Supplemantary Appendix 2

BLIND

May, 2019

## Reference distribution

### Rationale

The reference distribution for the subset of successful bills is constructed in order to capture the effect of subsetting on network statistics. In particular, to evaluate whether the observed value can be accounted by the small number of connections due to the fact that succesful bills are a small subset of all bills. Hence, as Barabasi-Lazlo (2016: section 3.10 ) : “For this we turn to the random network model as a guide: If the property is present in the model, it means that randomness can account for it. If the property is absent in random networks, it may represent some signature of order, requiring a deeper explanation.”

In this case, the central mechanism is subsetting, and thus, the random compoment of the benchmark model addresses this specific mechanims. In order to do so, I construct a reference distribution by generating networks from subsets of bills selected at random.

In particular, for each year, a random sample of the size of the actual number of succesful bills on that year is generated from all bill; the derived policy networks of those randomly selected bills is generate; and descriptors are obtained. The process is repeated 10,000 times and reference distribution for each statistics is constructed.

In order to compute the small world quotient, for each simulated network, I simulate 100 erdo-renyi random graphs of the same size, and then obtain the mean of the simulated average local clustering coefficient and average path length.

### Code

The databases used to generate the simulations are:

* Edgelist from all bills for each legislative year under analysis
* Database of legislators, with information on coalition membership for each legislative year under analysis.

I present two codes, one that generates the data for one year, and the other that runs a loop over all year under analysis. Since each year simulates n\_sim1=1,000 networks from random subsetting, the running time is long. Hence, the choice for the user to run each year separetedly.

The user may specify: + n\_sim1: number of networks simulated under random subsetting + n\_sim2: number of networks simulated to obtain expected average local clustering coefficient and average path length on a random graph of the same size.

The database generated “totsim\_[year].rds” contains n\_sim1 observations, and the following variables, that correspond to network statistics:

* density
* clustering: global clustering coefficient
* clustering\_localave: average local clustering coefficient
* ave\_path: average path length
* modularity\_pact: weighted modularity by coalition
* clustlocalave\_er: mean of average local cluster of n\_sim2 e-r graphs of the same size as the ith simulated graph.
* avepath\_er: mean average path length of n\_sim2 e-r graphs of the same size as the ith simulated
* per\_only: percent of legislators in communities with only one coalition
* per\_cross: percent of legislators in communities with at least 25% of a different coalition

####

#### Code for each year

In this code, the user runs the code each year, and thus,the user needs to also specify nsamp and year each time.

|  |  |
| --- | --- |
| **Legislative year** | **Number of succesful bills** |
| 2006 | 35 |
| 2007 | 36 |
| 2008 | 29 |
| 2009 | 13 |
| 2010 | 35 |
| 2011 | 36 |
| 2012 | 36 |
| 2013 | 22 |
| 2014 | 43 |
| 2015 | 40 |
| 2016 | 16 |

This code defines year=2006.

nsamp<-36
year<-2006

n\_sim1<-1000
n\_sim2<-100

nom<-paste("net\_sponsor", year, ".Rda", sep="")
net\_sponsor <-get(load(nom)) # edge list of succesful bills from 2006 legislative year

totsim<-tibble(num\_sim=0, density=0,
 clustering=0,
 clustering\_localave=0,
 ave\_path=0,
 modularity\_pact=0, clustlocalave\_er=0, avepath\_er=0, per\_only=0, per\_cross=0)

net\_sponsor %>% mutate(dup=duplicated(idboletin)) %>% filter(dup==FALSE) %>% select(year, idboletin)->bills

set.seed(1)

for (i in (1:n\_sim1)){

 # for each i in the loop generated a network graph for a random subset of bills
 samp<-sample\_n(bills, nsamp)

 net\_sponsor %>% filter(idboletin %in% unique(c(samp$idboletin))) -> net\_sim

 vertex<-get(load(paste("vertex\_", year, ".Rda", sep="")))

 vertex %>% dplyr::filter(id\_nuevo %in% unique(c(net\_sim$from,net\_sim$to))) -> vertex\_sim

 igraph\_sim <- graph\_from\_data\_frame(net\_sim, directed=FALSE, vertices=vertex\_sim)

 E(igraph\_sim)$weight <- 1

 #Simplify

 igraph\_sim1 <- igraph::simplify(igraph\_sim)

 ## Pacts

 pact\_num<- as.numeric(factor(V(igraph\_sim1)$pact))

 ## Communities
 ##
 comm\_sim <- cluster\_walktrap(igraph\_sim1)

 ## Generate statistics for each simulation

if (length(unique(pact\_num))>2) {
 cc<-tibble(com=seq(1,length(comm\_sim),1),al=prop.table(table(membership(comm\_sim),V(igraph\_sim1)$pact), margin=1)[,1],
 conc=prop.table(table(membership(comm\_sim),V(igraph\_sim1)$pact), margin=1)[,2],
 otro=prop.table(table(membership(comm\_sim),V(igraph\_sim1)$pact), margin=1)[,3],
 tot=addmargins(table(membership(comm\_sim),V(igraph\_sim1)$pact))[1:length(comm\_sim),4])
} else {

 cc<-tibble(com=seq(1,length(comm\_sim),1),al=prop.table(table(membership(comm\_sim),V(igraph\_sim1)$pact), margin=1)[,1],
 conc=prop.table(table(membership(comm\_sim),V(igraph\_sim1)$pact), margin=1)[,2],
 otro=0,
 tot=addmargins(table(membership(comm\_sim),V(igraph\_sim1)$pact))[1:length(comm\_sim),3])
}

 cc %>% mutate (only=ifelse(al==0 & conc>0, tot, ifelse(al>0 & conc==0, tot, 0)),
 cross=ifelse(al>=.25 & conc>=.25, tot, 0)) %>% group\_by() %>%
 summarise(tot=sum(tot), only=sum(only), cross=sum(cross)) %>% ungroup() %>%
 mutate(per\_only=only/tot, per\_cross=cross/tot) %>% select(per\_only, per\_cross) ->cd

### For each network i, generate n2\_sims random graphs

 nv<-vcount(igraph\_sim1)
 ne<-ecount(igraph\_sim1)

 base\_sim1<-tibble(clustering\_localave=0, ave\_path=0 )

 for (j in (1:n\_sim2)){

 g\_sim\_sim1<-erdos.renyi.game(nv, ne, type="gnm")

 sim\_sim1<-tibble(clustering\_localave=transitivity(g\_sim\_sim1, type ="average"),
 ave\_path=mean\_distance(g\_sim\_sim1))

 base\_sim1<-rbind(base\_sim1, sim\_sim1)

 }

 base\_er<-base\_sim1[2:(n\_sim2+1),]

 mean<-apply(base\_er, MARGIN =2, mean)

 sim<-tibble(num\_sim=i, density=graph.density(igraph\_sim1),
 clustering=transitivity(igraph\_sim1, type ="undirected"),
 clustering\_localave=transitivity(igraph\_sim1, type ="average"),
 ave\_path=mean\_distance(igraph\_sim1),
 modularity\_pact=modularity(igraph\_sim1,pact\_num, weights=E(igraph\_sim1)$weight), clustlocalave\_er=mean[1], avepath\_er=mean[2])

 sim<-cbind(sim, cd)

 totsim<-rbind(totsim, sim)

}

totsim$year<-year
totsim<-totsim[2:(n\_sim1+1),]
totsim %>% mutate(q=(clustering\_localave/clustlocalave\_er)/(ave\_path/avepath\_er)) -> totsim

write\_rds(totsim, paste("RandomSubsetSim", year, ".rds", sep=""))

#### Code for all years

Beware that this may take a very long time! For that purpose, the codes prints the year when it begins the simulation.

n\_sim1<-1000
n\_sim2<-100

n\_legyear=read\_rds("n\_aprobyear.rds")

for (year in (2006:2015)){
print(year)

 nsamp<-n\_legyear$tot\_aprob[n\_legyear$legyear==year]

nom<-paste("net\_sponsor", year, ".Rda", sep="")
net\_sponsor <-get(load(nom)) # edge list of succesful bills from 2006 legislative year

totsim<-tibble(num\_sim=0, density=0,
 clustering=0,
 clustering\_localave=0,
 ave\_path=0,
 modularity\_pact=0, clustlocalave\_er=0, avepath\_er=0, per\_only=0, per\_cross=0)

net\_sponsor %>% mutate(dup=duplicated(idboletin)) %>% filter(dup==FALSE) %>% select(year, idboletin)->bills

set.seed(1)

for (i in (1:n\_sim1)){

 # for each i in the loop generated a network graph for a random subset of bills
 samp<-sample\_n(bills, nsamp)

 net\_sponsor %>% filter(idboletin %in% unique(c(samp$idboletin))) -> net\_sim

 vertex<-get(load(paste("vertex\_", year, ".Rda", sep="")))

 vertex %>% dplyr::filter(id\_nuevo %in% unique(c(net\_sim$from,net\_sim$to))) -> vertex\_sim

 igraph\_sim <- graph\_from\_data\_frame(net\_sim, directed=FALSE, vertices=vertex\_sim)

 E(igraph\_sim)$weight <- 1

 #Simplify

 igraph\_sim1 <- igraph::simplify(igraph\_sim)

 ## Pacts

 pact\_num<- as.numeric(factor(V(igraph\_sim1)$pact))

 ## Communities
 ##
 comm\_sim <- cluster\_walktrap(igraph\_sim1)

 ## Generate statistics for each simulation

if (length(unique(pact\_num))>2) {
 cc<-tibble(com=seq(1,length(comm\_sim),1),al=prop.table(table(membership(comm\_sim),V(igraph\_sim1)$pact), margin=1)[,1],
 conc=prop.table(table(membership(comm\_sim),V(igraph\_sim1)$pact), margin=1)[,2],
 otro=prop.table(table(membership(comm\_sim),V(igraph\_sim1)$pact), margin=1)[,3],
 tot=addmargins(table(membership(comm\_sim),V(igraph\_sim1)$pact))[1:length(comm\_sim),4])
} else {

 cc<-tibble(com=seq(1,length(comm\_sim),1),al=prop.table(table(membership(comm\_sim),V(igraph\_sim1)$pact), margin=1)[,1],
 conc=prop.table(table(membership(comm\_sim),V(igraph\_sim1)$pact), margin=1)[,2],
 otro=0,
 tot=addmargins(table(membership(comm\_sim),V(igraph\_sim1)$pact))[1:length(comm\_sim),3])
}

 cc %>% mutate (only=ifelse(al==0 & conc>0, tot, ifelse(al>0 & conc==0, tot, 0)),
 cross=ifelse(al>=.25 & conc>=.25, tot, 0)) %>% group\_by() %>%
 summarise(tot=sum(tot), only=sum(only), cross=sum(cross)) %>% ungroup() %>%
 mutate(per\_only=only/tot, per\_cross=cross/tot) %>% select(per\_only, per\_cross) ->cd

### For each network i, generate n2\_sims random graphs

 nv<-vcount(igraph\_sim1)
 ne<-ecount(igraph\_sim1)

 base\_sim1<-tibble(clustering\_localave=0, ave\_path=0 )

 for (j in (1:n\_sim2)){

 g\_sim\_sim1<-erdos.renyi.game(nv, ne, type="gnm")

 sim\_sim1<-tibble(clustering\_localave=transitivity(g\_sim\_sim1, type ="average"),
 ave\_path=mean\_distance(g\_sim\_sim1))

 base\_sim1<-rbind(base\_sim1, sim\_sim1)

 }

 base\_er<-base\_sim1[2:(n\_sim2+1),]

 mean<-apply(base\_er, MARGIN =2, mean)

 sim<-tibble(num\_sim=i, density=graph.density(igraph\_sim1),
 clustering=transitivity(igraph\_sim1, type ="undirected"),
 clustering\_localave=transitivity(igraph\_sim1, type ="average"),
 ave\_path=mean\_distance(igraph\_sim1),
 modularity\_pact=modularity(igraph\_sim1,pact\_num, weights=E(igraph\_sim1)$weight), clustlocalave\_er=mean[1], avepath\_er=mean[2])

 sim<-cbind(sim, cd)

 totsim<-rbind(totsim, sim)

}

totsim$year<-year
totsim<-totsim[2:(n\_sim1+1),]
totsim %>% mutate(q=(clustering\_localave/clustlocalave\_er)/(ave\_path/avepath\_er)) -> totsim

write\_rds(totsim, paste("RandomSubsetSim", year, ".rds", sep=""))

}