Question 1)

Three mistakes in the pipline?

1) Logistic regression should be used instead of OLS. OLS doesn't give back probabilities for binary events. That is the job of classification algorithms like logistic regression which use sigmoud to give a probability between 0 and 1

$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+...+\beta_n X_n)}}$$

2) It seems like he did not do train test split. Train test split is crucial for making models. Train test split helps modelers maximize the real world results of their model. Train test split helps prevent over fitting or underfitting through the use of regularization techniques. Test data helps assess important measures like Precision and Recall

3) He also might wanna catagorize loans by loans by their amount i.e. loans betwen 1k-10k, 10k-50k,50k-150k,150k-1m. He can give different categories different weights in logisitic regresison. This will make it so that smaller loans don't outweight the bigger loans on shere numbers alone.

3B) He should change his threshold form .5 to .80 because he wants to approve 20% of loans. It's also safer becuase higher probability threshold means historically safer option. He might reject larger loans that might have paid off with .80% but it is the safer option.

Below I show through simulations that logistic regression gives back probabilities.

In [139…

```r
n <- 1000

annual_income <- rnorm(n, mean = 70, sd = 20)
annual_income[annual_income < 0] <- abs(annual_income[annual_income < 0])

credit_score <- rnorm(n, mean = 650, sd = 100)
credit_score[credit_score < 300] <- 300 + abs(credit_score[credit_score < 300])

amount_of_loan <- rnorm(n, mean = 252.5, sd = 120)

duration_of_loan <- rnorm(n, mean = 15.5, sd = 7.5)

outcome <- ifelse(annual_income > 100 | credit_score > 800 | duration_of_loan < 6 | amount_of_loan < 80, 1, 0)

purpose_of_loan <- sample(1:4, n, replace = TRUE, prob = c(0.3, 0.3, 0.2, 0.2))
purpose_dummy <- model.matrix(~ 0 + factor(purpose_of_loan) - 1)
colnames(purpose_dummy) <- c("Home_Purchase_Mortgage", "Mortgage_Refinance", "Unsecured_Personal_Loan", "Loan_to_Start_Expand_Business")

loan_data <- data.frame(
  AnnualIncome = annual_income,
  CreditScore = credit_score,
  AmountOfLoan = amount_of_loan,
  DurationOfLoan = duration_of_loan,
  LoanPaidOff = outcome,
  purpose_dummy
)
head(loan_data, 5)
```

A data.frame: 5 × 9

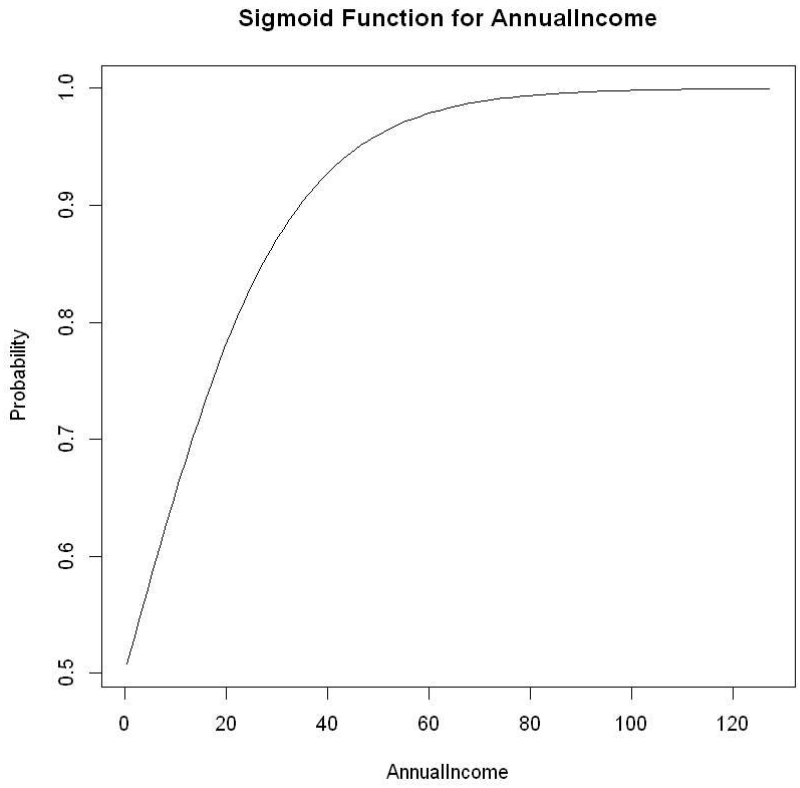
	AnnualIncome	CreditScore	AmountOfLoan	DurationOfLoan	LoanPaidOff	Home_Purchase_Mortgage	Mortgage_Refinance	Unsecured_Personal_Loan	L
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
1	51.84986	683.1542	308.2505	20.36693	0	0	0	1	
2	82.88038	616.2425	227.5786	36.63046	0	1	0	0	
3	111.42354	757.9207	200.1470	11.35679	1	1	0	0	
4	54.88931	684.2441	342.7492	30.21849	0	0	0	0	
5	60.80423	610.3247	345.5871	29.46184	0	0	1	0	

You can see that Logisistic regression gives probabilites. OLS cannot give probailities back

In [137…

```
logit_model <- glm(LoanPaidOff ~ AnnualIncome + CreditScore + AmountOfLoan + DurationOfLoan + Home_Purchase_Mortgage + Mortgage_Refinance +

summary(logit_model)

sigmoid <- function(x) {
  return(1 / (1 + exp(-x)))}
coef_annual_income <- coef(logit_model)["AnnualIncome"]

annual_income_values <- seq(min(loan_data$AnnualIncome), max(loan_data$AnnualIncome), length.out = 100)
sigmoid_values <- sigmoid(coef_annual_income * annual_income_values)

plot(annual_income_values, sigmoid_values, type = "l", main = "Sigmoid Function for AnnualIncome", xlab = "AnnualIncome", ylab = "Probabili
```

```
Call:
glm(formula = LoanPaidOff ~ AnnualIncome + CreditScore + AmountOfLoan +
    DurationOfLoan + Home_Purchase_Mortgage + Mortgage_Refinance +
    Unsecured_Personal_Loan + Loan_to_Start_Expand_Business,
    family = binomial, data = loan_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2114  -0.7241  -0.3345   0.7063   2.7130

Coefficients: (1 not defined because of singularities)
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                 -6.5292470  0.7425368  -8.793  < 2e-16 ***
AnnualIncome                 0.0637226  0.0051161  12.455  < 2e-16 ***
CreditScore                  0.0066420  0.0008793   7.554 4.24e-14 ***
AmountOfLoan                -0.0058950  0.0007544  -7.814 5.52e-15 ***
DurationOfLoan              -0.1046133  0.0117322  -8.917  < 2e-16 ***
Home_Purchase_Mortgage       0.2875333  0.2311007   1.244    0.213
Mortgage_Refinance          -0.0023659  0.2382681  -0.010    0.992
Unsecured_Personal_Loan     -0.2531282  0.2665964  -0.949    0.342
Loan_to_Start_Expand_Business         NA         NA      NA       NA
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1306.84  on 999  degrees of freedom
Residual deviance:  907.92  on 992  degrees of freedom
AIC: 923.92

Number of Fisher Scoring iterations: 5
```



In [38]:
```
rm(list = ls())
```

Question 2: Non-Negative Least Squares (15 points)

In class, we have considered *penalized* regression methods, which combine the MSE loss with a suitable penalty function. We can also add regularization using *constraints*, the most famous form of which is *non-negative least squares*:

$$\arg\min_{\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|_2^2 \text{ such that } \beta_i \geq 0 \text{ for all } i = 1, \ldots p$$

2B) When true Betas are positive then both OLS and NNLS will have same optimal soluton

In [110…

```r
suppressWarnings(library(CVXR, warn.conflicts=FALSE))
set.seed(123)

n <- 10000
p <- 10
beta <- 2:11  # beta is just -9 to-6

X <- matrix(rnorm(n * p), nrow=n)
colnames(X) <- paste0("beta_", beta)
Y <- X %*% beta + rnorm(n)

betaHat <- Variable(p)
objective <- Minimize(sum((Y - X %*% betaHat)^2))
problem <- Problem(objective, constraints = list(betaHat >= 0))
result <- solve(problem)
pos <- matrix(round(result$getValue(betaHat),2), ncol = 1)
rownames(pos) <- paste0("$\\beta_{", 1:p, "}$")
colnames(pos) <- "Non-negative True beta"
```

In [111…

```r
set.seed(123)
n <- 10000
p <- 10
beta <- 2:11

X <- matrix(rnorm(n * p), nrow=n)
colnames(X) <- paste0("beta_", beta)
Y <- X %*% beta + rnorm(n)

# Create a data frame combining X and Y
data <- data.frame(Y = Y, X)

# Run OLS regression
model <- lm(Y ~ 0 + X, data = data)

# Summary of the regression results
ols <- matrix(round(coef(model),2), ncol = 1)
colnames(ols) <- "OLS Solution"

olspos = cbind(ols, pos)
olspos
```

A matrix: 10 × 2 of type dbl

	OLS Solution	Non-negative True beta
β_1	2.01	2.01
β_2	2.97	2.97
β_3	3.99	3.99
β_4	5.00	5.00
β_5	6.01	6.01
β_6	7.01	7.01
β_7	8.01	8.01
β_8	9.01	9.01
β_9	10.00	10.00
β_{10}	10.98	10.98

Question 2C)

It will give sparse solution if the true betas are negative themselves. The true betas from the oracle. Then the predicted betas will be zero. You can see that. Below I ran the code for negative betas and positive betas. the positives betas produced non zoro results.

In [92]:
```
suppressWarnings(library(CVXR, warn.conflicts=FALSE))
set.seed(123)

n <- 10000
p <- 10
beta <- 2:11  # beta is just -9 to-6

X <- matrix(rnorm(n * p), nrow=n)
colnames(X) <- paste0("beta_", beta)
Y <- X %*% beta + rnorm(n)

betaHat <- Variable(p)
objective <- Minimize(sum((Y - X %*% betaHat)^2))
problem <- Problem(objective, constraints = list(betaHat >= 0))
```

```
result <- solve(problem)
pos <- matrix(round(result$getValue(betaHat),2), ncol = 1)
rownames(pos) <- paste0("$\\beta_{", 1:p, "}$")
colnames(pos) <- "Non-negative True beta"
```

In [94]:
```
set.seed(123)

n <- 100
p <- 10
beta <- -11:-2 # This time I'll try

X <- matrix(rnorm(n * p), nrow=n)
colnames(X) <- paste0("beta_", beta)
Y <- X %*% beta + rnorm(n)

betaHat <- Variable(p)
objective <- Minimize(sum((Y - X %*% betaHat)^2))
problem <- Problem(objective, constraints = list(betaHat >= 0))
result <- solve(problem)
neg <- matrix(round(result$getValue(betaHat),2), ncol = 1)
rownames(neg) <- paste0("$\\beta_{", 1:p, "}$")
colnames(neg) <- "Negative True beta"
```

In [95]:
```
both = cbind(pos, neg)
both
```

A matrix: 10 × 2 of type dbl

	Non-negative True beta	Negative True beta
β_1	2.01	0
β_2	2.97	0
β_3	3.99	0
β_4	5.00	0
β_5	6.01	0
β_6	7.01	0
β_7	8.01	0
β_8	9.01	0
β_9	10.00	0
β_{10}	10.98	0

Question 3: Variable Selection & Linear Models (20 points)

Download the SRBCT microarray data from the course website. This is a gene expression data set from a childhood cancer study with $n = 83$ patients and $p = 2308$ genes. Your response (outcome) is the gene expression profile of the gene p53, a major oncogene that acts as a tumor suppressor.

Your goal is to select other genes whose expression profiles are associated with p53 so as to find other possible genes to target in drug therapies.

In [27]:
```
library(glmnet)
```

In [28]:
```
X <- read.csv("X_SR.csv", header = TRUE)
y <- read.csv("Y_SR.csv", header = TRUE)
y <- as.matrix(y)  # If y is a data frame with one column
# or
y <- y[,1]  # If y is a matrix with one column
```

In [29]:
```
head(y)
```

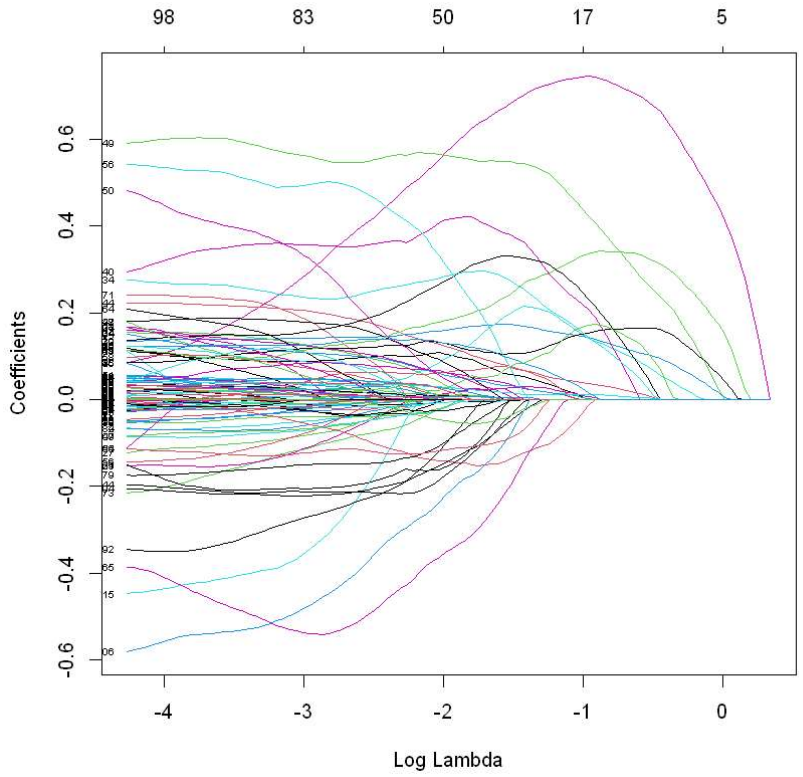
1.7005 · 1.4845 · 0.6491 · 1.4434 · 1.8265 · 2.6457

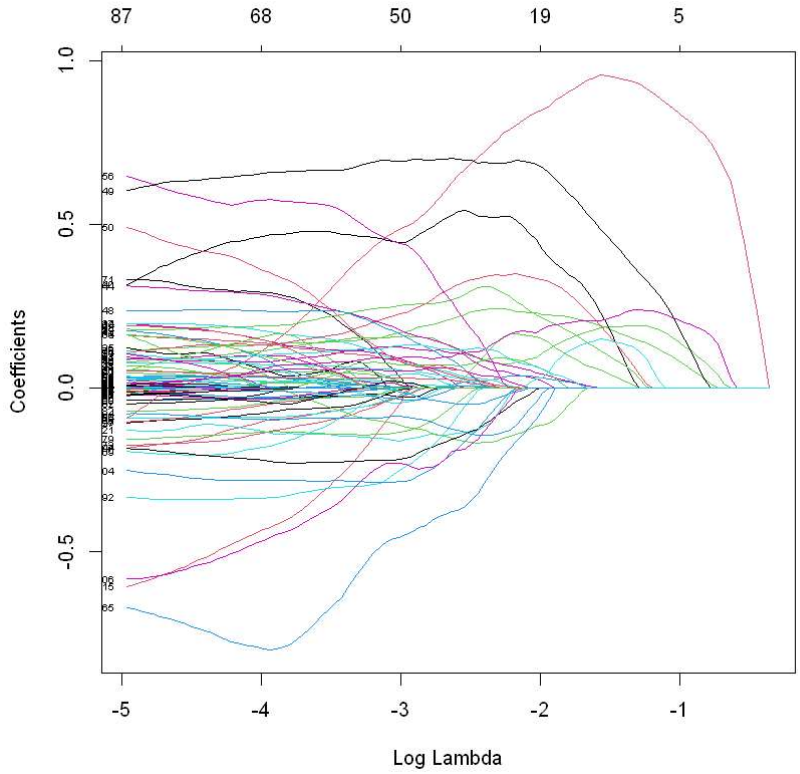
In [30]:
```
head(X,5)
```

A data.frame: 5 × 2307

	X3.2025	X0.0681	X1.046	X0.1243	X0.4941	X3.1207	X3.7106	X1.8416	X1.2607	X2.9001	...	X0.7653	X1.6679.1	X0.1493	X0.6918	X1.4151	X0.2
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	...	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<d
1	1.6547	0.0710	1.0409	0.0520	0.2045	2.1609	2.4452	1.1473	0.7371	1.9989	...	1.0665	3.6014	0.3048	1.7957	1.0701	0..
2	3.2779	0.1160	0.8926	0.1014	0.2818	1.9773	3.2590	1.4106	0.9548	2.0775	...	1.2674	1.5152	0.2382	0.8720	0.6819	0..
3	1.0060	0.1906	0.4302	0.1035	0.2984	1.6804	5.8901	0.2958	0.7381	1.6610	...	0.4743	1.0282	0.1049	0.5632	1.2264	0.
4	2.7098	0.2367	0.3693	0.2190	0.3711	1.7800	3.2376	0.6769	0.8546	0.6808	...	0.7039	0.5961	0.0707	0.4001	1.5271	0.
5	2.0588	0.0823	0.9021	0.1288	0.3961	1.7199	2.1729	1.5609	0.6581	1.5038	...	1.9235	1.3635	0.0580	0.7737	2.8123	0..

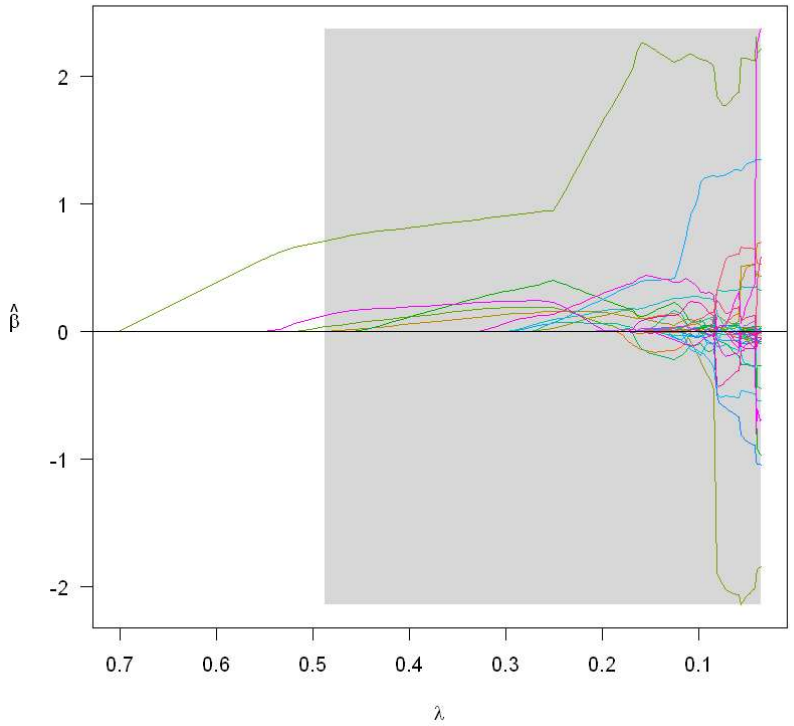
In [31]:
```
elastic_net_model <- glmnet(X, y, alpha = .5)

# Plot Elastic Net
plot(elastic_net_model, xvar = "lambda", label = TRUE)

# Fit Lasso model
lasso_model <- glmnet(X, y, alpha = 1)
plot(lasso_model, xvar = "lambda", label = TRUE)
```

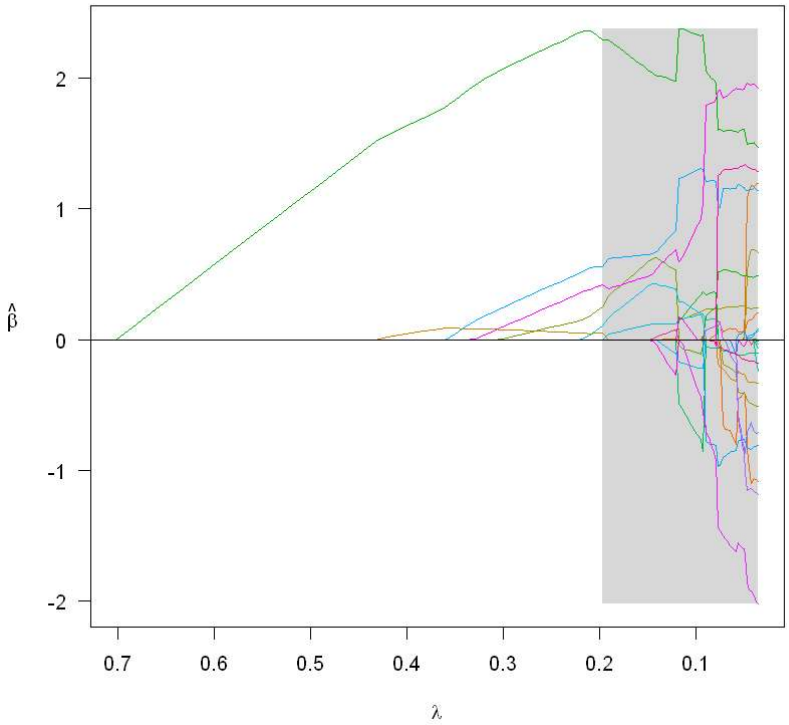

98
83
50
17
5
Coefficients
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-4
-3
-2
-1
0
Log Lambda



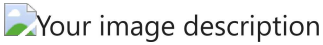
Below are the plots for SCAD and MCP

In [32]:
```
library(ncvreg)

# SCAD model
scad_model <- ncvreg(X, y, penalty = "SCAD")

# Plot SCAD coefficient
plot(scad_model)

# MCP model
mcp_model <- ncvreg(X, y, penalty = "MCP")

plot(mcp_model)
```


2
1
0
-1
-2
β̂
0.7
0.6
0.5
0.4
0.3
0.2
0.1
λ



3B) What are the top genese associated with p53. I beleive it's the below genes. There are 4 columns of the best genese from each of the four models. I would choose to report genese that show up in these 10 by 4 matrix. Genes such as 2281, 56, 4659, 3534, 4089 etc



Below is how i estracted those genes

In [60]:
```
#elastic_net_model$beta
beta_matrix <- lasso_model$beta
na_counts <- rowSums(is.na(beta_matrix))
zero_counts <- rowSums(beta_matrix == 0)
least_zero_rows <- order(zero_counts)[1:10]
least_zero_rows_beta <- beta_matrix[least_zero_rows, ]
head(least_zero_rows_beta, 10)
```

```
 [[ suppressing 10 column names 's0', 's1', 's2' ... ]]
```

```
4 x 10 sparse Matrix of class "dgCMatrix"

X0.2281 . 0.07301861 0.1453038 0.2167896 0.28005960 0.331792927 0.3732311
X0.2945 . .          .         .         0.02222327 0.072447204 0.1146325
X1.56   . .          .         .         .          0.001559192 0.0225771
X1.4659 . .          .         .         .          .           .

X0.2281 0.412472888 0.44416703 0.47238139
X0.2945 0.149438182 0.17265285 0.19321832
X1.56   0.041663248 0.05761304 0.07258420
X1.4659 0.005385527 0.01599961 0.02649913
```

In [ ]:

Question 4)

In [ ]:
```r
library(MASS)
library(mvtnorm)
```

In [ ]:
```r
library(glmnet)
set.seed(12)

n <- 100  # Number of observations
p <- 150  # Number of variables

cor_matrix <- matrix(0.80, nrow = p, ncol = p)
diag(cor_matrix) <- 1
X <- rmvnorm(n, mean = rep(0, p), sigma = cor_matrix)

beta <- c(1:p)

y <- X %*% beta + rnorm(n)
```

You can see high correlation

In [135…
```r
cormat = cor(X)
head(cormat,5)
```

A matrix: 5 × 150 of type dbl

1.0000000	0.7641157	0.7457660	0.7708147	0.7900184	0.7758112	0.7688534	0.7558713	0.7389718	0.7674886	...	0.7701148	0.7371899	0.7322761	0.765306
0.7641157	1.0000000	0.7195994	0.7645427	0.7228451	0.7064093	0.7063400	0.7201055	0.6723874	0.7631816	...	0.7795520	0.7187713	0.7407701	0.737521
0.7457660	0.7195994	1.0000000	0.7438948	0.7751254	0.7105564	0.7716995	0.7247976	0.7440197	0.7397973	...	0.7191575	0.7547990	0.7011056	0.680115
0.7708147	0.7645427	0.7438948	1.0000000	0.7504449	0.7228669	0.7224988	0.7175737	0.7530730	0.7125661	...	0.7426357	0.7564177	0.7249648	0.743656
0.7900184	0.7228451	0.7751254	0.7504449	1.0000000	0.7744287	0.7870578	0.7542018	0.7591108	0.7670330	...	0.7619610	0.7507964	0.7693220	0.794059

In [136…
```
head(y,5)
```

A matrix: 5 × 1 of type dbl

-820.4312
-3281.5239
-4464.2224
-14651.0893
-6856.1869

Below is the ridge MSE based on regularizatoin and OLS MSE in bottom. I don't know why OLS MSE is smaller than ridge. OLS might be overfitting

In [137…
```
alpha <- 0  # Ridge regression (alpha = 0)
lambda_seq <- seq(0, 4, by = .1)  # Sequence of lambda values
ridge_model <- glmnet(X, y, alpha = alpha, lambda = lambda_seq)

ridge_predicted <- predict(ridge_model, newx = X, s = lambda_seq)

ridge_mse <- apply(ridge_predicted, 2, function(pred) mean((pred - y)^2))
print("Ridge MSE")
print(ridge_mse)

ols_model <- lm(y ~ X)
y_predicted <- predict(ols_model)
mse <- mean((y_predicted - y)^2)
```

```
[1] "Ridge MSE"
      s1       s2       s3       s4       s5       s6       s7       s8
 464.2906  495.9607  529.8510  566.0993  604.8523  646.2664  690.5084  737.7557
      s9       s10      s11      s12      s13      s14      s15      s16
 788.1977  842.0358  899.4848  960.7734 1026.1452 1095.8595 1170.1927 1249.4390
     s17      s18      s19      s20      s21      s22      s23      s24
1333.9116 1423.9444 1519.8926 1622.1348 1731.0742 1847.1402 1970.7906 2102.5128
     s25      s26      s27      s28      s29      s30      s31      s32
2242.8269 2392.2870 2551.4843 2721.0495 2901.6558 3094.0220 3298.9157 3517.1572
     s33      s34      s35      s36      s37      s38      s39      s40
3749.6230 3997.2496 4261.0384 4542.0604 4841.4621 5160.4748 5500.4275 5862.7626
     s41
6249.0369
```

In [138…
```
print("OLS MSE below")
print(mse)
```

```
[1] "OLS MSE below"
[1] 2.113656e-23
```

Ridge regression should have lower MSE because it doesn't overfit. Maybe it's because there are more variables than observations. This looks like the job for lasso

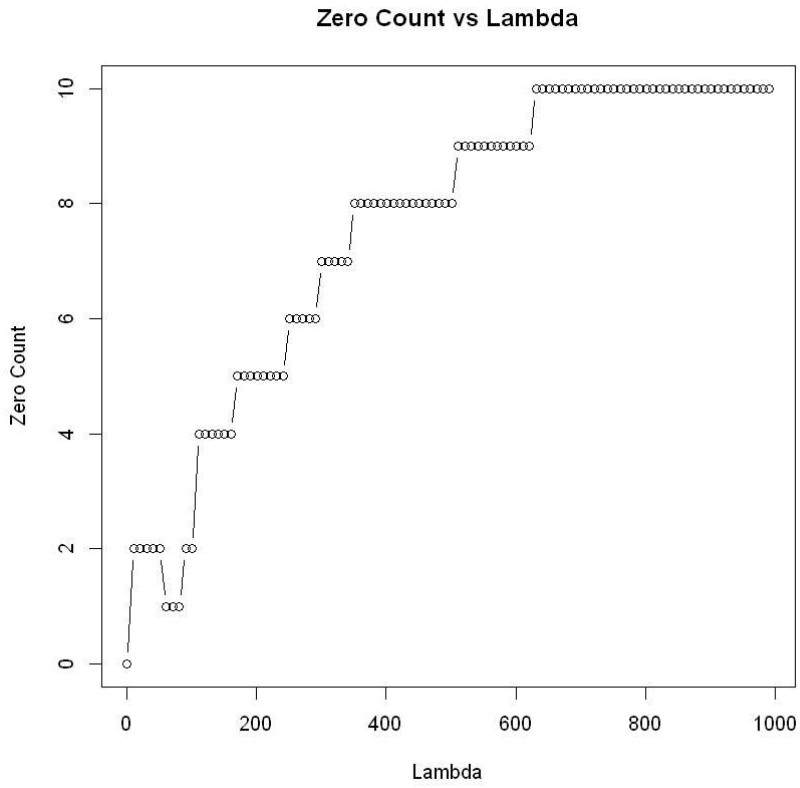
Question 5)

5 A & B)

In [82]:
```
df <- read.csv("boston.csv")

y <- df$medv
X <- as.matrix(df[, -which(names(df) == "medv")])

X_standardized <- scale(X)
```

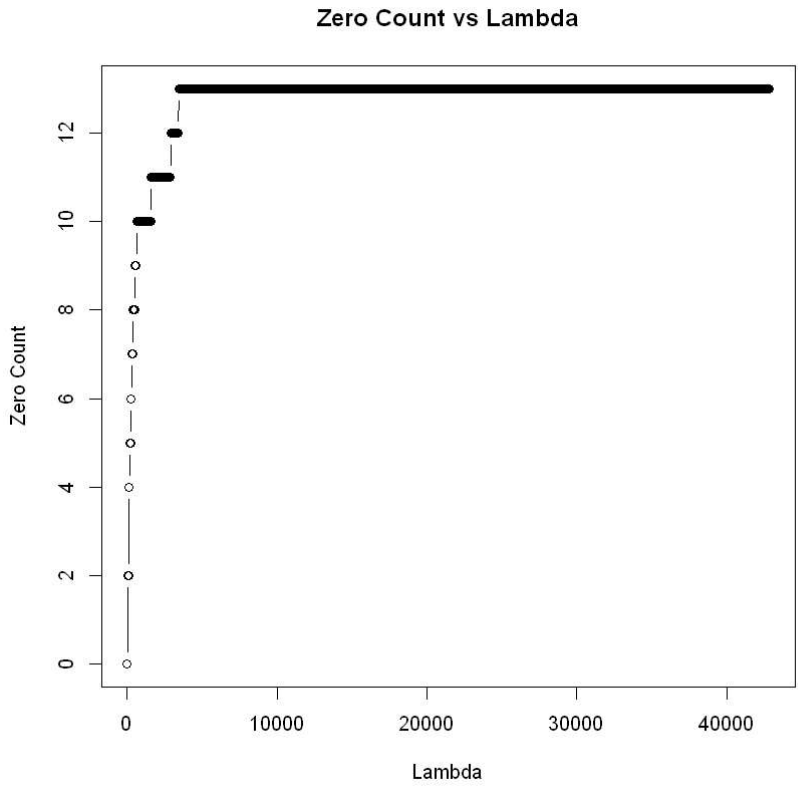
In [83]:
```
lasso_coord_descent <- function(X, y, lambda, max_iterations = 300, tolerance = 1e-5) {

  ## This is the start beta
  beta <- rep(0, ncol(X))

  beta_old <- beta
  for(iter in 1:max_iterations) {
    for(j in 1:length(beta)) {
```

```r
      y_hat_except_j <- X[, -j] %*% beta[-j]
      r <- y - y_hat_except_j
      rho <- crossprod(X[, j], r)
      beta[j] <- soft_threshold(rho, lambda) / crossprod(X[, j])
    }
    if(sqrt(sum((beta - beta_old)^2)) < tolerance) {
      break
    }
    beta_old <- beta
  }
  return(beta)
}

soft_threshold <- function(rho, lambda) {
  if(rho < -lambda) return(rho + lambda)
  if(rho > lambda) return(rho - lambda)
  return(0)
}

lambda <- seq(1, 1000, by = 10)
zero_counts <- numeric(length(lambda))

for (i in seq_along(lambda)) {
  beta_estimates <- lasso_coord_descent(X_standardized, y, lambda[i])
  zero_counts[i] <- sum(beta_estimates == 0)
}

plot(lambda, zero_counts, type = "b", xlab = "Lambda", ylab = "Zero Count", main = "Zero Count vs Lambda")
```



In [84]:

```
LAMBDA_MAX <- max(abs(crossprod(X, y)))
lambda <- seq(0.00000000000000001 *LAMBDA_MAX, .01*LAMBDA_MAX, length.out=1000)

zero_counts <- numeric(length(lambda))

for (i in seq_along(lambda)) {
  beta_estimates <- lasso_coord_descent(X_standardized, y, lambda[i])
  zero_counts[i] <- sum(beta_estimates == 0)
}
plot(lambda, zero_counts, type = "b", xlab = "Lambda", ylab = "Zero Count", main = "Zero Count vs Lambda")
```



Compare Coord D with CVXR. I think CVXR is much slower. CD does help lasso a lot.

In [ ]:
```
library(CVXR)
```

In [90]:
```
df <- read.csv("boston.csv")

y <- df$medv
X <- as.matrix(df[, -which(names(df) == "medv")])

X_standardized <- scale(X)


lambda_values <- seq(0.1, 400, by=10)
beta_coefficients <- list()
b0_values <- numeric(length(lambda_values))

for (i in seq_along(lambda_values)) {
    lambda <- lambda_values[i]
```

```
    beta <- Variable(p)
    b0 <- Variable(1)

    objective <- Minimize(sum((y - (b0 + X_standardized %*% beta))^2) + lambda * p_norm(beta, 1))

    problem <- Problem(objective)
    result <- solve(problem)

    beta_coefficients[[i]] <- result$getValue(beta)
    b0_values[i] <- result$getValue(b0)
}
```