Complex and social networks: introduction to igraph

Lab work 1

Sergio H. Martinez Mateu & Egon Ferri

Description of the task

To produce two plots showing:

1. The clustering coefficient and the average shortest-path as a function of the parameter p of the WS model
2. The average shortest-path length as a function of the network size of the ER model

Solution

Plot 1

Here we create a couple of functions to compute the clustering coefficient (C) and the average shortest-path length (L) of a simulated WS model. We fix arbitrary values for the initial lattice parameters. By default, the size of the network is set to an arbitrary large number which yields reasonable computational time (1000).

```r
# Function for clustering coefficient
cluster_coeff <- function(x, n = 1000){
  g <- watts.strogatz.game(dim = 1, size = n, nei = 4, p = x)
  CC <- transitivity(g)
  return(CC)
}

# Function for average shortest-path length
avg_short_path <- function(x, n = 1000){
  g <- watts.strogatz.game(dim = 1, size = n, nei = 4, p = x)
  ASP <- average.path.length(g)
  return(ASP)
}
```

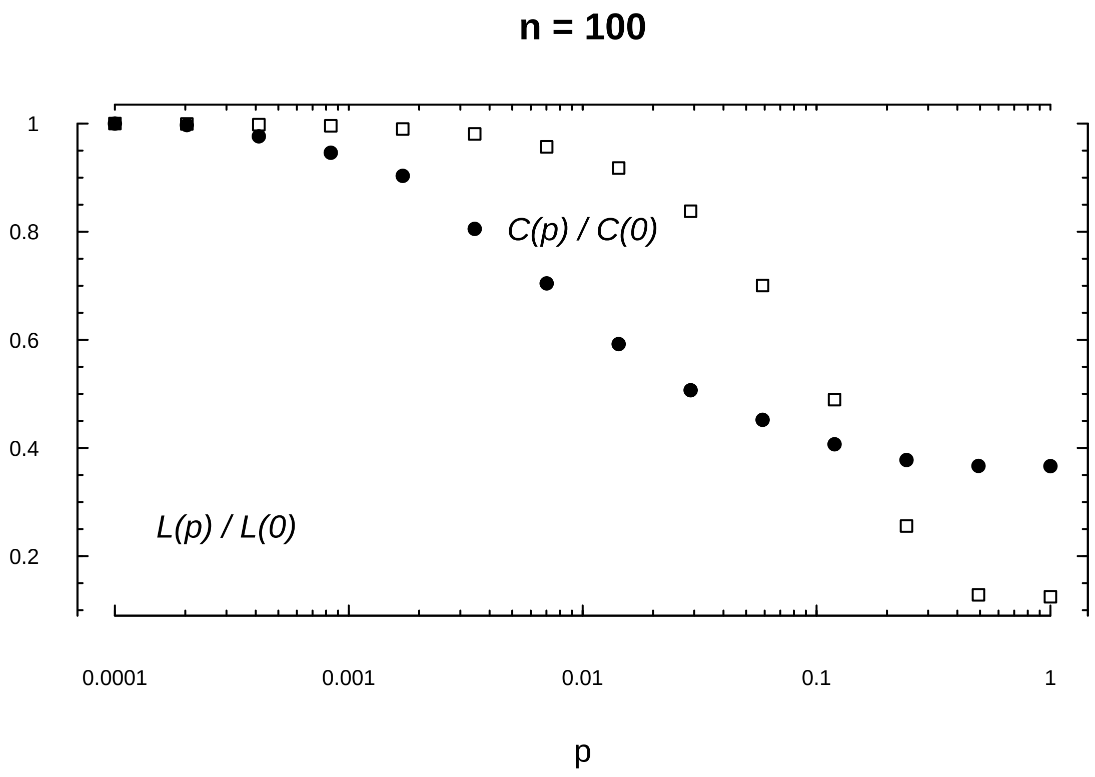
Now we set all the parameters. C and L will be computed several times (`nsim`) for each value of the probability parameter p (`p_list`). For curiosity, we decided also to repeat the exercise several times with different increasing values of n (`n_list`).

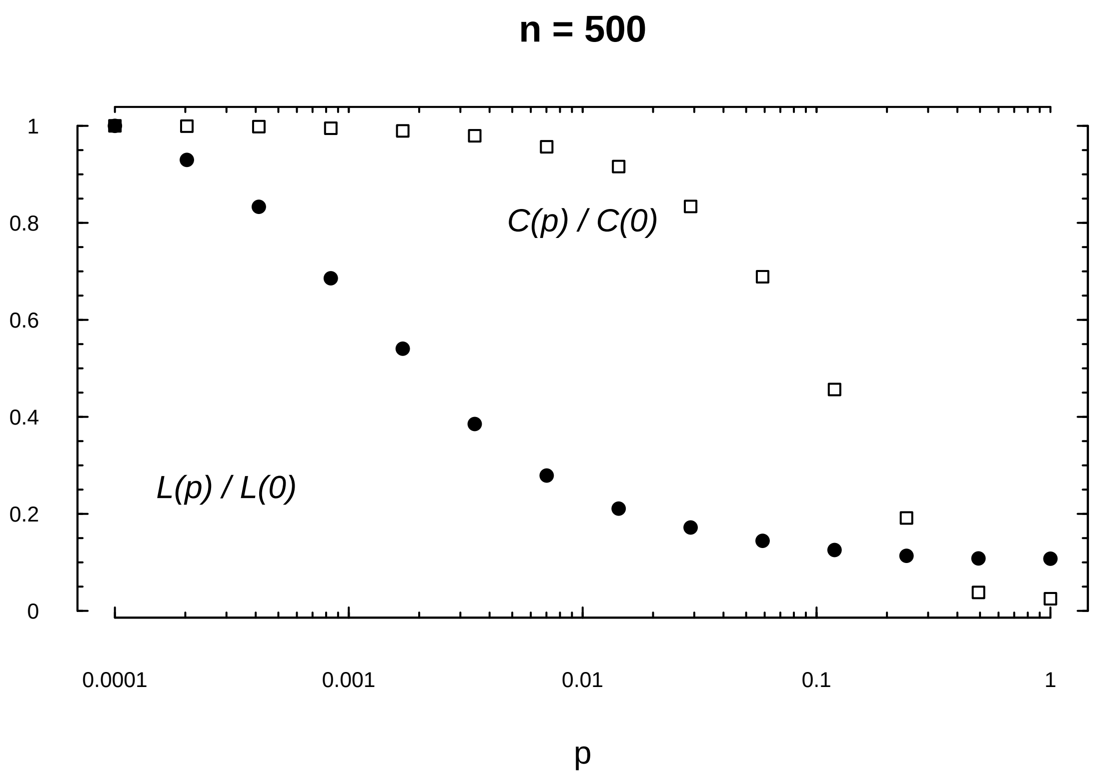
```r
# Number of simulations
nsim <- 100

# Values of the p parameter
p_list <- 0.1^(seq(4,0, length.out = 14))

# values of the n parameter
n_list <- c(100, 500, 1000, 2000)
```

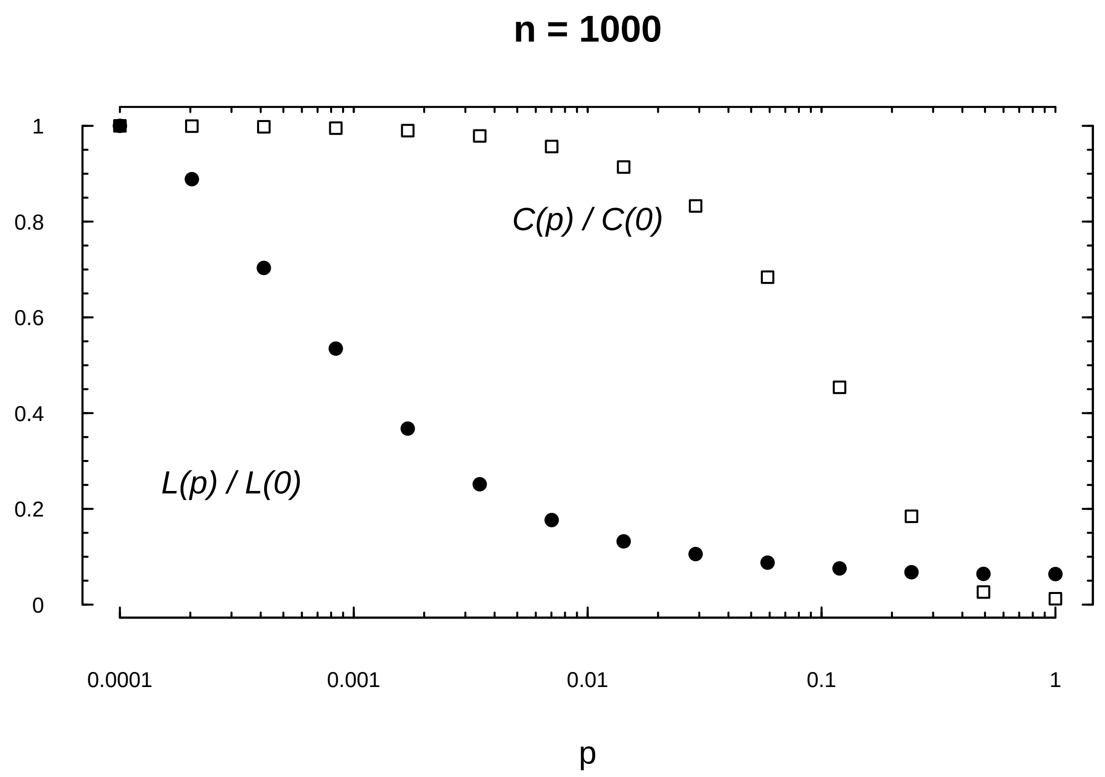
The results are the average of all the repetitions for each value of p. Finally, the results are normalized dividing by the largest value. Finally we do the plots. We tried to reproduce the same graphical parameters but we ended up with an ugly piece of code that we have moved to another script (plot1.R).

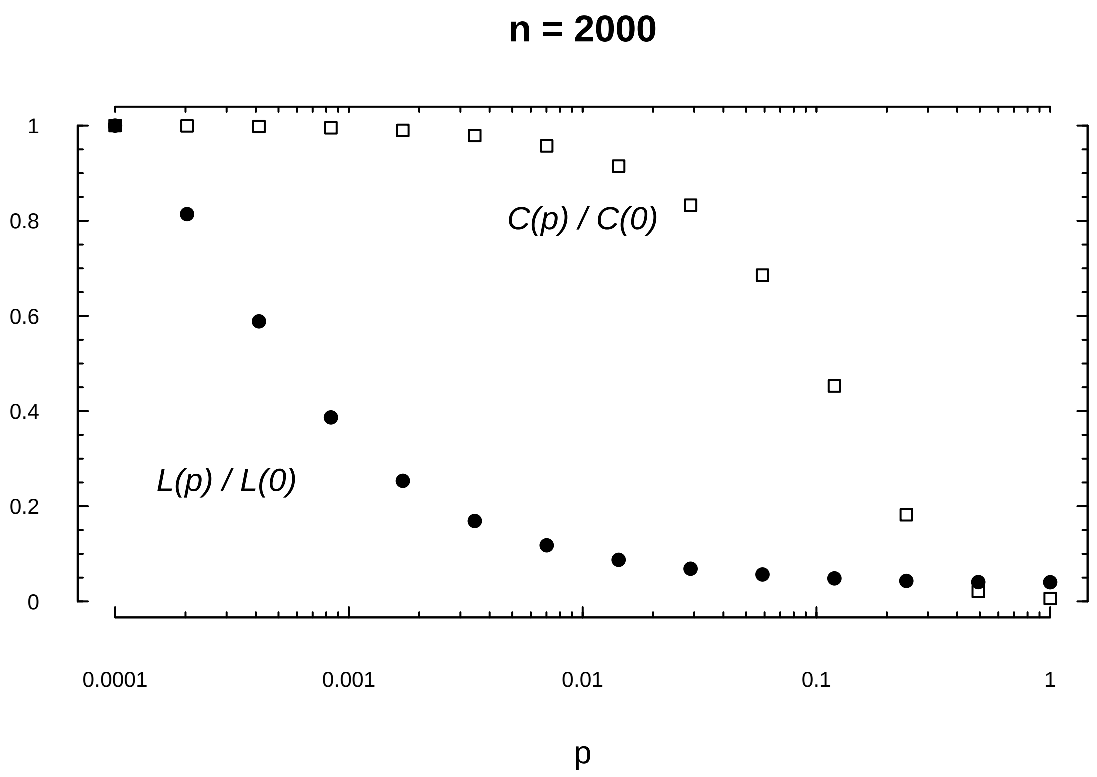
```r
source("plot1.R")
```

```
for(n in n_list){
  cc_matrix<- replicate(nsim, sapply(p_list, function(x) cluster_coeff(x, n)))
  cc_avg <- rowMeans(cc_matrix)
  cc_normalized <- cc_avg/max(cc_avg)

  asp_matrix<- replicate(nsim, sapply(p_list, function(x) avg_short_path(x, n)))
  asp_avg <- rowMeans(asp_matrix)
  asp_normalized <- asp_avg/max(asp_avg)

  plot1(cc_normalized, asp_normalized, n)
}
```




n = 500
C(p) / C(0)
L(p) / L(0)
1
0.8
0.6
0.4
0.2
0
0.0001
0.001
0.01
0.1
1
p


n = 1000
C(p) / C(0)
L(p) / L(0)
1
0.8
0.6
0.4
0.2
0
0.0001
0.001
0.01
0.1
1
p



Plot 2

In this case we create a function to obtain the average shortest-path length (ASPL) of an ER network.

```
avg_short_path_ER <- function(x, n = 1000){
  g <- erdos.renyi.game(n = n, p.or.m = x)
  ASP <- average.path.length(g)
  return(ASP)
}
```

Now we want to see how the average shortest path grows with the number of nodes in ER networks. Since ER networks that are generated with too large values of p yield trivially low average shortest path lengths (between 1 and 2), we want to choose low values of p that allow the ASPL to grow. However, ER networks generated with too low p values end up being disconnected. For this reason, we need specially chosen values of p. Using the formula indicated in the exercise statement, we know that the minimum value of p that we have to choose in order to obtain almost surely a connected network is:

$$p = (1 + \epsilon)\frac{\ln(n)}{n}$$

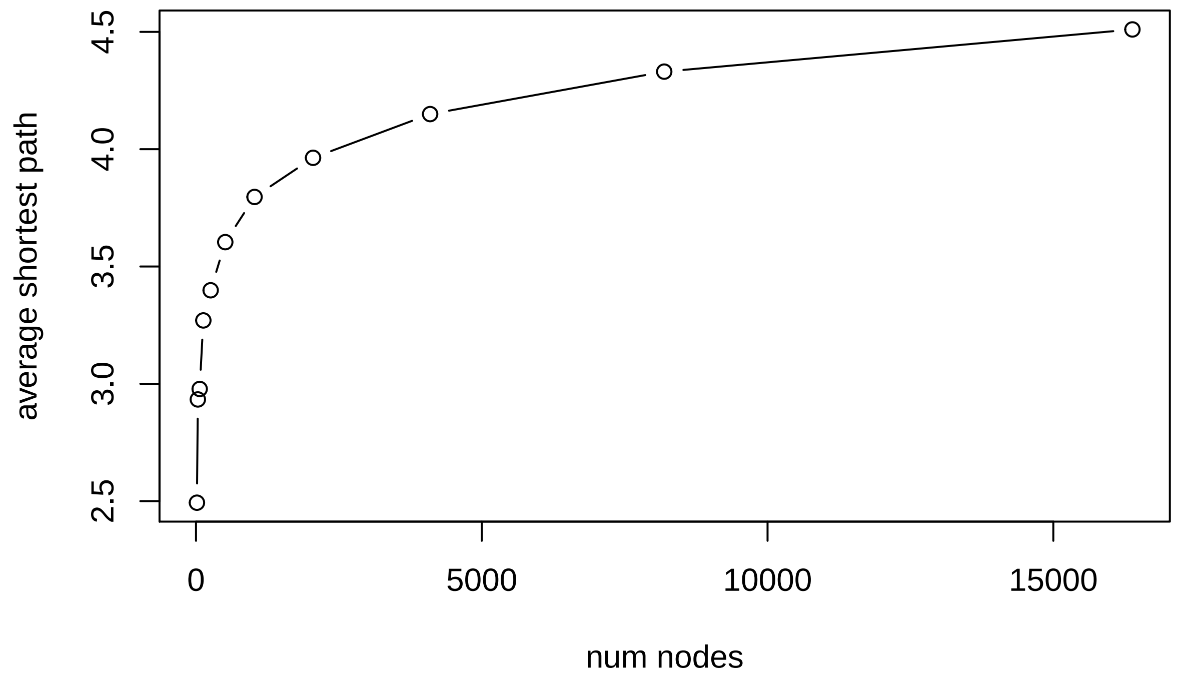
where n is the number of nodes in the network and ϵ is an arbitrary small positive constant. Thus, we will adjust the value of p according to the formula for each value of n. We tried with $\epsilon = 0.01$, which seems to work well. On the other hand, we want to get the curve for large values of n, but this is very computationally expensive for which here we have reduced the number of simulations.

```
# Number of simulations
nsim <- 10
```

```r
# Values of the n parameter
n_list <- 2^(4:14)

# Values of the p parameter
epsilon <- 0.01
p_list <- (1+epsilon)*(log(n_list))/n_list
```

Finally we compute and plot the results.

```r
asp_matrix<- replicate(nsim, sapply(1:length(n_list), function(x) avg_short_path_ER(p_list[x], n = n_li
asp_avg <- rowMeans(asp_matrix)

plot(n_list, asp_avg, xlab = "num nodes", ylab = "average shortest path", type = "b")
```


average shortest path
num nodes