Hye-Min Jung
Jeong Lim Kim
Jayoung Kang

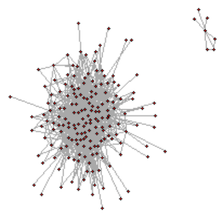
HW 4

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

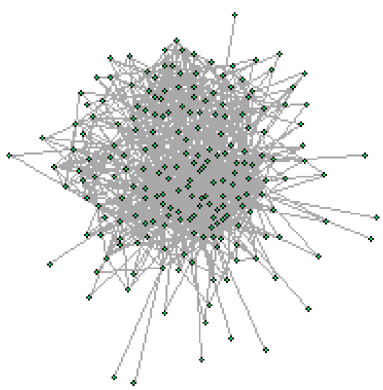
- Transformation: log(degree+1)
- In this way, I can normalize distribution, resolving the skewness in the degree data. The distribution of d becomes closer to the normal gaussian distribution and suitable for fitting linear regression. Instead of simple log transformation, log(x + 1) is used for values that contain 0. This enable us to avoid log transformation returning negative infinity.

```
hh <- read.csv("microfi_households.csv", row.names="hh")
hh$village <- factor(hh$village)

## We'll kick off with a bunch of network stuff.
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 c06095e:
##    [1] 1002  1001  1020  1042  1053  1163  1003  1004  1026  1029  1076
##   [12] 1159  1106  1031  1048  1081  1006  1005  1008  1016  1021  1024
##   [23] 1089  1103  1007  1019  1155  1015  1040  1044  1045  1078  1088
##   [34] 1110  1115  1140  1145  1009  1018  1060  1064  1073  1153  1067
##   [45] 1099  1010  1162  1012  1143  1013  1023  1028  1034  1065  1117
##   [56] 1139  1154  1157  1173  1014  1068  1071  1148  1017  1036  1062
##   [67] 1112  1118  1120  1129  1134  1165  1183  1126  1122  1049  1058
##   [78] 1093  1108  1114  1119  1022  1043  1079  1033  1102  1104  1105
##   [89] 1152  1169  1171  1025  1027  1147  1032  1035  1037  1039  1041
##  [100] 1113  1174  1069  1116  1132  1178  1146  1080  1086  1101  1172
## + ... omitted several vertices

## 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
```

```r
# 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)
```

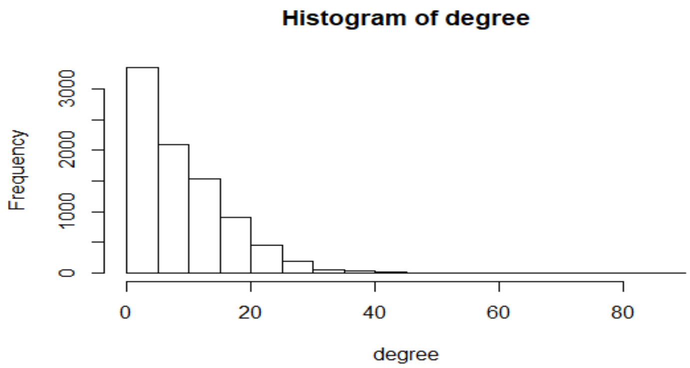


```r
plot(village33, vertex.size=3, edge.curved=FALSE)
```

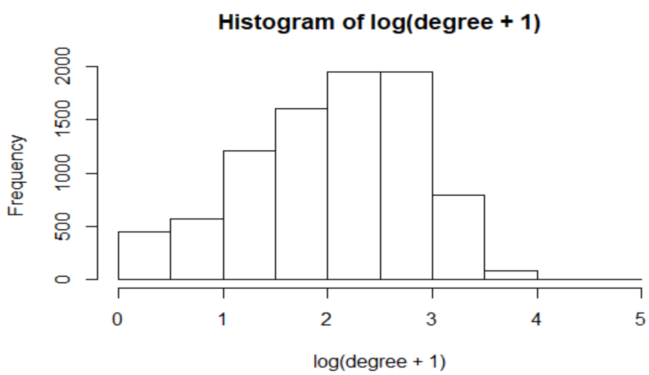


```r
#####  now, on to your homework stuff
## match id's; I call these 'zebras' because they are like crosswalks
zebra <- match(rownames(hh), V(hhnet)$name)

## 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
```

```r
# > 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

hist(degree)
```




```r
hist(log(degree+1))
```




```r
d<-log(degree+1)
```

[2]. Build a model to predict d from x, our controls. Comment on how tight the fit is, and what that implies for estimation of a treatment effect.

- R^2 using normal regression: 0.08223472
- R^2 using LASSO: 0.08187873
- Since R^2 is quite low, this indicates that the part of d that can be predicted with x's isn't very high. This implies that the treatment effect, when estimated without the controls will not be overestimated by too much, given that the controls we have are enough to account for the confounding effect.

```
#without LASSO
reg_dx <- glm(d ~ .-loan, data=hh)
summary(reg_dx)

1-5826.8/6348.9

## [1] 0.08223472



#With LASSO
x = sparse.model.matrix(~.-loan, data=hh)[,-1]
treat <- gamlr(x,d,lambda.min.ratio=1e-4)
dhat <- predict(treat, x, type="response")
cor(drop(dhat),d)^2

## [1] 0.08187873
```

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

```
dhat<-reg_dx$fitted.values

causal <- gamlr(cBind(d,dhat,x),hh$loan,free=2,lmr=1e-4)

coef(causal)["d",]

## [1] 0.01803462
```

[4]. Compare the results from [3] to those from a straight (naive) lasso for loan on d and x. Explain why they are similar or different.

- The results are similar because the dependent part of d wasn't large, so including x in the naïve LASSO would not change the estimate for the coefficient of d by too much.

```
naive <- gamlr(cBind(d,x),hh$loan)

coef(naive)["d",]

## [1] 0.01868003
```

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

- The standard error for the estimator is 0.00376866. This represents the uncertainty of the estimator because given ± one standard deviation from the estimator, about 68.3% of the time, the true value of the measured quantity falls within the stated uncertainty range.

```
y <- hh$loan
n <- nrow(x)

gamb <- c() # empty gamma

for(b in 1:20){
    ## 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",])

    print(b)
}

## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
```

```
## [1] 19
## [1] 20

summary(gamb)

##      Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.009246 0.014755 0.018455 0.017108 0.020190 0.021830

coef(causal)["d",]/sd(gamb)

## [1] 4.78542

se<-sd(gamb)
se

## [1] 0.00376866

{hist(gamb)
  abline(v=quantile(gamb,0.025),col=3,lwd=2)
  abline(v=quantile(gamb,0.975),col=3,lwd=2)}
```