Analyze Grade Networks

Evan Green

2/7/2017

```
library(igraph)
library(stats4)
library(poweRlaw)
library(ggplot2)
library(reshape2)
library(dplyr)
library(MASS)
library(stargazer)
library(visreg)
library(randomForest)
library(RColorBrewer)
k_num_psets <- 7
nice_colors <- brewer.pal(5,"Set1")
setwd("/Users/evangreen/Desktop/Senior\ Thesis/Grade\ Data")

grades <- read.csv("FullGrades.csv", as.is=T, strip.white = TRUE)
genders <- read.csv("CodeGender.csv", as.is = T, strip.white=TRUE)
grades <- merge(genders,grades,by.x = "code",by.y = "student")

mylist <- list()
my_files <- paste("CollabPS", 1:7, ".csv", sep = "")
names_vec <- paste("CollabPS", 1:7, sep = "")
raw_collabs <- lapply(my_files, read.csv, as.is = T,strip.white = TRUE)
names(raw_collabs) <- names_vec
```

```
create_edge_list <- function(raw_file,cutoff = 20){
  #get rid
  raw_file$collabs <- unlist(sapply(raw_file$collabs,function(x)gsub("\\[|\\]", "", x)))
  edges <- as.data.frame(matrix(NA,ncol=2,nrow=100))
  colnames(edges) <- c("Source","Sink")
  edge_num <- 1
  over_reports <- 0
  for(student in unique(raw_file$student)){
    student_entries <- which(raw_file$student==student)
    sources <- raw_file[student_entries[length(student_entries)], "collabs"]
    sources <- strsplit(sources,split=", ")[[1]]
    num_collabs <- length(sources)
    if( num_collabs > cutoff){
      over_reports <- over_reports + 1
      next
    }else{
      for(elt in unique(sources)){
        if (elt != student & elt != 0 & student != 0){
          edges[edge_num, "Source"]<- trimws(elt)
          edges[edge_num, "Sink"] <- trimws(student)
          edge_num <- edge_num + 1
        }
```

```
      }
    }
  }
  edges <- edges[!is.na(edges$Source),]

  cat("There were", over_reports, "Over reports with a cutoff of", cutoff,"\n")
  return(edges)
}
```

```
#For people who dropped after 4,
#I will keep the edges that they are in but not analyze their grades
edge_lists <- lapply(raw_collabs,create_edge_list,100)


people_who_dropped <- grades[grades$did_drop_after4,1]
lapply(edge_lists, function(x) sum(x[,1] %in% people_who_dropped | x[,2] %in% people_who_dropped))


people_who_dropped <- grades[grades$did_drop_immediately,1]

remove_edge_mistakes <- function(edges, banned_people = people_who_dropped){
  column_1 <- edges[,1] %in% banned_people
  column_2 <- edges[,2] %in% banned_people
  return(edges[!column_1 & !column_2, ])
}
edge_lists <- lapply(edge_lists,remove_edge_mistakes)
create_graph_list <- function(edges){
  graphs <- list()
  for(i in 1:length(edges)){
    if(i <= 5){
      people_who_dropped <- grades[grades$did_drop_immediately,1]
    }else{
      people_who_dropped <- grades[grades$did_drop_after4,1]
    }
    graphs[[i]] <- graph.data.frame(edges[[i]],
                                    vertices=grades[!grades$code %in%people_who_dropped,])
  }
  return(graphs)
}
graphs <- create_graph_list(edge_lists)
```

```
qplot(x=Var1, y=Var2, data=melt(cor(grades[!grades$did_drop_after4 & grades$test1!=0 & grades$test2 != 
  scale_fill_gradient2(limits=c(-1, 1))

correlations <- cor(grades[!grades$did_drop_after4 & grades$test1!=0 & grades$test2 != 0,3:11])
barplot((colSums(correlations)-1)/8)
```

```
####Need to determine what the subset is for this
#might need to get rid of the 0's because they have too much influences
grades_2 <- read.csv("Grades_Imputed.csv",as.is = T,strip.white = T)
subset_for_grade_summary <- !grades_2$did_drop_after4
grades_2 <- grades_2[subset_for_grade_summary,]
grades_2[,grep("test",colnames(grades_2))] <- grades_2[,grep("test",colnames(grades_2))]/2
```

```r
grades_by_gender <- grades_2[,-1] %>%
  group_by(Sex) %>%
  summarise_if(is.numeric, funs(mean))

grades_by_gender_sd <- grades_2[,-1] %>%
  group_by(Sex) %>%
  summarise_if(is.numeric, funs(sd))

counts_by_gender <- grades_2 %>%
  group_by(Sex) %>%
  summarise(count = n())


grades_averages <- grades_2[,-1] %>%
  summarise_if(is.numeric, funs(mean,median))

plotTop <- 45
barplot(height = as.matrix(grades_by_gender[2:3,-1]),
        beside = TRUE, las = 1,
        col = nice_colors[1:2],
        ylim = c(0, plotTop),
        cex.names = 0.75,
        main = "Average Grades By Gender",
        ylab = "Grade",
        xlab = "Assignment",
        border = "black", axes = TRUE)
#add a legend
legend("topleft",pch = 15, col = nice_colors[1:2],
       c("Female","Male"),
       bty = "n")
top_score <- 30

#female error bars
x_women <- seq(from = 1.5, by = 3, length.out = ncol(grades_by_gender) - 1)
y_women <- as.matrix(grades_by_gender[2,-1])
offset <-  as.matrix(grades_by_gender_sd[2,-1] * 1.96)
arrows(x_women,top_score, x_women, y_women-offset,
       angle=90, code=3, length=.05)
#male
x_men <- seq(from = 2.5, by = 3, length.out = ncol(grades_by_gender) - 1)
y_men <- as.matrix(grades_by_gender[3,-1])
offset <-  as.matrix(grades_by_gender_sd[3,-1] * 1.96)
arrows(x_men,top_score, x_men, y_men-offset,
       angle=90, code=3, length=.05)


barplot(table(grades_2$Sex),col = nice_colors[c(3,1,2)],
        main = "Number of Students by Gender")
par(mfrow = c(3,3))
exclude_q <- grades_2$Sex!="?"
for(i in 1:9){
  boxplot(grades_2[exclude_q,i+2] ~ grades_2$Sex[exclude_q],main = colnames(grades_2)[i+2])
}
```

```r
grades_2
```

```r
#A lot of the edges are edges that appear 7 times, 48% of edges either 6 or 7

repeated_edges <- as.data.frame(matrix(NA,ncol=7,nrow = k_num_psets))
colnames(repeated_edges) <- 1:k_num_psets
rownames(repeated_edges) <- 1:k_num_psets

for(i in 1:length(edge_lists)){
  for(ii in 1:length(edge_lists)){
    repeated_edges[i, ii] <- sum(duplicated(rbind(edge_lists[[i]],
                                                   edge_lists[[ii]]))) /
      mean(c(nrow(edge_lists[[i]]),nrow(edge_lists[[ii]])))
  }
}
#average overlap
(mean(as.matrix((repeated_edges))) * 49 - 7)/42
#min overlap
min(repeated_edges)

heatmap(as.matrix(repeated_edges))
plot(as.matrix(repeated_edges)[seq(2,length.out = 6, by = 8)])
lines(as.matrix(repeated_edges)[seq(2,length.out = 6, by = 8)])

#Ties are not lost independently, people will go in and out relationships it seems but
#lets investigate a little more

students <- unique(grades$code)
possible_ties <- data.frame(expand.grid(students,students))
possible_ties[,3:(3+k_num_psets-1)] <- 0
colnames(possible_ties) <- c("Source","Sink",paste("hw",1:7,sep = ""))

charity_ties <- possible_ties

which_edges <- c()
for(i in 1:length(edge_lists)){
  which_edges <- append(which_edges,apply(edge_lists[[i]],1, paste, collapse = "->",sep = ""))
  for(tie in 1:nrow(edge_lists[[i]])){
    source_std <- edge_lists[[i]]$Source[tie]
    sink_std <- edge_lists[[i]]$Sink[tie]
    col_1 <- possible_ties$Source == source_std
    col_2 <- possible_ties$Sink == sink_std
    possible_ties[col_1 & col_2,2 + i] <- 1
    col_3 <- edge_lists[[i]]$Source == sink_std
    col_4 <- edge_lists[[i]]$Sink == source_std
    #check if charity
    if(!any(col_3 & col_4)){
      if(grades[grades$code == source_std, 2 + i  ] > grades[grades$code == sink_std, 2 + i]){
        charity_ties[col_1 & col_2,2 + i] <- 1
      }
    }
  }
}
#remove ties that never occur but keep track of the percentage
```

```r
possible_ties <- possible_ties[!possible_ties$Source == possible_ties$Sink,]
mean(rowSums(possible_ties[,3:9])==0)
#97% of possible ties never occur
charity_ties <- charity_ties[rowSums(possible_ties[,3:9])!=0,]
possible_ties <- possible_ties[rowSums(possible_ties[,3:9])!=0,]

mean(rowSums(possible_ties[,3:9]))
#Average tie exists 3 times once it occurs

#just a check that I'm doing it right
barplot(table(rowSums(possible_ties[,3:9])) * 1:7)

#Takes in a row and sees how many times it changes from 0 to 1
x <- possible_ties[1,3:9]
count_switches <- function(x){
  return(sum(x[-length(x)] != x[-1]))
}
possible_ties$Switches <- apply(possible_ties[,3:9],1,count_switches)
table(possible_ties$Switches)
possible_ties$Count <- rowSums(possible_ties[,3:9])
table(possible_ties$Switches,possible_ties$Count)
tapply(possible_ties$Switches,possible_ties$Count,mean)

#now lets get some expectations for randomly generated data
nTrials <- 10 ** 4
keep_track <- matrix(nrow = 6, ncol = nTrials)
keep_track_first <- matrix(nrow = 6, ncol = nTrials)
keep_track_last <- matrix(nrow = 6, ncol = nTrials)
for(count in 1:6){
  for(i in 1:nTrials){
    trial <- sample(c(rep(1,count),rep(0,k_num_psets - count)),replace = F)
    keep_track[count, i] <- count_switches(trial)
    info <- which(trial == 1)
    keep_track_first[count, i] <-info[1]
    keep_track_last[count, i] <- info[length(info)]
  }
}
simul <- rowMeans(keep_track)
apply(keep_track,1,sd)
table(possible_ties$Switches,possible_ties$Count)
real <- tapply(possible_ties$Switches,possible_ties$Count,mean)
barplot(rbind(simul,real[-7]),beside = T,col = nice_colors[1:2],
        main = "Mean Number of Times Collaboration Relationship Changes",
        xlab = "Number of Collaborations over the Semester")
legend("topright",pch = 15,
       col = nice_colors,
       c("Expected","Observed"),
       bty = "n")

#lower switching than would be expected by chance
#Also while it is symmetric around 3.5 in expectation,
#There is a different in 3 v. 4 in real life but not for the other symmetric
#pairs
```

```r
#don't see an effect for 1 and 6 showing that those ties are not over represented
#in the first or last problem set

#difference to not appear to be significant but them being less would
#make sense


edges_num_appeared <- table(table(which_edges))
edges_num_appeared_accounted_for <- edges_num_appeared * 1:k_num_psets

barplot(edges_num_appeared)
barplot(edges_num_appeared_accounted_for)

edges_num_appeared_pct<-round(edges_num_appeared_accounted_for / sum(edges_num_appeared_accounted_for),

barplot(edges_num_appeared_pct)
pie(edges_num_appeared_accounted_for)


###This is also interesting for other numbers ####

num_times_collaboratored <- 7
edges_seven_times <- table(which_edges)
edges_seven_times <- names(edges_seven_times)[edges_seven_times >= num_times_collaboratored]

is_reciprocal <- function(edge, edge_list = edges_seven_times){
  edge_comp <- strsplit(edge,"->",fixed = T)[[1]]
  edge_comp <- paste(rev(edge_comp), collapse = "->")
  return(edge_comp %in% edge_list)
}
edges_seven_times_reciprocal <- sapply(edges_seven_times,is_reciprocal)
sum(edges_seven_times_reciprocal) / 2 #because it occurs two times
sum(!edges_seven_times_reciprocal)
```

```r
#One thing to look at could be if the people who are helped more than they help
#do worse than those who help them, especially on tests
#Now, the question is why do relationships not change as much as random chance?
possible_ties$First <- apply(possible_ties[,3:9],1,function(x) which( x == 1)[1])
tapply(possible_ties$First,possible_ties$Count,mean)
rowMeans(keep_track_first)


simul_first <- rowMeans(keep_track_first)
apply(keep_track_first,1,sd)
real_first <- tapply(possible_ties$First,possible_ties$Count,mean)
barplot(rbind(simul_first,real_first[-7]),beside = T,col = nice_colors[1:2],
        main = "Mean Time when First Collaboration Happens",
        xlab = "Number of Collaborations over the Semester")
legend("topright",pch = 15,
       col = nice_colors,
       c("Simulated","Observed"),
       bty = "n")
```

```r
possible_ties$Last <- apply(possible_ties[,3:9],1,function(x) which( x == 1)[sum(x==1)])
real_last <- tapply(possible_ties$Last,possible_ties$Count,mean)

table(possible_ties$Last,possible_ties$Count)

simul_last <- rowMeans(keep_track_last)
barplot(rbind(simul_last,real_last[-7]),beside = T,col = nice_colors[1:2],
        main = "Mean Time when Last Collaboration Happens",
        xlab = "Number of Collaborations over the Semester")


#So collaborations are starting later, but not finishing earlier
```

```r
#Need to create a data set that includes information about each tie
#that can be used to help understand the network
# Potential Features
#
# Percent of time that tie has been there
# What percent of that time it was reciprocal
# Grade when we worked together, if applicable, for the sink
# f->f
# f->m
# m->f
# m->m
# Is it closing some transitivity
# Average Degrees of source
# Average degrees of sink
# centrality of source and sink (betweenness, eigen, constraint)
# Avg Grade of source
# avg grade of sink
# Member of the largest component
# count of number of people this person has an in_degree with
# count of number of peopel this person has an out_degree with

grades_for_ties <- grades
grades_for_ties[,grepl("hw",colnames(grades_for_ties))] <- grades_for_ties[, grepl("hw",colnames(grades_


#need to create a data set that includes information about each tie
possible_ties$Gender_Sink <- sapply(possible_ties$Sink, function(x) grades$Sex[grades$code == x ])
possible_ties$Gender_Source <- sapply(possible_ties$Source, function(x) grades$Sex[grades$code == x ])
possible_ties$Tie_Class <- paste(possible_ties$Gender_Source, possible_ties$Gender_Sink,
                                 sep = "->")


column_names <- colnames(possible_ties[,3:9])



get_collaborators <- function( student,ties = possible_ties){
  sources <- ties$Source[ties$Sink == student]
  sinks <- ties$Sinks[ties$Source == student]
  full_collabs <- union(sources,sinks)
  return(full_collabs)
```

```r
}


possible_ties$Recip_pct <- 0
possible_ties$Transitivity_PCT <- 0
possible_ties$Charity_PCT <- 0
for(i in 1:nrow(possible_ties)){
  possible_ties$Grades_Sink_Collab[i] <- mean(as.numeric(grades_for_ties[grades_for_ties$code == possibl
  possible_ties$Grades_Source_Collab[i]  <- mean(as.numeric(grades_for_ties[grades_for_ties$code == poss
  possible_ties$Grades_Source[i] <- mean(as.numeric(grades_for_ties[grades_for_ties$code == possible_ti
  possible_ties$Grades_Sink[i] <-  mean(as.numeric(grades_for_ties[grades_for_ties$code == possible_tie


  recip_row <- possible_ties$Source == possible_ties$Sink[i] & possible_ties$Sink == possible_ties$Sour
  possible_ties$Reciprocal[i] <- sum(recip_row)
  if(possible_ties$Reciprocal[i] == 1){
    possible_ties$Recip_pct[i] <- mean(as.numeric(possible_ties[recip_row,column_names[possible_ties[i,
  }
  possible_ties$Transitivity_PCT[i] <- length(intersect(get_collaborators(possible_ties$Source[i]),
                                                    get_collaborators(possible_ties$Sink[i]))) /
                                             length(union(get_collaborators(possible_ties$Source[i]),
                                                   get_collaborators(possible_ties$Sink[i])))
  possible_ties$Charity_PCT[i] <- mean(charity_ties[i,3:9][possible_ties[i,3:9]==1])

}
```

```r
possible_ties[possible_ties$Tie_Class=="m->?",]
possible_ties$Same_Gender <- possible_ties$Tie_Class %in% c("m->m","f->f")
possible_ties$Util_PCT <- possible_ties$Count / (8 - possible_ties$First)
lm1 <- lm(data = possible_ties,
   Count ~ Reciprocal + Recip_pct +Same_Gender+ Transitivity_PCT)
#summary(lm1)
#t.test(possible_ties$Count ~ possible_ties$Reciprocal)
lm1<-stepAIC(lm1, trace = 0)

#I(Grades_Sink - Grades_Sink_Collab) +
#   I(Grades_Source - Grades_Source_Collab + Transitivity_PCT)
lm1 <- lm(data = possible_ties,
   Count ~ Same_Gender + Reciprocal + Recip_pct +I(Grades_Sink > Grades_Sink_Collab) +
     I(Grades_Source > Grades_Source_Collab) + Grades_Source + Grades_Sink
    + Grades_Source_Collab +Grades_Sink_Collab +I(Transitivity_PCT>=.25) +Transitivity_PCT +Charity_PCT
#summary(lm1)
lm2<- stepAIC(lm1,trace = 0)
summary(lm2)
visreg(lm2)


plot(jitter(possible_ties$Count)~ jitter(possible_ties$Transitivity_PCT),col = rgb(0,0,0,.5))
local_ave <- ksmooth(possible_ties$Transitivity_PCT,possible_ties$Count)
points(local_ave$x,local_ave$y,col="red")
```

```r
stargazer(lm2)
```

```
######### adapted From analysis_advice_networks.R from Hirokazu Shirado########

is_same_gender <- function(p1, p2,info = grades){
  return(info$Sex[info$code == p1] == info$Sex[info$code == p2])
}

count_same_gender <- function(g){
  elist <- get.edgelist(g)
  num = 0
  for (i in 1:ecount(g)){
    if (is_same_gender(elist[i, 1], elist[i, 2])){
      num = num + 1
    }
  }
  return(num)
}



random_add_links <- function(g, num_add){
  num = 0
  while (num < num_add){
    s <- sample(1:vcount(g), 2)
    if (are_adjacent(g, s[1], s[2]) == FALSE){
      g <- add_edges(g, s)
      num = num + 1
    }
  }
  return(g)
}
```

```
#Do this with only people who remained in the class the whole time
edge_lists <- lapply(raw_collabs,create_edge_list,100)
people_who_dropped <- grades[grades$did_drop_after4,1]
edge_lists <- lapply(edge_lists,remove_edge_mistakes)
graphs <- lapply(edge_lists, graph.data.frame,
                 vertices=grades[!(grades$code %in%people_who_dropped),])
assort_by_pset <- sapply(graphs,function(x) assortativity_nominal(x, as.factor(V(x)$Sex),directed = T))
plot(assort_by_pset)
lines(1:k_num_psets,assort_by_pset)


###Figure out how many edges I need to add #####
#also from analysis_advice_networks.R
diff_begin_to_end <- graphs[[k_num_psets]] %m% graphs[[1]]
#the "core" network
core_network <- graphs[[1]] %m% (graphs[[1]] %m%  graphs[[k_num_psets]])



count_same_gender(graphs[[1]] %m% graphs[[k_num_psets]])
ecount(graphs[[1]] %m% graphs[[k_num_psets]])

same_gender_edges <- count_same_gender(diff_begin_to_end)
```

```r
total_new_edges <- ecount(diff_begin_to_end)

####test if the assortivity increase could happen by chance ####
num_trials <- 10 ** 4 #change to 4 if actually running
assort_trials <- numeric(num_trials)
same_gender_trials <- numeric(num_trials)
for(trial in 1:num_trials){
  g_trial <- random_add_links(core_network, ecount(diff_begin_to_end))
  assort_trials[trial] <- assortativity_nominal(g_trial, as.factor(V(g_trial)$Sex))
  gad <- g_trial %m% core_network
  same_gender_trials[trial] <- count_same_gender(gad)
  if(trial %% 500 == 0){
    print(trial)
  }
}


hist(assort_trials,main = "Gender Assortativity in Random Graphs",
     breaks =25, xlim = c(range(assort_trials)[1],
                          max(max(assort_trials),assort_by_pset[k_num_psets])+.03),
     xlab = "Assortativity",
     col = brewer.pal(5,"Pastel1")[2])
#abline(v = assort_by_pset[1])
text(x=assort_by_pset[k_num_psets], y=560,
     labels='Assortativity\non last homework', col='blue')
arrows(x0=assort_by_pset[k_num_psets],
       y0=490, x1=assort_by_pset[k_num_psets], y1=20,
       col='blue', length=0.1, lwd=3)
mean(assort_trials>=assort_by_pset[k_num_psets])

hist(same_gender_trials,main = "Number of New Edges of the Same Gender in Random Graphs",
     breaks = 25, xlim = range(same_gender_trials) + 3*c(-1,1))
abline(v = same_gender_edges)
#This is highly significant but not completely unreasonable
mean(same_gender_trials>same_gender_edges)
```

```r
#question to answer: Are Women more likely to help people who are worse than them,
#this would confirm the Kuhn and Villeval study that showed women respond inequity
#If someone gets help from someone else on a problem set that is unreciprocated
#that is an act of charity.
#Which gender is more likely to give charity?


num_hws <- length(edge_lists)

get_gender<- function(student, info = grades){
  return(info$Sex[info$code == student])
}

is_charity <- function(row, edge_list, info = grades, hw_num=1){
  source <- row[1]
  sink <- row[2]
  col_name <- paste("hw", hw_num, sep = "")
```

```r
  source_gender <- get_gender(source)
  sink_gender   <- get_gender(sink)

  message <- paste(source_gender, "->", sink_gender,sep = "")

  if(info[info$code == source,col_name] >= info[info$code == sink,col_name]){
    #check if it reciprocal
    if(sum(edge_list$Sink == source & edge_list$Source == sink)==0){
      message <- paste(message,"_Charity",sep = "")
      #This is charity
    }
  }
  return(message)
}

edge_types <- list(k_num_psets)
for(i in 1:k_num_psets){
  edge_types[[i]]<-apply(edge_lists[[i]],1,is_charity,edge_list = edge_lists[[i]],hw_num = i)
}



lapply(edge_types,table)


edge_types_df <- as.data.frame(matrix(0,ncol = 10+4, nrow = length(edge_lists)))
colnames(edge_types_df) <- union(names(table(edge_types[[6]])),names(table(edge_types[[5]])))
for(i in 1:length(edge_lists)){
  this_table <- table(edge_types[[i]])
  edge_types_df[i,names(this_table)] <- this_table
}
######look at this thing #######
#8 types of edges look at this
#looks like men get more chaiity

#get rid of the non-binary student
edge_types_df <- edge_types_df[,-c(5,10)]

colnames(edge_types_df)[9:12] <- paste(colnames(edge_types_df)[c(1,3,5,7)],
                                       "proportions", sep= "_")
edge_types_df <- rbind(edge_types_df, colSums(edge_types_df))
for(i in 1:nrow(edge_types_df)){
  for(ii in 1:4){
    edge_types_df[i, ii + 8] <- edge_types_df[i,ii*2] / (edge_types_df[i,ii * 2] + edge_types_df[i,ii *
  }
}

num_rows <- nrow(edge_types_df)

conf <- NULL
a <- list()
for(i in 1:4){
  col_num <- 2 * i
```

```r
  a[[i]] <- rep(0, sum(edge_types_df[num_rows,(col_num-1):(col_num)]))
  a[[i]][1:edge_types_df[num_rows,(col_num)]] <- 1
  ff<-t.test(a[[i]])
  if(is.null(conf)){
    conf <- ff$conf.int[1:2]
  }else{
    conf <- rbind(conf,ff$conf.int[1:2])
  }

}



barplot(as.matrix(edge_types_df[num_rows, 9:12]),ylim = c(0,.5),
        names.arg = c("f->f","f->m","m->f","m->m"),col = nice_colors[2:4],
        main ="Proportion of Ties that are Charitable")
height <- as.matrix(edge_types_df[num_rows, 9:12])
x_plot <- seq(from = .7, by = 1.2, length.out = 4)
arrows(x_plot,conf[,2], x_plot, conf[,1],
       angle=90, code=3, length=.05)

p_vals <-c(t.test(a[[1]],a[[2]])$p.value
           ,t.test(a[[2]],a[[3]])$p.value
           ,t.test(a[[4]],a[[3]])$p.value
           ,t.test(a[[1]],a[[3]])$p.value
           ,t.test(a[[1]],a[[4]])$p.value
           ,t.test(a[[2]],a[[4]])$p.value,
           t.test(c(a[[1]],a[[4]]),c(a[[2]],a[[3]]))$p.value,
           t.test(c(a[[1]],a[[2]]),c(a[[3]],a[[4]]))$p.value)

names(p_vals) <-c("fm v. ff", "fm v mf","mf v mm",
                  "ff v mf","ff v mm","fm v mm",
                  "same v. different","female v. male")


p.adjust(p_vals, method = "holm", n = length(p_vals))
p.adjust(p_vals, method = "holm", n = length(p_vals))[
  p.adjust(p_vals, method = "holm", n = length(p_vals))<.05]
#a couple things are significant
```

```r
edge_counts <- lapply(edge_lists,function(x)
  apply(x,1,function(y) sum(paste(y,collapse = "->") == which_edges)))

######check if charity is more likely to be from rare ties#####
counts_df <- NULL
for(i in 1:k_num_psets){
  next_one <- table(edge_types[[i]],edge_counts[[i]])
  if(is.null(counts_df)){
    counts_df <- next_one
  }else{
    counts_df <- rbind(counts_df,next_one)
  }
}
counts_df <- as.data.frame.array(counts_df)
```

```r
counts_df$Total <- rowSums(counts_df)
counts_df$Score <- apply(counts_df,1,function(x) sum(x[1:k_num_psets] * 1:k_num_psets) /
                           sum(x[1:k_num_psets]))

counts_df$is_charity <- grepl("Charity",rownames(counts_df))

score_non_charity <- sum(counts_df$Total[!counts_df$is_charity] *
                           counts_df$Score[!counts_df$is_charity]) /
  sum(counts_df$Total[!counts_df$is_charity])

score_charity <- sum(counts_df$Total[counts_df$is_charity] *
                       counts_df$Score[counts_df$is_charity]) /
  sum(counts_df$Total[counts_df$is_charity])

charity <- c()
normal <- c()
for(i in 1:nrow(counts_df)){
  for(j in 1:k_num_psets){
    if(grepl("Charity",rownames(counts_df)[i])){
      charity <- append(charity,rep(j-1,counts_df[i,j]))
    }else{
      normal <- append(normal,rep(j-1,counts_df[i,j]))
    }
  }
}
t.test(charity,normal)

hist(charity,ylim = c(0,1),freq = F,xlim = c(0,7))
hist(normal+.1,add = T,col="red",freq = F)
barplot(table(charity)/sum(table(charity)),col = rgb(0,0,1,.5),ylim = c(0,.5),
        main = "Number of Times a Connection Reappears")
barplot(table(normal)/sum(table(normal)),add = T,col = rgb(1,0,0,.5))
legend("topleft",
       pch = 15,
       col = c(rgb(0,0,1,.5),rgb(1,0,0,.5)),
       c("Charity","Not Charity"),
       bty = "n")
```

```r
# Do i want to do this with everyone in the netowrk or exclude
#those who dropped after 4 or smartly exclude them?
graphs <- create_graph_list(edges = edge_lists)

columns <- c("NVertices","NEdges","Density","Transitivity","Reciprocity","Assortivity",
             "Diameter","NumLeaves",
             "Components","NumLonely","NBiggest","NumWomen","NumMen","NumUnknown",
             "WomensInDegree","WomensOutDegree","MensInDegree","MensOutDegree")
graphProperties <- data.frame(matrix(NA, nrow = k_num_psets, ncol = length(columns)) )
colnames(graphProperties) <- columns

degree_by_gender_indiv <-rep( list(list()), 4 )
names(degree_by_gender_indiv) <- c("Women's in","Women's Out","Men's In","Men's Out")

for(fileNum in 1:k_num_psets){
for(col in columns){
```

```
  if(col == "NVertices"){
    statToAdd <- length(V(graphs[[fileNum]]))
  }else if(col == "NEdges"){
    statToAdd <- length(E(graphs[[fileNum]]))
  }else if(col == "Density"){
    statToAdd <- graph.density(graphs[[fileNum]])
  }else if(col =="Transitivity"){
    statToAdd <- transitivity(graphs[[fileNum]])
  }else if(col == "Reciprocity"){
    statToAdd <- reciprocity(graphs[[fileNum]])
  }else if(col =="Assortivity"){
    statToAdd <- assortativity.nominal(graphs[[fileNum]],
                                       types = as.factor(V(graphs[[fileNum]])$Sex))
  }else if(col == "Components"){
    statToAdd <- components(graphs[[fileNum]])$no
  }else if(col == "NumLonely"){
    statToAdd <- sum(components(graphs[[fileNum]])$csize == 1)
  }else if(col == "NBiggest"){
    statToAdd <- max(components(graphs[[fileNum]])$csize)
  }else if (col == "NumWomen"){
    statToAdd <- sum(V(graphs[[fileNum]])$Sex == "f")
  }else if (col == "NumMen"){
    statToAdd <- sum(V(graphs[[fileNum]])$Sex == "m")
  }else if (col == "NumUnknown"){
    statToAdd <- sum(V(graphs[[fileNum]])$Sex == "?")
  }else if (col == "WomensInDegree"){
    intermed <- degree(graphs[[fileNum]],mode = "in")[V(graphs[[fileNum]])$Sex == "f"]
    degree_by_gender_indiv$`Women's in` <- append(degree_by_gender_indiv$`Women's in`, intermed)
    statToAdd <- mean(intermed)
  }else if (col == "WomensOutDegree"){
    intermed <- degree(graphs[[fileNum]],mode = "out")[V(graphs[[fileNum]])$Sex == "f"]
    degree_by_gender_indiv$`Women's Out` <- append(degree_by_gender_indiv$`Women's Out`, intermed)
    statToAdd <- mean(intermed)
  }else if (col == "MensInDegree"){
    intermed <- degree(graphs[[fileNum]],mode = "in")[V(graphs[[fileNum]])$Sex == "m"]
    degree_by_gender_indiv$`Men's In` <- append(degree_by_gender_indiv$`Men's In`, intermed)
    statToAdd <- mean(intermed)
  }else if (col == "MensOutDegree"){
    intermed <- degree(graphs[[fileNum]],mode = "out")[V(graphs[[fileNum]])$Sex == "m"]
    degree_by_gender_indiv$`Men's Out` <- append(degree_by_gender_indiv$`Men's Out`, intermed)
    statToAdd <- mean(intermed)
  }else if (col == "Diameter"){
    statToAdd <- diameter(graphs[[fileNum]], directed=T)
  }else if (col == "NumLeaves"){
    out <- degree(graphs[[fileNum]],mode = "out")
    inD <- degree(graphs[[fileNum]],mode = "in")
    statToAdd <- sum(out == 0 & inD  > 1 )
  }
  graphProperties[fileNum,col] <- statToAdd
}
}

for(col in columns){
```

```
  plot(graphProperties[,col],main=col,
       ylab = col,
       xlab = "Assignment Number",
       col = nice_colors[4],
       pch=16,
       cex=.1)
  lines(graphProperties[,col],col=nice_colors[1],lwd=2)
}

for(col in grep("OutDegree",columns,value=T)){
  if(which(grep("OutDegree",columns,value=T)==col)==1){
    plot(graphProperties[,col],main="Out-Degree By Gender (Women == Red)",
         ylab = col,
         col = "red")
    abline(h = mean(graphProperties[,col]),col="red")
    lines(graphProperties[,col],col="red")
  }else{
    lines(graphProperties[,col],col=4)
    abline(h = mean(graphProperties[,col]),col=4)
  }
}

for(col in grep("InDegree",columns,value=T)){
  if(which(grep("InDegree",columns,value=T)==col)==1){
    plot(graphProperties[,col],main="InDegree By Gender (Women == Red)",
         ylab = col,
         col = "red",
         ylim = c(1.5,2.5))
    abline(h = mean(graphProperties[,col]),col="red")
    lines(graphProperties[,col],col="red")
  }else{
    lines(graphProperties[,col],col=4)
    abline(h = mean(graphProperties[,col]),col=4)
  }
}


degree_by_gender_indiv <- lapply(degree_by_gender_indiv, as.numeric)
hist(degree_by_gender_indiv$`Women's Out`, freq = F,
     xlim = range(degree_by_gender_indiv[names(degree_by_gender_indiv) %in% c("Women's Out", "Men's Out
     ylim = c(0,.55),col = rgb(0,0,1,.1), breaks = 8)
hist(degree_by_gender_indiv$`Men's Out`, freq = F, add  =T, col = rgb(1,0,0,.5),breaks = 20)
legend("topright",pch = 15,
       col = c(rgb(0,0,1,.1),rgb(1,0,0,.5)),
       c("Female","Male"),
       bty = "n")

hist(degree_by_gender_indiv$`Women's in`, freq = F,
     xlim = range(degree_by_gender_indiv[names(degree_by_gender_indiv) %in% c("Women's in", "Men's In")]
     ylim = c(0,.7),col = rgb(0,0,1,.1), breaks = 5)
hist(degree_by_gender_indiv$`Men's In`, freq = F, add  =T,
     col = rgb(1,0,0,.5),breaks = 5)
legend("topright",pch = 15, col = c(rgb(0,0,1,.1),rgb(1,0,0,.5)),
```

```r
       c("Female","Male"),
       bty = "n")

p.adjust(c(t.test(degree_by_gender_indiv[[1]],
                  degree_by_gender_indiv[[3]])$p.val,
t.test(degree_by_gender_indiv[[2]],
       degree_by_gender_indiv[[4]])$p.val),"holm")

lapply(degree_by_gender_indiv,mean)
```

```r
diff_begin_to_end <- graphs[[k_num_psets]] %m% graphs[[1]]
#the "core" network
core_network <- graphs[[1]] %m% (graphs[[1]] %m%  graphs[[k_num_psets]])


num_trials <- 1000
components <- numeric(num_trials)
num_loneley <- numeric(num_trials)
biggest_com <- numeric(num_trials)

for(trial in 1:num_trials){
  g_trial <- random_add_links(core_network, ecount(diff_begin_to_end))
  comp <- components(g_trial)
  components[trial] <- comp$no
  num_loneley[trial] <- sum(comp$csize==1)
  biggest_com[trial] <- max(comp$csize)
  if(trial %% 500 == 0){
    print(trial)
  }
}


hist(components,main = "Number of Components in Random Graphs",
     breaks =25, xlim = c(0,40),
     xlab = "Number of Components")
abline(v = 9)
#abline(v = assort_by_pset[1])



hist(num_loneley,main = "Number of Lonely People in Random Graphs",
     breaks =25, xlim = c(0,40),
     xlab = "Number of Components")
abline(v = 4)

hist(biggest_com,main = "Size of Biggest component in Random Graphs",
     breaks =25, xlim = c(0,110),
     xlab = "Number of Components")
abline(v = 94)
```

```r
out <- c()
ind <-c()
for(i in 1:k_num_psets){
  out <- c(out, igraph::degree(graphs[[i]],mode = "out"))
```

```r
  ind <- c(ind, igraph::degree(graphs[[i]],mode = "in"))
  hist(igraph::degree(graphs[[i]],mode = "out"), main = paste("out",i),breaks =20)
  hist(igraph::degree(graphs[[i]],mode = "in"), main = paste("in",i),breaks =20)
}
mean(out)
mean(ind)
range(out)
range(ind)

mean(ind<4)
mean(out<4)

data_pl <- displ$new(out+1)
est <- estimate_xmin(data_pl)
data_pl$xmin <- est$xmin
data_pl$pars <- est$pars
est
bs <- bootstrap_p(data_pl)
print(bs$p)
plot(bs)


data_pl <- displ$new(ind+1)
est <- estimate_xmin(data_pl)
data_pl$xmin <- est$xmin
data_pl$pars <- est$pars
est
bs <- bootstrap_p(data_pl)
print(bs$p)

#http://stats.stackexchange.com/questions/108843/how-to-test-whether-a-distribution-follows-a-power-law
for(i in 1:7){
data_pl <- displ$new(igraph::degree(graphs[[i]],mode = "out")+1)
est <- estimate_xmin(data_pl)
data_pl$xmin <- est$xmin
data_pl$pars <- est$pars
est
bs <- bootstrap_p(data_pl)
print(bs$p)
}
```

```r
women <- rep(0,32)
women[1:23] <- 1
men <- rep(0,100)
men[1:86] <- 1
queer <- rep(0,2)
queer[1] <- 1

t.test(c(women),men)

barplot(c(mean(women),mean(men),mean(queer)),beside = T)
#nope, not significant
```