Online Appendix: Not for Publication

Appendix C. Detailed Estimation Procedure

This section presents the detailed walk-through for the estimation procedure.

(1) Download ARD code: https://github.com/MengjiePan/BCMP
(2) Format survey data in the following manner:
- Create a dataset(csv,xls) that is m ARD nodes by K ARD responses for each village and save each file as ARD_SURVEY_i.csv
- Create a dataset that is n nodes by the K ARD-trait covariates from the census for each village and save each file as ARD_CENSUS_i.csv
- Create a dataset that is m ARD nodes by the L covariates from the census (e.g., GPS, household identifiers). Create another dataset that is $n-m$ Non ARD nodes by the L covariates from the census(same covariates as used for ARD Nodes). Use L covariates of these two datasets in a distance function to create a $n-m$ by m dataset. This will be used in k-nearest neighbours algorithm. Save each file as distance_i.csv

```
// import the CENSUS file
use ARD_CENSUS , clear

** Keep id_village and id_hhid as the first 2 variables followed by
** k ard traits( 8 in this example)
keep id_village id_hhid ard_t_floors ard_t_smartph ard_t_child  ///
ard_t_migrate ard_t_bike ard_t_gates ard_t_pass ard_t_goat

* If the dataset has j villages with id_village as 1 to j then

forvalues village =1(1)`j'{
preserve
keep if id_village == `village'
// each village csv is saved separately
export delimited using ARD_CENSUS_`village', replace
restore
}
```

```
// import the CENSUS file
use ARD_CENSUS , clear
** Keep id_village and id_hhid as the first 2 variables followed by
** k ard traits( 8 in this example)
keep id_village id_hhid ard_t_floors ard_t_smartph ard_t_child  ///
ard_t_migrate ard_t_bike ard_t_gates ard_t_pass ard_t_goat

* If the dataset has j villages with id_village as 1 to j then
forvalues village =1(1)`j'{
preserve
keep if id_village == `village'
// each village csv is saved separately
export delimited using ARD_CENSUS_`village', replace
restore
}
save ARD_CENSUS, replace // save dta file ARD_CENSUS
```

(3) Copy the downloaded R files in the same folder. The folder structure should be as shown in the figure below(for 4 villages)

- ARD_CENSUS.dta
- ARD_CENSUS_1.csv
- ARD_CENSUS_2.csv
- ARD_CENSUS_3.csv
- ARD_CENSUS_4.csv
- ARD_SURVEY.dta
- ARD_SURVEY_1.csv
- ARD_SURVEY_2.csv
- ARD_SURVEY_3.csv
- ARD_SURVEY_4.csv
- distance_1.csv
- distance_2.csv
- distance_3.csv
- distance_4.csv
- example.R
- latent_surface_model.R
- main.R

(4) Open the file example.R

(5) Download R Packages - `igraph`(Csardi and Nepusz, 2006) , `movMF`(Hornik and Grün, 2014), `xlsx`(Dragulescu et al., 2018) (if the datasets are in xls), `readstata13`(Garbuszus and Jeworutzki, 2018) (if the datasets are in Stata 13,14) [example.R downloads these packages]

(6) Enter the path to the folder in variable `r_folder` (Line 24). **Running the R Script example.R now** should generate the ARD Output in the Folder `OUT` in the current folder. The steps given next explain the process in detail through code snippets.

```
17 ################################################################################
18
19 ## Set Path ##
20
21 ## INSTRUCTION - Enter the path of the input folder in r_folder . Path should b
22 ## for e.g. - r_folder <- 'C:/Users/V/Dropbox/Data/ARD/')
23 ## Enter folder path below
24 r_folder <- ''
25
26 ## Setting the Path
27 setwd(r_folder)
```

(7) Preparing the datasets for constructing ARD :
- The datasets created in Step 2 are imported(Line 36-54) and are named `ard_survey`, `ard_census` and `distance.all` respectively
- Calculate the value of variable `total.prop` - *fraction of ties in the network that are made with members of group k, summed over K groups* using `example.R` (Line no 69-80). The variable `villagei` stores the `ard_census` traits.

```
36  ard_survey_file_list = list.files(pattern='ARD_SURVEY.*\\.csv')
37  ard_census_file_list = list.files(pattern="ARD_CENSUS.*\\.csv")
38  distance_file_list=list.files(pattern="distance.*\\.csv")
39
40  ard_survey_list = lapply(ard_survey_file_list, read.csv)
41  ard_census_list = lapply(ard_census_file_list, read.csv)
42  distance.all = lapply(distance_file_list, function(i){
43    read.csv(i, header=FALSE)
44  })
45
46  no_village=length(ard_survey_file_list)
47  ard_survey=ard_survey_list[[1]]
48  ard_census=ard_census_list[[1]]
49
50  for ( i in 2:no_village){
51
52    ard_survey=rbind(ard_survey,ard_survey_list[[i]])
53    ard_census=rbind(ard_census,ard_census_list[[i]])
54  }
```

```
69  total.prop=NULL
70  x.axis=NULL
71  for (vlg in 1:no_village){
72    villagei=ard_census[which(ard_census$id_village==vlg),]
73    villagei[which(villagei<0,arr.ind=T)]=NA
74    n=dim(villagei)[1]
75    temp=sum(x.axis)
76    for (k in c(3:(k_traits+2))){
77      x.axis=c(x.axis,sum(as.numeric(villagei[,k]==1),na.rm = T)/length(!is.na
78    }
79    total.prop=c(total.prop,sum(x.axis)-temp)
80  }
```

(8) Estimate the parameters of the model: $(\nu_i, z_i)_{i=1}^m$ for the m ARD households, ζ, $(\upsilon_k, \eta_k)_{k=1}^m$ (the latent trait distribution location and concentration parameters).

- Use `example.R` to call(Line 93) `main.R`, which calls(Line 23) function `f.metro` in `latent_surface_model.R`.
- The call to function `main` of `main.R` on Line 93 requires 4 input variables
  - `y` - use the `ard_survey` dataset that has been imported
  - `total.prop` - Calculated in Step 4
  - `muk.fix` - the positions of fixed variables calculated in Line 126-127 of `example.R`
  - `distance.matrix` - use the `distance.all` dataset that has been imported
- The Output of the call to `f.metro` is stored in variable `posterior` of `main.R`

```
source('main.R')
g.sims=list()
setwd(out_dir)

for (vlg in 1:no_village){
  y=ard_survey[ard_survey$id_village==vlg,c(3:(k_traits+2))]
  y[which(y<0,arr.ind=T)]=NA
  y=as.matrix(y)
  muk.fix.ind=sample(1:k_traits,size=4,replace=F)
  muk.fix=matrix(rnorm(12),nrow=4,ncol=3)
  muk.fix=sweep(muk.fix,MARGIN=1,1/sqrt(rowSums(muk.fix^2)),`*`)
  result=main(y=y,total.prop=total.prop[vlg],muk.fix=muk.fix,n.iter=3000, m.i
              is.sample=TRUE,distance.matrix=as.matrix(distance.all[[vlg]]),K
  g.sims=c(g.sims,list(result))
  save(g.sims,file="g.sims.RData")
}
```

```
20 main=function(y,total.prop,muk.fix,n.iter=3000, m.iter=3, n.thin=10,is.sample
21   n=dim(y)[1]
22   z.pos.init=generateRandomInitial(n,ls.dim)
23   out=f.metro(y,total.prop=total.prop,n.iter=n.iter, m.iter=m.iter, n.thin=n.
24   posterior=getPosterior(out,n.iter,m.iter,n.thin,n)
25   est.degrees=posterior$est.degrees
26   est.eta=posterior$est.eta
27   est.latent.pos=posterior$est.latent.pos
28   est.gi=getGi(est.degrees,est.eta)
```

(9) Estimate ν_i and z_i for the $n-m$ nodes that are in the census but not the ARD sample.

- `main.R` (Line no 30) calls function `getPosteriorAllnodes` in `main.R`. The call to the function takes variable `distance.matrix` as an input(which had been passed to function `main` from `example.R` in Step 5)
- Output is stored in variable `posteriorAll`. The estimated latent positions z_i are stored as an attribute of `posteriorAll` as `est.latent.pos.all`
- `getPosteriorAllnodes` estimates ν_i and z_i using k-means from `distance.matrix` variable. This variable has been calculated using the $K+L$ covariates for the m nodes in the ARD sample and $n-m$ Non-ARD nodes

```
20 main=function(y,total.prop,muk.fix,n.iter=3000, m.iter=3, n.thin=10,is.sample
21   n=dim(y)[1]
22   z.pos.init=generateRandomInitial(n,ls.dim)
23   out=f.metro(y,total.prop=total.prop,n.iter=n.iter, m.iter=m.iter, n.thin=n.
24   posterior=getPosterior(out,n.iter,m.iter,n.thin,n)
25   est.degrees=posterior$est.degrees
26   est.eta=posterior$est.eta
27   est.latent.pos=posterior$est.latent.pos
28   est.gi=getGi(est.degrees,est.eta)
29   if(is.sample){
30     posteriorAll=getPosteriorAllnodes(distance.matrix,est.gi,est.latent.pos,K
31     est.gi.all=posteriorAll$est.gi.all
32     est.latent.pos.all=posteriorAll$est.latent.pos.all
```

```r
46 getPosteriorAllnodes=function(distance.matrix,est.gi,est.latent.pos,Knn.K,ls.
47   n.ARD=dim(distance.matrix)[2]
48   n.nonARD=dim(distance.matrix)[1]
49   est.gi.all=NULL
50   est.latent.pos.all=NULL
51   for (ind in 1:dim(est.gi)[1]){
52     g.ARD=est.gi[ind,]
53     z.ARD=matrix(est.latent.pos[ind,],byrow=F,nrow=n.ARD,ncol=ls.dim)
54 
55     g.nonARD=NULL
56     z.nonARD=NULL
57     for (i in 1:n.nonARD){
```

(10) Draw a set of $b = 1, \ldots, B$ draws from the network formation probability model (now with estimated parameters for all nodes) from the posterior distribution.

- Use `main.R` (Line no 33) to call function `simulate.graph.all` . The output is stored in variable `g.sims` . `simulate.graph.all` calls(Line 108) `simulate.graph.once` for each run.
- Draw a parameter vector θ (all the above parameters) from the posterior.
- Draw a graph g_b given θ_b. (Line 130 - function `simulate.graph.once`)

```r
20 main=function(y,total.prop,muk.fix,n.iter=3000, m.iter=3, n.thin=10,is.sample
21   n=dim(y)[1]
22   z.pos.init=generateRandomInitial(n,ls.dim)
23   out=f.metro(y,total.prop=total.prop,n.iter=n.iter, m.iter=m.iter, n.thin=n.
24   posterior=getPosterior(out,n.iter,m.iter,n.thin,n)
25   est.degrees=posterior$est.degrees
26   est.eta=posterior$est.eta
27   est.latent.pos=posterior$est.latent.pos
28   est.gi=getGi(est.degrees,est.eta)
29   if(is.sample){
30     posteriorAll=getPosteriorAllnodes(distance.matrix,est.gi,est.latent.pos,K
31     est.gi.all=posteriorAll$est.gi.all
32     est.latent.pos.all=posteriorAll$est.latent.pos.all
33     g.sims=simulate.graph.all(est.degrees,est.eta,est.latent.pos,est.gi,est.g
34   }else{
35     g.sims=simulate.graph.all(est.degrees,est.eta,est.latent.pos,est.gi,est.g
```

```r
101 simulate.graph.all=function(est.degrees.ARD,est.eta,est.latent.pos.ARD,est.g
102   g.sims=list()
103   n.ARD=dim(est.degrees.ARD)[2]
104   n=dim(est.gi)[2]
105   for (ind in 1:length(est.eta)){
106     z=matrix(est.latent.pos[ind,],byrow=F,nrow=n,ncol=ls.dim)
107     z.ARD=matrix(est.latent.pos.ARD[ind,],byrow=F,nrow=n.ARD,ncol=ls.dim)
108     g.sims=c(g.sims,list(simulate.graph.once(z=z,g=est.gi[ind,],eta=est.eta[
109   }
110   return(g.sims)
111 }
```

```
simulate.graph.once=function(z,g,eta,d.ARD,z.ARD,g.ARD){
  n.ARD=length(g.ARD)
  adjexp=matrix(NA,nrow=n.ARD,ncol=n.ARD)
  diag(adjexp)=0
  for(i in 1:(n.ARD-1)){
    for (j in (i+1):n.ARD){
      adjexp[i,j]=adjexp[j,i]=exp(g.ARD[i]+g.ARD[j]+eta*sum(z.ARD[i,]*z.ARD[j
    }
  }
  const=sum(exp(d.ARD))/sum(adjexp)
  n=length(g)
  adj=matrix(NA,nrow=n,ncol=n)
  diag(adj)=0
  for(i in 1:(n-1)){
    for (j in (i+1):n){
      p.ij=exp(g[i]+g[j]+eta*sum(z[i,]*z[j,]))*const
      edge=rbinom(n=1,size=1,prob=min(p.ij,1))
      adj[i,j]=adj[j,i]=edge
    }
  }
```

(11) Compute network statistics of interest $S(g_b)$ for each draw g_b for $b = 1, ..., B$.

- Construct your own desired functions
- Or use a suggested code `example.R` (Line no 115-144)

```
121 for (vlg in 1:no_village){
122   est.closeness=NULL
123   centrality=NULL
124   est.max.eigenvalue=NULL
125   est.betweenness=NULL
126   est.avg.path.length=NULL
127
128   for(t in 1:times){
129     graph.temp=graph.adjacency(g.sims[[vlg]][[t]],mode="undirected")
130     centrality=rbind(centrality,evcent(graph.temp,scale=F)$vector)
131     est.max.eigenvalue=c(est.max.eigenvalue,evcent(graph.temp,scale=F)$value
132     est.closeness=rbind(est.closeness,closeness(graph.temp))
133     est.betweenness=rbind(est.betweenness,betweenness(graph.temp))
134     est.avg.path.length=c(est.avg.path.length,mean_distance(graph.temp,direc
135
136   }
137   centrality=colMeans(centrality)
138   write.table(as.matrix(centrality),file = paste0('centrality_',vlg,'.csv'),
139   est.centrality.all=c(est.centrality.all,list(centrality))
```

(12) Import the network characteristics that have been generated in folder `OUT`

```
***********import the network data that has been generated *************
cd `r_folder'
cd "OUT"

forvalues k=1(1)4{
import delimited using degree_`k'.csv, clear    //import degree data
** merge to get id_hhid , id_village using _n as uid **
append using degree.dta

import delimited using centrality_`k'.csv, clear    //import degree data
** merge to get id_hhid , id_village using _n as uid**
append using centrality.dta

import delimited using closeness_`k'.csv, clear //import degree data for
** merge to get id_hhid , id_village with _n as uid**
append using closeness.dta

}
```

(13) Import the graph simulations that have been generated from folder `OUT/SIMULATION`

(14) Conduct economic estimation of interest. For instance,

$$y_{iv} = \alpha + \beta \frac{1}{B} \sum_{b=1}^{B} S(g)_{iv,b} + \epsilon_{iv},$$

to estimate β, which is the parameter of interest in this example, where i is a node and v is the independent network for $v = 1, ..., V$ networks in the sample.

```
**MERGE with Census data **

use `CENSUS' , clear

merge 1:1 id_hhid id_village using centrality.dta

reg y centrality_var , cluster(id_village)
```