Hierarchy in a TC, Facility 3 Women

Benjamin W. Campbell

Table of Contents

# Introduction

This notebook is for a project related to modeling the hierarchy within a TC clinical setting, looking specifically at corrections and whether many of our prior expectations about hierarchy within the TC environment hold. For example, does maximum position within the hierarchy correlate with outcomes, such as graduation?

# Create Network Objects

The first step is to load in the data. We want weighted, directed networks of corrections at the weekly level.

## set up working directory  
wd <- getwd()  
setwd(wd)  
  
## load corrections edgelist  
edgelist <- read.table(paste0(wd,"/data/F3F-ledge"), stringsAsFactors = FALSE)  
  
## process data  
library(tidyverse)

## Warning: package 'tibble' was built under R version 3.6.2

## Warning: package 'tidyr' was built under R version 3.6.2

## Warning: package 'dplyr' was built under R version 3.6.2

library(lubridate)  
  
edgelist\_cleaned <- edgelist %>%  
 rename(Date = V1, Sender = V2, Reciever = V3, Weight = V4) %>%  
 filter(Sender != 0) %>%  
 group\_by(Date, Sender, Reciever) %>%  
 summarize(Weight = sum(Weight))  
  
summary(edgelist\_cleaned$Weight)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 1.000 1.000 1.091 1.000 5.000

edgelist\_cleaned$Date <- mdy(edgelist\_cleaned$Date)  
  
## get t time stamp  
edgelist\_cleaned$t <- as.numeric(round(difftime(edgelist\_cleaned$Date, min(edgelist\_cleaned$Date), units = "weeks"))+1)  
# get in rank  
edgelist\_cleaned$t <- match(edgelist\_cleaned$t, sort(unique(edgelist\_cleaned$t)))  
  
## aggregate to week  
edgelist\_weekly <- edgelist\_cleaned %>%  
 group\_by(Sender, Reciever, t) %>%  
 summarize(Weight = sum(Weight)) %>%  
 arrange(t)  
  
head(edgelist\_weekly)

## # A tibble: 6 x 4  
## # Groups: Sender, Reciever [6]  
## Sender Reciever t Weight  
## <chr> <chr> <int> <int>  
## 1 07-51-016 07-51-110W 1 1  
## 2 07-51-016 07-51-127W 1 1  
## 3 07-51-016 07-59-117W 1 1  
## 4 07-21-085 07-59-117W 2 1  
## 5 07-46-039 07-59-117W 2 1  
## 6 07-59-117W 07-51-110W 2 1

summary(edgelist\_weekly$Weight)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 1.000 1.000 1.297 1.000 6.000

## get into network format  
library(igraph)  
t\_steps <- sort(unique(edgelist\_weekly$t))  
net\_list <- as.list(rep(NA, length(t\_steps)))  
index = 0  
  
## function to make network for time slice  
create\_network <- function(edgelist, t){  
 # reduce edgelist to time t  
 t\_slice <- edgelist[edgelist$t == t,]  
 t\_graph <- graph.data.frame(t\_slice, directed = TRUE)  
 # return network  
 return(t\_graph)  
}  
  
## populate list  
for(t in t\_steps){  
 # increment index  
 index = index+1  
 # create network  
 net <- create\_network(edgelist\_weekly, t)  
 # insert into index'ed element of list  
 net\_list[[index]] <- net  
}

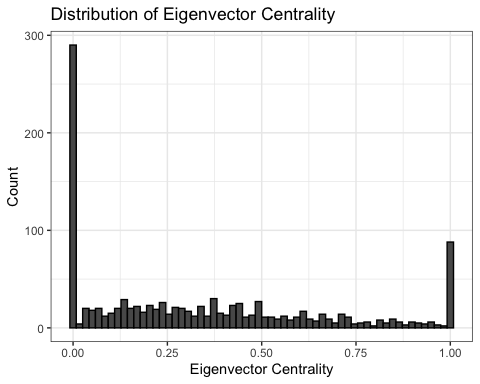
# Calculate Eigenvector Centrality

net\_list now contains a list of weighted and directed igraph objects. With this list, we can then go on to compute eigenvector centrality at the weekly level for every node.

# make function  
get\_eigen\_table <- function(graph, t){  
 # get weighted eigenvector centrality  
 scores <- eigen\_centrality(graph, weights = E(graph)$Weight, directed = TRUE)$vector  
 # put in table  
 t\_df <- tibble(  
 Id = as.character(names(scores)),  
 t = as.integer(t),  
 eigen\_cent = as.numeric(scores)  
 )  
 # return table  
 return(t\_df)  
}  
  
# initialize empty dataframe  
eigen\_df <- tibble()  
  
# loop through  
for(t in 1:length(net\_list)){  
 # get one network  
 net <- net\_list[[t]]  
 # get dataframe  
 t\_df <- get\_eigen\_table(net, t)  
 # bind to original dataframe  
 eigen\_df <- bind\_rows(eigen\_df, t\_df)  
}  
  
head(eigen\_df)

## # A tibble: 6 x 3  
## Id t eigen\_cent  
## <chr> <int> <dbl>  
## 1 07-51-016 1 0  
## 2 07-51-110W 1 0  
## 3 07-51-127W 1 0  
## 4 07-59-117W 1 0  
## 5 07-21-085 2 0  
## 6 07-46-039 2 0

ggplot(eigen\_df, aes(x = eigen\_cent)) +  
 geom\_histogram(colour="black", bins = 60) +  
 theme\_bw() +  
 ggtitle("Distribution of Eigenvector Centrality") +  
 xlab("Eigenvector Centrality") +  
 ylab("Count")

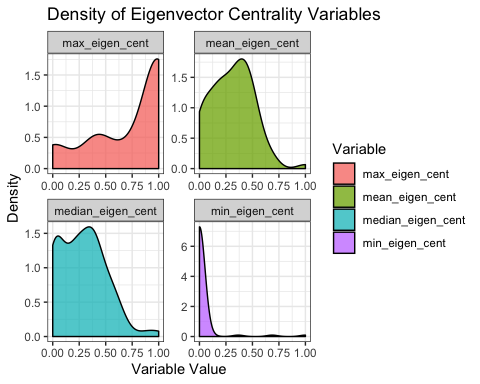


# Create Variables

So, given that we have an eigenvector centrality that is measured longitudinally, but only have a single observation of the outcome, how do we collapse this measure?

* We could look at minimum eigenvector centrality, which would tell us about the highest position in the hierarchy that anyone ever achieves.
* We could look at maximum eigenvector centrality, which would tell us about the lowest position in the hierarchy that anyone ever achieves.
* we could look at average or median eigenvector centrality, which would tell us something about the central tendency of someone in the networ with respect to where they are in the hierarchy.
* We could look at any of the prior measures over their last month or something there. This would tell us in general how well they do towards the end of their tenure.

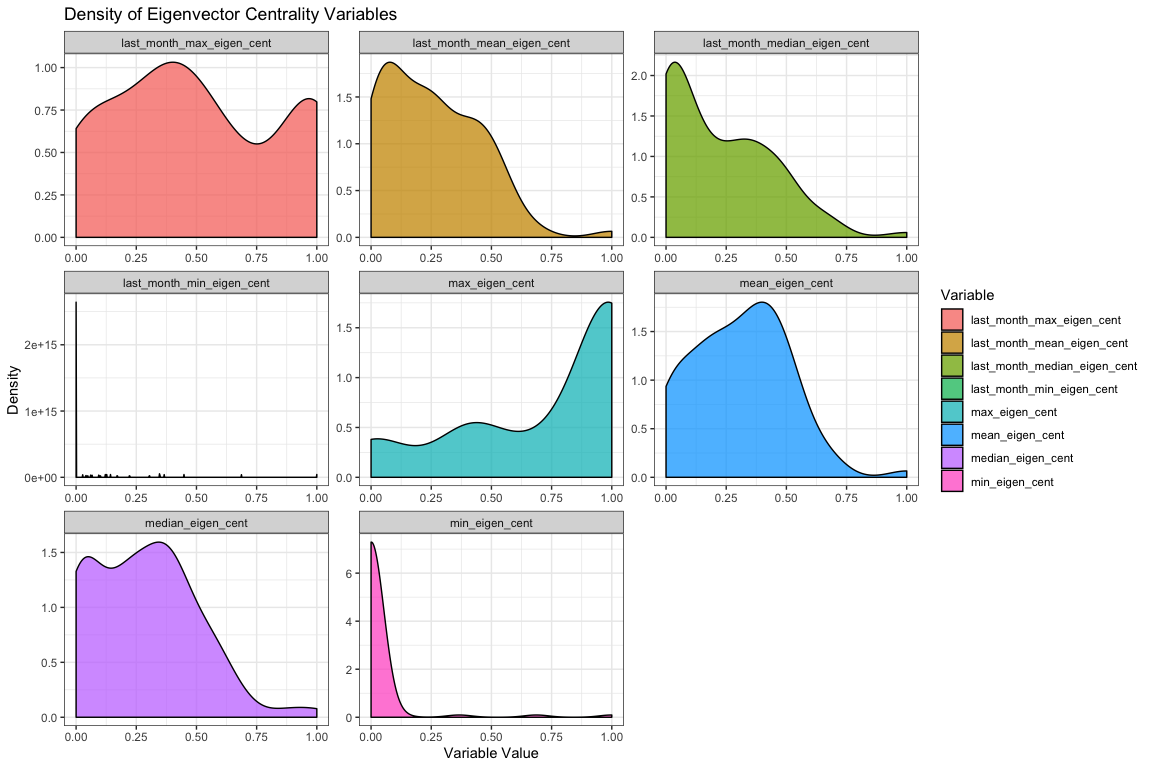
# min. eigen centrality -- highest position ever achieved in hierarchy  
# max. eigen centrality -- lowest position ever achieved in hierarchy  
# mean eigen centrality -- average position in the hierarchy  
# median eigen centrality -- another measure of central tendency  
nodal\_eigen <- eigen\_df %>%  
 group\_by(Id) %>%  
 summarize(min\_eigen\_cent = min(eigen\_cent),  
 max\_eigen\_cent = max(eigen\_cent),  
 mean\_eigen\_cent = mean(eigen\_cent),  
 median\_eigen\_cent = median(eigen\_cent))  
  
# plot df  
plot\_df <- nodal\_eigen %>%   
 gather("Variable", "Value",-Id)  
   
  
ggplot(plot\_df, aes(x = Value, fill = Variable)) +  
 geom\_density(colour="black", alpha = 0.75) +  
 theme\_bw() +  
 ggtitle("Density of Eigenvector Centrality Variables") +  
 xlab("Variable Value") +  
 ylab("Density") +   
 facet\_wrap(vars(Variable), scales = 'free')



# Summaries  
summary(nodal\_eigen)

## Id min\_eigen\_cent max\_eigen\_cent mean\_eigen\_cent   
## Length:82 Min. :0.00000 Min. :0.0000 Min. :0.0000   
## Class :character 1st Qu.:0.00000 1st Qu.:0.4448 1st Qu.:0.1658   
## Mode :character Median :0.00000 Median :1.0000 Median :0.3231   
## Mean :0.02735 Mean :0.7305 Mean :0.3091   
## 3rd Qu.:0.00000 3rd Qu.:1.0000 3rd Qu.:0.4449   
## Max. :0.99999 Max. :1.0000 Max. :1.0000   
## median\_eigen\_cent  
## Min. :0.00000   
## 1st Qu.:0.06692   
## Median :0.26672   
## Mean :0.28108   
## 3rd Qu.:0.41013   
## Max. :0.99999

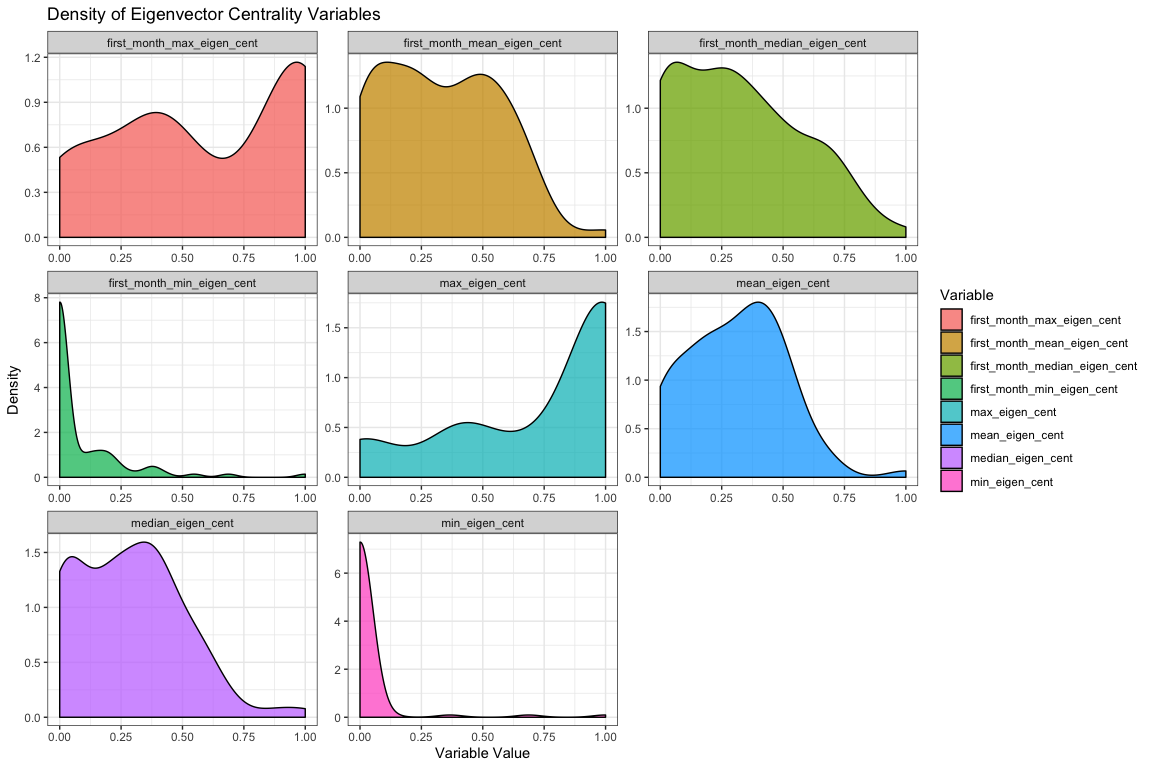
get\_last\_month\_variables <- function(id){  
 id\_df <- eigen\_df %>%   
 filter(Id == id) %>%   
 arrange(t) %>%   
 tail(4) %>%  
 summarize(Id = unique(Id),  
 last\_month\_min\_eigen\_cent = min(eigen\_cent),  
 last\_month\_max\_eigen\_cent = max(eigen\_cent),  
 last\_month\_mean\_eigen\_cent = mean(eigen\_cent),  
 last\_month\_median\_eigen\_cent = median(eigen\_cent))  
 return(id\_df)  
}  
  
ids <- unique(eigen\_df$Id)  
  
last\_month\_df <- tibble()  
  
for(i in ids){  
 id\_df <- get\_last\_month\_variables(i)  
 last\_month\_df <- bind\_rows(last\_month\_df, id\_df)  
}  
  
# plot df  
plot\_df\_full <- last\_month\_df %>%   
 gather("Variable", "Value",-Id) %>%  
 bind\_rows(plot\_df)  
   
  
ggplot(plot\_df\_full, aes(x = Value, fill = Variable)) +  
 geom\_density(colour="black", alpha = 0.75) +  
 theme\_bw() +  
 ggtitle("Density of Eigenvector Centrality Variables") +  
 xlab("Variable Value") +  
 ylab("Density") +   
 facet\_wrap(vars(Variable), scales = 'free', nrow = 3)



# Summaries  
summary(last\_month\_df)

## Id last\_month\_min\_eigen\_cent last\_month\_max\_eigen\_cent  
## Length:82 Min. :0.0000 Min. :0.0000   
## Class :character 1st Qu.:0.0000 1st Qu.:0.2096   
## Mode :character Median :0.0000 Median :0.4630   
## Mean :0.0591 Mean :0.5032   
## 3rd Qu.:0.0000 3rd Qu.:0.8853   
## Max. :1.0000 Max. :1.0000   
## last\_month\_mean\_eigen\_cent last\_month\_median\_eigen\_cent  
## Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.0797 1st Qu.:0.02171   
## Median :0.2248 Median :0.18160   
## Mean :0.2507 Mean :0.22077   
## 3rd Qu.:0.4126 3rd Qu.:0.35289   
## Max. :1.0000 Max. :0.99999

get\_first\_month\_variables <- function(id){  
 id\_df <- eigen\_df %>%   
 filter(Id == id) %>%   
 arrange(t) %>%   
 head(4) %>%  
 summarize(Id = unique(Id),  
 first\_month\_min\_eigen\_cent = min(eigen\_cent),  
 first\_month\_max\_eigen\_cent = max(eigen\_cent),  
 first\_month\_mean\_eigen\_cent = mean(eigen\_cent),  
 first\_month\_median\_eigen\_cent = median(eigen\_cent))  
 return(id\_df)  
}  
  
ids <- unique(eigen\_df$Id)  
  
first\_month\_df <- tibble()  
  
for(i in ids){  
 id\_df <- get\_first\_month\_variables(i)  
 first\_month\_df <- bind\_rows(first\_month\_df, id\_df)  
}  
  
# plot df  
plot\_df\_full <- first\_month\_df %>%   
 gather("Variable", "Value",-Id) %>%  
 bind\_rows(plot\_df)  
   
  
ggplot(plot\_df\_full, aes(x = Value, fill = Variable)) +  
 geom\_density(colour="black", alpha = 0.75) +  
 theme\_bw() +  
 ggtitle("Density of Eigenvector Centrality Variables") +  
 xlab("Variable Value") +  
 ylab("Density") +   
 facet\_wrap(vars(Variable), scales = 'free', nrow = 3)



# Summaries  
summary(first\_month\_df)

## Id first\_month\_min\_eigen\_cent first\_month\_max\_eigen\_cent  
## Length:82 Min. :0.00000 Min. :0.0000   
## Class :character 1st Qu.:0.00000 1st Qu.:0.2800   
## Mode :character Median :0.00000 Median :0.5142   
## Mean :0.08494 Mean :0.5891   
## 3rd Qu.:0.12508 3rd Qu.:1.0000   
## Max. :0.99999 Max. :1.0000   
## first\_month\_mean\_eigen\_cent first\_month\_median\_eigen\_cent  
## Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.1143 1st Qu.:0.07655   
## Median :0.3101 Median :0.27974   
## Mean :0.3269 Mean :0.31711   
## 3rd Qu.:0.5056 3rd Qu.:0.49919   
## Max. :1.0000 Max. :0.99999

# Join to Node Data

With the measures of hierarchy created, we can now process the node data and join these variables to it. Once all of that is taken care of we can move on to analysis!

# read node data  
nodes <- read.table(paste0(wd,"/data/F3F"), stringsAsFactors = FALSE, header = TRUE)  
  
sum(!is.na(nodes$wcid))

## [1] 76

# get total unique nodes  
length(unique(nodes$wcid))

## [1] 76

# get days in program  
nodes$days\_in\_program <- as.Date(as.character(nodes$exit), format="%m/%d/%Y")-as.Date(as.character(nodes$enter), format="%m/%d/%Y")  
  
# process recidivism  
nodes$recidFlag <- rep(0, times = nrow(nodes))  
nodes$recidFlag[!(is.na(nodes$recidate1))] <- 1  
   
nodes$recidDate <- as.character(nodes$recidate1)  
nodes[is.na(nodes$recidDate),]$recidDate <- "09/04/2009"  
  
nodes$gap <- as.Date(as.character(nodes$recidDate), format="%m/%d/%Y")-as.Date(as.character(nodes$exit), format="%m/%d/%Y")  
  
# remove folks who visit multiple times  
repeat\_visitors <- names(which(table(nodes$wcid) > 1))  
  
nodes <- nodes[!(nodes$wcid %in% repeat\_visitors),]  
  
# join network variables  
# first rename Id to wcid  
dat <- nodes %>%  
 rename(Id = wcid) %>%  
 select(Id, age, lsi, lsiExit, black, success, recidFlag, recidDate, gap, days\_in\_program) %>%  
 inner\_join(last\_month\_df, by = "Id") %>%  
 inner\_join(first\_month\_df, by = "Id") %>%  
 inner\_join(nodal\_eigen, by = "Id")  
  
dat$days\_in\_program <- as.numeric(dat$days\_in\_program)  
  
head(dat)

## Id age lsi lsiExit black success recidFlag recidDate gap  
## 1 08-51-170W 25 30 27 0 1 0 09/04/2009 221 days  
## 2 07-46-236W 21 27 17 1 1 0 09/04/2009 557 days  
## 3 07-51-309W 26 37 34 1 1 0 09/04/2009 473 days  
## 4 07-80-293W 31 23 19 1 1 0 09/04/2009 501 days  
## 5 08-11-054W 36 17 15 1 1 0 09/04/2009 361 days  
## 6 08-12-254W 23 25 27 1 0 1 2/5/2009 37 days  
## days\_in\_program last\_month\_min\_eigen\_cent last\_month\_max\_eigen\_cent  
## 1 165 0.000000e+00 1.000000e+00  
## 2 151 2.418680e-16 2.781722e-01  
## 3 165 2.431384e-17 1.545540e-01  
## 4 153 0.000000e+00 3.304702e-17  
## 5 137 0.000000e+00 3.594029e-01  
## 6 82 0.000000e+00 1.000000e+00  
## last\_month\_mean\_eigen\_cent last\_month\_median\_eigen\_cent  
## 1 5.006952e-01 5.013904e-01  
## 2 1.222207e-01 1.053553e-01  
## 3 1.109482e-01 1.446195e-01  
## 4 1.548209e-17 1.444067e-17  
## 5 1.481569e-01 1.166124e-01  
## 6 5.829337e-01 6.658674e-01  
## first\_month\_min\_eigen\_cent first\_month\_max\_eigen\_cent  
## 1 1.728230e-01 0.9412331  
## 2 0.000000e+00 0.4406197  
## 3 1.962519e-16 1.0000000  
## 4 7.040190e-17 0.3399692  
## 5 1.938084e-17 0.2365433  
## 6 2.124502e-01 1.0000000  
## first\_month\_mean\_eigen\_cent first\_month\_median\_eigen\_cent min\_eigen\_cent  
## 1 0.5018954 0.4467627 0  
## 2 0.1774241 0.1345383 0  
## 3 0.5918387 0.6836774 0  
## 4 0.2084837 0.2469828 0  
## 5 0.1107186 0.1031656 0  
## 6 0.6338963 0.6615674 0  
## max\_eigen\_cent mean\_eigen\_cent median\_eigen\_cent  
## 1 1.0000000 0.54262607 5.629731e-01  
## 2 0.8571429 0.16483883 1.053553e-01  
## 3 1.0000000 0.33525140 2.700985e-01  
## 4 0.4325248 0.11404619 5.172446e-17  
## 5 0.3594029 0.09272324 8.092874e-02  
## 6 1.0000000 0.69083615 8.750007e-01

# Exploratory Data Analysis

We’ve got the data put together, now is time to think about the relationship between these key variables and TC outcomes like graduation. The following network variables we think might matter most based upon their distributions:

* last\_month\_max\_eigen\_cent
* last\_month\_mean\_eigen\_cent
* max\_eigen\_cent
* mean\_eigen\_cent

# Neg, sig - as lower in hierarchy towards the end, less likely to be successful  
cor.test(dat$last\_month\_max\_eigen\_cent, dat$success)

##   
## Pearson's product-moment correlation  
##   
## data: dat$last\_month\_max\_eigen\_cent and dat$success  
## t = -1.2028, df = 71, p-value = 0.2331  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.35968208 0.09174163  
## sample estimates:  
## cor   
## -0.1413082

# Neg, sig - as lower in hierarchy towards the end, less likely to be successful  
cor.test(dat$last\_month\_mean\_eigen\_cent, dat$success)

##   
## Pearson's product-moment correlation  
##   
## data: dat$last\_month\_mean\_eigen\_cent and dat$success  
## t = -2.5176, df = 71, p-value = 0.01407  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.48444193 -0.06017831  
## sample estimates:  
## cor   
## -0.286282

# Neg, sig - as lower in hierarchy, less likely to be successful  
cor.test(dat$max\_eigen\_cent, dat$success)

##   
## Pearson's product-moment correlation  
##   
## data: dat$max\_eigen\_cent and dat$success  
## t = 1.4288, df = 71, p-value = 0.1575  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.06540361 0.38253335  
## sample estimates:  
## cor   
## 0.1671793

# Neg, sig - as lower in hierarchy, less likely to be successful  
cor.test(dat$mean\_eigen\_cent, dat$success)

##   
## Pearson's product-moment correlation  
##   
## data: dat$mean\_eigen\_cent and dat$success  
## t = -1.9968, df = 71, p-value = 0.04968  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.4374521182 -0.0005545991  
## sample estimates:  
## cor   
## -0.2305925

# Modeling Graduation

The first and most easy thing we could do is use simple linear modeling to examine the effect of some of these covariates on graduation, while controlling for the confounding effects of other variables. Here we will fit those models and show results:

last\_month\_max\_model <- glm(success ~  
 age +  
 lsiExit +  
 black +  
 days\_in\_program +  
 last\_month\_max\_eigen\_cent,  
 data = dat,  
 family = binomial(link = 'logit'))  
  
last\_month\_mean\_model <- glm(success ~  
 age +  
 lsiExit +  
 black +  
 days\_in\_program +  
 last\_month\_mean\_eigen\_cent,  
 data = dat,  
 family = binomial(link = 'logit'))  
  
max\_model <- glm(success ~  
 age +  
 lsiExit +  
 black +  
 days\_in\_program +  
 max\_eigen\_cent,  
 data = dat,  
 family = binomial(link = 'logit'))  
  
mean\_model <- glm(success ~  
 age +  
 lsiExit +  
 black +  
 days\_in\_program +  
 mean\_eigen\_cent,  
 data = dat,  
 family = binomial(link = 'logit'))  
  
library(texreg)

## Warning: package 'texreg' was built under R version 3.6.2

screenreg(l = list(last\_month\_max\_model, last\_month\_mean\_model, max\_model, mean\_model))

##   
## ==================================================================  
## Model 1 Model 2 Model 3 Model 4   
## ------------------------------------------------------------------  
## (Intercept) 0.18 -0.44 2.48 6.41   
## (5.67) (6.26) (5.98) (7.41)   
## age 0.01 0.01 0.01 -0.05   
## (0.11) (0.11) (0.10) (0.11)   
## lsiExit -0.36 \* -0.36 \* -0.45 \* -0.49 \*   
## (0.15) (0.16) (0.20) (0.22)   
## black 1.45 1.48 3.28 2.74   
## (2.61) (2.41) (3.56) (3.73)   
## days\_in\_program 0.08 \*\* 0.08 \*\* 0.10 \*\* 0.09 \*\*  
## (0.02) (0.03) (0.04) (0.03)   
## last\_month\_max\_eigen\_cent -0.66   
## (2.37)   
## last\_month\_mean\_eigen\_cent 0.27   
## (4.86)   
## max\_eigen\_cent -5.06   
## (3.97)   
## mean\_eigen\_cent -6.94   
## (4.75)   
## ------------------------------------------------------------------  
## AIC 29.52 29.59 27.45 26.79   
## BIC 42.47 42.55 40.41 39.75   
## Log Likelihood -8.76 -8.80 -7.73 -7.40   
## Deviance 17.52 17.59 15.45 14.79   
## Num. obs. 64 64 64 64   
## ==================================================================  
## \*\*\* p < 0.001; \*\* p < 0.01; \* p < 0.05

get\_pred\_prob\_plot <- function(model, xvar, xlab){  
 std <- qnorm(0.95 / 2 + 0.5)  
   
 #last\_month\_max\_eigen\_cent\_plot  
 data = model$data  
 new\_data <- data.frame(  
 age = rep(mean(data$age, na.rm = TRUE), nrow(data)),  
 lsiExit = rep(mean(data$lsiExit, na.rm = TRUE), nrow(data)),  
 black = rep(0, nrow(data)),  
 days\_in\_program = rep(mean(data$days\_in\_program, na.rm = TRUE), nrow(data)),  
 stupid\_placeholder = data[,xvar]  
 )  
   
 colnames(new\_data)[5] <- xvar  
   
 predicted\_data <- as.data.frame(predict(model, newdata = new\_data,  
 type="link", se=TRUE))  
   
 new\_data <- cbind(new\_data, predicted\_data)  
 new\_data$ymin <- model$family$linkinv(new\_data$fit - std \* new\_data$se)  
 new\_data$ymax <- model$family$linkinv(new\_data$fit + std \* new\_data$se)  
 new\_data$fit <- model$family$linkinv(new\_data$fit)  
   
 library(ggplot2)  
 p <- ggplot(new\_data, aes(x=new\_data[,xvar])) +  
 geom\_ribbon(data = new\_data, aes(y=fit, ymin=ymin, ymax=ymax), alpha = 0.5) +  
 geom\_line(data = new\_data, aes(x = new\_data[,xvar], y=fit), size = 1.5, colour = "firebrick4") +  
 scale\_y\_continuous(limits=c(0,1)) +  
 theme\_bw() +   
 theme(legend.position = c(0.2, 0.8),  
 axis.text=element\_text(size=12),  
 axis.title=element\_text(size=14,face="bold"))+  
 labs(x=xlab, y="Probability of Graduation")   
   
 return(p)  
}  
  
last\_month\_max\_pred\_prob <- get\_pred\_prob\_plot(last\_month\_max\_model,   
 "last\_month\_max\_eigen\_cent",  
 "Last Month's Highest Eigenvector Centrality")

## Warning: Ignoring unknown aesthetics: y

last\_month\_mean\_pred\_prob <- get\_pred\_prob\_plot(last\_month\_mean\_model,   
 "last\_month\_mean\_eigen\_cent",  
 "Last Month's Average Eigenvector Centrality")

## Warning: Ignoring unknown aesthetics: y

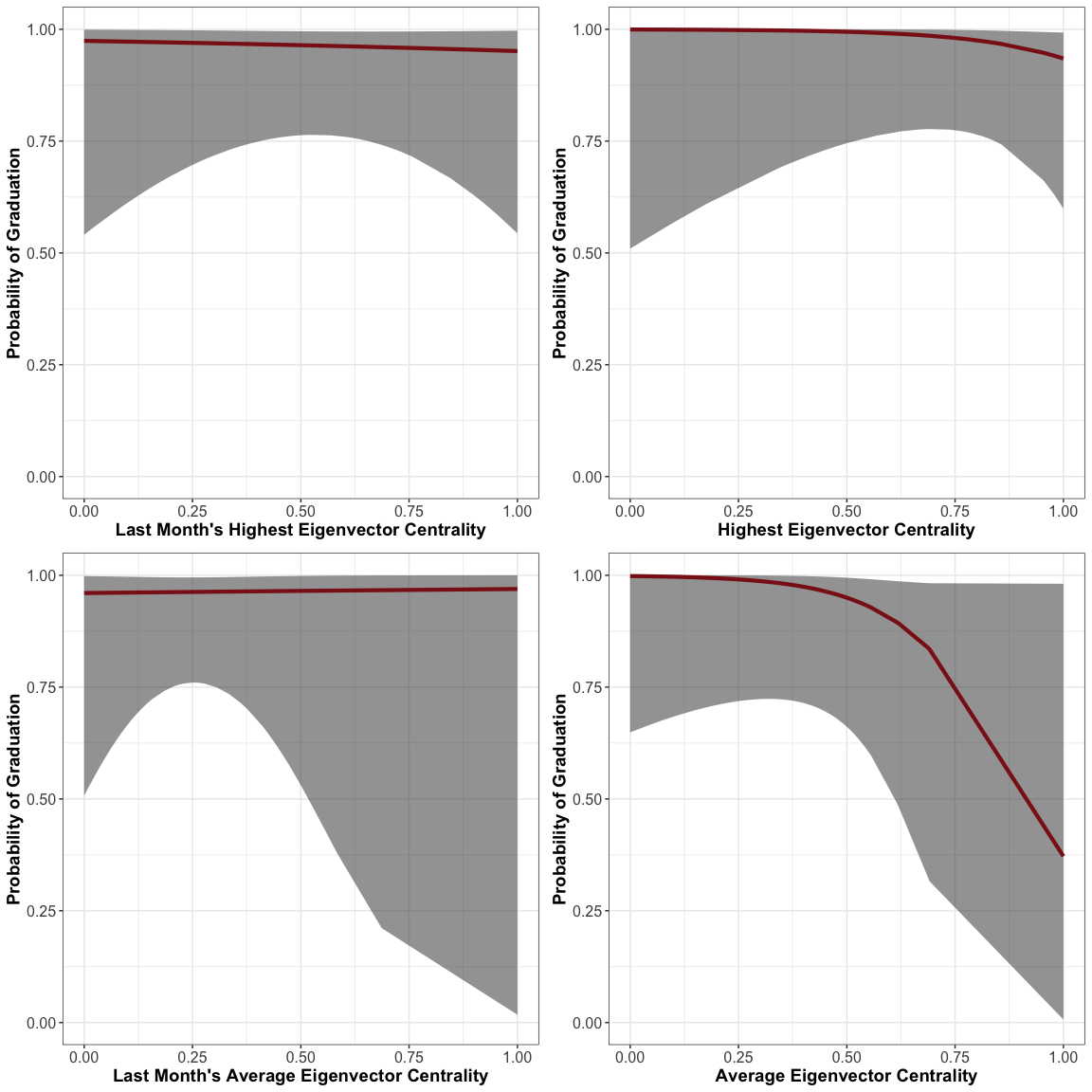
max\_eigen\_cent\_pred\_prob <- get\_pred\_prob\_plot(max\_model,   
 "max\_eigen\_cent",  
 "Highest Eigenvector Centrality")

## Warning: Ignoring unknown aesthetics: y

mean\_eigen\_cent\_pred\_prob <- get\_pred\_prob\_plot(mean\_model,   
 "mean\_eigen\_cent",  
 "Average Eigenvector Centrality")

## Warning: Ignoring unknown aesthetics: y

library(Rmisc)  
multiplot(plotlist = list(last\_month\_max\_pred\_prob, last\_month\_mean\_pred\_prob, max\_eigen\_cent\_pred\_prob, mean\_eigen\_cent\_pred\_prob),  
 cols = 2)



get\_pred\_prob\_dist <- function(model, xvar){  
 data = model$data  
   
 mean <- mean(data[,xvar], na.rm = TRUE)  
 std <- sd(data[,xvar], na.rm = TRUE)  
 input\_vec <- c(mean-2\*std, mean-1\*std, mean, mean+1\*std, mean+2\*std)  
   
 new\_data <- data.frame(  
 age = rep(mean(data$age, na.rm = TRUE), length(input\_vec)),  
 lsiExit = rep(mean(data$lsiExit, na.rm = TRUE), length(input\_vec)),  
 black = rep(0, length(input\_vec)),  
 days\_in\_program = rep(mean(data$days\_in\_program, na.rm = TRUE), length(input\_vec)),  
 stupid\_placeholder = input\_vec  
 )  
   
 colnames(new\_data)[5] <- xvar  
   
 predicted\_data <- as.data.frame(predict(model, newdata = new\_data,  
 type="link", se=TRUE))  
   
 probs <- model$family$linkinv(predicted\_data$fit)   
   
 return(probs)  
}  
  
# 2 std below mean, 1 std below mean, mean, 1 std above, 2 std above  
get\_pred\_prob\_dist(last\_month\_max\_model,   
 "last\_month\_max\_eigen\_cent")

## [1] 0.9755130 0.9699830 0.9632512 0.9550796 0.9451943

# 2 std below mean, 1 std below mean, mean, 1 std above, 2 std above  
get\_pred\_prob\_dist(last\_month\_mean\_model,   
 "last\_month\_mean\_eigen\_cent")

## [1] 0.9591186 0.9611052 0.9629990 0.9648039 0.9665238

# 2 std below mean, 1 std below mean, mean, 1 std above, 2 std above  
get\_pred\_prob\_dist(max\_model,   
 "max\_eigen\_cent")

## [1] 0.9983882 0.9935226 0.9743475 0.9038986 0.6996219

# 2 std below mean, 1 std below mean, mean, 1 std above, 2 std above  
get\_pred\_prob\_dist(mean\_model,   
 "mean\_eigen\_cent")

## [1] 0.9986011 0.9950710 0.9827864 0.9416793 0.8203508

# Descriptive Stats

max(edgelist\_cleaned$Date)-min(edgelist\_cleaned$Date)

## Time difference of 604 days

summary(dat)

## Id age lsi lsiExit   
## Length:73 Min. :19.0 Min. : 9.00 Min. : 8.00   
## Class :character 1st Qu.:26.0 1st Qu.:21.00 1st Qu.:17.00   
## Mode :character Median :32.0 Median :25.00 Median :20.00   
## Mean :32.1 Mean :25.05 Mean :21.14   
## 3rd Qu.:36.0 3rd Qu.:29.00 3rd Qu.:26.00   
## Max. :51.0 Max. :40.00 Max. :34.00   
## NA's :9   
## black success recidFlag recidDate   
## Min. :0.00000 Min. :0.0000 Min. :0.0000 Length:73   
## 1st Qu.:0.00000 1st Qu.:1.0000 1st Qu.:0.0000 Class :character   
## Median :0.00000 Median :1.0000 Median :0.0000 Mode :character   
## Mean :0.06849 Mean :0.7808 Mean :0.2329   
## 3rd Qu.:0.00000 3rd Qu.:1.0000 3rd Qu.:0.0000   
## Max. :1.00000 Max. :1.0000 Max. :1.0000   
##   
## gap days\_in\_program last\_month\_min\_eigen\_cent  
## Length:73 Min. : 10.0 Min. :0.00000   
## Class :difftime 1st Qu.:125.0 1st Qu.:0.00000   
## Mode :numeric Median :151.0 Median :0.00000   
## Mean :140.1 Mean :0.06467   
## 3rd Qu.:165.0 3rd Qu.:0.02720   
## Max. :368.0 Max. :0.99999   
##   
## last\_month\_max\_eigen\_cent last\_month\_mean\_eigen\_cent  
## Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.3333 1st Qu.:0.1127   
## Median :0.4948 Median :0.2577   
## Mean :0.5515 Mean :0.2748   
## 3rd Qu.:0.9161 3rd Qu.:0.4145   
## Max. :1.0000 Max. :1.0000   
##   
## last\_month\_median\_eigen\_cent first\_month\_min\_eigen\_cent  
## Min. :0.00000 Min. :0.00000   
## 1st Qu.:0.03472 1st Qu.:0.00000   
## Median :0.19464 Median :0.00000   
## Mean :0.24192 Mean :0.09542   
## 3rd Qu.:0.37443 3rd Qu.:0.13864   
## Max. :0.99999 Max. :0.99999   
##   
## first\_month\_max\_eigen\_cent first\_month\_mean\_eigen\_cent  
## Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.3958 1st Qu.:0.2000   
## Median :0.6868 Median :0.3570   
## Mean :0.6480 Mean :0.3603   
## 3rd Qu.:1.0000 3rd Qu.:0.5176   
## Max. :1.0000 Max. :1.0000   
##   
## first\_month\_median\_eigen\_cent min\_eigen\_cent max\_eigen\_cent   
## Min. :0.0000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.1596 1st Qu.:0.00000 1st Qu.:0.6253   
## Median :0.3118 Median :0.00000 Median :1.0000   
## Mean :0.3493 Mean :0.03072 Mean :0.8068   
## 3rd Qu.:0.5136 3rd Qu.:0.00000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :0.99999 Max. :1.0000   
##   
## mean\_eigen\_cent median\_eigen\_cent  
## Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.1953 1st Qu.:0.1581   
## Median :0.3353 Median :0.2982   
## Mean :0.3413 Mean :0.3103   
## 3rd Qu.:0.4658 3rd Qu.:0.4201   
## Max. :1.0000 Max. :1.0000   
##

apply(dat, 2, function(x) sd(x, na.rm = TRUE))

## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm  
## = na.rm): NAs introduced by coercion  
  
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm  
## = na.rm): NAs introduced by coercion  
  
## Warning in var(if (is.vector(x) || is.factor(x)) x else as.double(x), na.rm  
## = na.rm): NAs introduced by coercion

## Id age   
## NA 7.3298560   
## lsi lsiExit   
## 5.9788758 6.0680459   
## black success   
## 0.2543383 0.4165525   
## recidFlag recidDate   
## 0.4255894 NA   
## gap days\_in\_program   
## NA 45.0193642   
## last\_month\_min\_eigen\_cent last\_month\_max\_eigen\_cent   
## 0.1665136 0.3192832   
## last\_month\_mean\_eigen\_cent last\_month\_median\_eigen\_cent   
## 0.1938954 0.2174029   
## first\_month\_min\_eigen\_cent first\_month\_max\_eigen\_cent   
## 0.1789253 0.3295280   
## first\_month\_mean\_eigen\_cent first\_month\_median\_eigen\_cent   
## 0.2231894 0.2492205   
## min\_eigen\_cent max\_eigen\_cent   
## 0.1467620 0.2760325   
## mean\_eigen\_cent median\_eigen\_cent   
## 0.1820633 0.2115426

# Session Info

sessionInfo()

## R version 3.6.0 (2019-04-26)  
## Platform: x86\_64-apple-darwin15.6.0 (64-bit)  
## Running under: macOS Mojave 10.14.6  
##   
## Matrix products: default  
## BLAS: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRblas.0.dylib  
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib  
##   
## locale:  
## [1] en\_US.UTF-8/en\_US.UTF-8/en\_US.UTF-8/C/en\_US.UTF-8/en\_US.UTF-8  
##   
## attached base packages:  
## [1] stats graphics grDevices utils datasets methods base   
##   
## other attached packages:  
## [1] Rmisc\_1.5 plyr\_1.8.4 lattice\_0.20-38 texreg\_1.37.5   
## [5] igraph\_1.2.4.1 lubridate\_1.7.4 forcats\_0.4.0 stringr\_1.4.0   
## [9] dplyr\_1.0.2 purrr\_0.3.2 readr\_1.3.1 tidyr\_1.1.2   
## [13] tibble\_3.0.4 ggplot2\_3.2.1 tidyverse\_1.2.1  
##   
## loaded via a namespace (and not attached):  
## [1] tidyselect\_1.1.0 xfun\_0.9 haven\_2.1.0 colorspace\_1.4-1  
## [5] vctrs\_0.3.4 generics\_0.0.2 htmltools\_0.3.6 yaml\_2.2.0   
## [9] utf8\_1.1.4 rlang\_0.4.8 pillar\_1.4.6 glue\_1.4.2   
## [13] withr\_2.1.2 modelr\_0.1.4 readxl\_1.3.1 lifecycle\_0.2.0   
## [17] munsell\_0.5.0 gtable\_0.3.0 cellranger\_1.1.0 rvest\_0.3.4   
## [21] evaluate\_0.14 labeling\_0.3 knitr\_1.24 fansi\_0.4.0   
## [25] broom\_0.7.2 Rcpp\_1.0.5 scales\_1.0.0 backports\_1.1.4   
## [29] jsonlite\_1.6 hms\_0.4.2 digest\_0.6.20 stringi\_1.4.3   
## [33] grid\_3.6.0 cli\_1.1.0 tools\_3.6.0 magrittr\_1.5   
## [37] lazyeval\_0.2.2 crayon\_1.3.4 pkgconfig\_2.0.2 ellipsis\_0.3.1   
## [41] xml2\_1.2.0 assertthat\_0.2.1 rmarkdown\_1.12 httr\_1.4.0   
## [45] rstudioapi\_0.10 R6\_2.4.0 compiler\_3.6.0