Modulo6 - continuación

Jorge Fábrega

Version 2023

En esta sesión, vamos a explorar la estadística detrás de la formación de redes sociales a través de los Modelos de Grafos Aleatorios Esperados (ERGMs).

¿Cómo testear si un modelo explica bien o no las propiedades básicas de la red (datos) que tenemos? Como vimos en la sesión anterior, para cada red que observamos, existen otras redes posibles con el mismo número de nodos y una o más características similares a la red observada (por ej. número de enlaces, tríadas, etc.). Por lo tanto, cabe pregutar: ¿Qué tan probable es que la red que observamos sea el resultado de un proceso aleatorio?

Para explorar una respuesta, vimos que lo primero era revisar los modelos clásicos como Erdös-Renyi, que asumen una formación de enlaces independiente e idénticamente distribuida (i.i.d.). Visto en retrospectiva, esa aproximación no es muy realista. Luego, vimos cómo modelos como Small Worlds y Preferential Attachment también pueden quedar cortos. Necesitamos introducir más realismo a los modelos, lo que implica considerar, por ejemplo, que la probabilidad de formar un enlace entre dos nodos específicos puede estar relacionada con la existencia de otros enlaces en la red. Pero si hacemos eso, la formación de enlaces ya no será i.i.d.

Un modelo estadístico de una red busca explicar la formación de la red. Por eso, en esta sesión nos enfocaremos en las aproximaciones estadísticas para el testeo de hipótesis en redes, analizando cómo los Expected Random Graph Models nos permiten hacer inferencias topológicas y entender mejor la estructura y formación de las redes sociales que observamos. Hay otros métodos que no vamos a cubrir, pero que sería deseable que revisaran por su cuenta, tales como: Stochastic Block Models, Stochastic Actor Oriented Models, Latent Network Models y Stochastic Random Fields Networks.

Introducción a ERGMs

El Modelo de Grafos Aleatorios Esperados (ERGM) es una herramienta de larga data en el análisis de redes, actuando como el equivalente en redes a un modelo OLS en econometría. Este modelo permite testear hipótesis sobre qué factores influyen en la formación de una red de manera similar a una regresión.

El origen del ERGM se remonta a los desafíos abiertos por los trabajos pioneros sobre teoría de grafos à la Erdös-Renyi. Estos modelos eran demasiado simplistas para capturar la complejidad de las redes sociales reales que empezaban a generarse (de baja escala aún en términos del número de nodos). Por ello, en los años '80s y '90s, los ERGM comenzaron a tomar forma como una extensión más realista y matizada de los modelos de grafos aleatorios. Los trabajos de Paul Holland y Samuel Leinhardt sobre modelos p* en la década de 1980 fueron cruciales para el desarrollo de los ERGM, proporcionando un marco para modelar la dependencia de las aristas y la estructura de las redes sociales (nota: si ven un modelo sobre p* es esencialmente lo mismo que hablar de ERGM).

Los modelos p* de Holland y Leinhardt introdujeron la idea de modelar la probabilidad de formación de enlaces en redes sociales considerando la estructura existente de la red y, por ende, donde la probabilidad de cada enlace es condicional a lo que suceda con otros enlaces.

Otra contribución central fue el libro de Wasserman y Faust "Social Network Analysis: Methods and Applications" (1994) que está en la bibliografía del curso. Este trabajo es ampliamente reconocido por consolidar y expandir las aplicaciones de análisis de redes incluida los ERGMs durante los 90s.

Los avances en la estadística de redes en la década de 2000 ampliaron las capacidades de los ERGM, permitiendo modelar con mayor precisión complejas interacciones y dependencias en las redes. Hoy en día, los ERGM se utilizan en una variedad de campos y aplicaciones, incluyendo estimaciones dirigidas a entender las dinámicas de coaliciones políticas o las redes de colaboración científica, la modelización de la propagación de enfermedades a través de redes sociales y contactos humanos, para analizar redes de transacciones y patrones de consumo, así como influencias en redes de marketing, o para el análisis de redes de comunicación y detección de comportamientos anómalos o ataques cibernéticos.

Usos de ERGM

Si consideramos una red G, el ERGM nos permite examinar aspectos como:

- Si la cantidad de conexiones (densidad) es significativa en la formación de la red.
- Si la reciprocidad (relaciones no dirigidas) juega un papel importante.
- Si la formación de comunidades (tríadas) es relevante.
- Si los atributos de los nodos (distribución, homofilia, etc.) son factores clave.

Funcionamiento del ERGM

El ERGM funciona de manera similar a una regresión logística, pero aplicado a redes. Busca la probabilidad de que exista un enlace entre dos nodos dados, i y j, dada la estructura del grafo y las características consideradas relevantes. Matemáticamente, esto se representa como:

$$P(a_{ij} = 1|G)$$

siendo G las características de la red.

El resultado de un ERGM se presenta como un odd-ratio para cada una de estas características, calculado mediante la fórmula:

$$log(\frac{P(a_{ij} = 1|G)}{P(a_{ij} = 0|G)})$$

Por ejemplo, si un modelo considera solo el número de enlaces y se obtiene un valor de -1.19, la probabilidad correspondiente se calcula como:

$$\frac{\exp(-1.19)}{1 + \exp(-1.19)}$$

Es decir, una probabilidad aproximada de 0.23 de que exista el enlace.

Formalización del ERGM

En un ERGM, consideramos una red $G = (V, E)$ y definimos $Y_{ij} = Y_{ji}$ como una variable que toma el valor 1 si el enlace e_{ij} existe en E. La matriz adyacente aleatoria Y representa todas estas variables, donde y es una realización específica de esta matriz. El ERGM modela la distribución de probabilidad conjunta de los elementos de Y como:

$$P_\theta(Y = y) = \frac{1}{k} \exp\left\{\sum_H \theta_H g_H(y)\right\}$$

Aquí, H representa una 'configuración' o conjunto de posibles enlaces entre los vértices de G, y $g_H(y)$ es un producto de todos los y_{ij} en la configuración H, siendo 1 si la configuración H ocurre y 0 si no. La constante de normalización k depende de θ y asegura que la distribución sea válida.

Distribuciones Exponenciales en ERGM

Una razón clave para usar una función exponencial en este modelo es que las distribuciones exponenciales no tienen memoria (nota: ¿cómo es eso? Ver *aquí)* o *aquí*. Esto significa que podemos estudiar las probabilidades condicionales de cada enlace independientemente de los demás, asumiendo que son eventos independientes, lo cual simplifica significativamente el análisis.

Cómo Funcionan los Modelos de Grafos Aleatorios Esperados (ERGMs)

Vamos a explorar el funcionamiento de los ERGMs, basándonos en el Teorema de Hammersley–Clifford (1971) y el método Markov Chain Monte Carlo (MCMC).

Teorema de Hammersley–Clifford (1971)

Según este teorema, toda distribución de probabilidad puede representarse como una cadena de Markov si cumple con ciertos requisitos en la distribución de probabilidades. Las distribuciones exponenciales, que son la base de los ERGMs, cumplen con esos requisitos.

La implicancia es que toda familia de modelos de redes puede ser expresada potencialmente como un ERGM si se cuenta con las estadísticas fundamentales $(S1, S2, \ldots)$ de dicha familia de modelos. La probabilidad de un grafo g en un ERGM se representa como:

$$Pr(g) = \frac{e^{(\theta_1 S_1(g) + \theta_2 S_2(g) + \ldots + \theta_k S_k(g))}}{\sum_{g' \in G} e^{(\theta_1 S_1(g') + \theta_2 S_2(g') + \ldots + \theta_k S_k(g'))}}$$

Markov Chain Monte Carlo (MCMC)

En redes pequeñas (n < 8), es posible estimar todas las configuraciones para un set de hipótesis/configuraciones (véase paquete *ergmito* en R). En redes más grandes, deben hacerse aproximaciones estadísticas utilizando MCMC. Este método se basa en dos principios:

1. **Markov Chain**: Sólo el estado inmediatamente anterior importa, lo que permite separar la estimación en bloques.
2. **Monte Carlo**: Se utiliza la aleatorización para sacar estimaciones de parámetros.

La combinación de estos dos principios permite realizar simulaciones sucesivas para buscar la distribución que mejor explica las estadísticas observadas en la red.

Pasos para la Estimación vía ERGMs

1. **Paso 1**: Cada enlace se trata como una variable aleatoria.
2. **Paso 2**: Se postula un conjunto de hipótesis para explicar la formación de enlaces (ejemplo: enlaces aleatorios, por cercanía, homofilia, cierre triádico, etc.).
3. **Paso 3**: Cada hipótesis es una configuración que se suma a otras posibles configuraciones.
4. **Paso 4**: Las configuraciones, aunque locales, son homogéneas en toda la red, lo que ayuda a reducir la cantidad de parámetros.
5. **Paso 5**: Se realiza la estimación e interpretación de los parámetros del modelo.

Ejemplo de Estudio de una Red de Erdös-Renyi

Supongamos que queremos estudiar simultáneamente la probabilidad del grafo g considerando no sólo el número de enlaces sino también el número de tríadas. En una red de Erdös-Renyi (donde todos los enlaces son independientes entre sí):

$$Pr(g) = p^m(1-p)^{\binom{n}{2}-m} = \frac{p^m}{(1-p)^m} * (1-p)^{\binom{n}{2}}$$

$$Pr(g) = e^{(\log\left(\frac{p}{1-p}\right)*m+constante)}$$

$$Pr(g) = e^{(\theta_m m(g)+constante)} \text{ sobre todos los posibles grafos } g'$$

Entonces, sea $m(g)$ el número de enlaces y $T(g)$ el número de tríadas de la red g. La probabilidad del grafo g depende de:

$$\theta_m m(g) + \theta_T T(g)$$

$$e^{(\theta_m m(g)+\theta_T T(g))}$$

$$Pr(g) = \frac{e^{\theta_m m(g)+\theta_T T(g)}}{\sum_{g' \in G} e^{\theta_m m(g)+\theta_T T(g)}}$$

Implementación Práctica de un ERGM en R:

La implementación práctica de un ERGM (Modelo de Grafos Aleatorios Esperados) en R se puede realizar usando el paquete `statnet`, que es un compilado de varios paquetes relacionados con el análisis de redes. Para más detalles, consulta el Proyecto Statnet.

Algoritmo para ERGM en R

El algoritmo para estimar un ERGM en R generalmente sigue estos pasos:

1. Estimación de los parámetros $\theta_1, \theta_2, \theta_3, \ldots$ usando varios métodos, como el de máxima verosimilitud.
2. Simulación de redes con las propiedades $\theta_1, \theta_2, \theta_3, \ldots$.
3. Actualización de los valores de $\theta_1, \theta_2, \theta_3, \ldots$ a partir de esas redes simuladas.
4. Repetición del paso 2 hasta lograr convergencia.

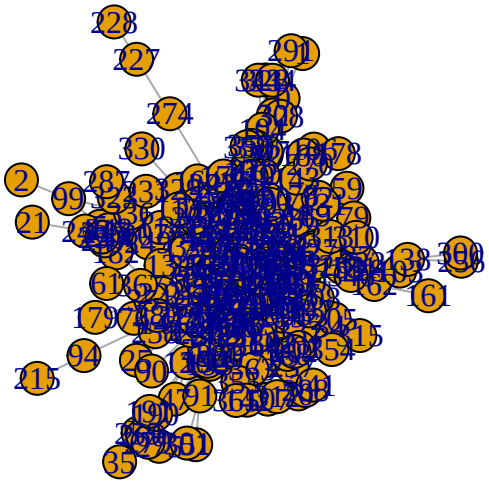
Aplicación a una red

Modelo Basado en el Número de Enlaces

Veamos brevemente la red que habíamos estado trabajando de interlocking. Es una red relativamente pequeña para los usos que le habíamos estado dando, pero relativamente grande para estimar ERGMs en mi computador. *¯_(**)_/¯*

```
# Cargamos paquetes
library(here)
library(igraph)
library(intergraph)
camino <- here()
```

```r
# como igraph puede interferir con statnet vamos a cargar la red
# luego vamos a pasarla a un formato que la lee statnet y
# luego vamos a sacar de la sesión a igraph
archivo <- paste0(camino,"/docs/red_empresas.igraph")
red.e <- read_graph(archivo, format="ncol")
componentes.e <- clusters(red.e)
g.e <- which.max(componentes.e$csize) # identificamos el gigante
red.e <- induced.subgraph(red.e, which(componentes.e$membership == g.e))

# Guardamos algunos estadígrafos de la red
e.grado <- degree_distribution(red.e)
e.clust <- transitivity(red.e)
e.betw <- betweenness(red.e)
e.close <- closeness(red.e)

# Visualización
plot(red.e)
```



```r
red.e <- intergraph::asNetwork(red.e)

detach("package:igraph", unload = TRUE)

library(netrankr)
library(ergm)
```

```r
# Estimamos el modelo ERGM con solo la variable 'edges' que representa enlaces
# La fórmula flomarriage ~ edges indica que estamos modelando la red de matrimonios
# basándonos únicamente en la presencia de enlaces entre los nodos
ergm_model <- ergm(red.e ~ edges)

# Mostramos un resumen del modelo para entender sus parámetros
summary(ergm_model)
```

```
## Call:
## ergm(formula = red.e ~ edges)
##
## Maximum Likelihood Results:
##
##       Estimate Std. Error MCMC % z value Pr(>|z|)
## edges -3.59595    0.03368      0  -106.8   <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 47037  on 33930  degrees of freedom
##  Residual Deviance:  8352  on 33929  degrees of freedom
##
## AIC: 8354  BIC: 8363  (Smaller is better. MC Std. Err. = 0)
```

Interpretación: La probabilidad de un enlace es $Pr(l_{ij} = 1) = \frac{e^{-3.59595}}{1+e^{-3.59595}} = 0.0267$.
El 'odd ratio' para el cambio de un enlace de 0 a 1 es 0.0267. Este resultado indica que la probabilidad de formar un enlace (edge) en la red es bastante baja en comparación con la probabilidad de no formar uno. Esto es consistente con la proporción real de enlaces en la red (906 links reales / 33920 posibles). Como el P-value es menor que 0.05, por convención, aceptamos que el número de enlaces es significativo (lo cual en realidad no dice mucho).

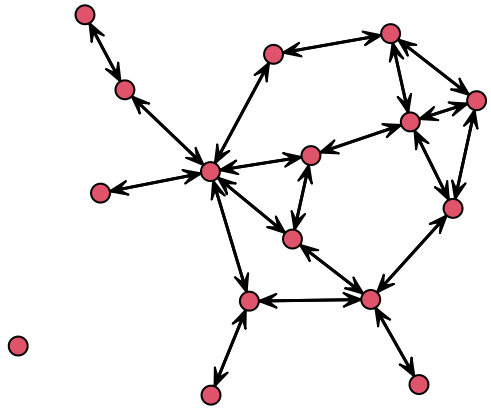
Ampliando el Modelo: Inclusión otros estadígrafos
Con ERGMs podemos testear el efecto de otros estadigrafos en una red. Por ejemplo, para una red altamente dispersa como la de interlocking se pueden agregar términos para que introduzcan restricciones que garanticen que haya nodos de cierto grado usando el término *degree(k)*. También se pueden considerar nodos que sean hubs usando el término *kstar(j)* donde j representa el número de nodos a los que está vinculada la estrella. Como la red tenía alta clusterización se puede incluir una medida para ello usando *triangle*. Todo esto requiere estudiar previamente la red y su estructura en detalle y tener hipótesis específicas sobre ella. De lo contrario, puedo asegurarles que la estimación será una pesadilla que nunca converge, con iteraciones infinitas y una importante pérdida de tiempo. De hecho, mientras escribo estas líneas está corriendo el siguiente código (por ya demasiado tiempo y sin éxito).

ergm_model2 <- ergm(red.e ~ edges + triangle + degree(2) + degree(3) + degree(4) + degree(5) + kstar(3)),control = control.ergm(MCMLE.density.guard = 25))

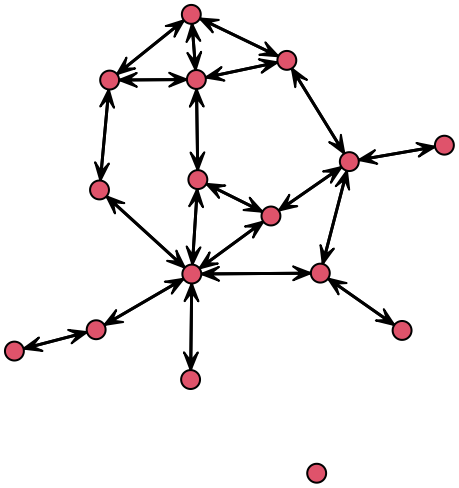
...

Bueno, como la red anterior requeriría de mucho más trabajo de análisis para ser sometida a un ERGM, vamos a continuar con una red muy útil y estudiada para este mismo objetivo. La red de matrimonios de familias adineradas de la Florencia del Renacimiento, más conocida como la red Medici.

```r
library(statnet)
library(netrankr)
library(intergraph)
data(florentine_m)
flomarriage <- intergraph::asNetwork(florentine_m)

gplot(flomarriage, main="Matrimonios")
```

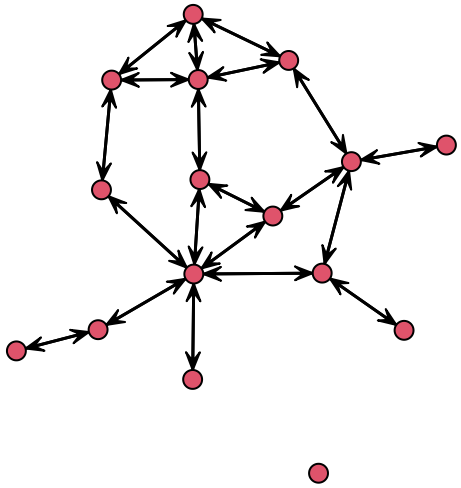



```
coordinadas_plot <- (gplot(flomarriage))
```



```
gplot(flomarriage, main="Matrimonios", coord = coordinadas_plot)
```

Matrimonios



```
tabla_atributos <- matrix(NA,nrow=16,ncol=2)
for(i in 1:16){
  tabla_atributos[i,1] <- flomarriage$val[[i]]$vertex.names
  tabla_atributos[i,2] <- flomarriage$val[[i]]$wealth
}
```

```
ergm_model2 <- ergm(flomarriage ~ edges)
```

```
## Starting maximum pseudolikelihood estimation (MPLE):
## Obtaining the responsible dyads.
## Evaluating the predictor and response matrix.
## Maximizing the pseudolikelihood.
## Finished MPLE.
## Evaluating log-likelihood at the estimate.
```

```
summary(ergm_model2)
```

```
## Call:
## ergm(formula = flomarriage ~ edges)
##
## Maximum Likelihood Results:
##
##       Estimate Std. Error MCMC % z value Pr(>|z|)
## edges  -1.6094     0.2449      0  -6.571   <1e-04 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 166.4  on 120  degrees of freedom
##  Residual Deviance: 108.1  on 119  degrees of freedom
##
## AIC: 110.1  BIC: 112.9  (Smaller is better. MC Std. Err. = 0)
```

Interpretación: El 'odd ratio' para el cambio de un enlace de 0 a 1 es -1.609. La probabilidad de un enlace es $Pr(lij = 1) = \frac{e^{-1.609}}{1+e^{-1.609}} = 0.1667$. Esto es consistente con la proporción real de enlaces en la red (20 links reales / 120 posibles).

Incorporemos clusterización (usando la opción *triangle*)

```
ergm_model_triangle <- ergm(flomarriage ~ edges + triangle)
summary(ergm_model_triangle)
```

```
## Call:
## ergm(formula = flomarriage ~ edges + triangle)
##
## Monte Carlo Maximum Likelihood Results:
##
##          Estimate Std. Error MCMC % z value Pr(>|z|)
## edges     -1.6598     0.3684      0  -4.505   <1e-04 ***
## triangle   0.1304     0.5643      0   0.231    0.817
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 166.4  on 120  degrees of freedom
##  Residual Deviance: 108.1  on 118  degrees of freedom
##
## AIC: 112.1  BIC: 117.7  (Smaller is better. MC Std. Err. = 0.008054)
```

```
# para interpretar: log odds
triadas0 <- ergm_model_triangle$coef[1] # 0 triada
triadas1 <- ergm_model_triangle$coef[1] + ergm_model_triangle$coef[2] # 1 triada
triadas2 <- ergm_model_triangle$coef[1] + 2*ergm_model_triangle$coef[2] # 2 triada

exp(triadas0)/(1+exp(triadas0)) # prob de link con 0 triada
```

```
##     edges
## 0.1597851
```

```
exp(triadas1)/(1+exp(triadas1)) # prob de link con 1 triada
```

```
##     edges
## 0.1780836
```

```
exp(triadas2)/(1+exp(triadas2)) # prob de link con 2 triada
```

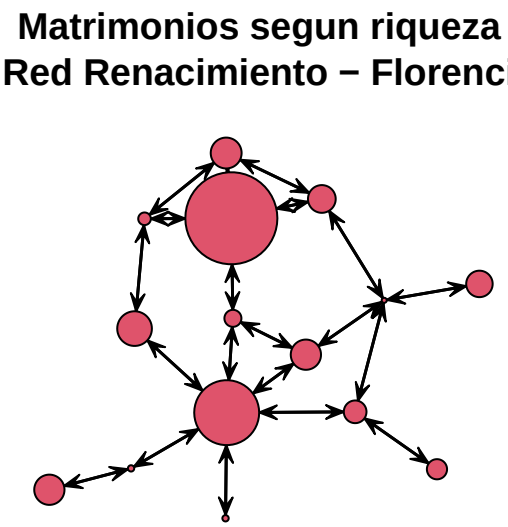
```
##     edges
## 0.1979837
```

Se incluyen triángulos en el modelo. El 'odd ratio' si T (número de triángulos) = 1: $-1.609 + 0.13 = -1.47$. La probabilidad de un enlace, dado 1 triángulo, es aproximadamente 0.185.

Como esta red tiene atributos de los nodos podemos además testear su impacto. Por ejemplo ¿tienen más conexiones/matrimonios aquellas familias con más riqueza?

Modelo con Covariables de Nodo: Riqueza

```r
# Incorporamos la 'riqueza' como covariable de nodo
# Esto permite evaluar cómo la riqueza de las familias afecta la formación de enlaces

riqueza <- flomarriage %v% 'wealth' # asigna el atributo wealth a la variable riqueza
gplot(flomarriage,
      vertex.cex = riqueza/30,
      main = "Matrimonios segun riqueza \n Red Renacimiento - Florencia",
      coord = coordinadas_plot)
```



.

```r
ergm_model_wealth <- ergm(flomarriage ~ edges + triangle + nodecov("wealth"))

# Resumen del modelo con la covariable de riqueza incluida
summary(ergm_model_wealth)
```

```
## Call:
## ergm(formula = flomarriage ~ edges + triangle + nodecov("wealth"))
##
## Monte Carlo Maximum Likelihood Results:
##
##                  Estimate Std. Error MCMC % z value Pr(>|z|)
## edges           -2.520652   0.550967      0  -4.575   <1e-04 ***
## triangle        -0.035238   0.669229      0  -0.053   0.9580
## nodecov.wealth   0.010087   0.005039      0   2.002   0.0453 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
##      Null Deviance: 166.4  on 120  degrees of freedom
##  Residual Deviance: 103.1  on 117  degrees of freedom
##
## AIC: 109.1  BIC: 117.5  (Smaller is better. MC Std. Err. = 0.006948)
```

```
# veamos la probabilidad que Medicci este conectado con Ridolfi (que lo esta)
# versus que Barbadori este conectado con Ridolfi (que no lo esta) - ignoremos triadas

e_ridolfi_medici <- ergm_model_wealth$coef[1] +
  ergm_model_wealth$coef[3]*flomarriage$val[[9]]$wealth +
  ergm_model_wealth$coef[3]*flomarriage$val[[13]]$wealth
e_ridolfi_bardabori <- ergm_model_wealth$coef[1] +
  ergm_model_wealth$coef[3]*flomarriage$val[[3]]$wealth +
  ergm_model_wealth$coef[3]*flomarriage$val[[13]]$wealth

exp(e_ridolfi_medici)/(1+exp(e_ridolfi_medici))
```

```
##     edges
## 0.2298176
```

```
exp(e_ridolfi_bardabori)/(1+exp(e_ridolfi_bardabori))
```

```
##     edges
## 0.1553138
```

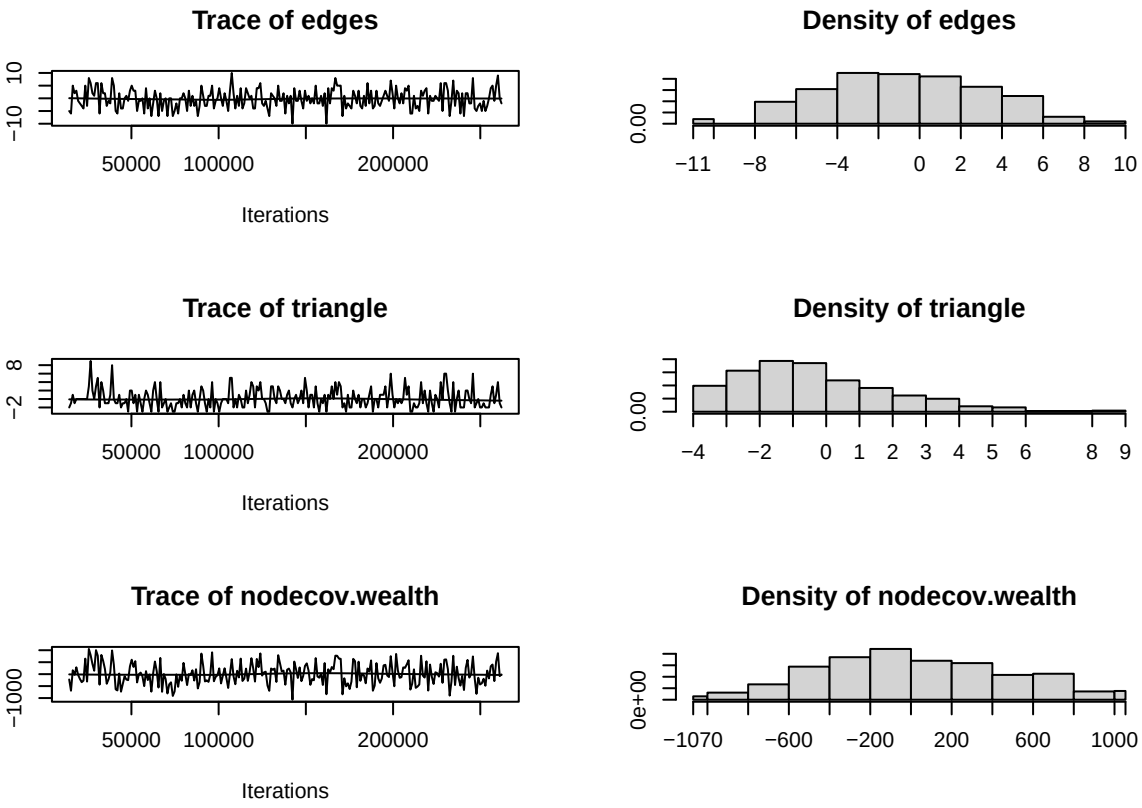
```
library(texreg)
screenreg(list(ergm_model2, ergm_model_triangle, ergm_model_wealth))
```

```
##
## ===================================================
##                  Model 1     Model 2     Model 3
## ---------------------------------------------------
## edges             -1.61 ***   -1.66 ***   -2.52 ***
##                   (0.24)      (0.37)      (0.55)
## triangle                       0.13       -0.04
##                               (0.56)      (0.67)
## nodecov.wealth                             0.01 *
##                                           (0.01)
## ---------------------------------------------------
## AIC              110.13      112.08      109.12
## BIC              112.92      117.66      117.48
## Log Likelihood  -54.07      -54.04      -51.56
## ===================================================
## *** p < 0.001; ** p < 0.01; * p < 0.05
```

Extensiones Posibles:
Análisis de Ajuste del Modelo: Podríamos realizar un análisis de bondad de ajuste (Goodness-of-Fit, GOF) para evaluar qué tan bien el modelo se ajusta a los datos observados. El paquete *ergm* en R trae opciones por defecto. Por ejemplo se puede estudiar homofilia con la funcion *nodematch()*. Para una lista completa de los atributos incluidos ver: *search.ergmTerms()*.

Veamos qué tan bien/mal se comportó la MCMC. Lo que buscamos es el *"caterpillar"*. Y como se ve en las imágenes siguientes no es bueno... pero podría ser peor.

```
mcmc.diagnostics(ergm_model_wealth)
```




```
## Sample statistics summary:
##
## Iterations = 14336:262144
## Thinning interval = 1024
## Number of chains = 1
## Sample size per chain = 243
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                    Mean      SD Naive SE Time-series SE
## edges           -0.1934   3.854   0.2472         0.2472
## triangle         0.1523   2.288   0.1468         0.1468
## nodecov.wealth  6.9383 446.255  28.6273        28.6273
##
## 2. Quantiles for each variable:
##
##                   2.5%    25% 50%   75% 97.5%
## edges             -7.0   -3.0   0   3.0   7.0
## triangle          -3.0   -2.0   0   1.0   5.0
## nodecov.wealth -803.5 -311.5 -30 299.5 859.2
##
##
## Are sample statistics significantly different from observed?
##                   edges  triangle nodecov.wealth     (Omni)
## diff.      -0.1934156 0.1522634      6.9382716         NA
```

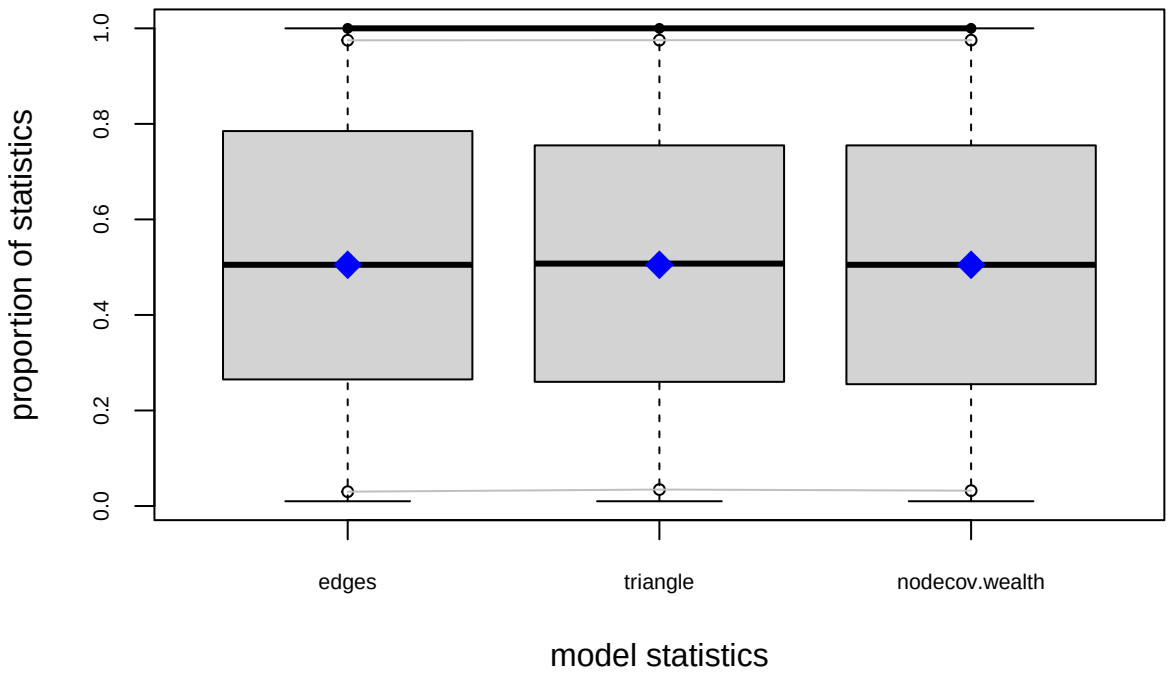
```
## test stat. -0.7823820 1.0371825      0.2423656 7.07232418
## P-val.      0.4339901 0.2996508      0.8084969 0.07427791
##
## Sample statistics cross-correlations:
##                    edges  triangle nodecov.wealth
## edges          1.0000000 0.7094734      0.8772612
## triangle       0.7094734 1.0000000      0.7464135
## nodecov.wealth 0.8772612 0.7464135      1.0000000
##
## Sample statistics auto-correlation:
## Chain 1
##                edges    triangle nodecov.wealth
## Lag 0     1.00000000  1.00000000    1.000000000
## Lag 1024 -0.02915322  0.07395969    0.048982208
## Lag 2048  0.01762678 -0.04465457   -0.032044707
## Lag 3072  0.03967930  0.10979024    0.117492222
## Lag 4096  0.02343200  0.01429836    0.052847099
## Lag 5120 -0.04736117 -0.02687005   -0.008734958
##
## Sample statistics burn-in diagnostic (Geweke):
## Chain 1
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##          edges       triangle nodecov.wealth
##       1.167371       1.677935       1.648701
##
## Individual P-values (lower = worse):
##          edges       triangle nodecov.wealth
##     0.24306078     0.09335976     0.09920888
## Joint P-value (lower = worse):  0.1694937
##
## Note: MCMC diagnostics shown here are from the last round of
##   simulation, prior to computation of final parameter estimates.
##   Because the final estimates are refinements of those used for this
##   simulation run, these diagnostics may understate model performance.
##   To directly assess the performance of the final model on in-model
##   statistics, please use the GOF command: gof(ergmFitObject,
##   GOF=~model).
```

Veamos ahora qué tan bien estuvo el modelo anterior (GOF)

```
ergm_model_wealth.gof <- gof(ergm_model_wealth ~ model) # usando las variables del modelo
ergm_model_wealth.gof
```

```
##
## Goodness-of-fit for model statistics
##
##                 obs  min    mean  max MC p-value
## edges            20   46   59.17   73          0
## triangle          3   27   67.61  114          0
## nodecov.wealth 2168 3957 5054.78 6453          0
```

```
plot(ergm_model_wealth.gof)
```




```
# contrastando con otras variables de la red que no estan en el modelo
ergm_model_wealth.gof.degree <- gof(ergm_model_wealth ~ degree + esp + distance) # esp = edgewise share
ergm_model_wealth.gof.degree
```

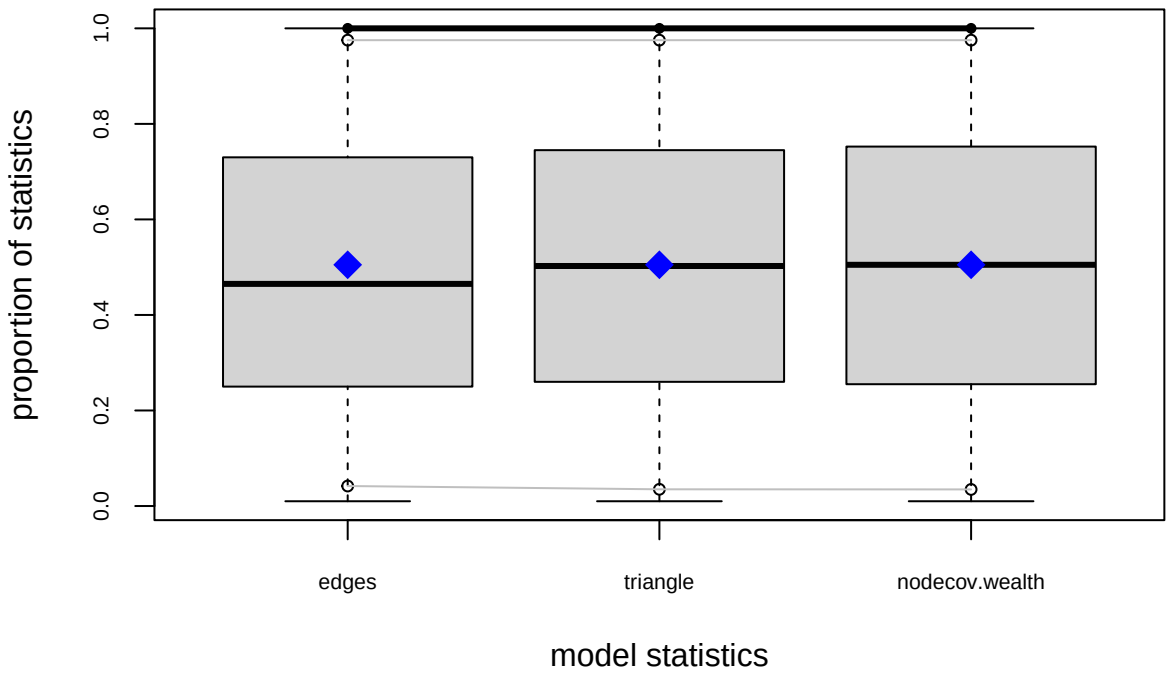
```
##
## Goodness-of-fit for degree
##
##          obs min mean max MC p-value
## degree0    1   0 0.00   0       0.00
## degree1    4   0 0.02   1       0.00
## degree2    2   0 0.05   1       0.00
## degree3    6   0 0.32   2       0.00
## degree4    2   0 0.69   4       0.32
## degree5    0   0 1.60   6       0.38
## degree6    1   0 2.28   6       0.56
## degree7    0   1 2.97   7       0.00
## degree8    0   0 3.35   8       0.02
## degree9    0   0 2.49   7       0.10
## degree10   0   0 1.45   6       0.56
## degree11   0   0 0.54   3       1.00
## degree12   0   0 0.18   3       1.00
## degree13   0   0 0.06   2       1.00
##
## Goodness-of-fit for edgewise shared partner
##
##       obs min  mean max MC p-value
```

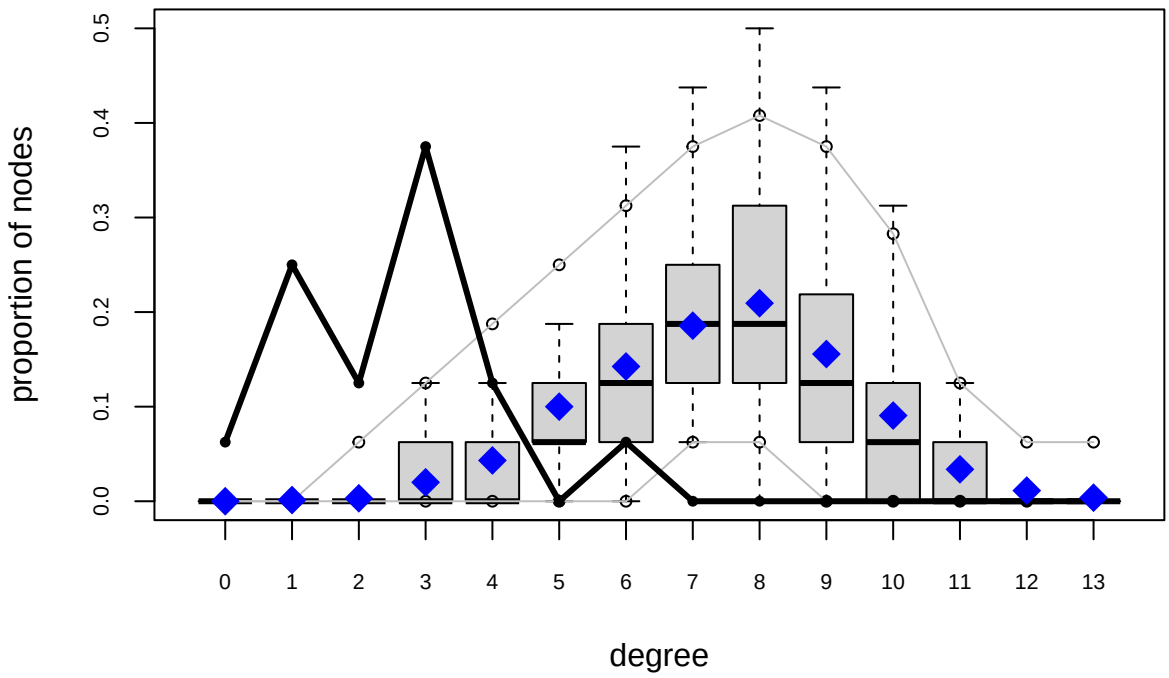
```
## esp0   12   0  1.22   7       0.00
## esp1    7   0  5.30  13       0.64
## esp2    1   0 10.75  22       0.02
## esp3    0   1 14.53  27       0.00
## esp4    0   3 12.96  29       0.00
## esp5    0   0  8.21  19       0.04
## esp6    0   0  4.42  25       0.32
## esp7    0   0  1.47  21       1.00
## esp8    0   0  0.41   8       1.00
## esp9    0   0  0.15   4       1.00
## esp10   0   0  0.03   3       1.00
##
## Goodness-of-fit for minimum geodesic distance
##
##     obs min  mean max MC p-value
## 1    20  43 59.45  82          0
## 2    35  38 59.19  69          0
## 3    32   0  1.36  14          0
## 4    15   0  0.00   0          0
## 5     3   0  0.00   0          0
## Inf  15   0  0.00   0          0
##
## Goodness-of-fit for model statistics
##
##                 obs  min    mean  max MC p-value
## edges            20   43   59.45   82          0
## triangle          3   28   68.34  175          0
## nodecov.wealth 2168 3598 5059.41 7321          0
```

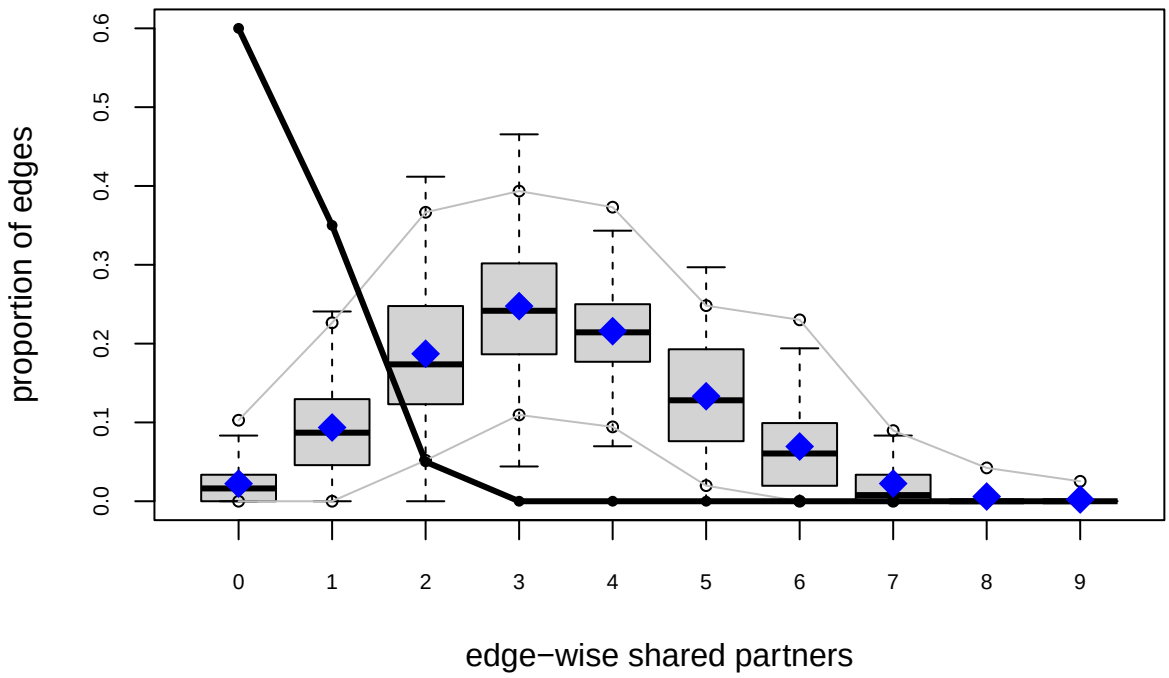
Nota: En la parte donde dice esp0, esp1, etc. las primeras dos columnas muestran la cantidad de pares de nodos que comparten 0, 1, 2, ... enlaces en común en la red observada (primera columna) y en las redes simuladas (segunda columna). Respecto del p-value, un valor bajo 0.05 indica que la red real es diferente a las redes aleatorias.

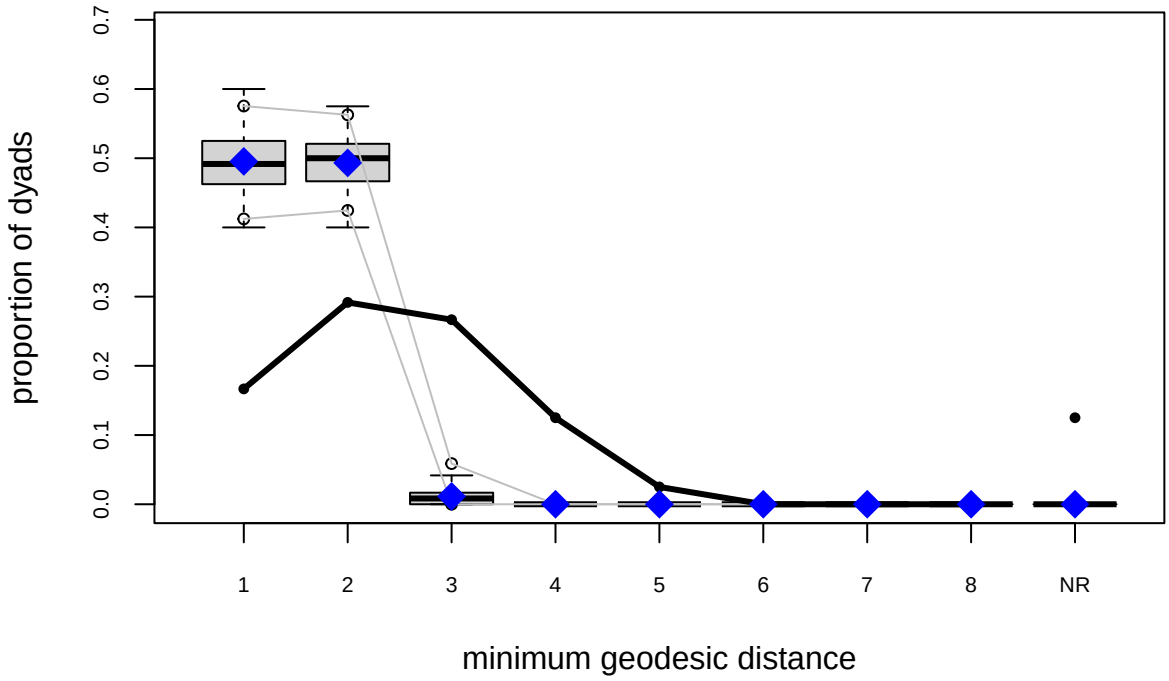
Estos resultados sugieren que el modelo ERGM actual no se ajusta bien a los datos de la red Medici.

```
plot(ergm_model_wealth.gof.degree)
```


proportion of statistics
0.0
0.2
0.4
0.6
0.8
1.0
edges
triangle
nodecov.wealth
model statistics


proportion of nodes
0.0
0.1
0.2
0.3
0.4
0.5
0
1
2
3
4
5
6
7
8
9
10
11
12
13
degree


proportion of edges
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
1
2
3
4
5
6
7
8
9
edge−wise shared partners


Goodness−of−fit diagnostics
proportion of dyads
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
1
2
3
4
5
6
7
8
NR
minimum geodesic distance

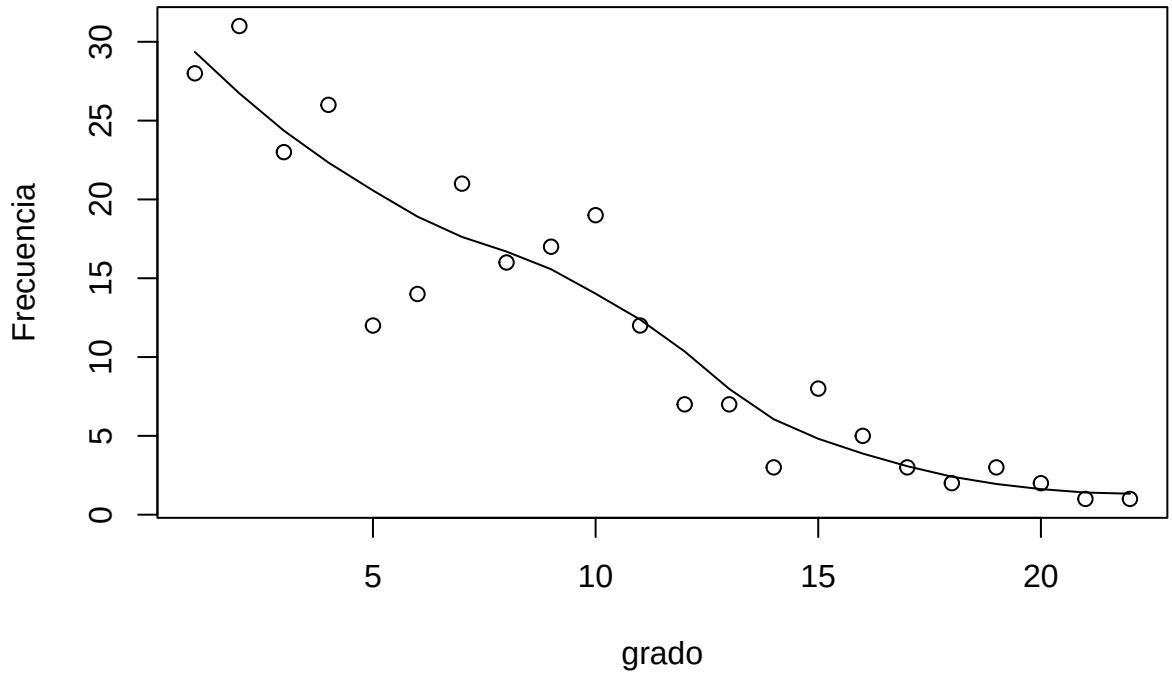
Cómo saber si una red es powerlaw o no? (o fallar en el intento)

El segundo tema que revisaremos en esta sesión es testear la existencia o no de una distribución de potencias. Como vimos en sesiones anteriores, un rasgo central de las redes reales es que parecen responder a un fenómeno libre de escalas y ello se ha observado en redes en fenómenos naturales, sociales y tecnológicos por igual. Alberto Barabasi ha sido central en el desarrollo y comprensión de este fenómeno.

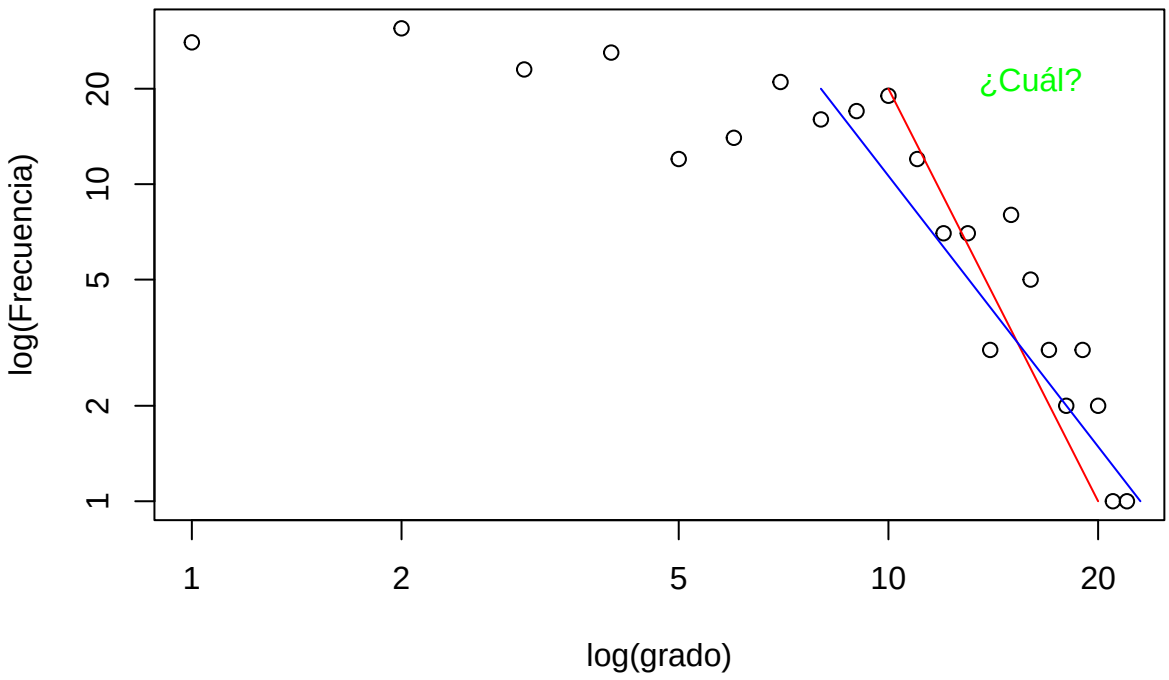
No obstante, en rigor, que una distribución pueda ser explicada o parecer una distribución de potencias no significa que necesariamente lo sea. Otras distribuciones eventualmente podrían explicar el mismo fenómeno que se está observando en una red real. Como vimos en clases pasadas, Aaron Clauset (*aqui* y *aqui*) ha mostrado evidencia que cuestiona la universalidad de las distribuciones de potencia. Ello podría ser anecdótico, excepto por el hecho que su evidencia muestra diferencias en la presencia de fenómenos libres de escala entre las redes de origen social, respecto de las de origen natural o tecnológico. Como ésta es una materia todavía en debate, es útil revisarla con mayor detención.

Para ello vamos a utilizar el paquete *poweRlaw* de Colin Gillespie (más *aquí*) y lo aplicaremos sobre la red de interlocking que vimos anteriormente. Recordemos la distribución de grado que teníamos.

```
##  [1] 28 31 23 26 12 14 21 16 17 19 12  7  7  3  8  5  3  2  3  2  1  1
```

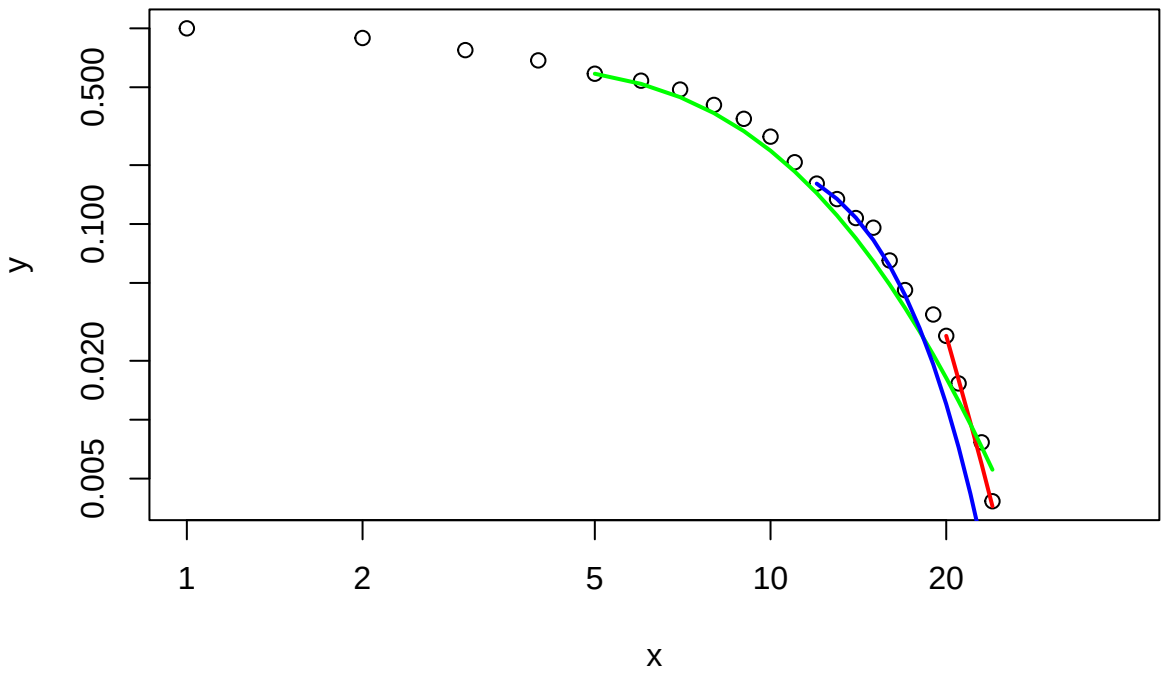


Si llevamos esa distribución a escala logarítmica obtenemos lo siguiente:



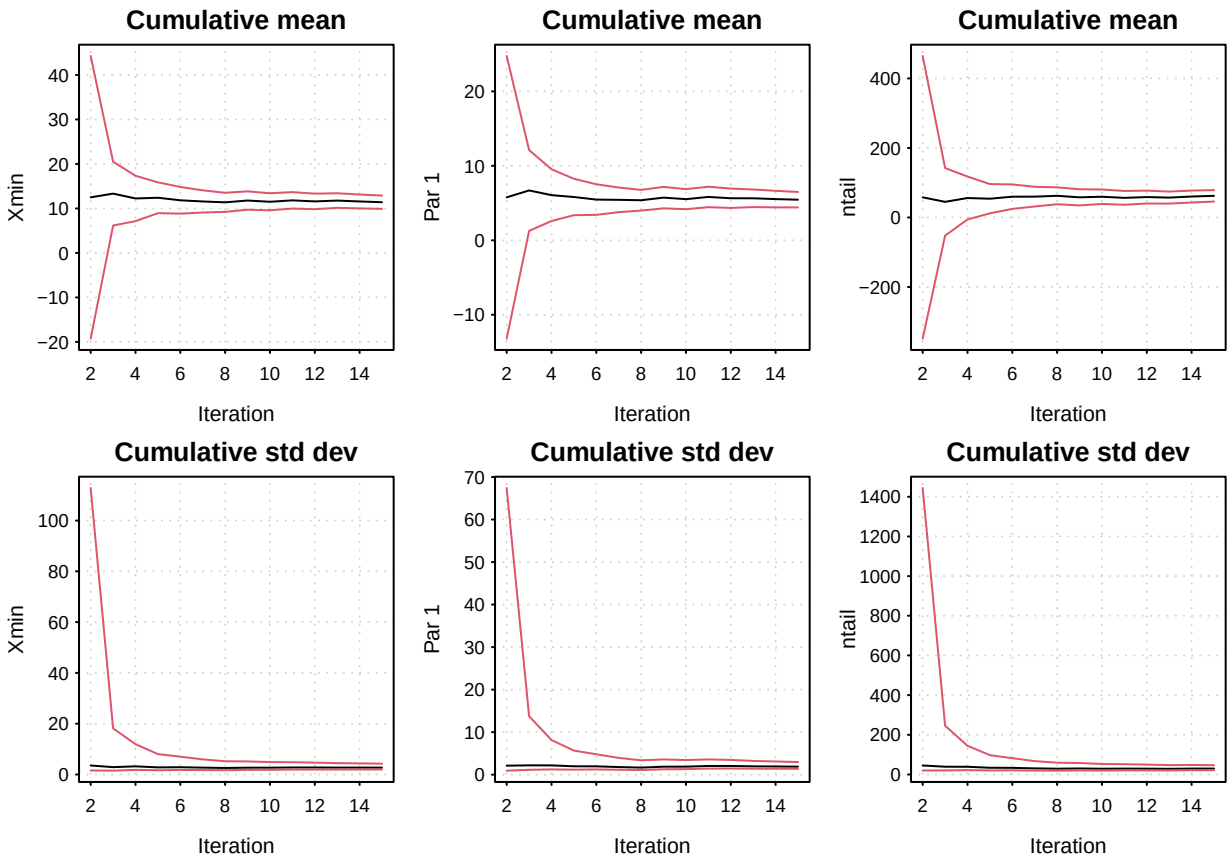
Consideremos tres posibles candidatos para explicar esa distribución: (a) powerlaw, (b) log-normal y (c) poisson.

```
red.e_pl <- displ$new(degree(red.e)) # discrete power law - displ - ajustamos una distribucion power la
est.e_pl <- estimate_xmin(red.e_pl) # estimamos el valor inferior de la distribucion
red.e_pl$setXmin(est.e_pl) # actualizamos el objeto red.e_pl con la estimacion del minimo

red.e_ln <- dislnorm$new(degree(red.e)) # ajustamos una distribucion log normal - dislnorm
est.e_ln <- estimate_xmin(red.e_ln)
red.e_ln$setXmin(est.e_ln)

red.e_po <- dispois$new(degree(red.e)) # ajustamos una distribucion poisson - dispois
est.e_po <- estimate_xmin(red.e_po)
red.e_po$setXmin(est.e_po)

plot(red.e_pl, xlim=c(1,40))
lines(red.e_pl, col="red", lwd=2)
lines(red.e_ln, col="green", lwd=2)
lines(red.e_po, col="blue", lwd=2)
```



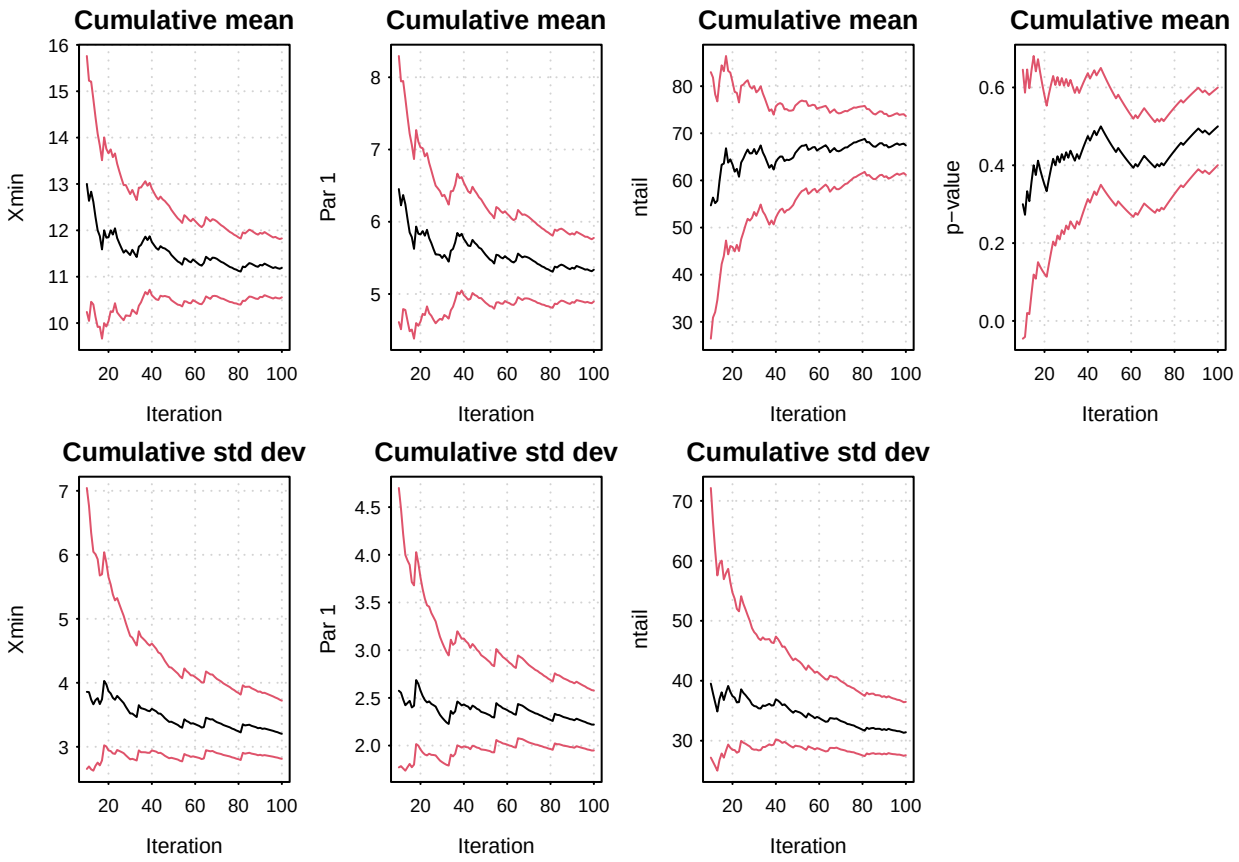
Supongamos que por inspección visual descartamos la Poisson, pero no las otras dos. Como necesitamos más información para decidir, generamos redes simuladas con similar distribución de grado

```
bs <- bootstrap(red.e_pl, no_of_sims=15, threads=2) # 50 es muy poco en una investigacion
plot(bs, trim=0.1) # trim indica el porcentaje de la muestra que no se despliega
```



Podemos ahora testear si los datos pueden explicarse por una power-law bajo la Ho: "Si, los datos se explican por una PowerLaw". Pero tenemos un solo dato (una red). Ante eso, lo que sugiere Clauset et al (2009) es generar vía boostraping miles de copias similares al original. El parámetro que nos interesa es el que queda guardado en el objeto "p". Si es menor que 0.1. Clauset et al (2009) sugiere que se puede descartar. En este caso no se puede descartar.

```
bs_p <- bootstrap_p(red.e_pl)
plot(bs_p)
```




```
bs_p$p  # si es cercano a cero, podemos estar seguros que no es una power law
```

```
## [1] 0.5
```

```
red.e_ln$setXmin(red.e_pl$getXmin())
est.ln <- estimate_pars(red.e_ln)
red.e_ln$setPars(est.ln)
```

Ahora, podria ser el caso que si bien no se descarta, de todos modos hay otra distribucion que se ajusta mejor a los datos. En este caso, tal distribución sería la log-normal.

Para comparar ambas, fijamos el mismo punto de partida para ambas y calculamos los parametros de la log normal para ese nuevo punto comun. Si no es cercano de cero, el test no nos permite descartar ninguna de las dos distribuciones (p-values lejanos de cero). En este caso, no nos permite descartar ninguna.

```
comparacion <-  compare_distributions(red.e_pl, red.e_ln) # Ho: son indistinguibles
comparacion$p_one_sided
```

```
## [1] 0.6560835
```

Para más detalles y ejemplos ver el siguiente *artículo*