Hierarchy in a TC, Facility 1 Women’s Unit

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/F1-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 1.092 1.000 7.000

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 "" "" 1 1  
## 2 "" "6780" 1 1  
## 3 "" "6781" 1 2  
## 4 "" "6970" 1 1  
## 5 "" "7040" 1 1  
## 6 "" "7045" 1 2

summary(edgelist\_weekly$Weight)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.000 1.000 1.000 1.201 1.000 15.000

## 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  
}

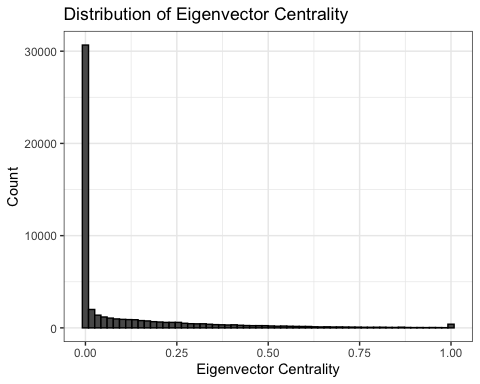
# Calculate Eigenvector Centrality

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 "" 1 3.07e-19  
## 2 "6780" 1 0.   
## 3 "6781" 1 6.03e- 1  
## 4 "6829" 1 8.41e-18  
## 5 "6832" 1 8.41e-18  
## 6 "6833" 1 8.41e-18

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")

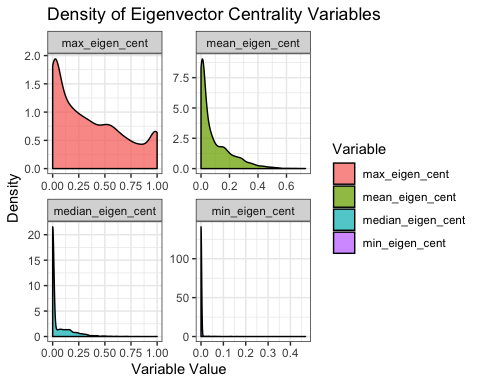


# Create Variables

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.

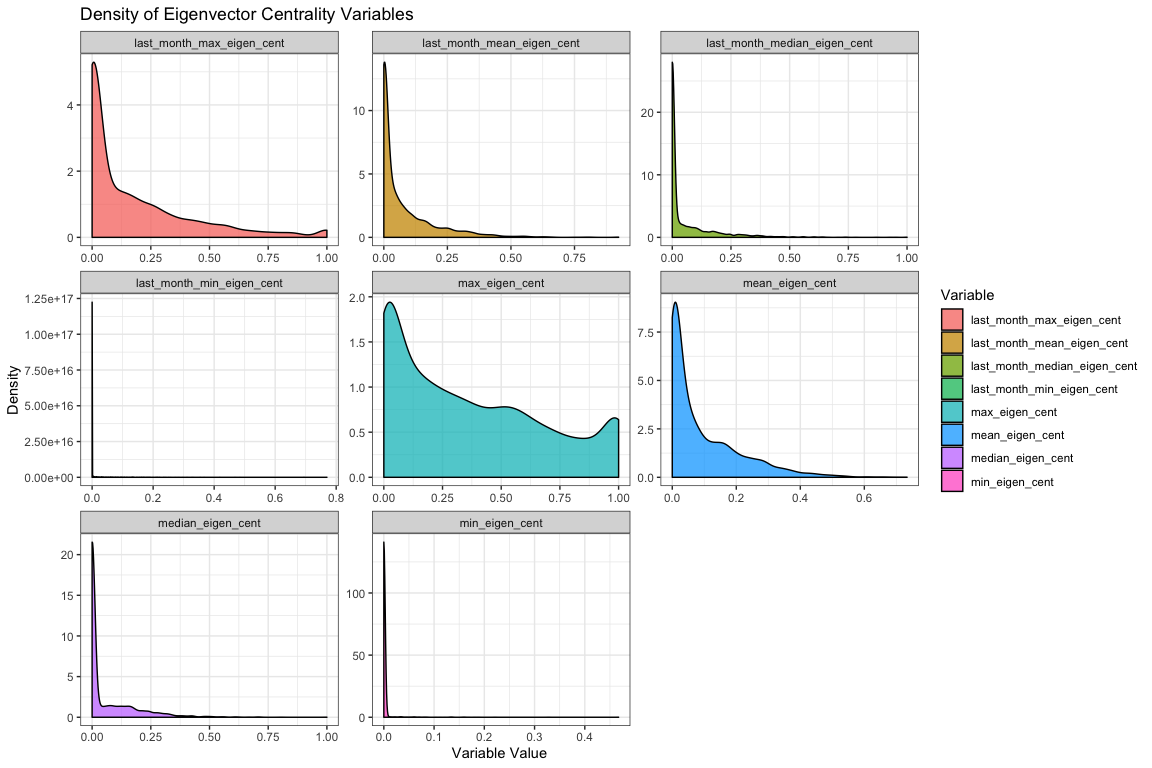
# min. eigen centrality -- highest position ever achieved in hierarchy  
# max. eigen centrality -- lowest position ever achieved in hierarchy  
# mean eigen centrality -- average position in the hierarchy  
# median eigen centrality -- another measure of central tendency  
nodal\_eigen <- eigen\_df %>%  
 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   
## Length:3579 Min. :0.000000 Min. :0.00000   
## Class :character 1st Qu.:0.000000 1st Qu.:0.06059   
## Mode :character Median :0.000000 Median :0.29253   
## Mean :0.001665 Mean :0.36581   
## 3rd Qu.:0.000000 3rd Qu.:0.60171   
## Max. :0.467261 Max. :1.00000   
## mean\_eigen\_cent median\_eigen\_cent  
## Min. :0.000000 Min. :0.00000   
## 1st Qu.:0.006639 1st Qu.:0.00000   
## Median :0.042096 Median :0.00000   
## Mean :0.092850 Mean :0.06067   
## 3rd Qu.:0.149015 3rd Qu.:0.09064   
## Max. :0.733465 Max. :1.00000

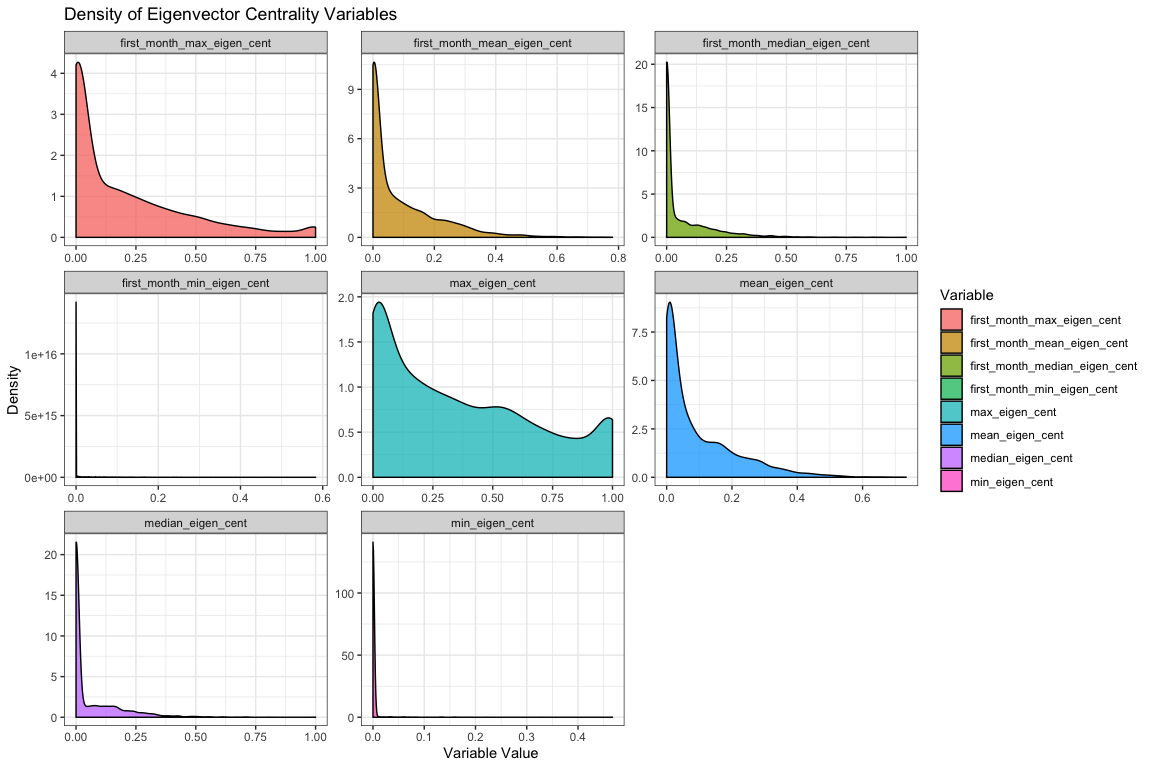
get\_last\_month\_variables <- function(id){  
 id\_df <- eigen\_df %>%   
 filter(Id == id) %>%   
 arrange(t) %>%   
 tail(4) %>%  
 summarize(Id = unique(Id),  
 last\_month\_min\_eigen\_cent = min(eigen\_cent),  
 last\_month\_max\_eigen\_cent = max(eigen\_cent),  
 last\_month\_mean\_eigen\_cent = 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:3579 Min. :0.00000 Min. :0.00000   
## Class :character 1st Qu.:0.00000 1st Qu.:0.00000   
## Mode :character Median :0.00000 Median :0.06641   
## Mean :0.01227 Mean :0.17552   
## 3rd Qu.:0.00000 3rd Qu.:0.26805   
## Max. :0.76955 Max. :1.00000   
## last\_month\_mean\_eigen\_cent last\_month\_median\_eigen\_cent  
## Min. :0.00000 Min. :0.0000000   
## 1st Qu.:0.00000 1st Qu.:0.0000000   
## Median :0.02107 Median :0.0000022   
## Mean :0.07546 Mean :0.0569599   
## 3rd Qu.:0.10695 3rd Qu.:0.0672389   
## Max. :0.92346 Max. :1.0000000

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:3579 Min. :0.00000 Min. :0.0000000   
## Class :character 1st Qu.:0.00000 1st Qu.:0.0000114   
## Mode :character Median :0.00000 Median :0.0910839   
## Mean :0.01316 Mean :0.2025326   
## 3rd Qu.:0.00000 3rd Qu.:0.3242833   
## Max. :0.58264 Max. :1.0000000   
## first\_month\_mean\_eigen\_cent first\_month\_median\_eigen\_cent  
## Min. :0.0000000 Min. :0.000000   
## 1st Qu.:0.0000029 1st Qu.:0.000000   
## Median :0.0299575 Median :0.000446   
## Mean :0.0860741 Mean :0.064237   
## 3rd Qu.:0.1367799 3rd Qu.:0.088615   
## Max. :0.7803747 Max. :1.000000

# 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/F1F"), stringsAsFactors = FALSE, header = TRUE)  
  
sum(!is.na(nodes$mdid))

## [1] 1268

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

## [1] 1218

# 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$mdid) > 1))  
  
nodes <- nodes[!(nodes$mdid %in% repeat\_visitors),]  
  
nodes$mdid <- as.character(nodes$mdid)  
  
# join network variables  
# first rename Id to wcid  
dat <- nodes %>%  
 rename(Id = mdid) %>%  
 select(Id, age, lsir, 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 lsir black success recidFlag recidDate gap  
## 1 10003 23 32 0 1 0 09/04/2009 1996 days  
## 2 10004 37 31 0 1 0 09/04/2009 2069 days  
## 3 10011 29 21 0 1 0 09/04/2009 1998 days  
## 4 10013 23 21 0 1 0 09/04/2009 1950 days  
## 5 10014 27 33 0 1 0 09/04/2009 2013 days  
## 6 10020 20 27 1 1 0 09/04/2009 2048 days  
## days\_in\_program last\_month\_min\_eigen\_cent last\_month\_max\_eigen\_cent  
## 1 178 0 0.30037033  
## 2 125 0 0.74822298  
## 3 148 0 0.14887343  
## 4 154 0 0.08900336  
## 5 179 0 0.26516136  
## 6 126 0 0.09461094  
## last\_month\_mean\_eigen\_cent last\_month\_median\_eigen\_cent  
## 1 0.10265144 0.05511771  
## 2 0.20648361 0.03885573  
## 3 0.05946920 0.04450168  
## 4 0.03131526 0.01812884  
## 5 0.09086889 0.04915711  
## 6 0.03063047 0.01395547  
## first\_month\_min\_eigen\_cent first\_month\_max\_eigen\_cent  
## 1 6.686340e-03 0.69067962  
## 2 4.132271e-18 0.19322713  
## 3 1.106229e-16 0.88055202  
## 4 0.000000e+00 0.08900336  
## 5 6.954163e-04 0.06401096  
## 6 6.029364e-03 0.04239213  
## first\_month\_mean\_eigen\_cent first\_month\_median\_eigen\_cent min\_eigen\_cent  
## 1 0.25524528 0.161807588 0  
## 2 0.08806488 0.079516189 0  
## 3 0.43933098 0.438385960 0  
## 4 0.03131526 0.018128844 0  
## 5 0.02065874 0.008964286 0  
## 6 0.01784234 0.011473941 0  
## max\_eigen\_cent mean\_eigen\_cent median\_eigen\_cent  
## 1 0.69067962 0.21266986 0.12478159  
## 2 1.00000000 0.25784129 0.11688207  
## 3 0.88055202 0.27916723 0.14887343  
## 4 0.08900336 0.03131526 0.01812884  
## 5 0.68374349 0.18066291 0.03865921  
## 6 0.88006341 0.19208745 0.06850154

# 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 or recidivism. 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 = -8.4398, df = 1157, p-value < 2.2e-16  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.2943202 -0.1858137  
## sample estimates:  
## cor   
## -0.2408192

# 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 = -8.1499, df = 1157, p-value = 9.383e-16  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.2867386 -0.1778069  
## sample estimates:  
## cor   
## -0.2330035

# 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 = -2.3081, df = 1157, p-value = 0.02117  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.12479678 -0.01015842  
## sample estimates:  
## cor   
## -0.06770105

# 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 = -6.1361, df = 1157, p-value = 1.16e-09  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.2327323 -0.1211858  
## sample estimates:  
## cor   
## -0.1775292

# 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 +  
 lsir +  
 black +  
 days\_in\_program +  
 last\_month\_max\_eigen\_cent,  
 data = dat,  
 family = binomial(link = 'logit'))  
  
last\_month\_mean\_model <- glm(success ~  
 age +  
 lsir +  
 black +  
 days\_in\_program +  
 last\_month\_mean\_eigen\_cent,  
 data = dat,  
 family = binomial(link = 'logit'))  
  
max\_model <- glm(success ~  
 age +  
 lsir +  
 black +  
 days\_in\_program +  
 max\_eigen\_cent,  
 data = dat,  
 family = binomial(link = 'logit'))  
  
mean\_model <- glm(success ~  
 age +  
 lsir +  
 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) 3.13 \*\* 3.02 \*\* 3.28 \*\* 3.16 \*\*   
## (1.02) (1.01) (1.03) (1.01)   
## age 0.02 0.02 0.01 0.01   
## (0.02) (0.02) (0.02) (0.02)   
## lsir -0.22 \*\*\* -0.23 \*\*\* -0.22 \*\*\* -0.22 \*\*\*  
## (0.02) (0.02) (0.02) (0.02)   
## black -0.99 \*\*\* -1.01 \*\*\* -0.94 \*\* -1.00 \*\*\*  
## (0.29) (0.29) (0.29) (0.29)   
## days\_in\_program 0.05 \*\*\* 0.04 \*\*\* 0.05 \*\*\* 0.05 \*\*\*  
## (0.00) (0.00) (0.01) (0.00)   
## last\_month\_max\_eigen\_cent -2.36 \*\*\*   
## (0.47)   
## last\_month\_mean\_eigen\_cent -3.46 \*\*\*   
## (0.82)   
## max\_eigen\_cent -2.39 \*\*\*   
## (0.53)   
## mean\_eigen\_cent -4.83 \*\*\*  
## (1.06)   
## ------------------------------------------------------------------------------  
## AIC 458.40 465.86 461.62 462.31   
## BIC 488.27 495.72 491.49 492.18   
## Log Likelihood -223.20 -226.93 -224.81 -225.16   
## Deviance 446.40 453.86 449.62 450.31   
## Num. obs. 1072 1072 1072 1072   
## ==============================================================================  
## \*\*\* 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)),  
 lsir = rep(mean(data$lsir, na.rm = TRUE), nrow(data)),  
 black = rep(0, nrow(data)),  
 days\_in\_program = rep(mean(data$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

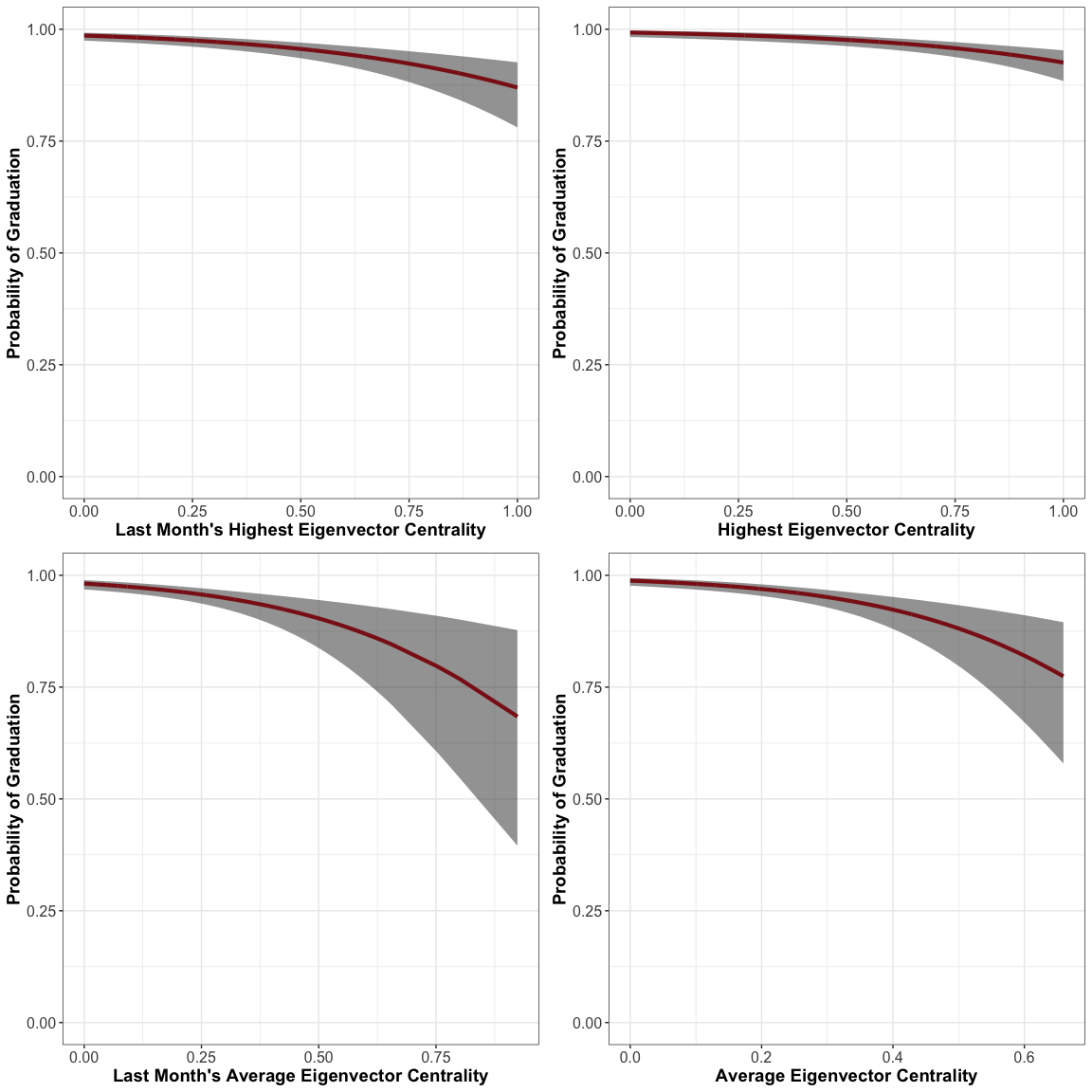
max\_eigen\_cent\_pred\_prob <- get\_pred\_prob\_plot(max\_model,   
 "max\_eigen\_cent",  
 "Highest Eigenvector Centrality")

## 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$age, na.rm = TRUE), length(input\_vec)),  
 lsir = rep(mean(data$lsir, 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] 0.9900810 0.9822509 0.9684365 0.9444781 0.9041342

# 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] 0.9869867 0.9792547 0.9670819 0.9481446 0.9192229

# 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] 0.9907064 0.9827482 0.9681941 0.9420851 0.8968277

# 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] 0.9891420 0.9812682 0.9678701 0.9454219 0.9087682

# Descriptive Stats

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

## Time difference of 2844 days

summary(dat)

## Id age lsir black   
## Length:1159 Min. :18.00 Min. : 0.00 Min. :0.0000   
## Class :character 1st Qu.:24.00 1st Qu.:22.00 1st Qu.:0.0000   
## Mode :character Median :30.00 Median :26.00 Median :0.0000   
## Mean :31.04 Mean :25.46 Mean :0.2079   
## 3rd Qu.:37.00 3rd Qu.:30.00 3rd Qu.:0.0000   
## Max. :60.00 Max. :57.00 Max. :1.0000   
## NA's :87   
## success recidFlag recidDate gap   
## Min. :0.0000 Min. :0.0000 Length:1159 Length:1159   
## 1st Qu.:1.0000 1st Qu.:0.0000 Class :character Class :difftime   
## Median :1.0000 Median :0.0000 Mode :character Mode :numeric   
## Mean :0.8619 Mean :0.1821   
## 3rd Qu.:1.0000 3rd Qu.:0.0000   
## Max. :1.0000 Max. :1.0000   
##   
## days\_in\_program last\_month\_min\_eigen\_cent last\_month\_max\_eigen\_cent  
## Min. : 4 Min. :0.00000 Min. :0.0000   
## 1st Qu.:117 1st Qu.:0.00000 1st Qu.:0.1614   
## Median :134 Median :0.00000 Median :0.2962   
## Mean :135 Mean :0.03336 Mean :0.3529   
## 3rd Qu.:164 3rd Qu.:0.03139 3rd Qu.:0.4931   
## Max. :182 Max. :0.76955 Max. :1.0000   
##   
## last\_month\_mean\_eigen\_cent last\_month\_median\_eigen\_cent  
## Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.0690 1st Qu.:0.03978   
## Median :0.1350 Median :0.10531   
## Mean :0.1698 Mean :0.14645   
## 3rd Qu.:0.2447 3rd Qu.:0.20748   
## Max. :0.9235 Max. :1.00000   
##   
## first\_month\_min\_eigen\_cent first\_month\_max\_eigen\_cent  
## Min. :0.0000000 Min. :0.0000   
## 1st Qu.:0.0000000 1st Qu.:0.1962   
## Median :0.0000721 Median :0.3336   
## Mean :0.0363867 Mean :0.3952   
## 3rd Qu.:0.0520294 3rd Qu.:0.5456   
## Max. :0.5826417 Max. :1.0000   
##   
## first\_month\_mean\_eigen\_cent first\_month\_median\_eigen\_cent  
## Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.0900 1st Qu.:0.06146   
## Median :0.1620 Median :0.13362   
## Mean :0.1891 Mean :0.16245   
## 3rd Qu.:0.2654 3rd Qu.:0.22706   
## Max. :0.7450 Max. :0.89780   
##   
## min\_eigen\_cent max\_eigen\_cent mean\_eigen\_cent median\_eigen\_cent  
## Min. :0.000000 Min. :0.0000 Min. :0.0000 Min. :0.00000   
## 1st Qu.:0.000000 1st Qu.:0.4242 1st Qu.:0.1260 1st Qu.:0.07946   
## Median :0.000000 Median :0.6176 Median :0.1919 Median :0.14515   
## Mean :0.003394 Mean :0.6256 Mean :0.2097 Mean :0.16811   
## 3rd Qu.:0.000000 3rd Qu.:0.8542 3rd Qu.:0.2791 3rd Qu.:0.23377   
## Max. :0.328016 Max. :1.0000 Max. :0.6594 Max. :0.71200   
##

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   
## 2.209459e+03 8.372237e+00   
## lsir black   
## 7.187878e+00 4.060073e-01   
## success recidFlag   
## 3.451014e-01 3.860552e-01   
## recidDate gap   
## NA NA   
## days\_in\_program last\_month\_min\_eigen\_cent   
## 3.440706e+01 7.224133e-02   
## last\_month\_max\_eigen\_cent last\_month\_mean\_eigen\_cent   
## 2.502803e-01 1.370406e-01   
## last\_month\_median\_eigen\_cent first\_month\_min\_eigen\_cent   
## 1.451763e-01 6.445804e-02   
## first\_month\_max\_eigen\_cent first\_month\_mean\_eigen\_cent   
## 2.578225e-01 1.308426e-01   
## first\_month\_median\_eigen\_cent min\_eigen\_cent   
## 1.372359e-01 1.946869e-02   
## max\_eigen\_cent mean\_eigen\_cent   
## 2.622255e-01 1.144890e-01   
## median\_eigen\_cent   
## 1.238580e-01

# 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