SNA toolbox: an introduction

Justyna P. Zwolak

April 11, 2018

Introduction

This document is intended as an introductory guide to SNA analysis. As such, it does not present or discuss any statistical analysis and modeling that would use the calculated network measures. It only shows how to calculate various network metrics and how to visualize social networks. In order to conduct a network analysis one need in addition to this file a data set, i.e., at least one file with edge lists; a file with information about the nodes and the `SNAfunctions(G).R`, where all functions handling the work behind the scenes are defined.

Before starting the analysis, make sure that the file structure on your computer is compatible with the code in this document. In particular, this code assumes that the `SNA analysis.Rmd` file is in the same folder as the `SNAfunctions(G).R` file and the `Data` folder. The data structure should be as follows:

```
main
|- SNA analysis.Rmd
|- SNAfunctions(G).R
|- Data
    |- Edge_lists
        |- s1c1-A.csv
        |- s1c1-B.csv
        |- ...
    |- Nodes.csv
```

where the `s1c1-A.csv`, `s1c1-B.csv`, etc., are the csv files with edge list from different collections.

Now, once the file structure is set, I will set the current working directory to make the path operations easier. I will also initiate packages that I will be using throughout the analysis.

```
## Warning: package 'igraph' was built under R version 3.4.4
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
## Loading required package: survival
```

```
## tnet: Analysis of Weighted, Two-mode, and Longitudinal networks.
## Type ?tnet for help.
```

I start by creating a data frame with the nodes' information to summarize the data.

```r
my.data <- read.csv("Data/Nodes.csv", header = TRUE, na.strings = c("NA", "", "#N/A"))
colnames(my.data)
```

```
## [1] "Ids"         "Status"      "Gender"      "Ethnic.Grp" "Acad.Plan"
## [6] "Sec.F"       "Sec.S"       "Grade.F"
```

As can be seen, my data frame contains information about the Ids (two-digit identifiers), status (one of five possible levels: ST/LA/TA/INS/Other, where "Other" was assigned to individuals who did not appear on the initial roster), gender (M/F), ethnic group, academic plan, section in the fall semester, section (if any) in the spring semester and final grade. Here are some numbers to summarize the data. Out of the 117 students who took the fall section of the course the total of 85 took the consecutive course in the spring.

```
sum(my.data$Status == "ST")
```

```
## [1] 117
```

```
sum(my.data$Sec.S %in% c("A", "B") & my.data$Status == "ST", na.rm = TRUE)
```

```
## [1] 85
```

The enrollment and staff in numbers are presented below:

```
table(my.data$Sec.F, my.data$Status)
```

```
##
##     INS LA Other ST TA
##   A   1  3     3 55  1
##   B   1  3     1 62  2
```

To use in the analysis the network metrics one needs to create a data frame containing all the in- and out-of-class centralities for both sections. However, before doing any combined analysis on both section I need to get a sense for whether the two sections are similar enough so that it makes sense to aggregate them for analysis. To do so, I will compare basic network descriptive.

The in-class networks: comparison between sections

Response rates for all surveys for both sections:

```
resp.rate = c(97, 86, 78, 81, 77, 96, 84, 89, 80, 86)
section = c(rep("A", 5), rep("B", 5))
resp.rate.all = data.frame(resp.rate, section)

data.frame(id = c("A_mean", "A_sd", "B_mean", "B_sd"),
           values = c(mean(resp.rate[1:5]), sd(resp.rate[1:5]),
                      mean(resp.rate[6:10]), sd(resp.rate[6:10])))
```

```
##       id    values
## 1 A_mean 83.800000
## 2   A_sd  8.167007
## 3 B_mean 87.000000
## 4   B_sd  6.000000
```

```
kruskal.test(resp.rate ~ section, data = resp.rate.all)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  resp.rate by section
## Kruskal-Wallis chi-squared = 0.70244, df = 1, p-value = 0.402
```

Kruskal-Wallis shows no statistically significant differences in response rates between the two sections.

Now I want to compare the two groups in terms of student demographic information. I start with gender.

```
gender = xtabs(~ Sec.F + Gender, data=my.data[my.data$Status=="ST",])
gender
```

```
##      Gender
## Sec.F  F  M
##     A 22 33
##     B 33 29
```

```
chisq.test(gender)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  gender
## X-squared = 1.5501, df = 1, p-value = 0.2131
```

Now I look the the ethnicity distributions.

```
demog = xtabs(~ Sec.F + Ethnic.Grp, data=my.data[my.data$Status=="ST",])
demog
```

```
##      Ethnic.Grp
## Sec.F ASIAN BLACK CNRETH HISPA TWOMORE WHITE
##     A     3     4      0    43       0     5
##     B     6     3      1    40       3     9
```

```
fisher.test(demog)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  demog
## p-value = 0.3385
## alternative hypothesis: two.sided
```

And finally at the major.

```
major = xtabs(~ Sec.F + Acad.Plan, data=my.data[my.data$Status=="ST",])
major
```

```
##      Acad.Plan
## Sec.F ACCT:BACC BIOL:BS BIOMEG:BS CHEM:BA CHEM:BS CIVLEG:BS COMPEG:BS
##     A         0       8         1       1       0         6         5
##     B         1      11         3       3       1         1         1
##      Acad.Plan
## Sec.F COMPSC:BS DUAL ELEG:BS ENVEG:BS GEOSC:BS HSA:BHSA IT:BS MATH:BA
##     A         9   11       0        1        1        0     1       1
##     B         5   25       2        1        0        1     0       0
##      Acad.Plan
## Sec.F MECHEG:BS MRNBIO:BS PSYC:BA TRANSIENT
##     A         5         1       3         1
##     B         4         1       2         0
```

```
fisher.test(major)
```

```
##
##  Fisher's Exact Test for Count Data
```

```
##
## data:  major
## p-value = 0.06882
## alternative hypothesis: two.sided
```

The in-class networks: network comparison

I start by reading the data from both sections:

```
files.list <- list.files("Data/Edge_lists", full.names = TRUE)
nodes <- read.csv("Data/Nodes.csv", header = TRUE, na.strings = c("NA", ""),
                  stringsAsFactors = FALSE)
sna.data <- read.csv("Data/Nodes.csv", header = TRUE,
                     na.strings = c("NA", "", "#N/A"))
```

Rather than analyzing each network separately, I will create a list of graph lists for each section. Working with lists will help to limit the number of the repeated code.

```
dat.A <- reading.graph(1)
dat.B <- reading.graph(2)
dat.A
```

```
## [[1]]
## IGRAPH 956202b DNW- 63 280 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from 956202b (vertex names):
##  [1] 16 ->119 16 ->114 16 ->112 16 ->20  16 ->76  11 ->92  11 ->103
##  [8] 11 ->100 11 ->79  76 ->185 76 ->193 76 ->114 76 ->112 76 ->119
## [15] 76 ->16  76 ->20  76 ->195 114->193 114->119 114->112 114->16
## [22] 114->76  114->195 27 ->44  27 ->63  6  ->96  6  ->99  6  ->185
## [29] 6  ->105 66 ->114 66 ->195 66 ->19  118->5   118->86  118->12
## [36] 118->186 118->188 118->88  118->195 119->16  119->76  119->185
## + ... omitted several edges
##
## [[2]]
## IGRAPH 9654b8b DNW- 63 294 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from 9654b8b (vertex names):
##  [1] 31 ->188 31 ->193 31 ->111 31 ->16  31 ->3   114->63  114->48
##  [8] 114->188 114->185 114->193 114->105 114->65  114->119 114->11
## [15] 63 ->12  63 ->26  111->3   111->195 111->193 111->16  111->6
## [22] 26 ->12  26 ->193 103->11  103->34  103->6   103->193 103->55
## [29] 22 ->21  22 ->96  22 ->40  22 ->4   22 ->5   22 ->193 6  ->188
## [36] 6  ->195 6  ->193 6  ->111 6  ->21  6  ->105 6  ->95  21 ->5
## + ... omitted several edges
##
## [[3]]
## IGRAPH 3032f0b DNW- 63 245 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from 3032f0b (vertex names):
##  [1] 118->55  118->19  118->88  118->27  80 ->48  80 ->40  80 ->112
##  [8] 80 ->110 26 ->12  26 ->63  12 ->188 12 ->26  12 ->63  12 ->68
## [15] 12 ->186 12 ->195 12 ->193 114->105 114->119 114->195 114->186
## [22] 114->185 114->188 114->96  114->32  48 ->195 32 ->31  32 ->118
## [29] 32 ->195 32 ->193 32 ->105 32 ->65  32 ->188 32 ->105 32 ->114
## [36] 32 ->33  32 ->11  113->195 113->193 113->20  113->112 113->75
## + ... omitted several edges
##
## [[4]]
## IGRAPH d9111b2 DNW- 63 276 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from d9111b2 (vertex names):
##  [1] 3 ->16  4 ->193 4 ->95  4 ->105 4 ->48  4 ->79  4 ->188 4 ->186
##  [9] 5 ->12  5 ->195 5 ->193 5 ->185 5 ->26  5 ->33  5 ->11  6 ->72
## [17] 6 ->16  6 ->3   11->195 11->193 11->185 11->33  11->26  11->6
## [25] 12->188 12->186 12->96  12->114 12->195 12->193 12->88  12->185
## [33] 16->114 16->6   16->55  16->3   19->100 19->80  19->112 19->68
## [41] 19->195 20->63  20->55  20->78  20->93  21->32  21->96  21->195
## + ... omitted several edges
##
## [[5]]
## IGRAPH 39b6f95 DNW- 63 249 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from 39b6f95 (vertex names):
##  [1] 3 ->16  4 ->106 4 ->78  4 ->100 4 ->193 4 ->188 5 ->27  5 ->12
##  [9] 5 ->72  5 ->195 5 ->193 5 ->188 5 ->186 5 ->185 6 ->111 6 ->72
## [17] 6 ->105 6 ->21  11->195 11->193 11->188 11->186 11->185 12->188
## [25] 12->5   12->27  12->72  12->195 12->193 12->186 12->185 16->12
## [33] 16->186 19->80  19->55  19->119 19->99  19->193 19->195 20->70
## [41] 20->88  20->16  20->78  20->93  26->117 26->93  27->5   27->12
## + ... omitted several edges
```

Now I will generate lists of graphs with only relevant nodes. In particular, presently each network includes all the nodes that were coded as “Other” (i.e., individuals who were not on the roster but appeared on the survey). I use the `remove.disconnected` function to remove all disconnected “Others” nodes from my networks while keeping the disconnected students in it. Note that this operation does not affect the numbers of edges.

```
dat.A.rel <-lapply(dat.A, remove.disconnected)
dat.B.rel <-lapply(dat.B, remove.disconnected)
dat.A.rel
```

```
## [[1]]
## IGRAPH 44f422f DNW- 60 280 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from 44f422f (vertex names):
##  [1] 16 ->119 16 ->114 16 ->112 16 ->20  16 ->76  11 ->92  11 ->103
##  [8] 11 ->100 11 ->79  76 ->185 76 ->193 76 ->114 76 ->112 76 ->119
## [15] 76 ->16  76 ->20  76 ->195 114->193 114->119 114->112 114->16
## [22] 114->76  114->195 27 ->44  27 ->63  6  ->96  6  ->99  6  ->185
## [29] 6  ->105 66 ->114 66 ->195 66 ->19  118->5   118->86  118->12
## [36] 118->186 118->188 118->88  118->195 119->16  119->76  119->185
## + ... omitted several edges
##
## [[2]]
## IGRAPH b102361 DNW- 60 294 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from b102361 (vertex names):
##  [1] 31 ->188 31 ->193 31 ->111 31 ->16  31 ->3   114->63  114->48
##  [8] 114->188 114->185 114->193 114->105 114->65  114->119 114->11
## [15] 63 ->12  63 ->26  111->3   111->195 111->193 111->16  111->6
## [22] 26 ->12  26 ->193 103->11  103->34  103->6   103->193 103->55
## [29] 22 ->21  22 ->96  22 ->40  22 ->4   22 ->5   22 ->193 6  ->188
## [36] 6  ->195 6  ->193 6  ->111 6  ->21  6  ->105 6  ->95  21 ->5
## + ... omitted several edges
##
## [[3]]
## IGRAPH 3e4043d DNW- 60 245 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from 3e4043d (vertex names):
##  [1] 118->55  118->19  118->88  118->27  80 ->48  80 ->40  80 ->112
##  [8] 80 ->110 26 ->12  26 ->63  12 ->188 12 ->26  12 ->63  12 ->68
## [15] 12 ->186 12 ->195 12 ->193 114->105 114->119 114->195 114->186
## [22] 114->185 114->188 114->96  114->32  48 ->195 32 ->31  32 ->118
## [29] 32 ->195 32 ->193 32 ->105 32 ->65  32 ->188 32 ->105 32 ->114
## [36] 32 ->33  32 ->11  113->195 113->193 113->20  113->112 113->75
## + ... omitted several edges
##
## [[4]]
## IGRAPH 5472e70 DNW- 60 276 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from 5472e70 (vertex names):
##  [1] 3 ->16  4 ->193 4 ->95  4 ->105 4 ->48  4 ->79  4 ->188 4 ->186
##  [9] 5 ->12  5 ->195 5 ->193 5 ->185 5 ->26  5 ->33  5 ->11  6 ->72
## [17] 6 ->16  6 ->3   11->195 11->193 11->185 11->33  11->26  11->6
## [25] 12->188 12->186 12->96  12->114 12->195 12->193 12->88  12->185
## [33] 16->114 16->6   16->55  16->3   19->100 19->80  19->112 19->68
## [41] 19->195 20->63  20->55  20->78  20->93  21->32  21->96  21->195
## + ... omitted several edges
##
## [[5]]
## IGRAPH bef6560 DNW- 60 249 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from bef6560 (vertex names):
##  [1] 3 ->16  4 ->106 4 ->78  4 ->100 4 ->193 4 ->188 5 ->27  5 ->12
##  [9] 5 ->72  5 ->195 5 ->193 5 ->188 5 ->186 5 ->185 6 ->111 6 ->72
## [17] 6 ->105 6 ->21  11->195 11->193 11->188 11->186 11->185 12->188
## [25] 12->5   12->27  12->72  12->195 12->193 12->186 12->185 16->12
## [33] 16->186 19->80  19->55  19->119 19->99  19->193 19->195 20->70
## [41] 20->88  20->16  20->78  20->93  26->117 26->93  27->5   27->12
## + ... omitted several edges
```

In a similar way I can remove all instructional staff using the `remove.instr` function. Note that now both the number of nodes and the number of edges has changed.

```
dat.A.st <-lapply(dat.A.rel, remove.instr)
dat.B.st <-lapply(dat.B.rel, remove.instr)
dat.A.st
```

```
## [[1]]
## IGRAPH e596bc4 DNW- 55 207 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from e596bc4 (vertex names):
##  [1] 16 ->119 16 ->114 16 ->112 16 ->20  16 ->76  11 ->92  11 ->103
##  [8] 11 ->100 11 ->79  76 ->114 76 ->112 76 ->119 76 ->16  76 ->20
## [15] 114->119 114->112 114->16  114->76  27 ->44  27 ->63  6  ->96
## [22] 6  ->99  6  ->105 66 ->114 66 ->19  118->5   118->86  118->12
## [29] 118->88  119->16  119->76  119->112 119->114 4  ->22  4  ->80
## [36] 4  ->48  96 ->95  96 ->6   96 ->38  96 ->99  96 ->105 96 ->33
## + ... omitted several edges
##
## [[2]]
## IGRAPH 54e274d DNW- 55 203 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from 54e274d (vertex names):
##  [1] 31 ->111 31 ->16  31 ->3   114->63  114->48  114->105 114->65
##  [8] 114->119 114->11  63 ->12  63 ->26  111->3   111->16  111->6
## [15] 26 ->12  103->11  103->34  103->6   103->55  22 ->21  22 ->96
## [22] 22 ->40  22 ->4   22 ->5   6  ->111 6  ->21  6  ->105 6  ->95
## [29] 21 ->5   21 ->6   21 ->22  21 ->96  21 ->72  112->76  112->5
## [36] 112->96  112->80  112->113 112->119 5  ->76  5  ->12  5  ->112
## + ... omitted several edges
##
## [[3]]
## IGRAPH 0aec958 DNW- 55 161 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from 0aec958 (vertex names):
##  [1] 118->55  118->19  118->88  118->27  80 ->48  80 ->40  80 ->112
##  [8] 80 ->110 26 ->12  26 ->63  12 ->26  12 ->63  12 ->68  114->105
## [15] 114->119 114->96  114->32  32 ->31  32 ->118 32 ->105 32 ->65
## [22] 32 ->105 32 ->114 32 ->33  32 ->11  113->20  113->112 113->75
## [29] 96 ->112 96 ->38  96 ->99  96 ->5   96 ->100 96 ->119 20 ->75
## [36] 20 ->113 75 ->20  75 ->92  75 ->113 66 ->34  66 ->4   66 ->70
## + ... omitted several edges
##
## [[4]]
## IGRAPH a4f762f DNW- 55 184 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from a4f762f (vertex names):
##  [1] 3 ->16  4 ->95  4 ->105 4 ->48  4 ->79  5 ->12  5 ->26  5 ->33
##  [9] 5 ->11  6 ->72  6 ->16  6 ->3   11->33  11->26  11->6   12->96
## [17] 12->114 12->88  16->114 16->6   16->55  16->3   19->100 19->80
## [25] 19->112 19->68  20->63  20->55  20->78  20->93  21->32  21->96
## [33] 21->40  22->110 22->106 22->31  26->5   26->11  26->33  27->118
## [41] 27->99  27->103 27->19  31->110 31->22  31->106 32->31  32->38
## + ... omitted several edges
##
## [[5]]
## IGRAPH 51c99ca DNW- 55 170 --
## + attr: name (v/c), Status (v/c), Gender (v/c), Ethnic.Grp (v/c),
## | Acad.Plan (v/c), Sec.F (v/c), Sec.S (v/c), Grade.F (v/n), weight
## | (e/n)
## + edges from 51c99ca (vertex names):
##  [1] 3 ->16  4 ->106 4 ->78  4 ->100 5 ->27  5 ->12  5 ->72  6 ->111
##  [9] 6 ->72  6 ->105 6 ->21  12->5   12->27  12->72  16->12  19->80
## [17] 19->55  19->119 19->99  20->70  20->88  20->16  20->78  20->93
## [25] 26->117 26->93  27->5   27->12  27->72  27->19  31->112 31->32
## [33] 31->113 31->63  32->63  32->113 32->31  32->112 33->95  33->105
## [41] 33->65  34->3   34->86  38->100 38->96  38->119 38->99  48->118
## + ... omitted several edges
```

Density

Graph density is a measure of how well connected the network is as a whole. It is calculated as the portion of the potential connections in a network that are actual connections. For a directed network, density is defined as:

$$\Delta = \frac{\ell}{n(n-1)},$$

where ℓ is the number of all ties in the network and n is the number of nodes in the network. Density takes values between 0 (network without ties) and 1 (fully connected network).

In my analysis, I use the standard `graph.density` function from `igraph` package and, by default, a binary version of ties. Also, unless otherwise stated, I use the network with removed disconnected "Others" but I keep the instructional staff.

```r
data.frame(Id = c("SNA1", "SNA2", "SNA3", "SNA4", "SNA5"),
           F15A = round(sapply(dat.A.rel, graph.density, loops=FALSE), 3),
           F15B = round(sapply(dat.B.rel, graph.density, loops=FALSE), 3))
```

```
##     Id  F15A  F15B
## 1 SNA1 0.079 0.068
## 2 SNA2 0.083 0.079
## 3 SNA3 0.069 0.090
## 4 SNA4 0.078 0.080
## 5 SNA5 0.070 0.083
```

Diameter

Diameter is the length of the longest path between two nodes, where path is defined as a sequence of edges connecting a sequence of distinct nodes. It provides information about the span of a network. The standard version of the `diameter` function from the `igraph` package allows to calculate the weighted diameter of a network. Note, that by default in `igraph` weights are treated as a cost rather than an advantage, i.e., it is three times as hard to get to a node via an edge of weight 3 than via an edge of weight 1. Since the weight from our survey have the opposite meaning, the numbers need to be inverted for calculating weighted diameter. To do so, we will use the `mod.diameter` function. To use the binary edges one has to set the `weights` argument to `NA`. For comparison, I calculate the weighted (`X.wgt`) diameter, the diameter with inverted weight (`X.inv.wgt`) and the binary version (`X.flat`).

```r
data.frame(Id = c("SNA1", "SNA2", "SNA3", "SNA4", "SNA5"),
           A.wgt = sapply(dat.A.rel, diameter),
           B.wgt = sapply(dat.B.rel, diameter),
           A.inv.wgt = round(sapply(dat.A.rel, mod.diameter), 1),
           B.inv.wgt = round(sapply(dat.B.rel, mod.diameter), 1),
           A.flat = sapply(dat.A.rel, diameter, weights = NA),
           B.flat = sapply(dat.B.rel, diameter, weights = NA))
```

```
##     Id A.wgt B.wgt A.inv.wgt B.inv.wgt A.flat B.flat
## 1 SNA1    27    22       4.0       4.3     10      9
## 2 SNA2    15    19       3.7       3.5      7      8
## 3 SNA3    20    19       3.0       3.3      8      7
## 4 SNA4    22    22       4.0       4.3      9      9
## 5 SNA5    23    24       5.2       3.5     10      9
```

Average path length

The average path length (APL) is the shortest path between two nodes, averaged over all pairs of nodes. It is an indicator of how close together nodes are to one another. I calculate the average path length using the standard `mean_distance` from `igraph`. Note that in this approach the weights are ignored by default. For comparison, I calculate the APL for networks with (`X.all`) and without (`X.st`) instructors.

```r
data.frame(Id = c("SNA1", "SNA2", "SNA3", "SNA4", "SNA5"),
           A.all = round(sapply(dat.A.rel, mean_distance), 1),
           B.all = round(sapply(dat.B.rel, mean_distance), 1),
           A.st = round(sapply(dat.A.st, mean_distance), 1),
           B.st = round(sapply(dat.B.st, mean_distance), 1))
```

```
##     Id A.all B.all A.st B.st
## 1 SNA1   3.7   3.1  4.0  3.3
## 2 SNA2   3.1   3.3  3.2  3.5
## 3 SNA3   2.9   3.2  3.1  3.4
## 4 SNA4   3.5   3.7  3.6  3.9
## 5 SNA5   3.3   3.7  3.5  3.8
```

Reciprocity

Reciprocity is a tendency of pairs of nodes to form mutual connections between each other. In network science, reciprocity is a measure of the likelihood of vertices in a directed network to be mutually linked. It is calculated for unweighted graphs as: directed network, density is defined as:

$$\rho = \frac{\ell^{\leftrightarrow}}{\ell},$$

where $\ell^{\leftrightarrow}$ is the number of mutual edges ℓ is the number of all edges in the network. Like density, reciprocity takes values between 0 (non of the ties is returned) and 1 (all tied are mutual). I use the standard `reciprocity` function from the `igraph` package. For comparison I present networks without (`X.st`) and with (`X.all`) instructors. Since instructors did not take the survey but were names by many students, values in the latter are about 20%-35% lower.

```
a <- data.frame(Id = c("SNA1", "SNA2", "SNA3", "SNA4", "SNA5"),
           A.st = round(sapply(dat.A.st, reciprocity), 2),
           B.st = round(sapply(dat.B.st, reciprocity), 2),
           A.all = round(sapply(dat.A.rel, reciprocity), 2),
           B.all = round(sapply(dat.B.rel, reciprocity), 2))
```

Transitivity

Transitivity refers to the extent to which the relation between two nodes is transitive, i.e., two connected nodes have a common neighbor ("a friend of my friend is also my friend"). For transitivity, I use the `clustering_w()` function from package `tnet` as it allows more flexibility when deciding how to deal with weighted ties. In particular, my networks are directed and weighted and I want to compare the transitivity when the weights are treated as binary (`"bi"` method) and with the arithmetic mean to control the switch off directed ties into undirected (`"am"` method). Other options for switching off the directionality of ties are: `"gm"` for the geometric mean, `"mi"` for the minimum method and `"ma"` for the maximum method. Function graph `graph.tnet` converts the `igraph` objects into a format compatible with `tnet` package.

```
tr = data.frame(Id = c("SNA1", "SNA2", "SNA3", "SNA4", "SNA5"),
           A.bi = round(sapply(graph.tnet(dat.A.rel), clustering_w, measure = "bi"), 2),
           A.am = round(sapply(graph.tnet(dat.A.rel), clustering_w, measure = "am"), 2),
           B.bi = round(sapply(graph.tnet(dat.B.rel), clustering_w, measure = "bi"), 2),
           B.am = round(sapply(graph.tnet(dat.B.rel), clustering_w, measure = "am"), 2))
tr
```

```
##     Id A.bi A.am B.bi B.am
## 1 SNA1 0.68 0.69 0.70 0.73
## 2 SNA2 0.40 0.42 0.50 0.52
## 3 SNA3 0.50 0.52 0.51 0.53
## 4 SNA4 0.48 0.51 0.56 0.58
## 5 SNA5 0.49 0.50 0.52 0.54
```

If the ratio between the weighted clustering coefficient (weighted transitivity) and the binary coefficient is higher than 1, it can be argued that triplets made up by strong ties are more likely to be closed than triplets made up by weak ties. On the contrary, it can be argued that triplets made of by weak ties are more likely to be closed than those made up by strong ties if the ratio is less than 1. In our case:

```
data.frame(Id =  c("SNA1", "SNA2", "SNA3", "SNA4", "SNA5"),
           A = round(tr$A.am/tr$A.bi, 3),
           B = round(tr$B.am/tr$B.bi, 3))
```

```
##     Id     A     B
## 1 SNA1 1.015 1.043
## 2 SNA2 1.050 1.040
## 3 SNA3 1.040 1.039
## 4 SNA4 1.062 1.036
## 5 SNA5 1.020 1.038
```