CS-7863: Scientific and Statistical Computing
Homework #5
Noah L. Schrick – 1492657

1.) Penalized Regression and Classification

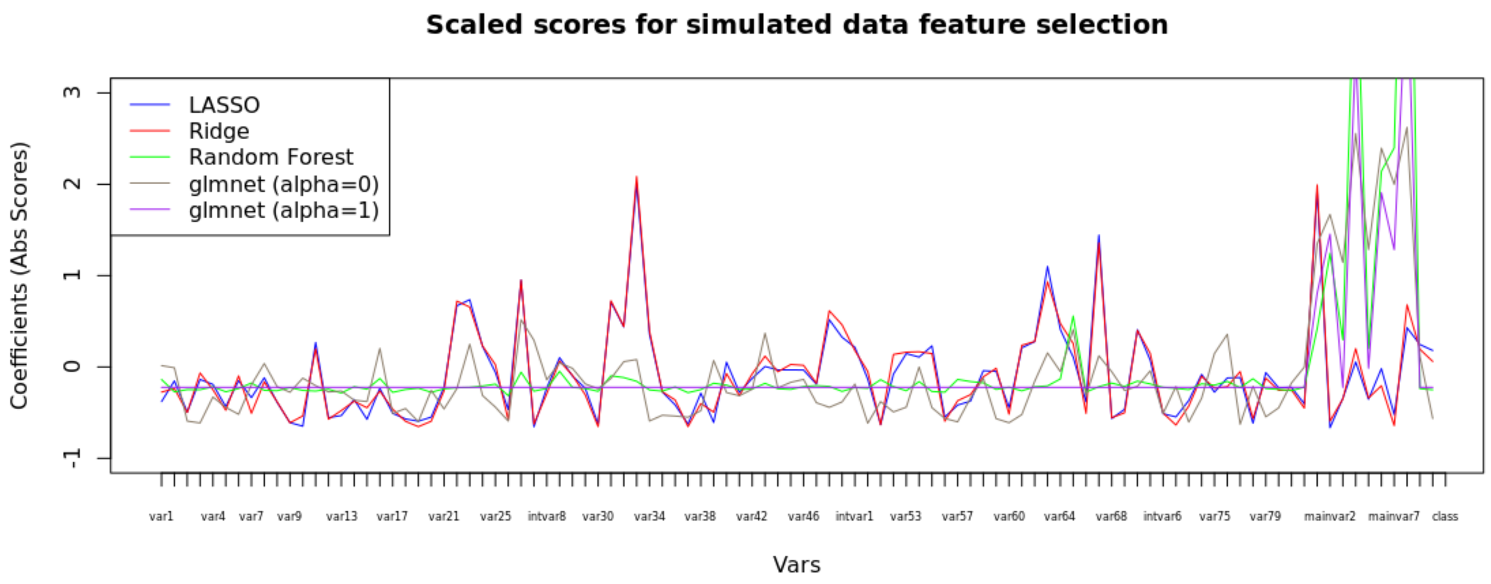
a.)

Separate files were made to store the functions. The files were sourced in a main R script. Not all code is shown due to the length of the functions and auxiliary functions.

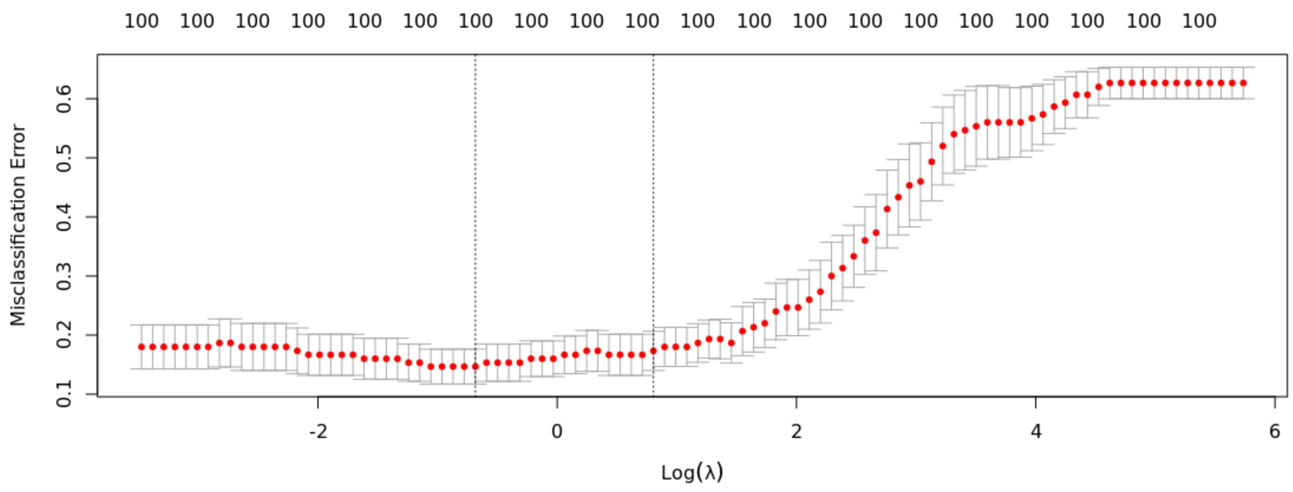
LASSO Code

```
# Regression coeffs for LASSO
lasso_betas <- function(X,y,beta_init=NULL){
  ridge_betas(X,y,beta_init=beta_init,lam=0,alpha=1,method="BFGS")
}

# Adjust betas
lasso_coeff <- function(X, y, lambda=0.03125, tol=1e-2, lap_penalty=0){
  unpen_beta <- lasso_betas(X, y, beta_init=numeric(101))
  old_loss <- unpen_beta$loss
  lasso_converged <- FALSE
  loop_count <- 0
  while (!lasso_converged){
    if(all(lap_penalty==0)) {GL_penalty <- 0} else{
      GL_penalty = lap_penalty %*% as.matrix(unpen_beta$betas[1:100]) %*%
as.matrix(t(unpen_beta$betas[1:100]))
    }

    beta_LS <- optim(unpen_beta$betas,  # guess
           fn=function(beta){penalized_loss(X, y, beta, lam=0, alpha=1)}, #
objective
           gr=function(beta){ridge_grad(X, y, beta, lam=0)}, # gradient
           method = "BFGS") #, control= list(trace = 2))

    for(i in 1:length(beta_LS$par)){
      if(abs(beta_LS$par[i]) <= lambda){ #lambda is 0, so alpha?){
        beta_LS$par[i] <- 0
      }
      else if (beta_LS$par[i] > lambda){
        beta_LS$par[i] <- beta_LS$par[i]-lambda
      }
      else{
        beta_LS$par[i] <- beta_LS$par[i]+lambda
      }
      if(!all(GL_penalty==0)){beta_LS$par[i] <- beta_LS$par[i] - GL_penalty[i]}
    }
    unpen_beta <- lasso_betas(X,y,beta_init=beta_LS$par)
    lasso_converged <- abs(unpen_beta$loss - old_loss) < tol
    if(mod(loop_count, 25) == 0){
      cat("Loop:", loop_count, "Convergence:", abs(unpen_beta$loss - old_loss),"\
n")
    }
    old_loss <- unpen_beta$loss
    loop_count <- loop_count + 1
  }
  print(loop_count)
  return(unpen_beta)
}
```

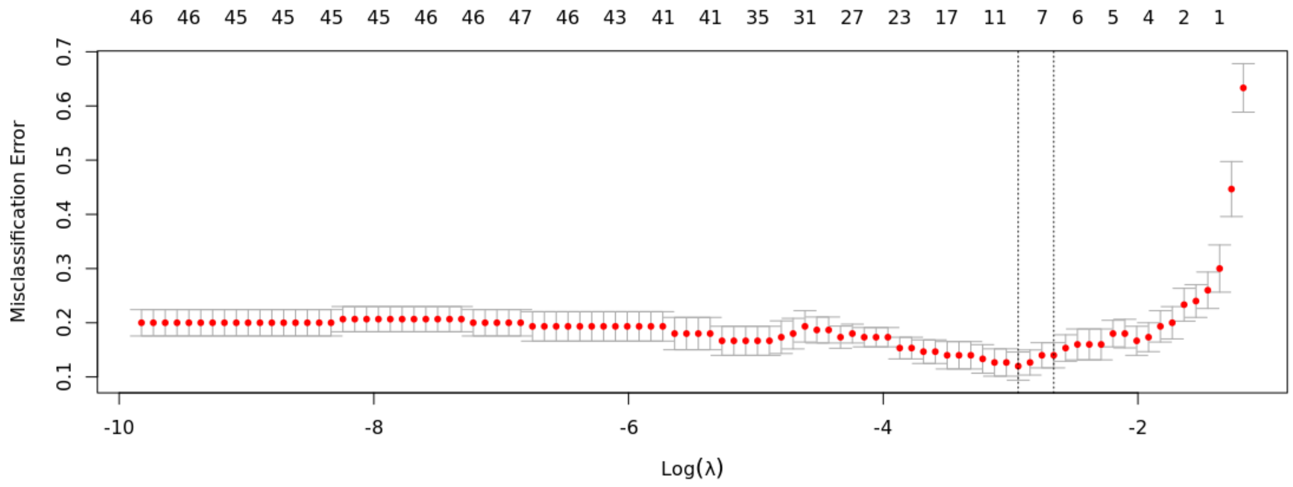
Using the LASSO Code, convergence takes only a few steps if the LASSO beta update and replacement step has bounds set to 0. If the bounds are tuned with the ridge tuning function, convergence takes anywhere from 200-225 steps.
Tables were made to store the scores for LASSO, Ridge, Random Forest, and glmnet. These tables are not shown due to their size. Instead, a graph is shown below. The scores were scaled so that the mean is 0 and the standard deviation is 1.
All approaches increased in score for the main variables, including glmnet with alpha=1, which was otherwise around 0. LASSO and Ridge performed and scored very similarly.



glmnet with alpha=0:



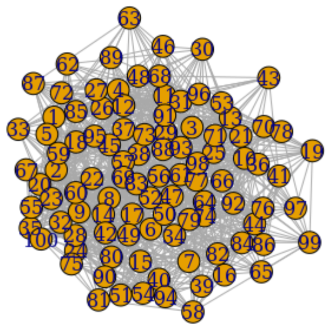
glmnet with alpha=1:



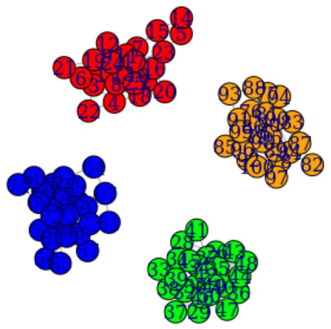
This pattern for glmnet with alpha=1 is also seen in the comparison plot. When alpha is set to 1, the resulting glmnet scores are near 0 for all auxiliary variables, with large spikes in scoring for the main variables.

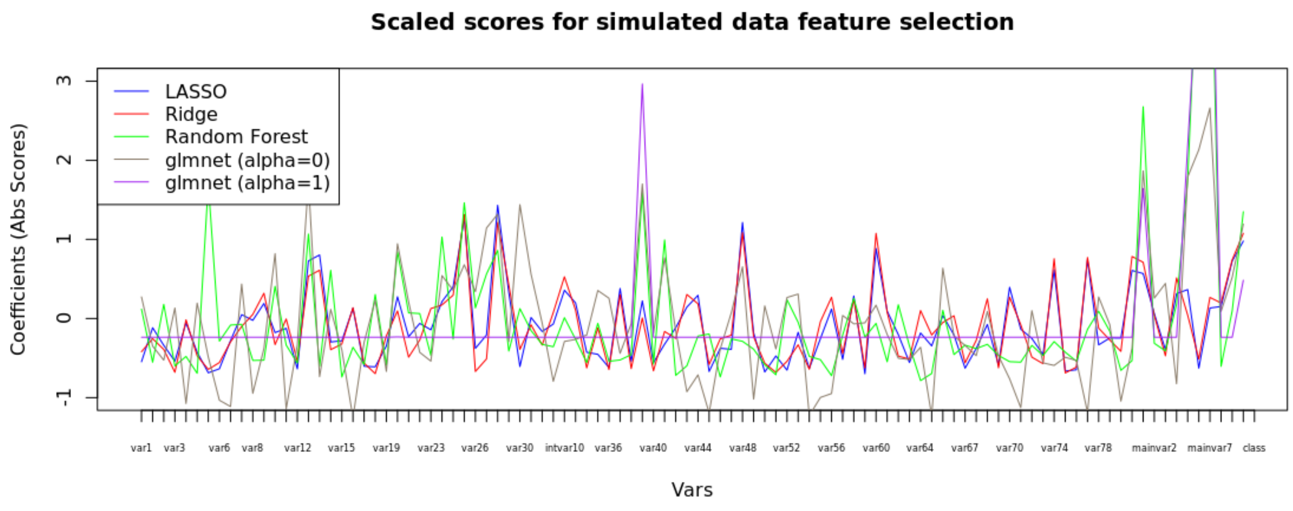
b.)
Repeating Part A, using an Erdos-Renyi graph structure as the input for the simulated data generation with npdro.

Generated Graph:



Community Detection:

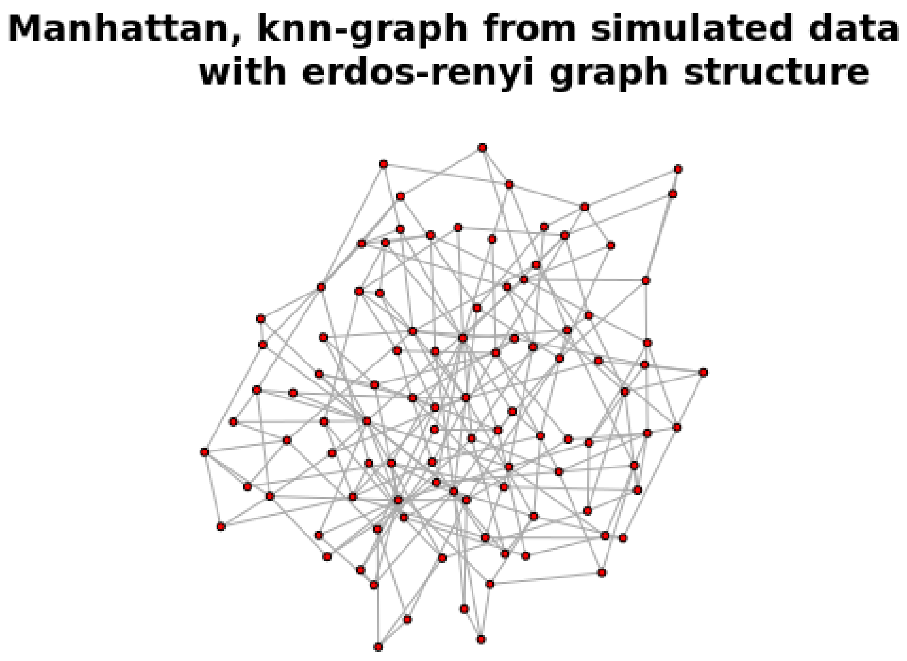




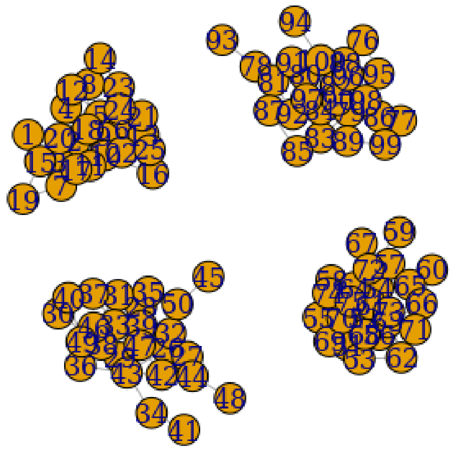
Compared to Part A, there are many more observable peaks. The scores for Random Forest in Part A remained fairly low for all but the main variables. When using the graph structure, Random Forest has amplitudes that were not present without the graph structure. This pattern is observed for the other approaches as well. Glmnet with alpha=1 remained about the same, with scores near 0 until the main variables.

c.)

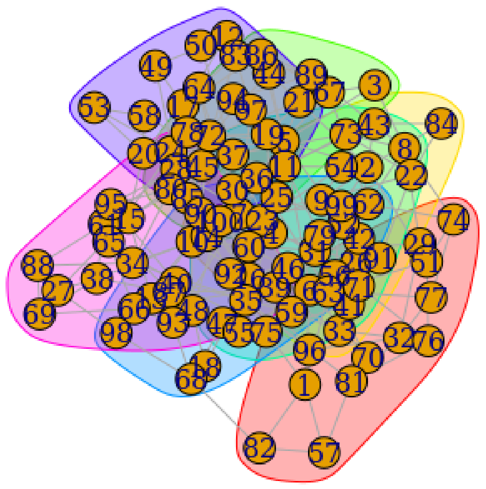
Using knn on the simulated data:


Manhattan, knn-graph from simulated data
with erdos-renyi graph structure

Original Community Detection:

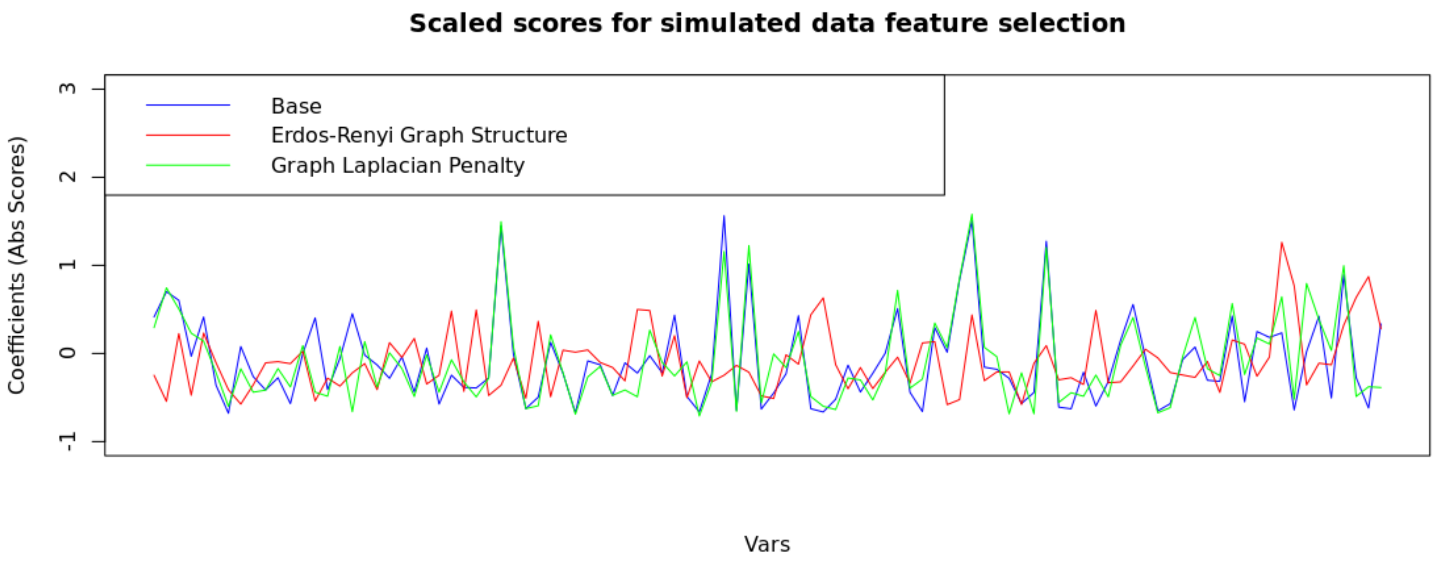


Using Louvain Clustering on the Derived Graph:



In the original graph, 4 main clusters were observed. After deriving the graph from the data, the louvain clustering split the data into noticeably more communities, some with only 1 or 2 members.

d.)
For the graph below, “Base” refers to the LASSO approach performed in Part A, “Erdos-Renyi” refers to the approach in Part B, and “Graph Laplacian Penalty” refers to the addition of the penalty for Part D.


Scaled scores for simulated data feature selection
Base
Erdos-Renyi Graph Structure
Graph Laplacian Penalty
Coefficients (Abs Scores)
Vars
3
2
1
0
-1

2.) Gradient Descent with Learning Parameter and Momentum

```
## Write fn with learning param
grad.rosen <- function(xvec, a=2, b=100){
  x <- xvec[1];
  y <- xvec[2];
  f.x <- -2*(a-x) - 4*b*x*(y-x^2)
  f.y <- 2*b*(y-x^2)
  return( c(f.x, f.y))
}

a = 2
b=100
alpha = .0001 # learning rate
p = c(0,0) # start for momentum

xy = c(-1.8, 3.0)  # guess for solution

# gradient descent
epochs = 1000000
for (epoch in 1:epochs){
  p = -grad.rosen(xy,a,b);
  xy = xy + alpha*p;
}
print(xy)
```

```
[1] 1.999996 3.999982
```