STATS 305A_HW 4

Muhammad Ahmed Chaudhry

12/9/2021

In the knitted markdown, some code lines get cropped out due to page margins. However, the R markdown and knitted files may also be viewed at https://github.com/Ahmed14974/STATS305A

Problem 4.1 c)

We run the `generate-outlier-data.r` file separately which generates our data for us with a train-test split and also provides us with a function to minimize the objective function in equation (4.1).

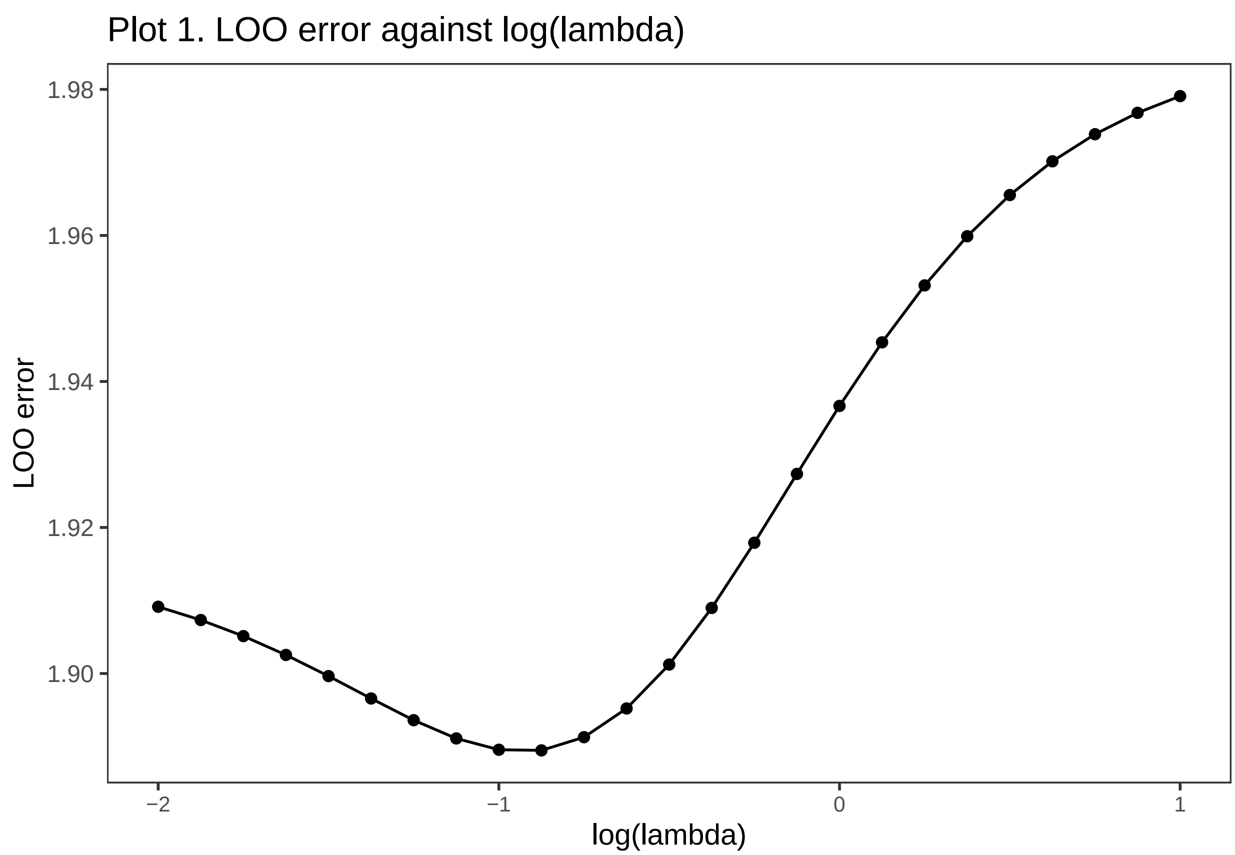
```
set.seed(2022)
data <- generate.data()

# Getting the training and test matrices
X_train <- as.matrix(data[[1]])
y_train <- as.matrix(data[[2]])
X_test <- as.matrix(data[[3]])
y_test <- as.matrix(data[[4]])
```

Next, to implement the leave one out cross validation procedure, we use the `minimize.robust.ridge` function provided and build from it:

```
iters <- seq(-2, 1, length.out = 25)

lambdas <- 10^iters

l_lambda <- c()

for (j in 1:length(lambdas)){
  beta_hat <- minimize.robust.ridge(X_train, y_train, lambda = lambdas[j])

  epsilon_hat <- y_train - (X_train %*% beta_hat)

  l_prime <- ((exp(epsilon_hat)) / (1+exp(epsilon_hat)))  - ((exp(-1 * epsilon_hat)) / ((1+exp(-1 * epsi
  l_2prime <- ((exp(epsilon_hat)) / ((1+exp(epsilon_hat))^2)) + ((exp(-1 * epsilon_hat)) / ((1+exp(-1 *

  XTL <- (t(X_train)) %*% (diag(as.vector(l_2prime)))

  H <-  ((1) / (dim(X_train)[1])) * (XTL %*% X_train) + (lambdas[j] * diag(dim(X_train)[2]))

  H_inv <- solve(H)

  # H_k <- list()
  H_k_inv <- list()
  g_k <- list()
```

```
  delta_hat <- list()
  epsilon_hat_neg_k <- c()
  loss <- c()

  for (i in 1:dim(X_train)[1]){
    numerator <- ((l_2prime[i]) / dim(X_train)[1]) * (H_inv %*% X_train[i,]) %*% (t(X_train[i,]) %*% H_i
    denominator <- as.numeric(1 - ((l_2prime[i]/dim(X_train)[1]) * (t(X_train[i,]) %*% H_inv) %*% (X_tra
    # H_k_iter <- H - ( (1 / dim(X_train)[1]) * (l_2prime[i]) * (X_train[i,] %*% t(X_train[i,])) )
    # H_k[[i]] <- H_k_iter
    H_k_inv_iter <- H_inv + (numerator/denominator)
    H_k_inv[[i]] <- H_k_inv_iter
    g_k_iter <- (l_prime[i]/(dim(X_train)[1])) * (X_train[i,])
    g_k[[i]] <- g_k_iter
    delta_hat[[i]] <-  (-1) * (H_k_inv_iter %*% g_k_iter)
    eps_hat_k <- epsilon_hat[i] + as.numeric((-1) * (X_train[i,] %*% delta_hat[[i]]))
    epsilon_hat_neg_k <- append(epsilon_hat_neg_k, eps_hat_k)
    loss_iter <- log(1 + exp(eps_hat_k)) + log(1 + exp(-1 * eps_hat_k))
    loss <- append(loss, loss_iter)
  }


  l_lambda <- append(l_lambda, mean(loss))
}
```

```
plot_data <- as.data.frame(cbind(iters, l_lambda))


plot_1 <- ggplot(plot_data,
       aes(x = iters, y = l_lambda)) +
  geom_point(aes(x = iters, y = l_lambda))+
  geom_line(aes(x = iters,
                y = l_lambda))+
  xlab("log(lambda)") +
  ylab("LOO error") +

  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("Plot 1. LOO error against log(lambda)")
  # geom_text(
  #   label=round(plot_data$avg_gap,5),
  #   vjust = -3.0,
  #   # hjust = 1.0,
  #   position = position_dodge(width=1),
  #   size =2
  # )


plot_1
```



From the plot above we see that the leave-one-out error first decreases as log(lambda) increases from -2 to just above -1, and then starts to increase again as log(lambda) increases to 1.

d) From the analysis above, we get the $\hat{\lambda}$ that minimizes the LOO error, from which we can get ERR_{test}

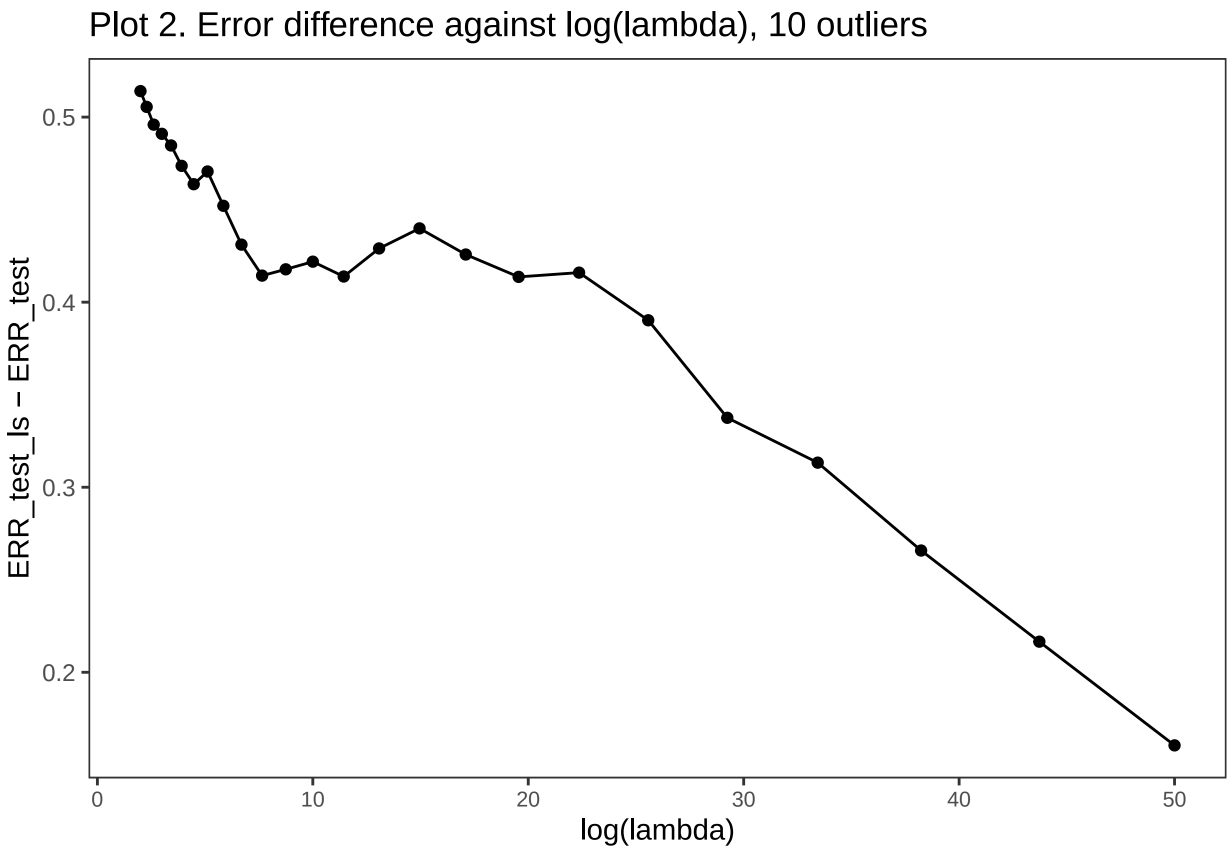
```
lambda_hat <- 10^iters[which.min(l_lambda)]

beta_hat_new <- minimize.robust.ridge(X_train, y_train, lambda = lambda_hat)

error <- y_test - (X_test %*% beta_hat_new)

ERR_test <- median(abs(error))
```

Now, we fit the least squares regularized regression:

```
iters <- seq(log10(2), log10(50), length.out = 25)

lambdas <- 10^iters

ERR_test_ls_lambda <- c()

for (i in 1:length(lambdas)){
  XTX <- t(X_train) %*% X_train

  XTXlambda <- XTX + (lambdas[i] * diag(dim(X_train)[2]))

  XTXlambda_inv <- solve(XTXlambda)

  beta_hat_ls <- (XTXlambda_inv) %*% (t(X_train) %*% y_train)
```

```r
  error_ls <- y_test - (X_test %*% beta_hat_ls)

  ERR_test_ls <- median(abs(error_ls))

  ERR_test_ls_lambda <- append(ERR_test_ls_lambda, ERR_test_ls)
}


ERR_diff <- ERR_test_ls_lambda - ERR_test
```

```r
plot_data_2 <- as.data.frame(cbind(lambdas, ERR_diff))


plot_2 <- ggplot(plot_data_2,
       aes(x = lambdas, y = ERR_diff)) +
  geom_point(aes(x = lambdas, y = ERR_diff))+
  geom_line(aes(x = lambdas, y = ERR_diff))+
  xlab("log(lambda)") +
  ylab("ERR_test_ls - ERR_test") +

  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("Plot 2. Error difference against log(lambda), 10 outliers")
  # geom_text(
  #   label=round(plot_data$avg_gap,5),
  #   vjust = -3.0,
  #   # hjust = 1.0,
  #   position = position_dodge(width=1),
  #   size =2
  # )


plot_2
```



We see that as the regularization increases, the error difference decreases towards 0. This is because with greater regularization, the parameters are penalized enough to not be affected by outliers and so the median absolute prediction error gets closer and closer to that from the tuned robust ridge regression.

e) Now, we repeat the same experiment as in part d) but vary the number of outliers, and generate the plots needed:

0 outliers:

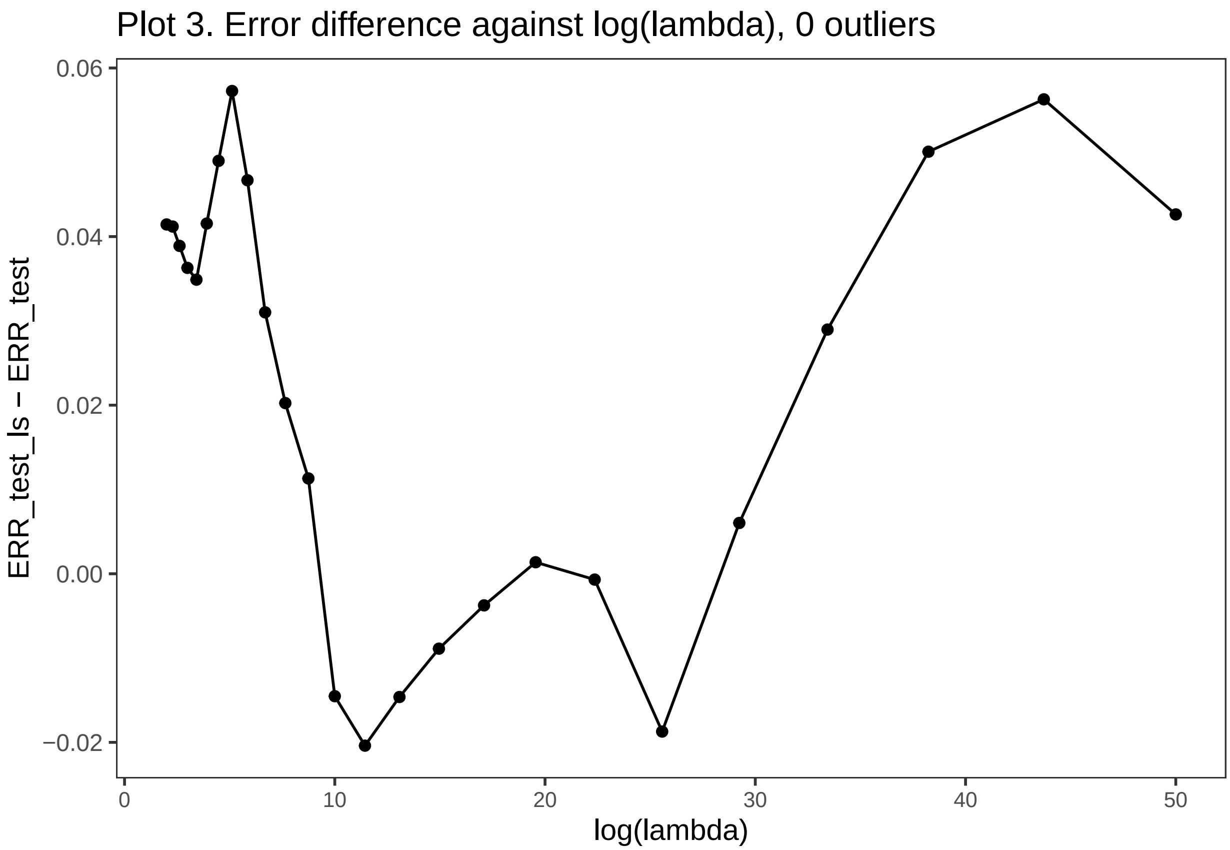
```
# rm(list=ls())
# source('generate-outlier-data.r')

# Generating the data
set.seed(2022)
data <- generate.data(num.outliers = 0)

# Getting the training and test matrices
X_train <- as.matrix(data[[1]])
y_train <- as.matrix(data[[2]])
X_test <- as.matrix(data[[3]])
y_test <- as.matrix(data[[4]])


# Tuning the robust ridge regressiion
iters <- seq(-2, 1, length.out = 25)

lambdas <- 10^iters
```

```r
l_lambda <- c()

for (j in 1:length(lambdas)){
  beta_hat <- minimize.robust.ridge(X_train, y_train, lambda = lambdas[j])

  epsilon_hat <- y_train - (X_train %*% beta_hat)

  l_prime <- ((exp(epsilon_hat)) / (1+exp(epsilon_hat)))  - ((exp(-1 * epsilon_hat)) / ((1+exp(-1 * epsi
  l_2prime <- ((exp(epsilon_hat)) / ((1+exp(epsilon_hat))^2)) + ((exp(-1 * epsilon_hat)) / ((1+exp(-1 *

  XTL <- (t(X_train)) %*% (diag(as.vector(l_2prime)))

  H <-  ((1) / (dim(X_train)[1])) * (XTL %*% X_train) + (lambdas[j] * diag(dim(X_train)[2]))

  H_inv <- solve(H)

  # H_k <- list()
  H_k_inv <- list()
  g_k <- list()
  delta_hat <- list()
  epsilon_hat_neg_k <- c()
  loss <- c()

  for (i in 1:dim(X_train)[1]){
    numerator <- ((l_2prime[i]) / dim(X_train)[1]) * (H_inv %*% X_train[i,]) %*% (t(X_train[i,]) %*% H_i
    denominator <- as.numeric(1 - ((l_2prime[i]/dim(X_train)[1]) * (t(X_train[i,]) %*% H_inv) %*% (X_tra
    # H_k_iter <- H - ( (1 / dim(X_train)[1]) * (l_2prime[i]) * (X_train[i,] %*% t(X_train[i,])) )
    # H_k[[i]] <- H_k_iter
    H_k_inv_iter <- H_inv + (numerator/denominator)
    H_k_inv[[i]] <- H_k_inv_iter
    g_k_iter <- (l_prime[i]/(dim(X_train)[1])) * (X_train[i,])
    g_k[[i]] <- g_k_iter
    delta_hat[[i]] <-  (-1) * (H_k_inv_iter %*% g_k_iter)
    eps_hat_k <- epsilon_hat[i] + as.numeric((-1) * (X_train[i,] %*% delta_hat[[i]]))
    epsilon_hat_neg_k <- append(epsilon_hat_neg_k, eps_hat_k)
    loss_iter <- log(1 + exp(eps_hat_k)) + log(1 + exp(-1 * eps_hat_k))
    loss <- append(loss, loss_iter)
  }


  l_lambda <- append(l_lambda, mean(loss))
}


# Getting best lambda and then ERR_test

lambda_hat <- 10^iters[which.min(l_lambda)]

beta_hat_new <- minimize.robust.ridge(X_train, y_train, lambda = lambda_hat)

error <- y_test - (X_test %*% beta_hat_new)

ERR_test <- median(abs(error))
```

```r
# Fitting the least squares ridge and getting the error differences

iters <- seq(log10(2), log10(50), length.out = 25)

lambdas <- 10^iters

ERR_test_ls_lambda <- c()

for (i in 1:length(lambdas)){
  XTX <- t(X_train) %*% X_train

  XTXlambda <- XTX + (lambdas[i] * diag(dim(X_train)[2]))

  XTXlambda_inv <- solve(XTXlambda)

  beta_hat_ls <- (XTXlambda_inv) %*% (t(X_train) %*% y_train)


  error_ls <- y_test - (X_test %*% beta_hat_ls)

  ERR_test_ls <- median(abs(error_ls))

  ERR_test_ls_lambda <- append(ERR_test_ls_lambda, ERR_test_ls)
}


ERR_diff <- ERR_test_ls_lambda - ERR_test
```

```r
ERR_diff_0 <- ERR_diff

plot_data_3 <- as.data.frame(cbind(lambdas, ERR_diff_0))


plot_3 <- ggplot(plot_data_3,
       aes(x = lambdas, y = ERR_diff_0)) +
  geom_point(aes(x = lambdas, y = ERR_diff_0))+
  geom_line(aes(x = lambdas, y = ERR_diff_0))+
  xlab("log(lambda)") +
  ylab("ERR_test_ls - ERR_test") +

  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("Plot 3. Error difference against log(lambda), 0 outliers")
  # geom_text(
  #   label=round(plot_data$avg_gap,5),
  #   vjust = -3.0,
  #   # hjust = 1.0,
  #   position = position_dodge(width=1),
  #   size =2
  # )
```

```
plot_3
```



We see that with 0 outliers, there is no observable pattern since no robustness is needed. In fact, for some values of log(lambda), the least squares ridge regression performs bettern than the tuned robust ridge regression which is not surprising, as no robustness is needed. It seems that for very high values of log(lambda), the least squares ridge regression is underefitting.

5 outliers:

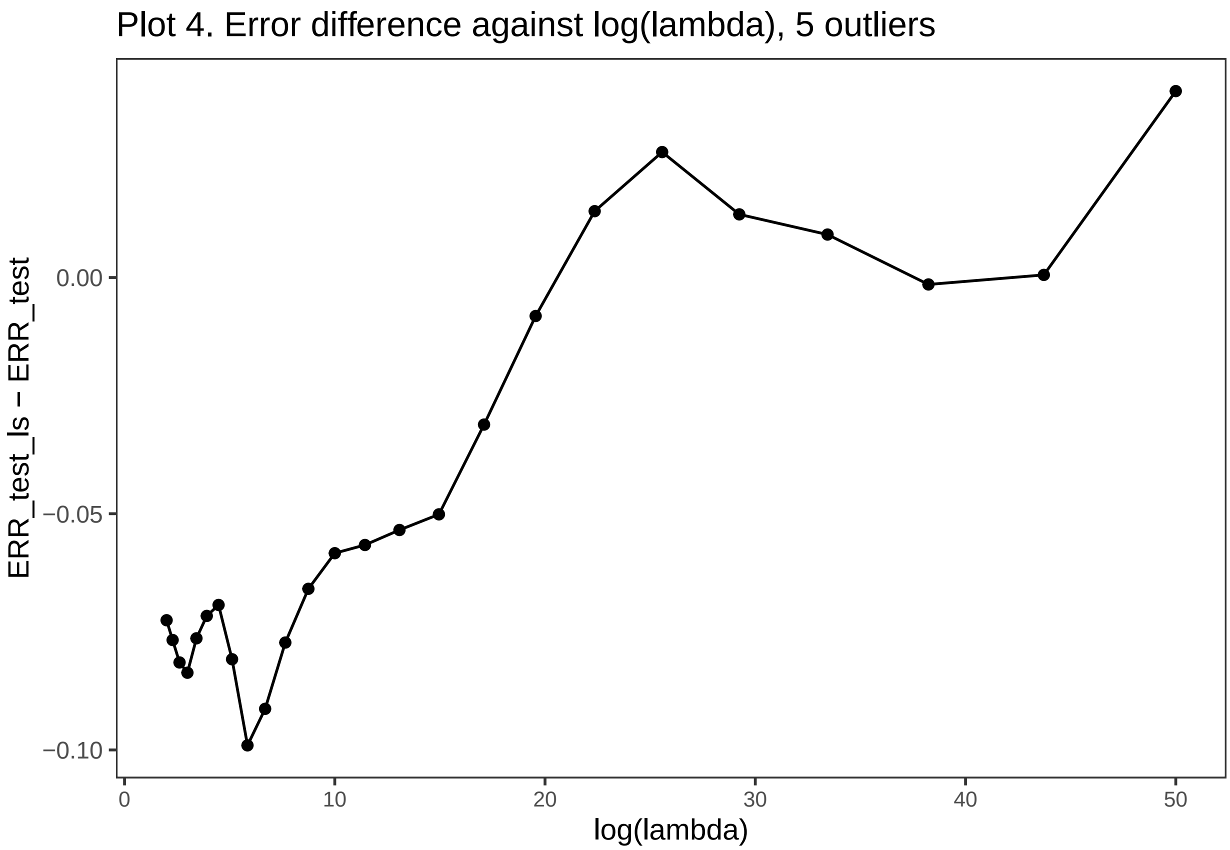
```
# Generating the data
set.seed(2022)
data <- generate.data(num.outliers = 5)

# Getting the training and test matrices
X_train <- as.matrix(data[[1]])
y_train <- as.matrix(data[[2]])
X_test <- as.matrix(data[[3]])
y_test <- as.matrix(data[[4]])


# Tuning the robust ridge regression
iters <- seq(-2, 1, length.out = 25)

lambdas <- 10^iters

l_lambda <- c()
```

```
for (j in 1:length(lambdas)){
  beta_hat <- minimize.robust.ridge(X_train, y_train, lambda = lambdas[j])

  epsilon_hat <- y_train - (X_train %*% beta_hat)

  l_prime <- ((exp(epsilon_hat)) / (1+exp(epsilon_hat)))  - ((exp(-1 * epsilon_hat)) / ((1+exp(-1 * eps
  l_2prime <- ((exp(epsilon_hat)) / ((1+exp(epsilon_hat))^2)) + ((exp(-1 * epsilon_hat)) / ((1+exp(-1 *

  XTL <- (t(X_train)) %*% (diag(as.vector(l_2prime)))

  H <-  ((1) / (dim(X_train)[1])) * (XTL %*% X_train) + (lambdas[j] * diag(dim(X_train)[2]))

  H_inv <- solve(H)

  # H_k <- list()
  H_k_inv <- list()
  g_k <- list()
  delta_hat <- list()
  epsilon_hat_neg_k <- c()
  loss <- c()

  for (i in 1:dim(X_train)[1]){
    numerator <- ((l_2prime[i]) / dim(X_train)[1]) * (H_inv %*% X_train[i,]) %*% (t(X_train[i,]) %*% H_
    denominator <- as.numeric(1 - ((l_2prime[i]/dim(X_train)[1]) * (t(X_train[i,]) %*% H_inv) %*% (X_tra
    # H_k_iter <- H - ( (1 / dim(X_train)[1]) * (l_2prime[i]) * (X_train[i,] %*% t(X_train[i,])) )
    # H_k[[i]] <- H_k_iter
    H_k_inv_iter <- H_inv + (numerator/denominator)
    H_k_inv[[i]] <- H_k_inv_iter
    g_k_iter <- (l_prime[i]/(dim(X_train)[1])) * (X_train[i,])
    g_k[[i]] <- g_k_iter
    delta_hat[[i]] <-  (-1) * (H_k_inv_iter %*% g_k_iter)
    eps_hat_k <- epsilon_hat[i] + as.numeric((-1) * (X_train[i,] %*% delta_hat[[i]]))
    epsilon_hat_neg_k <- append(epsilon_hat_neg_k, eps_hat_k)
    loss_iter <- log(1 + exp(eps_hat_k)) + log(1 + exp(-1 * eps_hat_k))
    loss <- append(loss, loss_iter)
  }


  l_lambda <- append(l_lambda, mean(loss))
}


# Getting best lambda and then ERR_test

lambda_hat <- 10^iters[which.min(l_lambda)]

beta_hat_new <- minimize.robust.ridge(X_train, y_train, lambda = lambda_hat)

error <- y_test - (X_test %*% beta_hat_new)

ERR_test <- median(abs(error))
```

```r
# Fitting the least squares ridge and getting the error differences

iters <- seq(log10(2), log10(50), length.out = 25)

lambdas <- 10^iters

ERR_test_ls_lambda <- c()

for (i in 1:length(lambdas)){
  XTX <- t(X_train) %*% X_train

  XTXlambda <- XTX + (lambdas[i] * diag(dim(X_train)[2]))

  XTXlambda_inv <- solve(XTXlambda)

  beta_hat_ls <- (XTXlambda_inv) %*% (t(X_train) %*% y_train)


  error_ls <- y_test - (X_test %*% beta_hat_ls)

  ERR_test_ls <- median(abs(error_ls))

  ERR_test_ls_lambda <- append(ERR_test_ls_lambda, ERR_test_ls)
}


ERR_diff <- ERR_test_ls_lambda - ERR_test
```

```r
ERR_diff_5 <- ERR_diff

plot_data_4 <- as.data.frame(cbind(lambdas, ERR_diff_5))


plot_4 <- ggplot(plot_data_4,
       aes(x = lambdas, y = ERR_diff_5)) +
  geom_point(aes(x = lambdas, y = ERR_diff_5))+
  geom_line(aes(x = lambdas, y = ERR_diff_5))+
  xlab("log(lambda)") +
  ylab("ERR_test_ls - ERR_test") +

  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("Plot 4. Error difference against log(lambda), 5 outliers")
  # geom_text(
  #   label=round(plot_data$avg_gap,5),
  #   vjust = -3.0,
  #   # hjust = 1.0,
  #   position = position_dodge(width=1),
  #   size =2
  # )
```

```r
plot_4
```



In this situation, we see that the least squares ridge regression again performs better than the tuned robust ridge regression for some values of log(lambda). Since there are only 5 outliers, perhaps the tuning is causing underfitting to some extent. Again for very large values of log(lambda), the least squares ridge regression starts to underfit.

We already got the plot for 10 outliers in part d).

15 outliers:

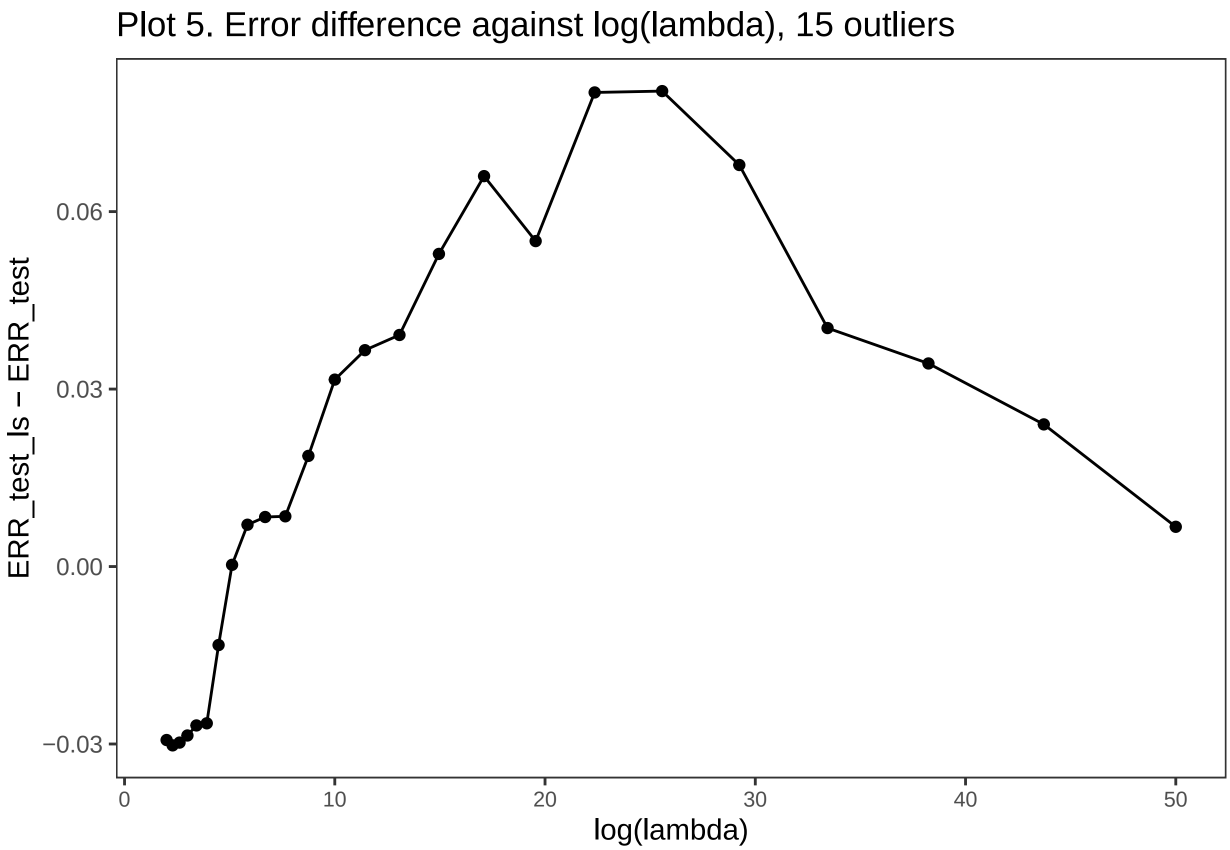
```r
# Generating the data
set.seed(2022)
data <- generate.data(num.outliers = 15)

# Getting the training and test matrices
X_train <- as.matrix(data[[1]])
y_train <- as.matrix(data[[2]])
X_test <- as.matrix(data[[3]])
y_test <- as.matrix(data[[4]])


# Tuning the robust ridge regression
iters <- seq(-2, 1, length.out = 25)

lambdas <- 10^iters

l_lambda <- c()
```

```r
for (j in 1:length(lambdas)){
  beta_hat <- minimize.robust.ridge(X_train, y_train, lambda = lambdas[j])

  epsilon_hat <- y_train - (X_train %*% beta_hat)

  l_prime <- ((exp(epsilon_hat)) / (1+exp(epsilon_hat)))  - ((exp(-1 * epsilon_hat)) / ((1+exp(-1 * epsi
  l_2prime <- ((exp(epsilon_hat)) / ((1+exp(epsilon_hat))^2)) + ((exp(-1 * epsilon_hat)) / ((1+exp(-1 *

  XTL <- (t(X_train)) %*% (diag(as.vector(l_2prime)))

  H <-  ((1) / (dim(X_train)[1])) * (XTL %*% X_train) + (lambdas[j] * diag(dim(X_train)[2]))

  H_inv <- solve(H)

  # H_k <- list()
  H_k_inv <- list()
  g_k <- list()
  delta_hat <- list()
  epsilon_hat_neg_k <- c()
  loss <- c()

  for (i in 1:dim(X_train)[1]){
    numerator <- ((l_2prime[i]) / dim(X_train)[1]) * (H_inv %*% X_train[i,]) %*% (t(X_train[i,]) %*% H_i
    denominator <- as.numeric(1 - ((l_2prime[i]/dim(X_train)[1]) * (t(X_train[i,]) %*% H_inv) %*% (X_tra
    # H_k_iter <- H - ( (1 / dim(X_train)[1]) * (l_2prime[i]) * (X_train[i,] %*% t(X_train[i,])) )
    # H_k[[i]] <- H_k_iter
    H_k_inv_iter <- H_inv + (numerator/denominator)
    H_k_inv[[i]] <- H_k_inv_iter
    g_k_iter <- (l_prime[i]/(dim(X_train)[1])) * (X_train[i,])
    g_k[[i]] <- g_k_iter
    delta_hat[[i]] <-  (-1) * (H_k_inv_iter %*% g_k_iter)
    eps_hat_k <- epsilon_hat[i] + as.numeric((-1) * (X_train[i,] %*% delta_hat[[i]]))
    epsilon_hat_neg_k <- append(epsilon_hat_neg_k, eps_hat_k)
    loss_iter <- log(1 + exp(eps_hat_k)) + log(1 + exp(-1 * eps_hat_k))
    loss <- append(loss, loss_iter)
  }


  l_lambda <- append(l_lambda, mean(loss))
}


# Getting best lambda and then ERR_test

lambda_hat <- 10^iters[which.min(l_lambda)]

beta_hat_new <- minimize.robust.ridge(X_train, y_train, lambda = lambda_hat)

error <- y_test - (X_test %*% beta_hat_new)

ERR_test <- median(abs(error))
```

```r
# Fitting the least squares ridge and getting the error differences

iters <- seq(log10(2), log10(50), length.out = 25)

lambdas <- 10^iters

ERR_test_ls_lambda <- c()

for (i in 1:length(lambdas)){
  XTX <- t(X_train) %*% X_train

  XTXlambda <- XTX + (lambdas[i] * diag(dim(X_train)[2]))

  XTXlambda_inv <- solve(XTXlambda)

  beta_hat_ls <- (XTXlambda_inv) %*% (t(X_train) %*% y_train)


  error_ls <- y_test - (X_test %*% beta_hat_ls)

  ERR_test_ls <- median(abs(error_ls))

  ERR_test_ls_lambda <- append(ERR_test_ls_lambda, ERR_test_ls)
}


ERR_diff <- ERR_test_ls_lambda - ERR_test
```

```r
ERR_diff_15 <- ERR_diff

plot_data_5 <- as.data.frame(cbind(lambdas, ERR_diff_15))


plot_5 <- ggplot(plot_data_5,
       aes(x = lambdas, y = ERR_diff_15)) +
  geom_point(aes(x = lambdas, y = ERR_diff_15))+
  geom_line(aes(x = lambdas, y = ERR_diff_15))+
  xlab("log(lambda)") +
  ylab("ERR_test_ls - ERR_test") +

  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("Plot 5. Error difference against log(lambda), 15 outliers")
  # geom_text(
  #   label=round(plot_data$avg_gap,5),
  #   vjust = -3.0,
  #   # hjust = 1.0,
  #   position = position_dodge(width=1),
  #   size =2
  # )
```

```
plot_5
```



Again, for very small values of log(lambda), the least squares ridge regression seems to be performing better, and then starts to underfit. However, as log(lambda) increases beyond approximately 25, we see that the least squares ridge regression starts to get closer to the tuned robust ridge regression in terms of median prediction error.

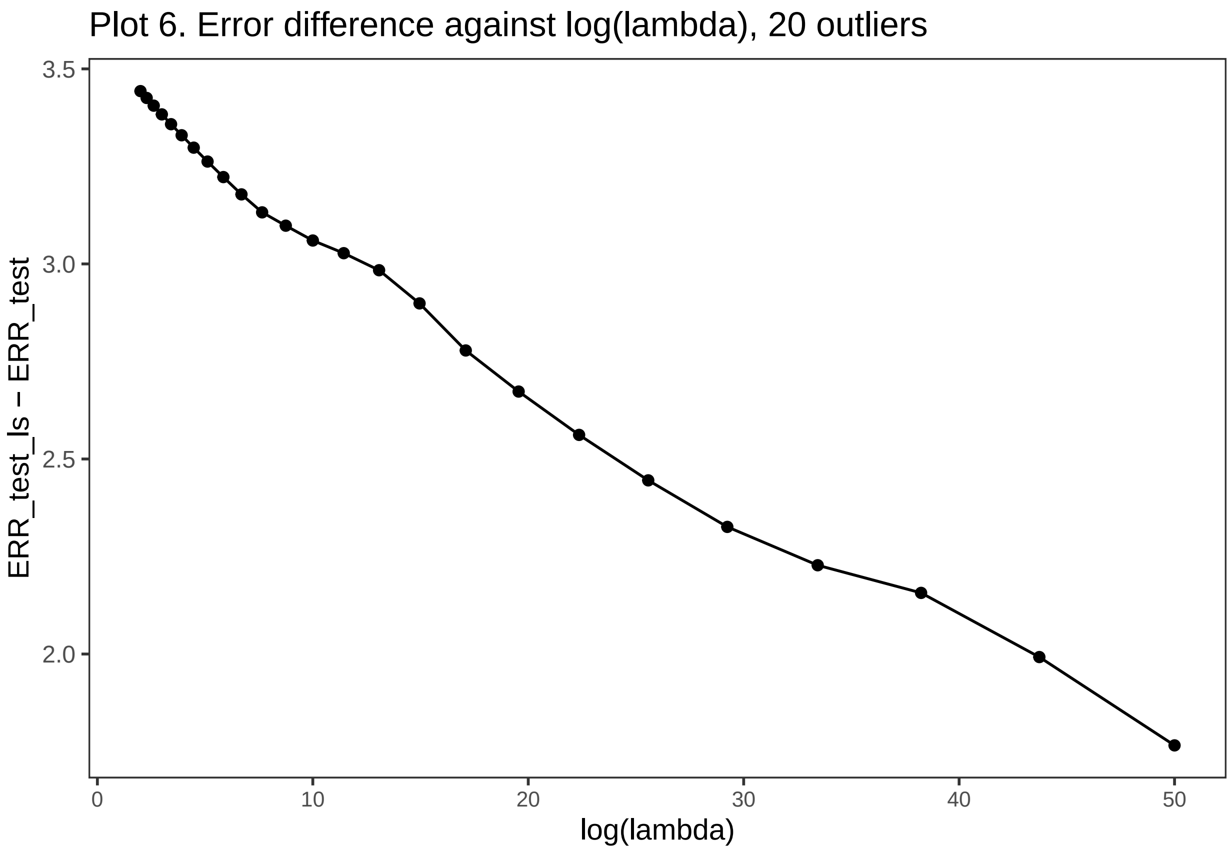
20 outliers:

```
# Generating the data
set.seed(2022)
data <- generate.data(num.outliers = 20)

# Getting the training and test matrices
X_train <- as.matrix(data[[1]])
y_train <- as.matrix(data[[2]])
X_test <- as.matrix(data[[3]])
y_test <- as.matrix(data[[4]])


# Tuning the robust ridge regression
iters <- seq(-2, 1, length.out = 25)

lambdas <- 10^iters

l_lambda <- c()

for (j in 1:length(lambdas)){
```

```r
  beta_hat <- minimize.robust.ridge(X_train, y_train, lambda = lambdas[j])

  epsilon_hat <- y_train - (X_train %*% beta_hat)

  l_prime <- ((exp(epsilon_hat)) / (1+exp(epsilon_hat)))  - ((exp(-1 * epsilon_hat)) / ((1+exp(-1 * epsi
  l_2prime <- ((exp(epsilon_hat)) / ((1+exp(epsilon_hat))^2)) + ((exp(-1 * epsilon_hat)) / ((1+exp(-1 *

  XTL <- (t(X_train)) %*% (diag(as.vector(l_2prime)))

  H <-  ((1) / (dim(X_train)[1])) * (XTL %*% X_train) + (lambdas[j] * diag(dim(X_train)[2]))

  H_inv <- solve(H)

  # H_k <- list()
  H_k_inv <- list()
  g_k <- list()
  delta_hat <- list()
  epsilon_hat_neg_k <- c()
  loss <- c()

  for (i in 1:dim(X_train)[1]){
    numerator <- ((l_2prime[i]) / dim(X_train)[1]) * (H_inv %*% X_train[i,]) %*% (t(X_train[i,]) %*% H_i
    denominator <- as.numeric(1 - ((l_2prime[i]/dim(X_train)[1]) * (t(X_train[i,]) %*% H_inv) %*% (X_tra
    # H_k_iter <- H - ( (1 / dim(X_train)[1]) * (l_2prime[i]) * (X_train[i,] %*% t(X_train[i,])) )
    # H_k[[i]] <- H_k_iter
    H_k_inv_iter <- H_inv + (numerator/denominator)
    H_k_inv[[i]] <- H_k_inv_iter
    g_k_iter <- (l_prime[i]/(dim(X_train)[1])) * (X_train[i,])
    g_k[[i]] <- g_k_iter
    delta_hat[[i]] <-  (-1) * (H_k_inv_iter %*% g_k_iter)
    eps_hat_k <- epsilon_hat[i] + as.numeric((-1) * (X_train[i,] %*% delta_hat[[i]]))
    epsilon_hat_neg_k <- append(epsilon_hat_neg_k, eps_hat_k)
    loss_iter <- log(1 + exp(eps_hat_k)) + log(1 + exp(-1 * eps_hat_k))
    loss <- append(loss, loss_iter)
  }


  l_lambda <- append(l_lambda, mean(loss))
}


# Getting best lambda and then ERR_test

lambda_hat <- 10^iters[which.min(l_lambda)]

beta_hat_new <- minimize.robust.ridge(X_train, y_train, lambda = lambda_hat)

error <- y_test - (X_test %*% beta_hat_new)

ERR_test <- median(abs(error))


# Fitting the least squares ridge and getting the error differences
```

```r
iters <- seq(log10(2), log10(50), length.out = 25)

lambdas <- 10^iters

ERR_test_ls_lambda <- c()

for (i in 1:length(lambdas)){
  XTX <- t(X_train) %*% X_train

  XTXlambda <- XTX + (lambdas[i] * diag(dim(X_train)[2]))

  XTXlambda_inv <- solve(XTXlambda)

  beta_hat_ls <- (XTXlambda_inv) %*% (t(X_train) %*% y_train)


  error_ls <- y_test - (X_test %*% beta_hat_ls)

  ERR_test_ls <- median(abs(error_ls))

  ERR_test_ls_lambda <- append(ERR_test_ls_lambda, ERR_test_ls)
}


ERR_diff <- ERR_test_ls_lambda - ERR_test
```

```r
ERR_diff_20 <- ERR_diff

plot_data_6 <- as.data.frame(cbind(lambdas, ERR_diff_20))


plot_6 <- ggplot(plot_data_6,
       aes(x = lambdas, y = ERR_diff_20)) +
  geom_point(aes(x = lambdas, y = ERR_diff_20))+
  geom_line(aes(x = lambdas, y = ERR_diff_20))+
  xlab("log(lambda)") +
  ylab("ERR_test_ls - ERR_test") +

  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("Plot 6. Error difference against log(lambda), 20 outliers")
  # geom_text(
  #   label=round(plot_data$avg_gap,5),
  #   vjust = -3.0,
  #   # hjust = 1.0,
  #   position = position_dodge(width=1),
  #   size =2
  # )


plot_6
```



For 20 outliers, we see a clear pattern. As the regularization for the least squares ridge regression increases, the median absolute prediction error approaches that of the tuned robust ridge regression, and for small values of log(lambda), the error difference is much higher than in plots seen for fewer outliers.

Finally, 25 outliers:

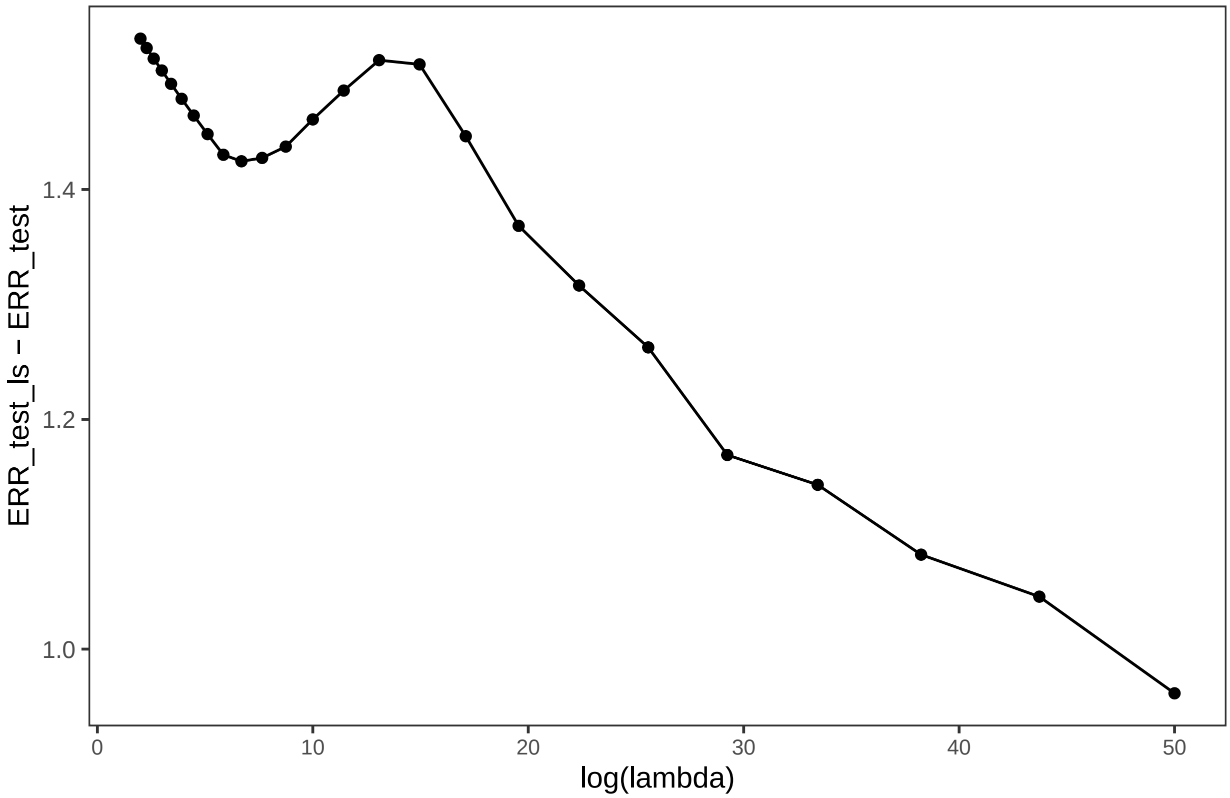
```
# Generating the data
set.seed(2022)
data <- generate.data(num.outliers = 25)

# Getting the training and test matrices
X_train <- as.matrix(data[[1]])
y_train <- as.matrix(data[[2]])
X_test <- as.matrix(data[[3]])
y_test <- as.matrix(data[[4]])


# Tuning the robust ridge regression
iters <- seq(-2, 1, length.out = 25)

lambdas <- 10^iters

l_lambda <- c()

for (j in 1:length(lambdas)){
  beta_hat <- minimize.robust.ridge(X_train, y_train, lambda = lambdas[j])

  epsilon_hat <- y_train - (X_train %*% beta_hat)
```

```
  l_prime <- ((exp(epsilon_hat)) / (1+exp(epsilon_hat)))  - ((exp(-1 * epsilon_hat)) / ((1+exp(-1 * epsi
  l_2prime <- ((exp(epsilon_hat)) / ((1+exp(epsilon_hat))^2)) + ((exp(-1 * epsilon_hat)) / ((1+exp(-1 *

  XTL <- (t(X_train)) %*% (diag(as.vector(l_2prime)))

  H <-  ((1) / (dim(X_train)[1])) * (XTL %*% X_train) + (lambdas[j] * diag(dim(X_train)[2]))

  H_inv <- solve(H)

  # H_k <- list()
  H_k_inv <- list()
  g_k <- list()
  delta_hat <- list()
  epsilon_hat_neg_k <- c()
  loss <- c()

  for (i in 1:dim(X_train)[1]){
    numerator <- ((l_2prime[i]) / dim(X_train)[1]) * (H_inv %*% X_train[i,]) %*% (t(X_train[i,]) %*% H_
    denominator <- as.numeric(1 - ((l_2prime[i]/dim(X_train)[1]) * (t(X_train[i,]) %*% H_inv) %*% (X_tra
    # H_k_iter <- H - ( (1 / dim(X_train)[1]) * (l_2prime[i]) * (X_train[i,] %*% t(X_train[i,])) )
    # H_k[[i]] <- H_k_iter
    H_k_inv_iter <- H_inv + (numerator/denominator)
    H_k_inv[[i]] <- H_k_inv_iter
    g_k_iter <- (l_prime[i]/(dim(X_train)[1])) * (X_train[i,])
    g_k[[i]] <- g_k_iter
    delta_hat[[i]] <-  (-1) * (H_k_inv_iter %*% g_k_iter)
    eps_hat_k <- epsilon_hat[i] + as.numeric((-1) * (X_train[i,] %*% delta_hat[[i]]))
    epsilon_hat_neg_k <- append(epsilon_hat_neg_k, eps_hat_k)
    loss_iter <- log(1 + exp(eps_hat_k)) + log(1 + exp(-1 * eps_hat_k))
    loss <- append(loss, loss_iter)
  }


  l_lambda <- append(l_lambda, mean(loss))
}


# Getting best lambda and then ERR_test

lambda_hat <- 10^iters[which.min(l_lambda)]

beta_hat_new <- minimize.robust.ridge(X_train, y_train, lambda = lambda_hat)

error <- y_test - (X_test %*% beta_hat_new)

ERR_test <- median(abs(error))


# Fitting the least squares ridge and getting the error differences

iters <- seq(log10(2), log10(50), length.out = 25)

lambdas <- 10^iters
```

```r
ERR_test_ls_lambda <- c()

for (i in 1:length(lambdas)){
  XTX <- t(X_train) %*% X_train

  XTXlambda <- XTX + (lambdas[i] * diag(dim(X_train)[2]))

  XTXlambda_inv <- solve(XTXlambda)

  beta_hat_ls <- (XTXlambda_inv) %*% (t(X_train) %*% y_train)


  error_ls <- y_test - (X_test %*% beta_hat_ls)

  ERR_test_ls <- median(abs(error_ls))

  ERR_test_ls_lambda <- append(ERR_test_ls_lambda, ERR_test_ls)
}


ERR_diff <- ERR_test_ls_lambda - ERR_test
```

```r
ERR_diff_25 <- ERR_diff

plot_data_7 <- as.data.frame(cbind(lambdas, ERR_diff_25))


plot_7 <- ggplot(plot_data_7,
       aes(x = lambdas, y = ERR_diff_25)) +
  geom_point(aes(x = lambdas, y = ERR_diff_25))+
  geom_line(aes(x = lambdas, y = ERR_diff_25))+
  xlab("log(lambda)") +
  ylab("ERR_test_ls - ERR_test") +

  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("Plot 7. Error difference against log(lambda), 25 outliers")
  # geom_text(
  #   label=round(plot_data$avg_gap,5),
  #   vjust = -3.0,
  #   # hjust = 1.0,
  #   position = position_dodge(width=1),
  #   size =2
  # )


plot_7
```

Plot 7. Error difference against log(lambda), 25 outliers



The plot obtained for 25 outliers is similar to the one for 20 outliers, as beyond a some deviations at the beginning, as the regularization for least squares increases, again it approaches the tuned robust ridge regression in terms of median absolute prediction error.