Project: Breast Cancer Diagnosis

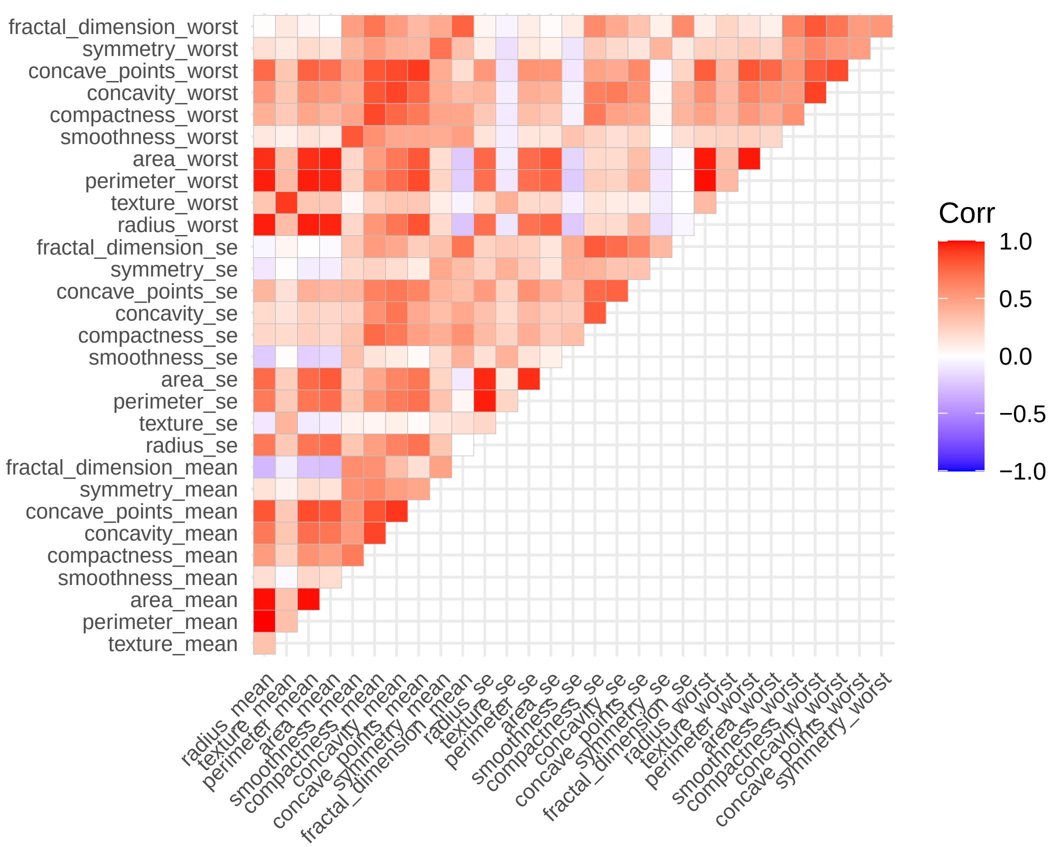
Shengzhi Luo

3/31/2022

```
ggplot2::theme_set(theme_minimal() + theme(legend.position = "bottom"))
```

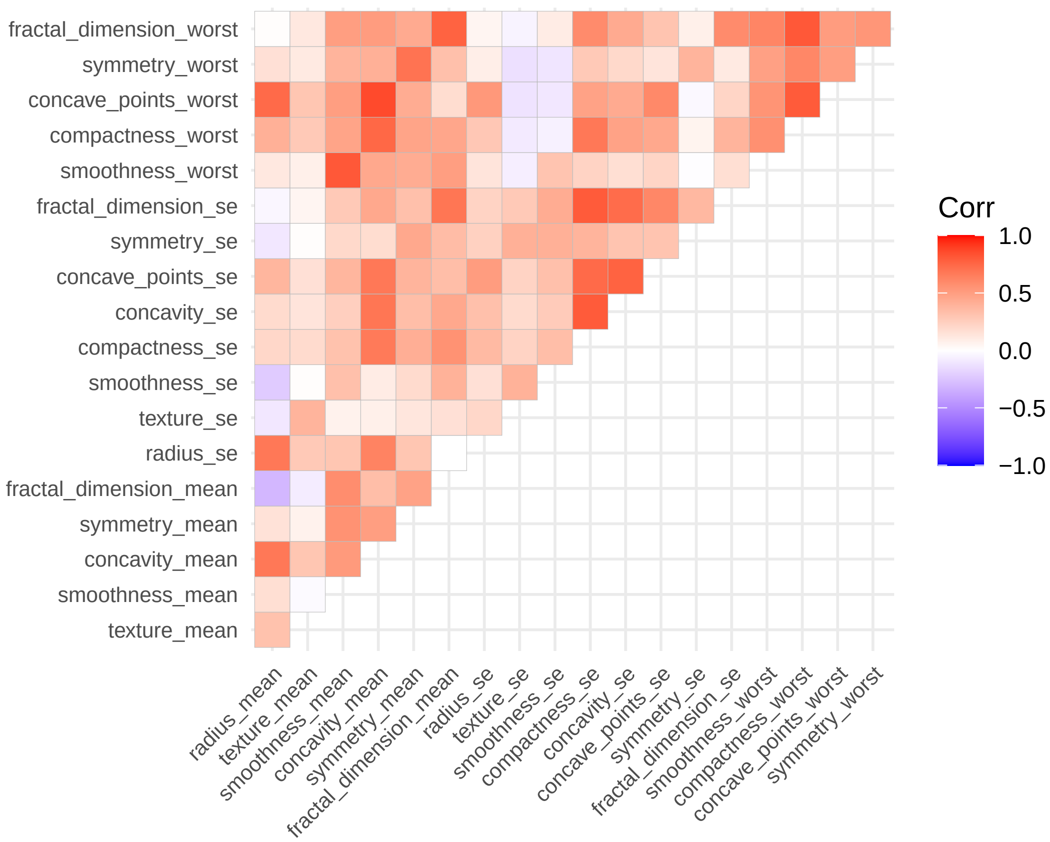
data import and data clean

```
#load the data
breast = read.csv("breast-cancer.csv") %>%
  janitor::clean_names() %>%
  dplyr::select(-1, -33) %>% #drop id and NA columns
  mutate(diagnosis = recode(diagnosis, "M" = 1, "B" = 0))
#check collinearity
corr = breast[2:31] %>%
  cor()
ggcorrplot(corr, type = "upper", tl.cex = 8)
```



```r
#remove some highly correlated variables
breast_dat <- breast %>% dplyr::select(-area_se, -perimeter_se, -area_worst, -perimeter_mean, -perimete

corr1 = breast_dat[2:20] %>%
  cor()
ggcorrplot(corr1, type = "upper", tl.cex = 8)
```




```r
#partition data into training and test data
set.seed(2022)
trainRows <- createDataPartition(y = breast_dat$diagnosis, p = 0.8, list = FALSE)
breast_train <- breast_dat[trainRows, ]
breast_test <-  breast_dat[-trainRows, ]

head(breast_dat, 5)
```

```
##   diagnosis radius_mean texture_mean smoothness_mean concavity_mean
## 1         1       17.99        10.38         0.11840         0.3001
## 2         1       20.57        17.77         0.08474         0.0869
## 3         1       19.69        21.25         0.10960         0.1974
## 4         1       11.42        20.38         0.14250         0.2414
## 5         1       20.29        14.34         0.10030         0.1980
##   symmetry_mean fractal_dimension_mean radius_se texture_se smoothness_se
## 1        0.2419                0.07871    1.0950     0.9053      0.006399
## 2        0.1812                0.05667    0.5435     0.7339      0.005225
## 3        0.2069                0.05999    0.7456     0.7869      0.006150
## 4        0.2597                0.09744    0.4956     1.1560      0.009110
## 5        0.1809                0.05883    0.7572     0.7813      0.011490
```

```
##   compactness_se concavity_se concave_points_se symmetry_se
## 1        0.04904      0.05373           0.01587     0.03003
## 2        0.01308      0.01860           0.01340     0.01389
## 3        0.04006      0.03832           0.02058     0.02250
## 4        0.07458      0.05661           0.01867     0.05963
## 5        0.02461      0.05688           0.01885     0.01756
##   fractal_dimension_se smoothness_worst compactness_worst concave_points_worst
## 1             0.006193           0.1622            0.6656               0.2654
## 2             0.003532           0.1238            0.1866               0.1860
## 3             0.004571           0.1444            0.4245               0.2430
## 4             0.009208           0.2098            0.8663               0.2575
## 5             0.005115           0.1374            0.2050               0.1625
##   symmetry_worst fractal_dimension_worst
## 1         0.4601                 0.11890
## 2         0.2750                 0.08902
## 3         0.3613                 0.08758
## 4         0.6638                 0.17300
## 5         0.2364                 0.07678
```

```
r = dim(breast_dat)[1] #row number
c = dim(breast_dat)[2] #column number
var_names = names(breast_dat)[-c(1,2)] #variable names

standardize = function(col) {
  mean = mean(col)
  sd = sd(col)
  return((col - mean)/sd)
}
stand_df = breast_dat %>%
  dplyr::select(radius_mean:fractal_dimension_worst) %>%
  map_df(.x = ., standardize) #standardize
X = stand_df #predictors
y = breast_dat[,1]#response
```

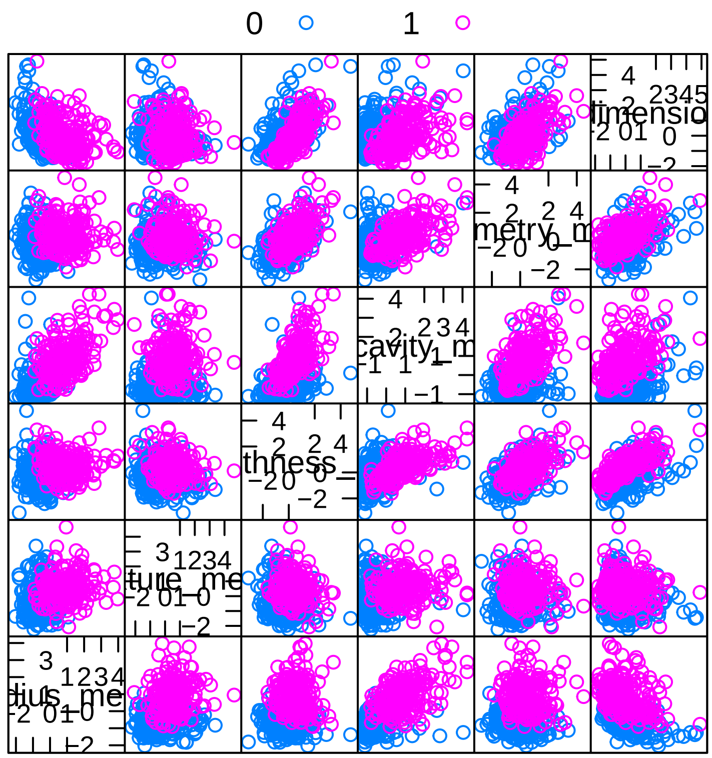
```
x_train <- breast_train[2:20] #predictors
y_train <- breast_train[1] #response
x_train_stan <- cbind(rep(1, nrow(x_train)), scale(x_train))

x_test <- breast_test[2:20]
x_test_stan <- cbind(rep(1, nrow(x_test)), scale(x_test))
```

feature plot

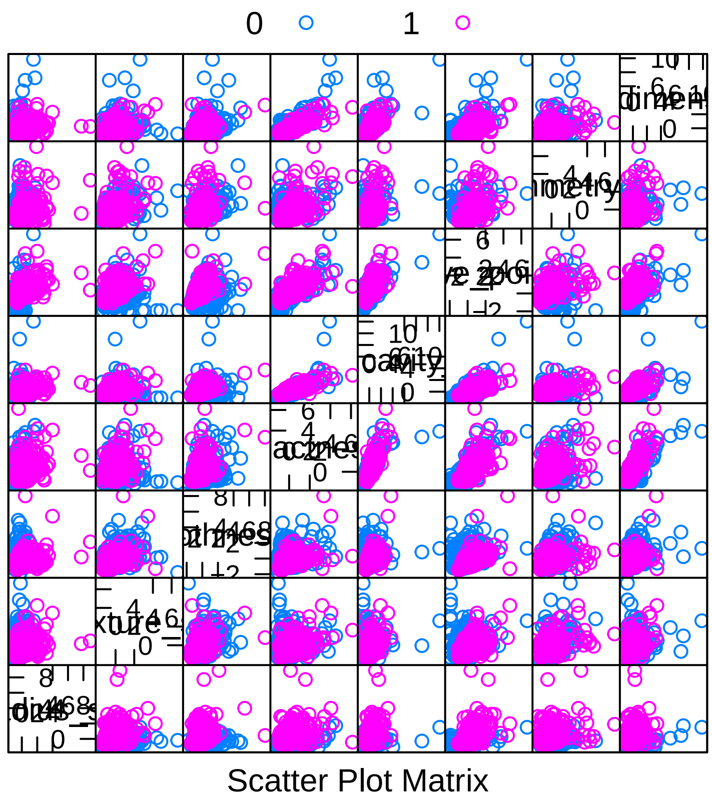
```
data = cbind(y,X)

featurePlot(x = data[, 2:7],
            y = factor(data$y),
            plot = "pairs",
            auto.key = list(columns = 2)
)
```

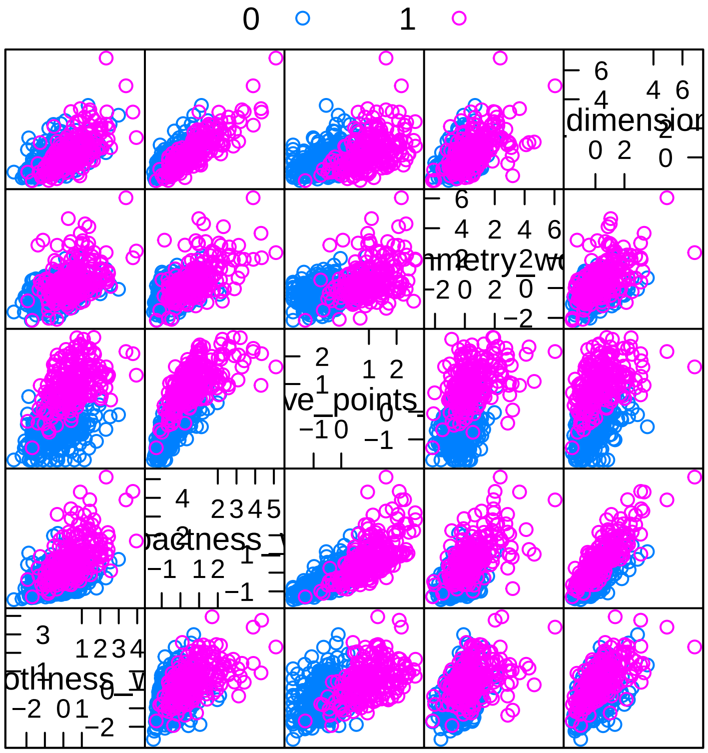


Scatter Plot Matrix

```
featurePlot(x = data[, 8:15],
            y = factor(data$y),
            plot = "pairs",
            auto.key = list(columns = 2)
)
```




```r
featurePlot(x = data[, 16:20],
            y = factor(data$y),
            plot = "pairs",
            auto.key = list(columns = 2)
)
```



Scatter Plot Matrix

```r
mean_data = breast_dat %>%
  group_by(diagnosis) %>%
  summarise(across(radius_mean: fractal_dimension_worst, ~ mean(.x, na.rm = TRUE)))
mean_data
```

```
## # A tibble: 2 x 20
##   diagnosis radius_mean texture_mean smoothness_mean concavity_mean
##       <dbl>       <dbl>        <dbl>           <dbl>          <dbl>
## 1         0        12.1         17.9          0.0925         0.0461
## 2         1        17.5         21.6          0.103          0.161
## # ... with 15 more variables: symmetry_mean <dbl>,
## #   fractal_dimension_mean <dbl>, radius_se <dbl>, texture_se <dbl>,
## #   smoothness_se <dbl>, compactness_se <dbl>, concavity_se <dbl>,
## #   concave_points_se <dbl>, symmetry_se <dbl>, fractal_dimension_se <dbl>,
## #   smoothness_worst <dbl>, compactness_worst <dbl>,
## #   concave_points_worst <dbl>, symmetry_worst <dbl>,
## #   fractal_dimension_worst <dbl>
```

Full logistic model

```r
glm.fit <- glm(diagnosis ~ .,
               data = breast_train,
               family = binomial)

summary(glm.fit)$coefficients %>% knitr::kable()
```

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-61.1480991	22.7005353	-2.6936853	0.0070667
radius_mean	0.4087734	0.5194231	0.7869758	0.4312960
texture_mean	0.7991309	0.2960648	2.6991761	0.0069511
smoothness_mean	112.0773037	108.6087742	1.0319360	0.3021021
concavity_mean	81.0558072	35.4365649	2.2873494	0.0221754
symmetry_mean	-74.1111829	39.6817182	-1.8676405	0.0618122
fractal_dimension_mean	-344.5973095	228.6957280	-1.5067938	0.1318635
radius_se	39.6728660	14.0741918	2.8188379	0.0048198
texture_se	-0.4026481	1.5678789	-0.2568107	0.7973249
smoothness_se	442.3410192	418.9499688	1.0558326	0.2910447
compactness_se	380.5961088	185.4664743	2.0521019	0.0401598
concavity_se	-74.9595207	51.3448406	-1.4599231	0.1443112
concave_points_se	-210.2627737	404.8257177	-0.5193909	0.6034882
symmetry_se	-486.7748560	225.4609542	-2.1590207	0.0308486
fractal_dimension_se	-3184.3013759	1568.2807496	-2.0304409	0.0423117
smoothness_worst	-41.9855490	75.0498013	-0.5594358	0.5758643
compactness_worst	-72.5516143	28.7121732	-2.5268590	0.0115088
concave_points_worst	144.8910643	66.8810152	2.1664005	0.0302806
symmetry_worst	80.0311702	29.5678265	2.7066978	0.0067956
fractal_dimension_worst	480.1713745	207.0899338	2.3186611	0.0204134

```r
glm.fit %>% predict(breast_test, type = "response")
```

```
##           14           21           27           28           30           43 
## 4.435159e-01 1.115467e-08 1.000000e+00 1.000000e+00 9.999140e-01 1.000000e+00 
##           50           52           60           62           68           71 
## 1.085158e-01 2.823488e-06 8.386401e-10 2.385735e-07 1.174097e-06 1.000000e+00 
##           75           87           88           90           98           99 
## 5.368857e-05 9.796329e-01 1.000000e+00 9.793390e-01 8.378933e-09 3.892338e-06 
##          100          108          109          116          125          126 
## 9.542554e-01 1.290556e-06 1.000000e+00 2.117744e-04 5.907608e-09 9.177100e-08 
##          128          135          141          149          152          164 
## 9.999965e-01 1.000000e+00 3.389340e-13 6.584321e-02 2.765556e-08 6.483175e-05 
##          165          171          180          183          192          196 
## 1.000000e+00 1.121857e-07 6.303818e-10 9.999949e-01 3.618526e-08 1.294629e-06 
##          198          199          212          213          217          222 
## 9.997821e-01 9.999741e-01 6.334345e-07 1.000000e+00 4.787718e-06 9.813174e-06 
##          237          241          244          249          250          258 
## 1.000000e+00 9.670721e-05 4.672950e-04 4.301725e-03 2.809702e-06 1.000000e+00 
##          261          264          265          275          284          292 
## 1.000000e+00 2.147741e-03 1.000000e+00 9.999649e-01 9.999801e-01 7.142970e-01 
##          294          300          312          317          320          323 
## 8.644211e-06 1.585715e-10 8.593789e-06 1.689966e-12 1.011243e-13 6.278814e-05 
##          324          325          327          332          333          343 
## 1.000000e+00 3.471332e-07 5.700384e-08 8.543291e-05 7.596965e-11 2.641163e-05 
##          349          354          356          357          358          360 
## 6.191597e-06 1.000000e+00 3.071796e-05 3.943388e-05 7.153699e-07 1.343892e-03 
##          364          377          386          394          398          408 
## 1.615436e-02 7.813589e-08 9.993107e-01 1.000000e+00 3.386318e-06 4.994989e-07 
##          413          418          421          434          439          440 
## 9.538364e-11 1.000000e+00 3.659893e-06 1.000000e+00 2.240033e-05 2.328171e-07 
##          441          444          453          456          458          459 
```

```
## 1.268173e-03 2.220446e-16 6.978071e-05 8.146849e-01 2.043044e-04 1.468687e-04
##          461          478          479          481          482          484
## 1.000000e+00 7.212287e-09 8.951327e-07 3.716606e-08 3.215714e-02 1.219353e-04
##          491          495          496          519          520          525
## 1.468055e-03 1.425063e-06 1.498459e-03 1.100079e-02 2.512633e-04 1.379163e-07
##          528          538          540          543          546          558
## 5.401006e-06 2.400588e-02 3.067226e-07 8.691250e-02 3.067696e-03 7.608324e-06
##          559          564          565          568          569
## 2.577921e-04 1.000000e+00 1.000000e+00 1.000000e+00 1.033328e-08
```

```r
pred <- predict(glm.fit, breast_test, type = "response")
y_test <- factor(breast_test$diagnosis)
auc_full <- auc(y_test, pred)
auc_full
```

```
## Area under the curve: 0.994
```

Newton-Raphson algorithm

```r
# Write a function that generate log-likelihood, gradient and Hessian
# Inputs:

# x - data variables
# y - outcome
# par - vector of beta parameters
func = function(x, y, par) {

# Log link x*beta
  u = x %*% par
  expu = exp(u)

loglik = vector(mode = "numeric", length(y))
for(i in 1:length(y))
  loglik[i] = y[i]*u[i] - log(1 + expu[i])
loglik_value = sum(loglik)

# Log-likelihood at betavec
p <- 1 / (1 + exp(-u))

# P(Y_i=1|x_i)
grad = vector(mode = "numeric", length(par))

#grad[1] = sum(y - p)
for(i in 1:length(par))
  grad[i] = sum(t(x[,i])%*%(y - p))

#Hess <- -t(x)%*%p%*%t(1-p)%*%x
Hess = hess_cal(x, p)
return(list(loglik = loglik_value, grad = grad, Hess = Hess))

}

# Function to return the Hessian matrix
hess_cal = function(x,p){
```

```
len = length(p)
hess = matrix(0, ncol(x), ncol(x))
for (i in 1:len) {

x_t = t(x[i,])

unit = t(x_t)%*%x_t*p[i]*(1-p[i])

#unit = t(x[i,])%*%x[i,]*p[i]*(1-p[i])

hess = hess + unit
}
return(-hess)

}
```

2. Newton-Raphson algorithm

input: x: predictors without intercept y: response variables beta: if not specified, 0 will be set to all coefficients tol: the threshold to end up the function if the difference between loglike function at 2 adjacent steps below this value. lambda_init: the initial lambda to control the number of each step and lambda will change in halving process. decay_rate: the ratio of decayed lambda to lambda at last step in havling process.

output: beta: a vector of coeffients3

```
newton_optimize = function(x, y, beta = NULL, tol = 0.00001, lambda_init = 1, decay_rate = 0.5){

  # add the intercept
  x = cbind(rep(1, nrow(x)), x)

  # if beta is not specified, set all initial coefficients to 0
  if (is.null(beta))
    beta = matrix(rep(0, ncol(x)))

  # calculate the initial gradient, Hessian matrix and negative loglike funtion
  optimization = func(x, y, beta)
  step = 1
  previous_loglik = -optimization$loglik

  # start the interations to optimize the beta
  while (TRUE) {
    print(paste("step:", step, "  negative loglike loss:", -optimization$loglik))

    # set initial lambda at this step equals to the parameters, this variable will change in havling st
    lambda = lambda_init

    # since there maybe some issues when calculate new beta, so we use try-catch sentence. If some erro
    beta_new <- tryCatch({
        beta - lambda * inv(optimization$Hess) %*% optimization$grad # calculate new beta, if no errors
      }, error = function(err) {return(beta)})


    # calculate gradient, Hessian and loglike
    optimization = func(x, y, beta_new)
```

```r
    # havling steps start only when it optimizes at opposite direction.
    # if it optimizes at opposite direction, lambda will be havled to make the step smaller.
    while (previous_loglik <= -optimization$loglik) {
      lambda = lambda * decay_rate # lambda decay

      # same reason to use try-catch
      # but if errors occur, although beta keeps, the lambda will be havled at next step, makes the res
      beta_new <- tryCatch({
        beta - lambda * inv(optimization$Hess) %*% optimization$grad
      }, error = function(err) {return(beta)})

      # optimize by decayed lambda
      optimization = func(x, y, beta_new)

      # if the optimized differences are too small, end up the function and return beta.
      if ((previous_loglik - -optimization$loglik) <= tol)
        return(beta)
    }

    # if the differences calculated from normal calculation or havling steps are too small, end up the
    if (abs(previous_loglik - -optimization$loglik) <= tol)
      return(beta)

    # save the negative loglike value at this step and will be used as previous loglike value at next s
    previous_loglik = -optimization$loglik

    # if the function is not ended up, then the new beta is valid. save it.
    beta = beta_new

    step = step + 1
  }

  # so the loop will be ended up by 2 conditions.
  # 1. the differences calculated by havling steps are too small.
  # 2. the differences calculated by normal optimization are too small.
  return(beta)
}
```

```r
breast_dat = read.csv("./breast-cancer.csv") %>%
  janitor::clean_names() %>%
  dplyr::select(-1, -33) %>% #drop id and NA columns
  mutate(diagnosis = recode(diagnosis, "M" = 1, "B" = 0))

breast_dat <- breast_dat %>% dplyr::select(-area_se, -perimeter_se, -area_worst, -perimeter_mean, -peri
trainRows <- createDataPartition(y = breast_dat$diagnosis, p = 0.8, list = FALSE)
x = breast_dat %>% dplyr::select(-diagnosis) %>% as.matrix()

# make the response variables
y = breast_dat %>%
  dplyr::select(diagnosis) %>%
  as.matrix()
glm.fit <- glm(diagnosis ~ .,
```

```
               data = breast_dat,
               subset = trainRows,
               family = binomial(link = "logit"))
```

Loading the data and run function

```
x = breast_dat %>% dplyr::select(-diagnosis) %>% as.matrix()

# make the response variables
y = breast_dat %>%
  dplyr::select(diagnosis) %>%
  as.matrix()


# calculate beta_hat by newton method 3
beta = newton_optimize(x, y, tol = 0.01)
```

```
## [1] "step: 1   negative loglike loss: 394.400745738609"
```

```
#coefficients of full and lasso models
newton_raphson_beta <- beta %>% as.vector()
coefnames <- rownames(coef(summary(glm.fit)))
cbind(coefnames, newton_raphson_beta) %>% knitr::kable()
```

coefnames	newton_raphson_beta
(Intercept)	0
radius_mean	0
texture_mean	0
smoothness_mean	0
concavity_mean	0
symmetry_mean	0
fractal_dimension_mean	0
radius_se	0
texture_se	0
smoothness_se	0
compactness_se	0
concavity_se	0
concave_points_se	0
symmetry_se	0
fractal_dimension_se	0
smoothness_worst	0
compactness_worst	0
concave_points_worst	0
symmetry_worst	0
fractal_dimension_worst	0

coordinate-wise optimization of a logistic-lasso model

```
x_train <- breast_train[2:20] #predictors
y_train <- breast_train[1] #response
x_train_stan <- cbind(rep(1, nrow(x_train)), scale(x_train))
```

```
x_test <- breast_test[2:20]
y_test <- breast_test[1]
```

```
#soft threshold
sfxn <- function(beta, lambda) {
  if (abs(beta) > lambda) {
    return(sign(beta) * (abs(beta) - lambda))
  }
  else {
    return(0)
  }
}
```

```
#coordinate-wise optimization function
coordwise_lasso <- function(lambda, x, y, betastart, tol = exp(-10), maxiter = 5000) {
  i <- 0
  n <- length(y)
  pnum <- length(betastart)
  betavec <- betastart
  loglik <- 0
  res <- c(0, loglik, betavec)
  prevloglik <- -Inf
  while (i < maxiter & abs(loglik - prevloglik) > tol & loglik < Inf) {
    i <- i + 1
    prevloglik <- loglik
    for (j in 1:pnum) {
      theta <- x %*% betavec
      p <- exp(theta) / (1 + exp(theta)) #probability of malignant cases
      w <- p*(1-p) #working weights
      w <- ifelse(abs(w-0) < 1e-5, 1e-5, w)
      z <- theta + (y - p)/w #working response
      zwoj <- x[, -j] %*% betavec[-j]
      betavec[j] <- sfxn(sum(w*(x[,j])*(z - zwoj)), lambda) / (sum(w*x[,j]*x[,j]))
    }
    theta <- x %*% betavec
    p <- exp(theta) / (1 + exp(theta)) #probability of malignant cases
    w <- p*(1-p) #working weights
    w <- ifelse(abs(w-0) < 1e-10, 1e-10, w)
    z <- theta + (y - p)/w
    loglik <- sum(w*(z - theta)^2) / (2*n) + lambda * sum(abs(betavec))
    res <- rbind(res, c(i, loglik, betavec))
  }
  return(res)
}
#coordwise_res <- coordwise_lasso(lambda = 0.006, x_train_stan, y_train, betastart = rep(0, #20))
#coordwise_res[nrow(coordwise_res), ]
```

We need to calculate lambdamax first to define a sequence of lambda.

```
x.matrix <- scale(x_train) %>% as.matrix()
y.matrix <- as.matrix(y_train)
lambdamax <- max(abs(t(x.matrix) %*% y.matrix)) #/ nrow(y.matrix)
lambda_seq1 <- exp(seq(log(lambdamax), -5, length = 50))
lambda_seq2 <- exp(seq(log(lambdamax), -5, length = 50))
```

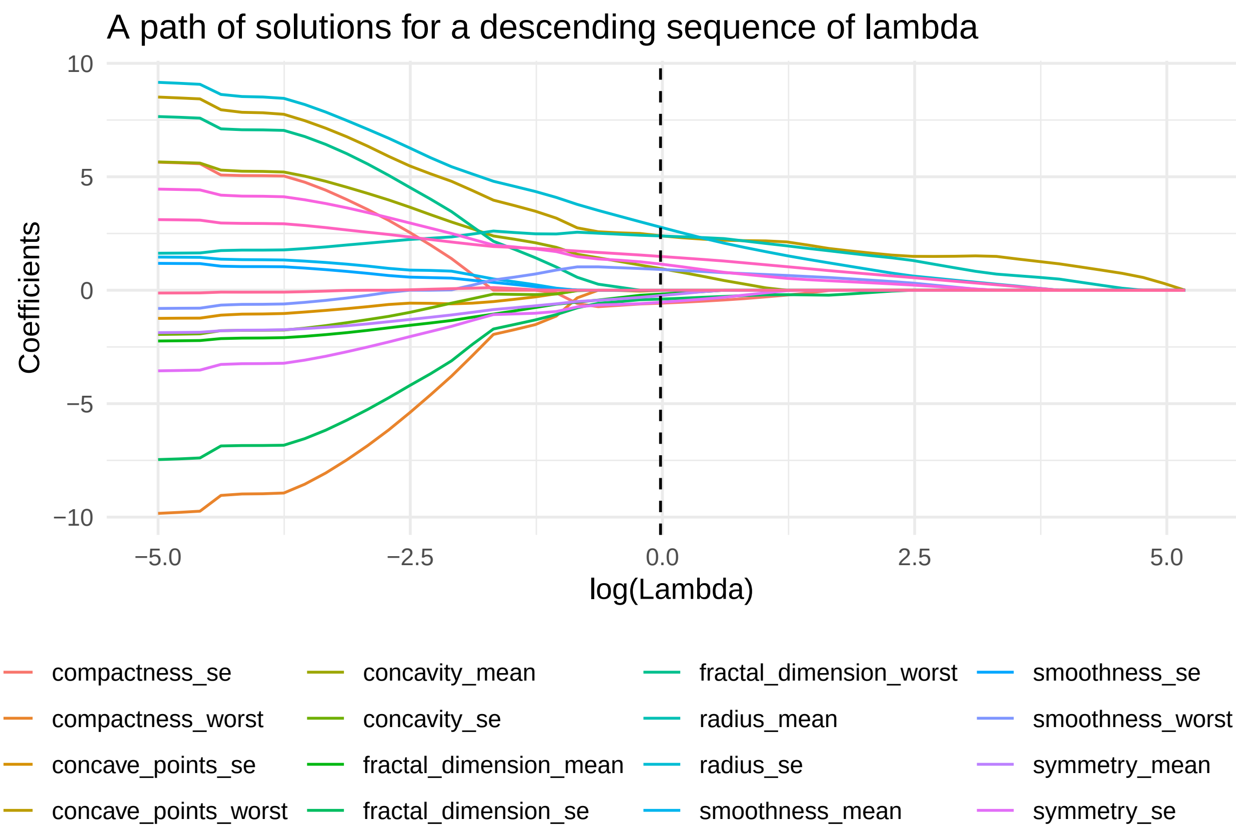
```r
#a path of solutions
pathwise <- function(x, y, lambda) {
  n <- length(lambda)
  betastart <- rep(0, 20)
  betas <- NULL
  for (i in 1:n) {
    coordwise_res <- coordwise_lasso(lambda = lambda[i],
                                     x = x,
                                     y = y,
                                     betastart = betastart)
    curbeta <- coordwise_res[nrow(coordwise_res), 3:22]
    betastart <- curbeta
    betas <- rbind(betas, c(curbeta))
  }
  return(data.frame(cbind(lambda, betas)))
}
pathwise_sol <- pathwise(x_train_stan, y_train, lambda_seq2)
round(pathwise_sol, 2) %>% knitr::kable()
```

lambda	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21
177.83	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
144.47	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.30	0.00	0.00
117.37	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.56	0.00	0.00
95.35	0.00	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.74	0.00	0.00
77.46	0.00	0.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.89	0.00	0.00
62.93	0.00	0.35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.03	0.00	0.00
51.12	0.00	0.49	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.16	0.00	0.00
41.53	0.00	0.57	0.08	0.00	0.00	0.00	0.00	0.09	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.27	0.00	0.00
33.74	-0.05	0.64	0.16	0.00	0.00	0.00	0.00	0.18	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.38	0.00	0.00
27.41	-0.11	0.71	0.24	0.00	0.00	0.00	0.00	0.27	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.49	0.00	0.00
22.27	-0.17	0.83	0.32	0.00	0.00	0.00	0.00	0.35	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	1.51	0.05	0.00
18.09	-0.22	0.99	0.40	0.00	0.00	0.00	0.00	0.44	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.00	1.50	0.11	0.00
14.70	-0.25	1.15	0.48	0.00	0.00	0.00	0.00	0.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.23	0.00	1.49	0.17	0.00
11.94	-0.29	1.31	0.55	0.00	0.00	0.00	0.00	0.63	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.31	0.00	1.49	0.23	0.00
9.70	-0.32	1.43	0.63	0.00	0.00	0.00	0.00	0.76	0.00	0.00	0.00	0.00	0.00	0.00	-0.04	0.38	0.00	1.55	0.28	0.00
7.88	-0.34	1.53	0.71	0.00	0.00	0.00	0.00	0.91	0.00	0.00	0.00	0.00	0.00	0.00	-0.10	0.44	0.00	1.64	0.33	0.00
6.40	-0.35	1.63	0.79	0.00	0.00	0.00	0.00	1.06	0.00	0.00	0.00	0.00	0.00	0.00	-0.16	0.50	0.00	1.73	0.37	0.00
5.20	-0.37	1.74	0.86	0.00	0.00	0.00	0.00	1.21	0.00	0.00	-0.01	0.00	0.00	0.00	-0.22	0.56	0.00	1.84	0.42	0.00
4.22	-0.39	1.84	0.95	0.00	0.00	0.00	0.00	1.36	0.00	0.00	-0.12	0.00	0.00	0.00	-0.21	0.60	0.00	1.99	0.46	0.00
3.43	-0.40	1.96	1.04	0.00	0.00	0.00	0.00	1.52	0.00	0.00	-0.21	0.00	0.00	-0.02	-0.20	0.65	0.00	2.13	0.52	0.00

lambda	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21
2.79	-0.41	2.06	1.12	0.00	0.10	0.00	0.00	1.70	0.00	0.00	-0.29	0.00	0.00	-0.11	-0.20	0.69	0.00	2.17	0.61	0.00
2.27	-0.41	2.17	1.20	0.00	0.27	0.00	0.00	1.88	0.00	0.00	-0.37	0.00	0.00	-0.21	-0.23	0.73	0.00	2.18	0.70	0.00
1.84	-0.41	2.28	1.29	0.00	0.43	0.00	0.00	2.07	0.00	0.00	-0.43	0.00	0.00	-0.31	-0.28	0.77	0.00	2.20	0.79	0.00
1.49	-0.40	2.32	1.36	0.00	0.61	-0.06	-0.06	2.29	0.00	0.00	-0.48	0.00	0.00	-0.39	-0.31	0.82	0.00	2.25	0.91	0.00
1.21	-0.39	2.36	1.42	0.00	0.79	-0.15	-0.12	2.53	0.00	0.00	-0.53	0.00	0.00	-0.46	-0.35	0.87	0.00	2.31	1.03	0.00
0.99	-0.38	2.39	1.49	0.00	0.95	-0.24	-0.17	2.77	0.00	0.00	-0.57	0.00	-0.01	-0.53	-0.39	0.91	0.00	2.39	1.15	0.00
0.80	-0.36	2.42	1.55	0.00	1.11	-0.32	-0.22	3.02	0.00	0.00	-0.61	0.00	-0.04	-0.59	-0.42	0.96	0.00	2.50	1.25	0.00
0.65	-0.33	2.47	1.61	0.00	1.27	-0.38	-0.33	3.27	0.00	0.00	-0.67	0.00	-0.01	-0.63	-0.50	0.99	0.00	2.53	1.32	0.13
0.53	-0.30	2.51	1.66	0.00	1.43	-0.44	-0.44	3.52	0.00	0.00	-0.73	0.00	0.00	-0.65	-0.58	1.03	0.00	2.58	1.38	0.27
0.43	-0.25	2.56	1.72	0.00	1.59	-0.50	-0.52	3.78	0.00	0.01	-0.59	-0.02	-0.01	-0.74	-0.78	1.03	-0.34	2.75	1.48	0.57
0.35	-0.17	2.48	1.79	0.09	1.89	-0.60	-0.62	4.09	0.00	0.05	-0.12	-0.17	-0.16	-0.94	-1.08	0.89	-1.14	3.18	1.70	1.02
0.28	-0.11	2.49	1.84	0.24	2.09	-0.69	-0.76	4.35	0.02	0.15	0.00	-0.19	-0.29	-1.01	-1.31	0.72	-1.52	3.48	1.82	1.43
0.23	-0.07	2.54	1.88	0.37	2.25	-0.77	-0.91	4.58	0.07	0.25	0.00	-0.18	-0.40	-1.05	-1.51	0.58	-1.74	3.73	1.91	1.80
0.19	-0.02	2.61	1.93	0.50	2.39	-0.85	-1.05	4.80	0.12	0.35	0.00	-0.17	-0.50	-1.07	-1.71	0.45	-1.95	3.97	1.99	2.16
0.15	0.00	2.48	2.02	0.67	2.71	-0.97	-1.19	5.12	0.09	0.44	0.69	-0.37	-0.57	-1.33	-2.38	0.22	-2.88	4.39	2.24	2.81
0.12	0.00	2.35	2.12	0.84	3.01	-1.09	-1.34	5.44	0.07	0.53	1.39	-0.57	-0.60	-1.59	-3.11	0.02	-3.79	4.80	2.50	3.46
0.10	0.05	2.29	2.23	0.87	3.33	-1.19	-1.44	5.84	0.04	0.56	1.99	-0.78	-0.58	-1.82	-3.68	0.00	-4.61	5.13	2.73	4.01
0.08	0.13	2.24	2.35	0.89	3.66	-1.29	-1.55	6.27	0.02	0.58	2.55	-0.98	-0.57	-2.05	-4.20	0.00	-5.40	5.48	2.96	4.54
0.07	0.19	2.16	2.46	0.96	3.98	-1.39	-1.66	6.70	0.00	0.65	3.08	-1.15	-0.63	-2.28	-4.74	-0.09	-6.16	5.90	3.20	5.06
0.05	0.24	2.07	2.56	1.06	4.27	-1.48	-1.77	7.09	0.00	0.74	3.56	-1.30	-0.73	-2.50	-5.26	-0.23	-6.85	6.35	3.42	5.56
0.04	0.29	1.99	2.65	1.15	4.54	-1.57	-1.87	7.48	-0.01	0.83	4.00	-1.43	-0.81	-2.71	-5.73	-0.34	-7.48	6.76	3.63	6.02
0.04	0.34	1.91	2.76	1.22	4.80	-1.63	-1.96	7.85	-0.04	0.91	4.41	-1.56	-0.89	-2.91	-6.17	-0.45	-8.06	7.14	3.82	6.43
0.03	0.38	1.84	2.85	1.28	5.03	-1.69	-2.03	8.18	-0.07	0.98	4.76	-1.67	-0.96	-3.08	-6.54	-0.53	-8.55	7.47	3.98	6.77
0.02	0.42	1.78	2.93	1.33	5.21	-1.74	-2.09	8.46	-0.08	1.03	5.03	-1.76	-1.03	-3.22	-6.83	-0.61	-8.94	7.75	4.12	7.04
0.02	0.42	1.77	2.94	1.34	5.24	-1.76	-2.11	8.52	-0.08	1.04	5.05	-1.77	-1.05	-3.23	-6.84	-0.62	-8.97	7.82	4.14	7.07
0.02	0.43	1.77	2.95	1.35	5.24	-1.76	-2.11	8.54	-0.09	1.04	5.05	-1.77	-1.05	-3.24	-6.85	-0.62	-8.98	7.84	4.15	7.07

lambda	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20	V21
0.01	0.44	1.75	2.96	1.37	5.29	-1.79	-2.13	8.63	-0.08	1.06	5.08	-1.78	-1.10	-3.27	-6.86	-0.66	-9.04	7.95	4.19	7.11
0.01	0.50	1.64	3.09	1.45	5.60	-1.85	-2.22	9.07	-0.12	1.17	5.57	-1.92	-1.23	-3.52	-7.39	-0.79	-9.74	8.43	4.42	7.58
0.01	0.50	1.64	3.10	1.45	5.63	-1.86	-2.23	9.12	-0.12	1.18	5.61	-1.94	-1.23	-3.54	-7.43	-0.80	-9.79	8.47	4.44	7.62
0.01	0.51	1.63	3.11	1.46	5.65	-1.87	-2.24	9.16	-0.12	1.18	5.64	-1.95	-1.24	-3.55	-7.46	-0.80	-9.83	8.51	4.46	7.65

```
colnames(pathwise_sol) <- c("lambda", rownames(coef(summary(glm.fit))))
pathwise_sol %>%
  pivot_longer(
    3:21,
    names_to = "variables",
    values_to = "coefficients") %>%
  ggplot(aes(x = log(lambda), y = coefficients, group = variables, color = variables)) +
  geom_line() +
  geom_vline(xintercept = log(0.981), linetype = 2) +
  ggtitle("A path of solutions for a descending sequence of lambda") +
  xlab("log(Lambda)") +
  ylab("Coefficients")
```



cross-validation

```r
set.seed(2022)
cv = function(data, lambda) {
  n <- nrow(data)
  data <- data[sample(n), ] #shuffle the data
  folds <- cut(seq(1, nrow(data)), breaks = 5, labels = FALSE) #Create 5 equal size folds
 # mse <- data.frame() #a data frame storing mse results
  #mse_lambda <- vector()
  #se <- vector() #a vector storing test errors
  res <- lambda
  #se <- vector() #a vectro storing test errors

    #Perform 5 fold cross validation
  for (i in 1:5) {
    #partition the data into train and test data
    testRows <- which(folds == i, arr.ind = TRUE)
    data_test <- data[testRows, ]
    data_train <- data[-testRows, ]
    x_train <- data_train[2:20]
    x_train_stan <- cbind(rep(1, nrow(x_train)), scale(x_train))
    y_train <- data_train[1]
    x_test <- data_test[2:20]
    #standardized test data
    x_test_stan <- cbind(rep(1, nrow(x_test)), scale(x_test))
    y_test <- data_test %>% mutate(diagnosis = factor(diagnosis))
    y_test <- y_test$diagnosis
    #Use the test and train data partitions to perform lasso
    path_sol <- pathwise(x = x_train_stan,
                         y = y_train,
                         lambda = lambda)
    auc <- vector()
    for (j in 1:length(lambda)) {
      curbeta <- as.numeric(path_sol[j, 2:21])
      theta <- x_test_stan %*% curbeta
      p <- exp(theta) / (1 + exp(theta))
      auc[j] <- auc(y_test, p)
      #y.pred <- ifelse(p > 0.5, 1, 0)
      #accuracy[j] <- mean(y.pred == y_test)
    }
    print(auc)
    res <- cbind(res, auc)
    print(res)
  }
  return(res)
    #se[j] <- sqrt(var(error)/5)
  #cv.auc.lambda <- rowMeans(mse)
  #return(cv.auc.lambda)
}
cv_test = cv(data = breast_train, lambda_seq2)
```

```
##  [1] 0.5000000 0.5000000 0.9475962 0.9475962 0.9533654 0.9552885 0.9576923
##  [8] 0.9567308 0.9634615 0.9692308 0.9711538 0.9750000 0.9774038 0.9798077
## [15] 0.9812500 0.9826923 0.9831731 0.9841346 0.9841346 0.9841346 0.9846154
```

```
## [22] 0.9850962 0.9860577 0.9860577 0.9860577 0.9865385 0.9855769 0.9841346
## [29] 0.9822115 0.9802885 0.9764423 0.9754808 0.9730769 0.9716346 0.9701923
## [36] 0.9692308 0.9682692 0.9668269 0.9653846 0.9649038 0.9629808 0.9625000
## [43] 0.9620192 0.9615385 0.9610577 0.9610577 0.9610577 0.9610577 0.9605769
## [50] 0.9605769
##                res       auc
##  [1,] 1.778334e+02 0.5000000
##  [2,] 1.444705e+02 0.5000000
##  [3,] 1.173666e+02 0.9475962
##  [4,] 9.534770e+01 0.9475962
##  [5,] 7.745970e+01 0.9533654
##  [6,] 6.292764e+01 0.9552885
##  [7,] 5.112190e+01 0.9576923
##  [8,] 4.153102e+01 0.9567308
##  [9,] 3.373947e+01 0.9634615
## [10,] 2.740967e+01 0.9692308
## [11,] 2.226739e+01 0.9711538
## [12,] 1.808985e+01 0.9750000
## [13,] 1.469605e+01 0.9774038
## [14,] 1.193895e+01 0.9798077
## [15,] 9.699107e+00 0.9812500
## [16,] 7.879477e+00 0.9826923
## [17,] 6.401223e+00 0.9831731
## [18,] 5.200302e+00 0.9841346
## [19,] 4.224683e+00 0.9841346
## [20,] 3.432099e+00 0.9841346
## [21,] 2.788209e+00 0.9846154
## [22,] 2.265119e+00 0.9850962
## [23,] 1.840164e+00 0.9860577
## [24,] 1.494934e+00 0.9860577
## [25,] 1.214473e+00 0.9860577
## [26,] 9.866277e-01 0.9865385
## [27,] 8.015284e-01 0.9855769
## [28,] 6.511552e-01 0.9841346
## [29,] 5.289932e-01 0.9822115
## [30,] 4.297498e-01 0.9802885
## [31,] 3.491253e-01 0.9764423
## [32,] 2.836266e-01 0.9754808
## [33,] 2.304159e-01 0.9730769
## [34,] 1.871880e-01 0.9716346
## [35,] 1.520700e-01 0.9701923
## [36,] 1.235405e-01 0.9692308
## [37,] 1.003633e-01 0.9682692
## [38,] 8.153432e-02 0.9668269
## [39,] 6.623782e-02 0.9653846
## [40,] 5.381107e-02 0.9649038
## [41,] 4.371568e-02 0.9629808
## [42,] 3.551427e-02 0.9625000
## [43,] 2.885150e-02 0.9620192
## [44,] 2.343873e-02 0.9615385
## [45,] 1.904143e-02 0.9610577
## [46,] 1.546911e-02 0.9610577
## [47,] 1.256698e-02 0.9610577
## [48,] 1.020931e-02 0.9610577
```

```
## [49,] 8.293961e-03 0.9605769
## [50,] 6.737947e-03 0.9605769
##  [1] 0.5000000 0.5000000 0.9623656 0.9623656 0.9618280 0.9655914 0.9677419
##  [8] 0.9698925 0.9725806 0.9763441 0.9817204 0.9844086 0.9892473 0.9930108
## [15] 0.9940860 0.9967742 0.9967742 0.9978495 0.9983871 0.9983871 0.9983871
## [22] 0.9989247 0.9989247 0.9989247 0.9989247 0.9989247 0.9994624 0.9989247
## [29] 0.9989247 0.9994624 0.9994624 0.9994624 0.9994624 0.9994624 0.9994624
## [36] 0.9994624 0.9994624 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## [43] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## [50] 1.0000000
##                 res       auc       auc
##  [1,] 1.778334e+02 0.5000000 0.5000000
##  [2,] 1.444705e+02 0.5000000 0.5000000
##  [3,] 1.173666e+02 0.9475962 0.9623656
##  [4,] 9.534770e+01 0.9475962 0.9623656
##  [5,] 7.745970e+01 0.9533654 0.9618280
##  [6,] 6.292764e+01 0.9552885 0.9655914
##  [7,] 5.112190e+01 0.9576923 0.9677419
##  [8,] 4.153102e+01 0.9567308 0.9698925
##  [9,] 3.373947e+01 0.9634615 0.9725806
## [10,] 2.740967e+01 0.9692308 0.9763441
## [11,] 2.226739e+01 0.9711538 0.9817204
## [12,] 1.808985e+01 0.9750000 0.9844086
## [13,] 1.469605e+01 0.9774038 0.9892473
## [14,] 1.193895e+01 0.9798077 0.9930108
## [15,] 9.699107e+00 0.9812500 0.9940860
## [16,] 7.879477e+00 0.9826923 0.9967742
## [17,] 6.401223e+00 0.9831731 0.9967742
## [18,] 5.200302e+00 0.9841346 0.9978495
## [19,] 4.224683e+00 0.9841346 0.9983871
## [20,] 3.432099e+00 0.9841346 0.9983871
## [21,] 2.788209e+00 0.9846154 0.9983871
## [22,] 2.265119e+00 0.9850962 0.9989247
## [23,] 1.840164e+00 0.9860577 0.9989247
## [24,] 1.494934e+00 0.9860577 0.9989247
## [25,] 1.214473e+00 0.9860577 0.9989247
## [26,] 9.866277e-01 0.9865385 0.9989247
## [27,] 8.015284e-01 0.9855769 0.9994624
## [28,] 6.511552e-01 0.9841346 0.9989247
## [29,] 5.289932e-01 0.9822115 0.9989247
## [30,] 4.297498e-01 0.9802885 0.9994624
## [31,] 3.491253e-01 0.9764423 0.9994624
## [32,] 2.836266e-01 0.9754808 0.9994624
## [33,] 2.304159e-01 0.9730769 0.9994624
## [34,] 1.871880e-01 0.9716346 0.9994624
## [35,] 1.520700e-01 0.9701923 0.9994624
## [36,] 1.235405e-01 0.9692308 0.9994624
## [37,] 1.003633e-01 0.9682692 0.9994624
## [38,] 8.153432e-02 0.9668269 1.0000000
## [39,] 6.623782e-02 0.9653846 1.0000000
## [40,] 5.381107e-02 0.9649038 1.0000000
## [41,] 4.371568e-02 0.9629808 1.0000000
## [42,] 3.551427e-02 0.9625000 1.0000000
## [43,] 2.885150e-02 0.9620192 1.0000000
```

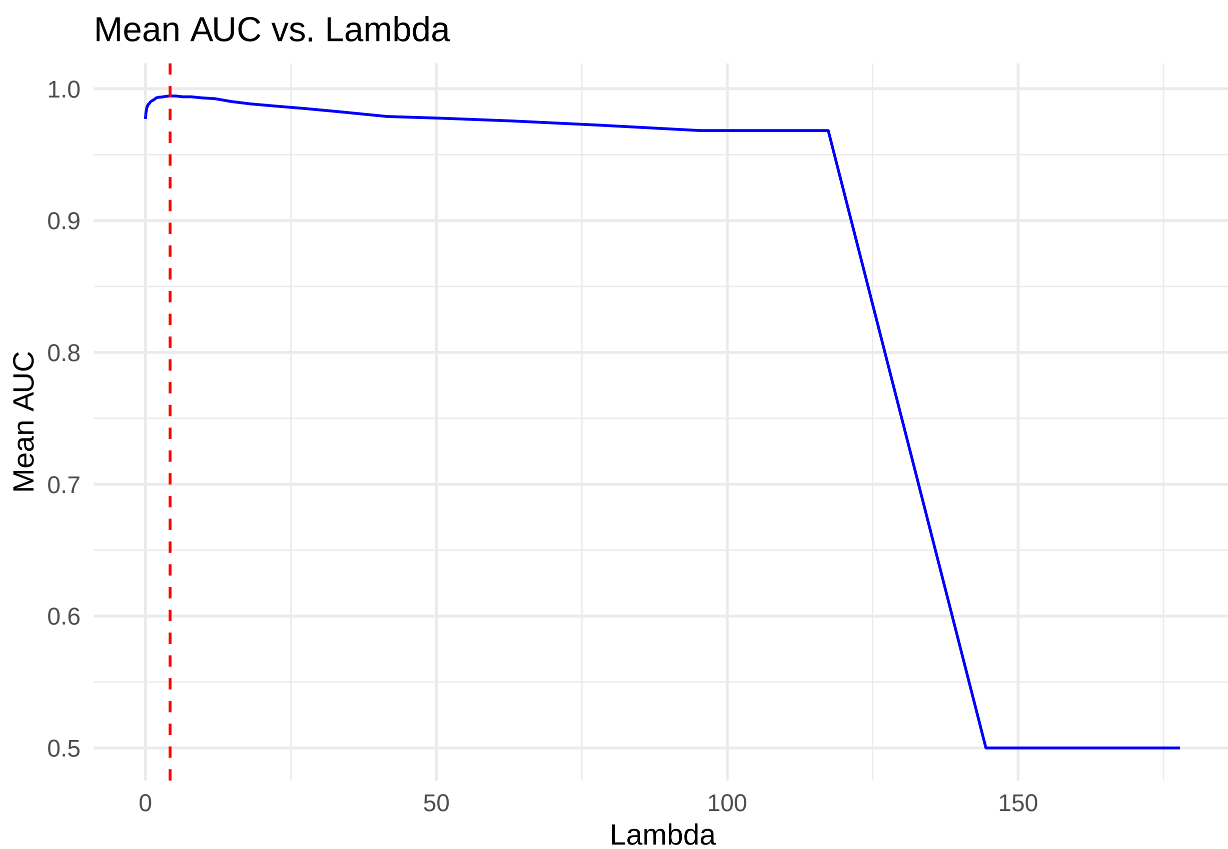
```
## [44,] 2.343873e-02 0.9615385 1.0000000
## [45,] 1.904143e-02 0.9610577 1.0000000
## [46,] 1.546911e-02 0.9610577 1.0000000
## [47,] 1.256698e-02 0.9610577 1.0000000
## [48,] 1.020931e-02 0.9610577 1.0000000
## [49,] 8.293961e-03 0.9605769 1.0000000
## [50,] 6.737947e-03 0.9605769 1.0000000
##  [1] 0.5000000 0.5000000 0.9764765 0.9764765 0.9849850 0.9869870 0.9884885
##  [8] 0.9899900 0.9909910 0.9909910 0.9899900 0.9894895 0.9909910 0.9949950
## [15] 0.9964965 0.9974975 0.9974975 0.9979980 0.9979980 0.9979980 0.9979980
## [22] 0.9984985 0.9984985 0.9984985 0.9989990 0.9989990 0.9989990 0.9984985
## [29] 0.9984985 0.9984985 0.9979980 0.9979980 0.9979980 0.9959960 0.9934935
## [36] 0.9934935 0.9909910 0.9904905 0.9904905 0.9834835 0.9834835 0.9839840
## [43] 0.9839840 0.9799800 0.9799800 0.9799800 0.9799800 0.9789790 0.9789790
## [50] 0.9789790
##                 res       auc       auc       auc
##  [1,] 1.778334e+02 0.5000000 0.5000000 0.5000000
##  [2,] 1.444705e+02 0.5000000 0.5000000 0.5000000
##  [3,] 1.173666e+02 0.9475962 0.9623656 0.9764765
##  [4,] 9.534770e+01 0.9475962 0.9623656 0.9764765
##  [5,] 7.745970e+01 0.9533654 0.9618280 0.9849850
##  [6,] 6.292764e+01 0.9552885 0.9655914 0.9869870
##  [7,] 5.112190e+01 0.9576923 0.9677419 0.9884885
##  [8,] 4.153102e+01 0.9567308 0.9698925 0.9899900
##  [9,] 3.373947e+01 0.9634615 0.9725806 0.9909910
## [10,] 2.740967e+01 0.9692308 0.9763441 0.9909910
## [11,] 2.226739e+01 0.9711538 0.9817204 0.9899900
## [12,] 1.808985e+01 0.9750000 0.9844086 0.9894895
## [13,] 1.469605e+01 0.9774038 0.9892473 0.9909910
## [14,] 1.193895e+01 0.9798077 0.9930108 0.9949950
## [15,] 9.699107e+00 0.9812500 0.9940860 0.9964965
## [16,] 7.879477e+00 0.9826923 0.9967742 0.9974975
## [17,] 6.401223e+00 0.9831731 0.9967742 0.9974975
## [18,] 5.200302e+00 0.9841346 0.9978495 0.9979980
## [19,] 4.224683e+00 0.9841346 0.9983871 0.9979980
## [20,] 3.432099e+00 0.9841346 0.9983871 0.9979980
## [21,] 2.788209e+00 0.9846154 0.9983871 0.9979980
## [22,] 2.265119e+00 0.9850962 0.9989247 0.9984985
## [23,] 1.840164e+00 0.9860577 0.9989247 0.9984985
## [24,] 1.494934e+00 0.9860577 0.9989247 0.9984985
## [25,] 1.214473e+00 0.9860577 0.9989247 0.9989990
## [26,] 9.866277e-01 0.9865385 0.9989247 0.9989990
## [27,] 8.015284e-01 0.9855769 0.9994624 0.9989990
## [28,] 6.511552e-01 0.9841346 0.9989247 0.9984985
## [29,] 5.289932e-01 0.9822115 0.9989247 0.9984985
## [30,] 4.297498e-01 0.9802885 0.9994624 0.9984985
## [31,] 3.491253e-01 0.9764423 0.9994624 0.9979980
## [32,] 2.836266e-01 0.9754808 0.9994624 0.9979980
## [33,] 2.304159e-01 0.9730769 0.9994624 0.9979980
## [34,] 1.871880e-01 0.9716346 0.9994624 0.9959960
## [35,] 1.520700e-01 0.9701923 0.9994624 0.9934935
## [36,] 1.235405e-01 0.9692308 0.9994624 0.9934935
## [37,] 1.003633e-01 0.9682692 0.9994624 0.9909910
## [38,] 8.153432e-02 0.9668269 1.0000000 0.9904905
```

```
## [39,] 6.623782e-02 0.9653846 1.0000000 0.9904905
## [40,] 5.381107e-02 0.9649038 1.0000000 0.9834835
## [41,] 4.371568e-02 0.9629808 1.0000000 0.9834835
## [42,] 3.551427e-02 0.9625000 1.0000000 0.9839840
## [43,] 2.885150e-02 0.9620192 1.0000000 0.9839840
## [44,] 2.343873e-02 0.9615385 1.0000000 0.9799800
## [45,] 1.904143e-02 0.9610577 1.0000000 0.9799800
## [46,] 1.546911e-02 0.9610577 1.0000000 0.9799800
## [47,] 1.256698e-02 0.9610577 1.0000000 0.9799800
## [48,] 1.020931e-02 0.9610577 1.0000000 0.9789790
## [49,] 8.293961e-03 0.9605769 1.0000000 0.9789790
## [50,] 6.737947e-03 0.9605769 1.0000000 0.9789790
##  [1] 0.5000000 0.5000000 0.9765306 0.9765306 0.9795918 0.9821429 0.9846939
##  [8] 0.9872449 0.9918367 0.9954082 0.9979592 0.9994898 1.0000000 1.0000000
## [15] 0.9994898 0.9994898 0.9994898 0.9994898 0.9989796 0.9969388 0.9943878
## [22] 0.9923469 0.9887755 0.9826531 0.9785714 0.9775510 0.9750000 0.9724490
## [29] 0.9719388 0.9719388 0.9714286 0.9714286 0.9709184 0.9704082 0.9704082
## [36] 0.9704082 0.9709184 0.9709184 0.9704082 0.9709184 0.9709184 0.9704082
## [43] 0.9704082 0.9704082 0.9709184 0.9709184 0.9709184 0.9704082 0.9704082
## [50] 0.9704082
##                res       auc       auc       auc       auc
##  [1,] 1.778334e+02 0.5000000 0.5000000 0.5000000 0.5000000
##  [2,] 1.444705e+02 0.5000000 0.5000000 0.5000000 0.5000000
##  [3,] 1.173666e+02 0.9475962 0.9623656 0.9764765 0.9765306
##  [4,] 9.534770e+01 0.9475962 0.9623656 0.9764765 0.9765306
##  [5,] 7.745970e+01 0.9533654 0.9618280 0.9849850 0.9795918
##  [6,] 6.292764e+01 0.9552885 0.9655914 0.9869870 0.9821429
##  [7,] 5.112190e+01 0.9576923 0.9677419 0.9884885 0.9846939
##  [8,] 4.153102e+01 0.9567308 0.9698925 0.9899900 0.9872449
##  [9,] 3.373947e+01 0.9634615 0.9725806 0.9909910 0.9918367
## [10,] 2.740967e+01 0.9692308 0.9763441 0.9909910 0.9954082
## [11,] 2.226739e+01 0.9711538 0.9817204 0.9899900 0.9979592
## [12,] 1.808985e+01 0.9750000 0.9844086 0.9894895 0.9994898
## [13,] 1.469605e+01 0.9774038 0.9892473 0.9909910 1.0000000
## [14,] 1.193895e+01 0.9798077 0.9930108 0.9949950 1.0000000
## [15,] 9.699107e+00 0.9812500 0.9940860 0.9964965 0.9994898
## [16,] 7.879477e+00 0.9826923 0.9967742 0.9974975 0.9994898
## [17,] 6.401223e+00 0.9831731 0.9967742 0.9974975 0.9994898
## [18,] 5.200302e+00 0.9841346 0.9978495 0.9979980 0.9994898
## [19,] 4.224683e+00 0.9841346 0.9983871 0.9979980 0.9989796
## [20,] 3.432099e+00 0.9841346 0.9983871 0.9979980 0.9969388
## [21,] 2.788209e+00 0.9846154 0.9983871 0.9979980 0.9943878
## [22,] 2.265119e+00 0.9850962 0.9989247 0.9984985 0.9923469
## [23,] 1.840164e+00 0.9860577 0.9989247 0.9984985 0.9887755
## [24,] 1.494934e+00 0.9860577 0.9989247 0.9984985 0.9826531
## [25,] 1.214473e+00 0.9860577 0.9989247 0.9989990 0.9785714
## [26,] 9.866277e-01 0.9865385 0.9989247 0.9989990 0.9775510
## [27,] 8.015284e-01 0.9855769 0.9994624 0.9989990 0.9750000
## [28,] 6.511552e-01 0.9841346 0.9989247 0.9984985 0.9724490
## [29,] 5.289932e-01 0.9822115 0.9989247 0.9984985 0.9719388
## [30,] 4.297498e-01 0.9802885 0.9994624 0.9984985 0.9719388
## [31,] 3.491253e-01 0.9764423 0.9994624 0.9979980 0.9714286
## [32,] 2.836266e-01 0.9754808 0.9994624 0.9979980 0.9714286
## [33,] 2.304159e-01 0.9730769 0.9994624 0.9979980 0.9709184
```

```
## [34,] 1.871880e-01 0.9716346 0.9994624 0.9959960 0.9704082
## [35,] 1.520700e-01 0.9701923 0.9994624 0.9934935 0.9704082
## [36,] 1.235405e-01 0.9692308 0.9994624 0.9934935 0.9704082
## [37,] 1.003633e-01 0.9682692 0.9994624 0.9909910 0.9709184
## [38,] 8.153432e-02 0.9668269 1.0000000 0.9904905 0.9709184
## [39,] 6.623782e-02 0.9653846 1.0000000 0.9904905 0.9704082
## [40,] 5.381107e-02 0.9649038 1.0000000 0.9834835 0.9709184
## [41,] 4.371568e-02 0.9629808 1.0000000 0.9834835 0.9709184
## [42,] 3.551427e-02 0.9625000 1.0000000 0.9839840 0.9704082
## [43,] 2.885150e-02 0.9620192 1.0000000 0.9839840 0.9704082
## [44,] 2.343873e-02 0.9615385 1.0000000 0.9799800 0.9704082
## [45,] 1.904143e-02 0.9610577 1.0000000 0.9799800 0.9709184
## [46,] 1.546911e-02 0.9610577 1.0000000 0.9799800 0.9709184
## [47,] 1.256698e-02 0.9610577 1.0000000 0.9799800 0.9709184
## [48,] 1.020931e-02 0.9610577 1.0000000 0.9789790 0.9704082
## [49,] 8.293961e-03 0.9605769 1.0000000 0.9789790 0.9704082
## [50,] 6.737947e-03 0.9605769 1.0000000 0.9789790 0.9704082
##  [1] 0.5000000 0.5000000 0.9783163 0.9783163 0.9826531 0.9877551 0.9892857
##  [8] 0.9908163 0.9928571 0.9928571 0.9933673 0.9938776 0.9938776 0.9943878
## [15] 0.9938776 0.9928571 0.9923469 0.9928571 0.9928571 0.9933673 0.9928571
## [22] 0.9928571 0.9928571 0.9928571 0.9928571 0.9903061 0.9897959 0.9897959
## [29] 0.9892857 0.9892857 0.9877551 0.9882653 0.9867347 0.9857143 0.9841837
## [36] 0.9836735 0.9821429 0.9821429 0.9821429 0.9821429 0.9785714 0.9785714
## [43] 0.9785714 0.9785714 0.9785714 0.9785714 0.9785714 0.9755102 0.9755102
## [50] 0.9755102
##                res       auc       auc       auc       auc       auc
##  [1,] 1.778334e+02 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [2,] 1.444705e+02 0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
##  [3,] 1.173666e+02 0.9475962 0.9623656 0.9764765 0.9765306 0.9783163
##  [4,] 9.534770e+01 0.9475962 0.9623656 0.9764765 0.9765306 0.9783163
##  [5,] 7.745970e+01 0.9533654 0.9618280 0.9849850 0.9795918 0.9826531
##  [6,] 6.292764e+01 0.9552885 0.9655914 0.9869870 0.9821429 0.9877551
##  [7,] 5.112190e+01 0.9576923 0.9677419 0.9884885 0.9846939 0.9892857
##  [8,] 4.153102e+01 0.9567308 0.9698925 0.9899900 0.9872449 0.9908163
##  [9,] 3.373947e+01 0.9634615 0.9725806 0.9909910 0.9918367 0.9928571
## [10,] 2.740967e+01 0.9692308 0.9763441 0.9909910 0.9954082 0.9928571
## [11,] 2.226739e+01 0.9711538 0.9817204 0.9899900 0.9979592 0.9933673
## [12,] 1.808985e+01 0.9750000 0.9844086 0.9894895 0.9994898 0.9938776
## [13,] 1.469605e+01 0.9774038 0.9892473 0.9909910 1.0000000 0.9938776
## [14,] 1.193895e+01 0.9798077 0.9930108 0.9949950 1.0000000 0.9943878
## [15,] 9.699107e+00 0.9812500 0.9940860 0.9964965 0.9994898 0.9938776
## [16,] 7.879477e+00 0.9826923 0.9967742 0.9974975 0.9994898 0.9928571
## [17,] 6.401223e+00 0.9831731 0.9967742 0.9974975 0.9994898 0.9923469
## [18,] 5.200302e+00 0.9841346 0.9978495 0.9979980 0.9994898 0.9928571
## [19,] 4.224683e+00 0.9841346 0.9983871 0.9979980 0.9989796 0.9928571
## [20,] 3.432099e+00 0.9841346 0.9983871 0.9979980 0.9969388 0.9933673
## [21,] 2.788209e+00 0.9846154 0.9983871 0.9979980 0.9943878 0.9928571
## [22,] 2.265119e+00 0.9850962 0.9989247 0.9984985 0.9923469 0.9928571
## [23,] 1.840164e+00 0.9860577 0.9989247 0.9984985 0.9887755 0.9928571
## [24,] 1.494934e+00 0.9860577 0.9989247 0.9984985 0.9826531 0.9928571
## [25,] 1.214473e+00 0.9860577 0.9989247 0.9989990 0.9785714 0.9928571
## [26,] 9.866277e-01 0.9865385 0.9989247 0.9989990 0.9775510 0.9903061
## [27,] 8.015284e-01 0.9855769 0.9994624 0.9989990 0.9750000 0.9897959
## [28,] 6.511552e-01 0.9841346 0.9989247 0.9984985 0.9724490 0.9897959
```

```
## [29,] 5.289932e-01 0.9822115 0.9989247 0.9984985 0.9719388 0.9892857
## [30,] 4.297498e-01 0.9802885 0.9994624 0.9984985 0.9719388 0.9892857
## [31,] 3.491253e-01 0.9764423 0.9994624 0.9979980 0.9714286 0.9877551
## [32,] 2.836266e-01 0.9754808 0.9994624 0.9979980 0.9714286 0.9882653
## [33,] 2.304159e-01 0.9730769 0.9994624 0.9979980 0.9709184 0.9867347
## [34,] 1.871880e-01 0.9716346 0.9994624 0.9959960 0.9704082 0.9857143
## [35,] 1.520700e-01 0.9701923 0.9994624 0.9934935 0.9704082 0.9841837
## [36,] 1.235405e-01 0.9692308 0.9994624 0.9934935 0.9704082 0.9836735
## [37,] 1.003633e-01 0.9682692 0.9994624 0.9909910 0.9709184 0.9821429
## [38,] 8.153432e-02 0.9668269 1.0000000 0.9904905 0.9709184 0.9821429
## [39,] 6.623782e-02 0.9653846 1.0000000 0.9904905 0.9704082 0.9821429
## [40,] 5.381107e-02 0.9649038 1.0000000 0.9834835 0.9709184 0.9821429
## [41,] 4.371568e-02 0.9629808 1.0000000 0.9834835 0.9709184 0.9785714
## [42,] 3.551427e-02 0.9625000 1.0000000 0.9839840 0.9704082 0.9785714
## [43,] 2.885150e-02 0.9620192 1.0000000 0.9839840 0.9704082 0.9785714
## [44,] 2.343873e-02 0.9615385 1.0000000 0.9799800 0.9704082 0.9785714
## [45,] 1.904143e-02 0.9610577 1.0000000 0.9799800 0.9709184 0.9785714
## [46,] 1.546911e-02 0.9610577 1.0000000 0.9799800 0.9709184 0.9785714
## [47,] 1.256698e-02 0.9610577 1.0000000 0.9799800 0.9709184 0.9785714
## [48,] 1.020931e-02 0.9610577 1.0000000 0.9789790 0.9704082 0.9755102
## [49,] 8.293961e-03 0.9605769 1.0000000 0.9789790 0.9704082 0.9755102
## [50,] 6.737947e-03 0.9605769 1.0000000 0.9789790 0.9704082 0.9755102
```

```r
cv_res <- as.data.frame(cv_test) #colnames(c("auc1", "auc2", "auc3", "auc4", "auc5"))
colnames(cv_res) <- c("res", "auc1", "auc2", "auc3", "auc4", "auc5")
cv_lambda <- cv_res[1]
mean_auc <- cv_res %>% dplyr::select(-1) %>% rowMeans()
cv_auc <- cbind(cv_lambda, mean_auc)
maxauc <- max(cv_auc$mean_auc)
bestlambda <- cv_auc[which(cv_auc$mean_auc == maxauc ),]$res
cv_auc %>%
  ggplot(x = res, y = mean_auc ) +
  geom_line(aes(x = res, y = mean_auc), col = "blue") +
  geom_vline(xintercept = bestlambda, linetype = "dashed", col = "red") +
  labs(title = "Mean AUC vs. Lambda",
       x = "Lambda",
       y = "Mean AUC")
```



Compare full model and lasso model

```
#corresponding betas of best lambda
lasso_beta <- pathwise_sol[which(pathwise_sol$lambda == bestlambda ),][2:21] %>% as.numeric()

#prediction performance function
predict <- function(x, y, betavec) {
  theta <- x %*% betavec
  p <- exp(theta) / (1 + exp(theta))
  auc <- auc(y, p)
  }
y_test <- factor(breast_test$diagnosis)

auc_lasso <- predict(x_test_stan, y_test, lasso_beta)
auc_lasso
```

```
## Area under the curve: 0.994
```

```
cbind(auc_full, auc_lasso) %>% knitr::kable()
```

auc_full	auc_lasso
0.9940432	0.9940432

```
#coefficients of full and lasso models
glm_beta <- glm.fit$coefficients %>% as.vector()
coefnames <- rownames(coef(summary(glm.fit)))
```

```r
cbind(coefnames, glm_beta, lasso_beta) %>% knitr::kable()
```

coefnames	glm_beta	lasso_beta
(Intercept)	-38.2608907757199	-0.38820719375466
radius_mean	0.831596350164732	1.84329203721761
texture_mean	0.379349353306726	0.95102116544444
smoothness_mean	40.9495517882559	0
concavity_mean	59.6833431432281	0
symmetry_mean	-71.7273964538226	0
fractal_dimension_mean	-320.977550822923	0
radius_se	21.7064316060089	1.36028936129994
texture_se	1.03769952599784	0
smoothness_se	336.121422308188	0
compactness_se	138.69659693517	-0.121541937125041
concavity_se	-29.7688436667209	0
concave_points_se	-79.2440722780414	0
symmetry_se	-277.790958353821	0
fractal_dimension_se	-1880.64791126806	-0.206713934103306
smoothness_worst	15.6738104647837	0.603682066107601
compactness_worst	-32.2868668259412	0
concave_points_worst	66.4905094896056	1.99112818165505
symmetry_worst	61.5114221961451	0.464828563771562
fractal_dimension_worst	268.401207994167	0