Werken met netwerkdata in R

Robin Khalfa

2025-05-21

1. Inleiding

In sociaalwetenschappelijk onderzoek behelst sociale netwerk analyse een belangrijk raamwerk om de relaties tussen actoren te bestuderen. In tegenstelling tot traditionele datamodellen die focussen op geïsoleerde eenheden, bieden netwerkdata inzicht in hoe entiteiten (zoals personen, organisaties of instituties) met elkaar verbonden zijn. Dit laat toe om onderzoeksvragen te beantwoorden over centrale posities, structurele hiërarchieën, groepsvorming, informatiestromen en nog veel meer.

Doelstelling van dit document

Deze RMarkdown-handleiding heeft als doel om onderzoekers op een gestructureerde en reproduceerbare manier wegwijs te maken in het werken met netwerkdata in R. De nadruk ligt op praktische dataverwerking en -voorbereiding, eerder dan op geavanceerde netwerkmodellen of visualisaties. Het document start vanuit ruwe netwerkdata in de vorm van node- en edgetabellen, en gaat stap voor stap door het proces van inlezen, structureren, manipuleren en voorbereiden van deze data voor latere netwerkstatistische analyses.

Structuur van dit document

Deze notebook is als volgt opgebouwd:

1. **Belangrijkste packages in R**: We starten met een overzicht van de meest relevante R-packages voor sociale netwerk analyse.
2. **Inladen van netwerkdata in R:** We tonen hoe netwerkdata correct kunnen worden ingelezen in R, en wat de verwachte structuur is van deze data.
3. **Inspecteren en manipuleren/cleanen van netwerkdata:** We bekijken hoe we netwerkdata kunnen inspecteren in R en hoe je attributen aan de data toevoegt, filtert, verbindingen verwijdert of subgroepen selecteert, alsook hoe je kan omgaan met ontbrekende waarden en data.
4. **Netwerkobjecten:** Hier bekijken we hoe we netwerkdata kunnen omzetten naar netwerkobjecten en hoe we attributen kunnen toevoegen aan netwerkobjecten.

2. Belangrijkste packages voor sociale netwerkanalyse in R

De functionaliteit van R wordt aanzienlijk uitgebreid door het laden van externe pakketten die door onderzoekers zijn ontwikkeld en gedeeld met de R-community. Deze pakketten bevatten functies die specifieke taken ondersteunen, zoals het inladen, analyseren en visualiseren van netwerkdata.

In deze cursus maken we hoofdzakelijk gebruik van de volgende pakketten:

`igraph` (Csardi & Nepusz, 2006)

- Eén van de meest uitgebreide en zelfstandige pakketten voor netwerkanalyse. Ondersteunt het aanmaken, manipuleren, analyseren én visualiseren van netwerken.
- Omvat een breed scala aan netwerkmaatstaven (graad, centraliteit, clustering, enz.) en is ook heel schaalbaar
- Werkt onafhankelijk van andere netwerkpakketten en gebruikt eigen datastructuren (`igraph`-objecten). Niet volledig compatibel met `network`-objecten.

```
#install.packages("igraph")
library(igraph)
```

```
## Warning: package 'igraph' was built under R version 4.3.3
```

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

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

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

`sna` (Butts, 2023b)

- Richt zich op structurele netwerkanalyse in de sociologische traditie
- Biedt veel klassieke SNA-maatstaven
- Wordt meestal gebruikt in combinatie met het network-package, aangezien het network-objecten vereist

```
#install.packages("sna")
library(sna)
```

```
## Warning: package 'sna' was built under R version 4.3.3
```

```
## Loading required package: statnet.common
```

```
## Warning: package 'statnet.common' was built under R version 4.3.3
```

```
##
## Attaching package: 'statnet.common'
```

```
## The following objects are masked from 'package:base':
##
##     attr, order
```

```
## Loading required package: network
```

```
## Warning: package 'network' was built under R version 4.3.3
```

```
##
## 'network' 1.19.0 (2024-12-08), part of the Statnet Project
## * 'news(package="network")' for changes since last version
## * 'citation("network")' for citation information
## * 'https://statnet.org' for help, support, and other information
```

```
##
## Attaching package: 'network'
```

```
## The following objects are masked from 'package:igraph':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges, delete.vertices,
##     get.edge.attribute, get.edges, get.vertex.attribute, is.bipartite,
##     is.directed, list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute
```

```
## sna: Tools for Social Network Analysis
## Version 2.8 created on 2024-09-07.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.
```

```
##
## Attaching package: 'sna'
```

```
## The following objects are masked from 'package:igraph':
##
##     betweenness, bonpow, closeness, components, degree, dyad.census,
##     evcent, hierarchy, is.connected, neighborhood, triad.census
```

`network`(Butts, 2015)

- Biedt een datastructuur voor netwerken in de vorm van network-objecten.
- Flexibel bij het toekennen van attributen aan knopen en randen.
- Basisobject voor sna, compatibel met andere packages zoals networkDynamic, ndtv, en ergm.

```
#install.packages("network") # wellicht niet nodig hier aangezien het installeren van sna vaak ook de i
library(network)
```

`tidygraph` (Pedersen)

- Maakt het mogelijk om netwerkdata te manipuleren volgens de logica van de tidyverse, o.a. met dplyr-achtige functies.
- Zeer intuïtieve syntax voor gebruikers vertrouwd met tidyverse.
- Werkt met tbl_graph objecten, intern gebaseerd op igraph.

```r
#install.packages("tidygraph")
library(tidygraph)
```

```
## Warning: package 'tidygraph' was built under R version 4.3.3
```

```
##
## Attaching package: 'tidygraph'
```

```
## The following object is masked from 'package:igraph':
##
##     groups
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

ggraph

- Visualisatiepakket voor netwerken gebaseerd op de ggplot2 syntax.
- Krachtige en flexibele netwerkvisualisaties binnen het tidyverse.
- Werkt samen met tidygraph en gebruikt tbl_graph als input.

```r
#install.packages("ggraph")
library(ggraph)
```

```
## Warning: package 'ggraph' was built under R version 4.3.3
```

```
## Loading required package: ggplot2
```

Andere mogelijke nuttige packages

Er bestaan ook nog vele andere nuttige packages die het analyseren van netwerkdata mogelijk maken, maar deze zijn voornamelijk bedoeld voor het uitvoeren van meer geavanceerde analyses. Sommige van deze packages zullen we tegenkomen later in de cursus, maar voel je gerust vrij om de documentatie van deze packages even na te kijken op: https://cran.r-project.org/web/packages/available_packages_by_date.html

Pakket	Beschrijving	Relatie tot andere packages
`networkDynamic`	Extensie van `network`, ondersteunt tijdsafhankelijke netwerken	Gebruikt `network`-objecten
`ndtv`	Visualisatie van dynamische netwerken	Gebruikt output van `networkDynamic`
`egor`	Werkt met egocentrische netwerken	Zelfstandige datastructuren
`ergm`	Statistische modellen voor netwerken (Exponential Random Graph Models)	Vereist `network`-objecten

Pakket	Beschrijving	Relatie tot andere packages
`ergm.ego`	Variant van `ergm` voor egocentrische netwerken	Gebruikt egocentrische data
`ergm.count`	Extensie van `ergm` voor netwerken met tellingen als randwaarden	Compatibel met `ergm`
`tergm`	Temporele extensie van `ergm`	Werkt met `networkDynamic`
`relevent`	Analyse van relationele gebeurtenissen via event history models	Alternatief voor `tergm`
`EpiModel`	Modellering van epidemieën op netwerken	Bouwt voort op `network` en `ergm`
`netdiffuseR`	Analyse van netwerkdiffusieprocessen	Zelfstandig pakket
`RSiena`	Longitudinale statistische modellen voor sociale netwerken	Vereist specifieke data- en modelstructuur

Enkele handige tips m.b.t. packages:

- Handig om even de specifieke versie en functies van een package te checken, dit kan als volgt:

```
help(package = sna)
```

- We kunnen ook een pakket ontkoppelen als we het niet langer geladen willen hebben. Dit is soms handig als twee pakketten niet goed samenwerken.

```
#detach(package:igraph)
```

3. Inladen van netwerkdata in R

Gedurende deze cursus zullen we werken met verschillende bronnen van netwerkdata, dit om bepaalde zaken te demonstreren alsook om bepaalde vormen van analyses uit te voeren. Deze datasets betreffen echte data afkomstig uit de sociale wetenschappen en zijn handig om bepaalde zaken in te oefenen.

Op basis van de demonstraties en enkele praktische oefeningen die doorheen deze cursus aan bod zullen komen, zullen jullie aan het einde van de workshop een eindoefening maken waar alle opgedane kennis zal worden getoetst op basis van geanomiseerde netwerkdata die lijken op data die jullie in het dagelijks werk gebruiken.

3.1. Inladen van .csv data

`.csv` bestanden omvatten een klassieke vorm van hoe data worden opgeslagen. Zoals we reeds hebben gezien kunnen deze ingeladen met de `read.csv()` alsook met de `read_csv()` functie. We laden hier een dataset in die de vriendschappen tussen elke leerling in een klas beschrijft (`class555_matrix.csv`).

```
#install.packages('readr') # Indien nodig, installeer het readr package voor het inladen van csv en txt
library(readr) # indien nodig
```

```
## Warning: package 'readr' was built under R version 4.3.3
```

```r
class_matrix <- read.csv(file = 'Data/class555_matrix.csv', header = TRUE, sep = ',') # base R functie
class_matrix <- read_csv(file = 'Data/class555_matrix.csv') # readr functie
```

```
## Rows: 24 Columns: 24
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (24): id1, id2, id3, id4, id5, id6, id7, id8, id9, id10, id11, id12, id1...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
head(class_matrix, n = 5) # Bekijk de eerste 5 rijen van dataset
```

```
## # A tibble: 5 x 24
##     id1   id2   id3   id4   id5   id6   id7   id8   id9  id10  id11  id12  id13
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     0     0     1     0     1     0     1     0     0     0     0     0     0
## 2     0     0     1     0     0     1     0     0     0     0     0     0     0
## 3     0     0     0     0     0     1     0     1     0     0     0     0     0
## 4     0     0     0     0     0     0     0     0     0     0     0     0     1
## 5     0     0     0     0     0     0     0     0     0     0     0     0     0
## # i 11 more variables: id14 <dbl>, id15 <dbl>, id16 <dbl>, id17 <dbl>,
## #   id18 <dbl>, id19 <dbl>, id20 <dbl>, id21 <dbl>, id22 <dbl>, id23 <dbl>,
## #   id24 <dbl>
```

```r
tail(class_matrix, n = 5) # Bekijk de laatste 5 rijen van de dataset
```

```
## # A tibble: 5 x 24
##     id1   id2   id3   id4   id5   id6   id7   id8   id9  id10  id11  id12  id13
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     0     0     0     0     0     0     0     0     0     0     0     0     0
## 2     0     0     0     0     1     0     0     0     0     0     0     0     0
## 3     0     0     0     0     0     0     0     0     0     0     0     0     0
## 4     0     0     0     0     1     0     0     0     0     0     0     0     1
## 5     0     0     0     0     0     1     0     0     0     1     0     0     0
## # i 11 more variables: id14 <dbl>, id15 <dbl>, id16 <dbl>, id17 <dbl>,
## #   id18 <dbl>, id19 <dbl>, id20 <dbl>, id21 <dbl>, id22 <dbl>, id23 <dbl>,
## #   id24 <dbl>
```

3.2. Inladen van .txt data

`.txt` bestanden omvatten eveneens een klassieke vorm van hoe data worden opgeslagen. Hier laden we het bestand `ego_network_example_data.txt` in:

```r
#install.packages('readr') # Indien nodig, installeer het readr package voor het inladen van csv en txt
#library(readr) # indien nodig
ego_matrix <- read_delim(file = 'Data/ego_network_example_data.txt', delim = ' ')
```

```
## Rows: 750 Columns: 6
## -- Column specification --------------------------------------------------------
```

```
## Delimiter: " "
## chr (4): location, location1, location2, location3
## dbl (2): ids, degree
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(ego_matrix, n = 5) # Bekijk de eerste 5 rijen van dataset
```

```
## # A tibble: 5 x 6
##     ids location degree location1 location2 location3
##   <dbl> <chr>     <dbl> <chr>     <chr>     <chr>
## 1     1 city          1 city      <NA>      <NA>
## 2     2 city          9 city      suburbs   city
## 3     3 city          2 city      city      <NA>
## 4     4 city          2 city      city      <NA>
## 5     5 city          5 city      city      city
```

```
tail(ego_matrix, n = 5) # Bekijk de laatste 5 rijen van de dataset
```

```
## # A tibble: 5 x 6
##     ids location degree location1 location2 location3
##   <dbl> <chr>     <dbl> <chr>     <chr>     <chr>
## 1   746 suburbs       2 suburbs   suburbs   <NA>
## 2   747 suburbs       3 suburbs   suburbs   suburbs
## 3   748 suburbs       2 city      suburbs   <NA>
## 4   749 suburbs       2 suburbs   suburbs   <NA>
## 5   750 suburbs       3 city      suburbs   city
```

3.3. Inladen van .dat bestanden

`.dat`-bestanden zijn vaak tekstbestanden met gestructureerde data, die kunnen ingelezen worden via `read.table()` of `read_delim()` uit het `readr` pakket. Hier laden we een dataset in betreffende een vriendschapsnetwerk:

```
# Base R
friendship_w1 <- read.table("Data/s50-network1.dat") # base R functie
head(friendship_w1, n = 5) # Bekijk de eerste 5 rijen van dataset
```

```
##   V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21
## 1  0  0  0  0  0  0  0  0  0   0   1   0   0   1   0   0   0   0   0   0   0
## 2  0  0  0  0  0  0  1  0  0   0   1   0   0   0   0   0   0   0   0   0   0
## 3  0  0  0  1  0  0  0  0  1   0   0   0   0   0   0   0   0   0   0   0   0
## 4  0  0  1  0  0  0  0  0  1   0   0   0   0   0   0   0   0   0   0   0   0
## 5  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
##   V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V34 V35 V36 V37 V38 V39 V40
## 1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 5   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
```

```
##   V41 V42 V43 V44 V45 V46 V47 V48 V49 V50
## 1   0   0   0   0   0   0   0   0   0   0
## 2   0   0   0   0   0   0   0   0   0   0
## 3   0   0   0   0   0   0   0   0   0   0
## 4   0   0   0   0   0   0   0   0   0   0
## 5   0   0   0   0   0   0   0   0   0   0
```

```
tail(friendship_w1, n = 5) # Bekijk de laatste 5 rijen van de dataset
```

```
##    V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21
## 46  0  0  0  0  0  0  0  0  0   0   0   0   0   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   0
## 48  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
## 49  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
## 50  0  0  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0
##    V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V34 V35 V36 V37 V38 V39 V40
## 46   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1
## 47   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
## 49   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
## 50   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##    V41 V42 V43 V44 V45 V46 V47 V48 V49 V50
## 46   0   0   0   0   1   0   0   0   1   0
## 47   0   0   0   0   0   0   0   0   0   0
## 48   0   0   0   0   0   1   0   0   1   0
## 49   0   0   0   0   0   1   0   1   0   0
## 50   0   0   0   0   0   0   0   0   0   0
```

Tidyverse alternatief:

```
#library(readr)
friendship_w1  <- read_delim("Data/s50-network1.dat", delim = " ", col_names = FALSE) # vaak tab-gesche
```

```
## Rows: 50 Columns: 51
## -- Column specification --------------------------------------------------------
## Delimiter: " "
## dbl (50): X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16,...
## lgl  (1): X1
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
friendship_w1  <- friendship_w1[-c(1)] # Verwijder eerste kolom die meer wordt ingeladen maar leeg is
head(friendship_w1, n = 5) # Bekijk de eerste 5 rijen van dataset
```

```
## # A tibble: 5 x 50
##      X2    X3    X4    X5    X6    X7    X8    X9   X10   X11   X12   X13   X14
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     0     0     0     0     0     0     0     0     0     0     1     0     0
## 2     0     0     0     0     0     0     1     0     0     0     1     0     0
## 3     0     0     0     1     0     0     0     0     1     0     0     0     0
## 4     0     0     1     0     0     0     0     0     1     0     0     0     0
```

```
## 5     0     0     0     0     0     0     0     0     0     0     0     0     0
## # i 37 more variables: X15 <dbl>, X16 <dbl>, X17 <dbl>, X18 <dbl>, X19 <dbl>,
## #   X20 <dbl>, X21 <dbl>, X22 <dbl>, X23 <dbl>, X24 <dbl>, X25 <dbl>,
## #   X26 <dbl>, X27 <dbl>, X28 <dbl>, X29 <dbl>, X30 <dbl>, X31 <dbl>,
## #   X32 <dbl>, X33 <dbl>, X34 <dbl>, X35 <dbl>, X36 <dbl>, X37 <dbl>,
## #   X38 <dbl>, X39 <dbl>, X40 <dbl>, X41 <dbl>, X42 <dbl>, X43 <dbl>,
## #   X44 <dbl>, X45 <dbl>, X46 <dbl>, X47 <dbl>, X48 <dbl>, X49 <dbl>,
## #   X50 <dbl>, X51 <dbl>
```

```
tail(friendship_w1, n = 5) # Bekijk de laatste 5 rijen van de dataset
```

```
## # A tibble: 5 x 50
##      X2    X3    X4    X5    X6    X7    X8    X9   X10   X11   X12   X13   X14
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     0     0     0     0     0     0     0     0     0     0     0     0     0
## 2     0     0     0     0     0     0     0     0     0     0     0     0     0
## 3     0     0     0     0     0     0     0     0     0     0     0     0     0
## 4     0     0     0     0     0     0     0     0     0     0     0     0     0
## 5     0     0     0     0     0     0     0     0     0     0     0     0     0
## # i 37 more variables: X15 <dbl>, X16 <dbl>, X17 <dbl>, X18 <dbl>, X19 <dbl>,
## #   X20 <dbl>, X21 <dbl>, X22 <dbl>, X23 <dbl>, X24 <dbl>, X25 <dbl>,
## #   X26 <dbl>, X27 <dbl>, X28 <dbl>, X29 <dbl>, X30 <dbl>, X31 <dbl>,
## #   X32 <dbl>, X33 <dbl>, X34 <dbl>, X35 <dbl>, X36 <dbl>, X37 <dbl>,
## #   X38 <dbl>, X39 <dbl>, X40 <dbl>, X41 <dbl>, X42 <dbl>, X43 <dbl>,
## #   X44 <dbl>, X45 <dbl>, X46 <dbl>, X47 <dbl>, X48 <dbl>, X49 <dbl>,
## #   X50 <dbl>, X51 <dbl>
```

3.4. Inladen van .RData bestanden

`.RData`-bestanden slaan één of meerdere R-objecten op. Gebruik load() om deze in het geheugen te laden.

```
#library(readr)
load("Data/Glasgow-friendship.RData")
load("Data/Glasgow-demographic.RData") # sex, age
load("Data/Glasgow-geographic.RData") # distance from school and others
ls() # controleer even welke data objecten zijn geladen
```

```
##  [1] "age"           "angle.1"       "angle.2"       "angle.3"
##  [5] "class_matrix"  "dist.school"   "distance.1"    "distance.2"
##  [9] "distance.3"    "ego_matrix"    "friendship.1"  "friendship.2"
## [13] "friendship.3"  "friendship_w1" "sex.F"
```

4. Inspecteren en manipuleren/cleanen van netwerkdata

Nadat we de data hebben ingelezen in R, is het handig om eerst de algemene kenmerken van de data even te controleren. Er bestaan een aantal handige built-in R functies die toelaten om zicht te krijgen op een aantal zaken. In dit verband is het vervolgens ook aangewezen om reeds rekening te houden met hoe de data er uit zien en eventueel enkele stappen te ondernemen om de data te manipuleren/cleanen volgens een structuur dat handig is voor verdere analyse.

We werken hiervoor met een aantal netwerkdatasets die normaal gezien reeds werden ingeladen in de R omgeving bij het inladen van de `Glasgow` data.

De `Glasgow` dataset maakt deel uit van de Teenage Friends and Lifestyle Study, die gericht was op het onderzoeken van de processen die de houding ten opzichte van roken en het rookgedrag zelf veranderen tijdens de vroege tot midden-adolescentie. De dataset bevat gegevens van 160 leerlingen die gedurende drie meetmomenten werden gevolgd, met de nadruk op vriendschapsnetwerken en gedragingen zoals roken, alcoholgebruik en andere levensstijlkenmerken. De gegevens van de oudere cohort zijn verzameld tussen 1995 en 1997, toen de leerlingen tussen de 13 en 15 jaar oud waren.

De dataset omvat verschillende variabelen, zoals vriendschapsnetwerken, demografische gegevens (zoals geslacht en leeftijd), en informatie over substantiëlen (alcohol, tabak, cannabis). Daarnaast bevat het ook gegevens over de mate van zakgeld, romantische relaties, en familiegerelateerde rookgewoonten. Meer informatie kan je hier terugvinden: https://www.stats.ox.ac.uk/~snijders/siena/Glasgow_data.htm

4.1. Inspecteren van de data (objecten)

We starten met het inspecteren van enkele kenmerken van de dataset `friendship.w1`, dewelke een vriendschapsnetwerk betreft onder de verschillende leerlingen binnen de `Glasgow` dataset zoals gemeten op het eerste meetmoment.

We controleren eerst of we toegang hebben tot de juiste dataset:

```
head(friendship.1)
```

```
##      s001 s002 s003 s004 s005 s006 s007 s008 s009 s010 s011 s012 s013 s014 s015
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    2    0    0    0    2    0    0    2
## s005    0    0    0    0    0    2    0    0    0    2    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s016 s017 s018 s019 s020 s021 s022 s023 s024 s025 s026 s027 s028 s029 s030
## s001    0    0    0    0    0    0    0    0    0    0    0    0    2    0    0
## s002    0    0    0    0    0    0    0    0    2    0    0    0    0    0    0
## s003    2    0    0    0    0    0    0    0    0    0    0    0    2    0    0
## s004    0    0    0    0    0    0    0    0    0    2    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    2    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s031 s032 s033 s034 s035 s036 s037 s038 s039 s040 s041 s042 s043 s044 s045
## s001    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    2    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    2    0    0    0    0    0    0
## s005    0    0    0    0    1    0    0    0    0    0    0    0    0    2    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s046 s047 s048 s049 s050 s051 s052 s053 s054 s055 s056 s057 s058 s059 s060
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    2    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    2    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s061 s062 s063 s064 s065 s066 s067 s068 s069 s070 s071 s072 s073 s074 s075
```

```
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    2    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s076 s077 s078 s079 s080 s081 s082 s083 s084 s085 s086 s087 s088 s089 s090
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    2    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s091 s092 s093 s094 s095 s096 s097 s098 s099 s100 s101 s102 s103 s104 s105
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    2    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s106 s107 s108 s109 s110 s111 s112 s113 s114 s115 s116 s117 s118 s119 s120
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    2    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s121 s122 s123 s124 s125 s126 s127 s128 s129 s130 s131 s132 s133 s134 s135
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    2    0    0    0    0    0    0    0    0    0    0    0    0
##      s136 s137 s138 s139 s140 s141 s142 s143 s144 s145 s146 s147 s148 s149 s150
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s151 s152 s153 s154 s155 s156 s157 s158 s159 s160
## s001    0    0   10   10   10   10    0   10   10   10
## s002    0    0   10   10   10   10    0   10   10   10
## s003    0    0   10   10   10   10    0   10   10   10
## s004    0    0   10   10   10   10    0   10   10   10
## s005    0    0   10   10   10   10    0   10   10   10
## s006    0    0   10   10   10   10    0   10   10   10
```

Laat ons nu enkele attributen van de dataset nagaan:

- Controleer of het object reeds een matrix 'class' is, zo niet, zet deze om in een matrix (dit kan heel eenvoudig met de functie `as.matrix()`):

```r
# Controleer of de data al een matrix is
if (any(class(friendship.1) == "matrix")) {
  print("De data is reeds een matrix")

  head(friendship.1)
} else {
  # Acties ondernemen als de data geen matrix is
  print("De data is geen matrix, converteren naar een matrix...")
  friendship.1 <- as.matrix(friendship.1)
  head(friendship.1)
}
```

```
## [1] "De data is reeds een matrix"
```

```
##      s001 s002 s003 s004 s005 s006 s007 s008 s009 s010 s011 s012 s013 s014 s015
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    2    0    0    0    2    0    0    2
## s005    0    0    0    0    0    2    0    0    0    2    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s016 s017 s018 s019 s020 s021 s022 s023 s024 s025 s026 s027 s028 s029 s030
## s001    0    0    0    0    0    0    0    0    0    0    0    0    2    0    0
## s002    0    0    0    0    0    0    0    0    2    0    0    0    0    0    0
## s003    2    0    0    0    0    0    0    0    0    0    0    0    2    0    0
## s004    0    0    0    0    0    0    0    0    0    2    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    2    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s031 s032 s033 s034 s035 s036 s037 s038 s039 s040 s041 s042 s043 s044 s045
## s001    0    0    1    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    2    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    2    0    0    0    0    0    0
## s005    0    0    0    0    1    0    0    0    0    0    0    0    0    2    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s046 s047 s048 s049 s050 s051 s052 s053 s054 s055 s056 s057 s058 s059 s060
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    2    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    2    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s061 s062 s063 s064 s065 s066 s067 s068 s069 s070 s071 s072 s073 s074 s075
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    2    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s076 s077 s078 s079 s080 s081 s082 s083 s084 s085 s086 s087 s088 s089 s090
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

```
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    2    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s091 s092 s093 s094 s095 s096 s097 s098 s099 s100 s101 s102 s103 s104 s105
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    2    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s106 s107 s108 s109 s110 s111 s112 s113 s114 s115 s116 s117 s118 s119 s120
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    2    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s121 s122 s123 s124 s125 s126 s127 s128 s129 s130 s131 s132 s133 s134 s135
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    2    0    0    0    0    0    0    0    0    0    0    0    0
##      s136 s137 s138 s139 s140 s141 s142 s143 s144 s145 s146 s147 s148 s149 s150
## s001    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s002    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s003    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s004    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s005    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s006    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
##      s151 s152 s153 s154 s155 s156 s157 s158 s159 s160
## s001    0    0   10   10   10   10    0   10   10   10
## s002    0    0   10   10   10   10    0   10   10   10
## s003    0    0   10   10   10   10    0   10   10   10
## s004    0    0   10   10   10   10    0   10   10   10
## s005    0    0   10   10   10   10    0   10   10   10
## s006    0    0   10   10   10   10    0   10   10   10
```

- Het is dus reeds een matrix, maar bevat het ook numerieke waarden?

```
# Mode van de matrix
if (is.matrix(friendship.1)) {
    # De matrix is al numeriek
    print("De matrix is al numeriek")

} else {
  # Acties ondernemen als de data geen matrix is
  print("De data is geen matrix met numerieke waarden, converteren naar een matrix met numerieke waarden
  friendship.1 <- as.numeric(friendship.1)
  head(friendship.1)

}
```

```
## [1] "De matrix is al numeriek"
```

- Controleer de dimensies van de matrix (aantal rijen en kolommen):

```
# Dimensie van de matrix
dim(friendship.1)
```

```
## [1] 160 160
```

- Controleer de structuur van de matrix:

```
# Structuur van de matrix
print(str(friendship.1))
```

```
##  int [1:160, 1:160] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:160] "s001" "s002" "s003" "s004" ...
##   ..$ : chr [1:160] "s001" "s002" "s003" "s004" ...
## NULL
```

- Bekijk een samenvatting van de matrix:

```
# Samenvatting van de matrix
summary(friendship.1)
```

```
##       s001             s002             s003             s004
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.465   Mean   : 0.5223   Mean   : 0.4713   Mean   : 0.5605
##  3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3        NA's   :3         NA's   :3         NA's   :3
##       s005             s006             s007             s008
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4841   Mean   : 0.4713   Mean   : 0.5032   Mean   : 0.5223
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s009             s010             s011             s012
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4777   Mean   : 0.4968   Mean   : 0.4586   Mean   : 0.5096
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s013             s014             s015             s016
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
```

```
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4968   Mean   : 0.4586   Mean   : 0.5414   Mean   : 0.5032
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s017              s018              s019              s020
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4968   Mean   : 0.4968   Mean   : 0.4777   Mean   : 0.4522
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s021              s022              s023              s024
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4522   Mean   : 0.4777   Mean   : 0.4968   Mean   : 0.4713
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s025              s026              s027              s028
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4968   Mean   : 0.4968   Mean   : 0.4904   Mean   : 0.5924
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s029              s030              s031              s032
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4459   Mean   : 0.4522   Mean   : 0.5096   Mean   : 0.4841
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s033             s034              s035              s036
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.465   Mean   : 0.4841   Mean   : 0.5159   Mean   : 0.4904
##  3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3        NA's   :3         NA's   :3         NA's   :3
##       s037              s038              s039              s040
##  Min.   : 0.0000   Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.5032   Mean   : 0.465   Mean   : 0.5223   Mean   : 0.4841
##  3rd Qu.: 0.0000   3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3        NA's   :3         NA's   :3
```

```
##       s041              s042              s043              s044
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4968   Mean   : 0.4586   Mean   : 0.4841   Mean   : 0.5669
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s045              s046              s047              s048
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4777   Mean   : 0.5159   Mean   : 0.4586   Mean   : 0.4713
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s049              s050              s051              s052
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.5287   Mean   : 0.4904   Mean   : 0.5223   Mean   : 0.4904
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s053             s054             s055             s056
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 0.000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.0000
##  Median : 0.000   Median : 0.0000   Median : 0.000   Median : 0.0000
##  Mean   : 0.465   Mean   : 0.4459   Mean   : 0.465   Mean   : 0.5669
##  3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 0.000   3rd Qu.: 0.0000
##  Max.   :10.000   Max.   :10.0000   Max.   :10.000   Max.   :10.0000
##  NA's   :3        NA's   :3         NA's   :3        NA's   :3
##       s057              s058              s059              s060
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4713   Mean   : 0.5541   Mean   : 0.5478   Mean   : 0.4777
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s061             s062             s063             s064
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.535   Mean   : 0.4713   Mean   : 0.4841   Mean   : 0.4968
##  3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3        NA's   :3         NA's   :3         NA's   :3
##       s065              s066              s067             s068
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.000   Median : 0.0000
##  Mean   : 0.5796   Mean   : 0.4841   Mean   : 0.535   Mean   : 0.4904
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.000   3rd Qu.: 0.0000
```

```
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3        NA's   :3
##       s069              s070              s071              s072
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.5159   Mean   : 0.4713   Mean   : 0.5032   Mean   : 0.4904
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s073              s074              s075              s076
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4586   Mean   : 0.5223   Mean   : 0.4968   Mean   : 0.4904
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s077             s078              s079              s080
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.000   Median : 0.000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.465   Mean   : 0.535   Mean   : 0.4777   Mean   : 0.4904
##  3rd Qu.: 0.000   3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.000   Max.   :10.000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3        NA's   :3        NA's   :3         NA's   :3
##       s081              s082              s083              s084
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4522   Mean   : 0.4586   Mean   : 0.4777   Mean   : 0.4713
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s085              s086              s087              s088
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4777   Mean   : 0.4968   Mean   : 0.5096   Mean   : 0.5223
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s089              s090              s091              s092
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.000
##  Mean   : 0.5414   Mean   : 0.4713   Mean   : 0.4904   Mean   : 0.465
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s093              s094              s095              s096
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
```

```
##  Mean   : 0.4841   Mean   : 0.5032   Mean   : 0.5032   Mean   : 0.4459
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s097              s098              s099              s100
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4713   Mean   : 0.4904   Mean   : 0.4841   Mean   : 0.4586
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s101              s102              s103              s104
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.5096   Mean   : 0.4459   Mean   : 0.4968   Mean   : 0.5478
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s105              s106              s107              s108
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4904   Mean   : 0.5159   Mean   : 0.4586   Mean   : 0.4713
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s109              s110              s111              s112
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4713   Mean   : 0.4713   Mean   : 0.4904   Mean   : 0.4841
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s113              s114              s115              s116
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.5159   Mean   : 0.4968   Mean   : 0.5669   Mean   : 0.4841
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s117              s118              s119              s120
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.000
##  Mean   : 0.4841   Mean   : 0.4522   Mean   : 0.4713   Mean   : 0.535
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s121              s122              s123              s124
##  Min.   : 0.000    Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
```

```
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.465   Mean   : 0.4841   Mean   : 0.4777   Mean   : 0.4459
##  3rd Qu.: 0.000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3        NA's   :3         NA's   :3         NA's   :3
##       s125             s126             s127             s128
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 0.0000   Median : 0.0000   Median : 0.000   Median : 0.000
##  Mean   : 0.4713   Mean   : 0.4904   Mean   : 0.465   Mean   : 0.465
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.000   3rd Qu.: 0.000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.000   Max.   :10.000
##  NA's   :3         NA's   :3         NA's   :3        NA's   :3
##       s129             s130             s131             s132
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.000
##  Mean   : 0.4841   Mean   : 0.4713   Mean   : 0.5223   Mean   : 0.465
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s133             s134             s135             s136
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4904   Mean   : 0.4904   Mean   : 0.4777   Mean   : 0.4713
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s137             s138             s139             s140
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4586   Mean   : 0.5032   Mean   : 0.4777   Mean   : 0.4904
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s141             s142             s143             s144
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.000   Median : 0.0000
##  Mean   : 0.4841   Mean   : 0.4713   Mean   : 0.465   Mean   : 0.4841
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3        NA's   :3
##       s145             s146             s147             s148
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.4586   Mean   : 0.4968   Mean   : 0.4841   Mean   : 0.5032
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
```

```
##       s149              s150              s151              s152
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.5096   Mean   : 0.4777   Mean   : 0.5223   Mean   : 0.4586
##  3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000   3rd Qu.: 0.0000
##  Max.   :10.0000   Max.   :10.0000   Max.   :10.0000   Max.   :10.0000
##  NA's   :3         NA's   :3         NA's   :3         NA's   :3
##       s153         s154         s155         s156         s157
##  Min.   :10   Min.   :10   Min.   :10   Min.   :10   Min.   : 0.0000
##  1st Qu.:10   1st Qu.:10   1st Qu.:10   1st Qu.:10   1st Qu.: 0.0000
##  Median :10   Median :10   Median :10   Median :10   Median : 0.0000
##  Mean   :10   Mean   :10   Mean   :10   Mean   :10   Mean   : 0.4713
##  3rd Qu.:10   3rd Qu.:10   3rd Qu.:10   3rd Qu.:10   3rd Qu.: 0.0000
##  Max.   :10   Max.   :10   Max.   :10   Max.   :10   Max.   :10.0000
##                                                     NA's   :3
##       s158         s159         s160
##  Min.   :10   Min.   :10   Min.   :10
##  1st Qu.:10   1st Qu.:10   1st Qu.:10
##  Median :10   Median :10   Median :10
##  Mean   :10   Mean   :10   Mean   :10
##  3rd Qu.:10   3rd Qu.:10   3rd Qu.:10
##  Max.   :10   Max.   :10   Max.   :10
##
```

- Wat zijn de kolomnamen?

```
# Kolomnamen
colnames(friendship.1)
```

```
##   [1] "s001" "s002" "s003" "s004" "s005" "s006" "s007" "s008" "s009" "s010"
##  [11] "s011" "s012" "s013" "s014" "s015" "s016" "s017" "s018" "s019" "s020"
##  [21] "s021" "s022" "s023" "s024" "s025" "s026" "s027" "s028" "s029" "s030"
##  [31] "s031" "s032" "s033" "s034" "s035" "s036" "s037" "s038" "s039" "s040"
##  [41] "s041" "s042" "s043" "s044" "s045" "s046" "s047" "s048" "s049" "s050"
##  [51] "s051" "s052" "s053" "s054" "s055" "s056" "s057" "s058" "s059" "s060"
##  [61] "s061" "s062" "s063" "s064" "s065" "s066" "s067" "s068" "s069" "s070"
##  [71] "s071" "s072" "s073" "s074" "s075" "s076" "s077" "s078" "s079" "s080"
##  [81] "s081" "s082" "s083" "s084" "s085" "s086" "s087" "s088" "s089" "s090"
##  [91] "s091" "s092" "s093" "s094" "s095" "s096" "s097" "s098" "s099" "s100"
## [101] "s101" "s102" "s103" "s104" "s105" "s106" "s107" "s108" "s109" "s110"
## [111] "s111" "s112" "s113" "s114" "s115" "s116" "s117" "s118" "s119" "s120"
## [121] "s121" "s122" "s123" "s124" "s125" "s126" "s127" "s128" "s129" "s130"
## [131] "s131" "s132" "s133" "s134" "s135" "s136" "s137" "s138" "s139" "s140"
## [141] "s141" "s142" "s143" "s144" "s145" "s146" "s147" "s148" "s149" "s150"
## [151] "s151" "s152" "s153" "s154" "s155" "s156" "s157" "s158" "s159" "s160"
```

- Wat zijn de rijnamen?

```
# Rijnamen
rownames(friendship.1)
```

```
##   [1] "s001" "s002" "s003" "s004" "s005" "s006" "s007" "s008" "s009" "s010"
##  [11] "s011" "s012" "s013" "s014" "s015" "s016" "s017" "s018" "s019" "s020"
##  [21] "s021" "s022" "s023" "s024" "s025" "s026" "s027" "s028" "s029" "s030"
##  [31] "s031" "s032" "s033" "s034" "s035" "s036" "s037" "s038" "s039" "s040"
##  [41] "s041" "s042" "s043" "s044" "s045" "s046" "s047" "s048" "s049" "s050"
##  [51] "s051" "s052" "s053" "s054" "s055" "s056" "s057" "s058" "s059" "s060"
##  [61] "s061" "s062" "s063" "s064" "s065" "s066" "s067" "s068" "s069" "s070"
##  [71] "s071" "s072" "s073" "s074" "s075" "s076" "s077" "s078" "s079" "s080"
##  [81] "s081" "s082" "s083" "s084" "s085" "s086" "s087" "s088" "s089" "s090"
##  [91] "s091" "s092" "s093" "s094" "s095" "s096" "s097" "s098" "s099" "s100"
## [101] "s101" "s102" "s103" "s104" "s105" "s106" "s107" "s108" "s109" "s110"
## [111] "s111" "s112" "s113" "s114" "s115" "s116" "s117" "s118" "s119" "s120"
## [121] "s121" "s122" "s123" "s124" "s125" "s126" "s127" "s128" "s129" "s130"
## [131] "s131" "s132" "s133" "s134" "s135" "s136" "s137" "s138" "s139" "s140"
## [141] "s141" "s142" "s143" "s144" "s145" "s146" "s147" "s148" "s149" "s150"
## [151] "s151" "s152" "s153" "s154" "s155" "s156" "s157" "s158" "s159" "s160"
```

- Handig om reeds te weten wat de verdeling van de waarden is binnen de matrix:

```
#Gebruik de table functie om zicht te krijgen op de verdeling van de waarden in de matrix
table(friendship.1, useNA = "ifany")
```

```
## friendship.1
##     0     1     2    10  <NA>
## 22363   109   478  2191   459
```

- Een andere handig functie is de `describe()` functie van de `Hmisc` library:

```
#install.packages('Hmisc')
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 4.3.2
```

```
##
## Attaching package: 'Hmisc'
```

```
## The following object is masked from 'package:network':
##
##     is.discrete
```

```
## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
describe(friendship.1[ , 's005']) # handig om te zien indien je enkele specifieke observaties wil bekij
```

```
## friendship.1[, "s005"]
##        n  missing distinct     Info     Mean      Gmd
##      157        3        3    0.179   0.4841    0.926
##
```

```
## Value          0     2    10
## Frequency    147     3     7
## Proportion 0.936 0.019 0.045
##
## For the frequency table, variable is rounded to the nearest 0
```

- Controleer op symmetrie van een netwerk: Bij undirected netwerken is de matrix symmetrisch. Dit is belangrijk om te controleren vóór je het netwerk visualiseert of analyseert.

```
# Check of de matrix symmetrisch is
isSymmetric(friendship.1)
```

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

- Zijn er zelfverbindingen (loops)?: Bij sociale netwerken verwacht je meestal geen zelfverbindingen (d.w.z. een actor die een relatie met zichzelf heeft):

```
# Check of er zelfverbindingen zijn
diag(friendship.1)
```

```
## s001 s002 s003 s004 s005 s006 s007 s008 s009 s010 s011 s012 s013 s014 s015 s016
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s017 s018 s019 s020 s021 s022 s023 s024 s025 s026 s027 s028 s029 s030 s031 s032
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s033 s034 s035 s036 s037 s038 s039 s040 s041 s042 s043 s044 s045 s046 s047 s048
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0   NA
## s049 s050 s051 s052 s053 s054 s055 s056 s057 s058 s059 s060 s061 s062 s063 s064
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s065 s066 s067 s068 s069 s070 s071 s072 s073 s074 s075 s076 s077 s078 s079 s080
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s081 s082 s083 s084 s085 s086 s087 s088 s089 s090 s091 s092 s093 s094 s095 s096
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s097 s098 s099 s100 s101 s102 s103 s104 s105 s106 s107 s108 s109 s110 s111 s112
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s113 s114 s115 s116 s117 s118 s119 s120 s121 s122 s123 s124 s125 s126 s127 s128
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s129 s130 s131 s132 s133 s134 s135 s136 s137 s138 s139 s140 s141 s142 s143 s144
##    0    0    0    0    0    0    0   NA    0    0    0    2    0    0    0    0
## s145 s146 s147 s148 s149 s150 s151 s152 s153 s154 s155 s156 s157 s158 s159 s160
##    0    0    0    0    0    0    0    0   10   10   10   10   NA   10   10   10
```

```
any(diag(friendship.1) != 0)
```

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

Bovenstaand toont aan dat we de data verder dienen te manipuleren en cleanen vooralleer we de data verder kunnen analyseren.

4.2. Manipuleren en cleanen van de data

Op basis van een eerste inspectie van de data worden vaak al heel wat zaken duidelijk. We kunnen nu overgaan tot het manipuleren en cleanen van de data. Onthou dat we hier slechts de basis aanraken van wat het manipuleren en cleanen van data betreft en dat een meer diepgaande verkenning van data cleaning en processing in R aangewezen is voor bepaalde vormen en bronnen van netwerkdata, zoals bijvoorbeeld het manipuleren van tekstuele input in een gegeven dataset (strings).

We starten met enkele basisoperaties op de `friendship.w1` dataset die we hierboven hebben gebruikt, maar gaan vervolgens ook in op het uitvoeren van enkele operaties op andere datasets, zoals node attributes. Belangrijk om op te merken is dat het manipuleren van matrix data en netwerkdat vaak een andere manier van werken impliceert in vergelijking met andere structuren zoals data frames.

- Logische indexering en het inspecteren van specifieke waarden

```
# Tel het totaal aantal cellen in de matrix met waarde gelijk aan 0
sum(friendship.1 == 0, na.rm = TRUE)
```

```
## [1] 22363
```

```
# Tel het totaal aantal cellen in de matrix met waarde gelijk aan 1
sum(friendship.1 == 1, na.rm = TRUE)
```

```
## [1] 109
```

```
# Tel het totaal aantal cellen in de matrix met waarde gelijk aan 2
sum(friendship.1 == 2, na.rm = TRUE)
```

```
## [1] 478
```

```
# Tel het totaal aantal cellen in de matrix met waarde gelijk aan 10
sum(friendship.1 == 10, na.rm = TRUE)
```

```
## [1] 2191
```

```
# Tel het totaal aantal cellen in de matrix met waarde groter dan 0
sum(friendship.1 > 0, na.rm = TRUE)
```

```
## [1] 2778
```

```
# Tel het totaal aantal cellen in de matrix met waarde groter dan 1
sum(friendship.1 > 1, na.rm = TRUE)
```

```
## [1] 2669
```

```
# Tel het totaal aantal cellen in de matrix met waarde groter dan 2
sum(friendship.1 > 2, na.rm = TRUE)
```

```
## [1] 2191
```

```
# # Bekijk welke cellen waarden hebben groter dan 0
# friendship.1 == 0
#
# # Bekijk welke cellen waarden hebben groter dan 1
# friendship.1 == 1
#
# # Bekijk welke cellen waarden hebben groter dan 2
# friendship.1 == 2
#
# # Bekijk welke cellen waarden hebben groter dan 10
# friendship.1 == 10
```

```
# # Bekijk welke cellen tussen de waarden 2 en 10 liggen
# friendship.1 %in% c(2,10) # == is niet geldig in dergelijke gevallen
```

```
# Bekijk alle relaties vanuit node 's015'
friendship.1["s015", ]
```

```
## s001 s002 s003 s004 s005 s006 s007 s008 s009 s010 s011 s012 s013 s014 s015 s016
##    0    0    0    2    0    0    0    2    0    0    0    2    0    0    0    0
## s017 s018 s019 s020 s021 s022 s023 s024 s025 s026 s027 s028 s029 s030 s031 s032
##    0    0    0    0    0    0    0    0    2    0    0    0    0    0    0    0
## s033 s034 s035 s036 s037 s038 s039 s040 s041 s042 s043 s044 s045 s046 s047 s048
##    0    0    0    0    0    0    2    0    0    0    0    0    0    0    0    0
## s049 s050 s051 s052 s053 s054 s055 s056 s057 s058 s059 s060 s061 s062 s063 s064
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s065 s066 s067 s068 s069 s070 s071 s072 s073 s074 s075 s076 s077 s078 s079 s080
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s081 s082 s083 s084 s085 s086 s087 s088 s089 s090 s091 s092 s093 s094 s095 s096
##    0    0    0    0    0    0    0    0    2    0    0    0    0    0    0    0
## s097 s098 s099 s100 s101 s102 s103 s104 s105 s106 s107 s108 s109 s110 s111 s112
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s113 s114 s115 s116 s117 s118 s119 s120 s121 s122 s123 s124 s125 s126 s127 s128
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s129 s130 s131 s132 s133 s134 s135 s136 s137 s138 s139 s140 s141 s142 s143 s144
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s145 s146 s147 s148 s149 s150 s151 s152 s153 s154 s155 s156 s157 s158 s159 s160
##    0    0    0    0    0    0    0    0   10   10   10   10    0   10   10   10
```

```
# Bekijk alle relaties naar knoop 's015'
friendship.1[ , "s015"]
```

```
## s001 s002 s003 s004 s005 s006 s007 s008 s009 s010 s011 s012 s013 s014 s015 s016
##    0    0    0    2    0    0    2    2    0    0    0    2    0    0    0    0
## s017 s018 s019 s020 s021 s022 s023 s024 s025 s026 s027 s028 s029 s030 s031 s032
##    2    0    0    0    0    0    0    0    2    2    0    0    0    0    0    0
## s033 s034 s035 s036 s037 s038 s039 s040 s041 s042 s043 s044 s045 s046 s047 s048
##    0    0    0    0    0    0    1    0    0    0    0    0    0    0    0   NA
## s049 s050 s051 s052 s053 s054 s055 s056 s057 s058 s059 s060 s061 s062 s063 s064
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s065 s066 s067 s068 s069 s070 s071 s072 s073 s074 s075 s076 s077 s078 s079 s080
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

```
## s081 s082 s083 s084 s085 s086 s087 s088 s089 s090 s091 s092 s093 s094 s095 s096
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s097 s098 s099 s100 s101 s102 s103 s104 s105 s106 s107 s108 s109 s110 s111 s112
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s113 s114 s115 s116 s117 s118 s119 s120 s121 s122 s123 s124 s125 s126 s127 s128
##    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
## s129 s130 s131 s132 s133 s134 s135 s136 s137 s138 s139 s140 s141 s142 s143 s144
##    0    0    0    0    0    0    0   NA    0    0    0    0    0    0    0    0
## s145 s146 s147 s148 s149 s150 s151 s152 s153 s154 s155 s156 s157 s158 s159 s160
##    0    0    0    0    0    0    0    0   10   10   10   10   NA   10   10   10
```

```
# Zoek naar specifieke relatie tussen twee nodes (zoals gecodeerd in de data)
friendship.1["s005", "s006"]
```

```
## [1] 2
```

- Soms moeten individuele waarden in de matrix aangepast worden, bv. bij fouten of hercodering.

```
# Verander de waarde tussen s010 en s020 naar 0 (verwijder verbinding)
#friendship.1["s010", "s020"] <- 0

# Herclassificeer alle waarden gelijk aan 2 als 1
friendship.1[friendship.1  == 2] <- 1

# Herclassificeer alle waarden gelijk aan 10 (ongedefinieerde relatie) als 0 (geen relatie)
friendship.1[friendship.1  == 10] <- 0

# Print opnieuwe de waarden om te zien of we dit correct hebben gedaan
table(friendship.1)
```

```
## friendship.1
##     0     1
## 24554   587
```

- Ontbrekende waarden worden in R doorgaans aangeduidt met ‘NA’, maar door dataproductie kunnen ook specifieke waarden impliceren dat bepaalde waarden ontbreken (bv. 999); laat ons dit even nagaan op een andere dataset waar er NA’s zijn:

```
# Aantal ontbrekende waarden in de matrix
sum(is.na(friendship.1))
```

```
## [1] 459
```

```
# Ken de ontbrekende waarden toe aan een object
missings <- sum(is.na(friendship.1))

# Hoeveel relaties omvat onze dataset?
sum(friendship.1)
```

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

```r
# Ken dit toe als 'één'
ones <- sum(friendship.1, na.rm=TRUE)

# MINITASK: Hoe tellen we aantal nullen in de dataset?
zeros <- sum(friendship.1 == 0, na.rm = TRUE)

# Op deze manier kunnen we de proportie NA's berekenen in de data
missings / (zeros + ones)
```

```
## [1] 0.01825703
```

Dit valt goed mee. Er bestaan over het algemeen twee benaderingen om om te gaan met ontbrekende waarden, met name :

1. Imputatie, gaande van het vervangen van NA's door 0 tot het vervangen van NA's met behulp van specifieke maten zoals de mode, mean etc. alsook heel geavanceerde imputatiestrategieëne (bv. KNN)
2. Het verwijderen van alle NA's op een listwise manier. Hier worden kort twee benaderingen getoond en kiezen we ervoor om de rijen en kolommen met ontbrekende waarden te verwijderen, vermits er geen grotere proportie NA's is.

```r
# Optie 1: Imputatie - vervang NA's door 0 (geen relatie aangenomen)
#friendship.1[is.na(friendship.1)] <- 0

# Optie 2: Listwise deletion
  # Vind alle rijen waar een of meerdere attributen NA zijn
ids_with_na <- which(apply(friendship.1, 1, function(x) any(is.na(x))))

  # Store de ID-namen van deze rijen (en kolommen)
ids_to_remove <- rownames(friendship.1)[ids_with_na]

  # Verwijder deze rijen en kolommen uit de matrix (listwise deletion)
friendship.cleaned <- friendship.1[!(rownames(friendship.1) %in% ids_to_remove),
                                   !(colnames(friendship.1) %in% ids_to_remove)]

  # Check de dimensie na het uitvoeren van deze operatie:
dim(friendship.1)
```

```
## [1] 160 160
```

```r
dim(friendship.cleaned)
```

```
## [1] 157 157
```

- Soms kan het handig zijn om de rij-en kolomnamen te veranderen, dit kan op de volgende wijze:

```r
# Bekijk huidige namen
#rownames(friendship.1)[1:5]
#colnames(friendship.1)[1:5]

# Stel: 's001' moet 'R001' worden
```

```
#rownames(friendship.1)[rownames(friendship.1) == "s001"] <- "R001"
#colnames(friendship.1)[colnames(friendship.1) == "s001"] <- "R001"

# voeg prefix 'R' toe aan alle ID's
#rownames(friendship.1) <- paste0("R", substring(rownames(friendship.1), 2))
#colnames(friendship.1) <- paste0("R", substring(colnames(friendship.1), 2))

# OF: vervang het prefix 's' door 'R' in alle rijnamen en kolomnamen
#rownames(friendship.1) <- gsub("^s", "R", rownames(friendship.1))
#colnames(friendship.1) <- gsub("^s", "R", colnames(friendship.1))
```

5. Netwerkobjecten

Een laatste onderdeel van deze notebook gaat in op het definiëren van specifieke netwerkobjecten in R voor het uitvoeren van specifieke SNA analyses in een latere fase. Belangrijk om te onthouden is dat packages zoals igraph en network vaak vereisen op objecten aan te maken die specifiek relateren aan het package. Dit maakt het er soms niet makkelijker op, maar het is wat het is :).

In wat volgt behandelen we enkele verschillende manieren waarop netwerkobjecten aangemaakt kunnen worden, zowel met behulp van het `igraph` als het `network` package. We maken hiervoor gebruik van data die vriendschappen in een klaslokaal representeren, verzameld door Daniel McFarland. De nodes, zijn leerlingen in een klaslokaal. Er zijn 24 studenten. In de enquête werd elke leerling gevraagd om klasgenoten met wie ze optrokken als vrienden te nomineren. In de enquête werd elke leerling ook gevraagd naar geslacht, ras en klasniveau. We hebben dus netwerkgegevens, gebaseerd op de vriendschappen tussen leerlingen, en gegevens over kenmerken op het niveau van de nodes.

In dit geval hebben we de netwerkgegevens opgeslagen als een matrix die de vriendschappen tussen elke leerling in de klas beschrijft. Het netwerk is binair en directed. De matrix is opgeslagen als een CSV-bestand met de naam `class555_matrix.csv`. De attribuutgegevens voor de leerlingen in deze klas heten `class555_attributedata.csv`. Het bevat gegevens over het geslacht, het ras en de klas voor elke leerling in de klas. We hebben ook een edgelist ter beschikking voor dit netwerk genaamd `class555_edgelist.csv`.

5.1. Inlezen en voorbereiden van netwerkdata

We laden eerst een matrixobject in met vriendschapsgegevens, en transformeren dit vervolgens naar een vorm die geschikt is voor netwerkanalyse:

```
# Inlezen van de data
library(igraph)
class_mat <- read.csv(file = "Data/class555_matrix.csv", header = TRUE)
head(class_mat)
```

```
##   id1 id2 id3 id4 id5 id6 id7 id8 id9 id10 id11 id12 id13 id14 id15 id16 id17
## 1   0   0   1   0   1   0   1   0   0    0    0    0    0    0    0    0    0
## 2   0   0   1   0   0   1   0   0   0    0    0    0    0    0    0    0    0
## 3   0   0   0   0   0   1   0   1   0    0    0    0    0    0    0    1    0
## 4   0   0   0   0   0   0   0   0   0    0    0    0    1    0    0    0    0
## 5   0   0   0   0   0   0   0   0   0    0    0    0    0    0    0    0    0
## 6   0   0   0   0   0   0   0   0   0    0    0    0    0    0    0    0    0
##   id18 id19 id20 id21 id22 id23 id24
## 1    0    0    0    1    0    0    0
```

```
## 2    0    0    0    0    0    0    0
## 3    0    0    0    0    0    0    1
## 4    1    0    0    0    0    0    0
## 5    0    0    0    0    0    0    0
## 6    0    0    0    0    0    0    0
```

```
class(class_mat)
```

```
## [1] "data.frame"
```

Zoals je ziet is dit nog een data.frame object, die we dienen te transformeren naar een matrix:

```
# Transformeren naar een matrix
class_mat <- as.matrix(class_mat)
head(class_mat)
```

```
##      id1 id2 id3 id4 id5 id6 id7 id8 id9 id10 id11 id12 id13 id14 id15 id16
## [1,]   0   0   1   0   1   0   1   0   0    0    0    0    0    0    0    0
## [2,]   0   0   1   0   0   1   0   0   0    0    0    0    0    0    0    0
## [3,]   0   0   0   0   0   1   0   1   0    0    0    0    0    0    0    1
## [4,]   0   0   0   0   0   0   0   0   0    0    0    0    1    0    0    0
## [5,]   0   0   0   0   0   0   0   0   0    0    0    0    0    0    0    0
## [6,]   0   0   0   0   0   0   0   0   0    0    0    0    0    0    0    0
##      id17 id18 id19 id20 id21 id22 id23 id24
## [1,]    0    0    0    0    1    0    0    0
## [2,]    0    0    0    0    0    0    0    0
## [3,]    0    0    0    0    0    0    0    1
## [4,]    0    1    0    0    0    0    0    0
## [5,]    0    0    0    0    0    0    0    0
## [6,]    0    0    0    0    0    0    0    0
```

```
class(class_mat)
```

```
## [1] "matrix" "array"
```

Gelukt! Laat ons nu ook eerst de rij en kolomnamen uniform maken:

```
#Rijen en kolommamen tranformeren
rownames(class_mat) <- 1:nrow(class_mat)
colnames(class_mat) <- 1:ncol(class_mat)
head(class_mat)
```

```
##   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
## 1 0 0 1 0 1 0 1 0 0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
## 2 0 0 1 0 0 1 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3 0 0 0 0 0 1 0 1 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  1
## 4 0 0 0 0 0 0 0 0 0  0  0  0  1  0  0  0  0  1  0  0  0  0  0  0
## 5 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 6 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Zijn er ontbrekende waarden?

```r
sum(is.na(class_mat))
```

```
## [1] 0
```

Wat is de verdeling van de waarden en welke waarden zijn er?

```r
table(class_mat)
```

```
## class_mat
##   0   1
## 499  77
```

We zien dat de rij- en kolomnamen lopen van 1 tot 24. De matrix zelf bestaat uit 0-en en 1-en, waarbij een 1 betekent dat i j als vriend nomineert en 0 betekent dat i j niet als vriend nomineerde, waarbij i de rij is en j de kolom. Leerling 1 nomineert bijvoorbeeld leerlingen 3, 5, 7 en 21 als vrienden. Laten we nu de attribuutgegevens inlezen.

```r
# Inlezen van de attribuut data
class_att <- read.csv(file = "Data/class555_attributedata.csv", header = TRUE)
class_att
```

```
##    id gender grade  race
## 1   1   Male    12 White
## 2   2 Female    12 White
## 3   3 Female    12 White
## 4   4 Female    12 White
## 5   5   Male    12 White
## 6   6 Female    12 White
## 7   7   Male    11 Black
## 8   8   Male    11 White
## 9   9   Male    11 White
## 10 10 Female    11 White
## 11 11 Female    10 White
## 12 12 Female    10 White
## 13 13   Male    10 White
## 14 14   Male    10 White
## 15 15 Female    10 White
## 16 16   Male    10 White
## 17 17 Female    10 White
## 18 18 Female    10 White
## 19 19 Female    10 White
## 20 20 Female    10 White
## 21 21 Female    10 White
## 22 22 Female    10 White
## 23 23 Female    10 White
## 24 24 Female    10 White
```

Dit is een eenvoudig data frame dat het geslacht, de klas en de etniciteit van elke student in het netwerk beschrijft. Het dataframe bevat dus de attributen van onze nodes. Merk op dat de volgorde van het dataframe moet overeenkomen met de volgorde in de matrix. Merk ook op dat de eerste kolom het ID is van elk knooppunt in het netwerk (dit moet overeenkomen met de ID's in de edgelist). We kunnen specifieke kolommen ophalen met een $ commando of door een specifieke kolom in het dataframe aan te roepen. Hier bekijken we de eerste vijf waarden voor geslacht:

```
class_att$gender[1:5]
```

```
## [1] "Male"   "Female" "Female" "Female" "Male"
```

5.2. Netwerken in igraph

5.2.1. Het gebruiken van een matrix om een netwerk te construeren

We beginnen met het omzetten van onze ruwe matrix naar een netwerk in igraph (of netwerk) formaat. Het voordeel hiervan is dat we alle grafische functies, meatstaven en statistische functies kunnen gebruiken die beschikbaar zijn in de verschillende netwerkpakketten. igraph, bijvoorbeeld, biedt een verscheidenheid aan functies voor het uitvoeren van netwerkberekeningen, plots en simulaties.

- Omzetten van matrix naar een igraph netwerk object:

```
# Omzetten van matrix naar een igraph netwerk object
class_matrix_net <- graph_from_adjacency_matrix(adjmatrix = class_mat,
                                                mode = "directed")
class_matrix_net
```

```
## IGRAPH 39bdea0 DN-- 24 77 --
## + attr: name (v/c)
## + edges from 39bdea0 (vertex names):
##  [1] 1 ->3  1 ->5  1 ->7  1 ->21 2 ->3  2 ->6  3 ->6  3 ->8  3 ->16 3 ->24
## [11] 4 ->13 4 ->18 7 ->1  7 ->9  7 ->10 7 ->16 8 ->3  8 ->9  8 ->13 9 ->5
## [21] 9 ->8  10->6  10->14 10->19 10->20 10->24 11->12 11->15 11->18 11->24
## [31] 12->11 12->15 12->24 13->8  14->10 14->13 14->19 14->21 14->24 15->10
## [41] 15->11 15->13 15->14 15->24 16->3  16->5  16->9  16->19 17->8  17->13
## [51] 17->18 17->23 17->24 18->13 18->17 18->23 18->24 19->14 19->16 19->20
## [61] 19->21 20->19 20->21 20->24 21->5  21->19 21->20 22->23 23->5  23->13
## [71] 23->17 23->18 24->6  24->10 24->14 24->15 24->21
```

We zien dat het `igraph`-object heel wat nuttige informatie bevat, waaronder de grootte van het netwerk (24 nodes), het aantal verbindingen (77) en specifieke informatie over de verbindingen (edges).

Nu gaan we de **node attributen** die we eerder inlazen, zoals **gender** en **etniciteit (race)**, koppelen aan ons `igraph`-object. Hiervoor gebruiken we de functie `set_vertex_attr()`. Deze functie maakt het mogelijk om een attribuut, zoals gender, te koppelen aan de knopen in het netwerk. Op die manier weten we welk geslacht elke knoop in het netwerk vertegenwoordigt.

De argumenten van de functie zijn:

- `graph` = het netwerk als een `igraph`-object
- `name` = de naam van het node attribuut dat je wil aanmaken in het netwerk
- `value` = een vector met de attribuutwaarden voor de nodes

```
# Instellen van de node attributes
class_matrix_net <- set_vertex_attr(graph = class_matrix_net,
                                    name = "gender",
```

```
                                    value = class_att$gender)


class_matrix_net <- set_vertex_attr(graph = class_matrix_net,
                                    name = "grade",
                                    value = class_att$grade)

class_matrix_net <- set_vertex_attr(graph = class_matrix_net,
                                    name = "race",
                                    value = class_att$race)
class_matrix_net
```

```
## IGRAPH 39bdea0 DN-- 24 77 --
## + attr: name (v/c), gender (v/c), grade (v/n), race (v/c)
## + edges from 39bdea0 (vertex names):
##  [1] 1 ->3  1 ->5  1 ->7  1 ->21 2 ->3  2 ->6  3 ->6  3 ->8  3 ->16 3 ->24
## [11] 4 ->13 4 ->18 7 ->1  7 ->9  7 ->10 7 ->16 8 ->3  8 ->9  8 ->13 9 ->5
## [21] 9 ->8  10->6  10->14 10->19 10->20 10->24 11->12 11->15 11->18 11->24
## [31] 12->11 12->15 12->24 13->8  14->10 14->13 14->19 14->21 14->24 15->10
## [41] 15->11 15->13 15->14 15->24 16->3  16->5  16->9  16->19 17->8  17->13
## [51] 17->18 17->23 17->24 18->13 18->17 18->23 18->24 19->14 19->16 19->20
## [61] 19->21 20->19 20->21 20->24 21->5  21->19 21->20 22->23 23->5  23->13
## [71] 23->17 23->18 24->6  24->10 24->14 24->15 24->21
```

5.2.2. Het gebruiken van een edgelist om een netwerk te construeren

Nu zullen we dezelfde taak uitvoeren van het construeren van een netwerk (als een `igraph`-object), maar deze keer gebruiken we een *edgelist* als datastructuur voor de invoer.

Onthoud dat een *edgelist* een dataset is waarin de verbindingen (*edges*) in het netwerk worden vastgelegd. De basisvorm houdt in dat de eerste kolom de zender van de relatie (de *tie*) weergeeft en de tweede kolom de ontvanger.

Een *edgelist* bevat alle informatie die ook in een matrix voorkomt, met als uitzondering dat ze geen informatie geeft over *isolaten* — dat wil zeggen knopen zonder inkomende of uitgaande verbindingen.

- Inladen van de edgelist:

```
# Inladen van de edge list
class_edges <- read.csv(file = "Data/class555_edgelist.csv")
class_edges
```

```
##    sender receiver weight
## 1       1        3      1
## 2       1        5      1
## 3       1        7      1
## 4       1       21      1
## 5       2        3      1
## 6       2        6      1
## 7       3        6      1
## 8       3        8      1
## 9       3       16      1
```

```
## 10      3       24      1
## 11      4       13      1
## 12      4       18      1
## 13      7        1      1
## 14      7        9      1
## 15      7       10      1
## 16      7       16      1
## 17      8        3      1
## 18      8        9      1
## 19      8       13      1
## 20      9        5      1
## 21      9        8      1
## 22     10        6      1
## 23     10       14      1
## 24     10       19      1
## 25     10       20      1
## 26     10       24      1
## 27     11       12      1
## 28     11       15      1
## 29     11       18      1
## 30     11       24      1
## 31     12       11      1
## 32     12       15      1
## 33     12       24      1
## 34     13        8      1
## 35     14       10      1
## 36     14       13      1
## 37     14       19      1
## 38     14       21      1
## 39     14       24      1
## 40     15       10      1
## 41     15       11      1
## 42     15       13      1
## 43     15       14      1
## 44     15       24      1
## 45     16        3      1
## 46     16        5      1
## 47     16        9      1
## 48     16       19      1
## 49     17        8      1
## 50     17       13      1
## 51     17       18      1
## 52     17       23      1
## 53     17       24      1
## 54     18       13      1
## 55     18       17      1
## 56     18       23      1
## 57     18       24      1
## 58     19       14      1
## 59     19       16      1
## 60     19       20      1
## 61     19       21      1
## 62     20       19      1
## 63     20       21      1
```

```
## 64     20       24       1
## 65     21        5       1
## 66     21       19       1
## 67     21       20       1
## 68     22       23       1
## 69     23        5       1
## 70     23       13       1
## 71     23       17       1
## 72     23       18       1
## 73     24        6       1
## 74     24       10       1
## 75     24       14       1
## 76     24       15       1
## 77     24       21       1
```

We zien opnieuw dat student 1 student 3, 5, 7 en 21 nomineert. Merk op dat we in veel gevallen extra informatie aan de verbindingen (*edges*) kunnen toevoegen in de vorm van *edge attributes*, of gewichten (*weights*), die de sterkte of het type van de relatie tussen i en j vastleggen.

Deze extra informatie kan eenvoudig worden weergegeven via bijkomende kolommen in de edgelist. In dit voorbeeld is een bijkomende kolom toegevoegd met de naam `weight`. Aangezien de relatie in dit geval binair is (vriend of geen vriend), zijn alle gewichten identiek en gelijk aan 1. In andere gevallen zouden de waarden kunnen variëren om de intensiteit van de relatie tussen i en j weer te geven.

Nu willen we opnieuw een `igraph`-object creëren, zoals eerder, maar deze keer op basis van de edgelist. Hiervoor gebruiken we de functie `graph_from_data_frame()`. De argumenten van deze functie zijn:

- `d`: de edgelist;
- `directed`: `TRUE` of `FALSE`, naargelang het netwerk gericht is of niet;
- `vertices`: een optioneel gegevensframe met knoopattributen (*vertex attributes*).

In dit voorbeeld creëren we een `igraph`-object genaamd `class_netbyedgelist`, op basis van de edgelist `class_edges`. Merk op dat het niet nodig is om de edgelist om te zetten naar een matrix; de functie aanvaardt rechtstreeks een gegevensframe als invoer.

```
# Inladen van de edge list
class_edge_net <- graph_from_data_frame(d = class_edges, directed = T)
class_edge_net
```

```
## IGRAPH 39c392c DNW- 24 77 --
## + attr: name (v/c), weight (e/n)
## + edges from 39c392c (vertex names):
##  [1] 1 ->3  1 ->5  1 ->7  1 ->21 2 ->3  2 ->6  3 ->6  3 ->8  3 ->16 3 ->24
## [11] 4 ->13 4 ->18 7 ->1  7 ->9  7 ->10 7 ->16 8 ->3  8 ->9  8 ->13 9 ->5
## [21] 9 ->8  10->6  10->14 10->19 10->20 10->24 11->12 11->15 11->18 11->24
## [31] 12->11 12->15 12->24 13->8  14->10 14->13 14->19 14->21 14->24 15->10
## [41] 15->11 15->13 15->14 15->24 16->3  16->5  16->9  16->19 17->8  17->13
## [51] 17->18 17->23 17->24 18->13 18->17 18->23 18->24 19->14 19->16 19->20
## [61] 19->21 20->19 20->21 20->24 21->5  21->19 21->20 22->23 23->5  23->13
## [71] 23->17 23->18 24->6  24->10 24->14 24->15 24->21
```

Zoals we kunnen zien, voegt `igraph` automatisch de verbindingsattributen (hier: `weight`) toe aan het `igraph`-object. Een belangrijk voordeel van het gebruik van een *edgelist* is dat het eenvoudig is om node attributen

op te nemen in het netwerkobject. Dit gebeurt door het gebruik van het argument `vertices` binnen de functie `graph_from_data_frame()`.

De invoer voor het argument `vertices` is een gegevensframe met de attributen van elke node — in dit geval `class_attributes`. `igraph` zal elke kolom in dit gegevensframe toevoegen aan het netwerkobject, met uitzondering van de eerste kolom, die verondersteld wordt de ID's van de nodes te bevatten. Deze ID's moeten exact overeenkomen met de ID's die in de *edgelist* voorkomen.

Hieronder voeren we de vorige opdracht opnieuw uit, maar deze keer met het `vertices`-argument:

```
# Inladen van de edge list en toevoegen van de attributen
class_edge_net <- graph_from_data_frame(d = class_edges, directed = T, vertices = class_att)
class_edge_net
```

```
## IGRAPH 39c5ca7 DNW- 24 77 --
## + attr: name (v/c), gender (v/c), grade (v/n), race (v/c), weight (e/n)
## + edges from 39c5ca7 (vertex names):
##  [1] 1 ->3  1 ->5  1 ->7  1 ->21 2 ->3  2 ->6  3 ->6  3 ->8  3 ->16 3 ->24
## [11] 4 ->13 4 ->18 7 ->1  7 ->9  7 ->10 7 ->16 8 ->3  8 ->9  8 ->13 9 ->5
## [21] 9 ->8  10->6  10->14 10->19 10->20 10->24 11->12 11->15 11->18 11->24
## [31] 12->11 12->15 12->24 13->8  14->10 14->13 14->19 14->21 14->24 15->10
## [41] 15->11 15->13 15->14 15->24 16->3  16->5  16->9  16->19 17->8  17->13
## [51] 17->18 17->23 17->24 18->13 18->17 18->23 18->24 19->14 19->16 19->20
## [61] 19->21 20->19 20->21 20->24 21->5  21->19 21->20 22->23 23->5  23->13
## [71] 23->17 23->18 24->6  24->10 24->14 24->15 24->21
```

De attributen worden in het `igraph`-object opgenomen in dezelfde volgorde als in het ingevoerde data frame. Dit is belangrijk wanneer men informatie uit het object wil extraheren. We kunnen de volgorde van de nodes in het netwerk controleren via:

```
V(class_edge_net)$name
```

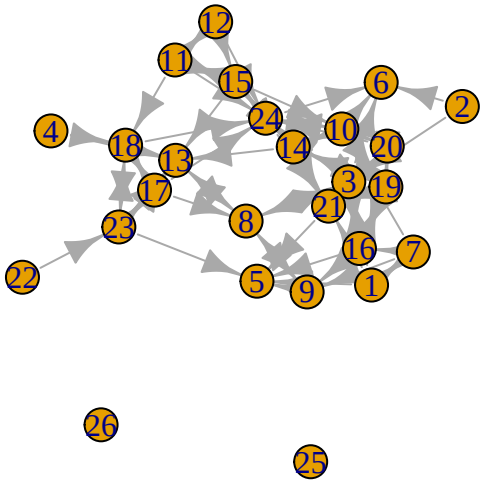
```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15"
## [16] "16" "17" "18" "19" "20" "21" "22" "23" "24"
```

Een laatste opmerking bij het construeren van netwerken op basis van een edgelist betreft het feit dat een edgelist geen informatie bevat over isolaten — nodes zonder inkomende of uitgaande verbindingen (edges). Indien dergelijke isolaten in het netwerk voorkomen, is voorzichtigheid geboden.

Een eenvoudige oplossing bestaat erin om het argument vertices te gebruiken, zelfs indien er geen bijkomende attributen beschikbaar zijn. Een gegevensframe met enkel de ID's van alle knopen volstaat, aangezien dit igraph informeert over de volledige omvang van het netwerk, inclusief de isolaten die anders zouden ontbreken.

Onderstaand voorbeeld toont hoe men een netwerk kan creëren dat ook de isolaten met ID's 25 en 26 omvat:

```
net <- graph_from_data_frame(d = class_edges, directed = TRUE,
                             vertices = data.frame(id = 1:26))
plot(net)
```



We kunnen node attributen, zoals `grade`, uit een `igraph`-object extraheren met behulp van de functie `vertex_attr()`. Dit is bijzonder nuttig, aangezien we daardoor niet telkens hoeven terug te grijpen naar de oorspronkelijke dataset om informatie over de knopen op te vragen.

Een belangrijk voordeel is dat alle aanpassingen aan het netwerk (zoals het verwijderen van isolaten) automatisch worden weerspiegeld in de aan het netwerkobject gekoppelde attributen. In tegenstelling hiermee blijven verwijderde knopen wel in het originele gegevensframe aanwezig. Dit vergemakkelijkt het beantwoorden van inhoudelijke vragen over de netwerkstructuur en de (demografische) kenmerken van de nodes.

De basiselementen van de functie `vertex_attr()` zijn:

- `graph`: het `igraph`-object;
- `name`: de naam van het knoopattribuut dat men wil extraheren.

```
vertex_attr(graph = class_edge_net, name = "grade")
```

```
##  [1] 12 12 12 12 12 12 11 11 11 11 10 10 10 10 10 10 10 10 10 10 10 10 10 10
```

Idem voor edge attributes zoas gewichten:

```
weights <- edge_attr(graph = class_edge_net, name = "weight")
```

Het is eveneens mogelijk om de edgelist van de adjacency matrix uit een igraph-object te extraheren. Dit is nuttig om twee redenen:

- Men kan een edgelist verkrijgen, zelfs indien de oorspronkelijke invoer een matrix was (of omgekeerd);

- De geëxtraheerde representaties zullen eventuele wijzigingen in het netwerk reflecteren (bijvoorbeeld het verwijderen van nodes of edges).

Bepaalde functies genereren `igraph`-objecten als output, en het is vaak wenselijk om deze om te zetten naar matrices of edgelists voor verdere analyse, gebruik hiervoor de functie `as_edgelist()`:

```
class_edges_temp <- as_edgelist(graph = class_edge_net, names = FALSE)
head(class_edges_temp)
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    1    5
## [3,]    1    7
## [4,]    1   21
## [5,]    2    3
## [6,]    2    6
```

Gebruik de functie as_adjacency_matrix() om de matrixrepresentatie van het netwerk te verkrijgen:

```
as_adjacency_matrix(graph = class_edge_net)
```

```
## 24 x 24 sparse Matrix of class "dgCMatrix"

##   [[ suppressing 24 column names '1', '2', '3' ... ]]

##
## 1  . . 1 . 1 . 1 . . . . . . . . . . . . . 1 . . .
## 2  . . 1 . . 1 . . . . . . . . . . . . . . . . . .
## 3  . . . . . 1 . 1 . . . . . . . 1 . . . . . . . 1
## 4  . . . . . . . . . . . . . 1 . . . . 1 . . . . .
## 5  . . . . . . . . . . . . . . . . . . . . . . . .
## 6  . . . . . . . . . . . . . . . . . . . . . . . .
## 7  1 . . . . . . . 1 1 . . . . . 1 . . . . . . . .
## 8  . . 1 . . . . . 1 . . . 1 . . . . . . . . . . .
## 9  . . . . 1 . . 1 . . . . . . . . . . . . . . . .
## 10 . . . . . 1 . . . . . . . . 1 . . . . 1 1 . . . 1
## 11 . . . . . . . . . . . . . 1 . . 1 . . 1 . . . . . 1
## 12 . . . . . . . . . . . . 1 . . . 1 . . . . . . . . 1
## 13 . . . . . . . . 1 . . . . . . . . . . . . . . . .
## 14 . . . . . . . . . . . 1 . . 1 . . . . . 1 . 1 . . 1
## 15 . . . . . . . . . . 1 1 . 1 1 . . . . . . . . . . 1
## 16 . . 1 . 1 . . . 1 . . . . . . . . . . . 1 . . . . .
## 17 . . . . . . . . 1 . . . . 1 . . . . 1 . . . . 1 1
## 18 . . . . . . . . . . . . . 1 . . . 1 . . . . . 1 1
## 19 . . . . . . . . . . . . . . 1 . 1 . . . 1 1 . . .
## 20 . . . . . . . . . . . . . . . . . . . . 1 . 1 . . 1
## 21 . . . . 1 . . . . . . . . . . . . . . . 1 1 . . . .
## 22 . . . . . . . . . . . . . . . . . . . . . . . . 1 .
## 23 . . . . 1 . . . . . . . . 1 . . . 1 1 . . . . . .
## 24 . . . . . 1 . . . 1 . . . 1 1 . . . . . 1 . . .
```

5.2.3. Het gebruiken van een adjacency list om een netwerk te construeren

Indien wenselijk en toepasbaar is het ook mogelijk om een netwerk te construeren op bases van een adjacency list:

- Inladen van de adjacency list:

```
#Inladen van de adjacency list
class_adjacency <- read.csv("Data/class555_adjacency_list.csv")
class_adjacency
```

```
##    id Nomination1 Nomination2 Nomination3 Nomination4 Nomination5
## 1   1           3           5           7          21          NA
## 2   2           3           6          NA          NA          NA
## 3   3           6           8          16          24          NA
## 4   4          13          18          NA          NA          NA
## 5   5          NA          NA          NA          NA          NA
## 6   6          NA          NA          NA          NA          NA
## 7   7           1           9          10          16          NA
## 8   8           3           9          13          NA          NA
## 9   9           5           8          NA          NA          NA
## 10 10           6          14          19          20          24
## 11 11          12          15          18          24          NA
## 12 12          11          15          24          NA          NA
## 13 13           8          NA          NA          NA          NA
## 14 14          10          13          19          21          24
## 15 15          10          11          13          14          24
## 16 16           3           5           9          19          NA
## 17 17           8          13          18          23          24
## 18 18          13          17          23          24          NA
## 19 19          14          16          20          21          NA
## 20 20          19          21          24          NA          NA
## 21 21           5          19          20          NA          NA
## 22 22          23          NA          NA          NA          NA
## 23 23           5          13          17          18          NA
## 24 24           6          10          14          15          21
```

De adjacency list heeft 24 rijen, één voor elke node, en 6 kolommen. De eerste kolom toont de id van de node (hier studenten) en de overige kolommen geven weer wie zij als vrienden benoemen. Als we naar de eerste rij kijken, zien we dat node 1 de knopen 3, 5, 7 en 21 als vrienden benoemt.

Het is vaak nuttig om onze adjacency list om te zetten naar een edge list (of matrix), aangezien het eenvoudig is om een netwerk te construeren vanuit een edge list, maar wat moeilijker vanuit een adjacency list (hoewel zie de functie graph_from_adj_list()). Hier zullen we functies uit het `reshape` packages gebruiken om onze adjacentie lijst om te zetten naar een edge list:

```
#Inladen van reshape package
#install.packages('reshape')
library(reshape)
```

```
## Warning: package 'reshape' was built under R version 4.3.3
```

```
##
## Attaching package: 'reshape'
```

```
## The following object is masked from 'package:tidygraph':
##
##     rename
```

We moeten vervolgens de kolommen in de adjacency-list identificeren die overeenkomen met de nominatiegegevens.

In dit geval zijn de kolommen van belang: Nomination1, Nomination2,... Nomination5.

Dit zijn de kolommen die we samen moeten stapelen om de ontvangerkolom in de edgelist te vormen. Hier zullen we een paste-opdracht gebruiken om een vector van kolomnamen te maken die overeenkomen met de nominatiekolommen.

```
nomination_columns <- paste("Nomination", 1:5, sep = "")
nomination_columns
```

```
## [1] "Nomination1" "Nomination2" "Nomination3" "Nomination4" "Nomination5"
```

Nu gebruiken we een reshape() functie om onze gegevens van een 'breed' formaat naar een 'lang' formaat te veranderen. We gaan dus van nodes als rijen naar edges als rijen. Het basisidee is om de id-variabele te nemen en te herhalen (de sender kolom vormen), terwijl we de nominatiegegevens stapelen om de receiver kolom te vormen.

In de reshape() functie stellen we data in op de hierboven gemaakte adjacency-list (class_adjacency); `varying` op de kolommen om te stapelen (nomination_columns); `v.names` op de naam van de variabele die in het lange formaat moet worden gemaakt; `idvar` op de id-variabele voor de nodes (id); en `direction` op `long`.

```
class_edgelist_adjacency <- reshape(data = class_adjacency,
                                    varying = nomination_columns,
                                    v.names = "receiver", idvar = "id",
                                    direction = "long")
head(class_edgelist_adjacency)
```

```
##     id time receiver
## 1.1  1    1        3
## 2.1  2    1        3
## 3.1  3    1        6
## 4.1  4    1       13
## 5.1  5    1       NA
## 6.1  6    1       NA
```

We kunnen zien dat de gegevens beginnen te lijken op een edgelist (lang in plaats van breed), maar er moet nog wat worden opgeschoond voordat we deze daadwerkelijk kunnen gebruiken. Bijvoorbeeld, we hebben die tweede kolom (die de nominatiekolom toont) eigenlijk niet nodig, dus willen we deze verwijderen. We willen ook een betere set kolomnamen aan de gegevens toevoegen.

```
class_edgelist_adjacency <- class_edgelist_adjacency[, -2]
colnames(class_edgelist_adjacency) <- c("sender", "receiver")
```

We moeten ook alle NA's uit de gegevens halen, die gewoon zijn gekopieerd uit de adjacency-list. We kunnen de `complete.cases()` functie gebruiken om alleen die rijen te behouden waar we geen NA-waarden hebben.

```
which_keep <- complete.cases(class_edgelist_adjacency)
class_edgelist_adjacency <- class_edgelist_adjacency[which_keep, ]
```

En laten we de gegevens ook opnieuw ordenen (hoewel dit niet strikt noodzakelijk is) om overeen te komen met de eerder ingelezen edgelist.

```
what_order <- order(class_edgelist_adjacency$sender)
class_edgelist_adjacency <- class_edgelist_adjacency[what_order, ]
head(class_edgelist_adjacency)
```

```
##     sender receiver
## 1.1      1        3
## 1.2      1        5
## 1.3      1        7
## 1.4      1       21
## 2.1      2        3
## 2.2      2        6
```

De edgelist ziet er redelijk goed uit, en op dit punt kunnen we de nieuw geconstrueerde edgelist nemen en er een igraph-object van maken, met behulp van de syntax van hierboven. Laat ons nu ook tonen hoe we deze verschillende stappen kunnen uitvoeren met het `network` package

5.3. Netwerken met het `network` package

In dit laatste deel demonstreren we hoe je dezelfde taken kunt uitvoeren als hierboven, maar dan met het `network`-package (Butts 2015) in plaats van `igraph`. Het enige verschil is dat het formaat en de functies iets anders zullen zijn met het `network`-package.

De objecten die door `network` worden gecreëerd, kunnen vervolgens worden gebruikt met andere pakketten, zoals `sna`, `ergm`, `latentnet`, enz., waarmee we netwerkberekeningen kunnen uitvoeren, grafieken kunnen maken en geavanceerde statistische modellen kunnen bouwen. Bovendien zijn de functies en mogelijkheden van `igraph` vaak anders dan de packages die bij `network` horen.

We starten met het loskoppelen van het `igraph`-packages en het laden van het `network`-package:

```
detach(package:igraph)
library(network)
```

5.3.1. Het gebruiken van een matrix om een netwerk te construeren

We maken een netwerkobject aan met als invoer een matrix. De functie die hiervoor wordt gebruikt is `network()`. De belangrijkste argumenten zijn:

- `x`: de naam van de invoermatrix;
- `directed`: TRUE of FALSE, naargelang het netwerk gericht is;
- `vertex.attr`: een lijst met node attributen.

```
class_matrix_net_2 <- network(x = class_mat, directed = TRUE)
class_matrix_net_2
```

```
##  Network attributes:
##   vertices = 24
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 77
##     missing edges= 0
##     non-missing edges= 77
##
##  Vertex attribute names:
##     vertex.names
##
## No edge attributes
```

Het is mogelijk om node attributen meteen toe te voegen bij het aanmaken van het netwerk. Hiervoor moeten de attributen als lijst worden doorgegeven aan het argument `vertex.attr`. Categorische variabelen zoals gender en etniciteit worden het best omgezet naar een `character`, om latere complicaties met te vermijden:

```
class_att$race <- as.character(class_att$race)
class_att$gender <- as.character(class_att$gender)
```

Vervolgens zetten we het data frame van attributen om naar een lijst met behulp van de `do.call()` functie, zodat elke kolom als een afzonderlijk element wordt geïnterpreteerd:

```
attribute_list <- do.call(list, class_att)
attribute_list
```

```
## $id
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
##
## $gender
##  [1] "Male"   "Female" "Female" "Female" "Male"   "Female" "Male"   "Male"
##  [9] "Male"   "Female" "Female" "Female" "Male"   "Male"   "Female" "Male"
## [17] "Female" "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##
## $grade
##  [1] 12 12 12 12 12 12 11 11 11 11 10 10 10 10 10 10 10 10 10 10 10 10 10 10
##
## $race
##  [1] "White" "White" "White" "White" "White" "White" "Black" "White" "White"
## [10] "White" "White" "White" "White" "White" "White" "White" "White" "White"
## [19] "White" "White" "White" "White" "White" "White"
```

Daarna maken we het netwerk opnieuw aan, ditmaal met de lijst van node attributen:

```
class_matrix_net_2 <- network(x = class_mat, directed = TRUE,
                                    vertex.attr = attribute_list)
class_matrix_net_2
```

```
##  Network attributes:
```

```
##   vertices = 24
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 77
##     missing edges= 0
##     non-missing edges= 77
##
##  Vertex attribute names:
##     gender grade id race vertex.names
##
## No edge attributes
```

Een alternatief voor het toevoegen van alle attributen in één keer is om attributen één voor één toe te voegen met `set.vertex.attribute()`. De vereiste argumenten zijn:

- `x`: het netwerkobject;
- `attrname`: de naam van het attribuut dat men wil toevoegen;
- `value`: een vector met de waarden van het attribuut.

In het onderstaande voorbeeld voegen we een nieuw attribuut toe, `gradenew`, gebaseerd op het bestaande `grade` attribuut:

```
set.vertex.attribute(x = class_matrix_net_2,
                     attrname = "gradenew",
                     value = class_att$grade)
class_matrix_net_2
```

```
##  Network attributes:
##   vertices = 24
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 77
##     missing edges= 0
##     non-missing edges= 77
##
##  Vertex attribute names:
##     gender grade gradenew id race vertex.names
##
## No edge attributes
```

Merk op dat deze functie het netwerkobject in-place bijwerkt; er is dus geen nood om het resultaat opnieuw toe te wijzen aan een object.

5.3.2. Het gebruiken van een edgelist om een netwerk te construeren

We construeren opnieuw het netwerk, maar deze keer gebruiken we de edgelist als input met het `network`-package. Net zoals bij gebruik van een matrix gebruiken we de `network()`-functie, waarbij nu de edgelist het primaire invoerargument vormt. Attributen worden toegevoegd via het argument `vertices`, dat een data frame vereist waarvan de eerste kolom de id's van de nodes bevat.

Deze aanpak is bijzonder nuttig bij het verwerken van geïsoleerde nodes en bij het garanderen van een consistente ordening van de nodes. We gebruiken de `network()`-functie als volgt:

```
class_edge_net_2 <- network(x = class_edges, directed = TRUE,
                                          vertices = class_att)
class_edge_net_2
```

```
##  Network attributes:
##   vertices = 24
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 77
##     missing edges= 0
##     non-missing edges= 77
##
##  Vertex attribute names:
##     gender grade race vertex.names
##
##  Edge attribute names:
##     weight
```

Zoals eerder kunnen we ook uit een network-object de matrix, edgelist of node attributen extraheren. De netwerkmatrix kan opnieuw worden opgevraagd via:

```
as.matrix(class_edge_net_2)
```

```
##    1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
## 1  0 0 1 0 1 0 1 0 0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
## 2  0 0 1 0 0 1 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 3  0 0 0 0 0 1 0 1 0  0  0  0  0  0  0  1  0  0  0  0  0  0  1
## 4  0 0 0 0 0 0 0 0 0  0  0  0  1  0  0  0  0  1  0  0  0  0  0
## 5  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 6  0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 7  1 0 0 0 0 0 0 0 1  1  0  0  0  0  0  1  0  0  0  0  0  0  0
## 8  0 0 1 0 0 0 0 0 1  0  0  0  1  0  0  0  0  0  0  0  0  0  0
## 9  0 0 0 0 1 0 0 1 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 10 0 0 0 0 0 1 0 0 0  0  0  0  0  1  0  0  0  0  1  1  0  0  1
## 11 0 0 0 0 0 0 0 0 0  0  0  1  0  0  1  0  0  1  0  0  0  0  1
## 12 0 0 0 0 0 0 0 0 0  0  1  0  0  0  1  0  0  0  0  0  0  0  1
## 13 0 0 0 0 0 0 0 1 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## 14 0 0 0 0 0 0 0 0 0  1  0  0  1  0  0  0  0  0  1  0  1  0  0  1
## 15 0 0 0 0 0 0 0 0 0  1  1  0  1  1  0  0  0  0  0  0  0  0  0  1
## 16 0 0 1 0 1 0 0 0 1  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
```

```
## 17 0 0 0 0 0 0 0 1 0  0  0  0  1  0  0  0  0  1  0  0  0  0  1  1
## 18 0 0 0 0 0 0 0 0 0  0  0  0  1  0  0  0  1  0  0  0  0  0  1  1
## 19 0 0 0 0 0 0 0 0 0  0  0  0  0  1  0  1  0  0  0  1  1  0  0  0
## 20 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  1
## 21 0 0 0 0 1 0 0 0 0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0
## 22 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
## 23 0 0 0 0 1 0 0 0 0  0  0  0  1  0  0  0  1  1  0  0  0  0  0  0  0
## 24 0 0 0 0 0 1 0 0 0  1  0  0  0  1  1  0  0  0  0  0  1  0  0  0  0
```

Om node attributen op te vragen gebruiken we `get.vertex.attribute()`. Bijvoorbeeld, om het `grade`-attribuut te verkrijgen:

```
get.vertex.attribute(x = class_edge_net_2, attrname = "grade")
```

```
##  [1] 12 12 12 12 12 12 11 11 11 11 10 10 10 10 10 10 10 10 10 10 10 10 10 10
```

Om te controleren of de attributen correct zijn toegewezen, kunnen we ook de `vertex.names` opvragen:

```
get.vertex.attribute(x = class_edge_net_2, attrname = "vertex.names")
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
```

5.3.3. Toevoegen van attributen aan het netwerk

Indien er edge attributen beschikbaar zijn (zoals relatiegewichten), kunnen deze aan het netwerk worden toegevoegd met `set.edge.attribute()`. De relevante argumenten zijn:

- `x`: het netwerkobject
- `attrname`: de naam van het attribuut
- `value`: een vector met waarden voor elke edge

Bijvoorbeeld, om het `weight`-attribuut toe te voegen:

```
set.edge.attribute(x = class_edge_net_2,
                   attrname = "weight", value = class_edges[, "weight"])
class_edge_net_2
```

```
##  Network attributes:
##   vertices = 24
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 77
##     missing edges= 0
##     non-missing edges= 77
##
##  Vertex attribute names:
```

```
##     gender grade race vertex.names
##
##  Edge attribute names:
##     weight
```

Deze edge attributen kunnen vervolgens worden opgevraagd met:

```
get.edge.attribute(class_edge_net_2, attrname = "weight")
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [77] 1
```

5.3.4. Functionaliteit tussen `igraph` en `network`

Het is vaak eveneens wenselijk om netwerkobjecten tussen het `igraph` en `network` package te converteren. Dit maakt het mogelijk om bijvoorbeeld visualisatie in het ene pakket uit te voeren en analyses in het andere.

Hiervoor gebruiken we het `intergraph`-package (Bojanowski, 2015):

```
#install.packages('intergraph')
library(intergraph)
```

```
## Warning: package 'intergraph' was built under R version 4.3.3
```

Dit package bevat twee kernfuncties:

- `asIgraph()`: converteert een `network`-object naar een `igraph`-object
- `asNetwork()`: converteert een `igraph`-object naar een `network`-object

Voorbeeld een conversie van een `igraph`-object naar een `network`-object:

```
network_from_igraph <- asNetwork(class_edge_net)
network_from_igraph
```

```
##  Network attributes:
##   vertices = 24
##   directed = TRUE
##   hyper = FALSE
##   loops = FALSE
##   multiple = FALSE
##   bipartite = FALSE
##   total edges= 77
##     missing edges= 0
##     non-missing edges= 77
##
##  Vertex attribute names:
##     gender grade race vertex.names
##
##  Edge attribute names:
##     weight
```

Deze conversie stelt gebruikers in staat om te profiteren van de sterktes van beide packages, afhankelijk van de analytische behoeften.

Einde van deze notebook