Online Appendix - Short Tutorial on empirical analysis

Introduction

This file contains a simple exercise to explain the estimation counterfactual simulation procedure.

We provide this example using a smaller dataset than the one in the paper, $n = 100$ nodes. This network data has been generated through simulation, trying to mimic the features of the dataset used in the paper, because we cannot share the proprietary Venture Xpert's data. Therefore, the interpretation of the following results is just for illustrative purposes.

Setup

The researchers should install the libraries `Bergm` and `statnet`.

```
# remove any data in memory
rm(list = ls())
# install libraries used for estimation
install.packages("statnet", dependencies = TRUE)
install.packages("Bergm", dependencies = TRUE)
```

We clear the memory and load the dataset that we will use in the analysis.

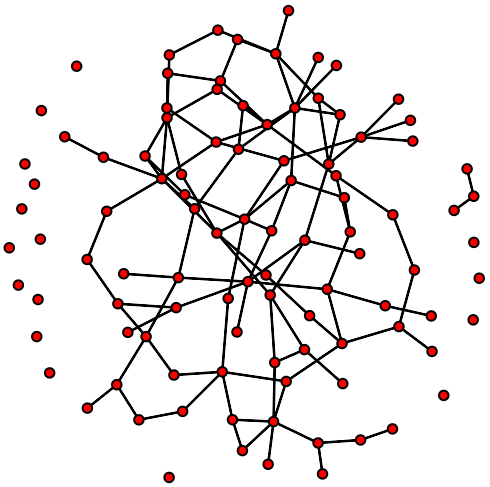
```
# remove any data in memory
rm(list = ls())
# load replication data
#setwd("/Users/Angelo/Dropbox/firmnetworksERGM/notes_sg/")
setwd("/Users/Angelo/Dropbox/firmnetworksERGM/notes_sg/")
load("data.RData")
```

To estimate the model and analyze the data we will use the `Bergm` library, so we need to load it.

```
library(Bergm)
```

The network in our data is show in the Figure below

```
plot(net)
```

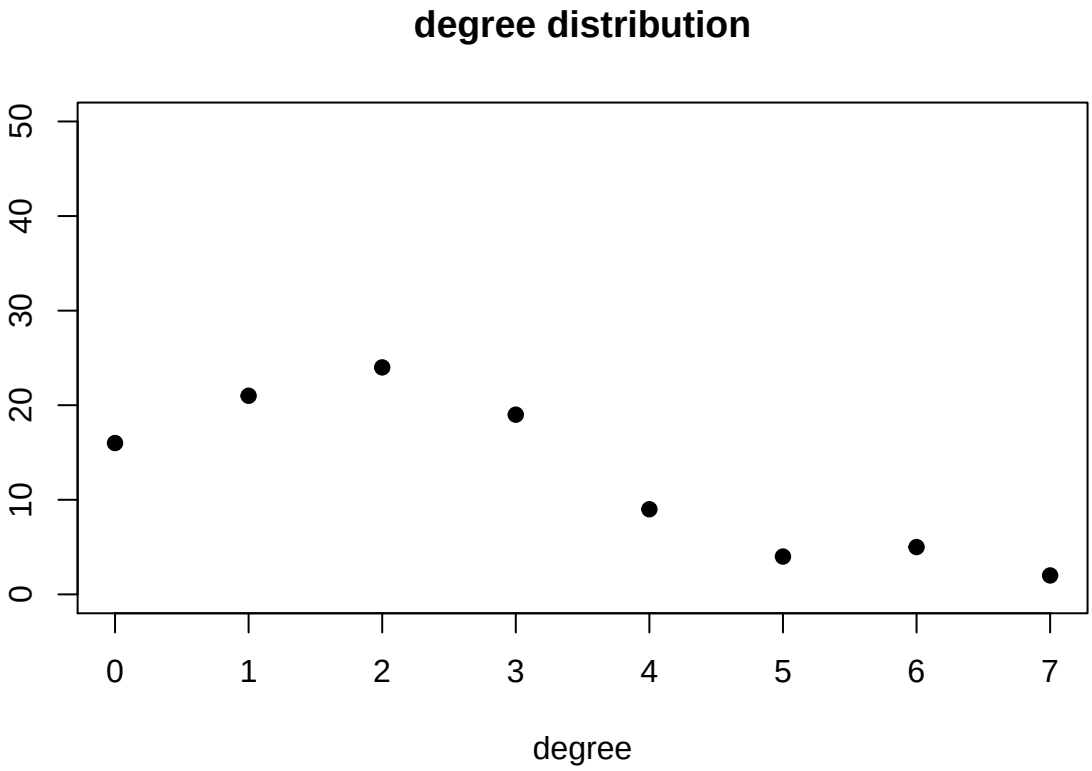


The degree distribution is shown below

```
ddd <- degreedist(net)
```

```
## degree0 degree1 degree2 degree3 degree4 degree5 degree6 degree7
##      16      21      24      19       9       4       5       2
```

```
plot(seq(0, length(ddd)-1), ddd, pch = 19,
     ylim = c(0,50), xlab = "degree", ylab = "",
     main = "degree distribution")
```



Specification, ERGMs and Estimation

The model we want to estimate is based on the payoff functions

$$U_i(g,x,\theta) = \sum_{j=1}^{n} g_{ij} \left[u(x_i,x_j;\alpha) + \beta \sum_{k\neq j,i}^{n} g_{jk} + \gamma \sum_{k\neq j,i}^{n} g_{jk}g_{ki} \right]$$

This translates into a potential function of the following form

$$Q(g,x,\theta) = \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij} \left[u(x_i,x_j;\alpha) + \frac{\beta}{2} \sum_{k\neq j,i}^{n} g_{jk} + \frac{2\gamma}{3} \sum_{k\neq j,i}^{n} g_{jk}g_{ki} \right]$$

We specify the function $u(x_i,x_j,\alpha)$ as in the main text of the paper.

$$\begin{aligned} u(x_i,x_j,\alpha) = {} & \alpha_0 + \alpha_1 \mathbf{1}_{\{firmtype_i = firmtype_j\}} && (1)\\ & + \alpha_2 \left|\log(capital_i) - \log(capital_j)\right| && (2)\\ & + \alpha_3 \left|age_i - age_j\right| + \alpha_4 \mathbf{1}_{\{state_i = state_j\}} && (3) \end{aligned}$$

where we use the notation $\mathbf{1}_{\{x_i=x_j\}}$ to denote the indicator that returns 1 if $x_i = x_j$ and 0 if $x_i \neq x_j$. In the payoff $u(x_i, x_j, \alpha)$, we interpret the parameter α_0 as the marginal cost of forming an additional link. Parameters $\alpha_1, \alpha_2, \alpha_3$ and α_4 detect homophily in preferences. The parameter α_1 is the benefit of forming a link where both firms belong to the same type; if $\alpha_1 < 0$, then firms prefer forming links to companies of a different type; on the other hand if $\alpha_1 > 0$, firms prefer forming links with companies of the same type. The parameter α_2 is the marginal benefit of forming links with companies, weighted by the difference in capital managed; if $\alpha_2 < 0$, then firms prefer forming links to companies with similar capital level; on the other hand, if $\alpha_2 > 0$, firms prefer forming links with companies with larger or lower capital.

The implied potential function for the model is

$$Q(g, x, \theta) = \alpha_0 \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij} + \alpha_1 \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}\mathbf{1}_{\{firmtype_i = firmtype_j\}} \tag{4}$$

$$+ \alpha_2 \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij} \left|\log(capital_i) - \log(capital_j)\right| \tag{5}$$

$$+ \alpha_3 \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij} \left|age_i - age_j\right| + \alpha_4 \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}\mathbf{1}_{\{state_i = state_j\}} \tag{6}$$

$$+ \frac{\beta}{2} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k\neq j,i}^{n} g_{ij}g_{jk} + \frac{2\gamma}{3} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k\neq j,i}^{n} g_{ij}g_{jk}g_{ki} \tag{7}$$

We can think of the first term $\sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}$ of the function Q, with parameter α_0 as (twice) the number of links in the network; the second term $\sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}\mathbf{1}_{\{firmtype_i = firmtype_j\}}$, corresponding to the parameter α_1 is the number of links among firms of the same type; the third term $\sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}\left|\log(capital_i) - \log(capital_j)\right|$ is the the number of links weighted by the absolute difference in (log) capital of firms i and j; the fourth term $\sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}\left|age_i - age_j\right|$ is the sum of links weighted by the absolute difference in age of the firms; the fifth term $\sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}\mathbf{1}_{\{state_i = state_j\}}$ is the number of links among firms of the same state; the sixth term $\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k\neq j,i}^{n} g_{ij}g_{jk}$ is the (endogenous) number of 2-stars; the last term $\frac{2}{3}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k\neq j,i}^{n} g_{ij}g_{jk}g_{ki}$ is the (endogenous) number of triangles.

Therefore the model reduces to an exponential random graph with covariates, 2-stars, and triangles.

We estimate the model using the exchange algorithm, as implemented in the package `Bergm`. In practice, we simulate from the posterior distribution of the parameters a few times, trying to improve the proposal distribution of the MCMC simulation.

In practice, we use the following procedure. We first estimate the model using a short simulation; we use the results of that simulation to improve the proposal distribution of the MCMC simulation in the next round. The proposal distribution is a random walk, and we modify the variance-covariance matrix of the parameters to improve the convergence of the sampler. We repeat this procedure 4 times, each time adjusting the number of parameters and network simulations.

The specification of the model is in the `formula`.

```
# formula for specification
formula <- net ~ edges + kstar(2) + triangles + nodematch("firmtype") +
          absdiff("logk") + absdiff("age") + nodematch("state")
```

Our first step involves a short simulation sampling 5000 parameters vectors from the posterior, and simulating a network with 500 iterations per parameter vector. Before the sampling we simulate the model for 1000 iterations (the so-called burn-in), to make sure that we are sampling from the correct posterior distribution.

The estimation uses parallel computations, as it runs 5 parallel simulations of the parameters.[1]

```
# estimation model first iteration, to explore parameter space
options(width = 200)
set.seed(1977)
```

[1]Depending on the availability of multiple processors, one can increase or decrease the number of parallel simulations, by changing the option 'nchains' in the following command.

```
model1 <- bergm(net ~ edges + kstar(2) + triangles +
                  nodematch("firmtype") + absdiff("logk") +
                  absdiff("age") + nodematch("state"),
                burn.in =  1000, # burnin posterior estimates
                main.iters = 5000, # posterior parameters proposals
                aux.iters = 500,  # network simulations per parameter proposal
                nchains =  5   # number of chains for snooker algo
                )

summary(model1)
```

```
##
##  Posterior Density Estimate for Model: y ~ edges + kstar(2) + triangles + nodematch("firmtype") + absdi
##
##                                    Mean          SD     Naive SE Time-series SE
## theta1 (edges)               -3.62464782 0.431071844 2.726338e-03   0.0612132034
## theta2 (kstar2)               0.13009493 0.059202723 3.744309e-04   0.0073232191
## theta3 (triangle)            -0.76600256 0.573778910 3.628896e-03   0.0828658246
## theta4 (nodematch.firmtype)   0.50057466 0.238419021 1.507894e-03   0.0392142991
## theta5 (absdiff.logk)        -0.44988466 1.651716909 1.044637e-02   0.2485254448
## theta6 (absdiff.age)         -0.03694477 0.007281281 4.605087e-05   0.0007532266
## theta7 (nodematch.state)      0.21643164 0.295276088 1.867490e-03   0.0465441304
##
##                                   2.5%         25%         50%         75%       97.5%
## theta1 (edges)               -4.30751518 -3.97404922 -3.60660215 -3.31004351 -2.75677781
## theta2 (kstar2)               0.00064167  0.08575085  0.14139781  0.17179058  0.23384532
## theta3 (triangle)            -2.09995248 -1.04897433 -0.67750755 -0.36357257  0.16205888
## theta4 (nodematch.firmtype)   0.06128035  0.33098339  0.48786038  0.64270220  0.96669966
## theta5 (absdiff.logk)        -3.94441945 -1.56620536 -0.31237976  0.73916996  3.01044447
## theta6 (absdiff.age)         -0.05137692 -0.04133661 -0.03606691 -0.03192379 -0.02324327
## theta7 (nodematch.state)     -0.38565667  0.02542735  0.22729522  0.41176040  0.71735609
##
##  Acceptance rate: 0.02
##
##
```

We use the output of this simulations to compute the covariances among the different parameters, so that we can use that to improve the proposal distribution and sample parameters in a much more efficient way.

```
## estimate covariance for better proposals
qq1 <- cov(model1$Theta)
```

So we input the value of this covariance in the estimation command. The next simulation includes 10000 parameters and 10000 network simulations for each parameter.

```
# estimation model second iteration, to explore parameter space
options(width = 200)
set.seed(1977)
model2 <- bergm(formula,
                burn.in = 1000, # burnin posterior estimates
                main.iters = 10000, # posterior parameters proposals
                aux.iters = 5000,  # network simulations per parameter proposal
                nchains = 5,   # number of chains for snooker algo
                sigma.eps = qq1  # covariance matrix for random walk proposal
)

summary(model2)
```

```
##
##  Posterior Density Estimate for Model: y ~ edges + kstar(2) + triangles + nodematch("firmtype") + absdi
##
##                                    Mean          SD     Naive SE Time-series SE
## theta1 (edges)               -3.59827765 0.307842587 1.376714e-03   0.0384313648
## theta2 (kstar2)               0.10853887 0.046603196 2.084158e-04   0.0044569908
## theta3 (triangle)            -0.79391635 0.511543653 2.287693e-03   0.0658385503
## theta4 (nodematch.firmtype)   0.51810117 0.175152884 7.833075e-04   0.0220144063
## theta5 (absdiff.logk)        -0.61909534 1.221071322 5.460797e-03   0.1680103515
## theta6 (absdiff.age)         -0.03527567 0.005978547 2.673687e-05   0.0004692129
## theta7 (nodematch.state)      0.26660422 0.170503667 7.625156e-04   0.0214744926
##
##                                    2.5%         25%         50%         75%       97.5%
## theta1 (edges)               -4.17526168 -3.80018966 -3.60578280 -3.40067420 -2.94555198
## theta2 (kstar2)               0.01432343  0.07799788  0.11108953  0.14500433  0.18790153
## theta3 (triangle)            -1.86236638 -1.08997494 -0.75260608 -0.46041879  0.05157689
## theta4 (nodematch.firmtype)   0.15492700  0.38480190  0.52697638  0.64304705  0.85314094
## theta5 (absdiff.logk)        -3.21440789 -1.54299503 -0.49770906  0.38857855  1.25250772
## theta6 (absdiff.age)         -0.04808406 -0.03928952 -0.03499358 -0.03126584 -0.02488713
## theta7 (nodematch.state)     -0.07818297  0.13763360  0.28383116  0.38095054  0.57635414
##
##  Acceptance rate: 0.01
##
##
```

We repeat the estimation two more times, improving the proposal distribution using the output of the previous simulation.

```
options(width = 200)
# estimate covariance for better proposals
qq2 <- cov(model2$Theta)

# estimation model third iteration, to explore parameter space
set.seed(1977)
model3 <- bergm(formula,
                burn.in = 1000, # burnin posterior estimates
                main.iters = 10000, # posterior parameters proposals
                aux.iters = 10000,  # network simulations per parameter proposal
                nchains = 5,   # number of chains for snooker algo
                sigma.eps = qq2  # covariance matrix for random walk proposal
)

summary(model3)
```

```
##
##  Posterior Density Estimate for Model: y ~ edges + kstar(2) + triangles + nodematch("firmtype") + absdi
##
##                                    Mean          SD     Naive SE Time-series SE
## theta1 (edges)               -3.47879661 0.344860208 1.542262e-03   0.0422523226
## theta2 (kstar2)               0.08309418 0.049193277 2.199990e-04   0.0045552923
## theta3 (triangle)            -0.85711586 0.647952338 2.897731e-03   0.0864772371
## theta4 (nodematch.firmtype)   0.54539132 0.209399377 9.364625e-04   0.0258836570
## theta5 (absdiff.logk)        -0.39012642 1.247028732 5.576882e-03   0.1681699191
## theta6 (absdiff.age)         -0.03581007 0.006839894 3.058894e-05   0.0003990825
## theta7 (nodematch.state)      0.22490967 0.224953848 1.006024e-03   0.0260265358
##
##                                    2.5%         25%         50%         75%       97.5%
## theta1 (edges)               -4.14527081 -3.70586854 -3.46473727 -3.23352225 -2.86295299
```

```
## theta2 (kstar2)             -0.02021560  0.05086291  0.08429421  0.11668708  0.17581401
## theta3 (triangle)           -2.52961767 -1.21710477 -0.79874288 -0.39559038  0.18740334
## theta4 (nodematch.firmtype)  0.16441946  0.40920425  0.55002704  0.66529113  1.01875411
## theta5 (absdiff.logk)       -2.79274846 -1.36040682 -0.28789530  0.39423084  2.12875874
## theta6 (absdiff.age)        -0.04746413 -0.04008338 -0.03577144 -0.03197768 -0.02354421
## theta7 (nodematch.state)    -0.25216546  0.08317889  0.23899328  0.35692376  0.66029163
##
##  Acceptance rate: 0.01
##
##
```

```
# estimate covariance for better proposals
qq3 <- cov(model3$Theta)

# estimation model third iteration, to explore parameter space
set.seed(1977)
model4 <- bergm(formula,
                burn.in = 1000, # burnin posterior estimates
                main.iters = 50000, # posterior parameters proposals
                aux.iters = 10000,  # network simulations per parameter proposal
                nchains = 5,   # number of chains for snooker algo
                sigma.eps = qq3  # covariance matrix for random walk proposal
)

summary(model4)
```

```
##
##  Posterior Density Estimate for Model: y ~ edges + kstar(2) + triangles + nodematch("firmtype") + absdi
##
##                                    Mean          SD     Naive SE Time-series SE
## theta1 (edges)              -3.55848464 0.360872776 7.217456e-04    0.021178192
## theta2 (kstar2)              0.09383891 0.053270887 1.065418e-04    0.002486273
## theta3 (triangle)           -0.99040958 0.721665346 1.443331e-03    0.044105676
## theta4 (nodematch.firmtype)  0.57787530 0.206283034 4.125661e-04    0.011278993
## theta5 (absdiff.logk)       -0.22935867 1.135832835 2.271666e-03    0.065878959
## theta6 (absdiff.age)        -0.03584567 0.007319874 1.463975e-05    0.000214215
## theta7 (nodematch.state)     0.22014211 0.225111504 4.502230e-04    0.012109335
##
##                                   2.5%         25%         50%         75%       97.5%
## theta1 (edges)              -4.25856215 -3.79175752 -3.55239295 -3.30577929 -2.85818118
## theta2 (kstar2)             -0.01990622  0.05946652  0.09890922  0.13143165  0.18653218
## theta3 (triangle)           -2.77884421 -1.37748819 -0.88998620 -0.48178333  0.12699006
## theta4 (nodematch.firmtype)  0.16805894  0.44386912  0.58147468  0.70626307  0.97570864
## theta5 (absdiff.logk)       -2.52599290 -0.94372166 -0.25165078  0.48552625  2.01857573
## theta6 (absdiff.age)        -0.04796086 -0.04010680 -0.03617941 -0.03216434 -0.02190746
## theta7 (nodematch.state)    -0.25216546  0.07995610  0.22930442  0.36222872  0.66990955
##
##  Acceptance rate: 0.01
##
##
```

We use the output from the last round as our final estimate.

Goodness of fit

To perform the goodness-of-fit tests we can run the following code.

```
gof <- bgof(model4, sample.size = 1000,
            aux.iters = model4$aux.iters,
            directed = FALSE,
            n.deg = 50, n.dist = 50, n.esp = 50)
```

In this example we do not report the goodness of fit test, to save some space.

Counterfactual simulations

Running counterfactual simulation is a little more involved.

For example, if we want to simulate the entry of 10 firms in the market all located in NY state, we need to create a new network with the 10 additional firms and then use the posterior estimated in the estimation step to simulate networks. Furthermore, the researcher needs to choose the value of the observed attributes of each new firm. There are several ways to make this choice, and this is the code we used in the paper.

We first collect all the data of the network in our data as follows.

```
# ENTRY of 3 firms in NY state
number_entry_firms <- 3

# observed network
obs.network.edgelist <- as.edgelist(net)
obs.network.size <- attr(obs.network.edgelist, "n")

# observed attributes
obs.attributes <- data.frame(cbind(
  get.vertex.attribute(net, "firmtype"),
  as.numeric(get.vertex.attribute(net, "logk")),
  as.numeric(get.vertex.attribute(net, "age")),
  get.vertex.attribute(net, "state")
), stringsAsFactors = FALSE)
names(obs.attributes) <- c("firmtype", "logk", "age", "state")

obs.attributes$logk <- as.numeric(obs.attributes$logk)
obs.attributes$age <- as.numeric(obs.attributes$age)
obs.attributes$newentrant <- 0
```

We then generate a new network object, for the policy counterfactual, containing both the incumbent and entrant firms.

```
# form network with new firms
net.policy.edgelist <- obs.network.edgelist
attr(net.policy.edgelist, "n") <- obs.network.size + number_entry_firms

attr(net.policy.edgelist, "vnames") <- 1: (obs.network.size + number_entry_firms)

attr.policy <- data.frame(matrix(0,
                                 ncol = 4,
                                 nrow = number_entry_firms),
                          stringsAsFactors = FALSE
                          )
names(attr.policy) <- c("firmtype", "logk", "age", "state")

attr.policy$firmtype  <- rep(1, number_entry_firms)
attr.policy$logk <- rnorm(number_entry_firms,
                          mean = mean(obs.attributes$logk),
                          sd = sd(obs.attributes$logk))
```

```r
attr.policy$state <- rep("New York", number_entry_firms)
attr.policy$newentrant <- rep(1, number_entry_firms)

attributes <- rbind(obs.attributes,attr.policy)

net.policy <- network(net.policy.edgelist, vertex.attr = attributes, directed = FALSE)
new.formula <- gsub("net", "net.policy", model4$formula)
policy.formula <- as.formula(paste(new.formula[2], new.formula[1], new.formula[3]))
```

After creating the new network and attributes for the policy counterfactual, we are now ready to run the simulations.

```r
number_of_simulations <- 1000
number_of_network_simulations <- 100000
```

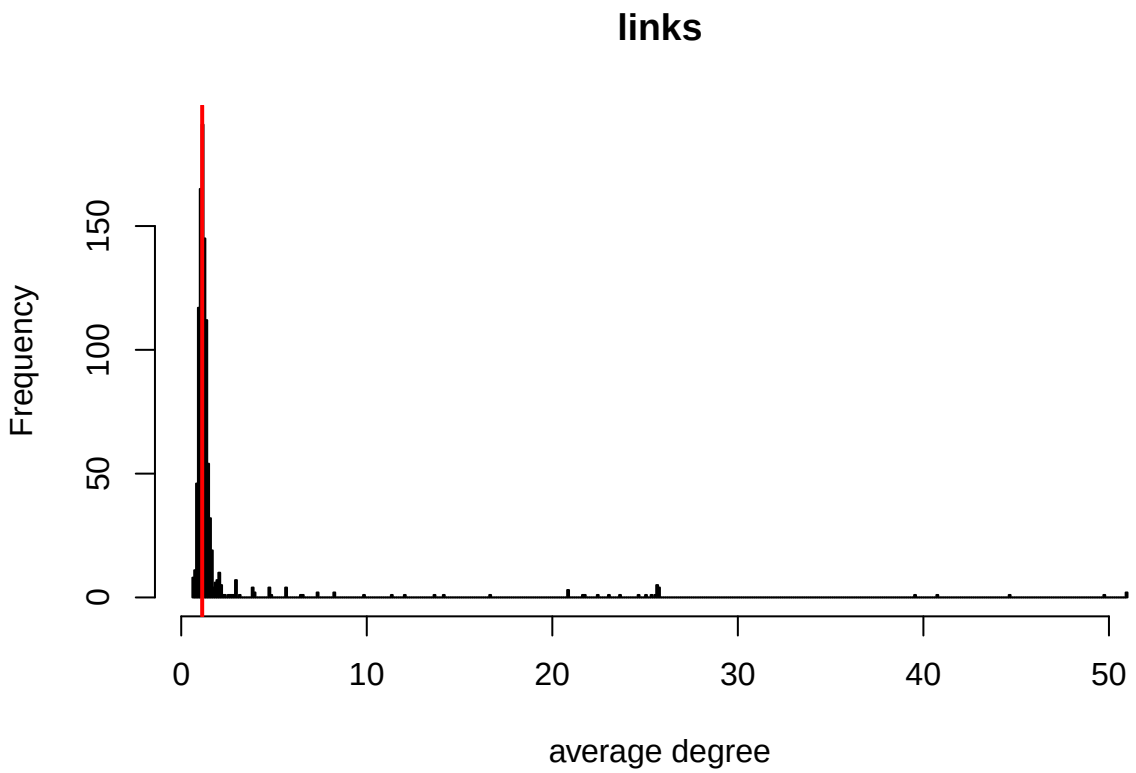
We run 1000 simulations. Each simulation uses a different vector of parameters from the posterior distribution, sampled randomly. We generate a simulated network for each vector of parameters, running the network simulation algorithm for 10^5 steps. For each simulated network, we compute the relevant network statistics. This corresponds to evaluating the posterior distribution of such network statistics.

```r
# extract a sample from the posterior distribution
post.sample <- sample(1:dim(model4$Theta)[1], size  =  number_of_simulations)

# initialize simulated networks as a list
nw.list <- vector("list", number_of_simulations)
class(nw.list) <- "network.list"

# simulate networks
for (i in 1:number_of_simulations){
  #cat(paste("NETWORK SIMULATION ", i, " OF ", number_of_simulations, "\n", sep =""))
  nw.list[[i]] <-  simulate(policy.formula, nsim = 1, coef = model4$Theta[post.sample[i],],
                   control = control.simulate(MCMC.burnin = number_of_network_simulations),
                   verbose = TRUE, seed = i)
}

# summary network statistics for figures
summary.stats <- data.frame(summary(nw.list ~ edges + triangles + kstar(2) ))
degree.distrib <- summary(nw.list ~ degree(0:(obs.network.size + number_entry_firms)))
homophily.firmtype <- summary(nw.list ~ nodematch("firmtype"))
homophily.state <- summary(nw.list ~ nodematch("state"))

# observed network statistics
obs.stats <-  summary( net ~ edges + triangles + kstar(2) )
obs.degree.distrib <- summary(net ~ degree(0:(obs.network.size + number_entry_firms)))
obs.homophily.firmtype <- summary(net ~ nodematch("firmtype"))
obs.homophily.state <- summary(net ~ nodematch("state"))
```

Finally we provide some visualization of the posterior distribution of network statistics.

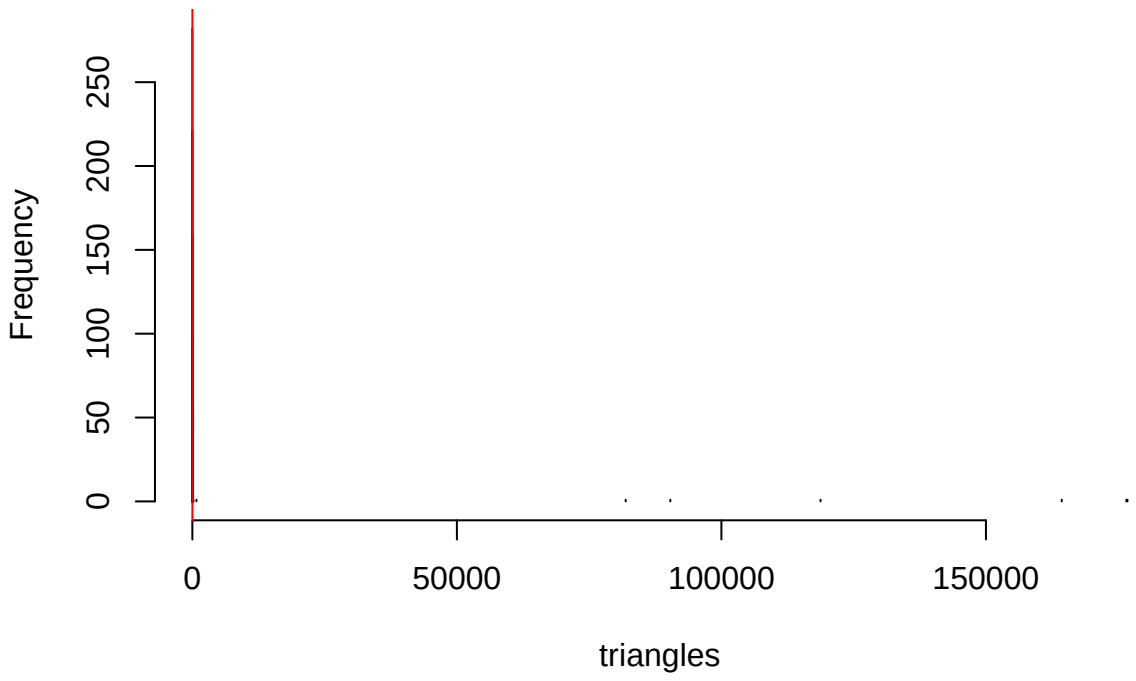
The first figure is the simulated number of links, which we visualize as average degree.

```r
# FIGURES
policy.network.size <- obs.network.size + number_entry_firms

# histograms of edges (density)
hist(summary.stats$edges/policy.network.size, breaks = 500, col = "grey",
     main = "links", xlab = "average degree")
abline(v = obs.stats[1]/obs.network.size, lwd = 2, col = "red")
```



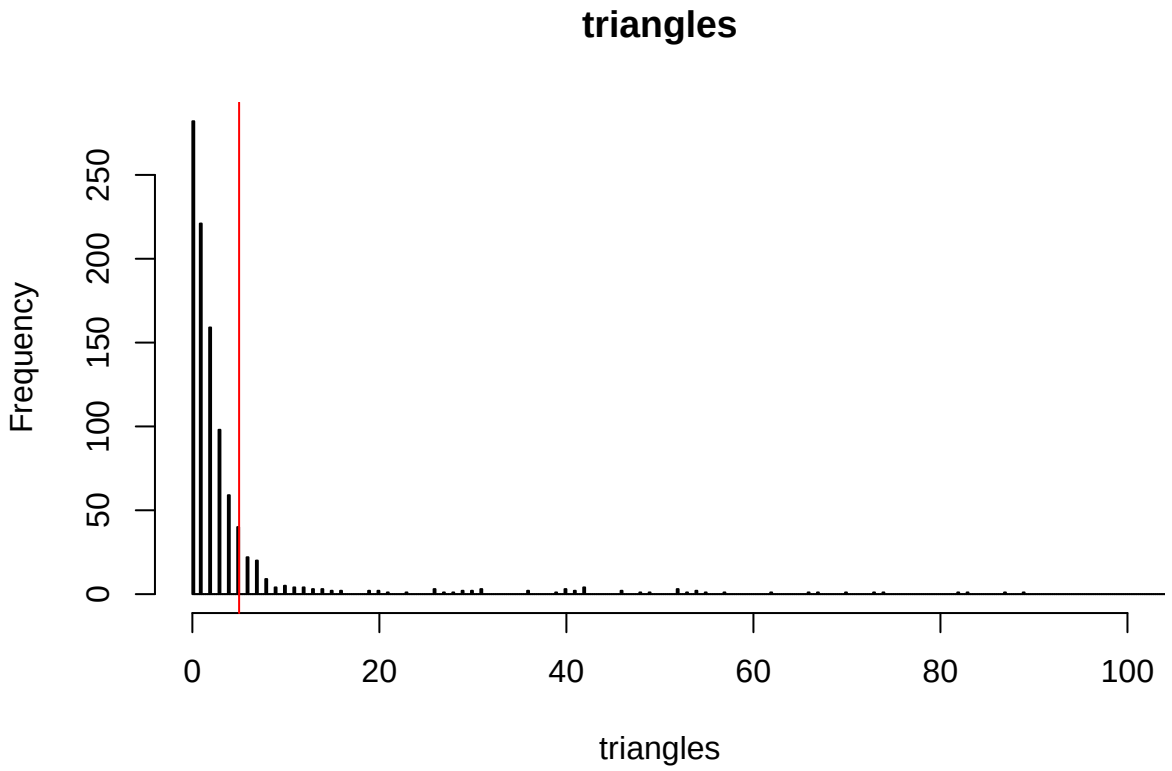
For triangles, the simulations generate some outliers, which suggest that with small probability the network may become really dense.

```
# histograms of triangles (clustering)
hist(summary.stats$triangle, breaks = 1000000, col = "grey", #xlim = c(0,1),
     main = "triangles", xlab = "triangles")
abline(v = obs.stats[2], lwd = 1, col = "red")
```



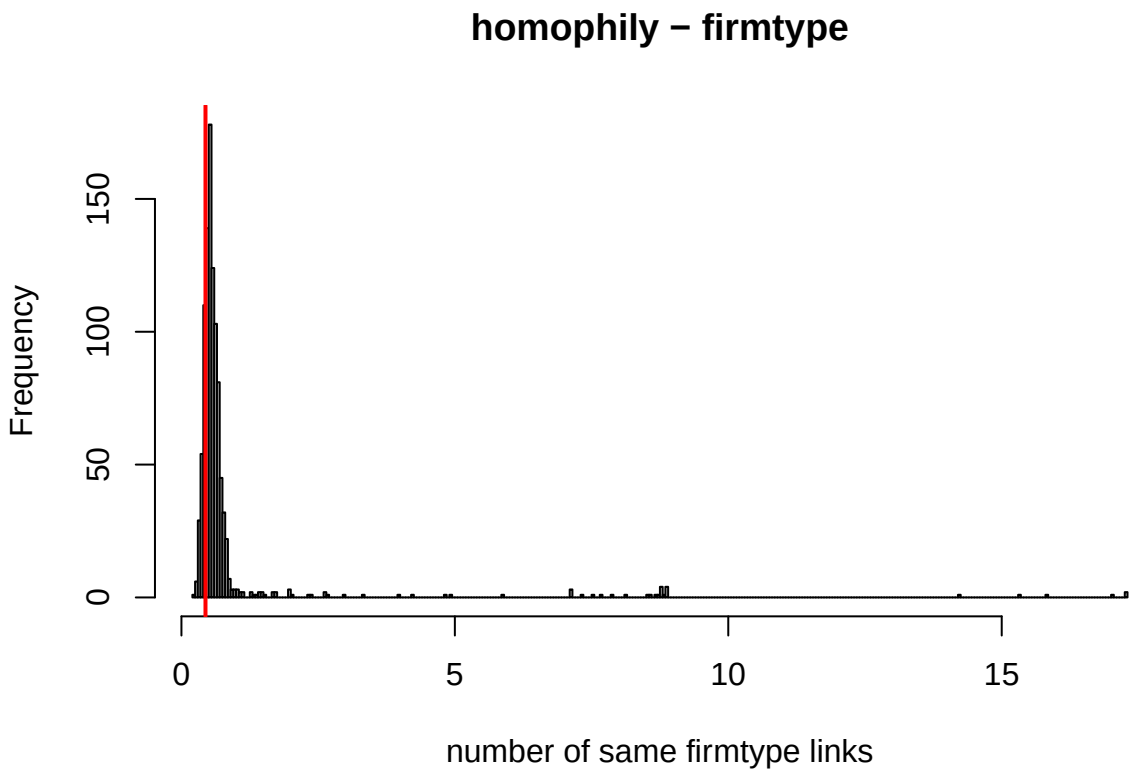
To better visualize the distribution of our couterfactual simulations' triangle count, we also provide zoomed-in version of the previous histogram, showing that most of the probability is attached to low values of the triangles.

```
# histograms of triangles (clustering) zooming in
hist(summary.stats$triangle, breaks = 1000000, col = "grey", xlim = c(0,100),
     main = "triangles", xlab = "triangles")
abline(v = obs.stats[2], lwd = 1, col = "red")
```

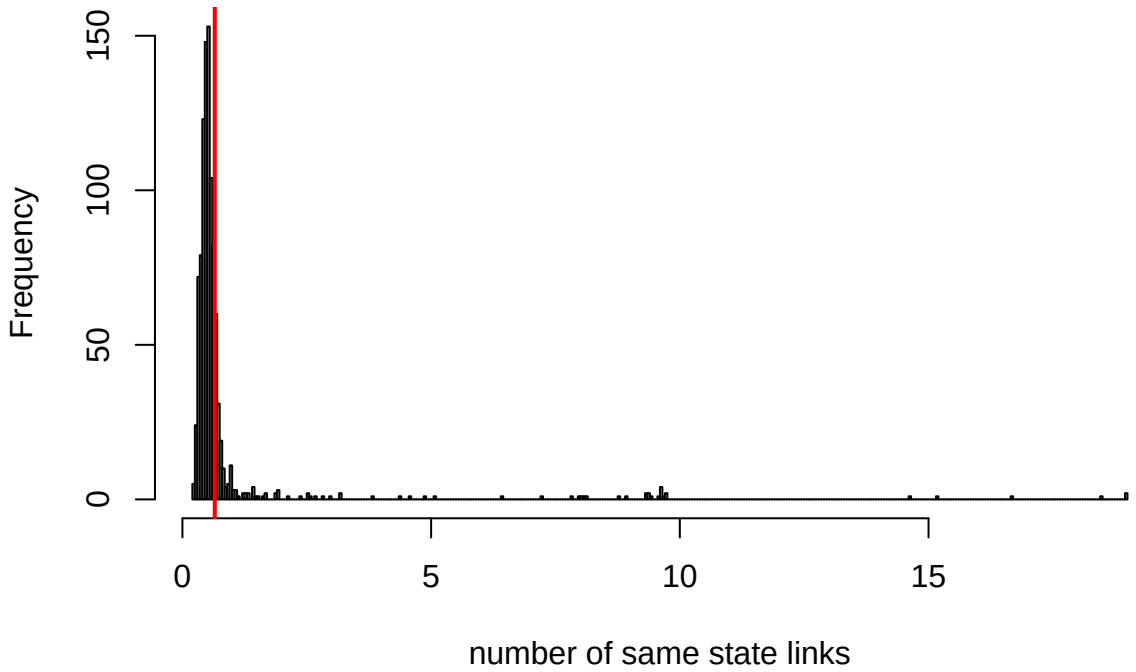


Finally, we report the homophily by firm type and state below.

```
# histograms of homophily firmtype
hist(homophily.firmtype/policy.network.size, breaks = 500, col = "grey",
     main = "homophily - firmtype", xlab = "number of same firmtype links")
abline(v = obs.homophily.firmtype/obs.network.size, lwd = 2, col = "red")
```




```
# histograms of homophily state
hist(homophily.state/policy.network.size, breaks = 500, col = "grey",
     main = "homophily - state", xlab = "number of same state links")
abline(v = obs.homophily.state/obs.network.size, lwd = 2, col = "red")
```



Conclusion

We have provided a short tutorial for the estimation and counterfactual simulation of the model. The tutorial offers a step-by-step guide for a researcher that has our data and wants to replicate the results in the paper.[2] It also provides a guide to a researcher that would like to apply our method to a different dataset or application. We hope that this will allow management and organization scholars to implement and improve our methods in future studies.

[2]However, the number of simulations for estimation and counterfactuals should be increased when using our dataset, because the network we use in the tutorial has $n = 100$ nodes, while in the paper the network is much larger.