REDES Revista Hispana para el Análisis de Redes Sociales

Vol.33,#1, (2022), 100-111

http://revistes.uab.es/redes https://doi.org/10.5565/rev/redes.937

Modelos estocásticos orientados en el actor utilizando RSiena (II). Formato de los Datos.

Alejandro Espinosa Rada[1]

ETH Zürich / The Mitchell Centre for Social Network Analysis, The University of Manchester

RESUMEN

Este texto es el segundo de una serie de seis guiones escritos en el lenguaje y ambiente de programación R que conjuntamente constituyen la introducción al software RSiena para estimar modelos estocásticos orientados en el actor para redes dinámicas. El texto está acompañado por los guiones para reproducir y estimar los modelos.

Palabras clave: *Modelos estocásticos orientados en el actor – Análisis estadísticos de redes sociales – RSiena - R.*

ABSTRACT

This article is the first document of six scripts written in the language and environment R, which introduces the RSiena software to estimate the stochastic actor-oriented models for dynamic networks. This first script is an example of commands for estimating a network and behavioural dependent variables incorporating the main functions, instructions, and exercises. The text incorporates the scripts into the document to reproduce and estimate the models.

Key words: *Stochastic actor-oriented model – Statistical analysis for social networks – RSiena – R.*

[1]*Contacto con los autores: Alejandro Espinosa Rada <anespinosa@uc.cl>*

INTRODUCCIÓN

Los materiales para esta serie incluyen 6 guiones (*scripts*) de ***RSiena*** escritos en el entorno y lenguage de programación **R** (2021) enfocado a la estadística computacional y de gráficos. Los guiones originales se encuentran disponibles en la página oficial[1] del programa para el análisis estadístico de datos de redes SIENA (acrónimo de *Simulation Investigation for Empirical Network Analysis*), con énfasis en redes sociales, implementado en el paquete ***RSiena*** y que cuenta con un manual de usuarios (Ripley et al., 2021).

Este guión es la segunda contribución de una serie de seis guiones que introducen el software **RSiena**. Si bien el primer guión (Espinosa-Rada, 2021) permite dar una visión panorámica sobre algunas de las funciones que ofrece esta perspectiva, esta contribución pretende explorar con mayor detención las distintas posibilidades que ofrece este programa tal forma de conectar el software con archivos de datos que poseen variados formatos.

El ambiente R

Este guión provee de explicaciones adicionales para utilizar el software **R**, los cuales pueden ser omitidos para personas que ya estén familiarizadas con este programa.

Primero que todo, antes de comenzar a utilizar **RSiena** se requiere del programa **R**. El cual se puede descargar de forma gratuita en su página web (https://cran.r-project.org). Tal como aparece en la figura 1 deberá descargar el software correspondiente a su sistema operativo.

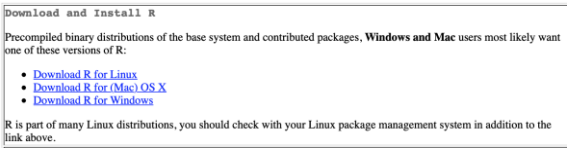

Figura 1. Descargar R

A continuación, aparecerá en su pantalla distintas versiones. En el momento que este tutorial fue escrito, la última versión correspondía a **R-4.04** como aparece en la figura 2.

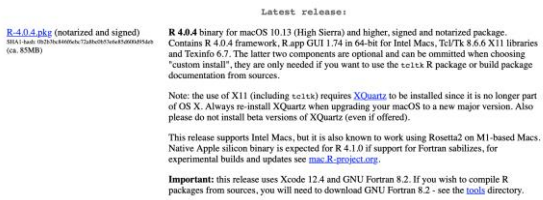

Figura 2. Última versión de R.

Para ejecutar correctamente **RSiena** en algunos casos (particularmente para usuarios que utilizan sistema operativo [Mac] OS X) también tendrá que descargar XQuartz[2].

Una vez descargado **R**, que debiese ser algo similar a la figura 3 (note que mi versión personal es un poco más antigua, dado que corresponde a la versión R 4.0.2), podrá abrir el programa y comenzar a ejecutar los códigos a continuación del símbolo >. En otras palabras, podrá poner un "comando" (ejemplo: 2+2) y luego presionar el comando "Enter" para que éste sea ejecutado. En su caso debiese aparecer el número 4 en su "consola"

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.72 (7847) x86_64-apple-darwin17.0]

[Workspace restored from /Users/anespinosa/.RData]
[History restored from /Users/anespinosa/.Rapp.history]

>
```

Figura 3. Ambiente R

Para personas que no estén familiarizadas con la programación, **R** (al igual que un universo de otros lenguajes de programación) pueden ser algo intimidantes si se utiliza el programa por defecto. Por ello, una alternativa bastante familiar que facilita varios procesos en e proceso de programación, interoperabilidad, mantención de códigos, entre otros, es **RStudio**. Sugerimos que a menos que esté seguro de lo que está haciendo, utilice este "intérprete".

Al igual que **R**, el programa **RStudio** también tene una versión gratuita (**RStudio Desktop**). Para descargar este software tendrá que visitar la página oficial[3] y en el costado superior

[1] Url de la página oficial de RSiena: https://www.stats.ox.ac.uk/~snijders/siena/
[2] Url donde podrá descargar XQuartz: https://www.xquartz.org
[3] Url de la página oficial de RStudio: https://rstudio.com

derecho aparecerá la opción "DOWNLOAD" (tal como aparece en la figura 4)

Figura 4. Descargar RStudio 1

Luego se dirigirá a una página similar a la que aparece en la figura 5. Presione la versión RStudio Desktop (a menos que quiera otra versión, en cuyo caso tendrá que pagar o utilizar en formato online).

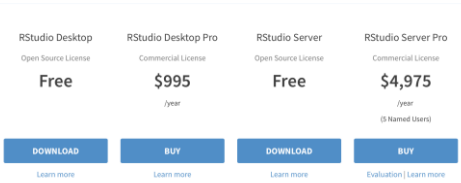

Figura 5. Descargar RStudio 2

Al presionar "DOWNLOAD", la página le redireccionará a una próxima página en donde podrá finalmente descargar el software (similar a lo que aparece en figura 6).

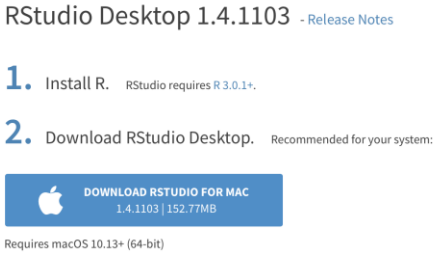

Figura 6. Descargar RStudio 3

Una vez instalado **R** y **RStudio** (considere que tiene que intsalar primero R y luego RStudio dado que RStudio "interpreta" a R). Al abrir **RStudio** observará que el programa tendrá cuatro sub-ventanas tal como aparece en la Figura 7.

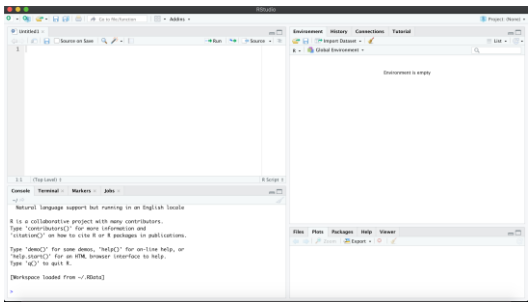

Figura 7. Ambiente de RStudio

En algunos casos puede que por defecto aparezcan tres ventanas en vez de cuatro. Si ése es el caso, diríjase a "File" presione "New File" y luego "R Script" (tal como aparece en la figura 8)

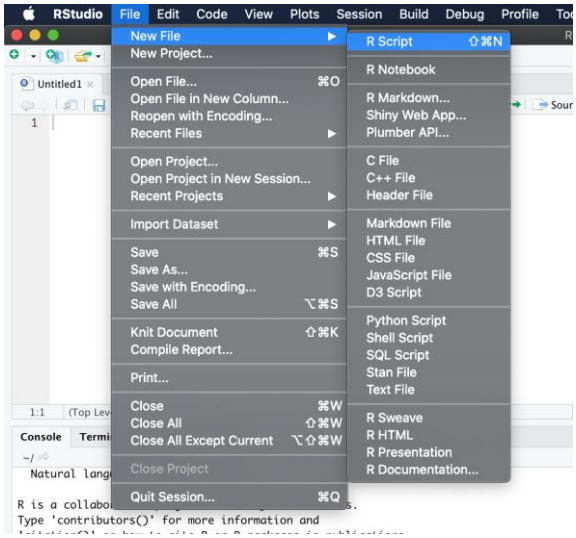

Figura 8. Nuevo Guión

Ahora debiesen aparecer las cuatro sub-ventanas. Revisaremos cada una brevemente. En primer lugar, una de las sub-ventanas (figura 9) es lo que denominaremos "guión" (*script*) en donde podrá escribir todos los códigos que se revisarán a continuación. No obstante, al insertar los códigos estos no serán ejecutados. Para ello tendrá que poner su cursor y presionar el botón izquierdo de su ratón en la línea que quiere ejecutar dentro del guión. Luego podrá presionar "Run" que se encuentra en la barra superior derecha como aparece en la figura 9. Por ejemplo, puede escribir 2+2 en la línea contigua al numeral primero, presionar su ratón en dicha línea y presionar luego "Run", a continuación, debiese aparecer un número 4 en su "consola". Asegúrese de familiarizarse con este procedimiento. Como atajo, en vez de presionar "Run" también puede presionar "Ctrl+Enter" (Microsoft) o "Cmd+Enter" (Mac). Otros atajos pueden encontrarse en la página de RStudio[4].

[4] Url de ayajos de RStudio: https://support.rstudio.com/hc/en-us/articles/200711853

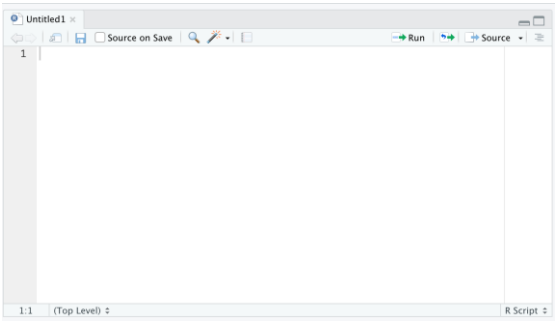

Figura 9. Sub-ventana del Guión

Como consejo de buenas prácticas, es aconsejable escribir en el guión y luego ejecutar en la "consola" (figura 10). Esto dado que el guión suele consistir en la documentación y seguidilla de pasos secuenciales que se han utilizados para obtener los resultados esperados. Además, permite ser una práctica adecuada para compartir y replicar análisis en el futuro.

Figura 10. Sub-ventana Consola

Otra de las sub-ventanas que aparecerán por defecto corresponde al "entorno" de su sesión (figura 11). En dicha sub-ventana aparecerán los distintos "objetos", "funciones", "datos", entre otros que creará durante este tutorial.

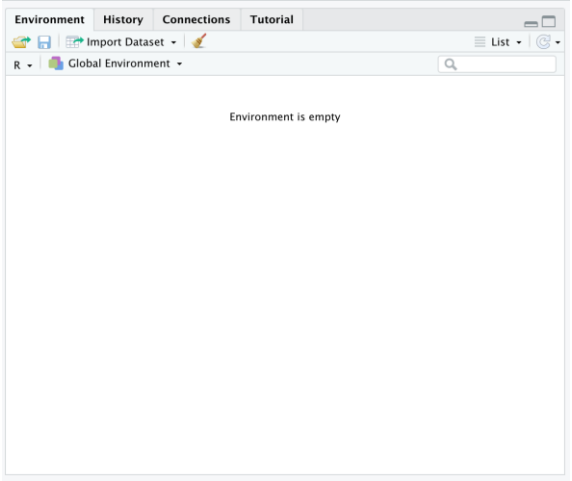

Figura 11. Sub-ventana Entorno

Por último, en la última ventana aparecerán visualizaciones cuando éstas sean solicitadas, al igual que información adicional cuando requiera ayuda sobre funciones particulares, entre otros.

Figura 12. Sub-ventana Visualización

Hasta aquí, hemos descargado **R** y visto algunas de las sub-ventanas del programa **RStudio.** No obstante, el programa tiene muchísimas otras opciones que no cubriremos aquí. Además, en adelante nos referiremos a **R** como el "lenaguaje" que puede ejecutarse indistintamente en la consola entregada por defecto por **R** o a través de **RStudio**, el cual facilita la interoperabilidad de **R**. Esperamos que esta breve introducción sea suficiente para proseguir con el contenido de este tutorial.

Otros puntos por considerar, que pueden ser de utilidad para aquellos que se estén introduciendo a este lenguaje son, en primer lugar, que **R** es sensible a los casos, por lo tanto, hay que tener cuidado sobre las capitalizaciones (ejemplo, no es lo mismo "a" que "A").

Por otro lado, la flecha-a-la-izquierda "<-" es equivalente a "=" (pudiéndose utilizar indistintamente pero que por convención se sugiere la flecha para evitar conflictos con la expresión matemática). Por ello, se utiliza frecuentemente para asignar expresiones dentro de objetos, como ejemplo "a <- b", lo cual en este caso significa que el objeto "a" obtiene el valor de "b". A menudo, es complejo utilizar ciertas expresiones evaluadas en **R**, y por ello es conveniente almacenarlas en un objeto que puede ser denominado arbitrariamente de cualquier forma, tal como "a" en este ejemplo.

Como ejemplo, es posible crear varios objetos los cuales pueden ser almacenados en el entorno **R** adoptándo a su vez distintas "clases". Para familiarizarse con ello puede explorar los siguientes comandos y observar cómo el tipo de elementos son asignados en **R**. Note que en comentarios (expresados en # los cuales no se ejecutarán en el programa) hay algunas notas y preguntas adicionales para ejercitar su comprensión sobre el uso de los comandos. Los comentarios se hacen al final de los comandos después de #, o en las líneas que empiezan con # señalándole a R de ignorar todo aquello que se encuentre más allá de dicho símbolo.

```
# Número

objeto1 <- 1

objeto1

class(objeto1)
```

```
# Transformar a un "número entero" (integer) en vez de su clase original "numérica"
objeto1 <- as.integer(objeto1)

# Número
objeto2 <- 1.54
objeto2
class(objeto2)
as.integer(objeto2) # ¿Es esto correcto?

# Caracter
objeto3 <- "two"
class(objeto3)

# Ecuación numérica
objeto4 <- ((1+3)*5)/3
class(objeto4)

# Vector numérico
objeto5 <- c(1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597)

# Vector de caracteres
objeto6 <- c("a","b","c", "a", "b")
class(objeto6)

# Otra forma de crear objetos
objeto1+objeto5 -> objeto7 # ¿Qué está hacienda esta función?

# Otro vector numérico
objeto8 <- c(1:50)
objeto8

# ¿Qué clase es este vector?
objeto9 <- c("a", 1, "b", 2)

# Transformar un vector de clase carácter a un vector tipo factor
objeto10 <- as.factor(objeto6) # ¿Nota alguna diferencia?
objeto10
```

Para solicitar ayuda en R se puede escribir el signo de pregunta y el nombre de la función en donde se requiera ayuda. Por ejemplo,si ejecuta ?library aparecerá un archivo titulado "cargando/adjuntando y enlistando paquetes" (*Loading/Attaching and Listing of Packages*).

Cualquier comando en **R** es una función entre paréntesis que encierra argumentos, o no encierra nada si no se necesita ningún argumento. Por ejemplo, si ejecuta la función q(), el programa cerrará la sesión actual. Esta función es un diminutivo de "terminar la sesión de R" y en el caso de que quisiera disponer de más información sobre esta función podrá ejecutar ?q().

En general, la sintaxis del comando para llamar a las funciones de R es function(x), donde function es una función guardada y x el nombre del objeto que se va a operar.

El paquete RSiena

El comando library carga los paquetes que se utilizarán durante la sesión. Para este caso, utilizaremos **RSiena** el cual es un conjunto de funciones para estimar los modelos estocásticos basado en los actores. En el primer tutorial de esta serie (Espinosa-Rada, 2021) podrá encontrar los elementos necesarios para instalar el paquete. Para efectos de este tutorial, ejecutando el siguiente comando en su consola debiese ser suficiente para proseguir: install.package(RSiena), luego podrá cargar el paquete:

```
library(RSiena)
```

Algunos paquetes adicionales son utilizados por RSiena, los cuales son denominados *required packages*; estos paquetes adicionales serán cargados automáticamente. En caso de que se requieran paquetes adicionales deberá instalarlos similar a como fue realizado con "RSiena".

Deberá tener todos los paquetes instalados.

```
?install.packages
```

O en RStudio puede hacer clic en la pestaña Packages, Install package(s), luego deberá seleccionar un espejo **CRAN** (ejemplo: Bristol si se encuentra en el Reino Unido). Finalmente, seleccione en la lista el paquete que quiera instalar.

De tal forma de proseguir con el tutorial, uno de los primeros elementos a considerar es con respecto a la interconexión entre **R** y su dispositivo local (ver Espinosa-Rada, 2021 en caso de que no haya podido realizar adecuadamente la conexión). Para ello, puede indagar a través de la consola dónde se

encuentra ejecutando la siguiente línea de código en la consola:

```
getwd()
```

Si posee una ubicación específica en su computadora que quisiera utilizar para cargar y descargar archivos desde **R** a su computadora o viceversa, entonces deberá especificar la ruta en donde se encuentra la carpeta. Para ello, puede especificar la ruta a través de códigos. Como ejemplo: setwd("/Users/nombre_usuario/Documents/Siena/") en una computadora con sistema operativo Mac o algo como setwd("C:\\Users\\nombre_usuario\\Documents\\Siena\\ ") en caso de que sea Microsoft. Recuerde que esta ruta es específica a su computadora, por lo que la ruta señalada anteriormente debe ser ajustada. Ahora puede comprobar si la conexión fue realizada con éxito a través del comando getwd() y resolver si es ésta la carpeta esperada. Si no es el caso, vuela a especificar la ruta utilizando "setwd()" como en el ejemplo anterior.

Si quisiera establecer el directorio de trabajo desde **RStudio** diríjase a la pestaña "Session", luego presione "Set Working Directory" y luego presione "Choose Directory", a continuación podrá seleccionar la ruta de su computadora que le permitirá conectar con la sesión de **R** que está utilizando (en la consola debiese haber aparecido una ruta al interior de setwd())

Ahora podremos proseguir. Para ello, podemos pregunatrnos ¿Qué hay en el directorio de trabajo que hemos seleccionado?

```
list.files()
```

Debiese aparecer una lista de todos los archivos que se encuentran en la carpeta que está utilizando.

También podrá explorar qué es lo que se encuentra disponible en **Rsiena**:

```
?RSiena
```

Esto corresponde a la **PÁGINA DE AYUDA** en formato *.html* que aparecerá en **RStudio** en la ventana correspondiente a las visualizaciones (figura 12). Al final de esta página, si presiona *Index* aparecerá una lista de todas las funciones disponibles en el paquete **RSiena**.

La misma lista se muestra gráficamente en la interfaz de **R** solicitándolo (aledaño al guión en **RStudio** en la sub-vetana del guión de la figura 9).

```
library(help=RSiena)
```

Para más información, el manual de **RSiena** se encuentra disponible en la página oficial de Siena.

Cargar los datos y manipulaciones preliminares

En esta sesión utilizaremos los datos s50 los cuales debiesen estar en el directorio de trabajo que fue anteriormente designado para trabajar en lo que sigue de tutorial. Los datos pueden ser descargados en la página official de SIENA[5] (*50-actor excerpt from the Teenage Friends and Lifestyle Study data set of West et al.: description and data set*). Una vez descargados los datos, asegúrese de guardarlos en la carpeta asignada en setwd() Cerciórese que la base de datos s50 se encuentra en su directorio de trabajo y descomprima los archivos en su directorio de trabajo.

En **R** cada objeto es nombrado de alguna forma reconocible para los usos subsecuentes dentro del guión (por ejemplo, aquí algunos objetos han sido denominados como *friend.data.w1* o *drink*) y luego podemos llamarlos como un objeto dentro de **R**. A continuación se le pedirá a **R** que busque el archivo "s50-network1.dat" (que se encuentra en la carpeta de su computadora conectada con **R** a través del comando "setwd()" como fue visto anteriormente).

Si llama un objeto directamente desde **R**, será considerado como un conjunto de datos, o en la terminología de **R** como *"data frame"*. **R** leerá la información en distintos formatos de datos, el comando para leer los datos, para este caso, será *read.table.*

En el archivo de ayuda para read.table, puede observar la sección "Value", el cual se encuentra en todas las páginas de ayuda: indica la clase del objeto que produce la función. Para read.table, el valor es un conjunto de datos; a continuación podemos ver de qué se trata.

```
?read.table # para tener más informaión

class(read.table("s50-network1.dat"))
```

Esto no es lo que esperamos, por ello convertiremos inmediátamente el conjunto de datos a un formato matriz antecediendo la función por "as.matrix" y guardando el archivo

[5] Url con los datos: http://www.stats.ox.ac.uk/~snijders/siena/siena_datasets.htm

en distintos objetos tal como aparece a continuación:

```
friend.data.w1 <- as.matrix(read.table("s50-network1.dat"))

friend.data.w2 <- as.matrix(read.table("s50-network2.dat"))

friend.data.w3 <- as.matrix(read.table("s50-network3.dat"))

drink <- as.matrix(read.table("s50-alcohol.dat"))

smoke <- as.matrix(read.table("s50-smoke.dat"))
```

Si encambio quisiera leer un archivo *.csv* (que se encuentra en su carpeta) tendremos que usar el comando read.csv.

Internamente la base de datos se encuentra ya contenida en **RSiena** sin tener que descargarlos de la página web. Por lo que el paso anterior puede omitirse y proseguir de la siguiente forma:

```
library(RSiena)

friend.data.w1 <- s501

friend.data.w2 <- s502

friend.data.w3 <- s503

drink <- s50a

smoke <- s50s
```

Nótese que estos dos procedimientos deberán producir cinco objetos (*friend.data.w1*, *friend.data.w2*, *friend.data.w3*, *drink* y *smoke*). Por otro lado, si bien utilizar los objetos internos contenidos en **RSiena** es más sencillo para efectos de este tutorial, la primera opción es usualmente utilizada cuando se trabaja con datos externos (disponiendo, por ejemplo, de matrices en formato *.csv* o *.dat*).

La explicación de la estructura de los datos y el formato se encuentra a continuación.

Distintos formatos de datos

El siguiente ejercicio consiste en importar archivos en distintos formatos y transformar dichos objetos al lenguaje **R** y luego modificarlos al formato de *data.frame*. Por ejemplo, si posee datos "data.dat" en su carpeta local, podrá leer los datos a través de un comando similar al que se encuentra a continuación:

```
data <- read.table("data.dat")
```

Una vez leído el archivo de texto en formato *data.frame* tendrá que asegurarse que la clase del objeto data sea data.frame

```
class(data)

# [1] "data.frame
```

Si sus datos no corresponden a un archivo *.dat*, formas alternativas de importar los datos pueden variar en formas tales como:

```
# ".csv"

data <- read.table("data.csv", header=TRUE, sep=",", row.names="iden")

# ".csv"

data <- read.csv("data.csv",header=TRUE)

# ".spss"

library(foreign)

data <- read.spss("data.spss")

# ".dta"

library(foreign)

data <- read.dta("data.dta")
```

Considere que en este caso los nombres de los archivos se llaman "data" y éstos pueden corresponder a un archivo *.csv* o provenientes de otros programas estadísticos usualmente utilizados en las ciencias sociales tales como SPSS (*.spss*) o STATA (*.dta*). De la misma forma, note que aquí hemos instalado otro paquete llamado *foreign* para poder importar los archivos de estos programas al ambiente **R** (si no dispone del paquete deberá de instalarlo antes de cargarlo utilizando la siguiente función: install.packages("foreign")). Variaciones de estos códigos dependerán de cada caso en particular.

Desde bases de datos a formato matriz

Si no posee alguna base en particular que quisiera explorar, puede proseguir esta sección incluyendo la siguiente función: data <- read.table("s50-network1.dat"). Las bases de datos poseen la forma de casos por variable, en donde las variables son columnas, y éstas ontinen los nombres de las variables tal como se observa a continuación:

```
names(data)
```

Para ver la base de datos puede utilizar el comando fix()

```
fix(data)
```

Tenga en consideración que todos los cambios que se realicen utilizando dicha función se guardarán automáticamente.

A continuación, procederemos con otro ejemplo en donde se crearán dos vectores

```
height <- c( 120, 150, 180, 200, 190, 177, 170, 125, 141, 157 )
```

```
weight <- c( 11, 14, 17, 18, 17, 18, 11, 12, 10, 15 )
```

La función c() combina cada argumento dentro de un vector. Los dos vectores pueden juntarse en una misma base de datos

```
data <- data.frame( height, weight )
```

y podemos observar los resultados

```
data
```

Los elementos de columnas particulares de las bases de datos pueden ser observadas utilizando el símbolo $ y el nombre de la variable. Por ejemplo:

```
names( data )
```

```
data$height
```

O utilizando [] y el nombre de la columna. Por ejemplo,

```
data[1]
```

```
Otras posibilidades para visualizar filas o columnas:
```

```
data[  , 1]
```

```
data[ 1,  ]
```

Si es quisiera ver la estructura de un objeto, tales como la base de datos, se puede utilizar el comando

```
str(data)
```

Los objetos también suelen tener atributos

```
attributes(data)
```

Matriz

Una "matriz" es un objeto numérico ordenado como una matriz con dimensiones (dim()) dadas por el número de filas y columnas. Por ejemplo el objeto friend.data.w1 tiene las siguientes dimensiones:

```
dim( friend.data.w1 )
```

Si quisiera explorar los datos a través de una copia de la matriz, como ejemplo asignando el nombre "data", se puede hacer a través del siguiente comando:

```
data <- friend.data.w1
```

Los objetos anteriormente creados que tenían el nombre "data" ahora ha sido sobre-escrito por el objeto friend-data.w1.

Los elementos al interior de las matrices también pueden ser accedidos utilizando corchetes ([]). Por ejemplo, el elemento de "data" de la fila 4 y columna 5 se encuentra dado por

```
data[ 4,5 ]
```

Las filas dos, tres y cuatro de una matriz con nombre *data* se encuentra dado por

```
data[ 2:4, ]
```

Columnas 2, 5, y 6 son dados por

```
data[ , c( 1, 3, 7) ]
```

Como habrá observado, las matrices poseen muchos ceros debido a que las redes sociales en general son muy dispersas.

Convirtiendo una base de datos a matriz

La mayoría de las clases pueden ser convertirdas a otras clases. Por ejemplo, si "data" es un objeto que tiene una estructura similar al de un formato matriz, entonces se puede convertir a una clase "matriz" utilizando el comando

```
data <- as.matrix( data )
```

Lista de arcos

Desde la página web de Siena se pueden descargar los datos arclistdata.dat[6] para ilustrar el siguiente ejemplo. Para guardar los datos puede abrir un editor de texto (ejemplo: Notepad en Microsofr o Sublime Text) y guardar el archivo con terminación en "dat". Como podrá apreciar, los datos tienen un formato de lista de relaciones (*edge list*) en donde la primera columna corresponde al indicador único del emisor del vínculo, la segunda columna corresponde al identificar único del receptor del vínculo, la tercera a la intensidad de la relación, y la cuarta columna corresponde al momento u ola de tiempo.

Descargue los datos y guárdelos en su actual directorio de trabajo (en función de setwd() asignado al principio del tutorial). Ahora podrá leer la información:

```
ArcList <- read.table( "arclistdata.dat", header=FALSE )
```

Considere la capitalización del nombre del objeto creado.

Ahora ArcList es una base de datos (esto lo vimos con anterioridad en el archivo de ayuda read.table). A continuación, se explora las

[6] Url con los datos: http://www.stats.ox.ac.uk/~snijders/siena/arclistdata.dat

dimensiones de la información (número de filas y columnas):

```
dim(ArcList)
```

Puede realizar una visualización recortada de la base de datos mirando las primeras filas del objeto (seis por defecto):

```
head(ArcList)
```

Esto es un archivo de datos con un formato de *lista de arcos*, con columnas: **id** (identificador único del emisor), **receiver id** (identificador único del receptor), **value of tie** (valor de la relación), **wave** (ola). Agregue los nombres a las columnas:

```
names(ArcList) <- c( "sid", "recid", "bff", "wid" )
```

La variable bff (best friend o "mejores amigos") no tiene mucha variabilidad:

```
table(ArcList$bff)
```

Esto indica que la no presencia de vínculos no son incluídos (los cuales tendrían un valor de 0), y en cambio no hay otros valores de los vínculos distintos a 1.

Puede ser aconsejable de ordernar los datos primero por filas de los remitentes, después por receptores, y luego por ola. Una forma de hacer esto es a través de la siguiente línea de códigos:

```
ArcList <- ArcList[ order( ArcList$sid, ArcList$recid, ArcList$wid), ]
```

Para entender esto, se sugiere mirar la página de ayuda de la función ?order(). En donde las filas de ArcList son ordenadas primero por ArcList$sid, luego ArcList$recid, y finalmente por ArcList$wid. Esta función reordena filas y luego sobre-escribe el objeto ArcList.

Otra forma de hacer lo mismo es a través de la función with. La función with(a, b) indica a **R** que b debe ser calculado, mientras que los otros nombres desconocidos hacen alusión al conjunto de datos a:

```
ArcList <- with( ArcList, ArcList[ order( sid, recid, wid), ] )
```

Una lista de arcos no entrega información sobre el número de nodos, dado que los casos aislados no son registrados.

El conjunto de casos no-aislados son

```
union(unique(ArcList$sid), unique(ArcList$recid))
```

y, dado que sabemos que hay 50 nodos con etiquetas asignadas de 1 a 50, los casos aislados son los siguientes dos nodos:

```
setdiff(1:50, union(unique(ArcList$sid), unique(ArcList$recid)))
```

Ahora suponga que queremos crear un conjunto de datos separado que registre cada una de las olas por separada. Seleccionando las olas:

```
SAff.1 <- with(ArcList, ArcList[ wid == 1, ] ) #extracts edges in wave 1

SAff.2 <- with(ArcList, ArcList[ wid == 2, ] ) #extracts edges in wave 2

SAff.3 <- with(ArcList, ArcList[ wid == 3, ] ) #extracts edges in wave 3
```

Esto puede arreglarse de forma más eficiente utilizando el siguiente comando:

```
SAff <- lapply(1:3, function(m){ with(ArcList, ArcList[ wid == m, ] ) } )
```

Considerando que SAff.1 es SAff[[1]], etc.

Para transformar una matriz de adjacencia, por ejemplo, friend.data.w1, en una lista de arcos, hay que crear un indicador de matrices de las entradas que sean distintas de cero:

```
ones <- !friend.data.w1 %in% 0
```

Esto creará una lista de relaciones del ancho deseado

```
edges <- matrix(0, sum(ones), 3)
```

Ahora se puede rellenar el contenido de las columnas considerando la lista de relaciones

```
edges[, 1] <- row(friend.data.w1)[ones]

edges[, 2] <- col(friend.data.w1)[ones]

edges[, 3] <- friend.data.w1[ones]
```

Si es preferible, se puede ordenar la lista de relaciones ordenando primero por transmisores y luego receptores

```
edges <- edges[order(edges[, 1], edges[, 2]), ]
```

Para transformar una lista de arcos en formato matriz, primero se remueve la cuarta columna que indica la ola, y con ellos podrá disponer de los transmisores, los receptores y el valor de la relación, y luego podrá ser transformado a un formato matriz (en un principio es una data.frame)

```
SAff.1.copy <- SAff.1[, 1:3]

SAff.1.copy <- as.matrix(SAff.1.copy)
```

Ahora se crea una matriz de adjacencia vacía

```
adj <- matrix(0, 50, 50)
```

luego, es posible reconstruir la matriz con los valores en el lugar esperado

```
adj[SAff.1.copy[,1:2]] <- SAff.1.copy[, 3]
```

Ahora adj es una matriz de adjacencia. Considere que en la sección 4.1 del manual de Siena puede revisar los distintos formatos correspondientes a los datos de redes. De la misma forma en la página de ayuda en **R** puede revisar el comando ?sienaDependent.

Leer archivos de datos PAJEK y UCINET

Omita esta sección si el manejo de los datos de UCINET[7] o Pajek[8] no es de su interés.

Para el caso de **UCINET** existen un par de opciones disponibles en la página web[9] en donde encontrará un link a un paquete realizado por Jesse Fagan y otro script realizado por Christian Steglich para conectar datos de éste software con **R**. Otra opción disponible es, por ejemplo, a través del paquete "netdiffuseR" que no revisaremos en este tutorial. Como ejemplo utilizaremos el paquete "multiplex". Para ello, primero tendremos que instalar el paquete install.packages("multiplex") y luego podremos proceder al análisis:

```
library(multiplex)

samp <- read.dl("http://vlado.fmf.uni-lj.si/pub/networks/data/ucinet/sampson.dat")

multigraph(samp, valued=TRUE)
```

Si tiene datos en formato **Pajek** puede usar el paquete network de tal forma de convertir los archivos a un objeto red (recuerde que para instalar este paquete tendrá que utilizar install.packages("network")). Este ejemplo fue extraído de ?read.paj

```
par(mfrow=c(2,2))

library(network)


test.net.1 <- read.paj("http://vlado.fmf.uni-lj.si/pub/networks/data/GD/gd98/A98.net")

plot(test.net.1,main=test.net.1%n%'title')


test.net.2 <- read.paj("http://vlado.fmf.uni-lj.si/pub/networks/data/mix/USAir97.net")

# plot using coordinates from the file in the file

plot(test.net.2,main=test.net.2%n%'title',


coord=cbind(test.net.2%v%'x',

            test.net.2%v%'y'),

            jitter=FALSE)
```

Casos perdidos

Hasta aquí se han revisado varias fuentes alternativas para leer datos en distintos formatos y de distintas fuentes para ser leídos al interior de **R**. Sin embargo, para finalizar este tutorial utilizaremos nuevamente los datos contenidos por defecto en el paquete **RSiena**.

Antes de comenzar a trabajar con los datos, vamos a asegurarnos que los datos sean los correctos. Una forma sencilla de revisar que los datos sean una matriz es a través del comando class():

```
class( friend.data.w1 )
```

Para enlistar las propiedades de un objeto: attributes(friend.data.w1) (diferentes clases tienen diferentes atributos)

Para revisar que todos los datos se encuentren correctamente leídos, se puede utilizar el comando dim(). La matriz de adjacencia debería tener las mismas dimensiones que los archivos originales (aquí 50 por 50) y los atributos debiesen corresponder a las tres columnas consideradas por cada individuo (50 por 3):

```
dim(friend.data.w1)

dim(drink)
```

Para evaluar que los datos sean correctos, incluyendo los datos perdidos, podemos utilizar los siguientes comandos para tabular las variables

```
table( friend.data.w1, useNA = 'always' )

table( friend.data.w2, useNA = 'always' )

table( friend.data.w3, useNA = 'always' )

table( drink, useNA = 'always' )
```

[7] Url de UCINET:

https://sites.google.com/site/ucinetsoftware/home

[8] Url de Pajek: http://mrvar.fdv.uni-lj.si/pajek/

[9] Url para conectar UCINET con R:

https://sites.google.com/site/ucinetsoftware/document/faq/connectingwithr

```
table( smoke, useNA = 'always' )
```

NA es el código de **R** para datos perdidos (Not Available o No Disponible).

Estos datos en particular no tienen casos perdidos (tal como se señala en la descripción de la página de Siena).

Sin embargo, si es que hay casos perdidos, es necesario indicarle a **R** los casos perdidos a través de códigos asignando NA a los elementos correspondientes.

Supongamos que los casos perdidos faltantes en la red de amistad fueran los casos 6 y 9. Esto llevaría a los siguientes comandos (para nuevos usuarios de **R**: la función c() utilizada aquí como c(6,9) construye un vector [c por columna] los cuales consisten en los valores 6 y 9. Esta función se ocupa bastante en las funciones basales de **R**)

```
friend.data.w1[,c(6,9)] <- NA
friend.data.w2[,c(6,9)] <- NA
friend.data.w3[,c(6,9)] <- NA
```

Comandos para el análisis descriptivo se encuentra en el script: RScriptSNADescriptives.R[10]

Seleccionando un subconjunto de datos

Para seleccionar un subconjunto de datos basados en una variable de un actor, digamos, aquellos que tienen valores 2 y 3 que beben alcohol en tiempo 1 (las posibilidades son interminable, pero este ejemplo busca identificar algún patrón en los datos)

```
use <- drink[, 1] %in% c(2, 3)
```

Esta función creará un vector lógico en donde aparecerá TRUE para los casos en donde se satisface esta condición. Para ver o revisar, para desplegar los vectores

```
drink[ , 1 ]
use
```

y el número de casos seleccionados se despliega como (para este caso sumará aquellos casos que sean TRUE y los tratará como valor 1 en sistema booleano):

```
sum( use )
```

o

```
table( use )
```

Para tener este conjunto más ordenado y uno al lado del otro, puede crear y mostrar una matriz con la información deseada,

```
aa <- matrix(nrow=50, ncol=2)
aa[,1] <- drink[,1]
aa[,2] <- use
aa
```

La primera columna contiene el nivel de consumo de alcohol y la segunda columna si el nivel es 2 o 3 (1) o no (0)

o una alternativa más corta:

```
aa <- cbind(drink[ , 1 ],use)
aa
```

Dada esta selección, submatrices pueden formarse en caso en que el análisis quiera hacerse utilizando solo este subconjunto:

```
friend1.data.w1 <- friend.data.w1[ use, use ]
friend1.data.w2 <- friend.data.w2[ use, use ]
drink1 <- drink[ use, ]
```

Como podrá apreciar, las matrices ahora serán más pequeñas tal como se puede observar a continuación:

```
dim(friend1.data.w1)
dim(friend1.data.w2)
dim(drink1 )
```

Una opción útil en **R,** que permite guardar el espacio de trabajo:

```
save.image("WorkspaceRscript01.Rdata")
```

Posteriormente, puede cargar esto en una nueva sesión:

```
load("WorkspaceRscript01.RData")
```

Para una próxima oportunidad podrá continuar desde aquí, no será necesario abrir y ejecutar este script denuevo, dado que tendrá la oportunidad de cargar el estado actual del espacio de trabajo. Sin embargo, los paquetes de **R** deberán ser cargados nuevamente (ejemplo: library(RSiena)).

Ahora puede proceder al análisis descriptivo de los datos en Rscript01DataFormat.R en su

[10] Url en donde se encuentra el archivo:
https://www.stats.ox.ac.uk/~snijders/siena/RSienaSNADescriptives.R

versión en inglés, y que será próximamente presentado en castellano.

Comentarios finales

En este guión se introdujo brevemente la descarga y usos básicos del software **R y RStudio**. En donde el primero corresponde al lenguaje de programación, mientras que el segundo es un "intérprete" del primero. A continuación, se exploró brevemente el paquete **RSiena**. Luego se importaron datos y se realizaron algunas manipulaciones preliminares. Además, se presentó cómo cambiar los datos desde una base de datos a objetos en formato matriz. Posteriormente, se introdujo la utilización de listas de arcos para el análisis de datos, además, se ofrecieron algunas alternativas para leer archivos provenientes de softwares populares del análisis de redes sociales como UCINET y Pajek. Finalmente, se asignaron casos perdidos a los datos y se realizó una selección de un subconjunto de datos.

Los siguientes guiones buscarán profundizar en el análisis descriptivo y en otros elementos de los modelos estocásticos basados en actores que permitirán obtener un entendimiento más profundo sobre las posibilidades que entrega **RSiena**. Esperamos que esta introducción permita guiar el uso de **R** y su interoperabilidad con distintos formatos y archivos para el análisis de redes, acorde al estado actual de su desarrollo y que sea accesible a la comunidad que habla español.

Agradecimientos

Agradezco particularmente a Tom A.B. Snijders (University of Oxford y University of Groningen) por su amabilidad en permitirme difundir, modificar y extender los siguientes guiones los cuales contaron también con contribuciones previas de Robin Gauthier, Ruth Ripley, Johan Koskinen, Paulina Preciado, Zsofia Boda y Christian Steglich. En un principio estos documentos iban ser una traducción textual a los guiones disponibles en la página de RSiena. No obstante, por sugerencia de Tom A. B. Snijders los guiones traducidos y modificados adquieren con esta contribución independencia de sus homólogos en inglés con el objeto de facilitar su difusión, mantensión y responsabilidad en su contenido en lengua castellana del cual espero reflejar adecuadamente sus intensiones originarias.

REFERENCIAS

Espinosa-Rada, Alejandro (2022). Modelos estocásticos orientados en el actor utilizando RSiena (I). Guión básico introductorio. *Revista Hispana para el Análisis de Redes Sociales, 33*(1), 92-99.

Ripley, Ruth; Snijders, Tom A.B.; Boda, Zsófia; Vörös, András, y Preciado, Paulina (2021). *Manual for SIENA version 4.0*. Oxford: University of Oxford, Department of Statistics, http://www.stats.ox.ac.uk/~snijders/siena/

Snijders, Tom A.B. (2001) The statistical evaluation of social network dynamics, *Sociological Methodology*, 40: 361-395.

Snijders, Tom A.B. (2017). Stochastic Actor-Oriented Models for Network Dynamics. *Annual Review of Statistics and Its Application*, 4: 343-363. DOI: http://dx.doi.org/10.1146/annurev-statistics-060116-054035

R Core Team (2021). *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. URL: https://www.R-project.org/

Remitido: 04-04-2021

Aceptado: 11-11-2021