Hierarchy in a TC, Facility 1 Men’s Unit 2

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 seniority correlate with hierarchy? Does maximum position within the hierarchy correlate with outcomes, such as graduation or recidivism?

# 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  
}

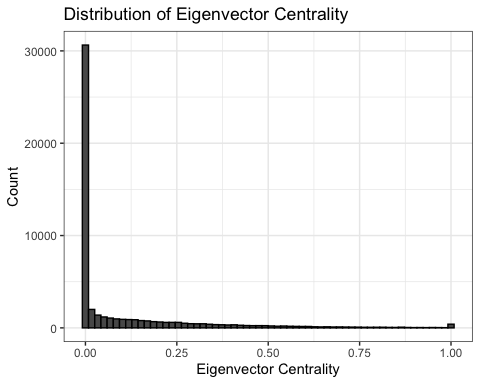
# Calculate Eigenvector Centrality

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 0.   
## 2 "6780" 1 0.   
## 3 "6781" 1 6.03e- 1  
## 4 "6829" 1 3.90e-18  
## 5 "6832" 1 3.90e-18  
## 6 "6833" 1 3.90e-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")

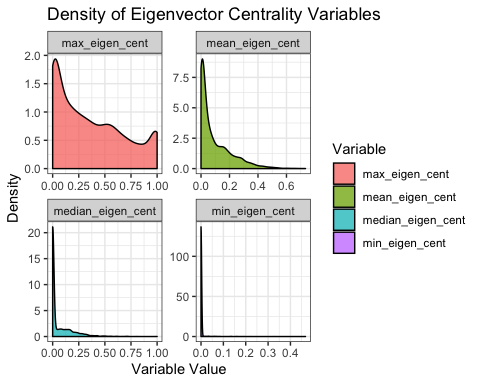


# Create Variables

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.

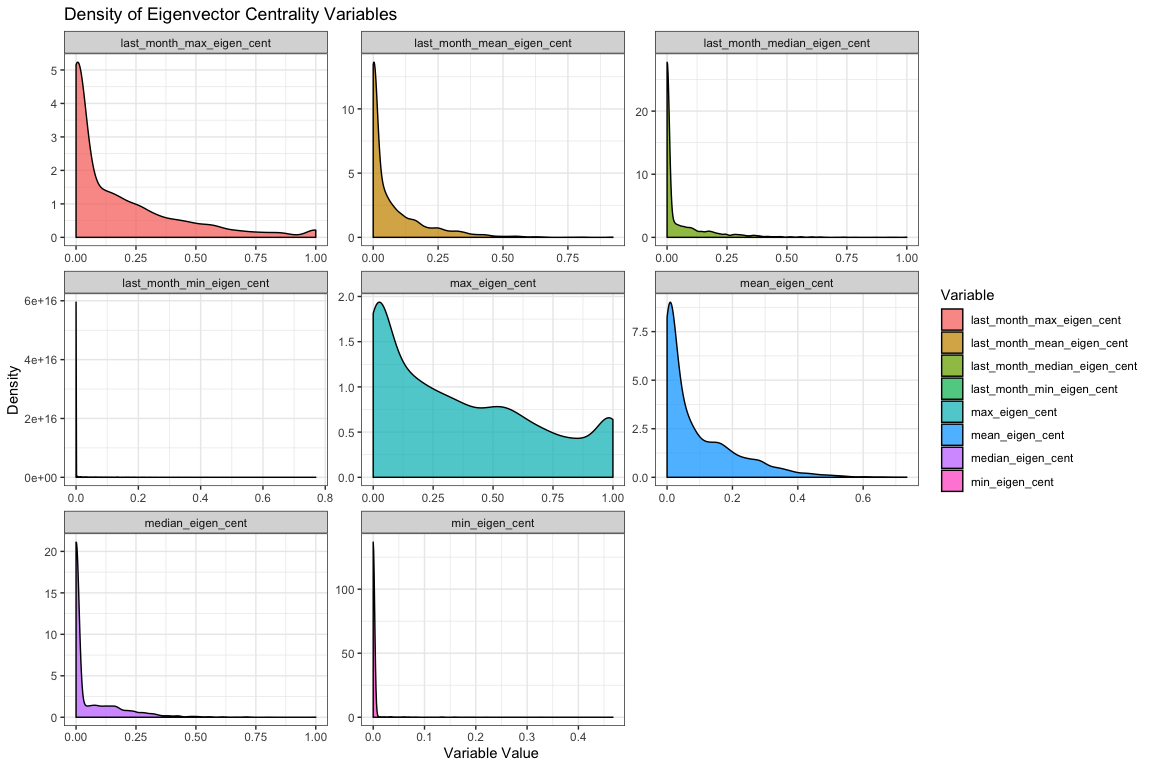
# min. eigen centrality -- highest position ever achieved in hierarchy  
# max. eigen centrality -- lowest position ever achieved in hierarchy  
# mean eigen centrality -- average position in the hierarchy  
# median eigen centrality -- another measure of central tendency  
nodal\_eigen <- eigen\_df %>%  
 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.:0.06124   
## Mode :character Median :0.000000 Median :0.29253   
## Mean :0.001738 Mean :0.36615   
## 3rd Qu.:0.000000 3rd Qu.:0.60300   
## Max. :0.467261 Max. :1.00000   
## mean\_eigen\_cent median\_eigen\_cent  
## Min. :0.000000 Min. :0.00000   
## 1st Qu.:0.006819 1st Qu.:0.00000   
## Median :0.042348 Median :0.00000   
## Mean :0.093161 Mean :0.06114   
## 3rd Qu.:0.149452 3rd Qu.:0.09185   
## Max. :0.733465 Max. :1.00000

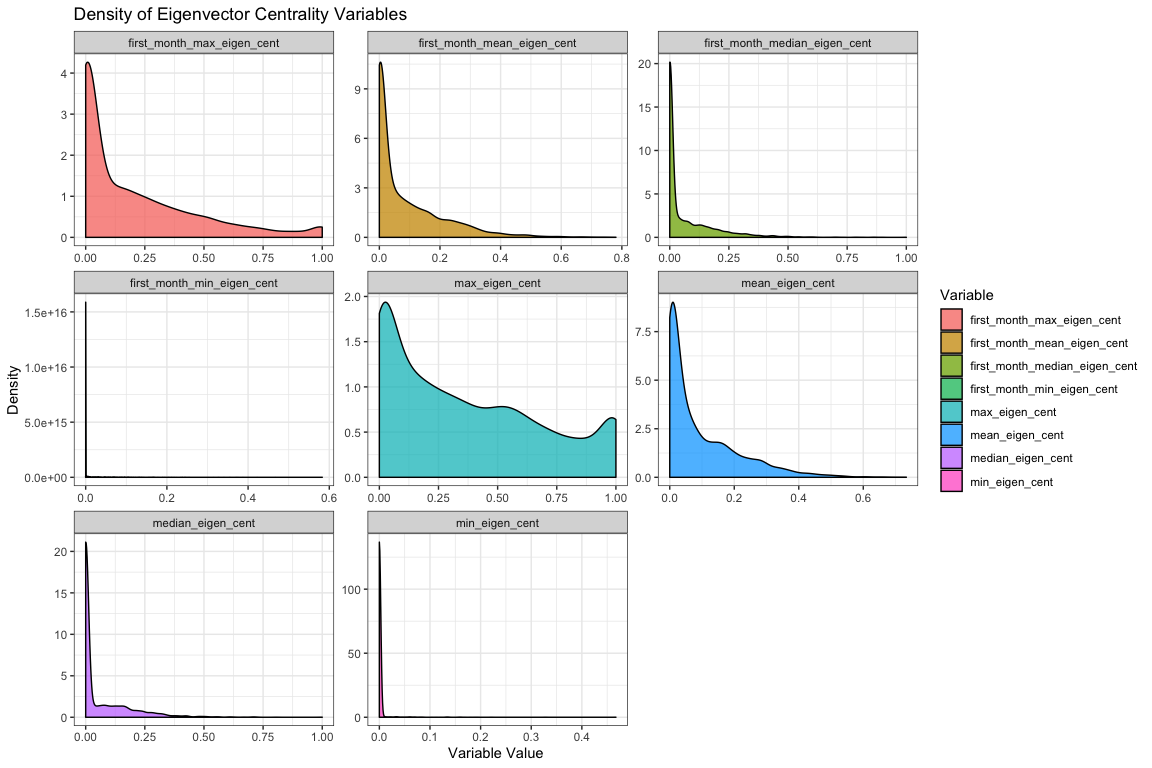
get\_last\_month\_variables <- function(id){  
 id\_df <- eigen\_df %>%   
 filter(Id == id) %>%   
 arrange(t) %>%   
 tail(4) %>%  
 summarize(Id = unique(Id),  
 last\_month\_min\_eigen\_cent = min(eigen\_cent),  
 last\_month\_max\_eigen\_cent = max(eigen\_cent),  
 last\_month\_mean\_eigen\_cent = 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. :0.0000000   
## Class :character 1st Qu.:0.00000 1st Qu.:0.0000001   
## Mode :character Median :0.00000 Median :0.0680333   
## Mean :0.01235 Mean :0.1766753   
## 3rd Qu.:0.00000 3rd Qu.:0.2700199   
## Max. :0.76955 Max. :1.0000000   
## 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 :0.02166 Median :0.0000054   
## Mean :0.07607 Mean :0.0575727   
## 3rd Qu.:0.10821 3rd Qu.:0.0680474   
## 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.:0.0000125   
## Mode :character Median :0.00000 Median :0.0912850   
## Mean :0.01323 Mean :0.2026845   
## 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. :0.0000000   
## 1st Qu.:0.0000031 1st Qu.:0.0000000   
## Median :0.0302873 Median :0.0004976   
## Mean :0.0861926 Mean :0.0643624   
## 3rd Qu.:0.1370388 3rd Qu.:0.0888436   
## Max. :0.7803747 Max. :1.0000000

# 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/F1M2"), stringsAsFactors = FALSE, header = TRUE)  
  
sum(!is.na(nodes$mdid))

## [1] 460

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

## [1] 428

# 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  
nodes$lsir <- as.numeric(nodes$lsir)

## Warning: NAs introduced by coercion

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 10391 19 NA 1 1 0 09/04/2009 1799 days  
## 2 10438 19 28 0 1 0 09/04/2009 1867 days  
## 3 10477 23 44 0 1 0 09/04/2009 1799 days  
## 4 10563 38 NA 0 1 0 09/04/2009 1817 days  
## 5 10568 19 24 0 1 0 09/04/2009 1845 days  
## 6 10585 33 25 1 1 0 09/04/2009 1806 days  
## days\_in\_program last\_month\_min\_eigen\_cent last\_month\_max\_eigen\_cent  
## 1 179 0.000000e+00 6.791937e-17  
## 2 179 2.772149e-16 1.101909e-02  
## 3 179 0.000000e+00 4.639245e-17  
## 4 179 0.000000e+00 1.024624e-07  
## 5 146 0.000000e+00 2.589594e-02  
## 6 179 0.000000e+00 7.854151e-03  
## last\_month\_mean\_eigen\_cent last\_month\_median\_eigen\_cent  
## 1 2.198814e-17 1.001660e-17  
## 2 2.883447e-03 2.573485e-04  
## 3 1.747619e-17 1.175615e-17  
## 4 2.561559e-08 2.702351e-17  
## 5 6.473985e-03 1.409438e-16  
## 6 1.963607e-03 1.394044e-07  
## first\_month\_min\_eigen\_cent first\_month\_max\_eigen\_cent  
## 1 0 1.665335e-16  
## 2 0 2.668316e-16  
## 3 0 8.326673e-17  
## 4 0 3.172212e-03  
## 5 0 3.333333e-01  
## 6 0 3.629925e-01  
## first\_month\_mean\_eigen\_cent first\_month\_median\_eigen\_cent min\_eigen\_cent  
## 1 4.163336e-17 0.000000e+00 0  
## 2 8.058568e-17 2.775558e-17 0  
## 3 2.131848e-17 1.003599e-18 0  
## 4 7.930531e-04 0.000000e+00 0  
## 5 8.597662e-02 5.286575e-03 0  
## 6 9.074813e-02 7.466490e-16 0  
## max\_eigen\_cent mean\_eigen\_cent median\_eigen\_cent  
## 1 0.023892566 0.0014597251 1.664366e-17  
## 2 0.011019091 0.0011083775 1.611714e-16  
## 3 0.053521274 0.0040161629 7.911119e-18  
## 4 0.006458899 0.0007811581 0.000000e+00  
## 5 0.333333333 0.0264475547 8.393845e-17  
## 6 0.362992525 0.0216152796 4.765231e-18

# 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 = -2.2713, df = 400, p-value = 0.02366  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.20834770 -0.01520042  
## sample estimates:  
## cor   
## -0.1128399

# 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 = -2.4326, df = 400, p-value = 0.01543  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.21599777 -0.02320816  
## sample estimates:  
## cor   
## -0.1207413

# 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 = -0.066697, df = 400, p-value = 0.9469  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.10110909 0.09450324  
## sample estimates:  
## cor   
## -0.003334827

# 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 = -1.7543, df = 400, p-value = 0.08014  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.1836181 0.0105170  
## sample estimates:  
## cor   
## -0.08738012

# 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) 2.11 2.07 1.83 2.03   
## (2.14) (2.12) (2.14) (2.10)   
## age -0.01 0.00 -0.01 -0.01   
## (0.03) (0.03) (0.03) (0.03)   
## lsir -0.34 \*\*\* -0.34 \*\*\* -0.31 \*\*\* -0.32 \*\*\*  
## (0.06) (0.06) (0.06) (0.06)   
## black -0.14 -0.14 0.06 -0.00   
## (0.70) (0.70) (0.68) (0.68)   
## days\_in\_program 0.08 \*\*\* 0.08 \*\*\* 0.08 \*\*\* 0.08 \*\*\*  
## (0.01) (0.01) (0.01) (0.01)   
## last\_month\_max\_eigen\_cent -3.39 \*\*   
## (1.17)   
## last\_month\_mean\_eigen\_cent -12.67 \*\*   
## (4.51)   
## max\_eigen\_cent -1.34   
## (1.05)   
## mean\_eigen\_cent -13.85   
## (7.07)   
## --------------------------------------------------------------------------  
## AIC 94.18 94.35 99.73 97.92   
## BIC 118.05 118.22 123.61 121.79   
## Log Likelihood -41.09 -41.17 -43.87 -42.96   
## Deviance 82.18 82.35 87.73 85.92   
## Num. obs. 395 395 395 395   
## ==========================================================================  
## \*\*\* 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

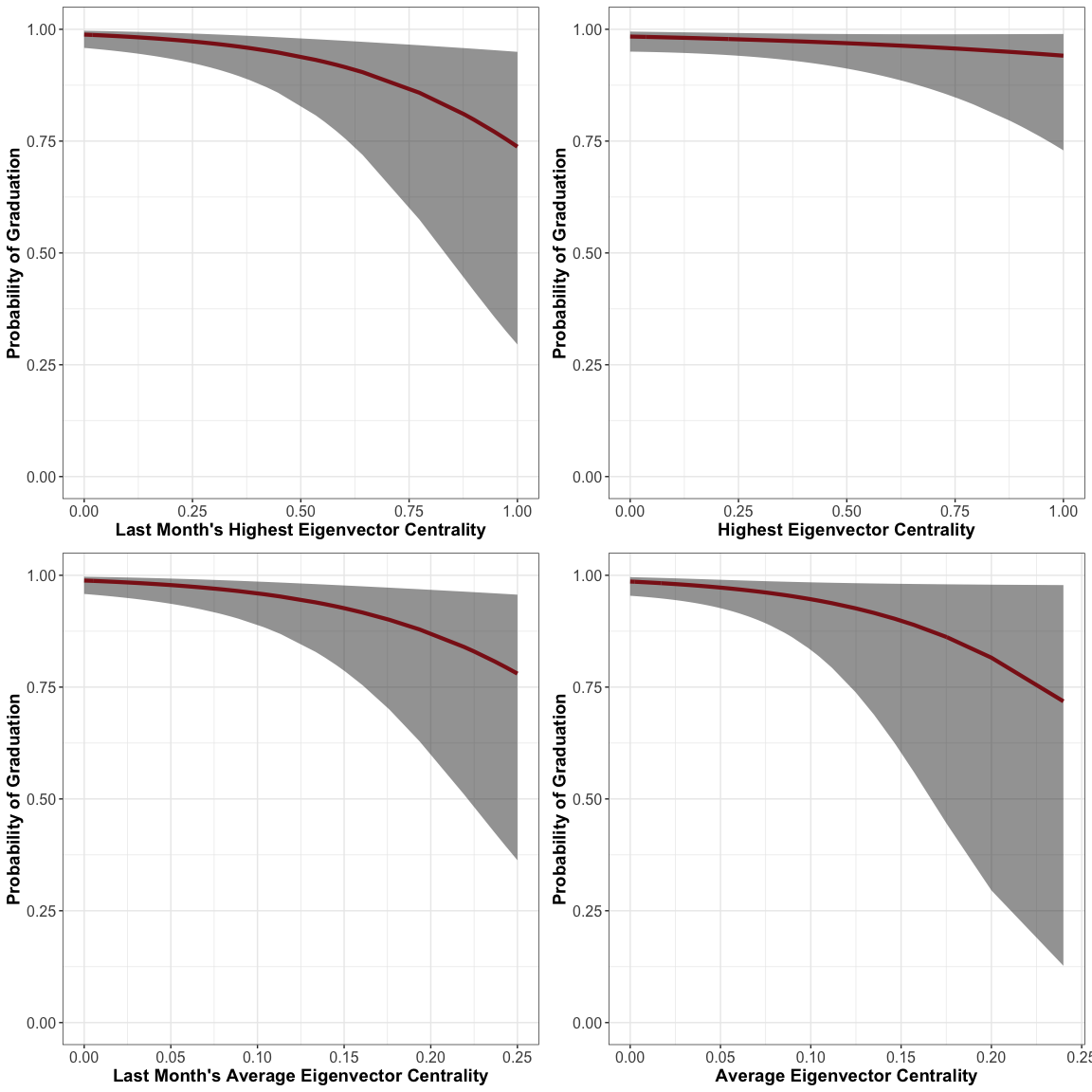
max\_eigen\_cent\_pred\_prob <- get\_pred\_prob\_plot(max\_model,   
 "max\_eigen\_cent",  
 "Highest Eigenvector Centrality")

## 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] 0.9951094 0.9916407 0.9857472 0.9757998 0.9591975

# 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] 0.9949654 0.9915567 0.9858728 0.9764537 0.9610029

# 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] 0.9894793 0.9855276 0.9801215 0.9727518 0.9627537

# 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] 0.9923346 0.9884598 0.9826605 0.9740236 0.9612541

# Descriptive Stats

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

## Time difference of 2844 days

summary(dat)

## Id age lsir black   
## Length:402 Min. :18.00 Min. :13.00 Min. :0.0000   
## Class :character 1st Qu.:23.00 1st Qu.:21.50 1st Qu.:0.0000   
## Mode :character Median :29.00 Median :25.00 Median :0.0000   
## Mean :30.91 Mean :26.01 Mean :0.2388   
## 3rd Qu.:37.75 3rd Qu.:30.50 3rd Qu.:0.0000   
## Max. :60.00 Max. :45.00 Max. :1.0000   
## NA's :7   
## success recidFlag recidDate gap   
## Min. :0.0000 Min. :0.0000 Length:402 Length:402   
## 1st Qu.:1.0000 1st Qu.:0.0000 Class :character Class :difftime   
## Median :1.0000 Median :0.0000 Mode :character Mode :numeric   
## Mean :0.8507 Mean :0.3756   
## 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000   
##   
## days\_in\_program last\_month\_min\_eigen\_cent last\_month\_max\_eigen\_cent  
## Min. : 8.0 Min. :0.0000000 Min. :0.0000000   
## 1st Qu.:131.0 1st Qu.:0.0000000 1st Qu.:0.0000000   
## Median :147.0 Median :0.0000000 Median :0.0000002   
## Mean :139.2 Mean :0.0004702 Mean :0.0548110   
## 3rd Qu.:161.0 3rd Qu.:0.0000000 3rd Qu.:0.0198609   
## Max. :179.0 Max. :0.1031561 Max. :1.0000000   
##   
## last\_month\_mean\_eigen\_cent last\_month\_median\_eigen\_cent  
## Min. :0.000e+00 Min. :0.000000   
## 1st Qu.:0.000e+00 1st Qu.:0.000000   
## Median :6.000e-08 Median :0.000000   
## Mean :1.484e-02 Mean :0.002028   
## 3rd Qu.:5.763e-03 3rd Qu.:0.000000   
## Max. :2.500e-01 Max. :0.161137   
##   
## first\_month\_min\_eigen\_cent first\_month\_max\_eigen\_cent  
## Min. :0.0000000 Min. :0.00000   
## 1st Qu.:0.0000000 1st Qu.:0.00000   
## Median :0.0000000 Median :0.00000   
## Mean :0.0006773 Mean :0.05499   
## 3rd Qu.:0.0000000 3rd Qu.:0.01123   
## Max. :0.1031561 Max. :1.00000   
##   
## first\_month\_mean\_eigen\_cent first\_month\_median\_eigen\_cent  
## Min. :0.000000 Min. :0.000000   
## 1st Qu.:0.000000 1st Qu.:0.000000   
## Median :0.000000 Median :0.000000   
## Mean :0.016834 Mean :0.005819   
## 3rd Qu.:0.003353 3rd Qu.:0.000000   
## Max. :0.311370 Max. :0.209192   
##   
## min\_eigen\_cent max\_eigen\_cent mean\_eigen\_cent   
## Min. :0.0000000 Min. :0.0000000 Min. :0.0000000   
## 1st Qu.:0.0000000 1st Qu.:0.0001671 1st Qu.:0.0000101   
## Median :0.0000000 Median :0.0398383 Median :0.0027869   
## Mean :0.0004637 Mean :0.1556597 Mean :0.0158750   
## 3rd Qu.:0.0000000 3rd Qu.:0.2303147 3rd Qu.:0.0182151   
## Max. :0.1031561 Max. :1.0000000 Max. :0.2398776   
##   
## median\_eigen\_cent   
## Min. :0.0000000   
## 1st Qu.:0.0000000   
## Median :0.0000000   
## Mean :0.0006107   
## 3rd Qu.:0.0000000   
## Max. :0.1031561   
##

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

## Id age   
## 1.252956e+03 9.376789e+00   
## lsir black   
## 6.324751e+00 4.268852e-01   
## success recidFlag   
## 3.567824e-01 4.848865e-01   
## recidDate gap   
## NA NA   
## days\_in\_program last\_month\_min\_eigen\_cent   
## 3.627536e+01 6.056429e-03   
## last\_month\_max\_eigen\_cent last\_month\_mean\_eigen\_cent   
## 1.591986e-01 4.109067e-02   
## last\_month\_median\_eigen\_cent first\_month\_min\_eigen\_cent   
## 1.174096e-02 6.382304e-03   
## first\_month\_max\_eigen\_cent first\_month\_mean\_eigen\_cent   
## 1.569057e-01 4.659996e-02   
## first\_month\_median\_eigen\_cent min\_eigen\_cent   
## 2.506904e-02 6.055525e-03   
## max\_eigen\_cent mean\_eigen\_cent   
## 2.414793e-01 2.983110e-02   
## median\_eigen\_cent   
## 6.144099e-03

# 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