BUS 41201 Homework 4 Assignment

Group 24: Shihan Ban, Yi Cao, Shri Lekkala, Ningxin Zhang

16 April 2024

Setup

```r
## microfinance network
## data from BANERJEE, CHANDRASEKHAR, DUFLO, JACKSON 2012

## data on 8622 households
hh <- read.csv("microfi_households.csv", row.names="hh")
hh$village <- factor(hh$village)

## We'll kick off with a bunch of network stuff.
## This will be covered in more detail in lecture 6.
## get igraph off of CRAN if you don't have it
## install.packages("igraph")
## this is a tool for network analysis
## (see http://igraph.sourceforge.net/)
library(igraph)
```

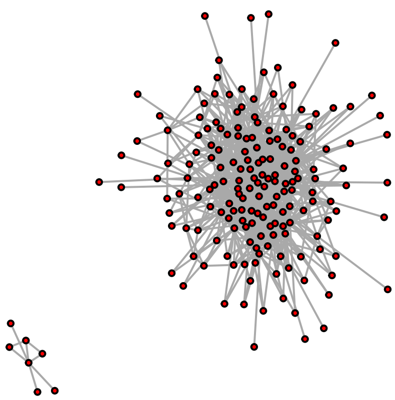
```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```r
edges <- read.table("microfi_edges.txt", colClasses="character")
## edges holds connections between the household ids
hhnet <- graph.edgelist(as.matrix(edges))
hhnet <- as.undirected(hhnet) # two-way connections.

## igraph is all about plotting.
V(hhnet) ## our 8000+ household vertices
```

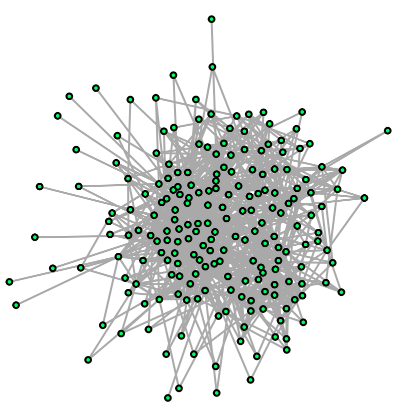
```
## + 8182/8182 vertices, named, from cb9a22e:
##    [1] 1002  1001  1020  1042  1053  1163  1003  1004  1026  1029  1076  1159
##   [13] 1106  1031  1048  1081  1006  1005  1008  1016  1021  1024  1089  1103
##   [25] 1007  1019  1155  1015  1040  1044  1045  1078  1088  1110  1115  1140
##   [37] 1145  1009  1018  1060  1064  1073  1153  1067  1099  1010  1162  1012
##   [49] 1143  1013  1023  1028  1034  1065  1117  1139  1154  1157  1173  1014
##   [61] 1068  1071  1148  1017  1036  1062  1112  1118  1120  1129  1134  1165
##   [73] 1183  1126  1122  1049  1058  1093  1108  1114  1119  1022  1043  1079
```

```
##    [85] 1033  1102  1104  1105  1152  1169  1171  1025  1027  1147  1032  1035
##    [97] 1037  1039  1041  1113  1174  1069  1116  1132  1178  1146  1080  1086
##   [109] 1101  1172  1059  1141  1142  1038  1094  1052  1092  1082  1095  1158
## + ... omitted several vertices
```

```r
## Each vertex (node) has some attributes, and we can add more.
V(hhnet)$village <- as.character(hh[V(hhnet),'village'])
## we'll color them by village membership
vilcol <- rainbow(nlevels(hh$village))
names(vilcol) <- levels(hh$village)
V(hhnet)$color = vilcol[V(hhnet)$village]
## drop HH labels from plot
V(hhnet)$label=NA

# graph plots try to force distances proportional to connectivity
# imagine nodes connected by elastic bands that you are pulling apart
# The graphs can take a very long time, but I've found
# edge.curved=FALSE speeds things up a lot.  Not sure why.

## we'll use induced.subgraph and plot a couple villages
village1 <- induced.subgraph(hhnet, v=which(V(hhnet)$village=="1"))
village33 <- induced.subgraph(hhnet, v=which(V(hhnet)$village=="33"))

# vertex.size=3 is small.  default is 15
plot(village1, vertex.size=3, edge.curved=FALSE)
```



```r
plot(village33, vertex.size=3, edge.curved=FALSE)
```



```r
library(gamlr)
```

```
## Loading required package: Matrix
```

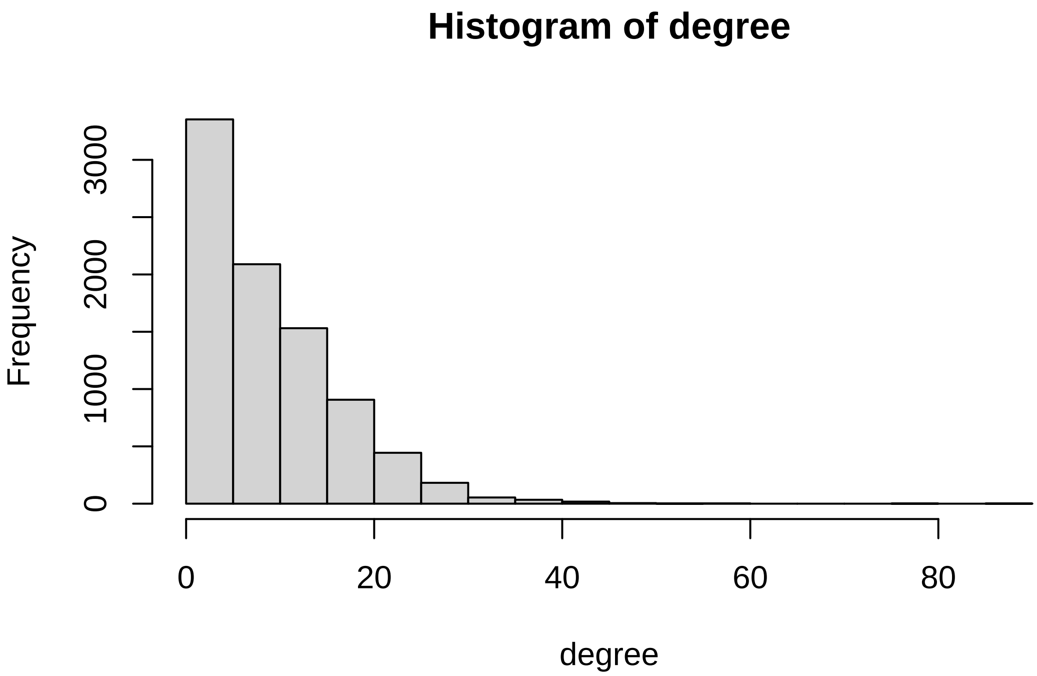
```r
## match id's; I call these 'zebras' because they are like crosswalks
zebra <- match(rownames(hh), V(hhnet)$name)

## calculate the `degree' of each hh:
##  number of commerce/friend/family connections
degree <- degree(hhnet)[zebra]
names(degree) <- rownames(hh)
degree[is.na(degree)] <- 0 # unconnected houses, not in our graph

## if you run a full glm, it takes forever and is an overfit mess
# > summary(full <- glm(loan ~ degree + .^2, data=hh, family="binomial"))
# Warning messages:
# 1: glm.fit: algorithm did not converge
# 2: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Question 1

I'd transform degree to create our treatment variable d. What would you do and why?

We can first plot a histogram of the degree variable to get an idea of it's structure:

```
hist(degree)
```



From the graph, it might be appropriate to perform a logarithmic transformation for the following reasons:

- It appears that the degree frequency is highly skewed to the right as there are many nodes with few connections (degree < 20), but few nodes with many connections (degree > 40). So by taking a log transformation, we can normalize the distribution, making it more symmetric and more suitable to statistical analyses.
- The histogram appears to follow an exponential / multiplicative relationship. So transforming the data logarithmically can make the relationship more linear, which is easier to model and interpret in regression models.
- We can reduce the range of variability in degree values, effectively performing a dimensionality reduction. This is useful to prevent the model being overly effected by outliers, i.e. households with a very high number of connections.

```
# Transform degree and add it to the hh dataset
hh$d = log1p(degree)
head(hh)
```

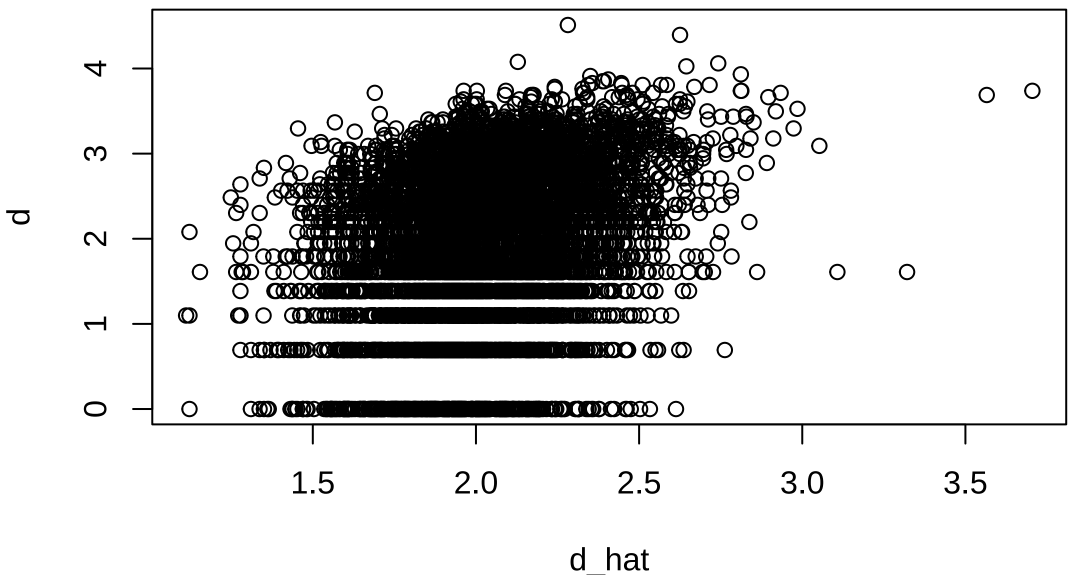
```
##      loan village religion  roof rooms beds electricity ownership leader
## 1001    0       1    hindu  tile     3    4           0     OWNED      0
## 1002    0       1    hindu  tile     1    1           1     OWNED      1
## 1003    0       1    hindu   rcc     3    4           1     OWNED      1
## 1004    0       1    hindu  tile     2    6           1     OWNED      0
```

```
## 1005    0       1    hindu  tile     3    4           1     OWNED      0
## 1006    0       1    hindu stone     2    1           1     OWNED      0
##             d
## 1001 1.791759
## 1002 2.079442
## 1003 1.098612
## 1004 1.609438
## 1005 2.197225
## 1006 2.302585
```

Question 2

Build a model to predict d from x, our controls.

```
# control variables
x = model.matrix(d ~ village + religion + roof + rooms + beds + electricity + ownership + leader - 1, da

# dependent variable
y = hh$loan

# treatment variable
d = hh$d

# Estimate d_hat with lasso regression of d on x.
treat = gamlr(x, d, lambda.min.ratio=1e-4)

# Isolate dhat (the part of treatment that we can predict with x's)
d_hat = predict(treat, x, type="response")

# Plot d_hat against d
plot(d_hat, d)
```



Comment on how tight the fit is, and what that implies for estimation of a treatment effect.

To assess the model fit, we can compute the in sample R2

```
# In-sample R2
cor(drop(d_hat), d)^2
```

```
## [1] 0.08187873
```

So the in-sample R2 value suggests that ≈ 8.19 % of the variance in d is explained by the control variables. Thus the model does not have a tight fit, and this implies that there may be other confounding variables not included in the model that account for d.

Thus the predictive power of our model is limited due to the large percentage of unexplained variance in d. And further analyses may lead to less accurate and biased estimates, so our confidence in a estimating treatment effect would be low.

Question 3

Use predictions from [2] in an estimator for effect of d on loan.

```r
# Second Stage Lasso

# Do a lasso of y on [d, d_hat, x], with d_hat unpenalized
causal = gamlr(cbind(d, d_hat, x), y, free=2)
```

```
## 'as(<dgeMatrix>, "dgCMatrix")' is deprecated.
## Use 'as(., "CsparseMatrix")' instead.
## See help("Deprecated") and help("Matrix-deprecated").
```

```r
# Second
print(coef(causal)["d",])
```

```
## [1] 0.0187176
```

Using the two-stage lasso process, we find the best predictor for y from d and x after the influence of d_hat is removed.

We observe that the coefficient of the log transformed degree variable is 0.0187176, which suggests that there is a small positive relationship between the degree of connectivity and the likelihood of adopting a loan.

```r
exp(coef(causal)["d",])
```

```
## [1] 1.018894
```

By taking the exponential of the coefficient, we compute the odds ratio between degree and loan. That is, a one unit increase in the log transformed degree of connection corresponds to an ≈ 1.89 % increase in the probability of a household taking a loan.

Question 4

Compare the results from [3] to those from a straight (naive) lasso for loan on d and x.

```
# Compute a naive lasso for loan
# We use binomial here since we know y (loan) is in [0,1]
naive = gamlr(cbind(d, x), y, family="binomial")

# Compare naive and 2-stage lasso
cat("The coefficient for d from the naive lasso is:", coef(naive)["d",], "\n")
```

```
## The coefficient for d from the naive lasso is: 0.1562872
```

```
cat("The coefficient for d from the causal lasso is:", coef(causal)["d",])
```

```
## The coefficient for d from the causal lasso is: 0.0187176
```

```
exp(coef(naive)["d",])
```

```
## [1] 1.169162
```

Explain why they are similar or different.

Firstly we observe that the coefficient for `d` from the naive model is approximately an order of 10 greater than the one from the causal model. That is, a one unit increase in log transformed degree would suggest there would be a an ≈ 16.9 % increase in the probability of a household taking a loan (compared to ≈ 1.89 % from Q3).

They are different as the naive model does not separate the treatment and the control variables, but rather uses them all as independent variables in the regression. This may result in the coefficient for `d` having contributions from confounding variables which are not accounted for, and thus indicates a much more significant effect than the causal model.

In comparison, the causal model involved a 2-stage LASSO process by incorporating **`d_hat`**, the predicted values of **`d`** based on **`x`**. This means that this model controlled for the portion of **`d`** that could be predicted from the control variables, aiming to isolate the more variation in d. (However there could still be effects from confounding variables that were not in the data set).

So the two-stage Lasso model provides a more conservative but likely more accurate estimate by explicitly modeling and removing the predictable part of **`d`** based on the observed covariates **`x`**.

Question 5

Bootstrap your estimator from [3] and describe the uncertainty.

```
## BOOTSTRAP
n <- nrow(x)

gamb = c() # empty gamma

for(b in 1:50){
    ## create a matrix of resampled indices
    ib = sample(1:n, n, replace=TRUE)

    ## create the resampled data
    xb = x[ib,]
    db = d[ib]
    yb = y[ib]

    ## run the treatment regression
    treatb = gamlr(xb,db,lambda.min.ratio=1e-3)
    dhatb = predict(treatb, xb, type="response")
    fitb = gamlr(cbind(db,dhatb,xb),yb,free=2)
    gamb = c(gamb,coef(fitb)["db",])
}
```

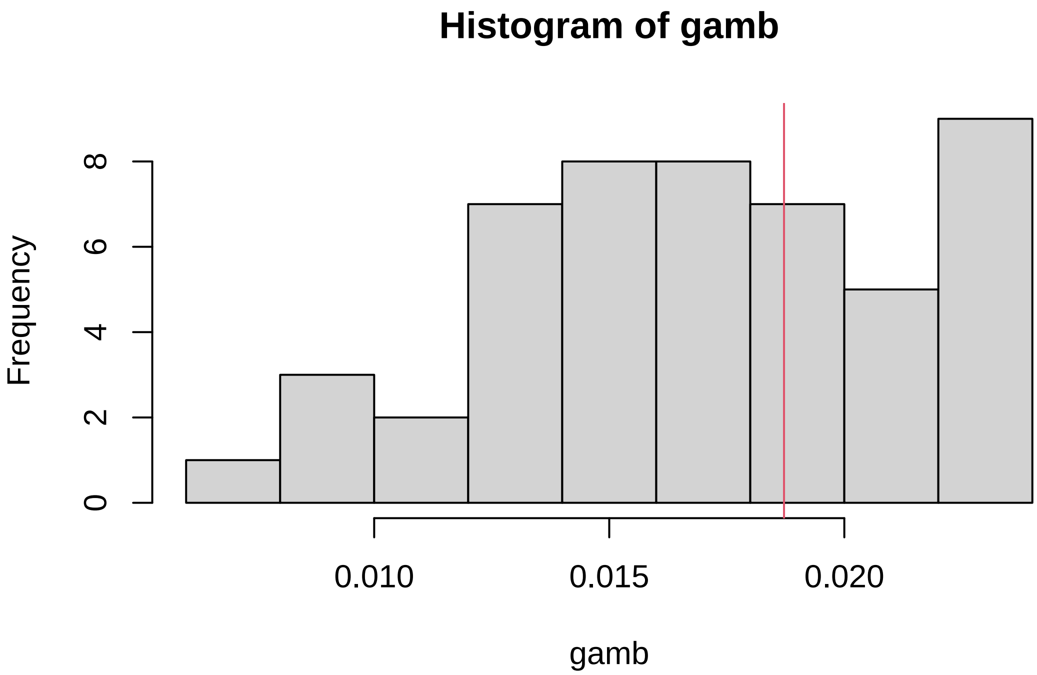
After running the bootstrap 50 times, we get the following summary statistics for the estimates:

```
summary(gamb)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.006419 0.013477 0.017018 0.016911 0.020340 0.023977
```

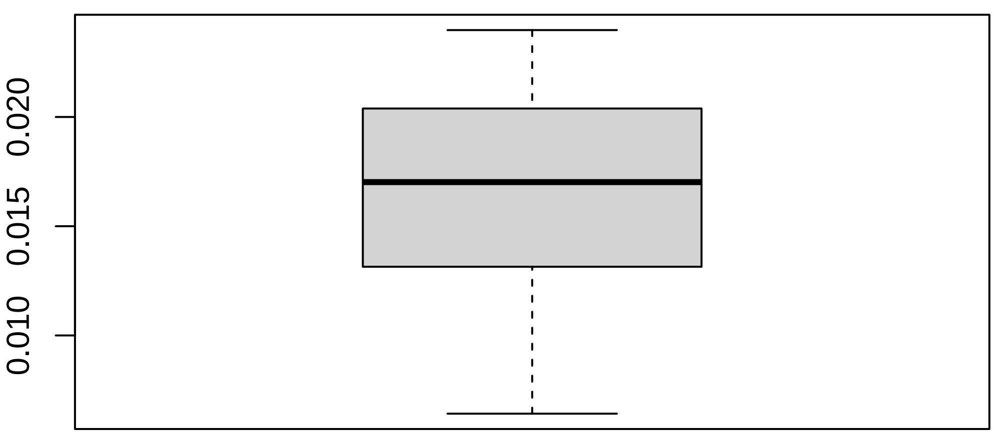
We can also plot a histogram to observe the variability in the estimates, with a red line for our original non-bootstrapped estimate:

```
hist(gamb)
abline(v=coef(causal)["d",], col=2)
```



We also plot a box plot and note the standard deviation:

```
boxplot(gamb)
```




```
cat("The standard deviation in estimates is: ", sd(gamb))
```

```
## The standard deviation in estimates is:  0.00454021
```

So after running the bootstrap 50 times, we observe that the median estimate is ≈ 0.0178, with a standard deviation of ≈ 0.005. So the bootstrap estimates seem to have fairly low variability which suggests that the estimate obtained this way is stable.

From the box-plot, we observe that the interquartile range is fairly narrow, which suggests that the estimates are centered around the median, and also suggests that the model prediction is robust.

Finally the histogram shows that the estimates seem to form a symmetric distribution, centered around the median. Further the original estimate from the full model is close to the center of the bootstrapped estimates which suggests that the original estimate was also robust.

[Bonus]

Can you think of how you'd design an experiment to estimate the treatment effect of network degree?

In order to design such as experiment, one would ideally create a randomized experiment where one randomly selected group of individuals would increase their network degree, and the others would not.

However, this is easier said than done, as in real life increasing one's connections or network is not as simple as just asking someone to do so, and such processes take considerable time. However, one possible alternative is to randomly assign certain households to receive targeted opportunities and encouragement to increase their network and social connections. This could be in the form of community events, participation in social groups, or general networking opportunities. The other households would not receive such opportunities and would be left as a control group.

Following the experiment, we could then remeasure the outcome of interest (loan uptake) and compare the outcomes between the two groups. However, one would have to make sure that the control group's network stayed similar and the target group's network did increase before forming any conclusions.

However, there a few concerns regarding such an experiment. Firstly there could be spillover effects, where the treatment affects the control group as there might be mutual connections between the two groups. In such cases it would be harder to measure the treatment effect. Further, there might be also ethical concerns that the experiment we perform is manipulating human relationships and interactions, which is a topic that requires some sensitivity. In addition, some may view it as unfair if some people in a village get more networking opportunities and encouragement whilst others do not have the same access.