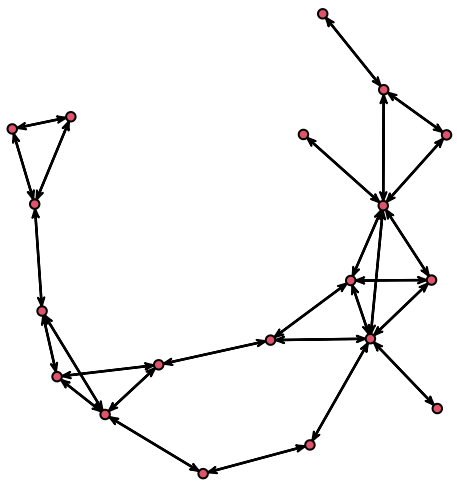
Ejercicio Práctico

La siguiente red muestra los contactos entre los terroristas que llevaron a cabo los secuestros del 11 de septiembre en 2001.

```
#install.packages("devtools")
library("devtools")
devtools::install_github("DougLuke/UserNetR")
library("statnet")
library("UserNetR")
library("tidyverse")
data("Krebs")
data_911 <- Krebs

adjacency_matrix <- as.sociomatrix(data_911)
plot.network(network(adjacency_matrix))
```



Los datos se encuentran almacenados en forma de una “adjacency matrix”, una matrix cuadrada donde cada fila (i)/columna (j) corresponde a un terrorista. Si la celda (ij) contiene un 1 significa que el terrorista i y el

terrorista j estaban relacionaos. Los ceros indican ausencia de relación. Por definición las celdas en que i=j contienen ceros. La "adjacency matrix" se ve así:

```
adjacency_matrix
```

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

Otra forma de almacenar la misma información es usando una "edgelist", es decir, una base de datos en cada fila corresponde a una relación entre un par de terroristas (ego y alter). Los valores en la variables ego y alter corresponden a los identificadores de cada terroristas (i/j en la adjacency matrix). Cuando dos terroristas no están relacionados entre si la fila correspondiente es omitida de la base de datos. La edgelist se ve así:

```
edgelist <- read.csv(url("https://raw.githubusercontent.com/mebucca/dar_soc4001/master/slides/class_10/

edgelist
```

```
##    ego alter
## 1    1     2
## 2    2     3
## 3    2     5
## 4    3     5
## 5    4     5
## 6    5     6
## 7    5     7
## 8    5    11
## 9    6     7
## 10   6    11
## 11   7     8
## 12   7    11
## 13   8    11
## 14   8     9
## 15   9    14
## 16   9    15
```

```
## 17  10    11
## 18  11    12
## 19  12    13
## 20  13    14
## 21  14    15
## 22  14    16
## 23  15    16
## 24  16    17
## 25  17    18
## 26  17    19
## 27  18    19
## 28   2     1
## 29   3     2
## 30   5     2
## 31   5     3
## 32   5     4
## 33   6     5
## 34   7     5
## 35  11     5
## 36   7     6
## 37  11     6
## 38   8     7
## 39  11     7
## 40  11     8
## 41   9     8
## 42  14     9
## 43  15     9
## 44  11    10
## 45  12    11
## 46  13    12
## 47  14    13
## 48  15    14
## 49  16    14
## 50  16    15
## 51  17    16
## 52  18    17
## 53  19    17
## 54  19    18
```

Ejercicios:

1. Pasar datos de ancho a largo para transformar la "edgelist" de tal forma que sea como la "adjacency matrix"

```
adjacency_tibble <- edgelist %>% mutate(edge = 1) %>%
  complete(ego,alter, fill=list(edge = 0)) %>%
  pivot_wider(names_from = alter, values_from = edge)

adjacency_tibble
```

```
## # A tibble: 19 x 20
##      ego   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`  `11`  `12`
##    <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
##  1     1     0     1     0     0     0     0     0     0     0     0     0     0
##  2     2     1     0     1     0     1     0     0     0     0     0     0     0
##  3     3     0     1     0     0     1     0     0     0     0     0     0     0
##  4     4     0     0     0     0     1     0     0     0     0     0     0     0
##  5     5     0     1     1     1     0     1     1     0     0     0     1     0
##  6     6     0     0     0     0     1     0     1     0     0     0     1     0
##  7     7     0     0     0     0     1     1     0     1     0     0     1     0
##  8     8     0     0     0     0     0     0     1     0     1     0     1     0
##  9     9     0     0     0     0     0     0     0     1     0     0     0     0
## 10    10     0     0     0     0     0     0     0     0     0     0     1     0
## 11    11     0     0     0     0     1     1     1     1     0     1     0     1
## 12    12     0     0     0     0     0     0     0     0     0     0     1     0
## 13    13     0     0     0     0     0     0     0     0     0     0     0     1
## 14    14     0     0     0     0     0     0     0     0     1     0     0     0
## 15    15     0     0     0     0     0     0     0     0     1     0     0     0
## 16    16     0     0     0     0     0     0     0     0     0     0     0     0
## 17    17     0     0     0     0     0     0     0     0     0     0     0     0
## 18    18     0     0     0     0     0     0     0     0     0     0     0     0
## 19    19     0     0     0     0     0     0     0     0     0     0     0     0
## # ... with 7 more variables: `13` <dbl>, `14` <dbl>, `15` <dbl>, `16` <dbl>,
## #   `17` <dbl>, `18` <dbl>, `19` <dbl>
## # i Use `colnames()` to see all variable names
```

2. Transforma la base de datos creada en 1 para que se vea nuevamente como la "edgelist".

```
edgelist_tibble <- adjacency_tibble %>%
  pivot_longer(-ego, names_to = "alter", values_to = "edge") %>%
  # filter(edge==1) %>%
  mutate(edge = if_else(edge==1,edge,NA_real_)) %>%
  drop_na(edge) %>%
  select(-edge)

edgelist_tibble
```

```
## # A tibble: 54 x 2
##      ego alter
##    <int> <chr>
##  1     1 2
##  2     2 1
##  3     2 3
##  4     2 5
##  5     3 2
##  6     3 5
##  7     4 5
##  8     5 2
##  9     5 3
## 10     5 4
## # ... with 44 more rows
## # i Use `print(n = ...)` to see more rows
```

3. Cargar la siguiente bases de datos World Inequality Data

```
options(repos = list(CRAN="http://cran.rstudio.com/"))
install.packages("devtools")
```

```
##
## The downloaded binary packages are in
##  /var/folders/0t/ft1mtmv168scq3sj9654ckz40000gn/T//RtmpchKS8T/downloaded_packages
```

```
devtools::install_github("WIDworld/wid-r-tool")
library("wid")

data_inequality <- download_wid(
  indicators = "shweal", # Shares of personal wealth
  areas = c("FR","GB","DE","US"),  # In France, Great Britain, Germany, USA
  perc = c("p0p50", "p90p100", "p99p100") # Bottom 50%, top 10% and top 1%
) %>% select(-variable) %>% arrange(country,year)

data_inequality %>% as_tibble()
```

```
## # A tibble: 2,512 x 4
##    country percentile  year value
##    <chr>   <chr>      <int> <dbl>
##  1 DE      p90p100     1895 0.863
##  2 DE      p99p100     1895 0.499
##  3 DE      p90p100     1896 0.863
##  4 DE      p99p100     1896 0.499
##  5 DE      p90p100     1897 0.869
##  6 DE      p99p100     1897 0.506
##  7 DE      p90p100     1899 0.870
##  8 DE      p99p100     1899 0.514
##  9 DE      p90p100     1902 0.868
## 10 DE      p99p100     1902 0.515
## # ... with 2,502 more rows
## # i Use `print(n = ...)` to see more rows
```

Hacer explícitos los NAs implícitos y rellenar valores perdidos con el valor del año anterior disponible para el mismo país en la misma variable chequear que los datos estén ordenados correctamente

```
data_inequality <- data_inequality %>% complete(country,percentile,year) %>%
  arrange(country,percentile,year) %>%
  group_by(country,percentile) %>%
  fill(value, .direction = "down")

data_inequality
```

```
## # A tibble: 3,112 x 4
## # Groups:   country, percentile [12]
##    country percentile  year value
##    <chr>   <chr>      <int> <dbl>
##  1 DE      p0p50       1807    NA
##  2 DE      p0p50       1817    NA
##  3 DE      p0p50       1827    NA
```

```
##  4 DE      p0p50       1837    NA
##  5 DE      p0p50       1847    NA
##  6 DE      p0p50       1857    NA
##  7 DE      p0p50       1867    NA
##  8 DE      p0p50       1877    NA
##  9 DE      p0p50       1887    NA
## 10 DE      p0p50       1895    NA
## # ... with 3,102 more rows
## # i Use 'print(n = ...)' to see more rows
```

4. Poner cada variable ("percentile") en una variable separadamente (wide)

```
data_inequality <- data_inequality %>%  mutate(id = row_number()) %>%
  pivot_wider(names_from = percentile, values_from = value) %>%
  select(-id)

data_inequality
```

```
## # A tibble: 1,663 x 5
## # Groups:   country [4]
##    country  year p0p50 p90p100 p99p100
##    <chr>   <int> <dbl>   <dbl>   <dbl>
##  1 DE       1807    NA  NA      NA
##  2 DE       1817    NA  NA      NA
##  3 DE       1827    NA  NA      NA
##  4 DE       1837    NA  NA      NA
##  5 DE       1847    NA  NA      NA
##  6 DE       1857    NA  NA      NA
##  7 DE       1867    NA  NA      NA
##  8 DE       1877    NA  NA      NA
##  9 DE       1887    NA  NA      NA
## 10 DE       1895    NA   0.863   0.499
## # ... with 1,653 more rows
## # i Use 'print(n = ...)' to see more rows
```