View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Diposit Digital de Documents de la UAB

REDES Revista Hispana para el Análisis de Redes Sociales

Vol.31,#2, (2020), 176-201

http://revistes.uab.es/redes https://doi.org/10.5565/rev/redes.883

Análisis de redes egocéntricas con R (II). Una red egocéntrica

Raffaele Vacca[1]
University of Florida

RESUMEN

Este texto es el segundo de una serie de cuatro que conjuntamente constituyen un taller sobre análisis de ego-redes (y/o redes personales) con R. El texto está acompañado por ficheros de datos y los scripts de lenguaje R necesarios para realizar las actividades propuestas.

Palabras clave: *Ego-redes - Redes personales – R.*

ABSTRACT

This text is the second of a series of four documents that together constitute a workshop on analysis of ego-networks (and / or personal networks) using R. The text is accompanied by data files and the R scripts necessary to carry out the suggested activities.

Key words: *Ego-networks – Personal networks – R.*

[1] *Contacto con los autores: Raffaele Vacca (r.vacca@ufl.edu)*

INTRODUCCIÓN

Después de aprender algunos conceptos básicos del lenguaje R, ahora nos centramos en las herramientas R para analizar datos de redes egocéntricas. Para simplificar, empezaremos con el análisis de una sola red egocéntrica.

Este documento cubre los siguientes temas:

- Almacenar datos de atributos de nivel de ego y datos de atributos de nivel de alter en R.
- Unir (fusionar) *dataframes* a nivel de ego y a nivel de alter.
- Almacenar datos de lazos alter-alter en R.
- Almacenar y manipular una red egocéntrica como un objeto "igraph".
- Visualizar una red egocéntrica.
- Calcular medidas de composición y estructura de la red egocéntrica.

DATOS AL NIVEL DE EGO Y AL NIVEL DE ALTER

Los datos de una red egocéntrica generalmente incluyen, como mínimo, tres tipos de conjuntos de datos:

- *Datos de atributos al nivel de alter (nivel 1)*. Son un conjunto de datos con atributos de alteri y/o relaciones ego-alter. Cada fila es un alter y cada columna es un atributo de la relación alter o ego-alter. Según la terminología de modelado multinivel, esto se llama "nivel 1", porque los alteri son las unidades básicas agrupadas dentro de los egos.

- *Datos de atributos de nivel de* ego *(nivel 2).* Son un conjunto de datos con atributos de egos. Cada fila es un ego (un encuestado) y cada columna es un atributo del ego. De acuerdo con la terminología de modelado multinivel, esto se llama "nivel 2", porque los egos son los grupos de nivel superior en los que se agrupan los alteri.

- *Datos de lazos entre alteri*. Estos son los datos sobre los lazos alter-alter según ha informado el ego.

Formatos de archivo

- *Datos de atributos de alter*. Dependiendo del software de recopilación de datos, es posible que tengamos un único conjunto de datos (en un solo archivo) que incluya alteri designados por todos los egos, o que tengamos conjuntos de datos separados (en archivos separados) para los alteri de cada ego (pero con las mismas variables en cada conjunto de datos).

- *Datos de los lazos de los alteri*. Normalmente, están en el formato de matriz de adyacencia o en el formato *edgelist*, según el software de recopilación de datos. Si los datos de lazos están en matrices de adyacencia, acostumbran a tener una matriz de adyacencia separada, en un archivo separado, para cada ego. Si están en *edgelists*, es posible tener una sola *edgelist* que incluya alteri de todos los egos (con una columna adicional que indique el ego que nominó a los dos alteri) o una *edgelist* separada (en un archivo separado) para cada ego.

- En este taller usamos principalmente datos de atributos del ego, editamos datos de atributos y modificamos datos de lazos como objetos ya almacenados en R. El script "R04_appendix.R" muestra cómo crear estos objetos R mediante la importación de archivos csv externos.

En el análisis egocéntrico frecuentemente se cambia entre los niveles 1 y 2 y se fusiona o une información de ambos niveles.

Unión de nivel 1

Para ciertos tipos de análisis tenemos que unir los atributos del ego (nivel 2) en un conjunto de datos de nivel alter (nivel 1).

- Como tenemos un ego para múltiples alteri, esta es una unión de uno a muchos, uniendo el mismo valor de un atributo del ego (una fila del ego) a múltiples alteri (múltiples filas de alter).

- - Los *dataframes* están vinculados por la identificación (ID) del alter.

Unión de nivel 2

En otros casos puede ser que queramos unir atributos de alter (nivel 1) en un conjunto de datos de nivel ego (nivel 2).

- Como existen múltiples alteri para un ego, primero, es necesario resumir o agregar los atributos de alter para cada ego.

- Si tenemos atributos continuos de los alteri, generalmente se resumen calculando promedios y medidas de dispersión (varianza, desviación estándar, etc.) de los atributos de los alteri para cada ego: por ejemplo, la edad media de los alteri para cada ego o la desviación estándar de frecuencia de contacto entre alteri y cada ego.

- Si tenemos atributos categóricos de los alteri, normalmente se resumen calculando proporciones de ciertas categorías o medidas de diversidad cualitativas (por ejemplo, varianza generalizada, entropía): por ejemplo, la proporción de mujeres o diversidad étnica en la red de cada ego.

- Una vez tengamos estas variables de resumen en la composición de la red egocéntrica, se pueden unir a un conjunto de datos a nivel del ego con la identificación (ID) de cada ego.

- Consideraremos el resumen y la unión de nivel 2 en las secciones siguientes sobre medidas de composición de la red egocéntrica y estructura de la red egocéntrica.

Unión de nivel 2

La unión de nivel 2 también puede involucrar variables de resumen en la estructura de la red egocéntrica: por ejemplo, grado medio de los alteri, densidad de la red egocéntrica o número de componentes. Estas variables también se pueden unir a un conjunto de datos a nivel del ego mediante la identificación (ID) de cada ego.

En R los *dataframes* se pueden unir usando la función "merge()". Sin embargo, utilizamos las funciones de unión "dplyr", cuyo código es más eficiente y legible:

- "left_ join( )" mantiene todas las filas en el *dataframe* izquierdo y descarta cualquier fila en el *dataframe* derecho que no coincida con una fila en el marco de datos izquierdo.

- "right_ join()" mantiene todas las filas en el *dataframe* derecho y descarta cualquier fila en el *dataframe*

izquierdo que no coincida con una fila en el *dataframe* derecho.

- "full_ join()" mantiene todas las filas de ambos *dataframes*.

McCarty y sus colegas (2019) ofrecen una discusión más detallada de todos los tipos de datos de red egocéntricas, niveles en datos de ego así como operaciones de cambio y unión entre niveles.

Las funciones del código que se presenta a continuación son las siguientes:

- Ver los datos del nivel de ego (nivel 2) y los datos de nivel de alter (nivel 1) tal como se almacenan en *dataframes* R.

Unir el nivel 1: llevar atributos de ego a un *dataframe* de nivel de alter.

```
# Load packages.
library(tidyverse)
## Registered S3 methods overwritten by 'ggplot2':
## method from
## [.quosures rlang
## c.quosures rlang
## print.quosures rlang
## -- Attaching packages --- tidyverse 1.2.1 --
## v ggplot2 3.1.1 v purrr 0.3.2
## v tibble 2.1.3 v dplyr 0.8.1
## v tidyr 0.8.3 v stringr 1.4.0
## v readr 1.3.1 v forcats 0.4.0
## -- Conflicts ------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag() masks stats::lag()

# Load data.
load("./Data/data.rda")

# View ego attributes
ego.df
## # A tibble: 102 x 9
## ego_ID ego.sex ego.age ego.arr ego.edu ego.inc empl ego.empl.bin
## <dbl> <fct> <dbl> <dbl> <fct> <dbl> <dbl> <fct>
## 1 28 Male 61 2008 Second~ 350 3 Yes
## 2 29 Male 38 2000 Primary 900 4 Yes
## 3 33 Male 30 2010 Primary 200 3 Yes

# View alter attributes
alter.attr.all
## # A tibble: 4,590 x 11
## alter_ID ego_ID alter_num alter.sex alter.age.cat alter.rel alter.nat
## <dbl> <dbl> <dbl> <fct> <dbl> <fct> <fct>
## 1 2801 28 1 Female 6 Close fa~ Sri Lanka
## 2 2802 28 2 Male 6 Other fa~ Sri Lanka
## 3 2803 28 3 Male 6 Close fa~ Sri Lanka
## 4 2804 28 4 Male 7 Close fa~ Sri Lanka
## 5 2805 28 5 Female 5 Close fa~ Sri Lanka
## 6 2806 28 6 Female 7 Close fa~ Sri Lanka
## 7 2807 28 7 Male 5 Other fa~ Sri Lanka
## 8 2808 28 8 Female 4 Other fa~ Sri Lanka
## 9 2809 28 9 Female 6 Other fa~ Sri Lanka
## 10 2810 28 10 Male 7 Other fa~ Sri Lanka
## # ... with 4,580 more rows, and 4 more variables: alter.res <fct>,
## # alter.clo <dbl>, alter.loan <fct>, alter.fam <fct>

# Level-1 join: Ego attributes into alter-level data.
(data <- left_join(alter.attr.all, ego.df, by= "ego_ID"))
## # A tibble: 4,590 x 19
## alter_ID ego_ID alter_num alter.sex alter.age.cat alter.rel alter.nat
```

```
## <dbl> <dbl> <dbl> <fct> <dbl> <fct> <fct>
## 1 2801 28 1 Female 6 Close fa~ Sri Lanka
## 2 2802 28 2 Male 6 Other fa~ Sri Lanka
## 3 2803 28 3 Male 6 Close fa~ Sri Lanka
## 4 2804 28 4 Male 7 Close fa~ Sri Lanka
## 5 2805 28 5 Female 5 Close fa~ Sri Lanka
## 6 2806 28 6 Female 7 Close fa~ Sri Lanka
## 7 2807 28 7 Male 5 Other fa~ Sri Lanka
## 8 2808 28 8 Female 4 Other fa~ Sri Lanka
## 9 2809 28 9 Female 6 Other fa~ Sri Lanka
## 10 2810 28 10 Male 7 Other fa~ Sri Lanka
## # ... with 4,580 more rows, and 12 more variables: alter.res <fct>,
## # alter.clo <dbl>, alter.loan <fct>, alter.fam <fct>, ego.sex <fct>,
## # ego.age <dbl>, ego.arr <dbl>, ego.edu <fct>, ego.inc <dbl>,
## # empl <dbl>, ego.empl.bin <fct>, ego.age.cat <dbl>

# Note that the new data frame has one row for each alter, and the same ego ID
# and ego attribute value is repeated for all alters belonging to the same ego.
dplyr::select(data, alter_ID, ego_ID, ego.age, ego.edu)
## # A tibble: 4,590 x 4
## alter_ID ego_ID ego.age ego.edu
## <dbl> <dbl> <dbl> <fct>
## 1 2801 28 61 Secondary
## 2 2802 28 61 Secondary
## 3 2803 28 61 Secondary
## 4 2804 28 61 Secondary
## 5 2805 28 61 Secondary
## 6 2806 28 61 Secondary
## 7 2807 28 61 Secondary
## 8 2808 28 61 Secondary
## 9 2809 28 61 Secondary
## 10 2810 28 61 Secondary
## # ... with 4,580 more rows
```

REDES EN R

Hay varios paquetes para el análisis de redes en R.

Los dos principales paquetes (o colecciones de paquetes) de red R son "igraph" y "statnet". "igraph" es un paquete único. Representa las redes como objetos de la clase "igraph". "statnet" no es un paquete, es una colección de paquetes. Representa las redes como objetos de la clase "network".

Los paquetes "igraph" y "statnet" en parte se superponen y en parte son complementarios. Históricamente, "igraph" se ha centrado más en los métodos de red desarrollados en informática y física, mientras que "statnet" se ha dedicado más a los métodos de red desarrollados en las ciencias sociales. Se pueden hacer varias cosas en ambos paquetes, "igraph" y "statnet". Incluyen la creación básica de la red, la importación de datos de la red, su manipulación, las métricas básicas tales como las de centralidad, y la visualización de la red. Por otro lado, algunas funciones solo se pueden hacer en "igraph" (por ejemplo, algoritmos de "detección de comunidades" y análisis de modularidad), algunas otras solo se pueden hacer en "statnet" (por ejemplo, ERGM).

En este taller utilizaremos principalmente el paquete "igraph". Esto se debe a que, personalmente, considero que la sintaxis de "igraph" es más intuitiva para principiantes, especialmente en operaciones simples de redes. También demostraremos brevemente cómo se almacenan las redes en objetos de "red" de "statnet". Se debe tener en cuenta que todo lo que haremos aquí con "igraph", también se puede hacer con "statnet". En general, hay que considerar que también tendremos que aprender a utilizar "statnet" si queremos tener un conjunto completo de herramientas analíticas para el análisis de redes sociales con R.

Se puede encontrar más información sobre el código y "statnet" en el archivo de script apéndice.

También usaremos brevemente el paquete "ggraph" para la visualización de la red, y el paquete "tidygraph" para facilitar la visualización y la manipulación de las redes. No entraremos en muchos detalles sobre estos dos

paquetes, pero se pueden encontrar más tutoriales y recursos para aprenderlos a utilizar en línea.

El paquete "ggraph" proporciona herramientas flexibles y fáciles para la visualización de redes al aplicar la gramática de gráficos "ggplot2" a los datos de red. Además de "ggraph", también se pueden visualizar redes con "igraph", que utiliza el módulo base de R (consúltese el apéndice).

El paquete "tidygraph" aplica los principios "tidyverse" de ordenación de datos (https://r4ds.had.co.nz/tidy-data.html) a las redes. Esto nos permite ver y manipular más fácilmente los componentes básicos de los datos de red en formato tabular, es decir, los datos como un *edgelist* y los datos de atributos de los nodos como un conjunto de datos tìpico de caso por variable.

El paquete "igraph"

En "igraph" las redes se llaman *graphs* y están representadas por objetos de la clase "igraph". Los nodos se llaman *vértices* y los lazos se llaman "edges".

Un objeto "gr" de igraph nos permite hacer lo siguiente:

- "V(gr)" muestra los vértices del gráfico (identificados por "nombre", si tienen un "nombre" de atributo o, de lo contrario, por enteros). Este es un objeto de clase secuencia de vértice ("igraph.vs").
- "E(gr)" muestra los bordes del gráfico. Este es un objeto de clase secuencia de lazo ("igraph.es").

Los nodos, lazos y gráficos tienen atributos. Se pueden importar desde archivos de datos externos, o se pueden configurar manualmente en R. Si importamos datos en un objeto "igraph" con un atributo de nodo llamado "edad", un atributo de lazo llamado "intensidad" y un atributo de gráfico llamado "tamaño", entonces:

- "V(gr)$edad" devuelve el atributo de nodo *edad* como un vector;
- "E(gr)$intensidad" devuelve el atributo de lazo *intensidad* como un vector;
- "gr$tamaño" devuelve el atributo de gráfico *tamaño*.

La opción "print()" permite imprimir un objeto "igraph" y devuelve alguna información de resumen sobre la red. Esto incluye cuántos nodos y lazos tiene, ya sea dirigidos, etiquetados (es decir, si los nodos tienen un atributo de "nombre"), ponderados (es decir, si los bordes tienen un atributo de "peso") o bipartitos (también conocido como red de modo 2).

Se pueden consultar nodos y lazos:

- *En base a los atributos*. Se pueden consultar nodos y lazos con atributos específicos (=valores de atributos) y guardarlos para reutilizarlos. Por ejemplo, "V(gr)[edad=30]" devuelve todos los nodos cuyo atributo "edad" es igual a 30; "E(gr)[intensidad=1]" devuelve todos los lazos cuya intensidad es 1.
- En base a la estructura de la red. La sintaxis "V(gr)[...]" y "E(gr)[]" también se puede usar con funciones específicas como por ejemplo, para consultar todos los nodos adyacentes a un nodo dado, o todos los lazos entre dos subconjuntos particulares de nodos. Las funciones principales aquí son "nei()", "inc()" y "%--%" (véase el código a continuación).
- Más información sobre la sintaxis de igraph útil para la indexación de nodos y lazos:

Para agregar nodos y lazos "igraph" tiene una sintaxis muy intuitiva:

- Si "gr" es un gráfico, "gr+"foo" agrega un vértice de nombres "foo" al gráfico.
- Para agregar bordes, podemos simplemente indexar "gr" como una matriz y asignar "1" donde queramos un borde.
- Existen otras formas y opciones más flexibles de agregar vértices y bordes: véase "help("+.igraph")".
- De forma alternativa, podemos usar "tidygraph" para agregar bordes simplemente agregando filas a los datos de la lista de bordes de la red egocéntrica (véase el código a continuación).
  - Esto es útil si tenemos redes egocéntricas sin el nodo del ego y necesitamos agregarlo para ciertos tipos de análisis (por ejemplo, para visualizar la red con el ego o ejecutar cálculos que requieren ego en la red).

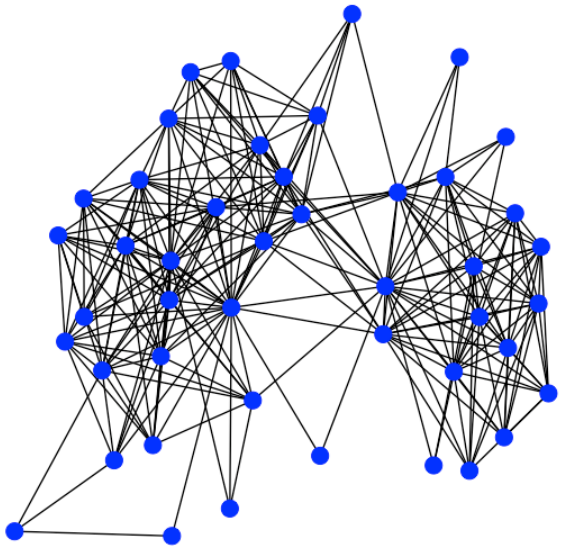
Las funciones del código que se ofrece a continuación son las siguientes:

- Buscar los atributos de vértice y borde en una red egocéntrica.
- Obtener estadísticas descriptivas para atributos de alteri (atributos de vértice) y lazos alter-alter (atributos de borde) en la red egocéntrica.
- Visualizar una red egocéntrica con "ggraph", estableciendo parámetros estéticos basados en atributos alter.
- Buscar vértices y bordes según atributos.

Agregar el vértice del ego a la red usando "tidygraph".

```
# Load packages.
library(igraph)
##
## Attaching package: 'igraph'
## The following objects are masked from 'package:dplyr':
##
## as_data_frame, groups, union
## The following objects are masked from 'package:purrr':
##
## compose, simplify
## The following object is masked from 'package:tidyr':
##
## crossing
## The following object is masked from 'package:tibble':
##
## as_data_frame
## The following objects are masked from 'package:stats':
##
## decompose, spectrum
## The following object is masked from 'package:base':
##
## union
library(ggraph)
library(tidygraph)
##
## Attaching package: 'tidygraph'
## The following object is masked from 'package:igraph':
##
## groups
## The following object is masked from 'package:stats':
##
## filter
library(summarytools)
## Registered S3 method overwritten by 'pryr':
## method from
## print.bytes Rcpp
## Warning in system2("/usr/bin/otool", c("-L", shQuote(DSO)), stdout = TRUE):
## running command ''/usr/bin/otool' -L '/Library/Frameworks/R.framework/
## Resources/library/tcltk/libs//tcltk.so'' had status 69
##
## Attaching package: 'summarytools'
## The following object is masked from 'package:tibble':
##
## view

# Get ego-network of ego 28
gr <- gr.28

# Show the graph. Note that ego is not included.
set.seed(607)
ggraph(gr) +
```

```
geom_edge_link() + # Draw edges
geom_node_point(size=5, color="blue") + # Draw nodes
theme_graph(base_family = 'Helvetica') # Set graph theme and font
## Using `nicely` as default layout
```



```
# Print the graph.
gr
## IGRAPH 4a3bad2 UNW- 45 259 --
## + attr: ego_ID (g/c), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), weight (e/n)
## + edges from 4a3bad2 (vertex names):
## [1] 2801--2802 2801--2803 2801--2804 2801--2805 2801--2806 2801--2807
## [7] 2801--2808 2801--2809 2801--2810 2801--2811 2801--2812 2801--2813
## [13] 2801--2814 2801--2815 2801--2818 2801--2820 2801--2823 2801--2825
## [19] 2801--2827 2801--2828 2801--2829 2801--2831 2801--2840 2801--2841
## [25] 2802--2803 2802--2804 2802--2805 2802--2806 2802--2807 2802--2808
## + ... omitted several edges

# The graph is Undirected, Named, Weighted. It has 45 vertices and 259 edges. It
# has a vertex attribute called "name", and an edge attribute called "weight".
# We see several vertex attributes (see codebook.xlsx for their meaning).
# An igraph object can be indexed as an adjacency matrix.
# Adjacency row of alter #3.
gr[3,]
## 2801 2802 2803 2804 2805 2806 2807 2808 2809 2810 2811 2812 2813 2814 2815
## 1 1 0 1 1 1 1 1 1 1 1 1 0 0 0
## 2816 2817 2818 2819 2820 2821 2822 2823 2824 2825 2826 2827 2828 2829 2830
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## 2831 2832 2833 2834 2835 2836 2837 2838 2839 2840 2841 2842 2843 2844 2845
## 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
gr["2803",]
## 2801 2802 2803 2804 2805 2806 2807 2808 2809 2810 2811 2812 2813 2814 2815
## 1 1 0 1 1 1 1 1 1 1 1 1 0 0 0
## 2816 2817 2818 2819 2820 2821 2822 2823 2824 2825 2826 2827 2828 2829 2830
## 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## 2831 2832 2833 2834 2835 2836 2837 2838 2839 2840 2841 2842 2843 2844 2845
## 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
```

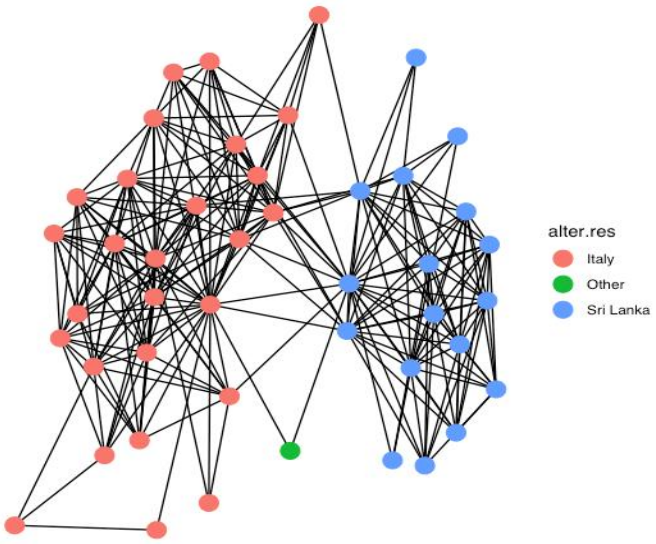
```
# Adjacency columns of alters 3-5
gr[,3:5]
## 45 x 3 sparse Matrix of class "dgCMatrix"
## 2803 2804 2805
## 2801 1 1 1
## 2802 1 1 1
## 2803 . 1 1
## 2804 1 . 1
## 2805 1 1 .
## 2806 1 1 1
## 2807 1 1 1
## 2808 1 1 1
## 2809 1 1 1
## 2810 1 1 1
## 2811 1 2 1
## 2812 1 . .
## 2813 . . .
## 2814 . . .
## 2815 . . .
## 2816 . . .
## 2817 . . .
## 2818 . . .
## 2819 . . .
## 2820 . . .
## 2821 . . .
## 2822 . . .
## 2823 . . .
## 2824 . . .
## 2825 . . .
## 2826 . . .
## 2827 . . .
## 2828 . . .
## 2829 . . .
## 2830 . . .
## 2831 . . .
## 2832 . . .
## 2833 . . .
## 2834 . . .
## 2835 . . .
## 2836 . . .
## 2837 . . .
## 2838 . . .
## 2839 . . .
## 2840 1 2 1
## 2841 1 1 1
## 2842 . . .
## 2843 . . .
## 2844 . . .
## 2845 . . .

# Convert and view the ego-network as a tidygraph object: you can now see the
# attribute data and edge list as tabular datasets.
(gr.tbl <- as_tbl_graph(gr))
## # A tbl_graph: 45 nodes and 259 edges
## #
## # An undirected simple graph with 1 component
## #
## # Node Data: 45 x 10 (active)
## name alter_num alter.sex alter.age.cat alter.rel alter.nat alter.res
## <chr> <dbl> <chr> <dbl> <chr> <chr> <chr>
```

```
## 1 2801 1 Female 6 Close fa~ Sri Lanka Sri Lanka
## 2 2802 2 Male 6 Other fa~ Sri Lanka Sri Lanka
## 3 2803 3 Male 6 Close fa~ Sri Lanka Sri Lanka
## 4 2804 4 Male 7 Close fa~ Sri Lanka Sri Lanka
## 5 2805 5 Female 5 Close fa~ Sri Lanka Sri Lanka
## 6 2806 6 Female 7 Close fa~ Sri Lanka Sri Lanka
## # ... with 39 more rows, and 3 more variables: alter.clo <dbl>,
## # alter.loan <chr>, alter.fam <chr>
## #
## # Edge Data: 259 x 3
## from to weight
## <int> <int> <dbl>
## 1 1 2 1
## 2 1 3 1
## 3 1 4 1
## # ... with 256 more rows

# Vertex and edge sequences and attributes ----
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Vertex sequences and edge sequences can be extracted from a graph.
# Vertex sequence of the whole graph. Notice that vertices are displayed by
# "names" (alter IDs in this case).
V(gr)
## + 45/45 vertices, named, from 4a3bad2:
## [1] 2801 2802 2803 2804 2805 2806 2807 2808 2809 2810 2811 2812 2813 2814
## [15] 2815 2816 2817 2818 2819 2820 2821 2822 2823 2824 2825 2826 2827 2828
## [29] 2829 2830 2831 2832 2833 2834 2835 2836 2837 2838 2839 2840 2841 2842
## [43] 2843 2844 2845

# Edge sequence of the whole graph. Notice that vertex names (alter IDs) are
# used here too.
E(gr)
## + 259/259 edges from 4a3bad2 (vertex names):
## [1] 2801--2802 2801--2803 2801--2804 2801--2805 2801--2806 2801--2807
## [7] 2801--2808 2801--2809 2801--2810 2801--2811 2801--2812 2801--2813
## [13] 2801--2814 2801--2815 2801--2818 2801--2820 2801--2823 2801--2825
## [19] 2801--2827 2801--2828 2801--2829 2801--2831 2801--2840 2801--2841
## [25] 2802--2803 2802--2804 2802--2805 2802--2806 2802--2807 2802--2808
## [31] 2802--2809 2802--2810 2802--2811 2802--2812 2802--2813 2802--2815
## [37] 2802--2823 2802--2831 2802--2840 2802--2841 2803--2804 2803--2805
## [43] 2803--2806 2803--2807 2803--2808 2803--2809 2803--2810 2803--2811
## [49] 2803--2812 2803--2840 2803--2841 2804--2805 2804--2806 2804—2807
## [55] 2804--2808 2804--2809 2804--2810 2804--2811 2804--2840 2804--2841
## + ... omitted several edges

# View vertex attributes and calculate statistics on them.
# Emotional closeness of alters to ego.
V(gr)$alter.clo
## [1] NA 3 NA NA NA NA 5 4 5 5 5 4 3 5 3 3 1 5 4 5 5 5 5
## [24] 5 2 3 4 5 5 5 5 3 5 3 4 5 5 5 5 3 3 3 3 3 5

# Mean closeness of alters to ego.
mean(V(gr)$alter.clo, na.rm=TRUE)
## [1] 4.1
# More decriptive statistics on emotional closeness.
summarytools::descr(V(gr)$alter.clo)
## Descriptive Statistics
## value
## N: 45
##
## value
```

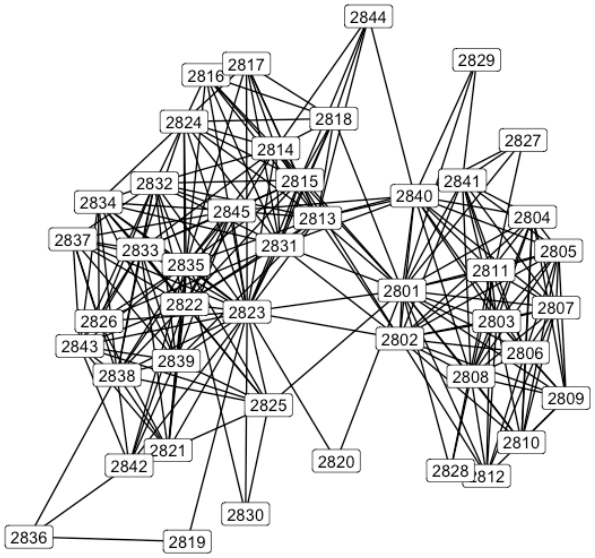
```
## ----------------- -------
## Mean 4.10
## Std.Dev 1.08
## Min 1.00
## Q1 3.00
## Median 5.00
## Q3 5.00
## Max 5.00
## MAD 0.00
## IQR 2.00
## CV 0.26
## Skewness -0.79
## SE.Skewness 0.37
## Kurtosis -0.38
## N.Valid 40.00
## Pct.Valid 88.89
# Alter's country of residence.
V(gr)$alter.res
## [1] "Sri Lanka" "Sri Lanka" "Sri Lanka" "Sri Lanka" "Sri Lanka"
## [6] "Sri Lanka" "Sri Lanka" "Sri Lanka" "Sri Lanka" "Sri Lanka"
## [11] "Sri Lanka" "Sri Lanka" "Italy" "Italy" "Italy"
## [16] "Italy" "Italy" "Italy" "Italy" "Other"
## [21] "Italy" "Italy" "Italy" "Italy" "Italy"
## [26] "Italy" "Sri Lanka" "Sri Lanka" "Sri Lanka" "Italy"
## [31] "Italy" "Italy" "Italy" "Italy" "Italy"
## [36] "Italy" "Italy" "Italy" "Italy" "Sri Lanka"
## [41] "Sri Lanka" "Italy" "Italy" "Italy" "Italy"
# Frequencies.
summarytools::freq(V(gr)$alter.res)
## Frequencies
##
## Freq % Valid % Valid Cum. % Total % Total Cum.
## --------------- ------ --------- -------------- --------- --------------
## Italy 27 60.00 60.00 60.00 60.00
## Other 1 2.22 62.22 2.22 62.22
## Sri Lanka 17 37.78 100.00 37.78 100.00
## <NA> 0 0.00 100.00
## Total 45 100.00 100.00 100.00 100.00
# This is more readable with the pipe operator (but make sure that freq() is
# actually being taken from summarytools).
V(gr)$alter.res %>% freq
## Frequencies
##
## Freq % Valid % Valid Cum. % Total % Total Cum.
## --------------- ------ --------- -------------- --------- --------------
## Italy 27 60.00 60.00 60.00 60.00
## Other 1 2.22 62.22 2.22 62.22
## Sri Lanka 17 37.78 100.00 37.78 100.00
## <NA> 0 0.00 100.00
## Total 45 100.00 100.00 100.00 100.00
# Alter IDs are stored in the "name" vertex attribute
V(gr)$name
## [1] "2801" "2802" "2803" "2804" "2805" "2806" "2807" "2808" "2809" "2810"
## [11] "2811" "2812" "2813" "2814" "2815" "2816" "2817" "2818" "2819" "2820"
## [21] "2821" "2822" "2823" "2824" "2825" "2826" "2827" "2828" "2829" "2830"
## [31] "2831" "2832" "2833" "2834" "2835" "2836" "2837" "2838" "2839" "2840"
## [41] "2841" "2842" "2843" "2844" "2845"
# We can also create new vertex attributes.
V(gr)$new.attribute <- 1:45
# Now a new attribute is listed for the graph.
gr
```

```
## IGRAPH 4a3bad2 UNW- 45 259 --
## + attr: ego_ID (g/c), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), new.attribute (v/n), weight (e/n)
## + edges from 4a3bad2 (vertex names):
## [1] 2801--2802 2801--2803 2801--2804 2801--2805 2801--2806 2801--2807
## [7] 2801--2808 2801--2809 2801--2810 2801--2811 2801--2812 2801--2813
## [13] 2801--2814 2801--2815 2801--2818 2801--2820 2801--2823 2801--2825
## [19] 2801--2827 2801--2828 2801--2829 2801--2831 2801--2840 2801--2841
## [25] 2802--2803 2802--2804 2802--2805 2802--2806 2802--2807 2802--2808
## + ... omitted several edges

# Also edges have attributes, in this case "weight".
E(gr)$weight
## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1
## [71] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1
## [106] 1 1 2 1 2 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2
## [141] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [176] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1
## [211] 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1
## [246] 1 1 1 1 1 1 1 1 1 1 1 2 1 1

# Frequencies of edge weights.
freq(E(gr)$weight)
## Frequencies
##
## Freq % Valid % Valid Cum. % Total % Total Cum.
## ----------- ------ --------- -------------- --------- --------------
## 1 235 90.73 90.73 90.73 90.73
## 2 24 9.27 100.00 9.27 100.00
## <NA> 0 0.00 100.00
## Total 259 100.00 100.00 100.00 100.00

# Also graphs have attributes, and we can query or assign them. For example,
# assign the "ego_ID" attribute for this graph: this is the personal network of
# ego ID 28.
gr$ego_ID <- 28

# Check that the new attribute was created
gr
## IGRAPH 4a3bad2 UNW- 45 259 --
## + attr: ego_ID (g/n), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), new.attribute (v/n), weight (e/n)
## + edges from 4a3bad2 (vertex names):
## [1] 2801--2802 2801--2803 2801--2804 2801--2805 2801--2806 2801--2807
## [7] 2801--2808 2801--2809 2801--2810 2801--2811 2801--2812 2801--2813
## [13] 2801--2814 2801--2815 2801--2818 2801--2820 2801--2823 2801--2825
## [19] 2801--2827 2801--2828 2801--2829 2801--2831 2801--2840 2801--2841
## [25] 2802--2803 2802--2804 2802--2805 2802--2806 2802--2807 2802--2808
## + ... omitted several edges

# Displaying vertex attributes in network visualization ----
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Show the graph, using different colors for different countries of residence.
# Vertex attribute with alter's country of residence.
V(gr)$alter.res
## [1] "Sri Lanka" "Sri Lanka" "Sri Lanka" "Sri Lanka" "Sri Lanka"
```

```
## [6] "Sri Lanka" "Sri Lanka" "Sri Lanka" "Sri Lanka" "Sri Lanka"
## [11] "Sri Lanka" "Sri Lanka" "Italy" "Italy" "Italy"
## [16] "Italy" "Italy" "Italy" "Italy" "Other"
## [21] "Italy" "Italy" "Italy" "Italy" "Italy"
## [26] "Italy" "Sri Lanka" "Sri Lanka" "Sri Lanka" "Italy"
## [31] "Italy" "Italy" "Italy" "Italy" "Italy"
## [36] "Italy" "Italy" "Italy" "Italy" "Sri Lanka"
## [41] "Sri Lanka" "Italy" "Italy" "Italy" "Italy"

# Plot with alter.res as node color
set.seed(607)
ggraph(gr) +
geom_edge_link() + # Draw edges
geom_node_point(aes(color= alter.res), size=5) + # Draw nodes setting alter.res
# as node color and fixed node size
theme_graph(base_family = 'Helvetica')
## Using `nicely` as default layout
```

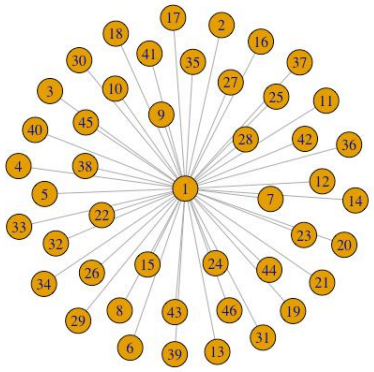



```
# Plot with alter ID labels instead of circles
set.seed(607)
ggraph(gr) +
geom_edge_link() +
geom_node_label(aes(label= name)) +
theme_graph(base_family = 'Helvetica')
## Using `nicely` as default layout
```



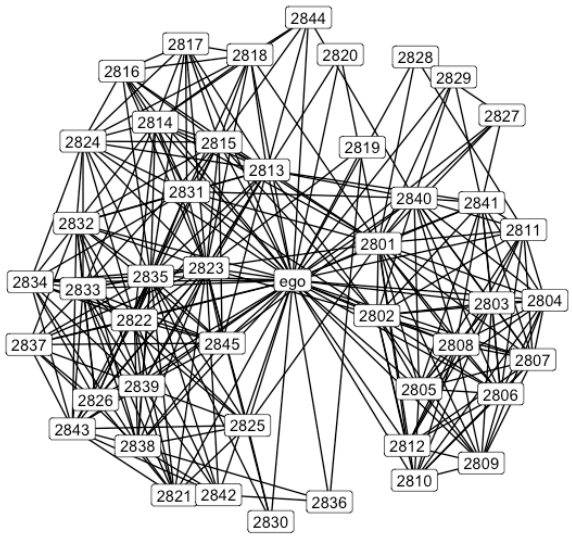

```
# Indexing vertices and edges based on attributes or network structure ----
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# View only female alters.
V(gr)[alter.sex=="Female"]
## + 8/45 vertices, named, from 4a3bad2:
## [1] 2801 2805 2806 2808 2809 2828 2832 2837

# View residence for women only.
V(gr)[alter.sex=="Female"]$alter.res
## [1] "Sri Lanka" "Sri Lanka" "Sri Lanka" "Sri Lanka" "Sri Lanka" "Sri Lanka"
## [7] "Italy" "Italy"

# Frequencies of countries of residence of female alters.
freq(V(gr)[alter.sex=="Female"]$alter.res)
## Frequencies
##
## Freq % Valid % Valid Cum. % Total % Total Cum.
## --------------- ------ --------- -------------- --------- --------------
## Italy 2 25.00 25.00 25.00 25.00
## Sri Lanka 6 75.00 100.00 75.00 100.00
## <NA> 0 0.00 100.00
## Total 8 100.00 100.00 100.00 100.00

# View nationality for alter 2833.
V(gr)[name=="2833"]$alter.nat
## [1] "Sri Lanka"

# Edit nationality for alter 2833
V(gr)[name=="2833"]$alter.nat <- "Italy"

# View all alters who know alter 2833 (vertices that are adjacent to vertex
# "2833").
V(gr)[nei("2833")]
## + 16/45 vertices, named, from 4a3bad2:
## [1] 2814 2815 2821 2822 2823 2824 2825 2826 2832 2834 2835 2837 2838 2839
## [15] 2843 2845

# View the sex of all alters who know alter 2833.
```

```
V(gr)[nei("2833")]$alter.sex
## [1] "Male" "Male" "Male" "Male" "Male" "Male" "Male"
## [8] "Male" "Female" "Male" "Male" "Female" "Male" "Male"
## [15] "Male" "Male"

# View all edges that are incident on alter 2833
E(gr)[inc("2833")]
## + 16/259 edges from 4a3bad2 (vertex names):
## [1] 2814--2833 2815--2833 2821--2833 2822--2833 2823--2833 2824--2833
## [7] 2825--2833 2826--2833 2832--2833 2833--2834 2833--2835 2833--2837
## [13] 2833--2838 2833--2839 2833--2843 2833--2845

# View the weight of these edges.
E(gr)[inc("2833")]$weight
## [1] 2 2 2 1 1 1 2 1 1 1 1 1 1 2 1

# Max weight of any edge that is incident on alter 2833.
E(gr)[inc("2833")]$weight %>% max
## [1] 2

# View all edges with weight==2 (alters "maybe" know each other).
E(gr)[weight==2]
## + 24/259 edges from 4a3bad2 (vertex names):
## [1] 2801--2818 2801--2829 2804--2811 2804--2840 2807--2811 2811--2812
## [7] 2811--2828 2813--2841 2813--2845 2814--2833 2815--2833 2816--2835
## [13] 2817--2835 2821--2826 2821--2833 2821--2839 2823--2830 2825--2833
## [19] 2827--2828 2830--2835 2831--2840 2833--2843 2834--2843 2840--2844

# Conditions in R indexing can always be combined: view all "maybe" edges
# that are incident on alter 2833.
E(gr)[weight==2 & inc("2833")]
## + 5/259 edges from 4a3bad2 (vertex names):
## [1] 2814--2833 2815--2833 2821--2833 2825--2833 2833--2843

# Adding the ego node to an ego-network ----
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Create ego's star network with all alters
ego.star <- make_star(n=46, mode="undirected")
plot(ego.star)
```



```
# Check out the current vertex sequences and names of the two graphs
V(ego.star)
```

```
## + 46/46 vertices, from 5ba3cf2:
## [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] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
V(ego.star)$name
## NULL
V(gr)$name
## [1] "2801" "2802" "2803" "2804" "2805" "2806" "2807" "2808" "2809" "2810"
## [11] "2811" "2812" "2813" "2814" "2815" "2816" "2817" "2818" "2819" "2820"
## [21] "2821" "2822" "2823" "2824" "2825" "2826" "2827" "2828" "2829" "2830"
## [31] "2831" "2832" "2833" "2834" "2835" "2836" "2837" "2838" "2839" "2840"
## [41] "2841" "2842" "2843" "2844" "2845"

# Add "name" vertex attribute and "weight" edge attribute to ego star network
V(ego.star)$name <- c("ego", V(gr)$name)
E(ego.star)$weight <- 1

# Join the original ego-network with the ego star
gr.ego <- gr %>%
as_tbl_graph(gr) %>% # Temporarily convert to tidygraph to perform the join
graph_join(ego.star) # Join gr with ego star
## Joining, by = "name"

# Convert back to igraph and make undirected
gr.ego <- as.igraph(gr.ego) %>%
as.undirected(mode="collapse")

# Look at the ego-network now
set.seed(607)
ggraph(gr.ego) +
geom_edge_link() +
geom_node_label(aes(label= name)) +
theme_graph(base_family = 'Helvetica')
## Using `nicely` as default layout
```




```
# ***** EXERCISE (1):
# Get the average alter closeness (vertex attribute $alter.clo) of Sri Lankan
# alters in the personal network gr. Use mean().
```

```
# *****
# ***** EXERCISE (2):
# How many edges in the personal network gr involve at least one close family
# member and are "probable"? Remember that "probable" edges (the two alters
# "maybe" know each other) have $weight==2 and close family members have
# $relation=="Close family". Use inc().
# *****
```

MEDIDAS DE COMPOSICIÓN DE LA RED EGOCÉNTRICA

La composición de la red egocéntrica se refiere a la distribución de los atributos de los alteri en la red egocéntrica. Se pueden calcular muchas medidas diferentes en R para describir la composición de la red egocéntrica. El resultado suele ser una variable de resumen a nivel del ego: esta variable asigna un número a cada ego, y ese número describe un atributo de la composición de la red egocéntrica para cada ego.

Hay que tener en cuenta que para calcular las medidas de composición de la red egocéntrica no necesitamos ningún dato de red (lazos alter-alter), sino solo el conjunto de datos de nivel de alter con atributos alternos (por ejemplo, edad, sexo, etnia, frecuencia de contacto, etc. del alter). En otras palabras, una lista de alteri con sus atributos (y sin información sobre los lazos entre ellos) es suficiente para examinar la composición de la red egocéntrica.

En esta sección, la red egocéntrica está representada simplemente por el *dataframe* de nivel alter y no necesitamos trabajar con el objeto de red "igraph".

Existen al menos tres tipos de medidas de composición sumarias para las redes egocéntricas:

- Medidas basadas en un atributo de los alteri. Por ejemplo, la edad media de la red.
- Medidas basadas en múltiples atributos de los alteri. Por ejemplo, la frecuencia promedio de contacto (atributo 1) entre el ego y los son miembros de la familia (atributo 2).
- Medidas de homofilia ego-alter. Estas son medidas resumidas de la medida en que los alteri son similares al ego con respecto a uno o más atributos. Por ejemplo, la proporción de personas que son del mismo sexo (etnia, grupo de edad) que el ego.

Las funciones del código que se presenta a continuación son las siguientes:

- Considerar el marco de datos de atributos de los alteri para un ego.
- Usar este marco de datos, calcular medidas de composición basadas en un atributo de los alteri.
- Calcular medidas de composición basadas en dos atributos de los alteri.
- Hacer la unión de nivel 1: unir los atributos del ego con datos de nivel de los alteri.
- Usar los datos unidos, calcular las medidas de composición de la homofilia entre el ego y los alteri.
- Calcular múltiples medidas de composición y ponerlas juntas en un marco de datos a nivel del ego.

Repetir el mismo cálculo de una variable de composición para todas las redes egocéntricas en los datos a la vez.

```
# For compositional measures all we need is the alter attribute data frame.
# This is that data frame for one ego (28).
alter.attr.28
## # A tibble: 45 x 11
## alter_ID ego_ID alter_num alter.sex alter.age.cat alter.rel alter.nat
## <dbl> <dbl> <dbl> <fct> <dbl> <fct> <fct>
## 1 2801 28 1 Female 6 Close fa~ Sri Lanka
## 2 2802 28 2 Male 6 Other fa~ Sri Lanka
## 3 2803 28 3 Male 6 Close fa~ Sri Lanka
## 4 2804 28 4 Male 7 Close fa~ Sri Lanka
## 5 2805 28 5 Female 5 Close fa~ Sri Lanka
## 6 2806 28 6 Female 7 Close fa~ Sri Lanka
```

```
## 7 2807 28 7 Male 5 Other fa~ Sri Lanka
## 8 2808 28 8 Female 4 Other fa~ Sri Lanka
## 9 2809 28 9 Female 6 Other fa~ Sri Lanka
## 10 2810 28 10 Male 7 Other fa~ Sri Lanka
## # ... with 35 more rows, and 4 more variables: alter.res <fct>,
## # alter.clo <dbl>, alter.loan <fct>, alter.fam <fct>

# Compositional measures based on a single alter attribute ----
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Summary of continuous variable: Average alter closeness.
# Check out the relevant variable
alter.attr.28$alter.clo

## [1] NA 3 NA NA NA NA 5 4 5 5 5 4 3 5 3 3 1 5 4 5 5 5 5
## [24] 5 2 3 4 5 5 5 5 3 5 3 4 5 5 5 5 3 3 3 3 3 5

# Battery of descriptive stats.
summarytools::descr(alter.attr.28$alter.clo)
## Descriptive Statistics
## alter.attr.28$alter.clo
## N: 45
##
## alter.clo
## ----------------- -----------
## Mean 4.10
## Std.Dev 1.08
## Min 1.00
## Q1 3.00
## Median 5.00
## Q3 5.00
## Max 5.00
## MAD 0.00
## IQR 2.00
## CV 0.26
## Skewness -0.79
## SE.Skewness 0.37
## Kurtosis -0.38
## N.Valid 40.00
## Pct.Valid 88.89
# Get a single summary measure (useful when writing functions)
mean(alter.attr.28$alter.clo, na.rm = TRUE)
## [1] 4.1

# Summary of categorical variable: Proportion female alters.
# Check out the relevant variable.
alter.attr.28$alter.sex
## [1] Female Male Male Male Female Female Male Female Female Male
## [11] Male Male Male Male Male Male Male Male Male Male
## [21] Male Male Male Male Male Male Male Female Male Male
## [31] Male Female Male Male Male Male Female Male Male Male
## [41] Male Male Male Male Male
## Levels: Male Female

# Get frequencies
freq(alter.attr.28$alter.sex)
## Frequencies
## alter.attr.28$alter.sex
## Type: Factor
##
## Freq % Valid % Valid Cum. % Total % Total Cum.
## ------------ ------ --------- -------------- --------- --------------
## Male 37 82.22 82.22 82.22 82.22
```

```
## Female 8 17.78 100.00 17.78 100.00
## <NA> 0 0.00 100.00
## Total 45 100.00 100.00 100.00 100.00

# Same for nationalities
freq(alter.attr.28$alter.nat)
## Frequencies
## alter.attr.28$alter.nat
## Type: Factor
##
## Freq % Valid % Valid Cum. % Total % Total Cum.
## --------------- ------ --------- -------------- --------- --------------
## Italy 0 0.00 0.00 0.00 0.00
## Other 2 4.44 4.44 4.44 4.44
## Sri Lanka 43 95.56 100.00 95.56 100.00
## <NA> 0 0.00 100.00
## Total 45 100.00 100.00 100.00 100.00

# Another way to get the proportion of a specific category (this is useful when
# writing functions).
mean(alter.attr.28$alter.sex == "Female")
## [1] 0.1777778
mean(alter.attr.28$alter.nat == "Sri Lanka")
## [1] 0.9555556

# The function dplyr::summarise() allows us to calculate multiple measures, name
# them, and put them all together in a data frame.
alter.attr.28 %>%
summarise(
mean.clo = mean(alter.clo, na.rm=TRUE),
prop.fem = mean(alter.sex=="Female"),
count.nat.slk = sum(alter.nat=="Sri Lanka"),
count.nat.ita = sum(alter.nat=="Italy"),
count.nat.oth = sum(alter.nat=="Other")
)
## # A tibble: 1 x 5
## mean.clo prop.fem count.nat.slk count.nat.ita count.nat.oth
## <dbl> <dbl> <int> <int> <int>
## 1 4.1 0.178 43 0 2

# What if we want to calculate the same measures for all ego-networks in the data?
# We'll have to use the data frame with all alter attributes from all egos.
alter.attr.all
## # A tibble: 4,590 x 11
## alter_ID ego_ID alter_num alter.sex alter.age.cat alter.rel alter.nat
## <dbl> <dbl> <dbl> <fct> <dbl> <fct> <fct>
## 1 2801 28 1 Female 6 Close fa~ Sri Lanka
## 2 2802 28 2 Male 6 Other fa~ Sri Lanka
## 3 2803 28 3 Male 6 Close fa~ Sri Lanka
## 4 2804 28 4 Male 7 Close fa~ Sri Lanka
## 5 2805 28 5 Female 5 Close fa~ Sri Lanka
## 6 2806 28 6 Female 7 Close fa~ Sri Lanka
## 7 2807 28 7 Male 5 Other fa~ Sri Lanka
## 8 2808 28 8 Female 4 Other fa~ Sri Lanka
## 9 2809 28 9 Female 6 Other fa~ Sri Lanka
## 10 2810 28 10 Male 7 Other fa~ Sri Lanka
## # ... with 4,580 more rows, and 4 more variables: alter.res <fct>,
## # alter.clo <dbl>, alter.loan <fct>, alter.fam <fct>

# dplyr allows us to "group" a data frame by a factor (here, ego IDs) so all
# measures we run on that data frame (means, proportions, etc.) are calculated
```

```
# by the groups given by that factor (here, for each ego ID).
alter.attr.all %>%
group_by(ego_ID) %>%
summarise(
mean.clo = mean(alter.clo, na.rm=TRUE),
prop.fem = mean(alter.sex=="Female"),
count.nat.slk = sum(alter.nat=="Sri Lanka"),
count.nat.ita = sum(alter.nat=="Italy"),
count.nat.oth = sum(alter.nat=="Other")
)
## # A tibble: 102 x 6
## ego_ID mean.clo prop.fem count.nat.slk count.nat.ita count.nat.oth
## <dbl> <dbl> <dbl> <int> <int> <int>
## 1 28 4.1 0.178 43 0 2
## 2 29 4.03 0.0889 44 1 0
## 3 33 3.62 0.378 32 2 11
## 4 35 3.78 0.289 33 4 8
## 5 39 3.73 0.244 39 5 1
## 6 40 3.32 0.356 34 1 10
## 7 45 4.02 0.244 14 19 12
## 8 46 3.48 0.4 33 7 5
## 9 47 4.05 0.267 45 0 0
## 10 48 4.07 0.311 39 4 2
## # ... with 92 more rows

# We'll talk more about this and show more examples in the next script (R03).
# Compositional measures based on multiple alter attributes ----
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Using indexing we can combine multiple alter attribute variables.
# Mean closeness of alters who are "Friends".
# Check out the relevant vector.
alter.attr.28$alter.clo[alter.attr.28$alter.rel=="Friends"]
## [1] 5 4 3 5 3 3 5 5 5 5 5 4 5 5 5 5 5 4 5 5 5 5 3 3 3 5

# Get its mean.
alter.attr.28$alter.clo[alter.attr.28$alter.rel=="Friends"] %>% mean
## [1] 4.444444

# Mean closeness of alters who are "Acquaintances".
alter.attr.28$alter.clo[alter.attr.28$alter.rel=="Acquaintances"] %>% mean
## [1] 2.75

# Count of close family members who live in Sri Lanka vs those who live in Italy.
# In Sri Lanka.
sum(alter.attr.28$alter.rel == "Close family" & alter.attr.28$alter.res == "Sri Lanka")
## [1] 5

# In Italy.
sum(alter.attr.28$alter.rel == "Close family" & alter.attr.28$alter.res == "Italy")
## [1] 0

# Again, we can put all these measures together into a data frame row with dplyr.
alter.attr.28 %>%
summarise(
mean.clo.fr = mean(alter.clo[alter.rel=="Friends"]),
mean.clo.acq = mean(alter.clo[alter.rel=="Acquaintances"]),
count.fam.slk = sum(alter.rel=="Close family" & alter.res=="Sri Lanka"),
count.fam.ita = sum(alter.rel=="Close family" & alter.res=="Italy")
)
## # A tibble: 1 x 4
## mean.clo.fr mean.clo.acq count.fam.slk count.fam.ita
## <dbl> <dbl> <int> <int>
```

```
## 1 4.44 2.75 5 0

# Compositional measures of homophily between ego and alters ----
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Level-1 join: Bring ego-level data into alter-level data frame for ego 28.
(data.28 <- left_join(alter.attr.28, ego.df, by= "ego_ID"))
## # A tibble: 45 x 19
## alter_ID ego_ID alter_num alter.sex alter.age.cat alter.rel alter.nat
## <dbl> <dbl> <dbl> <fct> <dbl> <fct> <fct>
## 1 2801 28 1 Female 6 Close fa~ Sri Lanka
## 2 2802 28 2 Male 6 Other fa~ Sri Lanka
## 3 2803 28 3 Male 6 Close fa~ Sri Lanka
## 4 2804 28 4 Male 7 Close fa~ Sri Lanka
## 5 2805 28 5 Female 5 Close fa~ Sri Lanka
## 6 2806 28 6 Female 7 Close fa~ Sri Lanka
## 7 2807 28 7 Male 5 Other fa~ Sri Lanka
## 8 2808 28 8 Female 4 Other fa~ Sri Lanka
## 9 2809 28 9 Female 6 Other fa~ Sri Lanka
## 10 2810 28 10 Male 7 Other fa~ Sri Lanka
## # ... with 35 more rows, and 12 more variables: alter.res <fct>,
## # alter.clo <dbl>, alter.loan <fct>, alter.fam <fct>, ego.sex <fct>,
## # ego.age <dbl>, ego.arr <dbl>, ego.edu <fct>, ego.inc <dbl>,
## # empl <dbl>, ego.empl.bin <fct>, ego.age.cat <dbl>

# Note the left join: We only retain rows in the left data frame (i.e., alters
# of ego 28), and discard all egos in the right data frame that do not
# correspond to those rows (alters).
# Example: Proportion of alters of the same sex as ego.
data.28 %>%
dplyr::select(alter_ID, ego_ID, alter.sex, ego.sex)

## # A tibble: 45 x 4
## alter_ID ego_ID alter.sex ego.sex
## <dbl> <dbl> <fct> <fct>
## 1 2801 28 Female Male
## 2 2802 28 Male Male
## 3 2803 28 Male Male
## 4 2804 28 Male Male
## 5 2805 28 Female Male
## 6 2806 28 Female Male
## 7 2807 28 Male Male
## 8 2808 28 Female Male
## 9 2809 28 Female Male
## 10 2810 28 Male Male
## # ... with 35 more rows

# First create a vector that is TRUE whenever alter has the same sex as ego in
# data.28 (the joined data frame for ego 28).
data.28$alter.sex == data.28$ego.sex
## [1] FALSE TRUE TRUE TRUE FALSE FALSE TRUE FALSE FALSE TRUE TRUE
## [12] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [23] TRUE TRUE TRUE TRUE TRUE FALSE TRUE TRUE TRUE FALSE TRUE
## [34] TRUE TRUE TRUE FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [45] TRUE

# The proportion we're looking for is simply the proportion of TRUE's in this
# vector.
mean(data.28$alter.sex == data.28$ego.sex)
## [1] 0.8222222
```

MEDIDAS DE ESTRUCTURA DE LA RED EGOCÉNTRICA

La estructura de la red egocéntrica se refiere a la distribución de lazos entre los alteri. La estructura de la red egocéntrica puede describirse utilizando diferentes medidas, tanto a nivel de alter (por ejemplo, medidas de centralidad de alter) como a nivel de ego (por ejemplo, densidad de la red egocéntrica).

Algunas medidas estructurales se calculan en la red alter-alter *excluyendo* al ego. Otras medidas (por ejemplo, el grado de intermediación o restricción del ego) se calculan en la red egocéntrica *incluyendo* al ego.

Ciertas medidas son una combinación de estructura y composición: se basan tanto en la distribución de lazos de los alteri como en la distribución de atributos de los alteri. Un ejemplo es el grado medio de centralidad (estructura de la red) de los alteri que son miembros de la familia (composición de la red).

Hay que tener en cuenta que cada vez que se considera la estructura de la red egocéntrica (ya sea por sí misma o en combinación con la composición), necesitamos datos sobre los lazos alter-alter. Solo el *dataframe* con los atributos de los alteri no será suficiente.

En esta sección, debemos trabajar con el objeto "igraph" que representa la red egocéntrica (posiblemente incorporando los atributos de los alteri si nuestras medidas son una combinación de estructura y composición).

Funciones del código que se presenta a continuación.

- Considerar ejemplos de medidas estructurales de la red egocéntrica solo en base a la distribución de lo lazos alter-alter, excluyendo al ego: densidad, número de componentes, grado medio de alteri, intervalo máximo de alteri, número de aislamientos. Se calculan utilizando las funciones "igraph".
- Considerar ejemplos de medidas estructurales que requieren que se incluya el ego en el objeto "igraph", y calcularlas: grado de intermediación o restricción del ego.
- Considerar ejemplos de medidas que combinan la estructura y composición de la red egocéntrica, y calcularlas utilizando el objeto "igraph": tipo de relación entre el ego y el alter más central; grado medio de centralidad de los alteri que son familiares cercanos; densidad de lazos entre alteri en ciertas categorías (por ejemplo, alteri que viven en Sri Lanka).
- Repetir el mismo cálculo de una variable estructural para todas las redes egocéntricas en los datos a la vez.

```
# For structural measures we need the ego network as a graph.
gr
## IGRAPH 4a3bad2 UNW- 45 259 --
## + attr: ego_ID (g/n), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), new.attribute (v/n), weight (e/n)
## + edges from 4a3bad2 (vertex names):
## [1] 2801--2802 2801--2803 2801--2804 2801--2805 2801--2806 2801--2807
## [7] 2801--2808 2801--2809 2801--2810 2801--2811 2801--2812 2801--2813
## [13] 2801--2814 2801--2815 2801--2818 2801--2820 2801--2823 2801--2825
## [19] 2801--2827 2801--2828 2801--2829 2801--2831 2801--2840 2801--2841
## [25] 2802--2803 2802--2804 2802--2805 2802--2806 2802--2807 2802--2808
## + ... omitted several edges

# Measures based only on the structure of alter-alter ties ----
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# Structural characteristics of the network.
# Network density.
edge_density(gr)
## [1] 0.2616162

# Number of components.
components(gr)
## $membership
## 2801 2802 2803 2804 2805 2806 2807 2808 2809 2810 2811 2812 2813 2814 2815
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
## 2816 2817 2818 2819 2820 2821 2822 2823 2824 2825 2826 2827 2828 2829 2830
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## 2831 2832 2833 2834 2835 2836 2837 2838 2839 2840 2841 2842 2843 2844 2845
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## $csize
## [1] 45
##
## $no
## [1] 1

# This is a list, we only need the 3rd element
components(gr)$no
## [1] 1

# Summarization of structural characteristics of the alters.
# Average degree.
degree(gr) %>% mean

## [1] 11.51111
# Max betweenness.
betweenness(gr) %>% max
## [1] 265.3514
# Number of "isolate" alters.
# Check out the alter degree vector.
degree(gr)
## 2801 2802 2803 2804 2805 2806 2807 2808 2809 2810 2811 2812 2813 2814 2815
## 24 17 13 12 12 13 13 12 10 9 11 8 18 14 16
## 2816 2817 2818 2819 2820 2821 2822 2823 2824 2825 2826 2827 2828 2829 2830
## 9 9 10 2 2 9 18 27 13 10 8 4 3 3 3
## 2831 2832 2833 2834 2835 2836 2837 2838 2839 2840 2841 2842 2843 2844 2845
## 15 16 16 12 21 3 10 14 14 16 13 7 12 5 12

# Count the number of isolates, i.e. alters with degree==0
sum(degree(gr)==0)
## [1] 0

# All sorts of more complicated structural analysis can be run on an ego-network
# using igraph and statnet functions. For example, there are the results of the
# Girvan-Newman community-detection algorithm on the ego-network.
cluster_edge_betweenness(gr, weights= NULL)
## IGRAPH clustering edge betweenness, groups: 10, mod: 0.41
## + groups:
## $`1`
## [1] "2801" "2802" "2803" "2804" "2805" "2806" "2807" "2808" "2809"
## [10] "2810" "2811" "2812" "2840" "2841"
##
## $`2`
## [1] "2813" "2814" "2815" "2816" "2817" "2818" "2824" "2831"
##
## $`3`
## [1] "2819" "2836"
##
## + ... omitted several groups/vertices

# What if we want to calculate the same structural measure (e.g. density) on all
# ego-networks? Again that's easy to do with tidyverse, using the purrr package.
# We'll have to use the list that contains all our ego-networks.
class(gr.list)
## [1] "list"
length(gr.list)
## [1] 102
gr.list %>% head
```

```
## $`28`
## IGRAPH 4a3bad2 UNW- 45 259 –
## + attr: ego_ID (g/c), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), weight (e/n)
## + edges from 4a3bad2 (vertex names):
## [1] 2801--2802 2801--2803 2801--2804 2801--2805 2801--2806 2801--2807
## [7] 2801--2808 2801--2809 2801--2810 2801--2811 2801--2812 2801--2813
## [13] 2801--2814 2801--2815 2801--2818 2801--2820 2801--2823 2801--2825
## [19] 2801--2827 2801--2828 2801--2829 2801--2831 2801--2840 2801--2841
## [25] 2802--2803 2802--2804 2802--2805 2802--2806 2802--2807 2802--2808
## + ... omitted several edges
##
## $`29`
## IGRAPH 4eb46cf UNW- 45 202 --
## + attr: ego_ID (g/c), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), weight (e/n)
## + edges from 4eb46cf (vertex names):
## [1] 2901--2902 2901--2904 2901--2906 2901--2907 2901--2908 2901--2909
## [7] 2901--2910 2901--2911 2901--2915 2901--2916 2901--2917 2901--2918
## [13] 2901--2919 2901--2927 2901--2928 2901--2929 2901--2930 2901--2931
## [19] 2901--2936 2901--2942 2901--2945 2902--2903 2902--2904 2902--2905
## [25] 2902--2906 2902--2907 2902--2908 2902--2909 2902--2910 2902--2911
## + ... omitted several edges
##
## $`33`
## IGRAPH 2ba8412 UNW- 45 207 --
## + attr: ego_ID (g/c), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), weight (e/n)
## + edges from 2ba8412 (vertex names):
## [1] 3301--3302 3301--3303 3301--3304 3301--3305 3301--3306 3301--3307
## [7] 3301--3308 3301--3309 3301--3310 3301--3311 3301--3312 3301--3313
## [13] 3301--3314 3301--3315 3301--3316 3301--3317 3301--3318 3301--3319
## [19] 3301--3320 3301--3321 3301--3322 3301--3323 3302--3303 3302--3304
## [25] 3302--3305 3302--3306 3302--3307 3302--3308 3302--3309 3302--3310
## + ... omitted several edges
##
## $`35`
## IGRAPH dc06518 UNW- 45 221 --
## + attr: ego_ID (g/c), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), weight (e/n)
## + edges from dc06518 (vertex names):
## [1] 3501--3502 3501--3503 3501--3504 3501--3505 3501--3506 3501--3507
## [7] 3501--3508 3501--3509 3501--3510 3501--3512 3501--3513 3501--3514
## [13] 3501--3516 3501--3517 3501--3522 3501--3523 3501--3524 3501--3525
## [19] 3501--3526 3501--3527 3501--3528 3501--3529 3501--3530 3501--3532
## [25] 3501--3533 3501--3534 3501--3535 3501--3536 3501--3540 3501--3543
## + ... omitted several edges
##
## $`39`
## IGRAPH 9eec913 UNW- 45 92 --
## + attr: ego_ID (g/c), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), weight (e/n)
```

```
## + edges from 9eec913 (vertex names):
## [1] 3901--3902 3901--3903 3901--3904 3901--3905 3901--3906 3901--3907
## [7] 3901--3911 3901--3913 3901--3920 3901--3924 3901--3925 3901--3931
## [13] 3901--3932 3901--3933 3901--3937 3901--3938 3901--3940 3901--3941
## [19] 3902--3903 3902--3904 3902--3905 3902--3906 3902--3907 3902--3908
## [25] 3902--3909 3902--3910 3902--3913 3902--3924 3902--3925 3902--3931
## + ... omitted several edges
##
## $`40`
## IGRAPH fd64325 UNW- 45 255 --
## + attr: ego_ID (g/c), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), weight (e/n)
## + edges from fd64325 (vertex names):
## [1] 4001--4003 4001--4007 4001--4008 4001--4009 4001--4036 4001--4040
## [7] 4001--4045 4002--4003 4002--4004 4002--4006 4002--4008 4002--4009
## [13] 4002--4010 4002--4011 4002--4012 4002--4013 4002--4014 4002--4015
## [19] 4002--4016 4002--4017 4002--4021 4002--4029 4002--4036 4002--4037
## [25] 4002--4038 4002--4040 4002--4041 4002--4042 4002--4043 4002--4044
## + ... omitted several edges

# We can use purrr::map() to run the same function on each element of the list.
purrr::map_dbl(gr.list, ~ edge_density(.x))
## 28 29 33 35 39 40
## 0.26161616 0.20404040 0.20909091 0.22323232 0.09292929 0.25757576
## 45 46 47 48 49 51
## 0.18484848 0.26969697 0.53737374 0.20303030 0.42828283 0.16969697
## 52 53 55 56 57 58
## 0.12929293 0.16565657 0.23838384 0.22828283 0.60000000 0.20505051
## 59 60 61 62 64 65
## 0.50505051 0.28888889 0.37676768 0.30202020 0.28282828 0.36060606
## 66 68 69 71 73 74
## 0.30909091 0.23333333 0.23636364 0.47777778 0.40707071 0.42525253
## 78 79 80 81 82 83
## 0.17575758 0.29292929 0.33434343 0.29797980 0.35353535 0.38888889
## 84 85 86 87 88 90
## 0.27575758 0.12525253 0.18989899 0.24444444 0.40101010 0.45353535
## 91 92 93 94 95 97
## 0.47171717 0.43535354 0.14141414 0.26767677 0.36767677 0.35050505
## 99 102 104 105 107 108
## 0.24040404 0.27777778 0.22929293 0.26666667 0.16262626 0.10606061
## 109 110 112 113 114 115
## 0.33535354 0.48686869 0.13939394 0.23636364 0.28787879 0.33232323
## 116 118 119 120 121 122
## 0.22929293 0.37676768 0.33131313 0.20101010 0.23838384 0.35959596
## 123 124 125 126 127 128
## 0.39090909 0.61111111 0.26464646 0.72828283 0.28080808 0.26262626

## 129 130 131 132 133 135
## 0.22727273 0.22020202 0.27979798 0.29090909 0.27575758 0.22525253
## 136 138 139 140 141 142
## 0.28686869 0.19797980 0.31515152 0.29292929 0.26868687 0.21010101
## 144 146 147 149 151 152
## 0.24949495 0.14747475 0.23030303 0.20707071 0.27272727 0.25151515
## 153 154 155 156 157 158
## 0.34242424 0.31818182 0.39393939 0.24646465 0.37777778 0.23737374
## 159 160 161 162 163 164
## 0.37979798 0.36767677 0.41010101 0.41111111 0.35454545 0.32222222

# We'll talk more about this and show more examples in the next script (R03).
# Structural measures requiring ego in the network ----
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
# We now need the ego network with ego included
gr.ego
## IGRAPH b2203d4 UNW- 46 304 --
## + attr: ego_ID (g/n), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), new.attribute (v/n), weight (e/n)
## + edges from b2203d4 (vertex names):
## [1] 2801--2802 2801--2803 2802--2803 2801--2804 2802--2804 2803--2804
## [7] 2801--2805 2802--2805 2803--2805 2804--2805 2801--2806 2802--2806
## [13] 2803--2806 2804--2806 2805--2806 2801--2807 2802--2807 2803--2807
## [19] 2804--2807 2805--2807 2806--2807 2801--2808 2802--2808 2803--2808
## [25] 2804--2808 2805--2808 2806--2808 2807--2808 2801--2809 2802--2809
## + ... omitted several edges

# Ego's betweenness centrality.
betweenness(gr.ego, v= "ego")
## ego
## 486.9722

# Ego's constraint.
constraint(gr.ego, nodes= "ego")
## ego
## 0.08635842

# Measures combining composition and structure: From structure to composition ----
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# From stucture to composition: Non-network attribute of alters selected based
# on structural characteristics.
# Type of relationship with alter with max betweenness.
# Logical index for alter with max betweenness.
ind <- betweenness(gr) == max(betweenness(gr))
# Get that alter.
V(gr)[ind]

## + 1/45 vertex, named, from 4a3bad2:
## [1] 2801

# Get type of relation of that alter.
V(gr)[ind]$alter.rel
## [1] "Close family"

# Note that there might be multiple alters with the same (maximum) value of
# betweenness. For that case, we'll need more complicated code (see exercise).
# Measures combining composition and structure: From composition to structure ----
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
# From composition to structure: Structural characteristics of alters selected
# based on composition.
# Avg degree of Close family.
# Vertex sequence of Close family members.
(clo.fam.vs <- V(gr)[alter.rel=="Close family"])
## + 5/45 vertices, named, from 4a3bad2:
## [1] 2801 2803 2804 2805 2806

# Get their avg degree.
degree(gr, v=clo.fam.vs) %>% mean
## [1] 14.8

# Count of ties between alters who live in Sri Lanka.
# First get the vertex sequence of alters who live in Sri Lanka.
(alters.sl <- V(gr)[alter.res=="Sri Lanka"])
## + 17/45 vertices, named, from 4a3bad2:
## [1] 2801 2802 2803 2804 2805 2806 2807 2808 2809 2810 2811 2812 2827 2828
```

```
## [15] 2829 2840 2841

# Then get the edges among them.
E(gr)[alters.sl %--% alters.sl]
## + 88/259 edges from 4a3bad2 (vertex names):
## [1] 2801--2802 2801--2803 2801--2804 2801--2805 2801--2806 2801--2807
## [7] 2801--2808 2801--2809 2801--2810 2801--2811 2801--2812 2801--2827
## [13] 2801--2828 2801--2829 2801--2840 2801--2841 2802--2803 2802--2804
## [19] 2802--2805 2802--2806 2802--2807 2802--2808 2802--2809 2802--2810
## [25] 2802--2811 2802--2812 2802--2840 2802--2841 2803--2804 2803--2805
## [31] 2803--2806 2803--2807 2803--2808 2803--2809 2803--2810 2803--2811
## [37] 2803--2812 2803--2840 2803--2841 2804--2805 2804--2806 2804--2807
## [43] 2804--2808 2804--2809 2804--2810 2804--2811 2804--2840 2804--2841
## [49] 2805--2806 2805--2807 2805--2808 2805--2809 2805--2810 2805--2811
## [55] 2805--2840 2805--2841 2806--2807 2806--2808 2806--2809 2806--2810
## + ... omitted several edges

# How many edges are there between alters who live in Sri Lanka?
E(gr)[alters.sl %--% alters.sl] %>% length
## [1] 88

# Density between alters who live in Sri Lanka.
# Get subgraph of relevant alters
induced_subgraph(gr, vids = alters.sl)
## IGRAPH 73da802 UNW- 17 88 --
## + attr: ego_ID (g/n), name (v/c), alter_num (v/n), alter.sex
## | (v/c), alter.age.cat (v/n), alter.rel (v/c), alter.nat (v/c),
## | alter.res (v/c), alter.clo (v/n), alter.loan (v/c), alter.fam
## | (v/c), new.attribute (v/n), weight (e/n)
## + edges from 73da802 (vertex names):
## [1] 2801--2802 2801--2803 2802--2803 2801--2804 2802--2804 2803--2804
## [7] 2801--2805 2802--2805 2803--2805 2804--2805 2801--2806 2802--2806
## [13] 2803--2806 2804--2806 2805--2806 2801--2807 2802--2807 2803--2807
## [19] 2804--2807 2805--2807 2806--2807 2801--2808 2802--2808 2803--2808
## [25] 2804--2808 2805--2808 2806--2808 2807--2808 2801--2809 2802--2809
## + ... omitted several edges

# Get the density of this subgraph.
induced_subgraph(gr, vids = alters.sl) %>% edge_density
## [1] 0.6470588
```

REFERENCIAS

McCarty, C., Lubbers, M. J., Vacca, R., & Molina, J. L. (2019). *Conducting Personal Network Research. A Practical Guide*. Nueva York: Guilford Press.

Remitido: 25-05-2020

Aceptado: 12-06-2020