SIECI TEMPORALNE

Karol Działowski
Zachodniopomorski Uniwersytet Technologiczny

11 listopada 2020

Spis treści

1 Wstęp

Celem laboratorium było zapoznanie się z podstawowymi funkcjami do przetwarzania sieci temporalnych. Sieci temporalne (dynamiczne) to takie sieci, które zmieniają się w czasie. Zadanie polegało na zapoznanie się z funkcjami tworzenia sieci temporalnych, wyznaczanie temporalnego stopnia wierzchołka, wyznaczanie ścieżek temporalnych, wizualizacji ścieżek na kompletnym grafie oraz wyznaczanie ścieżek *fwd* i *bkwd*.

W ramach laboratorium przeprowadzono też badania na sieci syntetycznej *small-world* [1] wygenerowanej 10 razy i złączonej w sieć dynamiczną. Ostatnim zadaniem było przeprowadzenie badań na sieci rzeczywistej.

2 Zadanie 1 - zapoznanie się z narzędziami do tworzenia sieci temporalnych

2.1 Przygotowanie środowiska

Przed rozpoczęciem pracy należy pobrać i wczytać biblioteki niezbędne do pracy z sieciami temporalnymi. Były to: *sna*, *tsna*, *ndtv*. Kod przedstawiono na listingu 1.

```
setwd("C:/Dev/complex_networks/lab_5/")
library(igraph)
install.packages("sna")
install.packages("tsna")
install.packages("ndtv")
library(sna)
library(tsna)
library(ndtv)
```

Listing 1: Wczytanie niezbędnych bibliotek.

2.2 Wczytanie *Daniel McFarland's Streaming Classroom Interactions Dataset*

Następnie wczytano sieć dynamiczną opisującą kontakty towarzyskie pomiędzy nauczycielami i uczniami z obserwacji przeprowadzonych przez Daniela McFarlanda w 1996 roku [2].

Wyznaczono temporalnego stopnia wierzchołków oraz wyliczono średni stopień wierzchołków dla wczytanej sieci. Kod przedstawiono w listingu 2.

```
> ?cls33_10_16_96
> data(McFarland_cls33_10_16_96)
> tDegree(cls33_10_16_96)
Time Series:
Start = 0
End = 49
Frequency = 1
    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 1  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0
 2  0  0  0  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0  0
...
47  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
48  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
49 NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
> mean(tDegree(cls33_10_16_96),na.rm=TRUE)
[1] 0.07346939
```

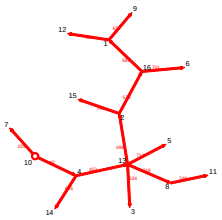
Listing 2: Sieć Daniela McFarlanda.

2.3 Sieć Jim Moody'ego - kontakty seksualne

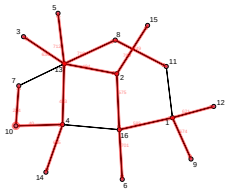
Następnie wczytano sieć Jima Moody'ego zawierającą symulację kontaktów seksualnych z 16 węzłami i 18 krawędziami [3]. Wyniki działania listingu 3 przedstawiono na rysunku 1.

```
?moodyContactSim
data(moodyContactSim)
v10path<-tPath(moodyContactSim,v=10,start=0)
plot(v10path)
plotPaths(moodyContactSim,v10path)
v1path<-tPath(moodyContactSim,v=1,start=0)
plotPaths(moodyContactSim,list(v10path,v1path))
```

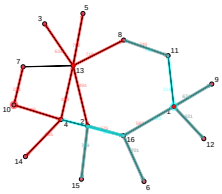
Listing 3: Sieć Jima Moody'ego.



(a) Tylko ścieżka 10



(b) Ścieżka 10 nałożona na graf



(c) Nałożenie ścieżki 10 i ścieżki 1 na graf.

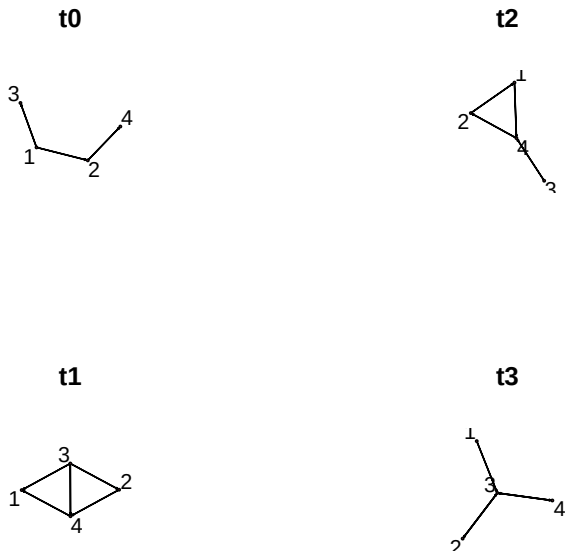
Rysunek 1: Temporalne ścieżki

2.4 Obiekt *networkDynamic*

Stworzono obiekt *networkDynamic* z 4 wygenerowanych sieci. Przedstawiono sieci składowe (poszczególne migawki czasu) na rysunku 2. Wyznaczono również stopnie wierzchołków w listingu 4.

```
nlist<-replicate(4,list(network(matrix(rbinom(16,5,0.1),ncol=4,nrow=4))))
> par(mfcol=c(2,2))
> plot(nlist[[1]],displaylabels=TRUE,main='t0')
> plot(nlist[[2]],displaylabels=TRUE,main='t1')
> plot(nlist[[3]],displaylabels=TRUE,main='t2')
> plot(nlist[[4]],displaylabels=TRUE,main='t3')
> dnet<-networkDynamic(network.list= nlist)
Neither start or onsets specified, assuming start=0
Onsets and termini not specified, assuming each network in network.list should have a discrete
    spell of length 1
Argument base.net not specified, using first element of network.list instead
Created net.obs.period to describe network
 Network observation period info:
  Number of observation spells: 1
  Maximal time range observed: 0 until 4
  Temporal mode: discrete
  Time unit: step
  Suggested time increment: 1
> tDegree(dnet)
Time Series:
Start = 0
End = 4
Frequency = 1
   1  2  3  4
0  2  2  1  1
1  2  2  3  3
2  2  2  1  3
3  1  2  5  2
4 NA NA NA NA
```

Listing 4: Obiekt networkDynamic



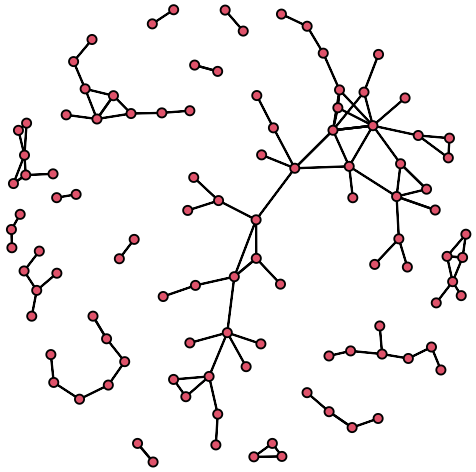
Rysunek 2: Obiekt networkDynamic w kolejnych migawkach

2.5 Wczytywanie sieci z plików

Najpierw wczytano sieć statyczną.

```
setwd("C:/Dev/complex_networks/lab_5/")
StaticEdges <- read.csv("StaticEdgelist.csv")
thenetwork <- network(
  StaticEdges,
  directed = FALSE,
  bipartite = FALSE
)
plot(thenetwork)
```

Listing 5: Wczytywanie sieci statycznej

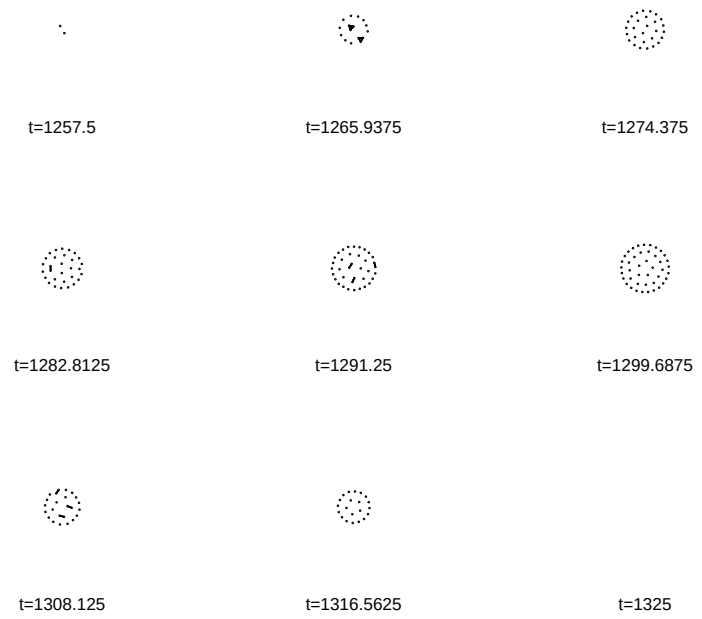


Rysunek 3: Sieć statyczna

Następnie stworzono sieć dynamiczną z plików *DynamicNodes.csv* oraz *DynamicEdges.csv*. Sposób postępowania przedstawiono na listingu 6. Uzyskany rysunek za pomocą funkcji *filmstrip* przedstawiono na rysunku 4.

```
DynamicNodes <- read.csv("DynamicNodes.csv")
DynamicEdges <- read.csv("DynamicEdges.csv")
tn <- networkDynamic(
  thenetwork,
  edge.spells = DynamicEdges,
  vertex.spells = DynamicNodes
)
network.dynamic.check(tn)
filmstrip(tn, displaylabels = FALSE)
```

Listing 6: Wczytywanie sieci dynamicznej.



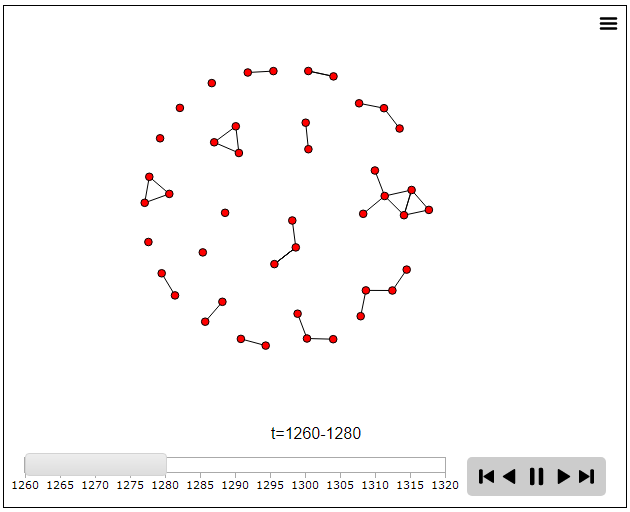
Rysunek 4: Sieć dynamiczna

2.6 Generowanie animacji

W celu wygenerowania animacji skorzystano z funkcji *compute.animation* oraz *render.d3movie*. Uzyskana animacja otwiera się w nowym oknie w przeglądarce. Interfejs pozwala na sterowanie animacją (pauzowanie, przewijanie itd.). Interfejs przedstawiono na rysunku 5.

```
compute.animation(
  tn,
  animation.mode = "kamadakawai",
  slice.par = list(
    start = 1260,
    end = 1300,
    interval = 1,
    aggregate.dur = 20,
    rule = "any"
  )
)
render.d3movie(
  tn,
  displaylabels = FALSE
)
```

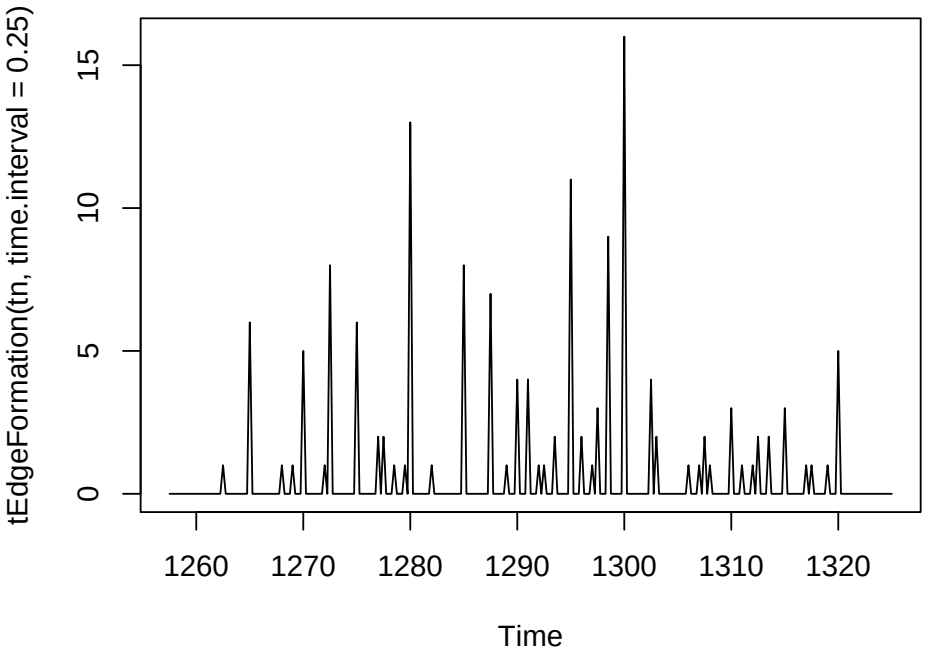
Listing 7: Generowanie animacji.



Rysunek 5: Interfejs animacji

2.7 Dynamika krawędzi

Wygenerowano wykres dynamiki krawędzi za pomocą komendy `plot(tEdgeFormation(tn, time.interval = .25))`. Wynik przedstawiono na rysunku 6.



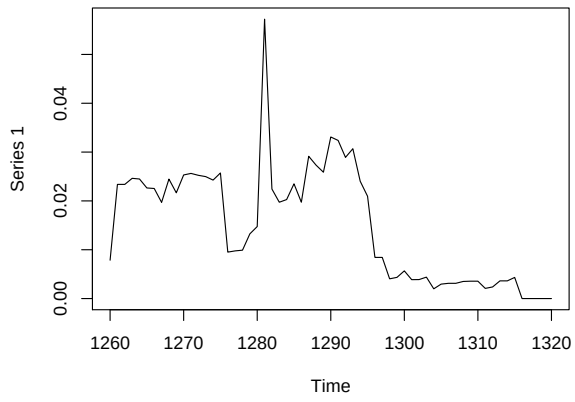
Rysunek 6: Dynamika krawędzi

2.8 Dynamika betweenness centrality i stopni wierzchołków

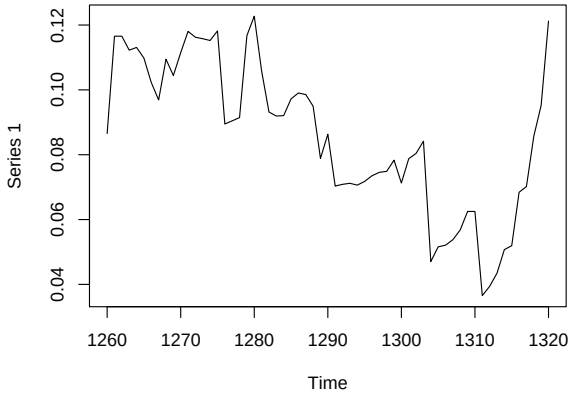
Kolejnym krokiem było wyznaczenie betweenness i stopni wierchołków w interwałach czasowych.

```r
dynamicBetweenness <- tSnaStats(
  tn,
  snafun = "centralization",
  start = 1260,
  end = 1320,
  time.interval = 1,
  aggregate.dur = 20,
  FUN = "betweenness"
)
plot(dynamicBetweenness)
dynamicDegree <- tSnaStats(
  tn,
  snafun = "centralization",
  start = 1260,
  end = 1320,
  time.interval = 1,
  aggregate.dur = 20,
  FUN = "degree"
)
plot(dynamicDegree)
```

Listing 8: Dynamika betweenness i degree



(a) Dynamika betweenness centrality



(b) Dynamika degree

2.9 Osiągalność węzłów *fwd* i *bkwd*

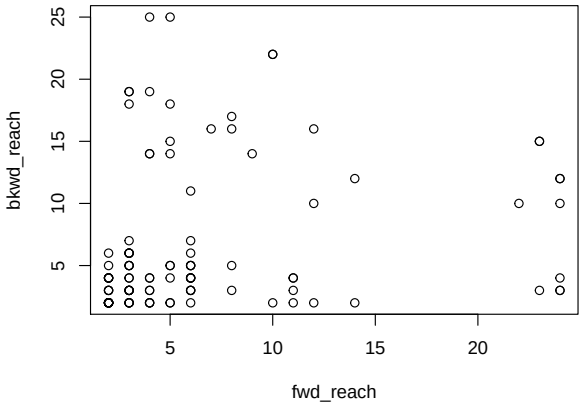
Osiągalność węzłów została opisana w treści ćwiczenia:

> Węzły i krawędzie pojawiają się i znikają, można określić ile węzłów jest osiągalnych z danego węzła w danym momencie, ale także ile węzłów było lub zostanie podłączony do danego węzła w trakcie istnienia sieci. Te przeszłe i przyszłe grupy węzłów mogą być wyznaczone z udziałem funkcji fwd i bckw. Można okreslić czy węzeł zajmuje centralne miejsca w sieci na początku lub na końcu obserwowanego okresu i jaki ma wpływ na dynamikę procesów w sieci. Przykładowo osoba zarażona chorobą stosunkowo wcześnie w epidemii może mieć znacznie większy wpływ na jej rozprzestrzenianie się niż osoba zarażona późno.
>
> Funkcji tReach () domyślnie oblicza rozmiar zestawu osiągalnego do przodu dla danego węzła, ale aby obliczyć zestaw osiągalny wstecz, po prostu określamy kierunek = "bkwd".

```r
fwd_reach <- tReach(tn)
bkwd_reach <- tReach(tn, direction = "bkwd")
```

```
plot(fwd_reach, bkwd_reach)
```

Listing 9: Osiągalność wprzód i w tył

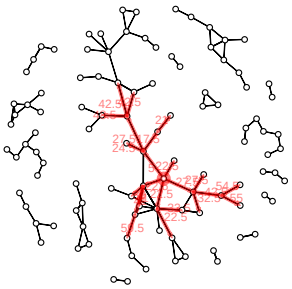


Rysunek 8: Osiągalność krawędzi.

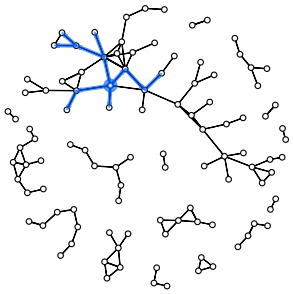
Następnie zwizualizowano wybrane ścieżki forward i backward na grafie.

```
FwdPath <- tPath(
  tn,
  v = 3,
  direction = "fwd"
)
plotPaths(
  tn,
  FwdPath,
  displaylabels = FALSE,
  vertex.col = "white"
)
BkwdPath <- tPath(
  tn,
  v = 3,
  direction = "bkwd",
  type = 'latest.depart'
)
plotPaths(
  tn,
  BkwdPath,
  path.col = rgb(0, 97, 255, max=255, alpha=166),
  displaylabels = FALSE,
  edge.label.col = rgb(0,0,0,0),
  vertex.col = "white"
)
```

Listing 10: Wizualizacja ścieżek fwd i bkwd



(a) Ścieżka FWD



(b) Ścieżka BKWD

3 Zadanie 2 - Model Small-World

Celem zadania było wygenerowanie 10 sieci modelu Small-World [1] ze zmiennym parametrem zagęszczenia połączeń. Każda z wygenerowanej sieci jest migawką sieci dynamicznej z danego dnia (jednostki czasowej). Następnie przeprowadzono badania na sieci.

3.1 Generowanie sieci

Wygenerowano 10 sieci syntetycznych przy użyciu modelu *Small-World* i zapisano jest do plików *generated_networksw_1.txt*, gdzie numer oznacza numer migawki sieci.

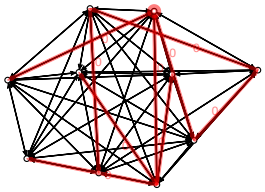
```
setwd("C:/Dev/complex_networks/lab_5/")
library(igraph)
library(sna)
library(tsna)
library(ndtv)

neis = c(1, 2, 1, 3, 0, 1, 4, 3, 2, 1)

for (i in 1:length(neis)) {
  net <- sample_smallworld(dim=1, size=10, nei=neis[i], p=0.1)
  plot(net, vertex.size=6, vertex.label=NA, layout=layout_in_circle)
  path <- paste("./generated_network/sw", i, ".txt", sep ="")
  write_graph(net, path, "edgelist")
}
```

3.2 Wyznaczanie ścieżek

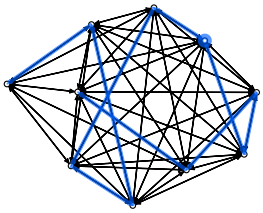
Dla wybranych węzłów z 1, 5 i 10 dnia wyznaczono ścieżki fwd, fwd & bkwd, bakwd i je zwizualizowano na rysunku 12.

```
FwdPath <- tPath(dnet, v = 1,direction = "fwd", start=1)
plotPaths(dnet,FwdPath,displaylabels = FALSE,vertex.col = "white")
BkwdPath <- tPath(dnet,v = 1,direction = "bkwd",type ='latest.depart',start=1)
plotPaths(dnet,BkwdPath,path.col = rgb(0, 97, 255, max=255, alpha=166),displaylabels = FALSE,edge.
    label.col = rgb(0,0,0,0),vertex.col = "white")

FwdPath <- tPath(dnet, v = 1,direction = "fwd", start=5)
plotPaths(dnet,FwdPath,displaylabels = FALSE,vertex.col = "white")
BkwdPath <- tPath(dnet,v = 1,direction = "bkwd",type ='latest.depart',start=5)
```

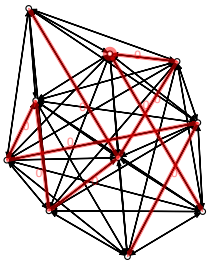
```
plotPaths(dnet,BkwdPath,path.col = rgb(0, 97, 255, max=255, alpha=166),displaylabels = FALSE,edge.
    label.col = rgb(0,0,0,0),vertex.col = "white")


FwdPath <- tPath(dnet, v = 1,direction = "fwd", start=10)
plotPaths(dnet,FwdPath,displaylabels = FALSE,vertex.col = "white")
BkwdPath <- tPath(dnet,v = 1,direction = "bkwd",type ='latest.depart',start=10)
plotPaths(dnet,BkwdPath,path.col = rgb(0, 97, 255, max=255, alpha=166),displaylabels = FALSE,edge.
    label.col = rgb(0,0,0,0),vertex.col = "white")
```



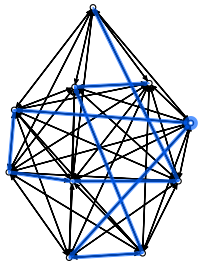
(a) FWD dla węzła 1 w dniu 1



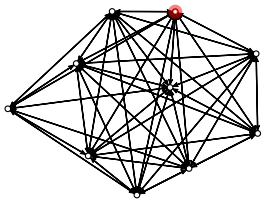
(b) BKWD dla węzła 1 w dniu 1



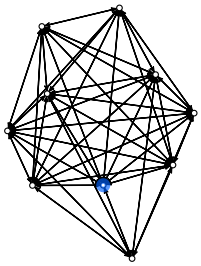
(a) BKWD dla węzła 1 w dniu 5



(b) BKWD dla węzła 1 w dniu 5



(a) FWD dla węzła 1 w dniu 10



(b) BKWD dla węzła 1 w dniu 10

Rysunek 12: Temporalne ścieżki

3.3 Temporal degree

Wyznaczono temporal degree dla węzłów w różnych momentach czasu za pomocą komendy `tDegree(dnet)`.

```
> tDegree(dnet)
Time Series:
Start = 0
End = 10
Frequency = 1
    1  2  3  4  5  6  7  8  9 10
 0  4  4  4  4  4  4  4  4  4  4
 1  6  8 14  6  8  8  8  4  8 10
 2  4  6  4  4  2  4  4  4  4  4
 3 12 16 12 12 12 12 10 14 10 10
 4  2  4  2  4  4  4  4  6  6  4
 5  4  4  6  2  4  4  4  4  4  4
 6 18 16 16 12 18 16 14 18 16 16
 7 14 12 10 14 12 12 10 14 10 12
 8  8 10 10  8  8  8  6  8  8  6
 9  4  4  4  2  4  6  2  4  4  6
10 NA NA NA NA NA NA NA NA NA NA
```

3.4 Animacja

Wygenerowano animację zmian sieci.

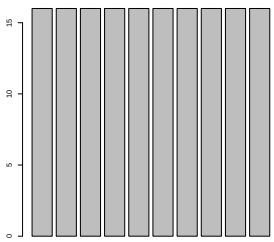
```
compute.animation(
  dnet,
  animation.mode = "kamadakawai",
  slice.par = list(
    start = 1,
    end = 10,
    interval = 1,
    aggregate.dur = 1,
    rule = "any"
  )
)
render.d3movie(
  dnet,
  displaylabels = FALSE
```

```
)
```

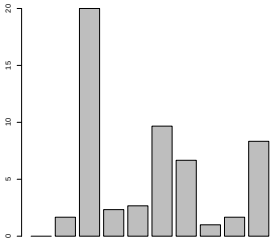
3.5 Rozkład betweenness

Wyznaczono rozkład betweenness dla poszczególnych okien czasowych oraz dynamikę zmiany betweenness.

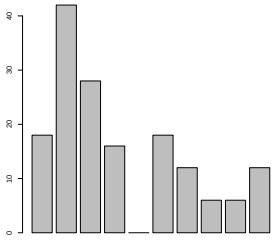
```
dynamicBetweenness <- tSnaStats(
  dnet,
  snafun = "centralization",
  start = 1,
  end = 10,
  time.interval = 1,
  aggregate.dur = 1,
  FUN = "betweenness"
)
plot(dynamicBetweenness)

for (i in c(0:9)) {
  path <- paste("./output/betweenness_", i, ".pdf", sep ="")
  pdf(file=path)
  barplot(betweenness(network.collapse(dnet, at=i)))
  dev.off()
}
```



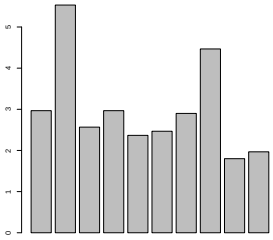
(a) Betweenness w dniu 1



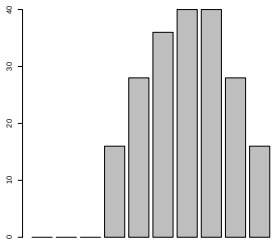
(b) Betweenness w dniu 2



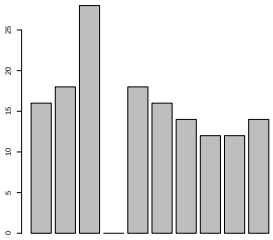
(a) Betweenness w dniu 3



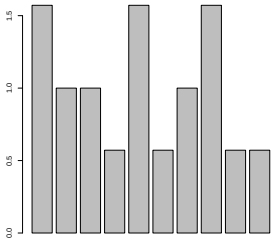
(b) Betweenness w dniu 4



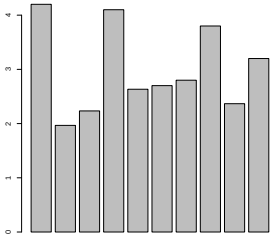
(a) Betweenness w dniu 5



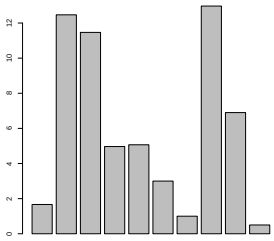
(b) Betweenness w dniu 6



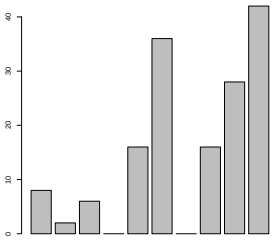
(a) Betweenness w dniu 7



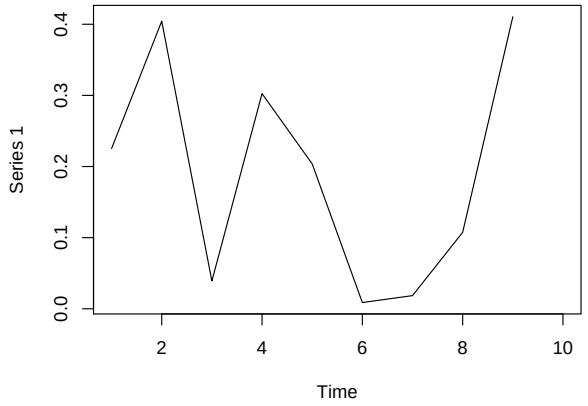
(b) Betweenness w dniu 8



(a) Betweenness w dniu 9



(b) Betweenness w dniu 10



Rysunek 18: Dynamiczne betweenness.

4 Zadanie 3

W zadaniu 3 użyto bazy danych CollegeMsg temporal network [4]. Do zadania trzeciego nie został dostarczony plik *import.zip* z podziałem wiadomości na okna czasowe. Oznaczenia kolumn i parametrtrów, takie jak *onset*, *terminus*, *head*, *tail*, *spells* nie są mi znane. Czas włożony w studiowanie tych pojęć i tego rodzaju reprezentacji nie byłby warty efektów.

Podjąłem się naiwnego rozwiązania zadania, jednak bezskutecznie. Przekształcając kolumnę z typu *unix timestamp* na typ *Data* umożliwiający wyciąganie informacji, np. o miesiącu.

```
setwd("C:/Dev/complex_networks/lab_5/")
library(igraph)
library(sna)
library(tsna)
library(ndtv)

df <- read.table("CollegeMsg.txt", sep=" ", header = FALSE)
df$V3 = as.Date(as.POSIXct(df$V3, origin="1970-01-01"))
df$month = months(df$V3)
> df
     V1  V2                  V3    month
1     1   2 2004-04-15 02:00:00 kwiecien
2     3   4 2004-04-16 02:00:00 kwiecien
3     5   2 2004-04-19 02:00:00 kwiecien
4     6   7 2004-04-20 02:00:00 kwiecien
5     8   7 2004-04-20 02:00:00 kwiecien
```

Literatura

[1] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, June 1998.

[2] Daniel A McFarland. Student resistance: How the formal and informal organization of classrooms facilitate everyday forms of student defiance. *American journal of Sociology*, 107(3):612–678, 2001.

[3] James Moody. Static representations of dynamic networks. *Duke Population Research Institute On-line Working Paper Series*, 2008.

[4] Pietro Panzarasa, Tore Opsahl, and Kathleen M Carley. Patterns and dynamics of users' behavior and interaction: Network analysis of an online community. *Journal of the American Society for Information Science and Technology*, 60(5):911–932, 2009.