Hausarbeit 3

Hoesli/Lattmann

Transfernetzwerk der Schweizer Fussballigen

Wie bereits bei der letzten Arbeit beschrieben stellt unser Netzwerk die Transfers zwischen allen Teams aus den beiden höchsten Fussballigen der Schweiz in der Saison 2021/22 dar. Die Knoten stellen die 20 Mannschaften dar (je 10 pro Liga), während die Kanten jeweils einen Transfer, also einen Spieler, welcher von einem Team zu einem anderen gewechselt ist, verkörpern. Weil die Richtung des Wechsels bestimmt ist, handelt es sich um ein gerichtetes Netzwerk.

Die Grösse des Netzwerks beträgt nicht überraschend 20, was der Anzahl der Knoten entspricht. Eine grössere Aussagekraft zur Beschreibung des Netzwerkes hat der Wert für die Dichte, welcher 0.242 beträgt und somit auf eine ziemlich hohe Dichte hinweist, welche sich auch beim Betrachten des Graphen widerspiegelt. Die hohe Dichte geht ausserdem darauf zurück, dass viele von allen möglichen Interaktionen zwischen den Teams ausgeschöpft werden. Passend zu diesem Ergebnis besteht das Netzwerk nur aus einer einzigen Komponente, womit alle Knoten über gewisse Pfade miteinander verbunden sind. Dies zeigt sich gewissermassen auch im Durchmesser des Netzwerkes, welcher 5 beträgt und die Länge des kürzesten Pfades zwischen den beiden am weitesten von einander entfernten Knoten ausdrückt. Anders ausgedrückt: Bei einer Netzwerkgrösse von 20 scheint ein Durchmesser von 5 eher klein zu sein, was widerum für ein dichtes Netzwerk spricht. Mit 0.176 fällt zuletzt der Clustering-Koeffizient nicht allzu hoch aus. Dies kommt wohl daher, da im Graphen nicht allzu viele "Dreicke" vorhanden sind bzw. gewisse Teams keine Spieler untereinander austauschten. Die Masszahlen drücken schliesslich aus, dass es sich beim vorliegenden Netzwerk um ein äusserst Dichtes Gebilde handelt, in welchem alle der 20 Akteure mit einigen der anderen Knoten interagieren, Interaktionspartner aber dennoch bewusst selektioniert werden (dafür spricht der eher geringe Clustering-Koeffizient).

Bei der Betrachtung der verschiedenen Zentralitäts-Masse (vgl. Abbildung 1) offenbaren sich einige Unterschiede bezüglich der Varianz der Zentralität zwischen den Knoten. Bei der degree-basierten Zentralisierung variieren die Werte für alle Knoten in einem überschaubaren Bereich von 3-7. Dies drückt aus, wie viele direkte Verbindungen zu anderen Knoten bei jedem Knoten vorhanden sind (Jansen, 2006). Gemäss Jansen weist das Mass auf die "mögliche Kommunikationsaktivität" hin, womit die Aktivität der Teams bezüglich dem Abgeben von Spielern nicht allzu fest variiert und somit auch kein Team einen extrem "zentralen" Platz im Graphen einnimmt. Die betweenness-basierte Zentralisierung offenbart grössere Unterschiede zwischen den einzelnen Knoten, da die Unterschiede bezüglich kürzesten Pfaden, welche durch den betrachteten Knoten führen, grösser sind als jene bezüglich direkten Verbindungen. Der Wertebereich reicht somit von 1.7 bis 28.8 und drückt gemäss Jansen (2006) die "mögliche Kommunikationskontrolle" aus. Das "zentralste" Team im Netzwerk wäre somit YB, welches über ein hohes Mass an "Kontrolle" darüber verfügt, wie Spieler innerhalb der Liga transferiert werden. Zulezt kann noch die closeness-basierte Zentralisierung betrachtet werden, welche aufgrund der Werte. welche alle nahe bei 0.5 liegen und nur wenig variieren, nahe legt dass alle Knoten ein ähnlich hohes Mass an Zentralität aufweisen. Die tiefen Werte hier implizieren, dass zwischen sämtlichen Knoten durchaus geringe durchschnittliche Distanzen bestehen. Gemäss Jansen (2006) dient das Mass auch als Indikator für die Unabhängiigkeit bzw. Effizienz der Knoten, welche somit bei allen Teams hoch sein dürfte.

Einserseits sagen somit alle verschiedenen Zentralisierungs-Masse aus, dass im vorliegenden Netzwerk nicht allzu grosse Unterschiede bezüglich der zentralen Positionierung der Knoten vorliegen und somit kaum "periphere" Akteure vorhanden sind. Auf der anderen Seite legt v.a. die betweenness-basierte Zentralisierung nahe, dass einige Knoten eine gewisse stärkere "Kontrolle" über das gesamte Transfergeschehen haben als

andere. Der Vergleich der verschiedenen Zentralitätsmasse in der Visualisierung zeigt wie bereits oben beschrieben, dass allgmein eher hohe Zentralität vorliegt. Vor allem bei der Betweenness-Zentralität offenbaren sich aber dennoch einige Unterschiede. Es zeigt sich, dass Knoten, welche auch optisch am zentralsten im Netzwerk liegen, ebenfalls grösser dargestellt werden (aufgrund der höheren Betweenness-Werte). Diese Knoten dürften somit eine wichtige Stellung für das Transfergeschehen in den beiden Ligen haben.

Da es sich um ein sehr zentrales Netzwerk handelt, lassen sich darin weder Brücken noch Cutpoints finden. Schaut man sich aber bspw. nur die Super League isoliert an (vgl. Abbildung 2), offenbaren sich der FC Basel und der FC Luzern als Cutpoints. Schliesst man die beiden Teams also aus dem Netzwerk aus, bilden sich mehrere Komponenten für die Liga. Dies kann bspw. so interpretiert werden, dass Teams aus der Challenge League für das gesamte Transfergeschehen wichtige "Brücken" bilden, welche schliesslich den Spieleraustausch zwsischen Teams in der Super League begünstigen und ansonsten vielleicht eher voneinander isoliert wären. Als zentrale Aussage lässt sich aus den empirischen Gegebenheiten ableiten, dass das Transfernetzwerk der beiden höchsten Schweizer Fussballigen ein sehr dichtes Gefüge ist, in welchem zwischen verschiedensten Teams Spieler ausgetauscht werden und die meisten Teams ebenfalls eine durchaus hohe Zentralität im Netzwerk aufweisen, was wiederum für einen starken "Wettbewerb" unter den Klubs spricht. Zudem wird dadurch auch aufgezeigt, wie wichtig die heimischen Ligen als Partner für die Kaderplanung sein können.

Anzahl Wörter: 795

Abbildung 1: Visualisierung nach Zentralität

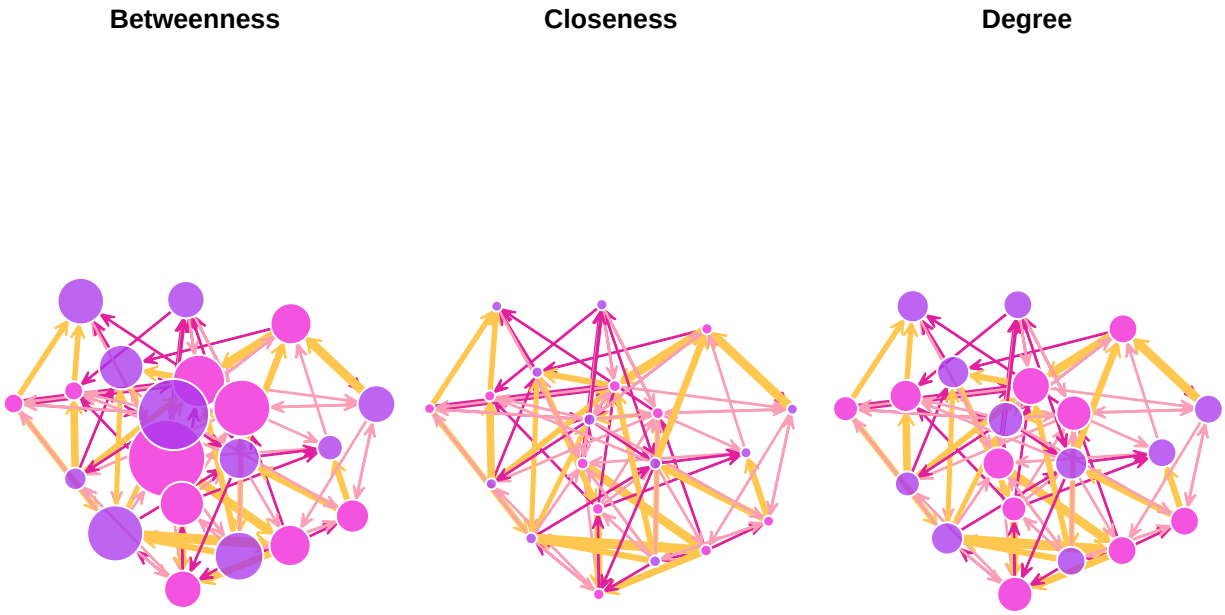
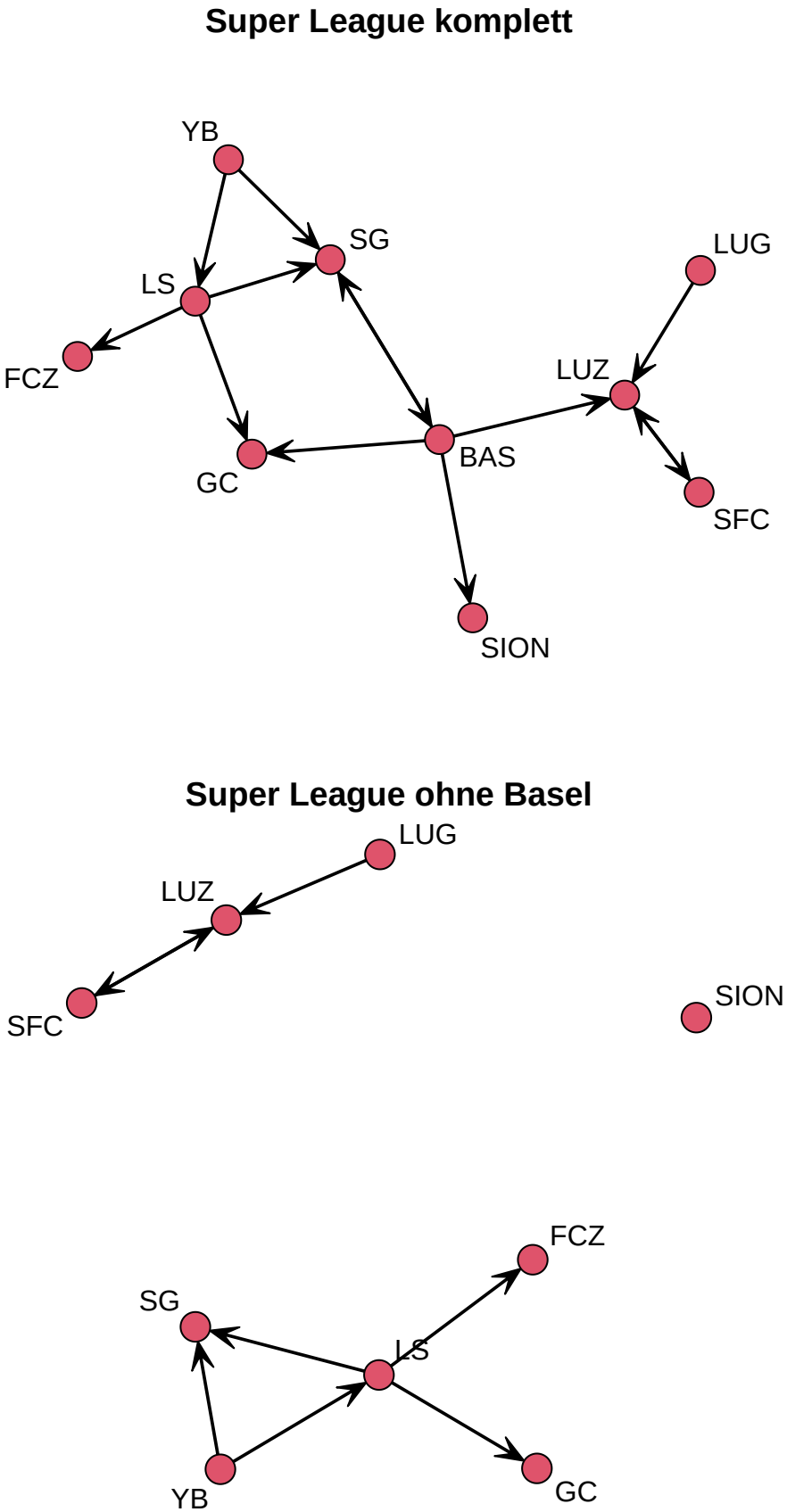


Abbildung 2: Subgraph der Super League



Appendix: Gesamten R-Code für den Bericht

```
## ----set up of packages,
## include=FALSE-----------------------------------------------------------------------------------------------
## install.packages('statnet')
library("RColorBrewer")
library("statnet")
library("scales")
library("formatR")
## ----set up of edge list ,
## tidy=TRUE---------------------------------------------------------------------------------------------------
edge_list <- read.csv("/Users/julienlattmann/Desktop/Edge List Values.csv",
    sep = ";")
# Ertsellen der Kanten-Liste aufgrund der gesammelten Daten
netmat2 <- edge_list
# Netzwerk-Objekt erstellen
transfer_net <- network(netmat2, matrix.type = "edgelist", directed = TRUE)
# Knoten Acronyms
network.vertex.names(transfer_net) <- c("YB", "BAS", "SFC", "LUG",
    "LUZ", "LS", "SG", "FCZ", "SION", "GC", "VAD", "THU", "SLO",
    "FCS", "AAR", "WIN", "WIL", "SCK", "XAM", "YS")
# Knoten-Attribut 'league' definieren
set.vertex.attribute(transfer_net, "league", c("SL", "SL", "SL",
    "SL", "SL", "SL", "SL", "SL", "SL", "SL", "ChL", "ChL", "ChL",
    "ChL", "ChL", "ChL", "ChL", "ChL", "ChL", "ChL"))
# Kanten-Attribut 'spendings' definieren: Transfersumme in
# Tsd.
spendings <- read.csv("/Users/julienlattmann/Desktop/Spendings.csv",
    sep = ";")
set.edge.attribute(transfer_net, "spendings", spendings)
# Kanten-Attribut 'type' definieren:
# Spieler-Transaktionstyp (Ablöse, Leihe, unbekannt)
type <- read.csv("/Users/julienlattmann/Desktop/Types_ID.csv",
    sep = ";")
set.edge.attribute(transfer_net, "type", type)
# Knoten-Attribut 'alldeg' definieren (abgekürzte Variante)
transfer_net %v% "alldeg" <- degree(transfer_net)
summary(transfer_net)
```

```
## Network attributes:
##   vertices = 20
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##  total edges = 92
##    missing edges = 0
##    non-missing edges = 92
##  density = 0.2421053
##
## Vertex attributes:
##
```

```
##  alldeg:
##    numeric valued attribute
##    attribute summary:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    6.00    7.75    9.00    9.20   11.25   15.00
##
##  league:
##    character valued attribute
##    attribute summary:
## ChL  SL
##  10  10
##   vertex.names:
##    character valued attribute
##    20 valid vertex names
##
## Edge attributes:
##
##  spendings:
##    integer valued attribute
##    92values
##
##  type:
##    integer valued attribute
##    92values
##
## Network edgelist matrix:
##       [,1] [,2]
##  [1,]    1   15
##  [2,]    1   18
##  [3,]    1    6
##  [4,]    1   17
##  [5,]    1    7
##  [6,]    1   12
##  [7,]    1   20
##  [8,]    2   16
##  [9,]    2    5
## [10,]    2    9
## [11,]    2    7
## [12,]    2   19
## [13,]    2   10
## [14,]    3    5
## [15,]    3   18
## [16,]    3   20
## [17,]    3   13
## [18,]    4   14
## [19,]    4   18
## [20,]    4   12
## [21,]    4    5
## [22,]    5   15
## [23,]    5   12
## [24,]    5   18
## [25,]    5   14
## [26,]    5    3
## [27,]    6   17
```

```
## [28,]    6   10
## [29,]    6    8
## [30,]    6    7
## [31,]    6   13
## [32,]    7    2
## [33,]    7   13
## [34,]    7   11
## [35,]    8   16
## [36,]    8   19
## [37,]    8   17
## [38,]    8   14
## [39,]    9   20
## [40,]   10   15
## [41,]   10   20
## [42,]   10   17
## [43,]   10   18
## [44,]   10   14
## [45,]   10   16
## [46,]   10   11
## [47,]   11   12
## [48,]   11    9
## [49,]   11    5
## [50,]   12   20
## [51,]   12    2
## [52,]   12   13
## [53,]   13   19
## [54,]   13   18
## [55,]   13   12
## [56,]   13    6
## [57,]   13    1
## [58,]   13    7
## [59,]   13    3
## [60,]   14    4
## [61,]   14    6
## [62,]   14    8
## [63,]   14   10
## [64,]   15    9
## [65,]   15   19
## [66,]   15    1
## [67,]   15    5
## [68,]   15   10
## [69,]   16    4
## [70,]   16    1
## [71,]   17    7
## [72,]   17   13
## [73,]   17    5
## [74,]   17    8
## [75,]   17   10
## [76,]   17    2
## [77,]   17    1
## [78,]   18   15
## [79,]   18   11
## [80,]   18   19
## [81,]   18    8
```

```
## [82,]   18    9
## [83,]   18    4
## [84,]   18    3
## [85,]   18   10
## [86,]   18    5
## [87,]   19   16
## [88,]   19    7
## [89,]   19    4
## [90,]   19    9
## [91,]   19    6
## [92,]   20    1
```

```
########## Start Hausarbeit 3 ##########
# Grösse des Netzwerkes berechnen
Grösse <- network.size(transfer_net)
Grösse
```

```
## [1] 20
```

```
# Dichte des Netzwerkes berechnen
Dichte <- gden(transfer_net)
Dichte
```

```
## [1] 0.2421053
```

```
# oder von Hand --> m/(n(n-1))
Dichte_H <- 92/(20 * (20 - 1))
Dichte_H
```

```
## [1] 0.2421053
```

```
# Komponenten des Netzwerkes berechnen
Komp <- components(transfer_net)
Komp
```

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

```
# Durchmesser des Netzwerkes berechnen
Dm <- geodist(transfer_net)
Dm_max <- max(Dm$gdist)
Dm_max
```

```
## [1] 5
```

```
# Clustering Coefficient des Netzwerkes berechnen
Clust <- gtrans(transfer_net)
Clust
```

```
## [1] 0.1760204
```

```
# Betweenness
bet <- betweenness(transfer_net, gmode = "graph")
bet
```

```
##  [1] 28.787271  9.448153  1.844246  8.142316 13.235220  8.196140  6.755556
##  [8]  5.327055  1.742857 15.538220  2.412302  9.550992 15.167995  6.878083
## [15]  6.734830  3.152686 11.477543 24.545531  8.092170 10.470833
```

```
# Degree
deg <- degree(transfer_net, gmode = "graph")
deg
```

```
##  [1] 5 3 3 4 7 4 6 4 5 6 3 5 5 4 4 4 4 6 5 5
```

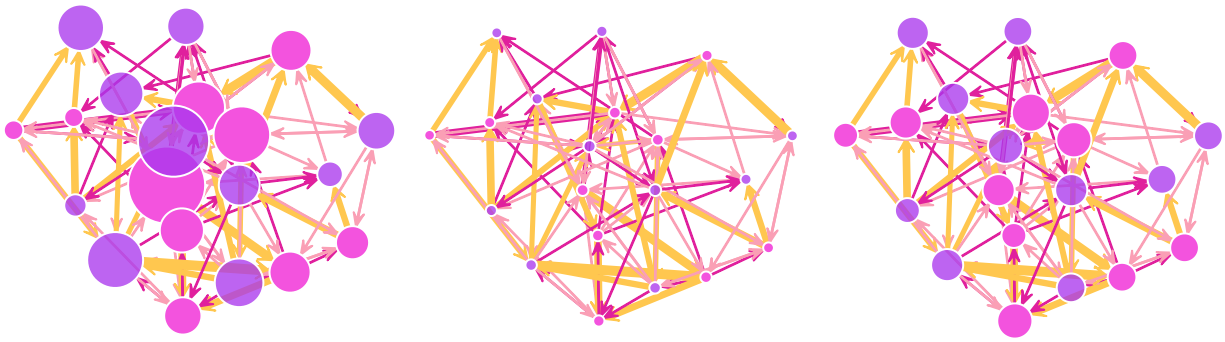
```
# Closeness
cls <- closeness(transfer_net, gmode = "graph")
cls
```

```
##  [1] 0.6551724 0.6333333 0.5428571 0.5937500 0.6551724 0.6333333 0.6129032
##  [8] 0.5588235 0.5757576 0.6551724 0.5937500 0.5937500 0.6333333 0.5588235
## [15] 0.5937500 0.5757576 0.6333333 0.7037037 0.6785714 0.5588235
```

```
par(mfrow = c(1, 3))
linecol_pal <- c("#e0209d", "#fa9fb5", "#ffc74f")
league_pal = c("#aa36eec4", "#f353de")
type_cat <- as.factor(get.edge.attribute(transfer_net, "type"))
league_cat = as.factor(get.vertex.attribute(transfer_net, "league"))
# Using Betweenness as vertex size
set.seed(7)
par(mar = c(0.5, 0.1, 5, 0.1))
gplot(transfer_net, vertex.col = league_pal[league_cat], displaylabels = FALSE,
    vertex.cex = sqrt(bet), vertex.border = "white", edge.col = linecol_pal[transfer_net %e%
        "type"], edge.lwd = sqrt(transfer_net %e% "spendings") *
        0.35, mode = "fruchtermanreingold", boxed.labels = TRUE,
    label.border = "white", label.pos = 0, label.bg = "white",
    label.col = league_pal[league_cat], label.cex = 0.85, usearrows = TRUE,
    main = "Betweenness")
# Using Closness as vertex size
set.seed(7)
par(mar = c(0.1, 0.1, 5, 0.1))
gplot(transfer_net, vertex.col = league_pal[league_cat], displaylabels = FALSE,
    vertex.cex = sqrt(cls), vertex.border = "white", edge.col = linecol_pal[transfer_net %e%
        "type"], edge.lwd = sqrt(transfer_net %e% "spendings") *
        0.35, mode = "fruchtermanreingold", boxed.labels = TRUE,
    label.border = "white", label.pos = 0, label.bg = "white",
    label.col = league_pal[league_cat], label.cex = 0.85, usearrows = TRUE,
    main = "Closeness")
# Using Degree as vertex size
set.seed(7)
par(mar = c(0.1, 0.1, 5, 0.1))
gplot(transfer_net, vertex.col = league_pal[league_cat], displaylabels = FALSE,
    vertex.cex = sqrt(deg), vertex.border = "white", edge.col = linecol_pal[transfer_net %e%
```

```
        "type"], edge.lwd = sqrt(transfer_net %e% "spendings") *
        0.35, mode = "fruchtermanreingold", boxed.labels = TRUE,
    label.border = "white", label.pos = 0, label.bg = "white",
    label.col = league_pal[league_cat], label.cex = 0.85, usearrows = TRUE,
    main = "Degree")
```

Betweenness **Closeness** **Degree**



```
# Cutpoints identifizieren (weak component rule for
# directed networks)
cps <- cutpoints(transfer_net, connected = "weak")
cps
```

```
## integer(0)
```

```
# Brücken identifizieren
bridges <- function(dat, mode = "graph", connected = c("strong",
    "weak")) {
    e_cnt <- network.edgecount(dat)
    if (mode == "graph") {
        cmp_cnt <- components(dat)
        b_vec <- rep(FALSE, e_cnt)
        for (i in 1:e_cnt) {
            dat2 <- dat
            delete.edges(dat2, i)
            b_vec[i] <- (components(dat2) != cmp_cnt)
```

```
        }
    } else {
        cmp_cnt <- components(dat, connected = connected)
        b_vec <- rep(FALSE, e_cnt)
        for (i in 1:e_cnt) {
            dat2 <- dat
            delete.edges(dat2, i)
            b_vec[i] <- (components(dat2, connected = connected) !=
                cmp_cnt)
        }
    }
    return(b_vec)
}
bridges(transfer_net)
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [37] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

```
brnet <- bridges(transfer_net)
# Welche Edges sind Brücken?
which(brnet == TRUE)
```

```
## [1] 39 92
```

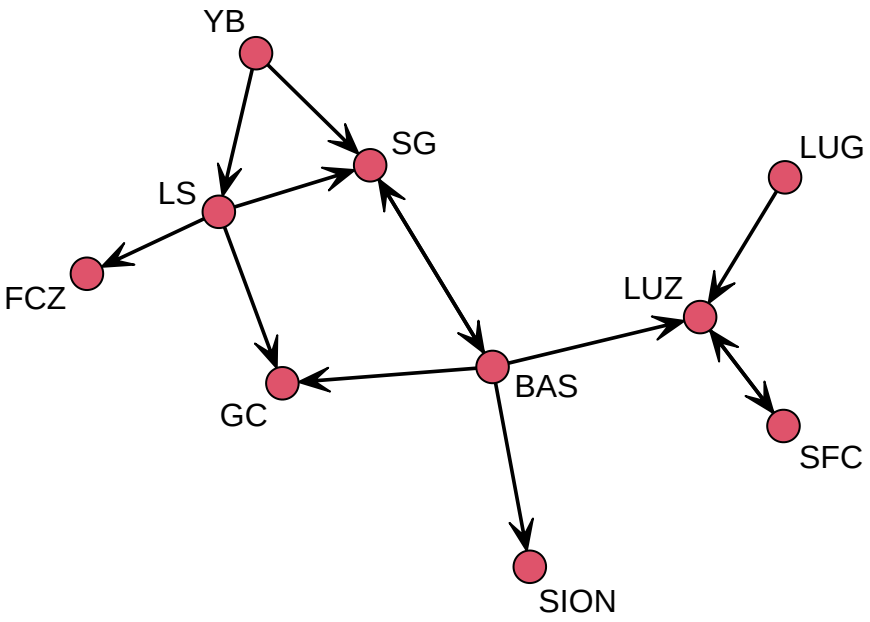
```
par(mar = c(1, 0, 1, 0), mfrow = c(1, 1))
net2 <- transfer_net
components(net2)
```

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

```
delete.edges(net2, c(39, 92))
components(net2)
```

```
## [1] 3
```

```
# gplot(net2,gmode='graph',vertex.col='red',
# edge.col=brnet+2, jitter=FALSE,displaylabels=TRUE)
# Cutpoints und Brücken für den Subgraphen der Super League
net_SL <- delete.vertices(transfer_net, 11:20)
gplot(net_SL, displaylabels = TRUE, main = "Super League komplett")
```




```
cutpoints(net_SL, connected = "weak")
```

```
## [1] 2 5 6
```

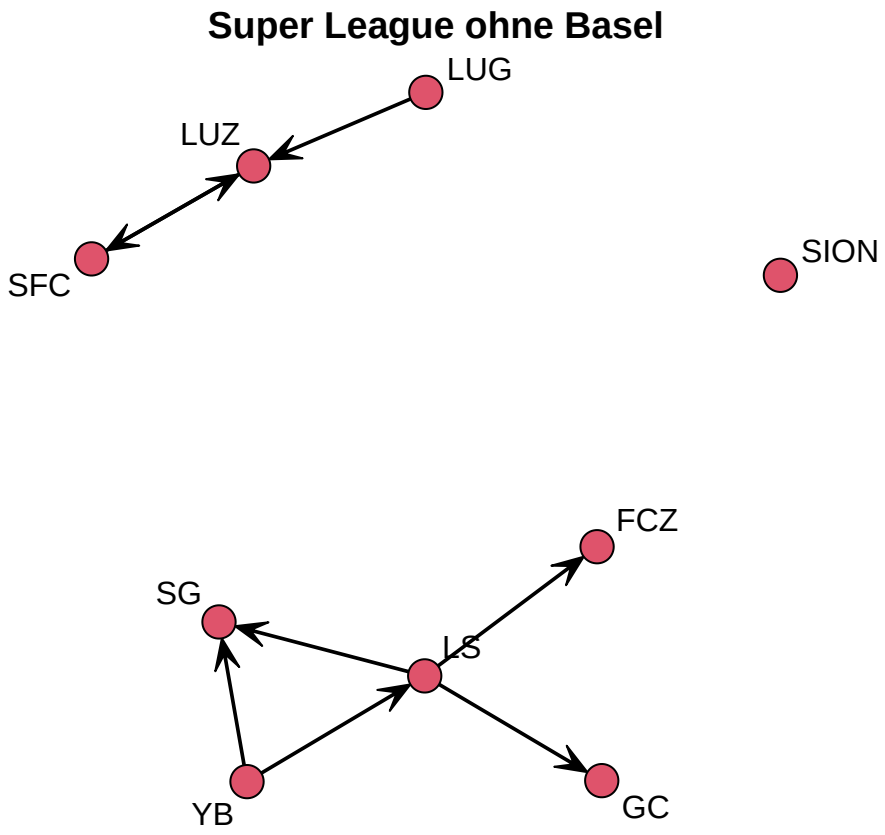
```
brnet_SL <- bridges(net_SL)
which(brnet_SL == TRUE)
```

```
## [1] 11
```

```
net_SL2 <- delete.vertices(net_SL, 2)
components(net_SL2)
```

```
## [1] 8
```

```
gplot(net_SL2, displaylabels = TRUE, main = "Super League ohne Basel")
```


Super League ohne Basel
LUG
LUZ
SFC
SION
FCZ
SG
LS
YB
GC