# Modulo6_formacion_redes

Jorge Fábrega

Version 2023

En este módulo complementamos lo visto sobre formación de redes con ejemplos prácticos para el análisis de redes y su comparación con modelos nulos.

Partamos viendo una red aleatoria generada a partir de una distribución conocida y una matriz de adyacencia. En este caso, usaremos una distribución binomial. La expresión formal de la distribución binomial es la siguiente:

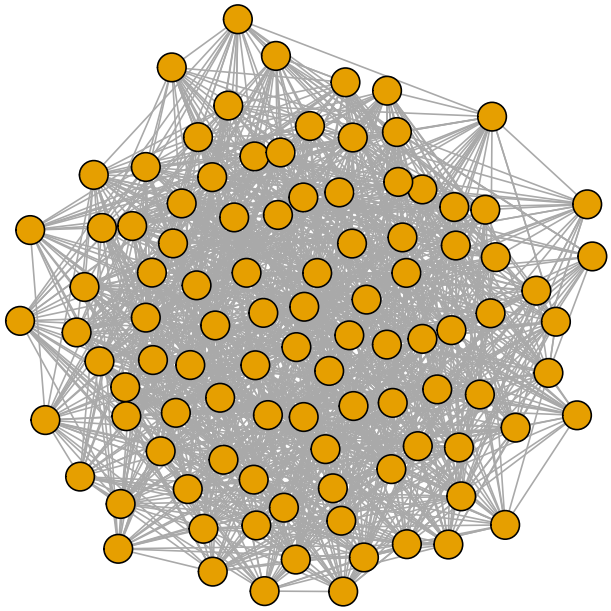
$$P(X = k) = \binom{L}{k} p^k (1-p)^{L-k}$$

donde:
- $P(X = k)$ es la probabilidad de tener exactamente $k$ enlaces.
- $L$ es el número total de posibles enlaces en la red. Para una red con $n$ nodos, $L = \frac{n(n-1)}{2}$ en una red no dirigida (sin bucles).
- $\binom{L}{k}$ es el coeficiente binomial, que indica el número de maneras de elegir $k$ enlaces de $L$ posibles.
- $p$ es la probabilidad de que un enlace específico exista.
- $k$ es el número de enlaces reales en la red.
- $L - k$ es el número de enlaces no realizados.

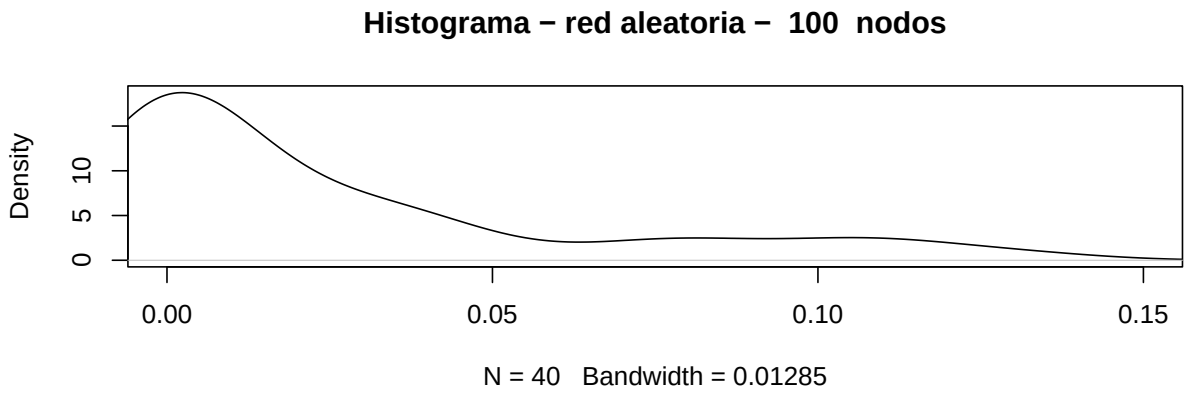
A partir de esta fórmula, generaremos una red en R usando el siguiente código:

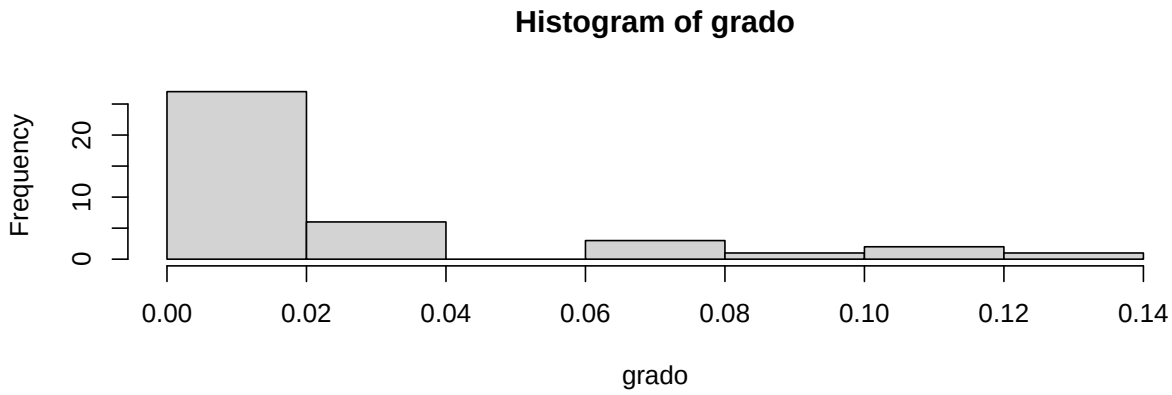
```
set.seed(1234)

a <- 100 # jugar con distintos numeros y repetir para mostrar concepto G(n,p)
prob <- 0.15
matriz <- rbinom(size = 1, n = a ^ 2,   p = prob) %>% matrix(., nrow = a, ncol = a)
red <- graph_from_adjacency_matrix(matriz)
red <- simplify(red, remove.loops = T)

# Nota: size se refiere al número de ensayos. Por lo tanto, los n casos son cada
# uno resultado de un único ensayo.
```

Veamos la red que se generó:



Y la distribución de grado:




Si repetimos el ejercicio con redes de distinto tamaño vamos a ver patrones similares, pero mientras más grande sea la red, más lento será procesar su matriz de adyacencia (naturalmente, dependiendo de la capacidad del computador donde se haga).

Como vimos anteriormente, al aumentar el número de nodos la distribución binomial converge a la distribución de Poisson.

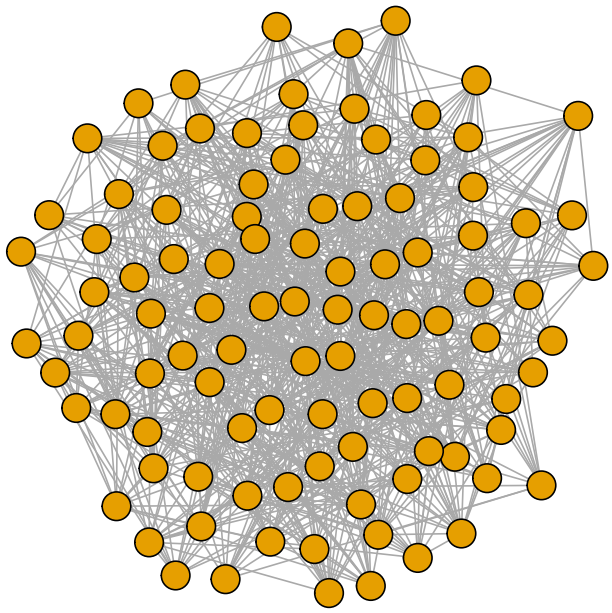
[Nota: en rigor, ello sucede cuando el número de nodos (n) tiende a infinito y la probabilidad de un link entre el nodo $i$ y $j$ ($p_{ij}$) es pequeño, de tal manera que el producto $np$ que es el parámetro de la distribución de Poisson permanece constante].

Por otro lado, por el Teorema del Límite Central cuando la media y varianza de una distribución Poisson es suficientemente mayor que 1, entonces converge a una distribución normal. Por lo tanto, a partir de una binomial para determinar qué nodos están conectados entre sí, llegamos a una distribución normal en la distribución de los grados de una red.

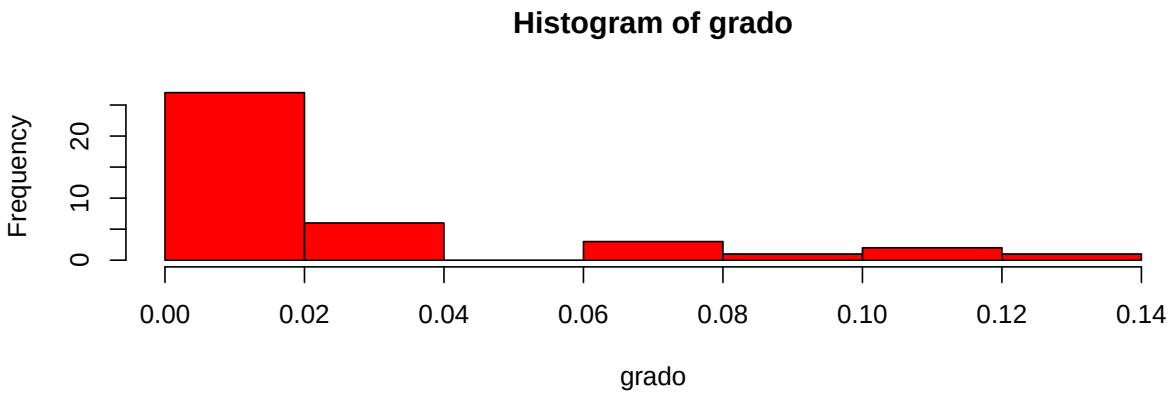
Repitamos el ejercicio anterior, pero esta vez usando igraph desde el inicio para crear una red aleatoria usamos la función *erdos.renyi.game()* para crear una red aleatoria:

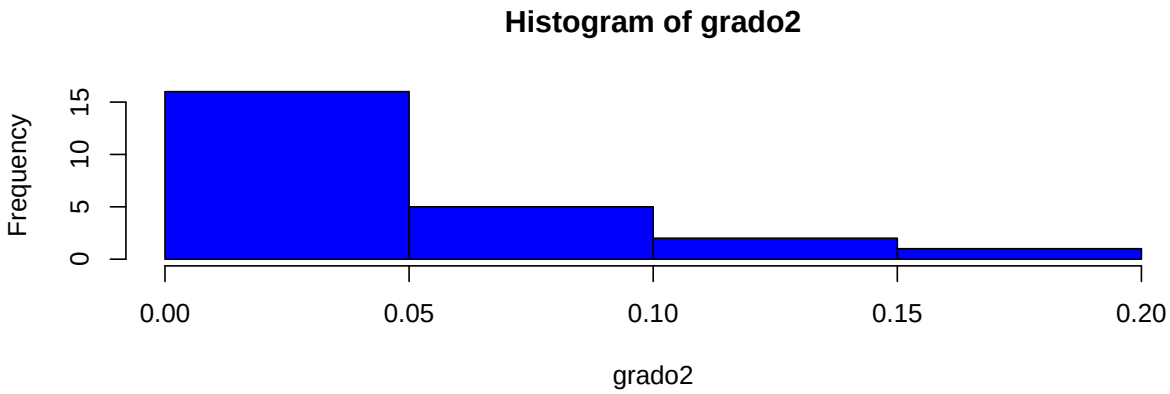
```
red2 <- erdos.renyi.game(a, prob, type = "gnp", loops = F)
# excluimos loops para hacerla comparable con la anterior
l.fr2 <- layout_with_fr(red) # mantenemos el layout de la red original
plot(red2,
     edge.arrow.size=.001,
     vertex.label=NA,
     vertex.size=10,
```

```
     layout=l.fr2)
```



```
grado2 <- degree_distribution(red2)

par(mfrow=c(2,1))
hist(grado, col = "red")
hist(grado2, col="blue")
```


Histogram of grado
Frequency
0
10
20
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
grado


Histogram of grado2
Frequency
0
5
10
15
0.00
0.05
0.10
0.15
0.20
grado2

**Creando comparaciones con modelos nulos**
Ahora, veamos la comparacion de una red basada en datos reales con una red aleatoria.

Esta comparación es el inicio de un método que nos permitirá hacer afirmaciones sobre la red real: por ejemplo, si tenemos una red y vemos que el nodo $z$ es muy central en ella, cabe preguntar si lo es por alguna dinámica especial en esa red o simplemente es un asunto del azar. Ese tipo de preguntas y las hipótesis que las acompañen van a poder ser testeadas con este tipo de aproximación.

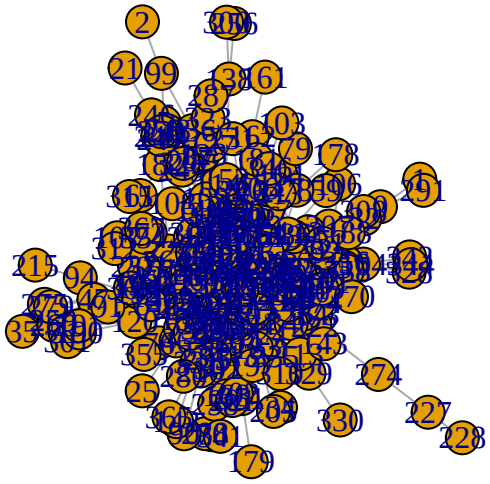
Para ello, debemos darle alguna estructura a la red aleatoria. La idea es la siguiente: sobre la red real no sabemos cómo se estructuró (tenemos hipótesis, pero no certezas), en cambio, el mecanismo que dio origen a la red aleatoria lo conocemos. Por lo tanto, podemos ajustar ese mecanismos para que en algún aspecto central se parezca a la red real. Anclado en esa igualdad, podemos entonces ver si ambas redes se parecen o no en alguna otra propiedad distinta de la red. De ser así, entonces no podríamos descartar que las conexiones de la red real sean causadas por puro azar.

Vamos a usar una red de interlocking de empresas. Aquí, dos empresas aparecen conectadas entre sí si tienen miembros en sus directorios que están en ambas.

```
archivo <- paste0(camino,"/docs/red_empresas.igraph")
red.e <- read_graph(archivo, format="ncol")

# nos quedamos con el componente gigante
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))

giant.component <- function(graph) {
  cl <- clusters(graph)
  induced.subgraph(graph, which(cl$membership == which.max(cl$csize)))
}

red.e
```

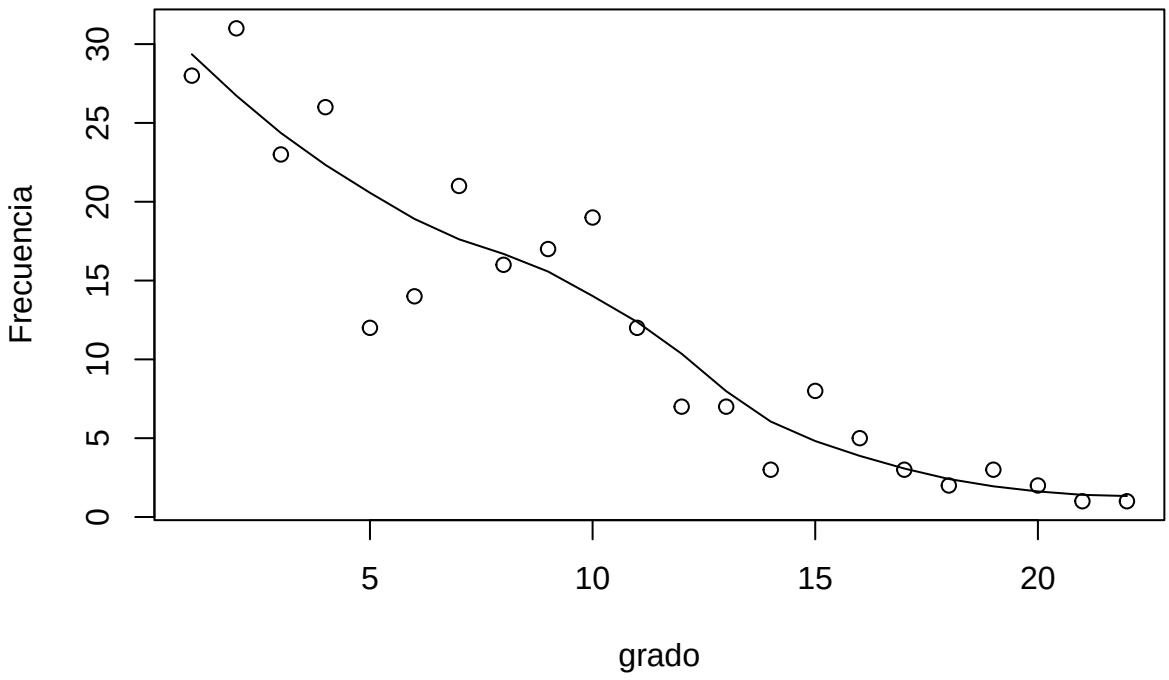
```
## IGRAPH c0f467e UN-- 261 906 --
## + attr: name (v/c)
## + edges from c0f467e (vertex names):
##  [1] 0  --1   0  --32  0  --52  0  --291 0  --307 0  --308 2  --99  3  --74
##  [9] 3  --134 3  --151 3  --173 3  --176 3  --225 3  --232 3  --297 3  --298
## [17] 3  --299 3  --334 3  --362 4  --89  4  --158 4  --172 4  --221 4  --222
## [25] 4  --233 4  --292 4  --329 334--4   4  --335 99 --5   5  --182 5  --207
## [33] 5  --244 5  --245 5  --246 5  --247 5  --248 5  --249 5  --250 7  --141
## [41] 7  --163 7  --214 7  --266 7  --314 7  --333 7  --337 52 --12  12 --147
## [49] 297--12  307--12  13 --36  13 --216 13 --224 14 --27  14 --33  14 --76
## [57] 14 --94  14 --238 14 --284 14 --296 15 --22  15 --29  15 --54  15 --80
## + ... omitted several edges
```

```
red.e.degree <- degree(red.e)
plot(red.e)
```



Veamos la distribución de grado de esta red:

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



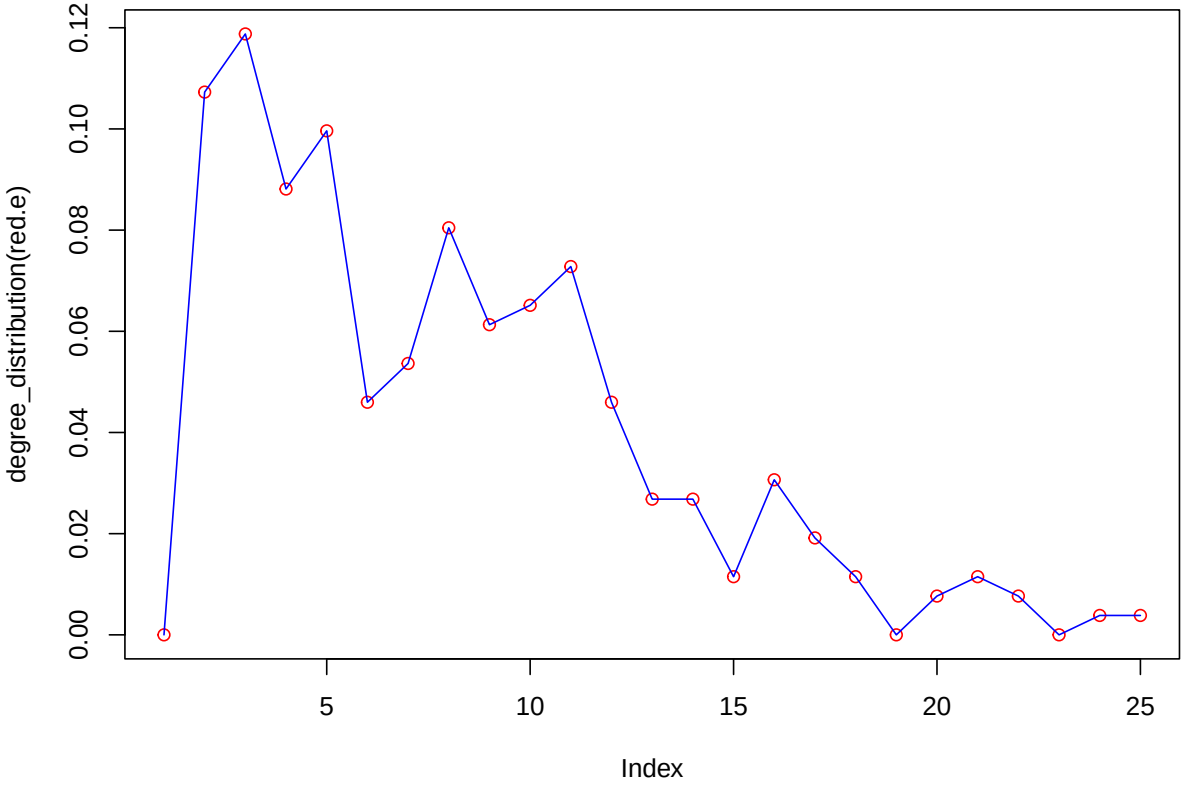

Veamos ahora otras propiedades de la red: Distancia promedio y coeficiente de clustering:

```
## [1] "Distancia promedio: 3.742"
```

```
## [1] "Clustering: 0.401"
```

Ahora que tenemos algunas métricas de la red real, veamos una red aleatoria con la cual compararla. Primero asignamos a la red aleatoria, la misma distribución de grado de la red real:

```
# comparemos la distribucion de grado con una red aleatoria
# creamos una red con la misma distribucion que red.e pero aleatoria

red.e.random <- sample_degseq(out.deg = red.e.degree, method = "vl")
# hay otros métodos distintos a "vl" más simples, pero no útiles en redes grandes
plot(degree_distribution(red.e), col="red", type="p")
lines(degree_distribution(red.e.random), col="blue", type = "l")
```

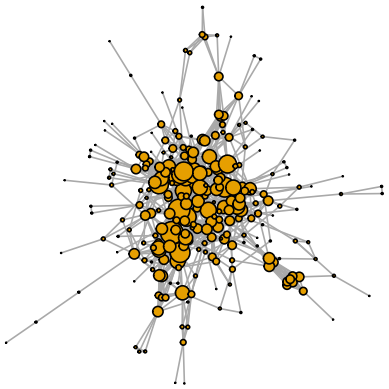



```
red.e.random
```

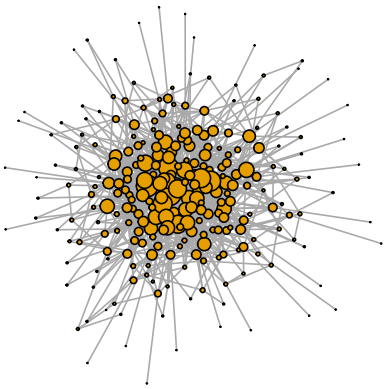
```
## IGRAPH c125b61 U--- 261 906 -- Degree sequence random graph
## + attr: name (g/c), method (g/c)
## + edges from c125b61:
##  [1]  1--228  1-- 14  1--227  1--154  1-- 40  1-- 45  2-- 89  3-- 24  3-- 27
## [10]  3--238  3-- 82  4-- 51  4--183  4--149  4-- 18  4--133  4-- 16  4-- 76
## [19]  4-- 25  4-- 64  4-- 78  4-- 28  4--176  4--130  4--193  4--126  5-- 45
## [28]  6--167  6-- 28  6-- 67  6-- 29  6--201  6--237  6-- 82  7--241  7--178
## [37]  7--169  7--207  8-- 97  9--105  9-- 83  9-- 68 10--202 10-- 98 10-- 28
## [46] 10--103 10-- 73 10-- 93 10--196 10--147 10-- 54 10-- 18 10--144 10-- 21
## [55] 11--127 11--216 11--118 11--121 11--182 11--196 12-- 74 12-- 15 12-- 42
## [64] 12-- 83 12-- 95 12--223 12--202 12--152 12--110 12-- 19 13--168 13-- 66
## + ... omitted several edges
```

Comparemos primero visualmente ambas redes:

**Red Real**



**Red Aleatoria**



```
## [[1]]
## IGRAPH c0f467e UN-- 261 906 --
## + attr: name (v/c)
## + edges from c0f467e (vertex names):
##  [1] 0  --1   0  --32  0  --52  0  --291 0  --307 0  --308 2  --99  3  --74
##  [9] 3  --134 3  --151 3  --173 3  --176 3  --225 3  --232 3  --297 3  --298
## [17] 3  --299 3  --334 3  --362 4  --89  4  --158 4  --172 4  --221 4  --222
## [25] 4  --233 4  --292 4  --329 334--4   4  --335 99 --5   5  --182 5  --207
## [33] 5  --244 5  --245 5  --246 5  --247 5  --248 5  --249 5  --250 7  --141
## [41] 7  --163 7  --214 7  --266 7  --314 7  --333 7  --337 52 --12  12 --147
## [49] 297--12  307--12  13 --36  13 --216 13 --224 14 --27  14 --33  14 --76
## [57] 14 --94  14 --238 14 --284 14 --296 15 --22  15 --29  15 --54  15 --80
## + ... omitted several edges
##
## [[2]]
## IGRAPH c125b61 U--- 261 906 -- Degree sequence random graph
## + attr: name (g/c), method (g/c)
## + edges from c125b61:
##  [1]  1--228  1-- 14  1--227  1--154  1-- 40  1-- 45  2-- 89  3-- 24  3-- 27
## [10]  3--238  3-- 82  4-- 51  4--183  4--149  4-- 18  4--133  4-- 16  4-- 76
## [19]  4-- 25  4-- 64  4-- 78  4-- 28  4--176  4--130  4--193  4--126  5-- 45
## [28]  6--167  6-- 28  6-- 67  6-- 29  6--201  6--237  6-- 82  7--241  7--178
## [37]  7--169  7--207  8-- 97  9--105  9-- 83  9-- 68 10--202 10-- 98 10-- 28
## [46] 10--103 10-- 73 10-- 93 10--196 10--147 10-- 54 10-- 18 10--144 10-- 21
```

```
## [55] 11--127 11--216 11--118 11--121 11--182 11--196 12-- 74 12-- 15 12-- 42
## [64] 12-- 83 12-- 95 12--223 12--202 12--152 12--110 12-- 19 13--168 13-- 66
## + ... omitted several edges
```

Consideremos ahora otras métricas para comparar ambas redes: diametro, densidad, conteo de tríadas (para ver la clasificación en igraph hacer click *aqui*) y coeficiente de clusterización entre la red original y una red aleatoria generada con la misma distribución de grado:

```r
# Función para calcular métricas de red
calculate_network_metrics <- function(network) {
  metrics <- list()
  metrics$diameter <- diameter(network)
  metrics$density <- round(graph.density(network),4)
  metrics$triad_count <- triad_census(network)[16]
  metrics$clustering_coefficient <- round(transitivity(network, type="global"),4)
  return(metrics)
}

# Función para comparar métricas entre dos redes
compare_network_metrics <- function(network1, network2) {
  metrics1 <- calculate_network_metrics(network1)
  metrics2 <- calculate_network_metrics(network2)
  comparison_table <- rbind(metrics1, metrics2)
  rownames(comparison_table) <- c("Red Real", "Red Aleatoria")
  return(comparison_table)
}

# Uso de la función
network_comparison <- compare_network_metrics(red.e, red.e.random)
print(network_comparison)
```

```
##               diameter density triad_count clustering_coefficient
## Red Real      9        0.0267  1148        0.4008
## Red Aleatoria 6        0.0267  135         0.0471
```

## Comparemos ahora la red real con distintos mecanismos de formación de una red

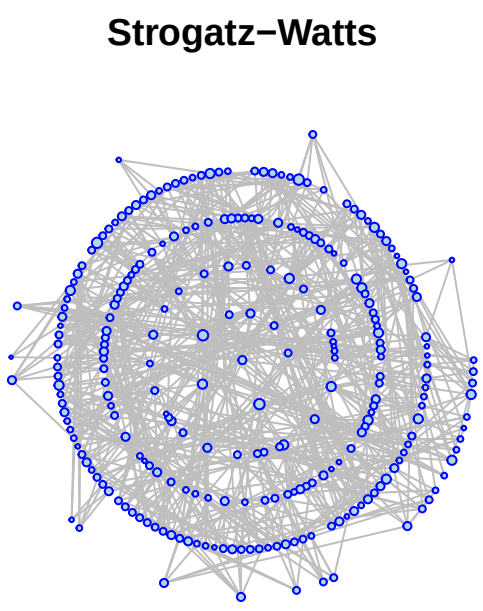
### Red de mundos pequeños

Veamos la versión de mundos pequeños (Strogatz-Watts) de nuestra red. Para ello usamos la función *watts.strogatz.game* en igraph.

```
size <- length(V(red.e))
sm <- watts.strogatz.game(1,size,3,0.1)
# la red es unidimensional (1)
# tiene tantos nodos como la red real (size)
# estoy asumiendo vecindarios de 3 nodos (en la red regular de inicio)
# y rewiring de 0.1 (se puede mejorar la precisión)
```

Veamos la red generada:

```
sm.grado <- degree(sm)
plot(sm,
     main="Strogatz-Watts",
     vertex.color="light blue",
     layout=layout.reingold.tilford(sm, circular=T),
     edge.color="grey",
     edge.width=E(sm)$weight/10,
     edge.arrow.size=0.1,
     vertex.size=sm.grado/2,
     vertex.frame.color="blue",
     vertex.label=NA)
```



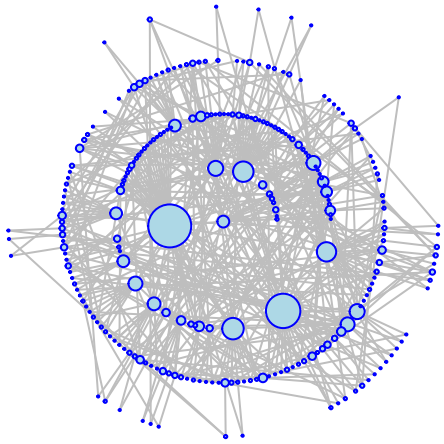
```
sm.distancia <- round(mean_distance(sm),3)
sm.clustering <- round(transitivity(sm, type="global"),3)
```

### Red de preferential attachment

Ahora generemos la versión preferential attachment de la red (Barabasi) usando la función *barabasi.game* en igraph:

```
red.pa <- barabasi.game(size,power=1, m=2, directed=F, algorithm="psumtree")
degree.red <- degree(red.pa)
l <- layout.reingold.tilford(red.pa, circular=T)
```

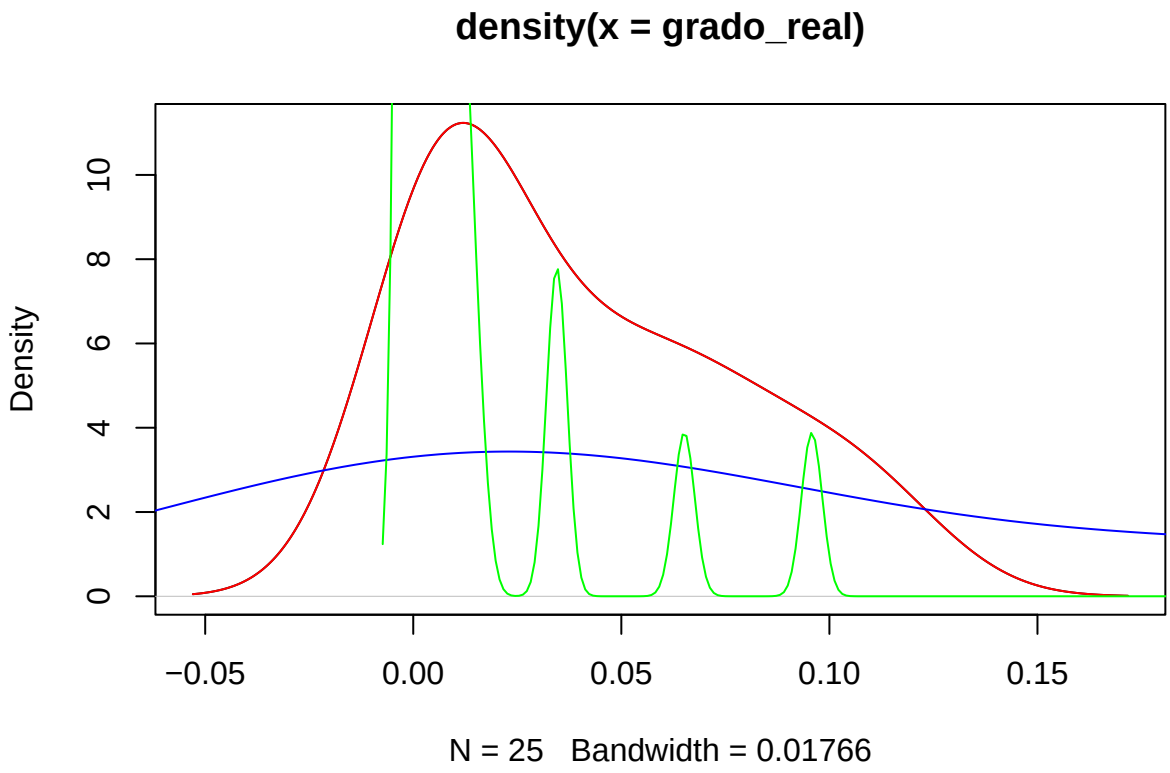
```
plot(red.pa,
     main="Red - Pref. Attachment",
     vertex.color="light blue",
     layout=layout.reingold.tilford(red.pa, circular=T),
     edge.color="grey",
     edge.width=E(red.pa)$weight/10,
     edge.arrow.size=0.1,
     vertex.size=degree.red/2,
     vertex.frame.color="blue",
     vertex.label=NA)
```




```
red.pa.distancia <- round(mean_distance(red.pa),3)
red.pa.clustering <- round(transitivity(red.pa, type="global"),3)
```

Veamos ahora las distribuciones de grado de las cuatro redes que tenemos

```r
grado_real <- degree_distribution(red.e)
grado_random <- degree_distribution(red.e.random)
grado_sm <- degree_distribution(sm)
grado_pa <- degree_distribution(red.pa)

plot(density(grado_real))
lines(density(grado_random), col="red")
lines(density(grado_sm), col="blue")
lines(density(grado_pa), col="green")
```




```r
#nótese que la red aleatoria tiene la misma densidad que la red real.
```

```
##      distancia promedio clustering diametro densidad tríadas
## Real              3.742      0.401        9   0.0267    1148
## ER                3.069      0.026        6   0.0267     135
## SW                3.965      0.293        7   0.0231     398
## PA                3.566      0.035        7   0.0153      43
```

Ahora bien, lo anterior compara la red real con una red de cada tipo. Un ejercicio completo lo que debe hacer es comparar con una amplia muestra de redes generadas según un mecanismo de formación de redes. De modo tal de obtener distribuciones que puedan ser usadas para hacer test estadísticos.

Para ello, tomamos alguno de los mecanismos y generamos miles de redes con ese mecanismo para después generar estadísticas a partir de ello. Dados los resultados de arriba, hagámoslo un esquema usando Small World.

```r
# Función para realizar pruebas t de una muestra
perform_t_test <- function(real_value, simulated_values) {
```

```
  if(length(simulated_values) > 1 && !is.na(real_value)) {
    t.test(x = simulated_values, mu = real_value)$p.value
  } else {
    return(NA)
  }
}

# --- Análisis de la Red Original ---
# Red original
v <- V(red.e)
size <- length(v)
# Nos aseguramos que la distribución de grados real no tenga NA
degree_distribution_real <- na.omit(degree.distribution(red.e))
avg_path_length_real <- average.path.length(red.e)

# --- Simulación de Redes y Cálculo de Métricas ---
total_redes <- 100
degree_distributions_sim <- matrix(NA, nrow=max(degree(red.e)), ncol=total_redes)
avg_path_lengths_sim <- numeric(total_redes)

for (i in 1:total_redes) {
    red <- watts.strogatz.game(1, size, 3, dens, loops=FALSE)
    red <- giant.component(red)

    # Distribución de grados
    dd <- degree.distribution(red)
    degree_distributions_sim[1:length(dd), i] <- dd

    # Longitud promedio del camino
    avg_path_lengths_sim[i] <- average.path.length(red)
}
```

Como tenemos un solo dato de la red real y muchos de la red aleatoria, podemos seguir dos caminos. El estadísticamente más apropiado es hacer una prueba T entre el valor de la muestra y el valor aleatorio (en particular porque es una muestra pequeña). Un camino alternativo (preferible cuando hay muchas simulaciones) es comparar el valor de la red real con la distribución de valores obtenidos de las redes aleatorias. En este caso se puede considerar como criterio conservador si el valor obtenido de la red real está a 3 o más desviaciones estándar de la media de los valores simulados. Esto significaría que está entre el 0.3% más alejado del valor medio simulado.

```
# --- Pruebas Estadísticas ---
# Prueba T de una muestra para Longitud Promedio del Camino
t_test_results <- sapply(avg_path_lengths_sim, function(length_sim) {
    t.test(x = c(length_sim, avg_path_length_real), mu = avg_path_length_real)$p.value
})
```

```
# --- Resultados  ---
# Crear un marco de datos para los resultados de Prueba T
results_df <- data.frame(
  Simulation = 1:length(t_test_results),
  P_Value = t_test_results
)

# Ver el marco de datos de Prueba T
head(results_df)
```

```
##   Simulation P_Value
## 1          1     0.5
## 2          2     0.5
## 3          3     0.5
## 4          4     0.5
## 5          5     0.5
## 6          6     0.5
```

```
summary(results_df$P_Value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.5     0.5     0.5     0.5     0.5     0.5
```