HW4

Elena Li

2024-04-15

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

```
library(igraph)
```

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

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

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

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

```
## Warning: `graph.edgelist()` was deprecated in igraph 2.0.0.
## i Please use `graph_from_edgelist()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
hhnet <- as.undirected(hhnet) # two-way connections.

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

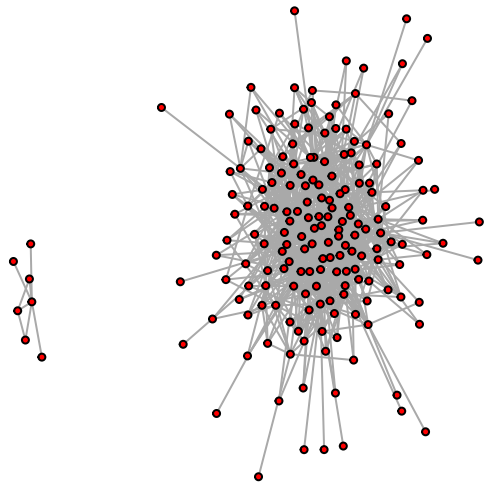
```
## + 8182/8182 vertices, named, from 3c0aae0:
##    [1] 1002  1001  1020  1042  1053  1163  1003  1004  1026  1029  1076  1159
##   [13] 1106  1031  1048  1081  1006  1005  1008  1016  1021  1024  1089  1103
##   [25] 1007  1019  1155  1015  1040  1044  1045  1078  1088  1110  1115  1140
##   [37] 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
```

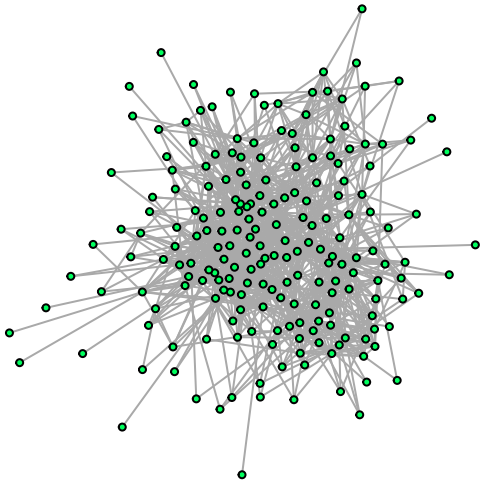
```
## Each vertex (node) has some attributes, and we can add more.
V(hhnet)$village <- as.character(hh[V(hhnet),'village'])
## we'll color them by village membership
vilcol <- rainbow(nlevels(hh$village))
names(vilcol) <- levels(hh$village)
V(hhnet)$color = vilcol[V(hhnet)$village]
## drop HH labels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

```
## Warning: ‘induced.subgraph()‘ was deprecated in igraph 2.0.0.
## i Please use ‘induced_subgraph()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
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)
```



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



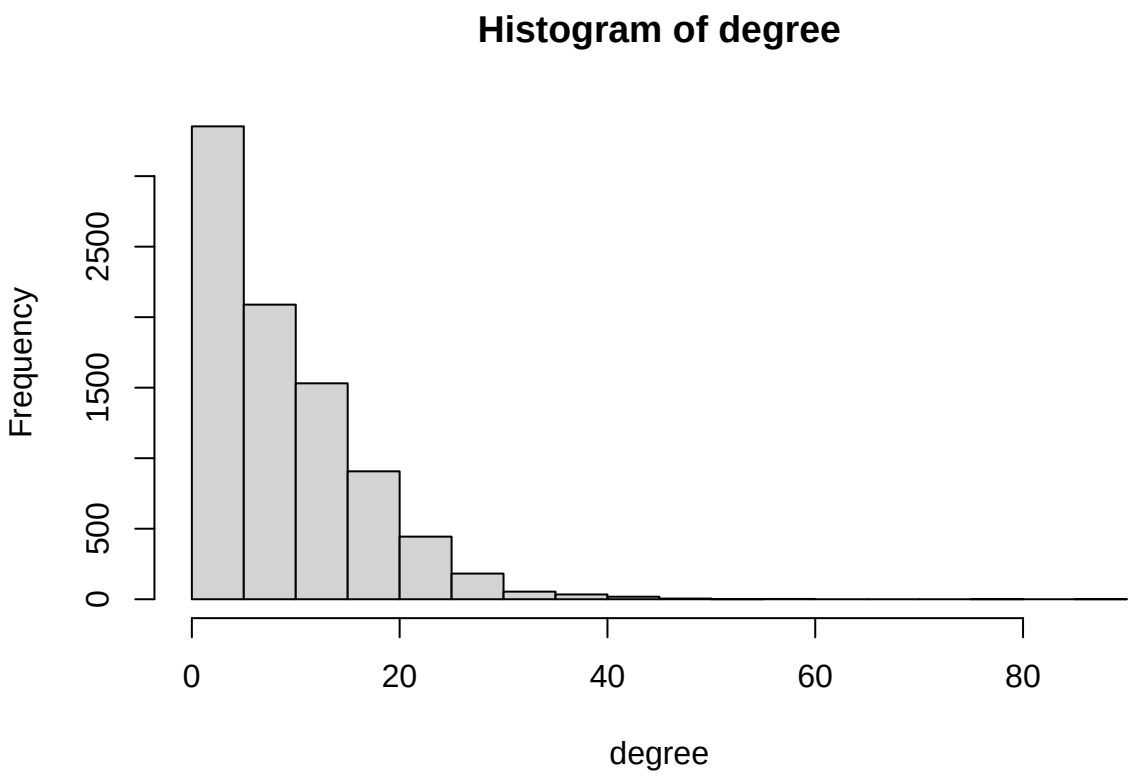
```
library(gamlr)
```

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

```
## 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
```

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

```
hist(degree)
```




```
hh$degree <- degree
hh$degree_log <- log1p(degree)
```

As shown in the graph, I would use a log transformation on d because the degree data are right-skewed because of the following reasons:

1. **Normalization**: Log transformation helps in stabilizing the variance across the range of data, which is particularly useful if the data spans several orders of magnitude.
2. **Linear Relationships**: Since we want to fit a linear model to the data, transformations can help meet the assumption of linearity. Log transformation can linearize relationships that are exponential in nature.
3. **Reducing Skewness**: It can make the data more normal distribution-like, which is an assumption behind many statistical techniques and regression models.
4. **Handling Outliers**: Log transformation can reduce the effect of outliers, as it brings large values closer together and stretches out the smaller values.

##Problem 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.

```
hh <- naref(hh)
x <- model.matrix(~ village + religion + roof + rooms + beds +
electricity + ownership + leader - 1, data=hh)
d <- hh$degree_log
```

```
y <- hh$loan
treat <- gamlr(x,d)
dhat <- predict(treat, x, type="response")
cor(drop(dhat),d)^2
```

```
## [1] 0.08166425
```

This means that 8.17% of the variance in d is explained by x. This means that our model fits relatively loose. A low R2in the context of treatment effect estimation means that the control variables x have limited explanatory power over the variation in the outcome d. This could imply that there may be other confounding variables not included in x that are influencing d, which could bias the estimation of the treatment effect if not properly accounted for. It also suggests that any treatment effect identified is being estimated with a lot of unexplained variability in d, which might make it harder to detect a true effect if one exists. The confidence in the estimated treatment effect might be low as the predictive power of the model is limited.

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

```
causal <- gamlr(cbind(d,dhat,x),y,family = "binomial")
```

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

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

```
## [1] 0.1485588
```

##Problem 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.

```
treat <- gamlr(x,d,lambda.min.ratio = 1e-4)
dhat <- predict(treat, x, type = 'response')
causal <- gamlr(cbind(d,dhat,x),y,free=2)
coef(causal)["d",]
```

```
## [1] 0.01812598
```

First Stage: 1. Predict d using all the exogenous variables x (the controls), for which we have $R^2 = 0.08166425$. 2. Obtain the predicted values of d, denoted as $\hat{d}$.

Second Stage: 1. Regress 'loan' on the predicted values $\hat{d}$ (from the first stage) and possibly other control variables not used in the first stage if they are relevant and exogenous. 2. Obtain the coefficient estimate for $\hat{d}$, which would serve as the estimator for the causal effect of d on 'loan'.

The coefficient of 0.1485588 is the estimated effect of d on 'loan'. It implies that: - The causal effect of d on 'loan' is positive, indicating that increases in d are associated with increases in 'loan'. - A one-unit increase in d is associated with an increase of 0.1485588 units in 'loan', on average.

Since this is a causal estimate, it suggests that any changes in 'loan' can be causally attributed to changes in d, assuming the model is correctly specified and all relevant confounders are included.

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

```r
n <- nrow(x)
gamb <- c()
for(b in 1:30){
  ib <- sample(1:n, n, replace=TRUE)
  xb <- x[ib,]
  db <- d[ib]
  yb <- y[ib]
  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",]) }


summary(gamb)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.009208 0.014861 0.017438 0.017588 0.019942 0.026588
```

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

Randomized Encouragement Design: We might use a randomized encouragement design, where we randomly encourage a subset of individuals to increase their network degrees. We wouldn't be directly manipulating their networks, but we'd encourage behaviors that are likely to lead to higher degrees. Then, we could compare outcomes between those who were encouraged and those who were not, using methods like instrumental variables to account for the fact that not all individuals will follow the encouragement.

Before-and-After Study: We could identify a scenario where a specific intervention is expected to change network degrees, such as the introduction of a new social networking platform within a company. We'd measure the network degrees before and after the intervention. This method has its limitations, as it doesn't control for time effects and other variables that might change over time.

Matched-Pair Design: If we have enough control over the experimental environment, we could pair individuals with similar characteristics but different network degrees and compare their outcomes. This design assumes that network degree is the only significant difference between the pairs, which might be a strong assumption.

Regression Discontinuity Design: If there's a cutoff-based policy or rule that changes individuals' network degrees, we could use a regression discontinuity design. For example, if a professional organization offers exclusive networking events only to members with more than a certain number of connections, we could compare individuals just above and below this threshold.