PDA Final Project - Statistical Learning of Scott County Injection Network Data

Alyson Singleton

12/20/2019

Introduction

Significant progress has been made in reducing HIV incidence among people who inject drugs (PWID) in the United States (US), but these declines have stalled in the face of frequent rapid HIV transmission events occurring during an escalating drug overdose epidemic [Singh S, et al. Annals of Internal Medicine 2018, Des Jarlais DC, et al. AIDS 2016]. The largest ever of these events among PWID in a non-urban setting in the US occurred in Scott County, Indiana in 2015, when nearly 200 people in a community of 25,000 individuals were newly diagnosed with HIV infection.

Several factors have been associated with rapid HIV transmission in PWID, including a lack of awareness that HIV is a threat in the local setting, high frequency of needle/syringe sharing, inaccessibility of sterile injecting equipment, recent changes in drug usage patterns, and large, interconnected risk networks [Des Jarlais DC, et al. AIDS 2016]. Concerning the structure of those networks, certain micro-structures can further exacerbate HIV transmission and persistence of infection. While instances of rapid HIV transmission are more likely in settings with these characteristics [Des Jarlais DC, et al. AIDS 2016], they are not inevitable, with the potential to intervene on any of these factors, including on the network itself.

The ongoing opioid crisis makes this line of research particularly timely. The 2015 outbreak in Indiana, driven largely by high-frequency injection of the prescription opioid oxymorphone, has raised concerns about the potential for other outbreaks of HIV among PWID. An understanding of the impact of network structure on HIV transmission dynamics will be fundamental to mitigating and preventing future outbreaks.

This final project used statistical learning techniques to investigate which structural network characteristics might be most predictive of the magnitude and velocity of subsequent epidemic behavior. I characterized the observed risk network from the HIV outbreak in Scott County, Indiana, as a case study. I generated a series of stochastic networks with comparable numbers of individuals and ties to the observed network to identify how HIV transmission is impacted by varying structural network characteristics. I created the simulated network data as a part of my work as a Research Assistant with the Marshall Research Group. This final project focuses on prediction, an analysis that was previously outside of the scope of the work, but that I have always been interested in. I will now briefly outline the methods used to create the data set as they are imperative for understanding which structures were forced into the networks and which were emergent and stochastically created.

Background on Observational Data

Our research utilizes previously published information on the observed contact tracing network from the 2015 outbreak in Scott County, Indiana [Campbell EM, et al. Journal of Infectious Diseases 2017, Peters PJ, et al. New England Journal of Medicine 2016]. The investigation of the outbreak was part of an emergency response conducted by the Indiana State Department of Health and the Centres for Disease Control and Prevention. The contact tracing investigation involved disease intervention specialists eliciting the names of past-year sexual and injection contacts of individuals newly diagnosed during the outbreak and offering HIV testing to these contacts.

The observed risk network comprised 420 individuals and 913 named ties, including the main component with 411 individuals. Among these ties, 79.2% were injection-related only, 7.9% were between sex-related only,

and 12.9% were multiplex (i.e., were between sexual partners who shared injection equipment). Nearly half of the individuals included in this network (44.5%) were diagnosed with HIV infection during the outbreak investigation.

Background on Simulation Methods

Model Setting

We adapted a previously published version of the TITAN Model, an agent-based model, to explore rapid HIV transmission in Scott County, Indiana [Goedel WC, et al. Clinical Infectious Diseases 2019]. Agent-based modelling is a simulation method that represents micro-level interactions between individual entities called agents to understand the emergence of macro-level trends. Our model was parameterized, where possible, using published information on injection and sexual behaviour in Scott County and supplemented with estimates from the literature where necessary and then calibrated to the observed number of incident HIV infections.

There were very few rural injection networks to serve as appropriate comparators to the observed network. As we aimed to characterize and identify the potentially unique qualities of this network and their contribution to HIV transmission, we instead compared the observed network to agent-built simulated networks.

Each iteration of the simulation models HIV transmission in a network of comparable size (420 agents) in discrete time-steps each representing 1 calendar month for 5 years. Each iteration includes the initialization of the model population, the formation of the contact network, the introduction of HIV into the network through a randomly selected agent, and the monitoring of the progression of HIV throughout the population. A total of 1,000 iterations were simulated.

Population Formation

The model first initialized a virtual population. The size of the initial population varied across simulations from 402 to 441, as we aimed for the number of agents included in the simulated risk network (i.e., the number of agents with at least one tie) to be within ten per cent of the number of nodes in the observed network (n = 420). The attributes of agents in the base population were assigned through stochastic processes to achieve the desired gender distribution and gender-specific prevalence of IDU [Campbell EM, et al. Journal of Infectious Diseases 2017]. HIV prevalence in the network was set at 0% at initialization, reflecting an entirely susceptible population.

Network Formation

Following population formation, the model created ties between agents to build a risk network. The number of ties between agents represented a primary calibration target, where we aimed to generate networks with a total number of ties within ten per cent of the observed network (n = 913). This process ensured that the networks had a comparable density to the observed network. These ties were distributed within the agent population through the following processes.

All agents were assumed to be able to participate in sexual behaviour in the model and were assigned a target number of sexual partners from a negative binomial distribution with a mean of one partner [Goedel WC, et al. Clinical Infectious Diseases 2019]. Given the low number of male-male sexual dyads reported during contact tracing [Campbell EM, et al. Journal of Infectious Diseases 2017], only male-female sexual dyads were assumed to be possible in the simulated networks. Agents who inject drugs were given an additional target number of injection partners based on the observed injection-specific degree distribution [Campbell EM, et al. Journal of Infectious Diseases 2017]. This degree distribution was represented with a step function, with each step representing a range of possible target numbers of injection partners that an agent might be assigned. Each step was assigned a probability of occurrence, calculated from the observed injection-specific

degree distribution. An observed 12.9% of the injection ties were also sexual ties and were constructed as such [Campbell EM, et al. Journal of Infectious Diseases 2017].

The network was constructed through an iterative process. The model iterated through the list of agents who had not yet reached their target numbers to match pairs of agents together. It created a list of eligible partners for each agent under the parameters governing partner compatibility (i.e., allowing for only male-female sexual ties to be formed, but ties of any combination of genders for injection ties), only including agents who were also not yet at their target number of partners. Once the model iterated over all agents who needed additional partners, agents were tied to one another based on this pairing algorithm. The pool of agents not yet at their target number was re-built, and the process was repeated until all individuals were paired or there were no more eligible partners for a given agent. This process introduces stochasticity into the structure of the set of simulated networks: the model provided agents target numbers based on the observed distributions while allowing their realized degree to differ.

Introduction of HIV Infection

After the formation of the contact network, a single agent engaging in IDU was chosen to seroconvert spontaneously, thus introducing HIV into the network. Hereafter, we refer to this agent as the initial infection.

HIV Transmission

Briefly, agents were assigned a target number of condomless vaginal intercourse acts per partner per month, drawn from a Poisson distribution with a mean of 13 acts per month [Crosby RA, et al. Annals of Epidemiology 2012]. Sexual acts that included the use of condoms were not explicitly simulated as they were assumed to carry a negligible risk of HIV transmission. Agents who inject drugs were also assigned a target number of injection acts per month from a Poisson distribution with a mean of 150 injection acts [Dasgupta S, et al. AIDS & Behavior 2019]. On average, 34% of these injection acts were estimated to include syringe sharing [Dasgupta S, et al. AIDS & Behavior 2019]. These acts were evaluated as the number of trials (n) in binomial distributions that model HIV transmission, where the probability of success (p) is the probability of transmission associated with particular behaviours.

Data Analysis

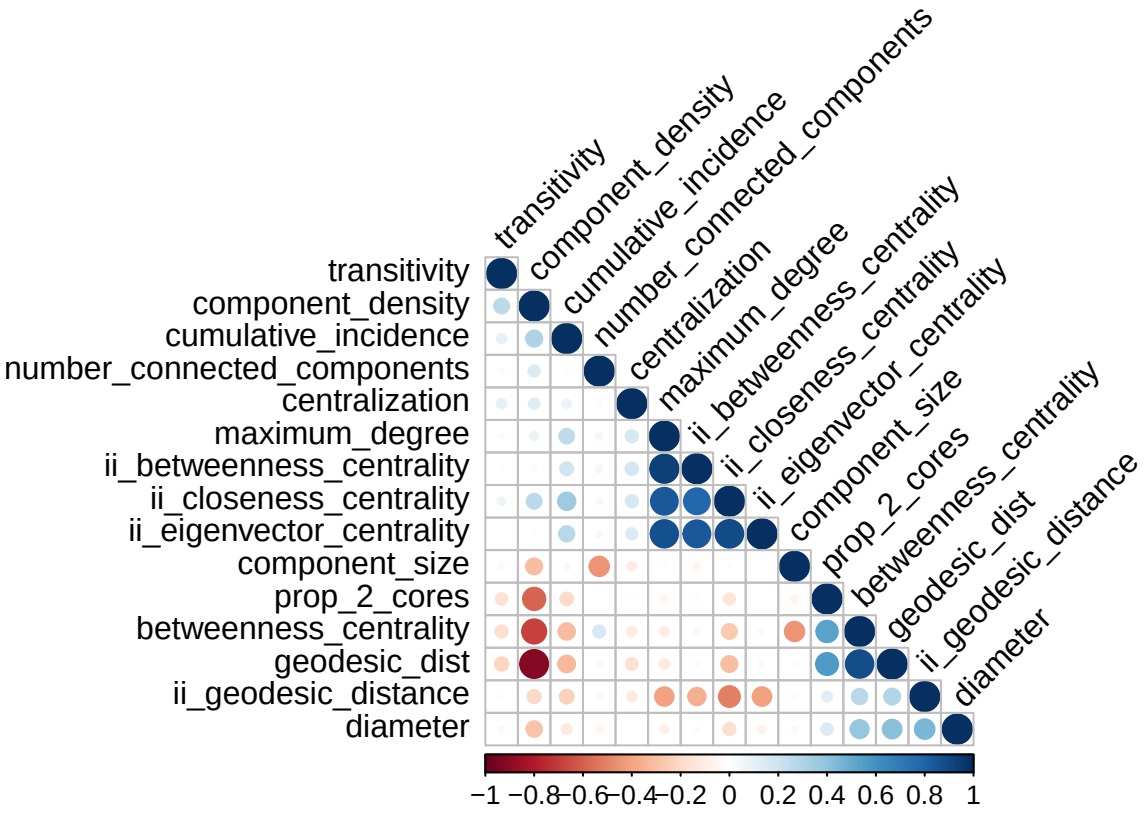
We measured possible differences using network measures that had been previously shown to either facilitate or limit rapid HIV transmission in many past published works. These measures included the number of components in the network (omitting isolated individuals), size of the largest component (referred to as main component), density of the main component, average betweenness centrality of the main component, betweenness centrality of the initial infection, average geodesic distance of the main component, geodesic distance of the initial infection, diameter of the main component, degree of the initial infection, centralization of the main component, the proportion of individuals located in a 2-core in the main component, and transitivity in the main component.

The primary comparison outcome measure across scenarios was the cumulative number of incident HIV infections over the simulation period. We also measured the number of months elapsed until ten incident infections, the number of HIV infections that incited the contact tracing investigation and the subsequent increase in testing activities to investigate how network structure may be related to epidemic velocity.

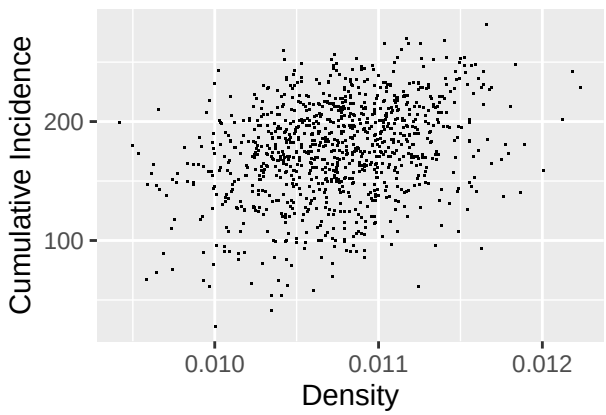
Prediction Analysis

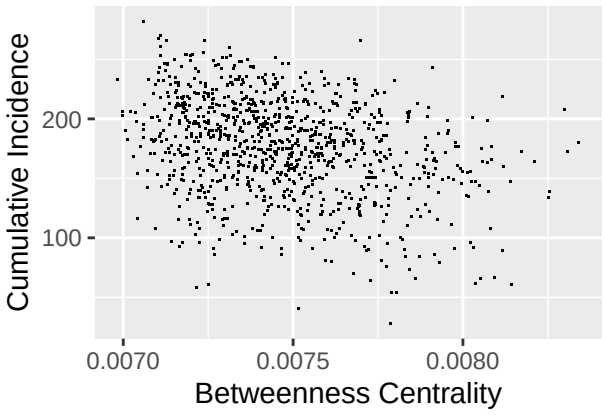
Data Investigation

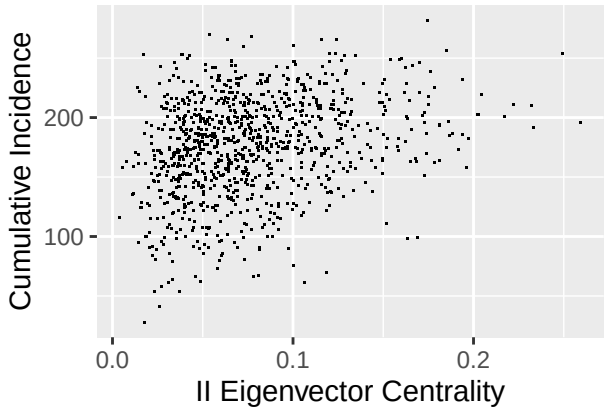
I began my investigation of the data by investigating the potential collinearity between the structural network measurements. Below you can see a correlation plot. I was unsprised to see groups of variables that were especially correlated. One group is a set of measurements on the placement of the randomly selected initial infection location (notated by "ii" for initial infection). There is a second group that is a set of structural characteristics that are mathematically similar, at least given our impletemented conditions on the network. This group is correlated because they investigate similar structures, although they each have valuable nuance and likely can differ more extemely in other types of networks, with different densities, for example. These correlations were part of the impetus for using variable selection methods and PCA. Not only will the following models attend to this through their variable selection processes, but they will likely offer some insight into which out of each of the groups might be the best predictors.

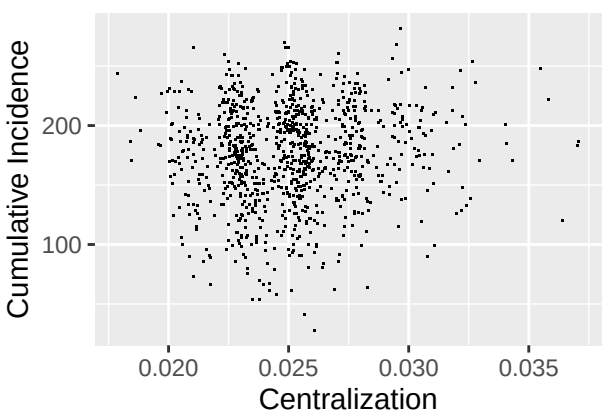


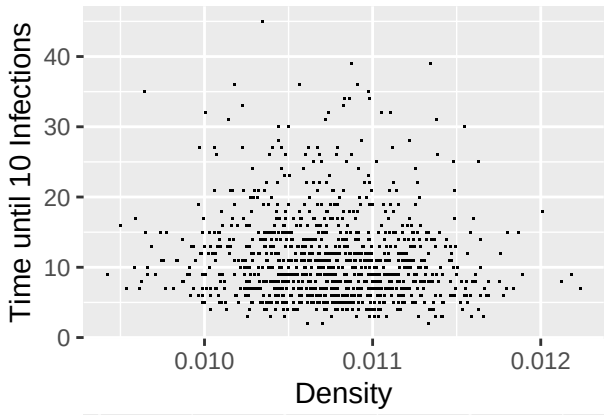
Next, I looked at the shape of the relationships between each of the network measurements and our two outcomes, cumulative infections and time elapsed until 10 infections. Upon inspection, it appears that most variables have weak relationships with the outcome and are generally fairly linear, with exceptions. I decided to assume linearity for this analysis, but recognize that there are a few characteristics that would have benefitted from non-linear representation. I had hoped to have time to investigate this, but in the end I did not. Also, in this analysis I am particularly concerned in interpretability of the coefficients, given the direct application of the outcomes. A few of the plots are displayed below that represent the range of shapes for each of the outcomes.

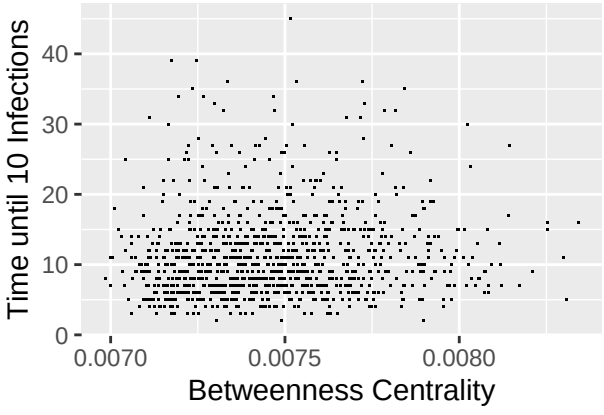

Cumulative Incidence
200
100
0.010
0.011
0.012
Density

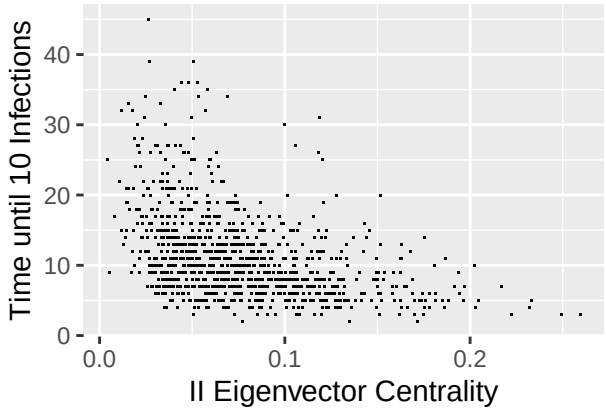

Cumulative Incidence
200
100
0.0070
0.0075
0.0080
Betweenness Centrality

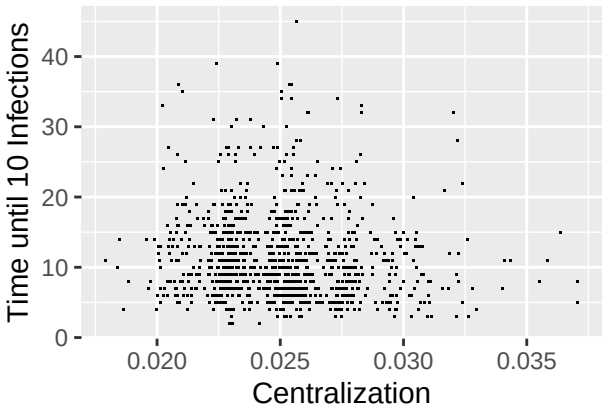

Cumulative Incidence
200
100
0.0
0.1
0.2
II Eigenvector Centrality


Cumulative Incidence
200
100
0.020
0.025
0.030
0.035
Centralization


Time until 10 Infections
40
30
20
10
0
0.010
0.011
0.012
Density


Time until 10 Infections
40
30
20
10
0
0.0070
0.0075
0.0080
Betweenness Centrality


Time until 10 Infections
40
30
20
10
0
0.0
0.1
0.2
II Eigenvector Centrality


Time until 10 Infections
40
30
20
10
0
0.020
0.025
0.030
0.035
Centralization

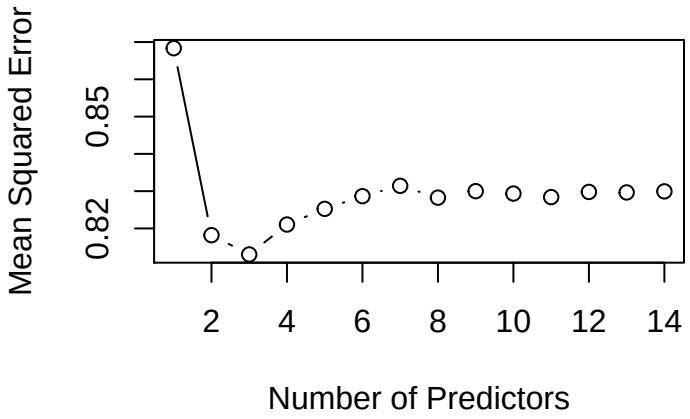
Variable Selection for Prediction of Cumulative Infections

Before beginning my variable selection I standardized all of the network measurements to have mean zero and standard deviation one. This gives the coefficients more appropriate magnitude. This is especially necessary in this data set given that all of the proposed explanatory variables are continuous and have variable ranges (component size 396 to 434 compared to betweenness centrality 0.0069 to 0.0083). By placing everything on a common metric of SD's, we get comparability across all of the coefficients.

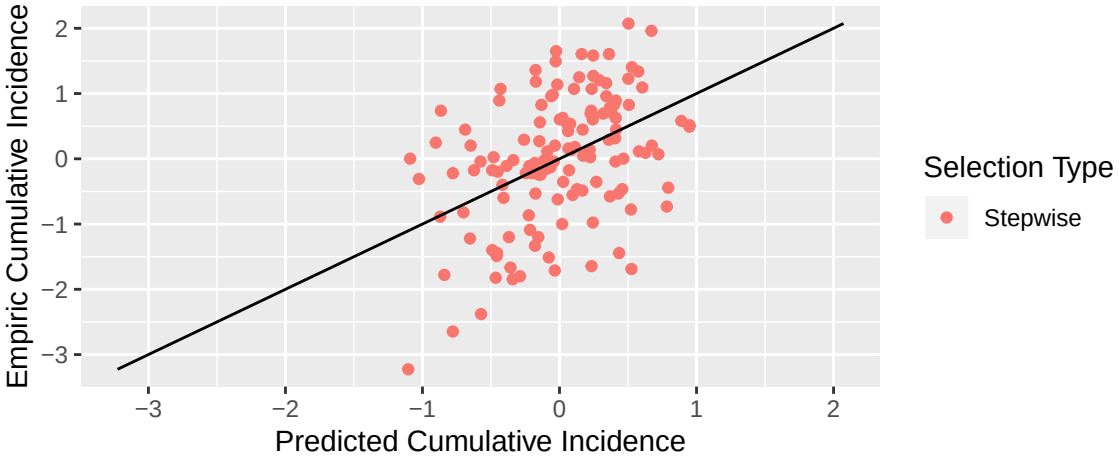
Additionally, I selected 150 random samples from the 1,000 simulations to hold as our testing set. The remaining 850 simulations are used to train the models and are referred to as the training set. I used cross-validation to construct a predictive model with a) stepwise regression; b) lasso regression; c) ridge regression.

Stepwise Variable Selection with Cross Validation

I use k-fold cross-validation to construct a model with stepwise regression. I let $k = 10$ to create 10 total folds to train and validate the model on. Below you can see a plot that compares the number of predictors included against the Mean Squared Error (MSE). At first, the MSE decreases as the number of predictors increases. Eventually, however, the MSE begins to increase. It is at this point that we conclude we have optimized the number of predictors to maximize both the predictive ability of our model and its external validity. In our case for cumulative infection, the number of variables selected by this method is three.

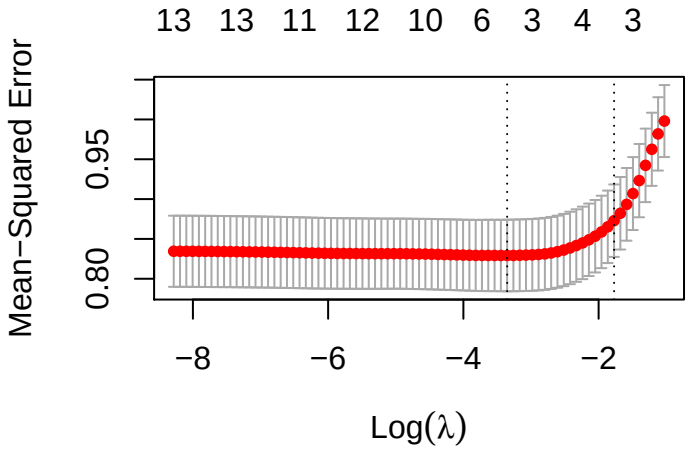


I measure and store both the R^2 value and MSE on the test set to evaluate this model's performance. Below see a plot comparing the predicted number of HIV infections with the empiric.

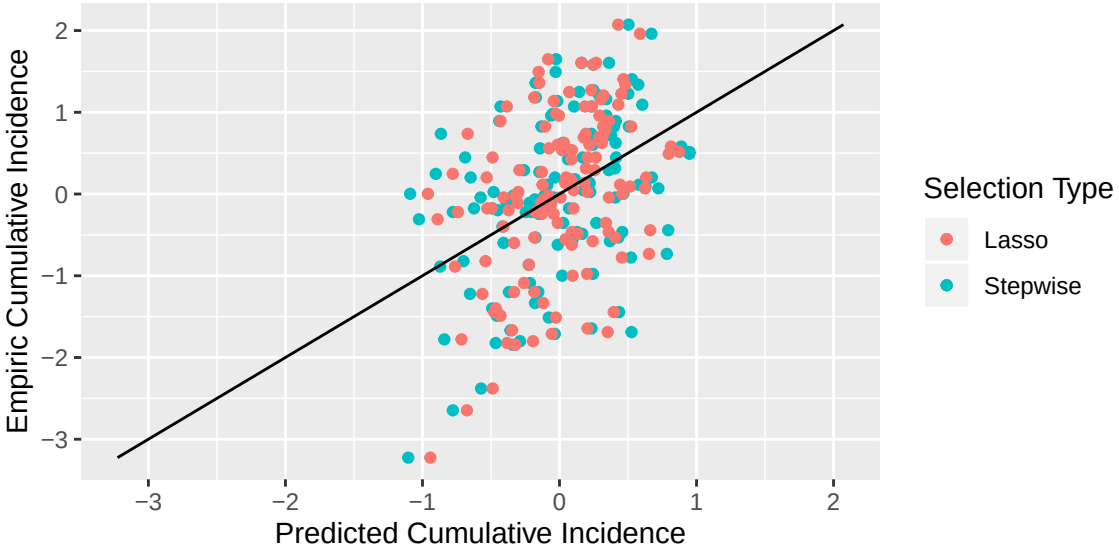


Lasso Variable Selection with Cross Validation

Next, I use k-fold cross-validation to construct a model, this time with lasso regression. I again let $k = 10$ to create 10 total folds to train and validate the model on. Below you can see a plot that compares the log(lambda), a hyperparameter, against the Mean Squared Error (MSE). We choose the lambda that minimizes the mean MSE across the k-folds, which is reported Table 3. The number of non-zero variables selected by this method is five.

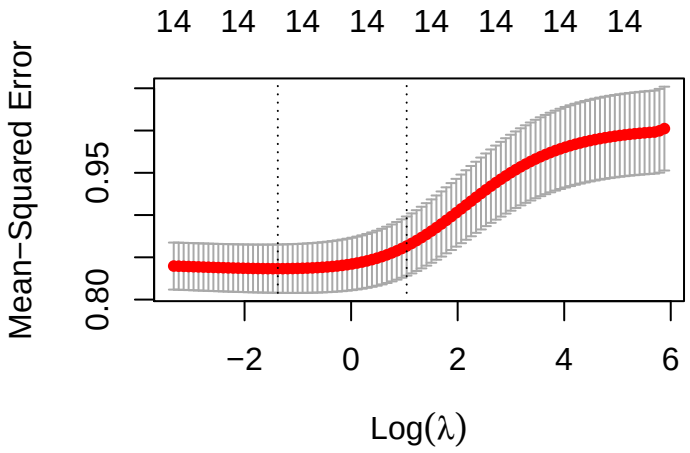


I again measure and store both the R^2 value and MSE on the test set to evaluate this model's performance. Below see a plot comparing the predicted number of HIV infections with the empiric for both stepwise and lasso models.

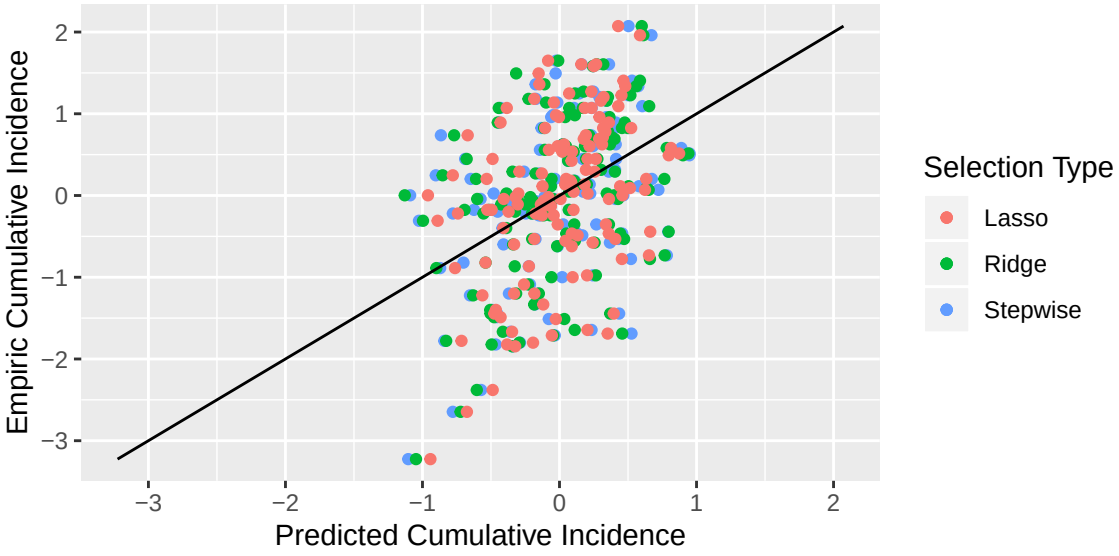


Ridge Variable Selection with Cross Validation

For the final time, I use k-fold cross-validation to construct a model with ridge regression. I again let $k = 10$ to create 10 total folds to train and validate the model on. Below you can see a plot that compares the log(lambda), a hyperparameter, against the Mean Squared Error (MSE). We choose the lambda that minimizes the mean MSE across the k folds, which is reported Table 3.



I again measure and store both the R^2 value and MSE on the test set to evaluate this model's performance. Below see a plot comparing the predicted number of HIV infections with the empiric for all three models. You can see from the plot that all models predict similarly, the predictions vary from the empiric values in the same fashion.



Variable Selection Results

Now that we have constructed and employed each of our models, we can compare the results. Stepwise chooses three variables, Closeness Centrality of the Initial Infection, Main Component Size, and Component Density. Lasso chooses two of the same, Eigenvector Centrality of the Initial Infection and Component Density, but replaces Main Component Size with Average Betweenness Centrality, Geodesic Distance of the Initial Infection, and Eigenvector Centrality of the Initial Infection. Each of the chosen variables have coefficients similar in magnitude and sign across the three methods. All of the coefficients' signs match my intuition for how they would affect the number of cumulative infections. For example, as density increases, the number of ties between individuals increases, and the potential for larger and faster spread of the virus increases. This is matched with a positive coefficient across all three methods. Similarly, the larger the Closeness Centrality of the Initial Infection, the more "influence" the initial infection has and the greater the potential for extreme epidemic behavior. This is also matched with a positive coefficient across all three methods. A formal interpretation of one of the coefficients that can be extrapolated to the others follows.

The coefficient for density is the change in cumulative infections measured in units of SDs for a one SD change in density while holding all other variables constant. Please see all of these values reflected below in Table 1.

Table 1: Cumulative Infections Coefficients Comparison

	Stepwise	Lasso	Ridge
Intercept	0.0000	0.0000	0.0000
Degree of the Initial Infection	NA	0.0000	0.0043
Betweenness Centrality of the Largest Component	NA	-0.1657	-0.0591
Betweenness Centrality of the Initial Infection	NA	0.0000	-0.0555
Closeness Centrality of the Initial Infection	0.2948	0.2388	0.1824
Eigenvector Centrality of the Initial Infection	NA	0.0231	0.1315
Component Size of the Largest Component	0.1498	0.0000	0.1009
Number of Connected Components	NA	0.0000	-0.0081
Density of the Largest Component	0.2497	0.0701	0.1967
Geodesic Distance of the Largest Component	NA	0.0000	-0.0166
Geodesic Distance of the Initial Infection	NA	-0.0027	-0.0435
Centralization of the Largest Component	NA	0.0000	0.0224
Proportion of 2-Cores	NA	0.0000	0.0179
Transitivity	NA	0.0000	0.0246
Diameter of the Largest Component	NA	0.0000	0.0163

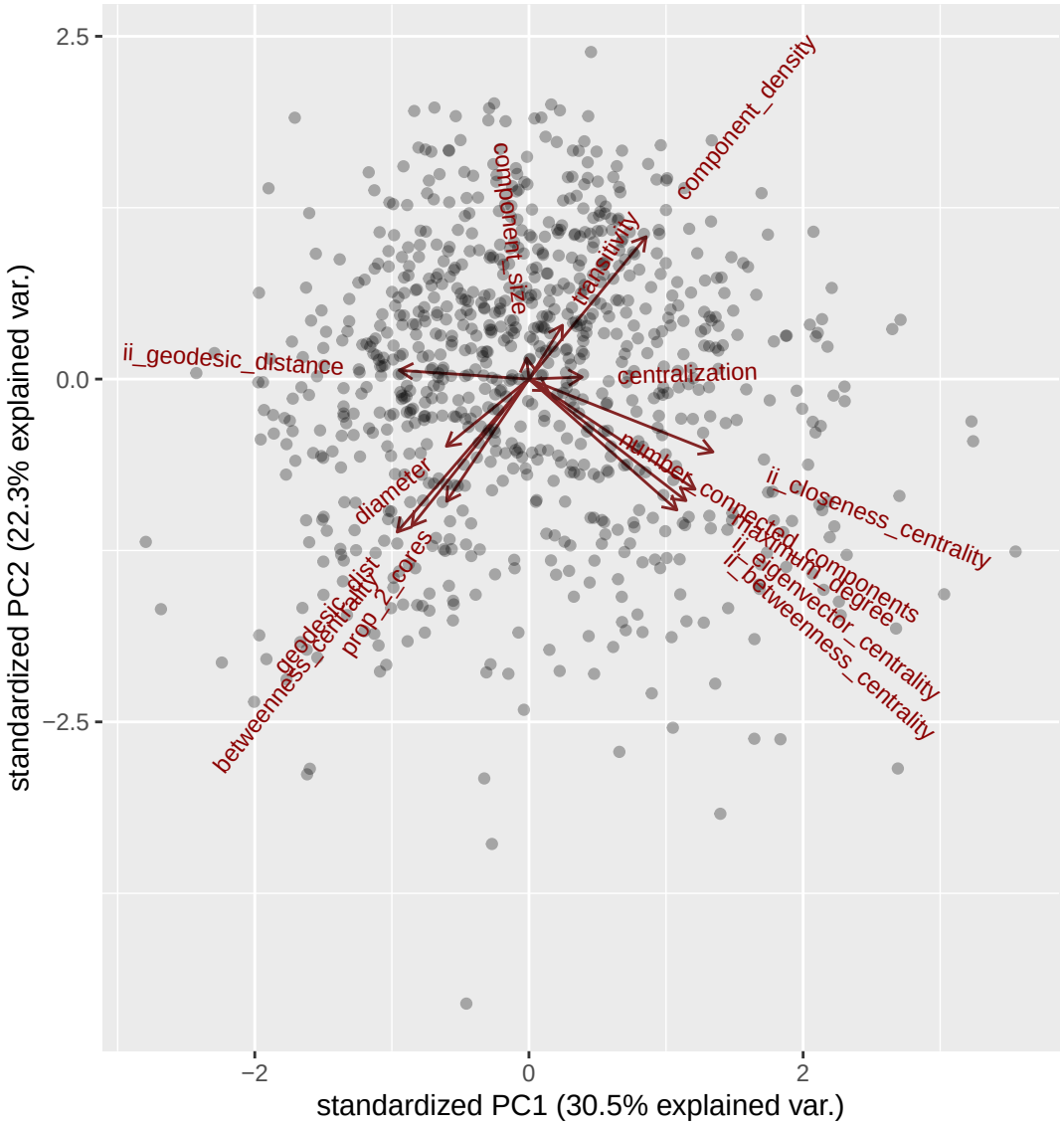
Lastly, we look at how each of our methods performed on the test set (the 150 simulations that we held out from the beginning). The ridge regression seems to do the best but only just barely. It has a slighly lower MSE and the highest R^2. However, the benefit is so marginal that I might prefer stepwise. Stepwise appears to be able to perform just as well with only three variables. All of the models perform quite poorly. See all values in Table 2.

Table 2: Cumulative Incidence Model Performance

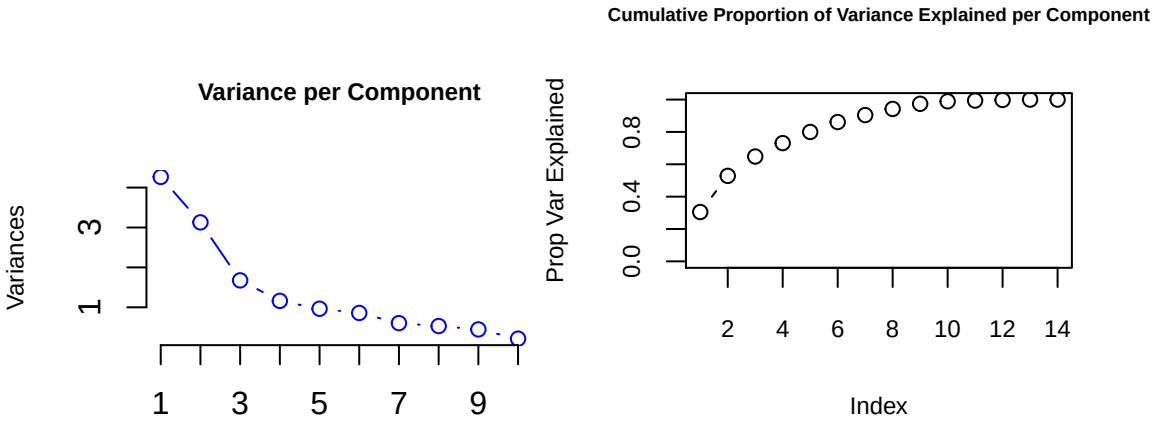
	No. of Coefficients	Lambda	R^2	MSE
Stepwise Regression	3	NA	0.1693498	0.8243094
Lasso Regression	5	0.0349838	0.1735235	0.8201676
Ridge Regression	14	0.2526115	0.1769443	0.8167728

Principal Component Analysis of Cumulative Infections

The performance of the stepwise, lasso, and ridge regressions are quite poor. Because the data set is wide and includes many variables that are correlated with one another, I thought it might be beneficial to employ principal component analysis (PCA). PCA is a transformation of a set of correlated variables over set of samples to linear combination of uncorrelated principal components over the same set of samples. Each principal component sums up a certain percentage of the total variation in the dataset. See below for a visual representation of how the variables contribute to the first two Principal Components constructed. The first principal component is represented on the x-axis and explains about a third of the total variance. The second principal component is represented on the y-axis and explains about a fourth of the total variance.



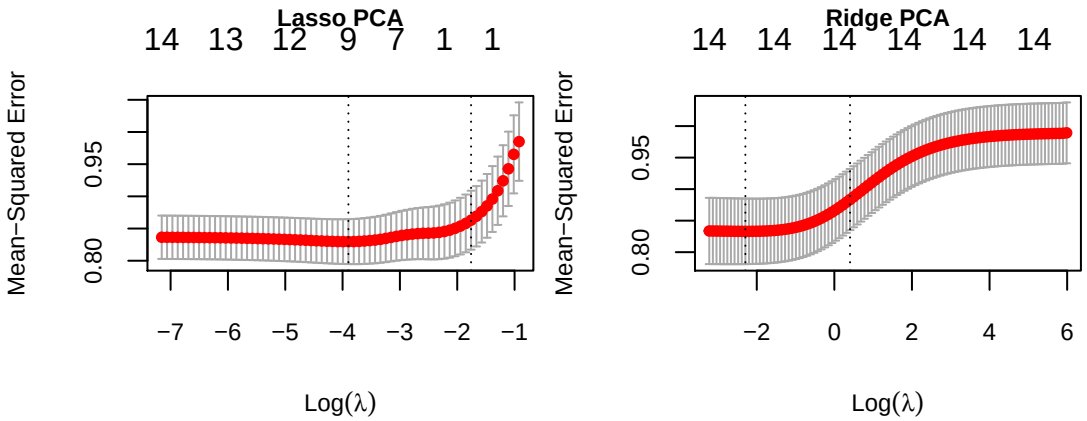
Below see how variance is represented by each component when I make no restrictions on the amount of principal components built and used. The first plot is the amount of variance represented by each component and the second is the cumulative proportion of the total variance explained as you add in principal components one by one.



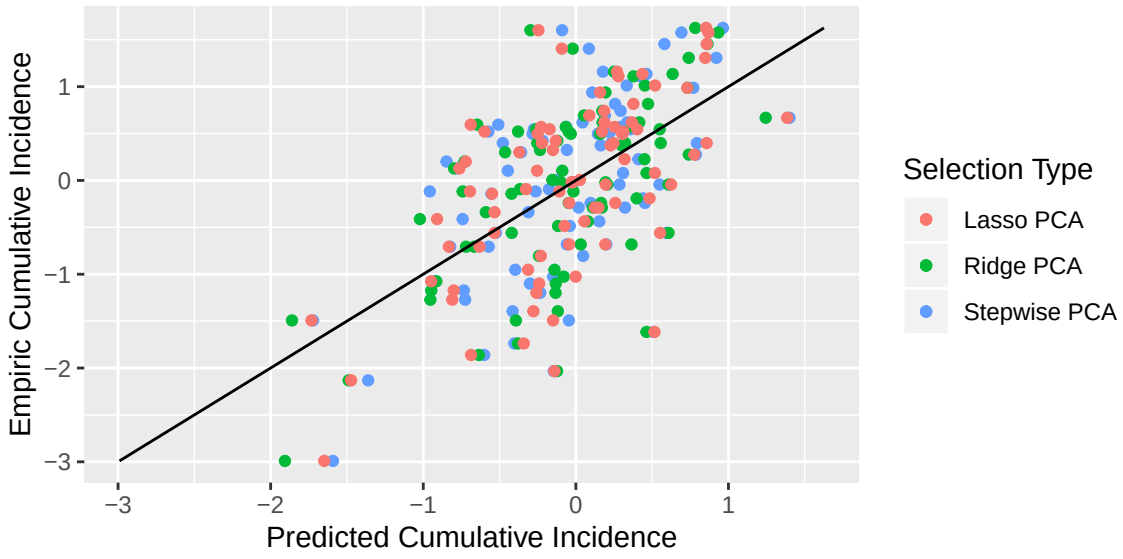
I store these principal components as my new options for explanatory variables. I used cross-validation to construct a predictive model with the principal components instead of the original variables through a) stepwise regression; b) lasso regression; c) ridge regression. I again use cross-validation to determine the best number of components to employ to address the cost-complexity trade-off. When using the principal components, stepwise chooses seven components and lasso chooses ten. Both of these are much larger than the amount of original variables chosen.

PCA Results

First, see below for the lambda plots that visualize the selection of the best lambda values for the lasso and ridge regressions on the principal components.



Below see a plot comparing the predicted number of HIV infections with the empiric for all three PCA models. Again, you can see from the plot that all models predict similarly (i.e. the predictions vary from the empiric values in the same fashion).



Using principal components appears to have improved our performance substantially. See Table 3. However, all of the models that use principal components notably elected to keep more of them. Additionally, using pricipal components severely restricts the ability to interpret our results. As it was quite important to me to understand which specific network structures might be the most influential, the models using PCA are less valuable. However, it is encouraging that I was able to improve the performance of the models and was a new skill that was interesting to learn!

Table 3: Cumulative Incidence Model Performance

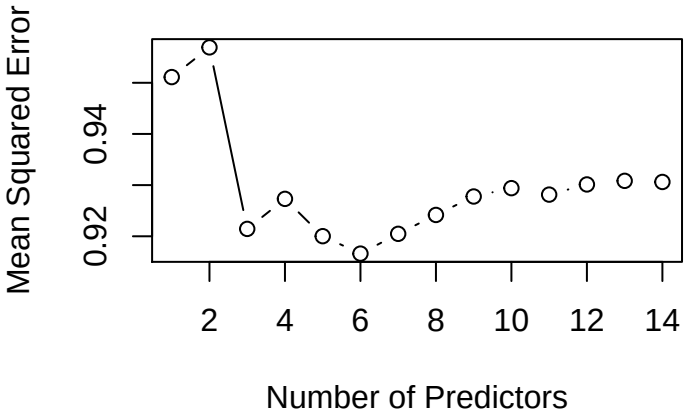
	No. of Coefficients	Lambda	R^2	MSE
Stepwise Regression	3	NA	0.1693498	0.8243094
Lasso Regression	5	0.0349838	0.1735235	0.8201676
Ridge Regression	14	0.2526115	0.1769443	0.8167728
Stepwise PCA Regression	7	NA	0.3490064	0.5810072
Lasso PCA Regression	10	0.0202759	0.3648388	0.5668770
Ridge PCA Regression	14	0.1009135	0.3799070	0.5534287

Variable Selection for Prediction of Time Elapsed Until Ten Infections

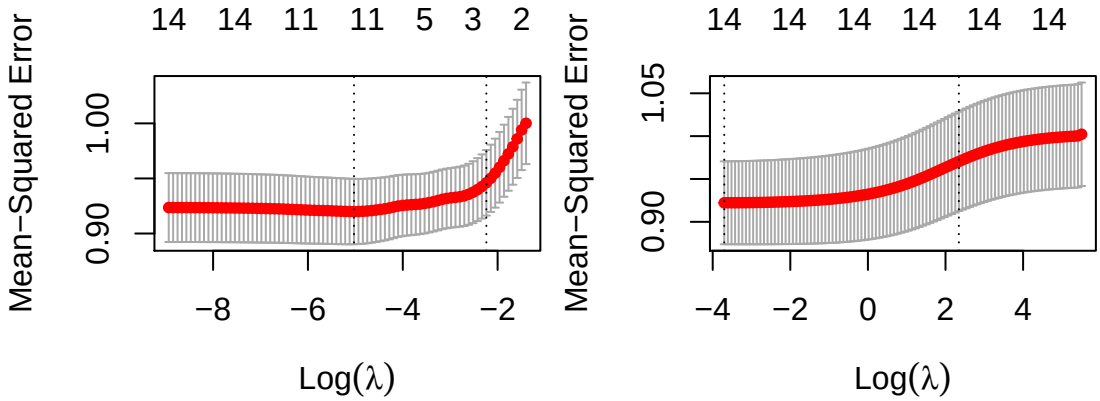
Next, I investigated our second outcome of interest: the number of months elapsed until ten incident infections. This is the number of HIV infections that incited the contact tracing investigation and the subsequent increase in testing activities. I measured this to investigate how network structure may be related to epidemic velocity. Specifically, this represents the notion of "how quickly the outbreak gets going". This is potentially of interest to public health officials given that the faster the outbreak begins the harader it is to implement prevention strategies to protect vulnerable individuals.

I used the same techniques outlined above for the analysis of the number cumulative infections. I will not re-outline all of the procedures here, but I will expand upon the results. The only change is the new outcome.

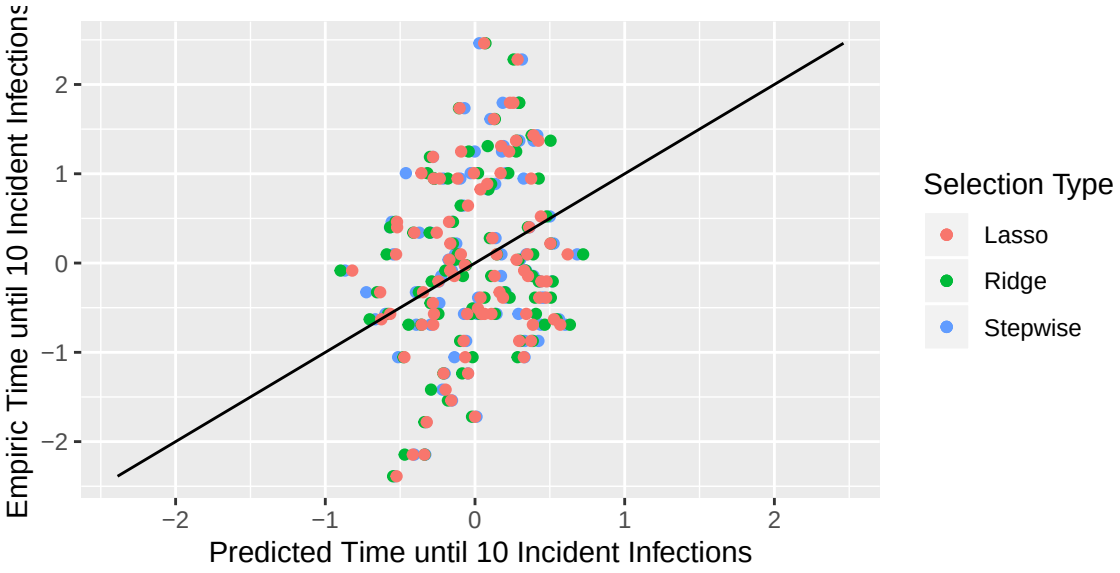
See below the plot that compares the number of predictors included against the Mean Squared Error (MSE) for stepwise variable selection for the number of months elapsed until ten infections. In this case, the optimal number of variables selected is six.



Below you can see the two plots that compares the log(lambda), a hyperparameter, against the Mean Squared Error (MSE) for the lasso and ridge regressions. We choose the lambda that minimizes the mean MSE across the k folds, both of which are reported Table 5.



After all three models have been trained and validated on the training set, I test them on the set of 150 values that were in our hold-out test set. Below see a plot comparing the predicted number of HIV infections with the empiric for all three models. The same as with the first outcome, you can that all models predict similarly (i.e. the predictions vary from the empiric values in the same fashion).



Model Performance Comparison

Now that we have constructed and employed each of our models, we can compare the results. Stepwise chooses six variables, Degree of the Initial Infection, Eigenvector Centrality of the Initial Infection, Main Component Size, Component Density, Proportion of 2-Cores, and Diameter of the Largest Component. Lasso chooses all six of these as well but adds four more (see Table 4). Each of the chosen variables have coefficients similar in magnitude and sign across the three methods. Not all of the coefficients' signs match my intuition for how they would affect the number of cumulative infections. One that does is the coefficient of the degree centrality of the initial infection. As the degree increases, the number of partners that initial individual can infect is large, and amount of time until ten infections goes down. This is matched with a negative coefficient across all three methods. On the other hand, with larger density we should see shorter time until ten infections. This is matched with a positive coefficient–opposite from what we might expect.

These results are very interesting and a substantial amount of time could be used to parse out what might be happening. As I am limited with my time, I am going to leave the analysis here for now, Please note that with more time I would have loved to go more in depth into the possible meaning behind the signs and sizes of these coefficients.

Lastly, we look at how each of our methods performed on the test set (the 150 simulations that we held out from the beginning). Similar to with our first outcome, the ridge regression model seems to do the best but only just barely over lasso. It has a slighly lower MSE and the highest R^2. Here the performance improvement seems large enough that I would choose either lasso or ridge over stepwise, even though they have more variables. All of the models perform very poorly. See all values in Table 5.

Table 4: Time Elapsed Coefficients Comparison

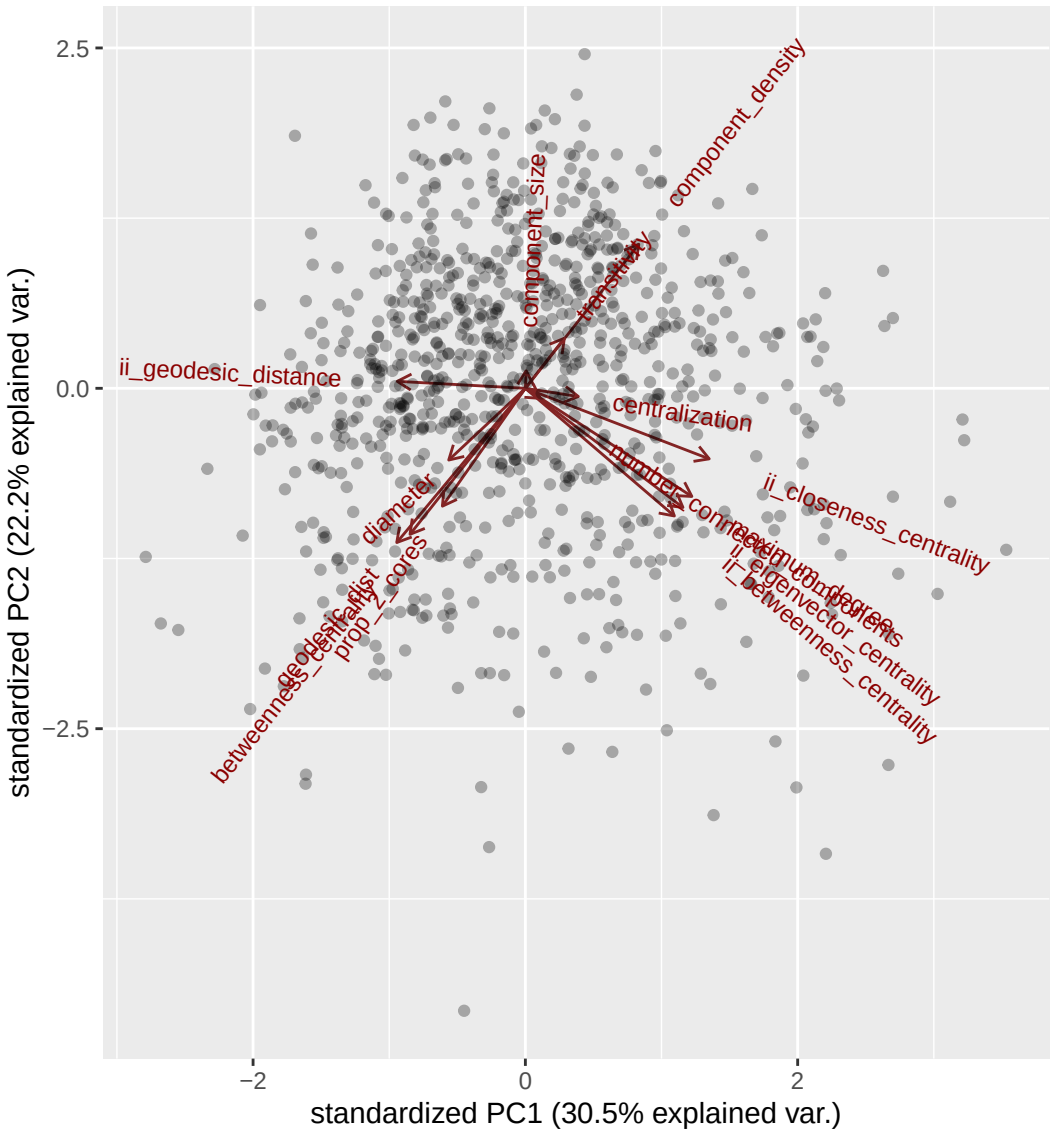
	Stepwise	Lasso	Ridge
Intercept	0.0000	0.0000	0.0000
Degree of the Initial Infection	-0.1475	-0.1026	-0.2229
Betweenness Centrality of the Largest Component	NA	0.0000	-0.0451
Betweenness Centrality of the Initial Infection	NA	0.0000	0.0861
Closeness Centrality of the Initial Infection	NA	0.0517	0.0561
Eigenvector Centrality of the Initial Infection	0.2569	0.1611	0.1900
Component Size of the Largest Component	0.1743	0.1593	0.1716
Number of Connected Components	NA	0.0133	0.0276
Density of the Largest Component	0.3583	0.3131	0.3693
Geodesic Distance of the Largest Component	NA	0.0000	0.0640
Geodesic Distance of the Initial Infection	NA	-0.0075	-0.0199
Centralization of the Largest Component	NA	0.0000	0.0004
Proportion of 2-Cores	0.0718	0.0514	0.0709
Transitivity	NA	0.0023	0.0041
Diameter of the Largest Component	0.0691	0.0621	0.0741

Table 5: Time Elapsed Model Performance

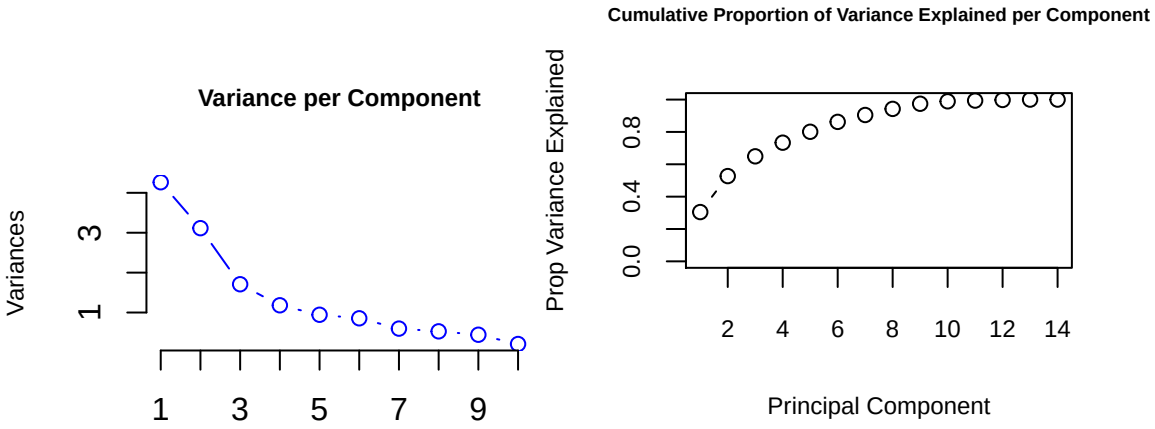
	No. of Coefficients	Lambda	R^2	MSE
Stepwise Regression	6	NA	0.0346460	0.9542580
Lasso Regression	10	0.0065492	0.0430486	0.9459519
Ridge Regression	14	0.0246571	0.0448203	0.9442006

Principal Component Analysis of Time Elapsed Until Ten Infections

Again, we employ PCA to see if we can improve performance, although we note that it will reduce the interpretability of our model. See below for a visualization of the two principal components that explain the most variance.

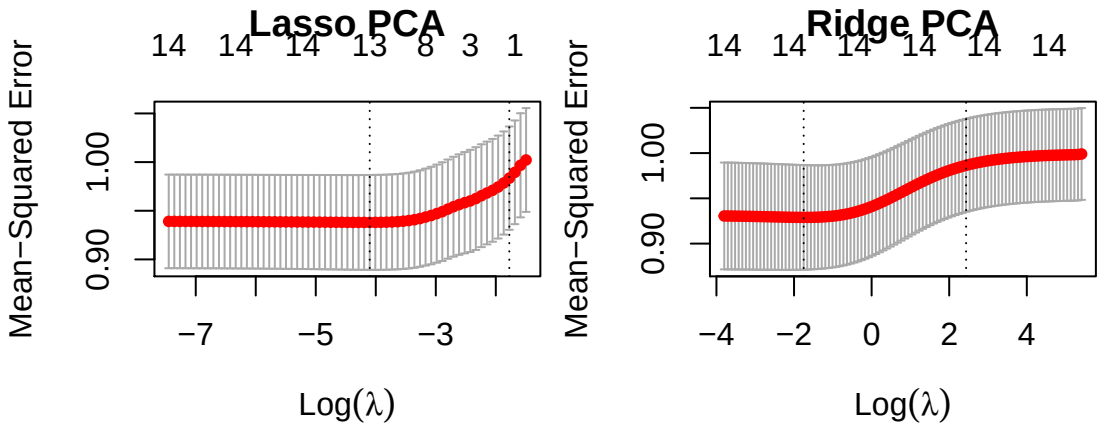


Below see how variance is represented by each component when I make no restrictions on the amount of principal components built and used. The first plot is the amount of variance represented by each component and the second is the cumulative proportion of the total variance explained as you add in principal components one by one.

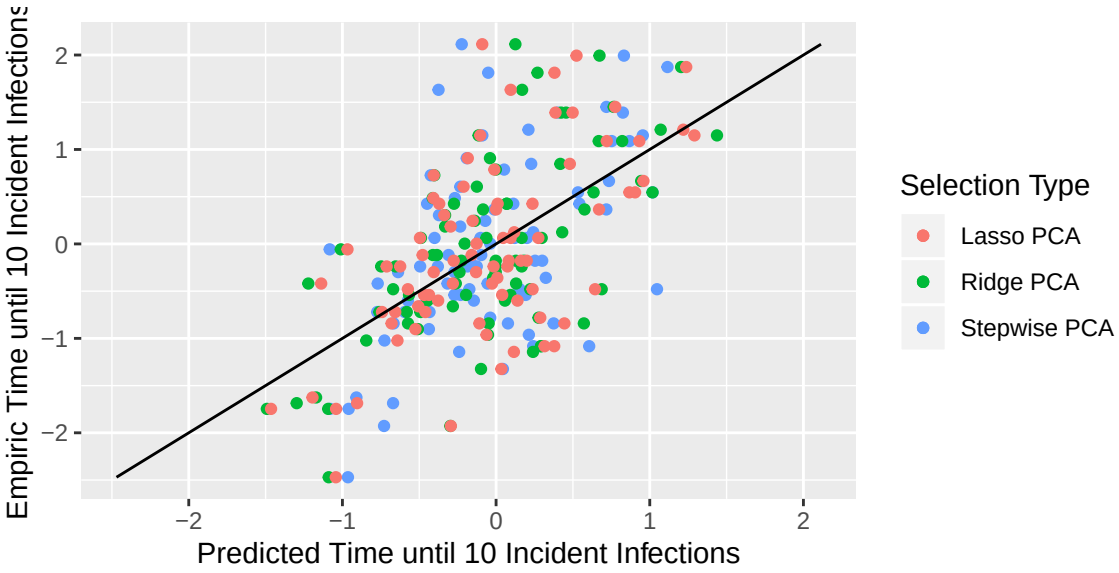


PCA Results

See below for the lambda plots that visualize the selection of the best lambda values for the lasso and ridge regressions on the principal components.



Below see a plot comparing the predicted number of months elapsed ntil ten HIV infections with the empiric value for all three PCA models. Again, you can see from the plot that all models predict similarly (i.e. the predictions vary from the empiric values in the same fashion). This is the only plot in which you can see a bit of separation between the three methods' predictions, although it is still minimal.



Model Performance Comparison

Again, the use of principal components appears to have improved our performance substantially, even more so than for the first outcome. See Table 6. Additionally, in this case, both stepwise models elect to keep the same number of variables. Lasso PCA chooses to keep two more than the original Lasso. The amount of variables retained differed more strongly between methods for our first outcome. As previously discussed, using pricipal components severely restricts the ability to interpret our results. However, it is again encouraging that I was able to improve the performance of the models.

Table 6: Time Elapsed Model Performance

	No. of Coefficients	Lambda	R^2	MSE
Stepwise Regression	6	NA	0.0346460	0.9542580
Lasso Regression	10	0.0065492	0.0430486	0.9459519
Ridge Regression	14	0.0246571	0.0448203	0.9442006
Stepwise PCA Regression	6	NA	0.3027960	0.6229371
Lasso PCA Regression	12	0.0165588	0.3627644	0.5693565
Ridge PCA Regression	14	0.1734721	0.3785470	0.5552551

Conclusion

This analysis has provided insight into which structural network components might be most predictive of HIV outbreak magnitude and velocity in a particular type of injection network. It allowed me to identify which out of a group of comparable measurements (such as those that measure the influence of the initial infection) might be the best predictors. Although the models performed quite poorly, the original goal of this analysis was to identify the most influential variables, which we were still able to do.

The poor performance of the models encouraged me to learn a technique that we had not employed in class: Principal Component Analysis. In the end, using this technique assisted in improving our models ability to predict the outbreak behavior. However, we were still unable to achieve any R-squared values above 0.5. There were a few limitations that likely contributed to the models' dismal performance. First, the assumption of non-linearity is likely quite hurtful. With more time, I would have loved to have used the skills we built in Problem Set 6. I predict that non-linear transformations would have helped improved our models. However, I elected to learn and employ PCA with my time in an effort to build a new skill. In the future, I would prioritize transformations as (1) now I know PCA and (2) they would lend to better interpretability. A second limitation is the limited amount of data. As this is all simulated data, with more time I would have run more simulations and had more data for training and testing.

In addition to these limitations, I have another theory of what is leading to the difficulties in prediction. I believe that in conditioning the network to have comparable density, size and degree distribution to the observed network, I too severly limited the amount of variability in the set of networks. I conditioned the networks in this way so that they would be representative to Scott County. As described at the start of this document, there is still a substantial amount of noise and randomness introduced into the set of networks. However, I believe that the structural conditions restrict the network in certain ways that make the models unable to make better predictions. The models are picking up all of the predictive information that they can, and what they are not picking up is really just random noise. I would even say that seeing this noise in the predictions is evidence that the models are doing a good job of not overfitting. In short, they are predicting what they can, and the other deviation from the predicted values is an accurate representation of the noise of the simulations. I would be interested to see what would happen if I released some of the restrictions on the networks. I would guess that some of the restrictions would be strong predictors themselves (a reason we chose to define them in the first place)!

I greatly enjoyed this project and this class. Thank you very much for your instruction and guidance this semester. Looking forward to the Spring!

Code Appendix

```
knitr::opts_chunk$set(echo = F, results='asis', warning=F, message=F, cache=T, fig.align="center", set.
# load libraries
```

```r
library(dplyr)
library(tidyr)
library(reshape2)
library(naniar)
library(grDevices)
library(ggplot2)
library(GGally)
library(MASS)
library(boot)
library(leaps)
library(corrplot)
library(glmnet)
library(neuralnet)
library(devtools)
library(ggbiplot)
library(caret)
library(standardize)
library(kableExtra)
library(knitr)
library(dplyr)
#load data
sc.set.original <- read.csv("~/Desktop/research/SC_inj_net_dynamics/scripts/A3_conditions_ten_percent.cs

#specify only structural exposure vars and cuminc outcome
sc.set.original[,1:6] <- NULL
sc.set <- sc.set.original[c(1:19,30)] #remove other outcomes, can investigate later
############################################
#correlation plot of all
# corrplot(cor(sc.set), method = "circle", type = "lower", tl.col = "black", tl.srt = 45, order = "hclu
#remove some with high correlation (avg degree, totalnodecount, totaledgecount, CCnodescount, CCedgesco
#make a seperate set with a few correlated variabls removed (remove totalnodecount and totaledgecount)
sc.set.limit <- sc.set[-c(1,10:13)]
corrplot(cor(sc.set.limit), method = "circle", type = "lower", tl.col = "black", tl.srt = 45, order = "
#corrplot.mixed(cor(sc.set.limit))
# Wrt CumInc
############################################
#pairs plots
#ggpairs(sc.set.limit,
#        diag=list(continuous="density"),
        #columns=c(sc.set.limit$cumulative_incidence),
#        axisLabels="show",
#        mapping = ggplot2::aes(shape = ".", alpha = 0.1),
#        progress=TRUE)
# not linear wrt cuminc: max degree,ii_betweenness_centrality,ii_eigenvector_centrality

ggplot(data=sc.set.limit) +
  geom_point(aes(x=sc.set.limit$component_density,y=sc.set.limit$cumulative_incidence), shape = ".") +
  labs(x="Density", y="Cumulative Incidence", size=6)

ggplot(data=sc.set.limit) +
  geom_point(aes(x=sc.set.limit$betweenness_centrality,y=sc.set.limit$cumulative_incidence), shape = "
  labs(x="Betweenness Centrality", y="Cumulative Incidence", size=6)
```

```
ggplot(data=sc.set.limit) +
  geom_point(aes(x=sc.set.limit$ii_eigenvector_centrality,y=sc.set.limit$cumulative_incidence), shape =
  labs(x="II Eigenvector Centrality", y="Cumulative Incidence", size=6)

ggplot(data=sc.set.limit) +
  geom_point(aes(x=sc.set.limit$centralization,y=sc.set.limit$cumulative_incidence), shape = ".") +
  labs(x="Centralization", y="Cumulative Incidence", size=6)

ggplot() +
  geom_point(aes(x=sc.set.limit$component_density,y=sc.set.original$ten_inf_ind), shape = ".") +
  labs(x="Density", y="Time until 10 Infections", size=6)

ggplot() +
  geom_point(aes(x=sc.set.limit$betweenness_centrality,y=sc.set.original$ten_inf_ind), shape = ".") +
  labs(x="Betweenness Centrality", y="Time until 10 Infections", size=6)

ggplot() +
  geom_point(aes(x=sc.set.limit$ii_eigenvector_centrality,y=sc.set.original$ten_inf_ind), shape = ".") +
  labs(x="II Eigenvector Centrality", y="Time until 10 Infections", size=6)

ggplot() +
  geom_point(aes(x=sc.set.limit$centralization,y=sc.set.original$ten_inf_ind), shape = ".") +
  labs(x="Centralization", y="Time until 10 Infections", size=6)
##### Prep Data for Regression Methods
sc.set.limit.indices <- sample(1:nrow(sc.set.limit), nrow(sc.set.limit)*(85/100), replace=F)
sc.set.limit <- sc.set.limit[sc.set.limit.indices,]
test.sc.set <- sc.set.limit[-sc.set.limit.indices,]

# standardize training set
sc.set.limit$maximum_degree <- scale(sc.set.limit$maximum_degree)
sc.set.limit$betweenness_centrality <- scale(sc.set.limit$betweenness_centrality)
sc.set.limit$ii_betweenness_centrality <- scale(sc.set.limit$ii_betweenness_centrality)
sc.set.limit$ii_closeness_centrality <- scale(sc.set.limit$ii_closeness_centrality)
sc.set.limit$ii_eigenvector_centrality <- scale(sc.set.limit$ii_eigenvector_centrality)
sc.set.limit$component_size <- scale(sc.set.limit$component_size)
sc.set.limit$number_connected_components <- scale(sc.set.limit$number_connected_components)
sc.set.limit$component_density <- scale(sc.set.limit$component_density)
sc.set.limit$geodesic_dist <- scale(sc.set.limit$geodesic_dist)
sc.set.limit$ii_geodesic_distance <- scale(sc.set.limit$ii_geodesic_distance)
sc.set.limit$centralization <- scale(sc.set.limit$centralization)
sc.set.limit$prop_2_cores <- scale(sc.set.limit$prop_2_cores)
sc.set.limit$transitivity <- scale(sc.set.limit$transitivity)
sc.set.limit$diameter <- scale(sc.set.limit$diameter)
sc.set.limit$cumulative_incidence <- scale(sc.set.limit$cumulative_incidence)

# standardize test set
test.sc.set$maximum_degree <- scale(test.sc.set$maximum_degree)
test.sc.set$betweenness_centrality <- scale(test.sc.set$betweenness_centrality)
test.sc.set$ii_betweenness_centrality <- scale(test.sc.set$ii_betweenness_centrality)
test.sc.set$ii_closeness_centrality <- scale(test.sc.set$ii_closeness_centrality)
test.sc.set$ii_eigenvector_centrality <- scale(test.sc.set$ii_eigenvector_centrality)
test.sc.set$component_size <- scale(test.sc.set$component_size)
test.sc.set$number_connected_components <- scale(test.sc.set$number_connected_components)
```

```r
test.sc.set$component_density <- scale(test.sc.set$component_density)
test.sc.set$geodesic_dist <- scale(test.sc.set$geodesic_dist)
test.sc.set$ii_geodesic_distance <- scale(test.sc.set$ii_geodesic_distance)
test.sc.set$centralization <- scale(test.sc.set$centralization)
test.sc.set$prop_2_cores <- scale(test.sc.set$prop_2_cores)
test.sc.set$transitivity <- scale(test.sc.set$transitivity)
test.sc.set$diameter <- scale(test.sc.set$diameter)
test.sc.set$cumulative_incidence <- scale(test.sc.set$cumulative_incidence)
# take predict.regsubsets function from variable selection code for predict function replacement
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}
#initializations
k.folds <- 10
variables.n <- dim(sc.set.limit)[2]-1
folds <- sample(1:k.folds,nrow(sc.set.limit),replace=TRUE)
cv.errors <- matrix(NA,k.folds,variables.n, dimnames=list(NULL, paste(1:variables.n)))
# find best fit using all the different fold options and no of variable options
for(j in 1:k.folds){
  best.fit=regsubsets(cumulative_incidence~.,data=sc.set.limit[folds!=j,],nvmax=variables.n)
  for(i in 1:variables.n){
    pred=predict.regsubsets(best.fit,sc.set.limit[folds==j,],id=i)
    cv.errors[j,i]=mean((sc.set.limit$cumulative_incidence[folds==j] - pred) ^2)
  }
}
#outputs
mean.cv.errors=apply(cv.errors,2,mean)
plot(mean.cv.errors,type='b', xlab= "Number of Predictors", ylab="Mean Squared Error")
hyper.param <- which.min(mean.cv.errors)
# use best no of variables and stepwise regress whole dataset
reg.best <- regsubsets(cumulative_incidence~.,data=sc.set.limit, nvmax=variables.n)
coef.of.best <- coef(reg.best,hyper.param)
#coef.of.best
no.coef.step <- length(coef.of.best)-1
#### ii_closeness_centrality, component size, component density
coef.of.second.best <- coef(reg.best,6)
#### ii_closeness_centrality, component size, component density AND ii_betweenness_centrality, ii_eigen
# predictions and performance
pred.step=predict.regsubsets(reg.best,test.sc.set,id=hyper.param)
mse.step <- mean((pred.step-test.sc.set$cumulative_incidence)^2) # 1351.092
# r2 on entire set
err.step <- pred.step - test.sc.set$cumulative_incidence
r2.step <- 1-var(err.step)/var(test.sc.set$cumulative_incidence)
r2.step <- r2.step[1]
ggplot() +
  geom_point(aes(x=pred.step,y=test.sc.set$cumulative_incidence, color="Stepwise")) +
  labs(y="Empiric Cumulative Incidence", x = "Predicted Cumulative Incidence", colour = "Selection Type
  geom_line(aes(x=test.sc.set$cumulative_incidence, y=test.sc.set$cumulative_incidence))
#initializations
```

```
grid=10^seq(10,-2,length=100)
x <- model.matrix(cumulative_incidence~.,data=sc.set.limit)[,-1]
y <- sc.set.limit$cumulative_incidence
# find best fit lambda
cv.out.lasso=cv.glmnet(x,y,alpha=1,nfolds = 10, type.measure="mse")
plot(cv.out.lasso)
bestlam.lasso=cv.out.lasso$lambda.min
# use bestlam and lasso entire data set
out.lasso=glmnet((model.matrix(cumulative_incidence~.,data=sc.set.limit)[,-1]),sc.set.limit$cumulative_
lasso.coef=predict(out.lasso,type="coefficients",s=bestlam.lasso)[0:variables.n+1,]
lasso.coef.non.zero <- lasso.coef[lasso.coef!=0]
no.coef.lasso <- length(lasso.coef.non.zero)-1
#### betweenness_centrality, ii_closeness_centrality, ii_eigenvector_centrality, component_density
# predictions and performance
pred.end.lasso=predict(out.lasso, s=bestlam.lasso, newx=(model.matrix(cumulative_incidence~.,data=test.
mse.lasso <- mean((pred.end.lasso-test.sc.set$cumulative_incidence)^2) # 1360.709
# r2 on entire set
err.lasso <- pred.end.lasso - test.sc.set$cumulative_incidence
r2.lasso <- 1-var(err.lasso)/var(test.sc.set$cumulative_incidence)
r2.lasso <- r2.lasso[1]
ggplot() +
  geom_point(aes(x=pred.step,y=test.sc.set$cumulative_incidence, color="Stepwise")) +
  geom_point(aes(x=pred.end.lasso,y=test.sc.set$cumulative_incidence, color="Lasso")) +
  labs(y="Empiric Cumulative Incidence", x = "Predicted Cumulative Incidence", colour = "Selection Type
  geom_line(aes(x=test.sc.set$cumulative_incidence, y=test.sc.set$cumulative_incidence))
#initializations
grid=10^seq(10,-2,length=100)
x <- model.matrix(cumulative_incidence~.,data=sc.set.limit)[,-1]
y <- sc.set.limit$cumulative_incidence
# find best fit lambda
cv.out.ridge=cv.glmnet(x,y,alpha=0,nfolds = 10, type.measure="mse")
plot(cv.out.ridge)
bestlam.ridge=cv.out.ridge$lambda.min
# use bestlam and ridge entire data set
out.ridge=glmnet((model.matrix(cumulative_incidence~.,data=sc.set.limit)[,-1]),sc.set.limit$cumulative_
ridge.coef=predict(out.ridge,type="coefficients",s=bestlam.ridge)[0:variables.n+1,]
no.coef.ridge <- length(ridge.coef)-1
#### betweenness_centrality, ii_closeness_centrality, ii_eigenvector_centrality, component_density
# predictions and performance
pred.end.ridge=predict(out.ridge, s=bestlam.ridge, newx=(model.matrix(cumulative_incidence~.,data=test.
mse.ridge <- mean((pred.end.ridge-test.sc.set$cumulative_incidence)^2) # 1360.709
# r2 on entire set
err.ridge <- pred.end.ridge - test.sc.set$cumulative_incidence
r2.ridge <- 1-var(err.ridge)/var(test.sc.set$cumulative_incidence)
r2.ridge <- r2.ridge[1]
ggplot() +
  geom_point(aes(x=pred.step,y=test.sc.set$cumulative_incidence, color="Stepwise")) +
  geom_point(aes(x=pred.end.ridge,y=test.sc.set$cumulative_incidence, color="Ridge")) +
  geom_point(aes(x=pred.end.lasso,y=test.sc.set$cumulative_incidence, color="Lasso")) +
  labs(y="Empiric Cumulative Incidence", x = "Predicted Cumulative Incidence", colour = "Selection Type
  geom_line(aes(x=test.sc.set$cumulative_incidence, y=test.sc.set$cumulative_incidence))
coefficients.comparison.df <- data.frame(
  c(coef.of.best[1], NA, NA, NA, coef.of.best[2], NA, coef.of.best[3], NA, coef.of.best[4], NA, NA, NA,
```

```
  c(lasso.coef[1], lasso.coef[2], lasso.coef[3], lasso.coef[4], lasso.coef[5], lasso.coef[6], lasso.coe
  c(ridge.coef[1], ridge.coef[2], ridge.coef[3], ridge.coef[4], ridge.coef[5], ridge.coef[6], ridge.coe

coefficients.comparison.df <- round(coefficients.comparison.df, 4)

colnames(coefficients.comparison.df) <- c("Stepwise",
                                          "Lasso",
                                          "Ridge")
rownames(coefficients.comparison.df) <- c("Intercept",
                                          "Degree of the Initial Infection",
                                          "Betweenness Centrality of the Largest Component",
                                          "Betweenness Centrality of the Initial Infection",
                                          "Closeness Centrality of the Initial Infection",
                                          "Eigenvector Centrality of the Initial Infection",
                                          "Component Size of the Largest Component",
                                          "Number of Connected Components",

                                          "Density of the Largest Component",
                                          "Geodesic Distance  of the Largest Component",
                                          "Geodesic Distance  of the Initial Infection",
                                          "Centralization of the Largest Component",
                                          "Proportion of 2-Cores",
                                          "Transitivity",
                                          "Diameter of the Largest Component")

kable(coefficients.comparison.df, "latex", caption = "Cumulative Infections Coefficients Comparison", bo
kable_styling(latex_options = c("striped", "hold_position"))
# build data frame to show performance (MSE) of each type of function on each variable
performance.df.1 <- data.frame(c(no.coef.step, no.coef.lasso, no.coef.ridge),
                               c(NA, bestlam.lasso, bestlam.ridge),
                               c(r2.step, r2.lasso, r2.ridge),
                               c(mse.step, mse.lasso, mse.ridge))
rownames(performance.df.1) <- c("Stepwise Regression", "Lasso Regression", "Ridge Regression")
colnames(performance.df.1) <- c("No. of Coefficients", "Lambda", "R^2", "MSE")

kable(performance.df.1, "latex", caption = "Cumulative Incidence Model Performance", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
sc.pca <- prcomp(sc.set.limit[,1:14], center=T,scale.=T)
#summary(sc.pca)
ggbiplot(sc.pca, alpha=0.3, varname.size=3, varname.color="black")
#plots to determine how many to use
screeplot(sc.pca, type="lines",col="blue", main= "Variance per Component",
          cex.main=0.75, cex.lab=0.75, cex.axis=0.75)
var <- sc.pca$sdev^2
propvar <- var/sum(var)
#plot(propvar, xlab = "Principal Component", ylab = "Proportion of Variance Explained",ylim = c(0,1), t
plot(cumsum(propvar),  ylab = "Prop Var Explained", main= "Cumulative Proportion of Variance Explained

PC.sc <- as.data.frame(cbind(sc.pca$x, sc.set.limit[,15]))
colnames(PC.sc)[15] <- "cumulative_incidence"


PC.sc.set.limit.indices <- sample(1:nrow(PC.sc), nrow(PC.sc)*(9/10), replace=F)
```

```
PC.sc <- PC.sc[PC.sc.set.limit.indices,]
PC.test.sc.set <- PC.sc[-PC.sc.set.limit.indices,]
########## use less based on those picked out of variable selection techniques
sc.pca.small <- prcomp(sc.set.limit[,c(2,4,6,8,11)], center=T,scale.=T)
#summary(sc.pca.small)
ggbiplot(sc.pca.small, alpha=0.3, varname.size=5, varname.color="black")
#plots to determine how many to use
screeplot(sc.pca.small, type="lines",col="blue")
var.small <- sc.pca.small$sdev^2
propvar.small <- var.small/sum(var.small)
plot(propvar.small, xlab = "Principal Component", ylab = "Proportion of Variance Explained",ylim = c(0,
plot(cumsum(propvar.small), xlab = "Principal Component", ylab = "Cumulative Proportion of Variance Expl
     ylim = c(0,1), type = "b")
ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 5)
lmFit_Step_PC <- train(cumulative_incidence ~ ., data = PC.sc, "lmStepAIC", scope =
                         list(lower = cumulative_incidence~1, upper = cumulative_incidence~.), directio

mod.step.pca = lm(cumulative_incidence ~ PC1 + PC2 + PC3 + PC8 + PC9 + PC10 + PC13, data = PC.test.sc.s
no.coef.step.pca <- dim(summary(mod.step.pca)$coefficients)[1]-1
step.rsq.pca <- summary(mod.step.pca)$r.squared
pred.step.pca <- predict(mod.step.pca,data = PC.test.sc.set)
mse.step.pca <- mean((pred.step.pca-PC.test.sc.set$cumulative_incidence)^2)
ggplot() +
  geom_point(aes(x=pred.step.pca,y=PC.test.sc.set$cumulative_incidence, color="Stepwise PCA")) +
  labs(y="Empiric Cumulative Incidence", x = "Predicted Cumulative Incidence", colour = "Selection Type
  geom_line(aes(x=PC.test.sc.set$cumulative_incidence, y=PC.test.sc.set$cumulative_incidence))
XP=data.matrix(PC.sc[,-15])
YP=data.matrix(PC.sc$cumulative_incidence)
lasso_PC=cv.glmnet(x=as.matrix(PC.sc[,-15]),y=as.matrix(PC.sc$cumulative_incidence),alpha=1,
                   nfolds = 10,type.measure="mse",family="gaussian")
plot(lasso_PC, main = "Lasso PCA", cex.main=0.75, cex.lab=0.75, cex.axis=0.75)
lambda.best.lasso.pca<- lasso_PC$lambda.min
#coef(lasso_PC, s=lasso_PC$lambda.min)


mod_lasso_PC = lm(cumulative_incidence ~PC1+PC2+PC3+PC4+PC5+PC7+PC8+PC9+PC10+PC13, data = PC.test.sc.se
no.coef.lasso.pca <- dim(summary(mod_lasso_PC)$coefficients)[1]-1
lasso.rsq.pca <- summary(mod_lasso_PC)$r.squared
pred.lasso.pca <- predict(mod_lasso_PC,data = PC.test.sc.set)
mse.lasso.pca <- mean((pred.lasso.pca-PC.test.sc.set$cumulative_incidence)^2)

## Ridge with PCA
XP=data.matrix(PC.sc[,-15])
YP=data.matrix(PC.sc$cumulative_incidence)
ridge_PC=cv.glmnet(x=as.matrix(PC.sc[,-15]),y=as.matrix(PC.sc$cumulative_incidence),alpha=0,
                   nfolds = 10,type.measure="mse",family="gaussian")
plot(ridge_PC, main = "Ridge PCA", cex.main=0.75, cex.lab=0.75, cex.axis=0.75)
lambda.best.ridge.pca<- ridge_PC$lambda.min
#coef(ridge_PC, s=ridge_PC$lambda.min)

mod_ridge_PC = lm(cumulative_incidence ~PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8+PC9+PC10+PC11+PC12+PC13+PC14, da
no.coef.ridge.pca <- dim(summary(mod_ridge_PC)$coefficients)[1]-1
ridge.rsq.pca <- summary(mod_ridge_PC)$r.squared
```

```r
pred.ridge.pca <- predict(mod_ridge_PC,data = PC.test.sc.set)
mse.ridge.pca <- mean((pred.ridge.pca-PC.test.sc.set$cumulative_incidence)^2)
ggplot() +
  geom_point(aes(x=pred.step.pca,y=PC.test.sc.set$cumulative_incidence, color="Stepwise PCA")) +
  geom_point(aes(x=pred.lasso.pca,y=PC.test.sc.set$cumulative_incidence, color="Lasso PCA")) +
  labs(y="Empiric Cumulative Incidence", x = "Predicted Cumulative Incidence", colour = "Selection Type
  geom_line(aes(x=PC.test.sc.set$cumulative_incidence, y=PC.test.sc.set$cumulative_incidence))
ggplot() +
  geom_point(aes(x=pred.step.pca,y=PC.test.sc.set$cumulative_incidence, color="Stepwise PCA")) +
  geom_point(aes(x=pred.ridge.pca,y=PC.test.sc.set$cumulative_incidence, color="Ridge PCA")) +
  geom_point(aes(x=pred.lasso.pca,y=PC.test.sc.set$cumulative_incidence, color="Lasso PCA")) +
  labs(y="Empiric Cumulative Incidence", x = "Predicted Cumulative Incidence", colour = "Selection Type
  geom_line(aes(x=PC.test.sc.set$cumulative_incidence, y=PC.test.sc.set$cumulative_incidence))
# build data frame to show performance (MSE) of each type of function on each variable

performance.df.2 <- data.frame(c(no.coef.step, no.coef.lasso, no.coef.ridge, no.coef.step.pca, no.coef.
                               c(NA, bestlam.lasso, bestlam.ridge, NA, lambda.best.lasso.pca, lambda.best
                               c(r2.step, r2.lasso, r2.ridge, step.rsq.pca, lasso.rsq.pca, ridge.rsq.pca)
                               c(mse.step, mse.lasso, mse.ridge, mse.step.pca, mse.lasso.pca, mse.ridge.p
rownames(performance.df.2) <- c("Stepwise Regression", "Lasso Regression", "Ridge Regression", "Stepwis
colnames(performance.df.2) <- c("No. of Coefficients", "Lambda", "R^2", "MSE")

kable(performance.df.2, "latex", caption = "Cumulative Incidence Model Performance", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
sc.set <- read.csv("~/Desktop/research/SC_inj_net_dynamics/scripts/A3_conditions_ten_percent.csv")

#specify only structural exposure vars and cuminc outcome
sc.set[,1:6] <- NULL
sc.set.limit <- sc.set[c(2:9,14:19,36)] #remove other outcomes, can investigate later
sc.set.limit$cumulative_incidence <- sc.set.limit$year_max
sc.set.limit$year_max <- NULL

sc.set.limit.indices <- sample(1:nrow(sc.set.limit), nrow(sc.set.limit)*(85/100), replace=F)
sc.set.limit <- sc.set.limit[sc.set.limit.indices,]
test.sc.set <- sc.set.limit[-sc.set.limit.indices,]

# standardize training set
sc.set.limit$maximum_degree <- scale(sc.set.limit$maximum_degree)
sc.set.limit$betweenness_centrality <- scale(sc.set.limit$betweenness_centrality)
sc.set.limit$ii_betweenness_centrality <- scale(sc.set.limit$ii_betweenness_centrality)
sc.set.limit$ii_closeness_centrality <- scale(sc.set.limit$ii_closeness_centrality)
sc.set.limit$ii_eigenvector_centrality <- scale(sc.set.limit$ii_eigenvector_centrality)
sc.set.limit$component_size <- scale(sc.set.limit$component_size)
sc.set.limit$number_connected_components <- scale(sc.set.limit$number_connected_components)
sc.set.limit$component_density <- scale(sc.set.limit$component_density)
sc.set.limit$geodesic_dist <- scale(sc.set.limit$geodesic_dist)
sc.set.limit$ii_geodesic_distance <- scale(sc.set.limit$ii_geodesic_distance)
sc.set.limit$centralization <- scale(sc.set.limit$centralization)
sc.set.limit$prop_2_cores <- scale(sc.set.limit$prop_2_cores)
sc.set.limit$transitivity <- scale(sc.set.limit$transitivity)
sc.set.limit$diameter <- scale(sc.set.limit$diameter)
sc.set.limit$cumulative_incidence <- scale(sc.set.limit$cumulative_incidence)
```

```r
# standardize test set
test.sc.set$maximum_degree <- scale(test.sc.set$maximum_degree)
test.sc.set$betweenness_centrality <- scale(test.sc.set$betweenness_centrality)
test.sc.set$ii_betweenness_centrality <- scale(test.sc.set$ii_betweenness_centrality)
test.sc.set$ii_closeness_centrality <- scale(test.sc.set$ii_closeness_centrality)
test.sc.set$ii_eigenvector_centrality <- scale(test.sc.set$ii_eigenvector_centrality)
test.sc.set$component_size <- scale(test.sc.set$component_size)
test.sc.set$number_connected_components <- scale(test.sc.set$number_connected_components)
test.sc.set$component_density <- scale(test.sc.set$component_density)
test.sc.set$geodesic_dist <- scale(test.sc.set$geodesic_dist)
test.sc.set$ii_geodesic_distance <- scale(test.sc.set$ii_geodesic_distance)
test.sc.set$centralization <- scale(test.sc.set$centralization)
test.sc.set$prop_2_cores <- scale(test.sc.set$prop_2_cores)
test.sc.set$transitivity <- scale(test.sc.set$transitivity)
test.sc.set$diameter <- scale(test.sc.set$diameter)
test.sc.set$cumulative_incidence <- scale(test.sc.set$cumulative_incidence)
# take predict.regsubsets function from variable selection code for predict function replacement
predict.regsubsets=function(object,newdata,id,...){
  form=as.formula(object$call[[2]])
  mat=model.matrix(form,newdata)
  coefi=coef(object,id=id)
  xvars=names(coefi)
  mat[,xvars]%*%coefi
}
#initializations
k.folds <- 10
variables.n <- dim(sc.set.limit)[2]-1
folds <- sample(1:k.folds,nrow(sc.set.limit),replace=TRUE)
cv.errors <- matrix(NA,k.folds,variables.n, dimnames=list(NULL, paste(1:variables.n)))
# find best fit using all the different fold options and no of variable options
for(j in 1:k.folds){
  best.fit=regsubsets(cumulative_incidence~.,data=sc.set.limit[folds!=j,],nvmax=variables.n)
  for(i in 1:variables.n){
    pred=predict.regsubsets(best.fit,sc.set.limit[folds==j,],id=i)
    cv.errors[j,i]=mean((sc.set.limit$cumulative_incidence[folds==j] - pred) ^2)
  }
}
#outputs
mean.cv.errors=apply(cv.errors,2,mean)
plot(mean.cv.errors,type='b', xlab= "Number of Predictors", ylab="Mean Squared Error")
hyper.param <- which.min(mean.cv.errors)
# use best no of variables and stepwise regress whole dataset
reg.best <- regsubsets(cumulative_incidence~.,data=sc.set.limit, nvmax=variables.n)
coef.of.best <- coef(reg.best,hyper.param)
#coef.of.best
no.coef.step <- length(coef.of.best)-1
#### ii_closeness_centrality, component size, component density
coef.of.second.best <- coef(reg.best,6)
#### ii_closeness_centrality, component size, component density AND ii_betweenness_centrality, ii_eigen
# predictions and performance
pred.step=predict.regsubsets(reg.best,test.sc.set,id=hyper.param)
mse.step <- mean((pred.step-test.sc.set$cumulative_incidence)^2) # 1351.092
# r2 on entire set
```

```
err.step <- pred.step - test.sc.set$cumulative_incidence
r2.step <- 1-var(err.step)/var(test.sc.set$cumulative_incidence)
r2.step <- r2.step[1]
ggplot() +
  geom_point(aes(x=pred.step,y=test.sc.set$cumulative_incidence, color="Stepwise")) +
  labs(y="Empiric Time until 10 Incident Infections", x = "Predicted Time until 10 Incident Infections"
  geom_line(aes(x=test.sc.set$cumulative_incidence, y=test.sc.set$cumulative_incidence))
#initializations
grid=10^seq(10,-2,length=100)
x <- model.matrix(cumulative_incidence~.,data=sc.set.limit)[,-1]
y <- sc.set.limit$cumulative_incidence
# find best fit lambda
cv.out.lasso=cv.glmnet(x,y,alpha=1,nfolds = 10, type.measure="mse")
plot(cv.out.lasso)
bestlam.lasso=cv.out.lasso$lambda.min
# use bestlam and lasso entire data set
out.lasso=glmnet((model.matrix(cumulative_incidence~.,data=sc.set.limit)[,-1]),sc.set.limit$cumulative_
lasso.coef=predict(out.lasso,type="coefficients",s=bestlam.lasso)[0:variables.n+1,]
lasso.coef.non.zero <- lasso.coef[lasso.coef!=0]
no.coef.lasso <- length(lasso.coef.non.zero)-1
#### betweenness_centrality, ii_closeness_centrality, ii_eigenvector_centrality, component_density
# predictions and performance
pred.end.lasso=predict(out.lasso, s=bestlam.lasso, newx=(model.matrix(cumulative_incidence~.,data=test.s
mse.lasso <- mean((pred.end.lasso-test.sc.set$cumulative_incidence)^2) # 1360.709
# r2 on entire set
err.lasso <- pred.end.lasso - test.sc.set$cumulative_incidence
r2.lasso <- 1-var(err.lasso)/var(test.sc.set$cumulative_incidence)
r2.lasso <- r2.lasso[1]


#initializations
grid=10^seq(10,-2,length=100)
x <- model.matrix(cumulative_incidence~.,data=sc.set.limit)[,-1]
y <- sc.set.limit$cumulative_incidence
# find best fit lambda
cv.out.ridge=cv.glmnet(x,y,alpha=0,nfolds = 10, type.measure="mse")
plot(cv.out.ridge)
bestlam.ridge=cv.out.ridge$lambda.min
# use bestlam and ridge entire data set
out.ridge=glmnet((model.matrix(cumulative_incidence~.,data=sc.set.limit)[,-1]),sc.set.limit$cumulative_
ridge.coef=predict(out.ridge,type="coefficients",s=bestlam.ridge)[0:variables.n+1,]
no.coef.ridge <- length(ridge.coef)-1
#### betweenness_centrality, ii_closeness_centrality, ii_eigenvector_centrality, component_density
# predictions and performance
pred.end.ridge=predict(out.ridge, s=bestlam.ridge, newx=(model.matrix(cumulative_incidence~.,data=test.s
mse.ridge <- mean((pred.end.ridge-test.sc.set$cumulative_incidence)^2) # 1360.709
# r2 on entire set
err.ridge <- pred.end.ridge - test.sc.set$cumulative_incidence
r2.ridge <- 1-var(err.ridge)/var(test.sc.set$cumulative_incidence)
r2.ridge <- r2.ridge[1]
ggplot() +
  geom_point(aes(x=pred.step,y=test.sc.set$cumulative_incidence, color="Stepwise")) +
  geom_point(aes(x=pred.end.lasso,y=test.sc.set$cumulative_incidence, color="Lasso")) +
```

```
  labs(y="Empiric Time until 10 Incident Infections", x = "Predicted Time until 10 Incident Infections"
  geom_line(aes(x=test.sc.set$cumulative_incidence, y=test.sc.set$cumulative_incidence))
ggplot() +
  geom_point(aes(x=pred.step,y=test.sc.set$cumulative_incidence, color="Stepwise")) +
  geom_point(aes(x=pred.end.ridge,y=test.sc.set$cumulative_incidence, color="Ridge")) +
  geom_point(aes(x=pred.end.lasso,y=test.sc.set$cumulative_incidence, color="Lasso")) +
  labs(y="Empiric Time until 10 Incident Infections", x = "Predicted Time until 10 Incident Infections"
  geom_line(aes(x=test.sc.set$cumulative_incidence, y=test.sc.set$cumulative_incidence))
coefficients.comparison.df <- data.frame(
  c(coef.of.best[1], coef.of.best[2], NA, NA, NA, coef.of.best[3], coef.of.best[4], NA, coef.of.best[5]
  c(lasso.coef[1], lasso.coef[2], lasso.coef[3], lasso.coef[4], lasso.coef[5], lasso.coef[6], lasso.coe
  c(ridge.coef[1], ridge.coef[2], ridge.coef[3], ridge.coef[4], ridge.coef[5], ridge.coef[6], ridge.coe

coefficients.comparison.df <- round(coefficients.comparison.df, 4)

colnames(coefficients.comparison.df) <- c("Stepwise",
                                          "Lasso",
                                          "Ridge")
rownames(coefficients.comparison.df) <- c("Intercept",
                                          "Degree of the Initial Infection",
                                          "Betweenness Centrality of the Largest Component",
                                          "Betweenness Centrality of the Initial Infection",
                                          "Closeness Centrality of the Initial Infection",
                                          "Eigenvector Centrality of the Initial Infection",
                                          "Component Size of the Largest Component",
                                          "Number of Connected Components",

                                          "Density of the Largest Component",
                                          "Geodesic Distance  of the Largest Component",
                                          "Geodesic Distance  of the Initial Infection",
                                          "Centralization of the Largest Component",
                                          "Proportion of 2-Cores",
                                          "Transitivity",
                                          "Diameter of the Largest Component")

kable(coefficients.comparison.df, "latex", caption = "Time Elapsed Coefficients Comparison", booktabs =
kable_styling(latex_options = c("striped", "hold_position"))
# build data frame to show performance (MSE) of each type of function on each variable
performance.df.1 <- data.frame(c(no.coef.step, no.coef.lasso, no.coef.ridge),
                               c(NA, bestlam.lasso, bestlam.ridge),
                               c(r2.step, r2.lasso, r2.ridge),
                               c(mse.step, mse.lasso, mse.ridge))
rownames(performance.df.1) <- c("Stepwise Regression", "Lasso Regression", "Ridge Regression")
colnames(performance.df.1) <- c("No. of Coefficients", "Lambda", "R^2", "MSE")

kable(performance.df.1, "latex", caption = "Time Elapsed Model Performance", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
sc.pca <- prcomp(sc.set.limit[,1:14], center=T,scale.=T)
#summary(sc.pca)
ggbiplot(sc.pca, alpha=0.3, varname.size=3, varname.color="black")
#plots to determine how many to use
screeplot(sc.pca, type="lines",col="blue", cex.main=0.75, cex.lab=0.75, cex.axis=0.75, main = "Variance
var <- sc.pca$sdev^2
```

```r
propvar <- var/sum(var)
#plot(propvar, xlab = "Principal Component", ylab = "Proportion of Variance Explained",ylim = c(0,1), t
plot(cumsum(propvar), xlab = "Principal Component", ylab = "Prop Variance Explained", main= "Cumulative

PC.sc <- as.data.frame(cbind(sc.pca$x, sc.set.limit[,15]))
colnames(PC.sc)[15] <- "cumulative_incidence"


PC.sc.set.limit.indices <- sample(1:nrow(PC.sc), nrow(PC.sc)*(9/10), replace=F)
PC.sc <- PC.sc[PC.sc.set.limit.indices,]
PC.test.sc.set <- PC.sc[-PC.sc.set.limit.indices,]
ctrl <- trainControl(method = "repeatedcv", number = 5, repeats = 5)
lmFit_Step_PC <- train(cumulative_incidence ~ ., data = PC.sc, "lmStepAIC", scope =
                         list(lower = cumulative_incidence~1, upper = cumulative_incidence~.), directio

mod.step.pca = lm(cumulative_incidence ~ PC1 + PC2 + PC8 + PC9 + PC10 + PC13, data = PC.test.sc.set)
no.coef.step.pca <- dim(summary(mod.step.pca)$coefficients)[1]-1
step.rsq.pca <- summary(mod.step.pca)$r.squared
pred.step.pca <- predict(mod.step.pca,data = PC.test.sc.set)
mse.step.pca <- mean((pred.step.pca-PC.test.sc.set$cumulative_incidence)^2)
ggplot() +
  geom_point(aes(x=pred.step.pca,y=PC.test.sc.set$cumulative_incidence, color="Stepwise PCA")) +
  labs(y="Empiric Time until 10 Incident Infections", x = "Predicted Time until 10 Incident Infections"
  geom_line(aes(x=PC.test.sc.set$cumulative_incidence, y=PC.test.sc.set$cumulative_incidence))
XP=data.matrix(PC.sc[,-15])
YP=data.matrix(PC.sc$cumulative_incidence)
lasso_PC=cv.glmnet(x=as.matrix(PC.sc[,-15]),y=as.matrix(PC.sc$cumulative_incidence),alpha=1,
                   nfolds = 10,type.measure="mse",family="gaussian")
plot(lasso_PC, main= "Lasso PCA")
lambda.best.lasso.pca<- lasso_PC$lambda.min
#coef(lasso_PC, s=lasso_PC$lambda.min)

mod_lasso_PC = lm(cumulative_incidence ~PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8+PC9+PC10+PC13+PC14, data = PC.t
no.coef.lasso.pca <- dim(summary(mod_lasso_PC)$coefficients)[1]-1
lasso.rsq.pca <- summary(mod_lasso_PC)$r.squared
pred.lasso.pca <- predict(mod_lasso_PC,data = PC.test.sc.set)
mse.lasso.pca <- mean((pred.lasso.pca-PC.test.sc.set$cumulative_incidence)^2)

## Ridge with PCA TIME
XP=data.matrix(PC.sc[,-15])
YP=data.matrix(PC.sc$cumulative_incidence)
ridge_PC=cv.glmnet(x=as.matrix(PC.sc[,-15]),y=as.matrix(PC.sc$cumulative_incidence),alpha=0,
                   nfolds = 10,type.measure="mse",family="gaussian")
plot(ridge_PC, main= "Ridge PCA")
lambda.best.ridge.pca<- ridge_PC$lambda.min

mod_ridge_PC = lm(cumulative_incidence ~PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8+PC9+PC10+PC11+PC12+PC13+PC14, da
no.coef.ridge.pca <- dim(summary(mod_ridge_PC)$coefficients)[1]-1
ridge.rsq.pca <- summary(mod_ridge_PC)$r.squared
pred.ridge.pca <- predict(mod_ridge_PC,data = PC.test.sc.set)
mse.ridge.pca <- mean((pred.ridge.pca-PC.test.sc.set$cumulative_incidence)^2)
ggplot() +
  geom_point(aes(x=pred.step.pca,y=PC.test.sc.set$cumulative_incidence, color="Stepwise PCA")) +
```

```
  geom_point(aes(x=pred.lasso.pca,y=PC.test.sc.set$cumulative_incidence, color="Lasso PCA")) +
  labs(y="Empiric Time until 10 Incident Infections", x = "Predicted Time until 10 Incident Infections"
  geom_line(aes(x=PC.test.sc.set$cumulative_incidence, y=PC.test.sc.set$cumulative_incidence))
ggplot() +
  geom_point(aes(x=pred.step.pca,y=PC.test.sc.set$cumulative_incidence, color="Stepwise PCA")) +
  geom_point(aes(x=pred.ridge.pca,y=PC.test.sc.set$cumulative_incidence, color="Ridge PCA")) +
  geom_point(aes(x=pred.lasso.pca,y=PC.test.sc.set$cumulative_incidence, color="Lasso PCA")) +
  labs(y="Empiric Time until 10 Incident Infections", x = "Predicted Time until 10 Incident Infections"
  geom_line(aes(x=PC.test.sc.set$cumulative_incidence, y=PC.test.sc.set$cumulative_incidence))
# build data frame to show performance (MSE) of each type of function on each variable

performance.df.2 <- data.frame(c(no.coef.step, no.coef.lasso, no.coef.ridge, no.coef.step.pca, no.coef.
                               c(NA, bestlam.lasso, bestlam.ridge, NA, lambda.best.lasso.pca, lambda.best
                               c(r2.step, r2.lasso, r2.ridge, step.rsq.pca, lasso.rsq.pca, ridge.rsq.pca)
                               c(mse.step, mse.lasso, mse.ridge, mse.step.pca, mse.lasso.pca, mse.ridge.p
rownames(performance.df.2) <- c("Stepwise Regression", "Lasso Regression", "Ridge Regression", "Stepwis
colnames(performance.df.2) <- c("No. of Coefficients", "Lambda", "R^2", "MSE")

kable(performance.df.2, "latex", caption = "Time Elapsed Model Performance", booktabs = T) %>%
kable_styling(latex_options = c("striped", "hold_position"))
```