STATS 305A_HW3

Muhammad Ahmed Chaudhry

11/19/2021

First we load the packages as needed:

Problem 3.3

First we read in the data:

```
lprostate_data <- read.delim("../Data/lprostate.dat", sep = "\t")

# removing extraneous row.names column
lprostate_data <- lprostate_data[,-1]
```

a) Next we implement the `predictKRR` function. First, we write a helper function which calculates the Gaussian (RBF) kernel between two vectors.

```
rbf_kernel <- function(x = NULL, z = NULL, bandwidth){
  # parameters:
  # vectors x and z
  # bandwidth tau of the Gaussian kernel, a real number > 0

  diff <- x - z
  sum_squares <- sum(diff^2)
  kernel <- exp((-1 * sum_squares) / (2*(bandwidth^2)))
  return(kernel)
}
```

Next we implement the predictKRR function:

```
predictKRR <- function(X = NULL, Z = NULL, alpha = NULL,
                       tau = 1, offset = 1){
  # parameters:
  # X, a d x n data matrix with training data with rows x_i^T
  # z, a d x m matrix of m data points to make new predictions on
  # alpha, an n vector \hat{alpha} which is the fitted parameters from 3.2b)
  # tau, the bandwidth of the Gaussian kernel, a real number > 0
  # offset, an offset parameter b which is a real number

  y_hat <- c()
  for (j in 1:dim(Z)[2]){
    sum <- offset
    for (i in 1:dim(X)[2]){
      sum <- sum + rbf_kernel(X[,i], Z[,j], tau) * alpha[i]
    }
    y_hat <- append(y_hat, sum)
```

```
  }
  return(y_hat)

}
```

b) Next we implement the fitKRR function. First, we write a helper function that builds our Gram Matrix:

```
gram_matrix <- function(X = NULL, bandwidth){
  # parameters:
  # X, a d x n data matrix with training data with rows x_i^T
  # bandwidth, of the Gaussian kernel, a real number > 0
  G <- matrix(data = NA, nrow = dim(X)[2], ncol = dim(X)[2])
  for (j in 1:dim(X)[2]){
    for (i in 1:dim(X)[2]){
      G[i,j] <- rbf_kernel(X[,i], X[,j], bandwidth = bandwidth)
    }
  }
  return(G)
}
```

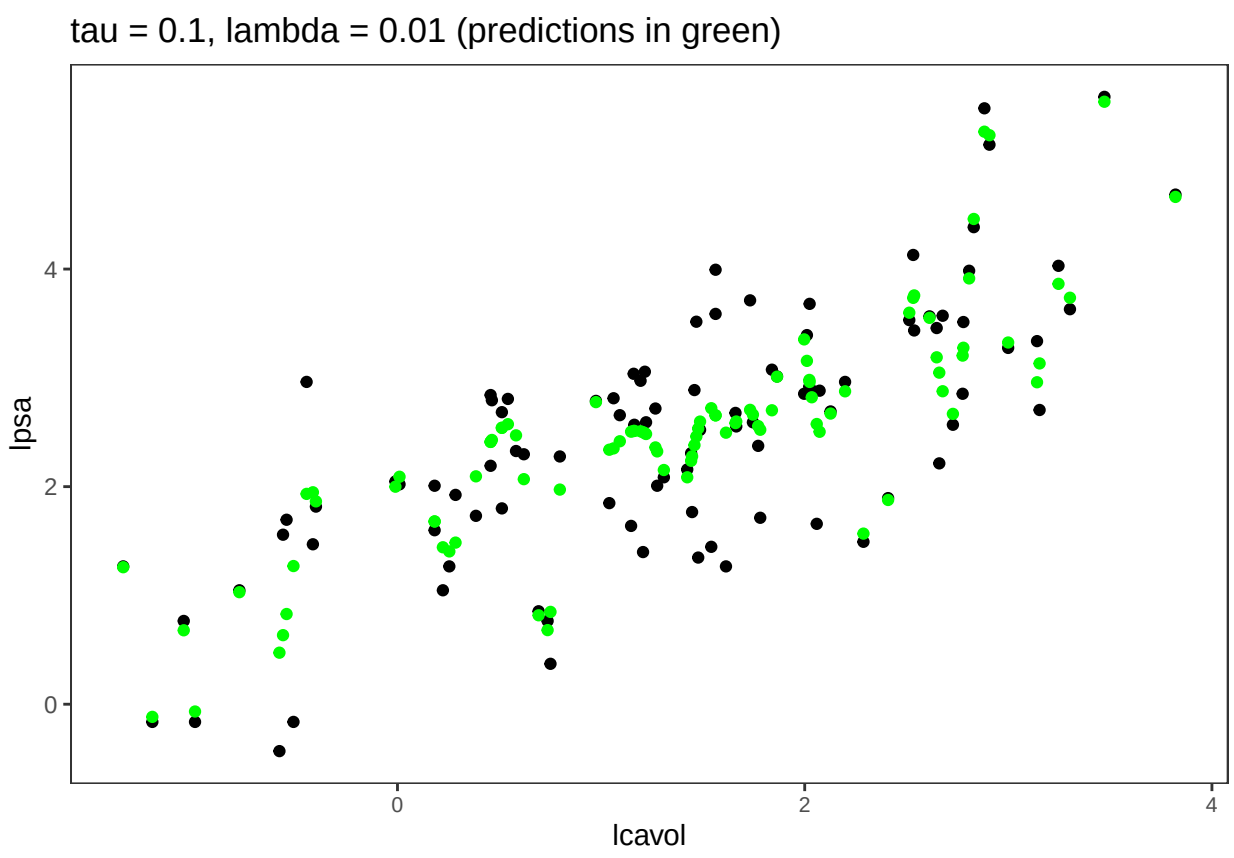
Next we write the fitKRR function:

```
fitKRR <- function(X = NULL, y = NULL,
                   lambda = 1, tau = 1){
  # parameters:
  # X, a d x n data matrix with training data with rows x_i^T
  # y, an n vector of responses
  # lambda, the regularization or shrinkage parameter
  # tau, the bandwidth of the Gaussian kernel, a real number > 0

  # getting alpha_hat
  y_centered <- y - mean(y)
  G <- gram_matrix(X, tau)
  id <- diag(rep(1, dim(G)[1]))
  inv_arg <- G + lambda * id
  inv <- solve(inv_arg)
  alpha_hat <- inv %*% y_centered

  # getting yMean
  y_hat_centered <- G %*% alpha_hat
  yMean <- mean(y_hat_centered)

  return(list(alpha_hat, yMean))
}
```

c) Next, we perform kernel ridge regression using lpsa as the response and lcavol as the only covariate x, with $\tau = 0.1$ fixed. For all plots generated, we remark that the green points are the predictions.

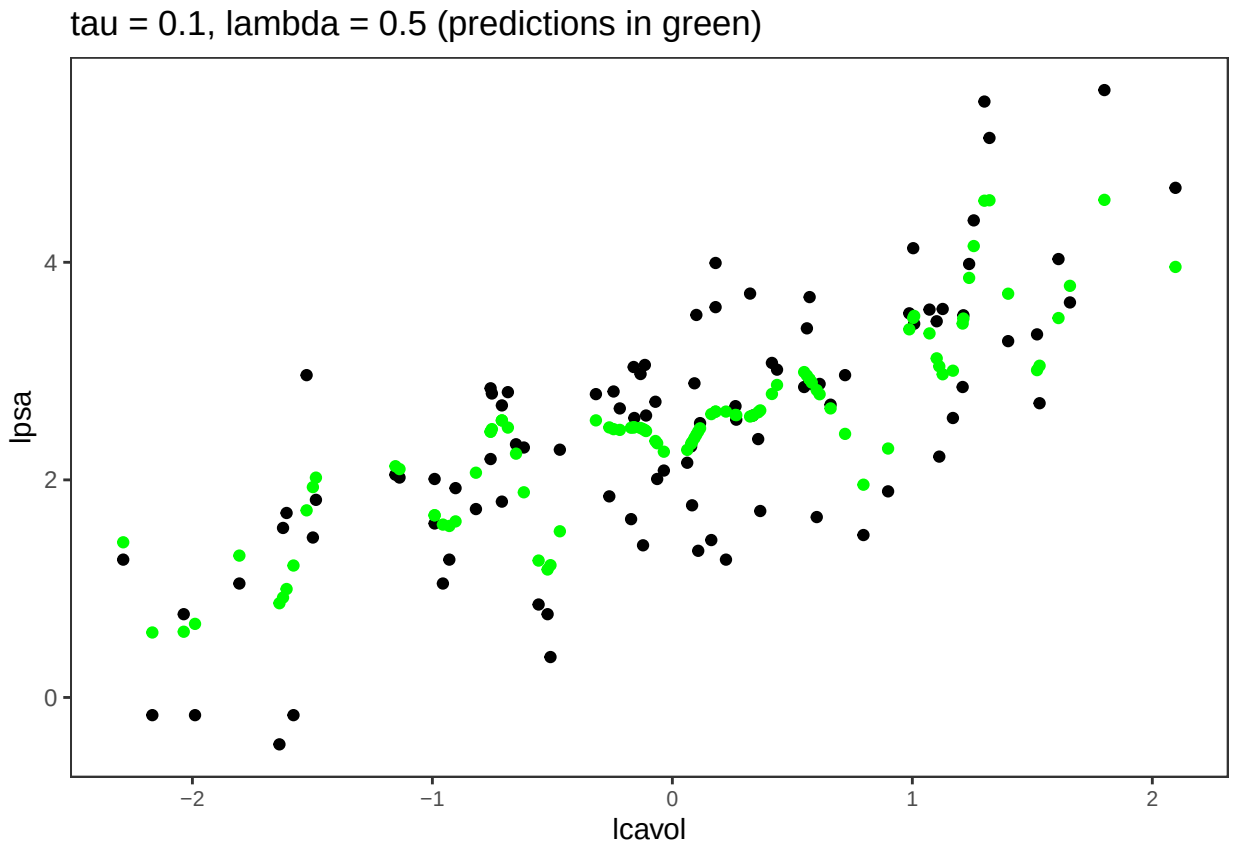
```
y = as.matrix(lprostate_data[,"lpsa"])

X = as.matrix(lprostate_data[,"lcavol"])
X_scaled = scale(X)

alpha_hat_1 <- unlist(fitKRR(X = t(X_scaled), y = y, tau = 0.1, lambda = 0.01)[1])

y_hat_1 <- predictKRR(X = t(X_scaled), Z = t(X_scaled), alpha = alpha_hat_1,
```

```
                      tau = 0.1, offset = mean(y))

plot_data_1 <- as.data.frame(cbind(X, y, y_hat_1))

ggplot(plot_data_1,
       aes(x = V1, y = V2)) +
  geom_point(aes(x = V1, y = V2))+
  geom_point(aes(x = V1,
                 y = y_hat_1), color = "green")+
  xlab("lcavol") +
  ylab("lpsa") +
  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("tau = 0.1, lambda = 0.01 (predictions in green)")
```



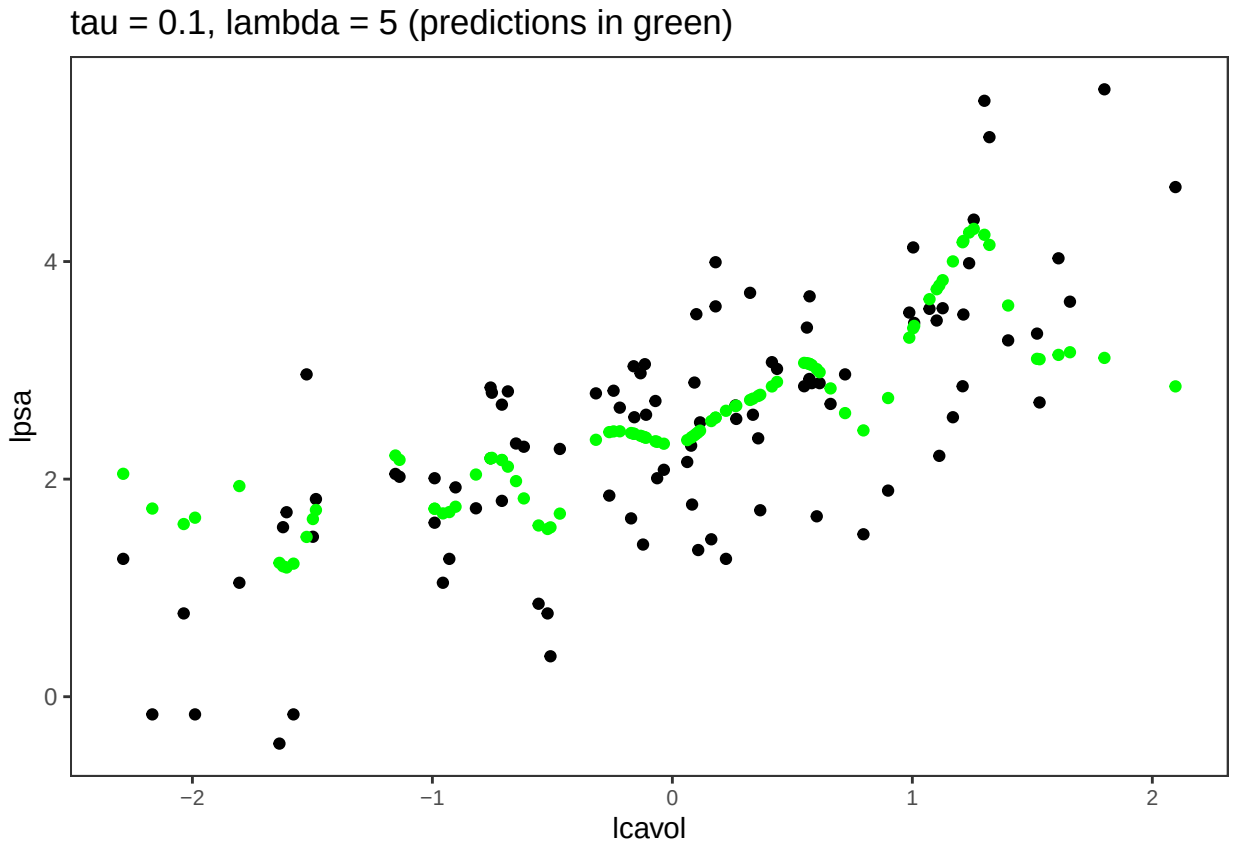

```
y = as.matrix(lprostate_data[,"lpsa"])

X = as.matrix(lprostate_data[,"lcavol"])
X_scaled = scale(X)

alpha_hat_1 <- unlist(fitKRR(X = t(X_scaled), y = y, tau = 0.1, lambda = 0.5)[1])

y_hat_1 <- predictKRR(X = t(X_scaled), Z = t(X_scaled), alpha = alpha_hat_1,
                      tau = 0.1, offset = mean(y))
```

```
plot_data_1 <- as.data.frame(cbind(X_scaled, y, y_hat_1))

ggplot(plot_data_1,
       aes(x = V1, y = V2)) +
  geom_point(aes(x = V1, y = V2))+
  geom_point(aes(x = V1,
                 y = y_hat_1), color = "green")+
  xlab("lcavol") +
  ylab("lpsa") +
  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("tau = 0.1, lambda = 0.5 (predictions in green)")
```




```
y = as.matrix(lprostate_data[,"lpsa"])

X = as.matrix(lprostate_data[,"lcavol"])
X_scaled = scale(X)

alpha_hat_1 <- unlist(fitKRR(X = X_scaled, y = t(y), tau = 0.1, lambda = 5)[1])

y_hat_1 <- predictKRR(X = t(X_scaled), Z = t(X_scaled), alpha = alpha_hat_1,
                      tau = 0.1, offset = mean(y))

plot_data_1 <- as.data.frame(cbind(X_scaled, y, y_hat_1))
```

```
ggplot(plot_data_1,
       aes(x = V1, y = V2)) +
  geom_point(aes(x = V1, y = V2))+
  geom_point(aes(x = V1,
                 y = y_hat_1), color = "green")+
  xlab("lcavol") +
  ylab("lpsa") +
  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("tau = 0.1, lambda = 5 (predictions in green)")
```

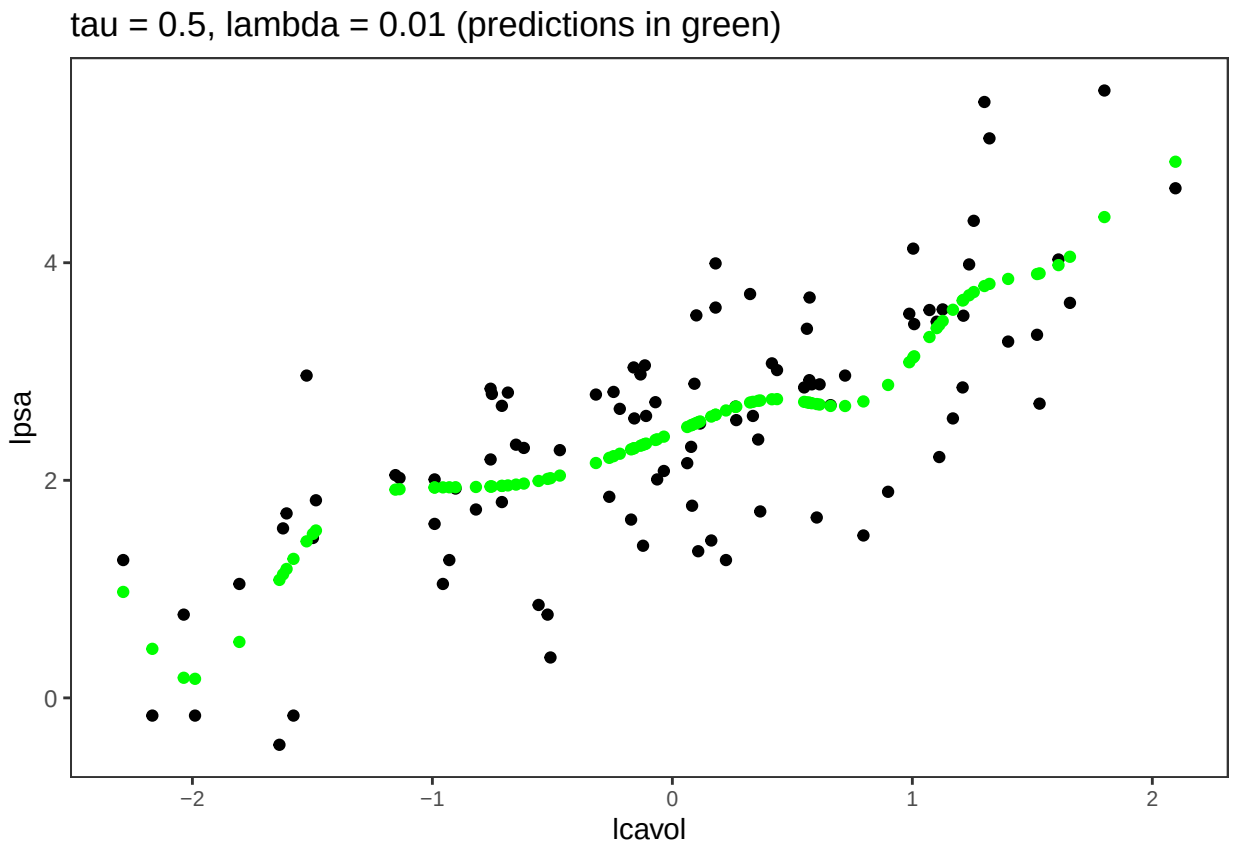


From the 3 plots above, we see that as lambda increases, the predictions move closer to the line of best fit. This is expected, as with increased regularization, variance is sharply decreasing with slow increase in bias, reducing overall MSE.

d) Next, we repeat the experiment above but with tau = 0.5

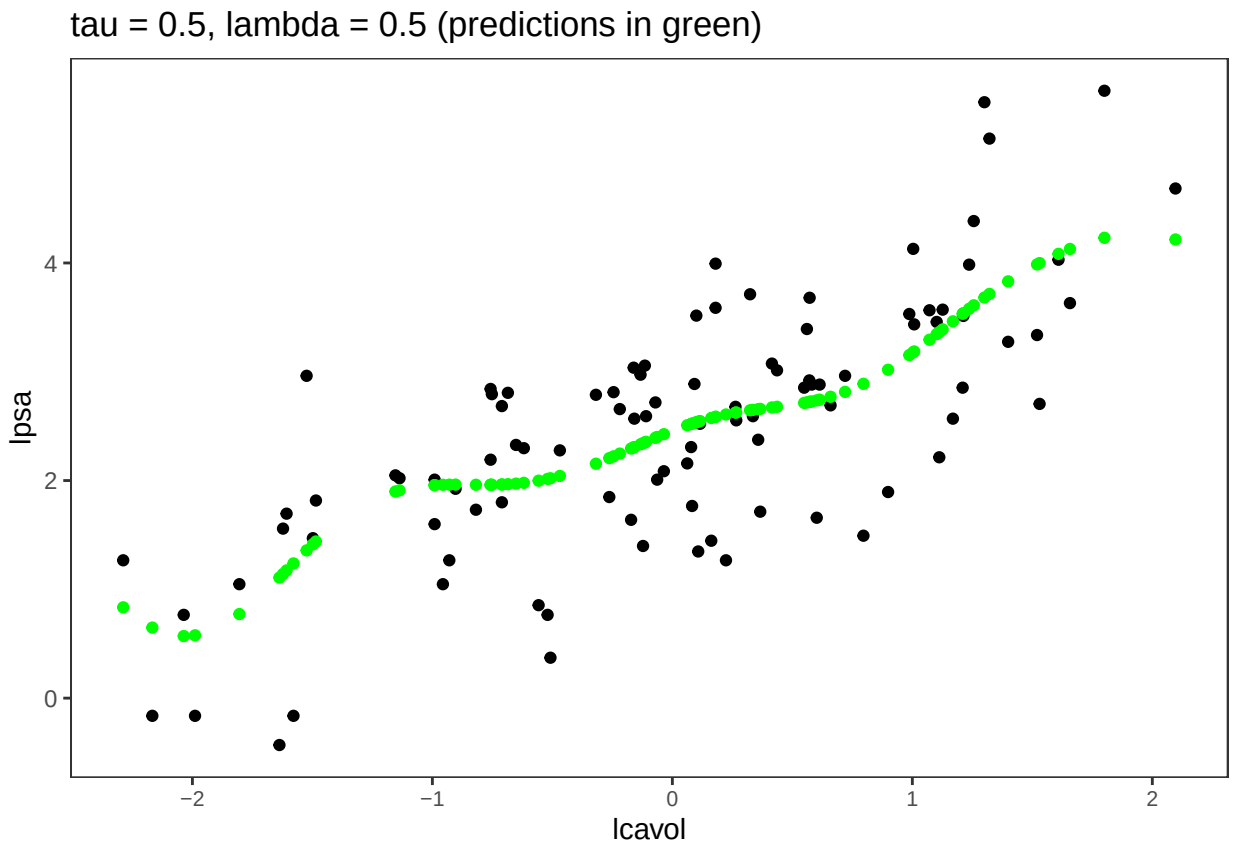
```
y = as.matrix(lprostate_data[,"lpsa"])

X = as.matrix(lprostate_data[,"lcavol"])
X_scaled = scale(X)

alpha_hat_1 <- unlist(fitKRR(X = t(X_scaled), y = y, tau = 0.5, lambda = 0.01)[1])

y_hat_1 <- predictKRR(X = t(X_scaled), Z = t(X_scaled), alpha = alpha_hat_1,
                      tau = 0.5, offset = mean(y))
```

```
plot_data_1 <- as.data.frame(cbind(X_scaled, y, y_hat_1))

ggplot(plot_data_1,
       aes(x = V1, y = V2)) +
  geom_point(aes(x = V1, y = V2))+
  geom_point(aes(x = V1,
                 y = y_hat_1), color = "green")+
  xlab("lcavol") +
  ylab("lpsa") +
  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("tau = 0.5, lambda = 0.01 (predictions in green)")
```



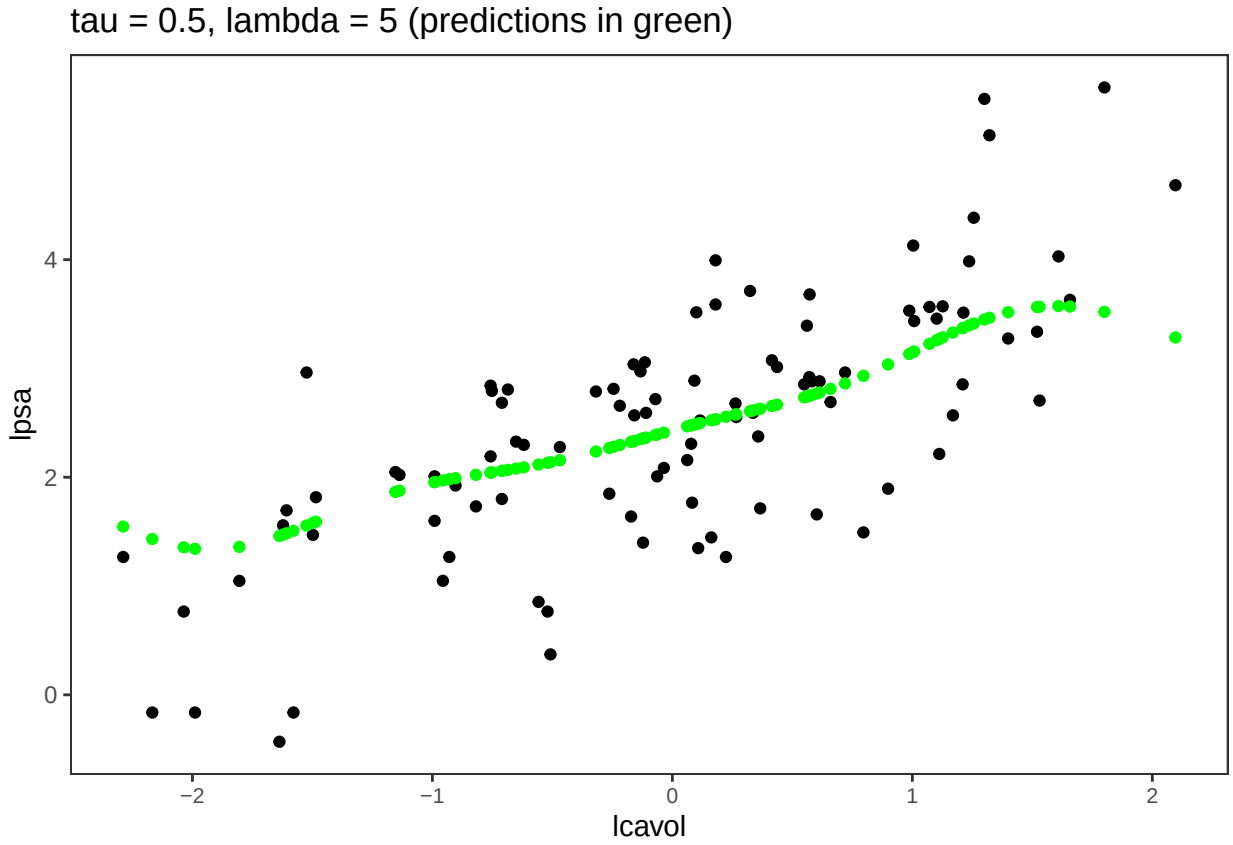

```
y = as.matrix(lprostate_data[,"lpsa"])

X = as.matrix(lprostate_data[,"lcavol"])
X_scaled = scale(X)

alpha_hat_1 <- unlist(fitKRR(X = t(X_scaled), y = y, tau = 0.5, lambda = 0.5)[1])

y_hat_1 <- predictKRR(X = t(X_scaled), Z = t(X_scaled), alpha = alpha_hat_1,
                      tau = 0.5, offset = mean(y))

plot_data_1 <- as.data.frame(cbind(X_scaled, y, y_hat_1))
```

```
ggplot(plot_data_1,
       aes(x = V1, y = V2)) +
  geom_point(aes(x = V1, y = V2))+
  geom_point(aes(x = V1,
                 y = y_hat_1), color = "green")+
  xlab("lcavol") +
  ylab("lpsa") +
  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("tau = 0.5, lambda = 0.5 (predictions in green)")
```




```
y = as.matrix(lprostate_data[,"lpsa"])

X = as.matrix(lprostate_data[,"lcavol"])
X_scaled = scale(X)

alpha_hat_1 <- unlist(fitKRR(X = t(X_scaled), y = y, tau = 0.5, lambda = 5)[1])

y_hat_1 <- predictKRR(X = t(X_scaled), Z = t(X_scaled), alpha = alpha_hat_1,
                      tau = 0.5, offset = mean(y))

plot_data_1 <- as.data.frame(cbind(X_scaled, y, y_hat_1))

ggplot(plot_data_1,
```

```
      aes(x = V1, y = V2)) +
  geom_point(aes(x = V1, y = V2))+
  geom_point(aes(x = V1,
                 y = y_hat_1), color = "green")+
  xlab("lcavol") +
  ylab("lpsa") +
  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("tau = 0.5, lambda = 5 (predictions in green)")
```

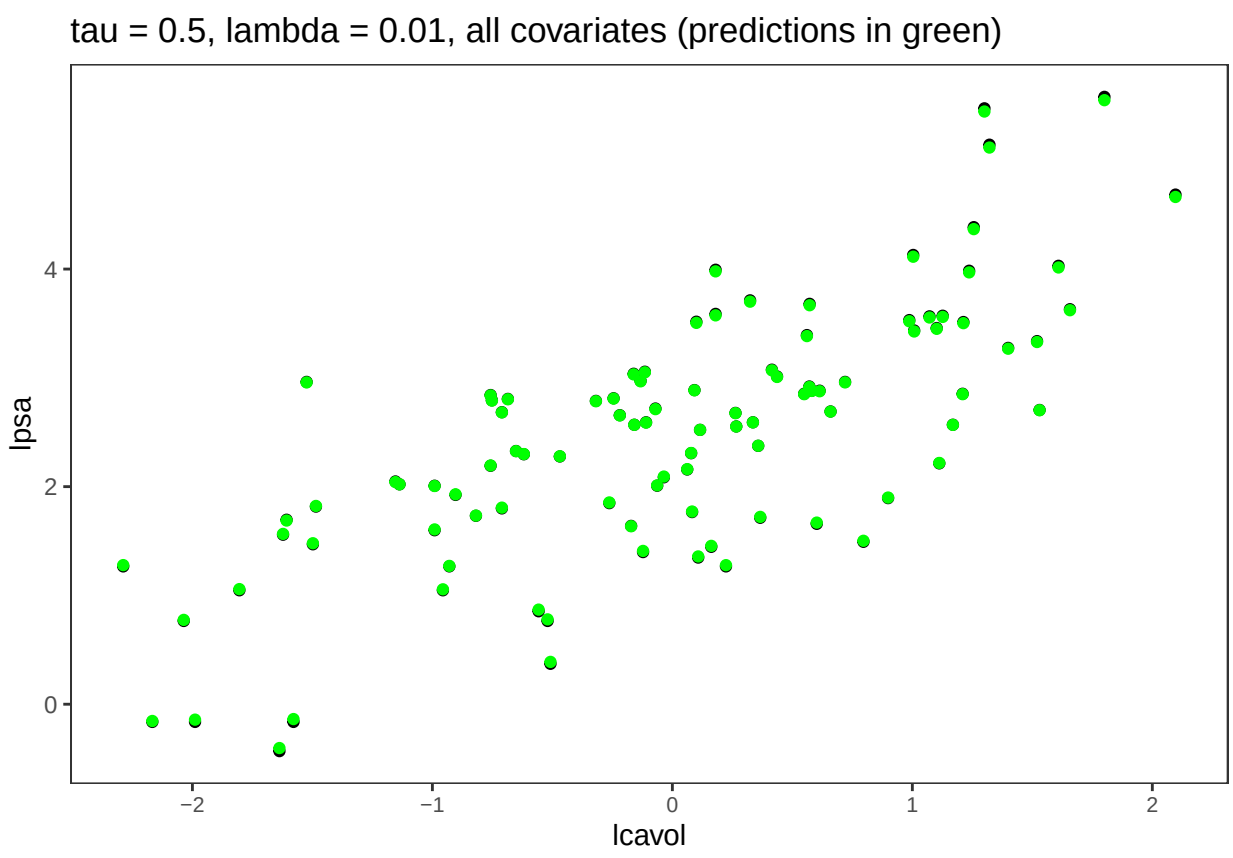


The same phenomenon as in part c) is observable here as well but to a lesser extent. This is because with tau = 0.5, we allow for a less flexible fit, and this also aligns with problem 2d) where we showed that as tau goes to 0, the predictions increasingly become the response values themselves. So as we moved from tau = 0.1 to tau = 0.5, we moved away from predictions tending to be the data points, and to a less flexible fit, reducing overfitting.

e) Now, we use all covariates to predict lpsa.

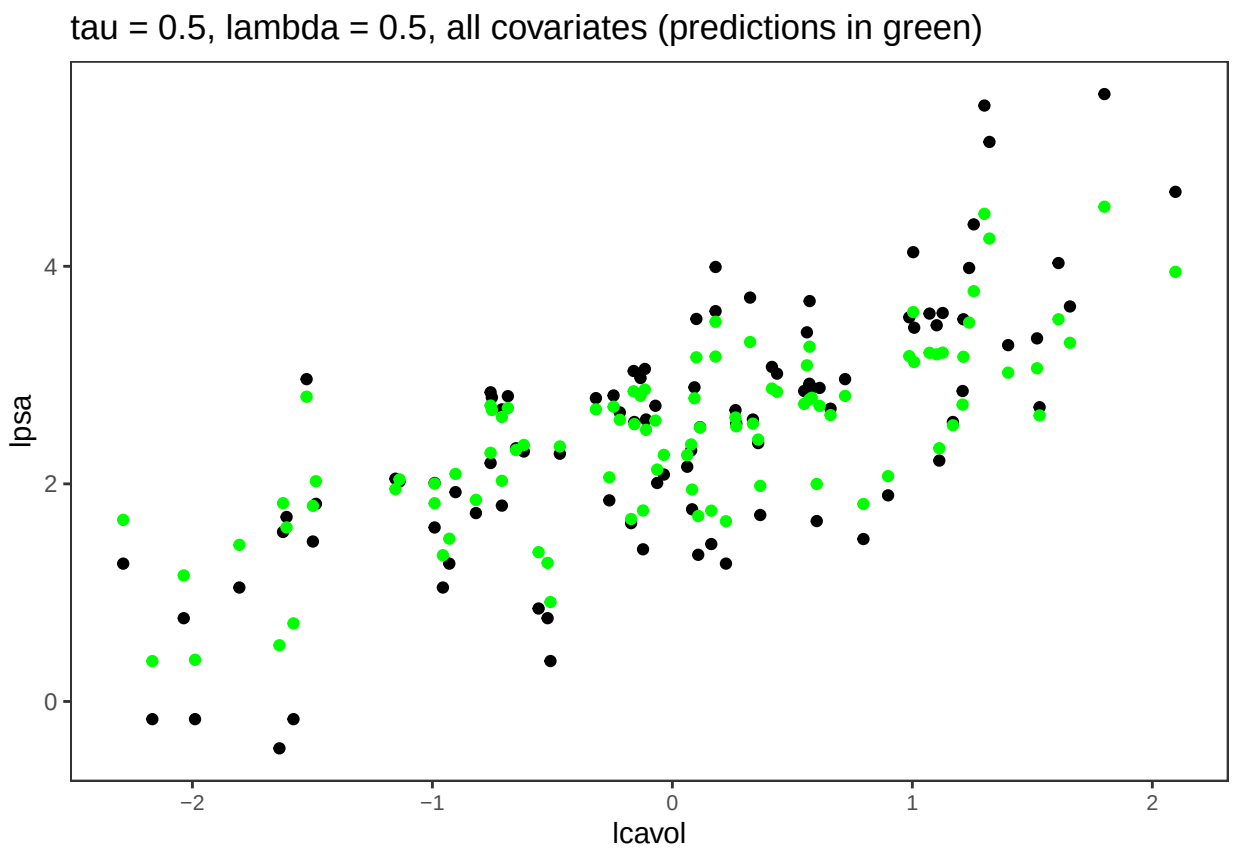
```
y = as.matrix(lprostate_data[,"lpsa"])

X = as.matrix(lprostate_data[,-9])
X_scaled = scale(X)

alpha_hat_1 <- unlist(fitKRR(X = t(X_scaled), y = y, tau = 0.5, lambda = 0.01)[1])

y_hat_1 <- predictKRR(X = t(X_scaled), Z = t(X_scaled), alpha = alpha_hat_1,
```

```
                      tau = 0.5, offset = mean(y))

plot_data_1 <- as.data.frame(cbind(X_scaled[,1], y, y_hat_1))

ggplot(plot_data_1,
       aes(x = V1, y = V2)) +
  geom_point(aes(x = V1, y = V2))+
  geom_point(aes(x = V1,
                 y = y_hat_1), color = "green")+
  xlab("lcavol") +
  ylab("lpsa") +
  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("tau = 0.5, lambda = 0.01, all covariates (predictions in green)")
```



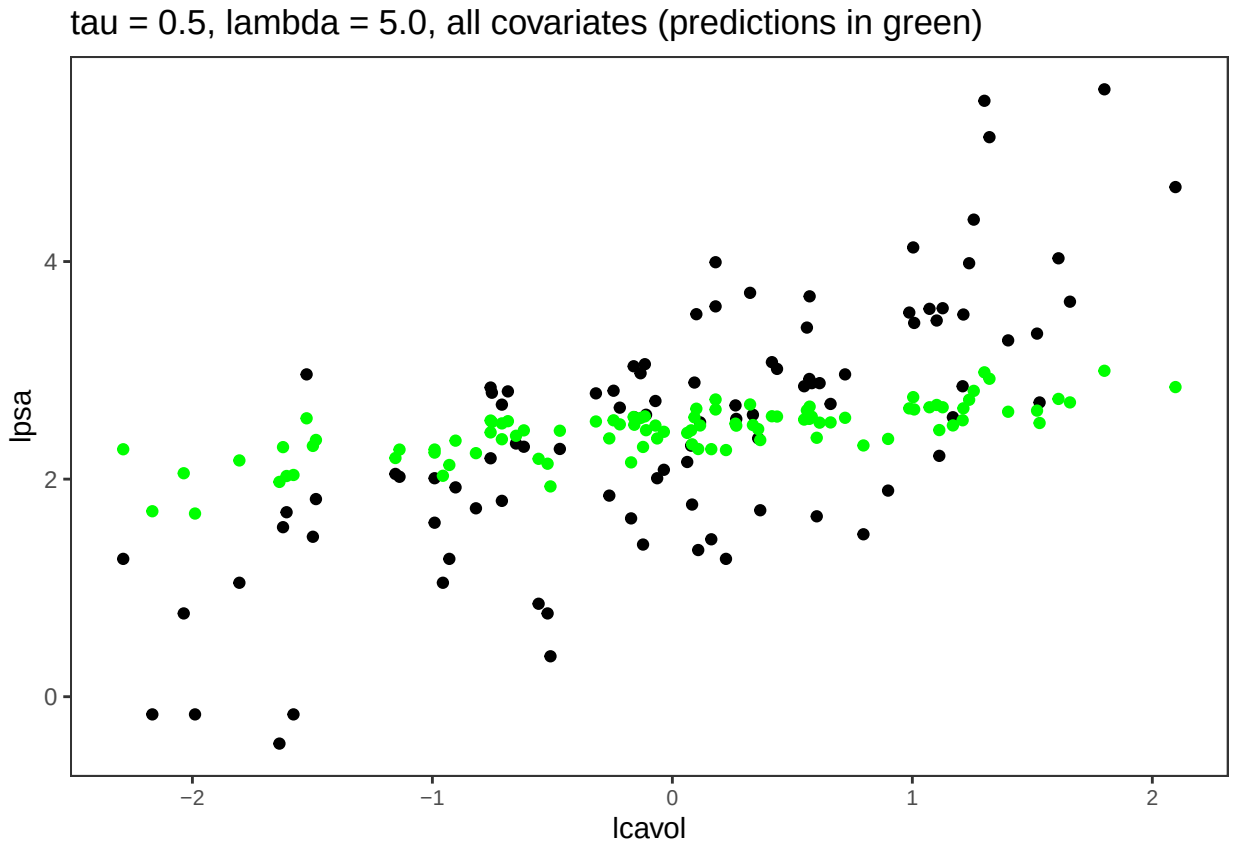

```
y = as.matrix(lprostate_data["lpsa"])

X = as.matrix(lprostate_data[,-9])
X_scaled = scale(X)

alpha_hat_1 <- unlist(fitKRR(X = t(X_scaled), y = y, tau = 0.5, lambda = 0.5)[1])

y_hat_1 <- predictKRR(X = t(X_scaled), Z = t(X_scaled), alpha = alpha_hat_1,
                      tau = 0.5, offset = mean(y))
```

```
plot_data_1 <- as.data.frame(cbind(X_scaled[,1], y, y_hat_1))

ggplot(plot_data_1,
       aes(x = V1, y = V2)) +
  geom_point(aes(x = V1, y = V2))+
  geom_point(aes(x = V1,
                 y = y_hat_1), color = "green")+
  xlab("lcavol") +
  ylab("lpsa") +
  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("tau = 0.5, lambda = 0.5, all covariates (predictions in green)")
```




```
y = as.matrix(lprostate_data[,"lpsa"])

X = as.matrix(lprostate_data[,-9])
X_scaled = scale(X)

alpha_hat_1 <- unlist(fitKRR(X = t(X_scaled), y = y, tau = 0.5, lambda = 5.0)[1])

y_hat_1 <- predictKRR(X = t(X_scaled), Z = t(X_scaled), alpha = alpha_hat_1,
                      tau = 0.5, offset = mean(y))

plot_data_1 <- as.data.frame(cbind(X_scaled[,1], y, y_hat_1))
```

```
ggplot(plot_data_1,
       aes(x = V1, y = V2)) +
  geom_point(aes(x = V1, y = V2))+
  geom_point(aes(x = V1,
                 y = y_hat_1), color = "green")+
  xlab("lcavol") +
  ylab("lpsa") +
  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("tau = 0.5, lambda = 5.0, all covariates (predictions in green)")
```



From the 3 plots above, with small lambda (0.01), we see classic overfitting, since there is veery little regularization and we are using a complex model with 8 covariates. As regularization increases with lambda values increasing from 0.01 to 0.5 to 5.0, we see that the overfitting reduces and the predictions tend to be closer to the line of best fit. The lambda = 0.5 case seems to be a middle ground where there does not seem to be overfitting, but the model is also not too rigid as seems to be the case slightly with lambda = 5.0.

f) First we calculate the sigma hat squared from the formula given in the homework. To do that, we first have to run the model using all covariates as in the previous part but with lambda = 0.1:

```
y = as.matrix(lprostate_data[,"lpsa"])

X = as.matrix(lprostate_data[,-9])
X_scaled = scale(X)

alpha_hat_1 <- unlist(fitKRR(X = t(X_scaled), y = y, tau = 0.5, lambda = 0.1)[1])
```

```
y_hat_1 <- predictKRR(X = t(X_scaled), Z = t(X_scaled), alpha = alpha_hat_1,
                      tau = 0.5, offset = mean(y))
```

That gives us the y_hat's needed. Next we need the H_lambda matrix:

```
G <- gram_matrix(t(X_scaled), bandwidth = 0.5)

id <- diag(rep(1, dim(G)[1]))
inv_arg <- G + 0.1 * id
inv <- solve(inv_arg)

H_lambda <- G %*% inv
H_lambda_squared <- H_lambda %*% H_lambda
```

Now, we are ready to calculate sigma_hat_squared:

```
diff <- y_hat_1 - y
norm_sq <- sum(diff^2)
n <- dim(G)[1]

denom <- n - 2*tr(H_lambda) + tr(H_lambda_squared)

sigma_hat_squared <- (1/denom)*norm_sq

sigma_hat_squared
```

```
## [1] 0.62937
```

We output the sigma_hat_squared value as above.

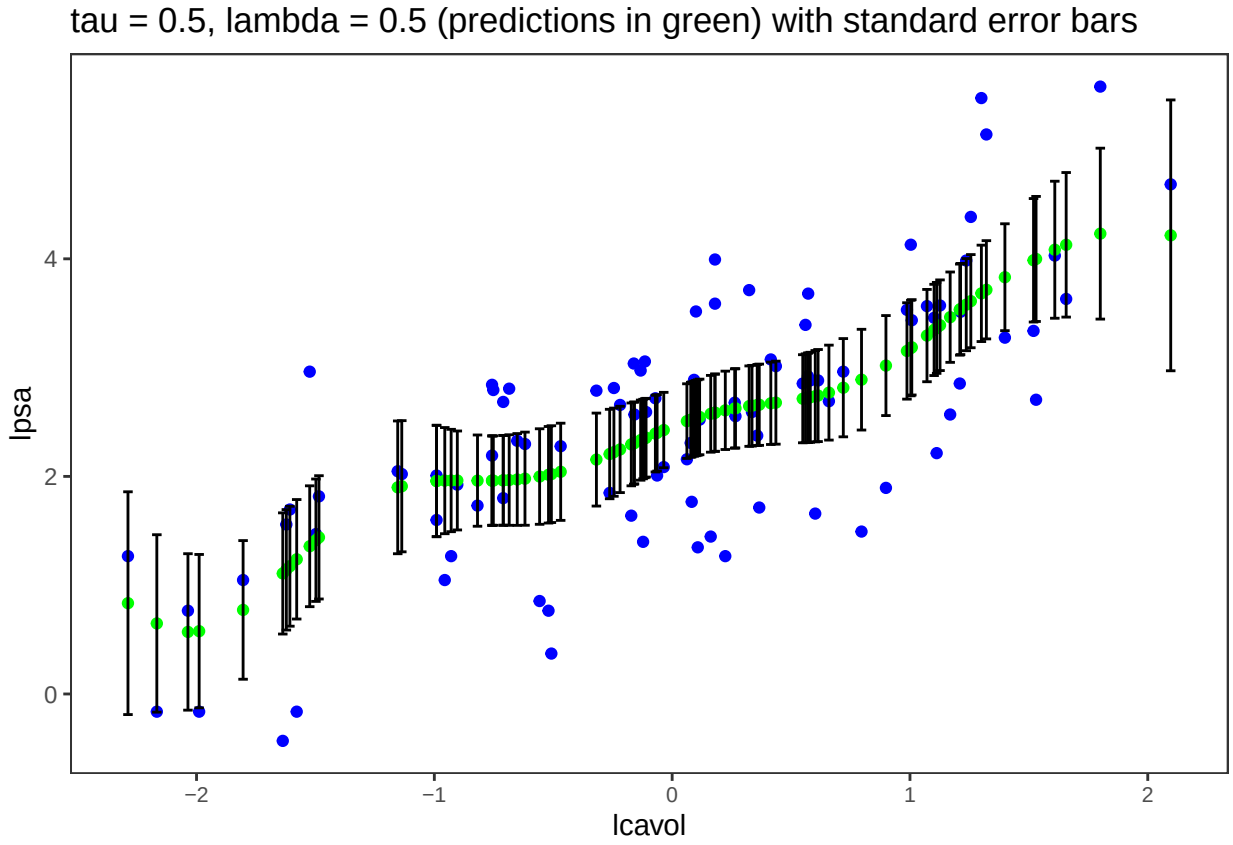
Next, we fit a one-covariate model again, as in part d):

```
y = as.matrix(lprostate_data[,"lpsa"])

X = as.matrix(lprostate_data[,"lcavol"])
X_scaled = scale(X)

alpha_hat_1 <- unlist(fitKRR(X = t(X_scaled), y = y, tau = 0.5, lambda = 0.5)[1])

y_hat_1 <- predictKRR(X = t(X_scaled), Z = t(X_scaled), alpha = alpha_hat_1,
                      tau = 0.5, offset = mean(y))
```

Now, to estimate the standard error of the predictions, we use the expression $Var(\hat{y}) = Var(H_\lambda y) = H_\lambda Var(y) H_\lambda^T$. We already know that the Variance of y is σ^2, which we have already estimated from the full-model with all covariates. The standard error is the square root of the variance of y_hat. So we can perform the calculation now:

```
G <- gram_matrix(t(X_scaled), bandwidth = 0.5)

id <- diag(rep(1, dim(G)[1]))
inv_arg <- G + 0.1 * id
inv <- solve(inv_arg)

H_lambda <- G %*% inv

HHT <- H_lambda %*% t(H_lambda)
```

```
std_errors <- sqrt(sigma_hat_squared * diag(HHT))

plot_data_1 <- as.data.frame(cbind(X_scaled[,1], y, y_hat_1))

ggplot(plot_data_1,
       aes(x = V1, y = V2)) +
  geom_point(aes(x = V1, y = V2), color = "blue")+
  geom_point(aes(x = V1,
                 y = y_hat_1), color = "green")+
  geom_errorbar(aes(ymin = y_hat_1 - 2*std_errors,
                    ymax = y_hat_1 + 2*std_errors), width = 0.04)+
  xlab("lcavol") +
  ylab("lpsa") +
  theme_bw() +
  theme(axis.text.x=element_text(size=8),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank())+
  ggtitle("tau = 0.5, lambda = 0.5 (predictions in green) with standard error bars")
```



In the plot above, the blue points are the training data points, the green points are the predictions, and the error bars are calculated using 2 times the estimated standard errors. Details of the estimated standard errors calculation is given above.