vorherigen Schritt können wir nun einen Merge durchführen. Dabei verwenden wir den Index, welcher als "name" in der urpsünglichen Tabelle enthalten war. Nun haben wir ein Dataframe mit allen benötigten Werten, welche für die Auswertung relevant sind.

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

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

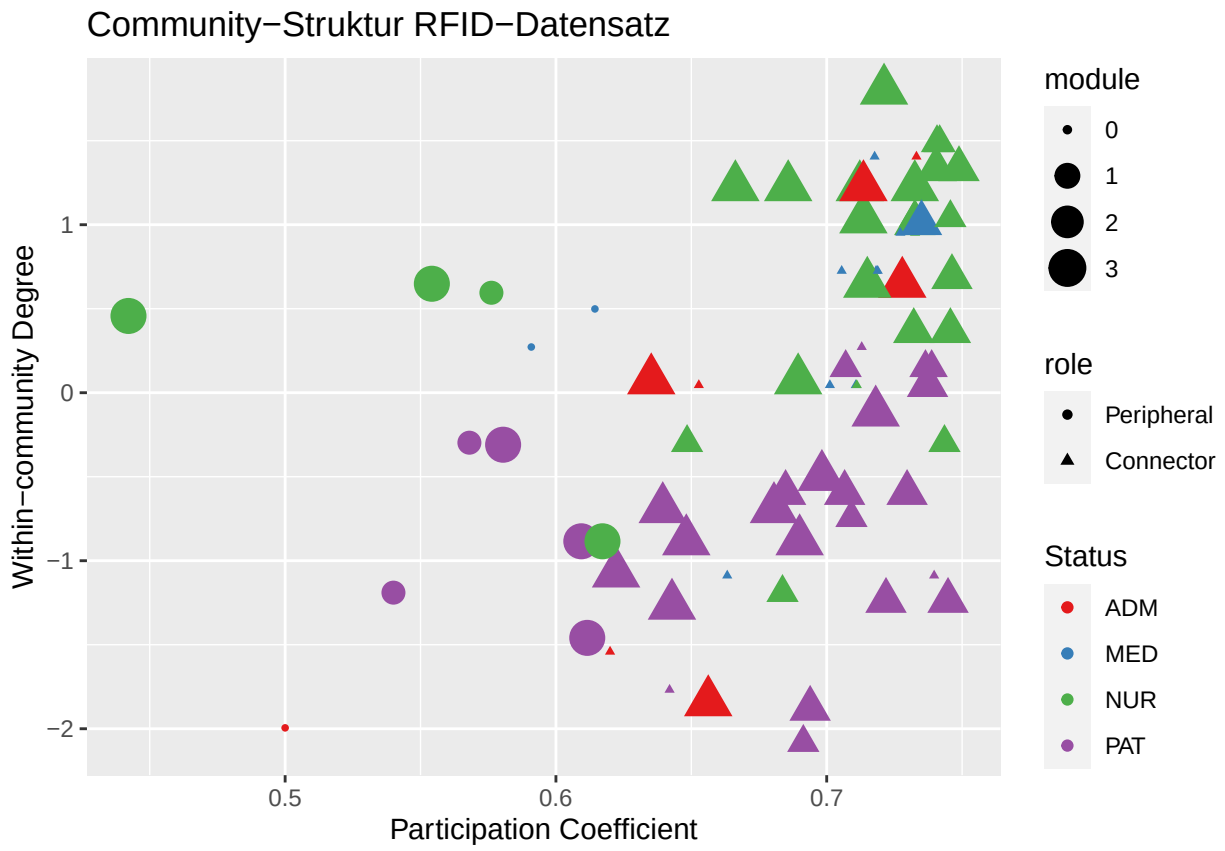
Nun liegt ein Dataframe vor, welches alle benötigten Daten beinhaltet und bereit für die Visualisierung ist.

10. Plot

Der letzte Schritt ist das Plotten mit ggplot. Als y-Werte verwenden wir dem "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(test, aes(y = connectivity, x = participation)) +
  ggtitle("Community-Struktur RFID-Datensatz") +
  xlab("Participation Coefficient") +
  ylab("Within-community Degree") +
  geom_point(aes(color=Status, shape=role, size=module)) +
  scale_color_brewer(palette = "Set1")
```




```
net <- df %>% as_tbl_graph()
net %>% as_tbl_graph() %>% activate(nodes) %>% mutate(name = row_number()) -> net2
rnc[[1]]
```

```
##    name module connectivity participation       role
## 58   58      0  -1.99485464     0.5000000 Peripheral
## 46   46      0  -1.76816661     0.6419753  Connector
## 71   71      0  -1.54147858     0.6200000  Connector
## 14   14      0  -1.08810253     0.6632653  Connector
## 39   39      0  -1.08810253     0.7396694  Connector
## 73   73      0  -0.63472647     0.7078189  Connector
## 28   28      0   0.04533761     0.6527778  Connector
## 9     9      0   0.04533761     0.7011719  Connector
## 12   12      0   0.04533761     0.7107438  Connector
## 63   63      0   0.04533761     0.7111111  Connector
```

```
## 65   65      0   0.27202563     0.5909091 Peripheral
## 48   48      0   0.27202563     0.7129291  Connector
## 18   18      0   0.49871366     0.6144000 Peripheral
## 30   30      0   0.72540169     0.7055324  Connector
## 16   16      0   0.72540169     0.7182261  Connector
## 35   35      0   0.72540169     0.7188366  Connector
## 11   11      0   0.95208971     0.7272000  Connector
## 13   13      0   0.95208971     0.7326389  Connector
## 15   15      0   1.40546577     0.7176931  Connector
## 1     1      0   1.40546577     0.7331363  Connector
## 66   66      1  -2.08237441     0.6913580  Connector
## 70   70      1  -1.18992823     0.5400000 Peripheral
## 32   32      1  -1.18992823     0.6836735  Connector
## 40   40      1  -0.74370515     0.7091413  Connector
## 38   38      1  -0.29748206     0.5680473 Peripheral
## 34   34      1  -0.29748206     0.6484375  Connector
## 3     3      1  -0.29748206     0.7434842  Connector
## 69   69      1   0.14874103     0.7069943  Connector
## 45   45      1   0.14874103     0.7364664  Connector
## 49   49      1   0.14874103     0.7387543  Connector
## 8     8      1   0.59496412     0.5761773 Peripheral
## 5     5      1   1.04118721     0.7325620  Connector
## 24   24      1   1.04118721     0.7456790  Connector
## 6     6      1   1.48741029     0.7407407  Connector
## 37   37      1   1.48741029     0.7417092  Connector
## 67   67      2  -1.88120902     0.6938776  Connector
## 47   47      2  -1.23884496     0.7218935  Connector
## 68   68      2  -1.23884496     0.7448015  Connector
## 50   50      2  -0.59648091     0.6848073  Connector
## 41   41      2  -0.59648091     0.7066116  Connector
## 43   43      2  -0.59648091     0.7296786  Connector
## 52   52      2   0.04588315     0.7372449  Connector
## 2     2      2   0.36706517     0.7321429  Connector
## 4     4      2   0.36706517     0.7456747  Connector
## 33   33      2   0.68824720     0.7462500  Connector
## 7     7      2   1.00942923     0.7325331  Connector
## 22   22      2   1.00942923     0.7350000  Connector
## 29   29      2   1.33061126     0.7404337  Connector
## 27   27      2   1.33061126     0.7488546  Connector
## 59   59      3  -1.84242376     0.6562500  Connector
## 75   75      3  -1.45919962     0.6115702 Peripheral
## 60   60      3  -1.26758755     0.6428571  Connector
## 56   56      3  -1.07597548     0.6222222  Connector
## 54   54      3  -0.88436340     0.6093750 Peripheral
## 61   61      3  -0.88436340     0.6171875 Peripheral
## 72   72      3  -0.88436340     0.6481481  Connector
## 44   44      3  -0.88436340     0.6900000  Connector
## 74   74      3  -0.69275133     0.6394558  Connector
## 53   53      3  -0.69275133     0.6805293  Connector
## 42   42      3  -0.50113926     0.6982249  Connector
## 55   55      3  -0.30952719     0.5804989 Peripheral
## 51   51      3  -0.11791512     0.7180900  Connector
## 31   31      3   0.07369695     0.6352041  Connector
## 62   62      3   0.07369695     0.6894531  Connector
```

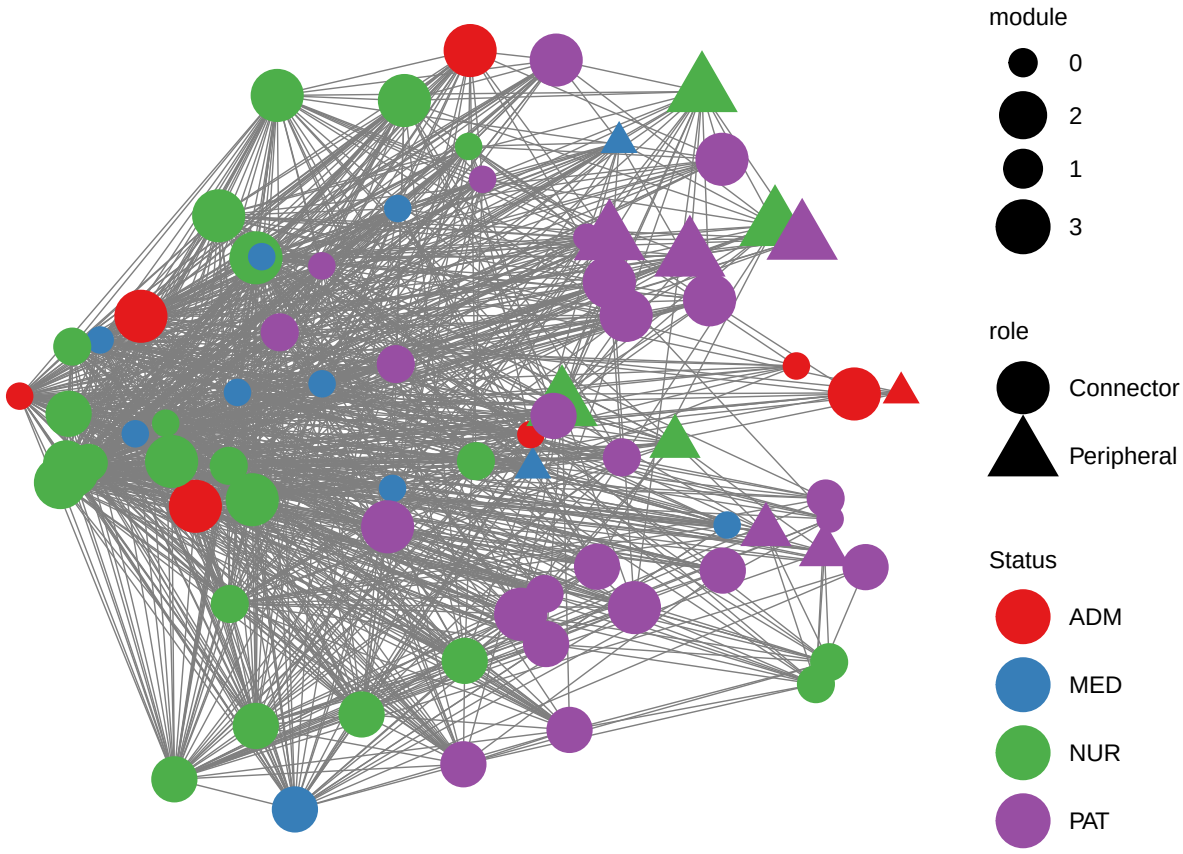
```
## 57   57      3   0.45692109     0.4421488 Peripheral
## 36   36      3   0.64853316     0.5541838 Peripheral
## 20   20      3   0.64853316     0.7149811  Connector
## 19   19      3   0.64853316     0.7279012  Connector
## 21   21      3   1.03175731     0.7135417  Connector
## 10   10      3   1.22336938     0.6662701  Connector
## 25   25      3   1.22336938     0.6857761  Connector
## 26   26      3   1.22336938     0.7122032  Connector
## 64   64      3   1.22336938     0.7135717  Connector
## 17   17      3   1.22336938     0.7325331  Connector
## 23   23      3   1.79820559     0.7211653  Connector
```

```
net2 %>% as_tbl_graph() %>% activate(nodes) %>% mutate(role <- merge(x = net2 , y = rnc[[1]][ , c("name
net3 %>% as_tbl_graph() %>% activate(nodes) %>% mutate(role = lapply(role, as.character)) -> net3
net3 %>% as_tbl_graph() %>% activate(nodes) %>% mutate(role = unlist(role)) -> net3

net3
```

```
## # A tbl_graph: 75 nodes and 1139 edges
## #
## # An undirected simple graph with 1 component
## #
## # Node Data: 75 x 4 (active)
##   Status  name module role
##   <chr>  <int>  <int> <chr>
## 1 ADM        1      0 Connector
## 2 NUR        2      2 Connector
## 3 NUR        3      1 Connector
## 4 NUR        4      2 Connector
## 5 NUR        5      1 Connector
## 6 NUR        6      1 Connector
## # ... 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
```

```
ggnet2(net3, mode = "eigen", node.color = "Status", palette = "Set1", node.size = "module", shape = "ro
```



Interpretation der Visualisierungen

Conclusio

Wiederholung der Fragestellung

Zusammenfassung der zentralen Ergebnisse

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