HA-1

Ishwara Hegde, Jonathan Nieman and Aleksei Samkov

10/3/2020

Preamble

Code chunk loads all the necessary packages and sets the Markdown structure.

Problem 1.3: Lasso vs OLS

Part 1

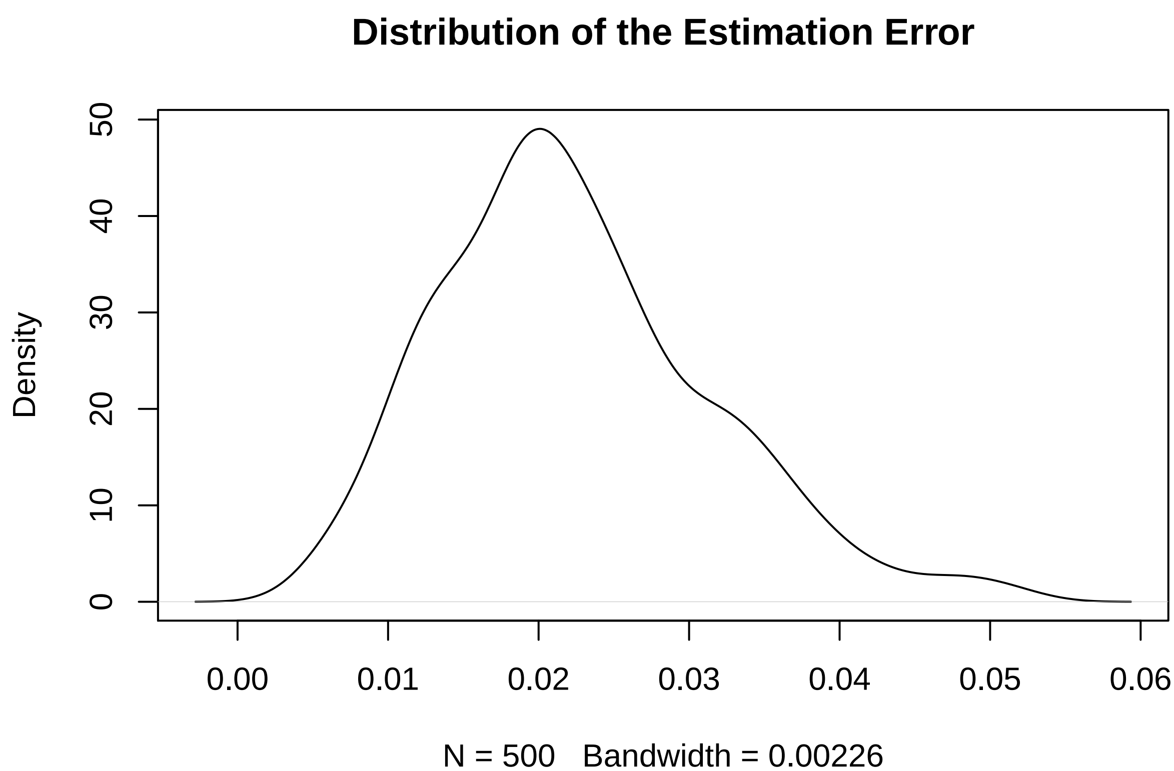
```
#Calibration

n<-500
rho <- 0.5
p <-11
B<-500

#Generating the data
Sigma <- outer(1:p,1:p,FUN=function(x,y) rho^(abs(x-y)))
X<-rmvnorm(n,sigma=Sigma,method = "chol")

beta <- matrix(c(5,rep(1,10)),nrow=p,ncol=1)
beta_one <- matrix(c(5,rep(1,10)),nrow=11,ncol=1)

epsilon <-rnorm(n,mean=0,sd=1)

Y <-X%*%beta +epsilon
```

```
#performing OLS B times

#The "tidy" way to run OLS:


# I exclude the intercept here cause I include it in definition of beta
OLS<-lm(Y~0+X)
OLS<-tidy(OLS)

beta_hat<-OLS$estimate
#Estimation error

error<- t(beta_hat-beta_one)%*%solve(cov(X))%*%(beta_hat-beta_one)
```

Now I just run a loop to simulate it 500 times.Note, I use the OLS formula below and not the lm model just for variety.

```
est_err<-matrix(,nrow=B,ncol=1)
for (i in 1:B) {
        Sigma <- outer(1:p,1:p,FUN=function(x,y) rho^(abs(x-y)))
        X<-rmvnorm(n,sigma=Sigma,method = "chol")

        epsilon <-rnorm(n,mean=0,sd=1)

        Y <- X%*%beta +epsilon
        beta_hat<-solve((t(X)%*%X)) %*% (t(X)%*%Y)
       est_err[i,1]<- t(beta_hat-beta_one)%*%cov(X)%*%(beta_hat-beta_one)

}
```

####Plot



This resembles a chi-square distribution as expected.

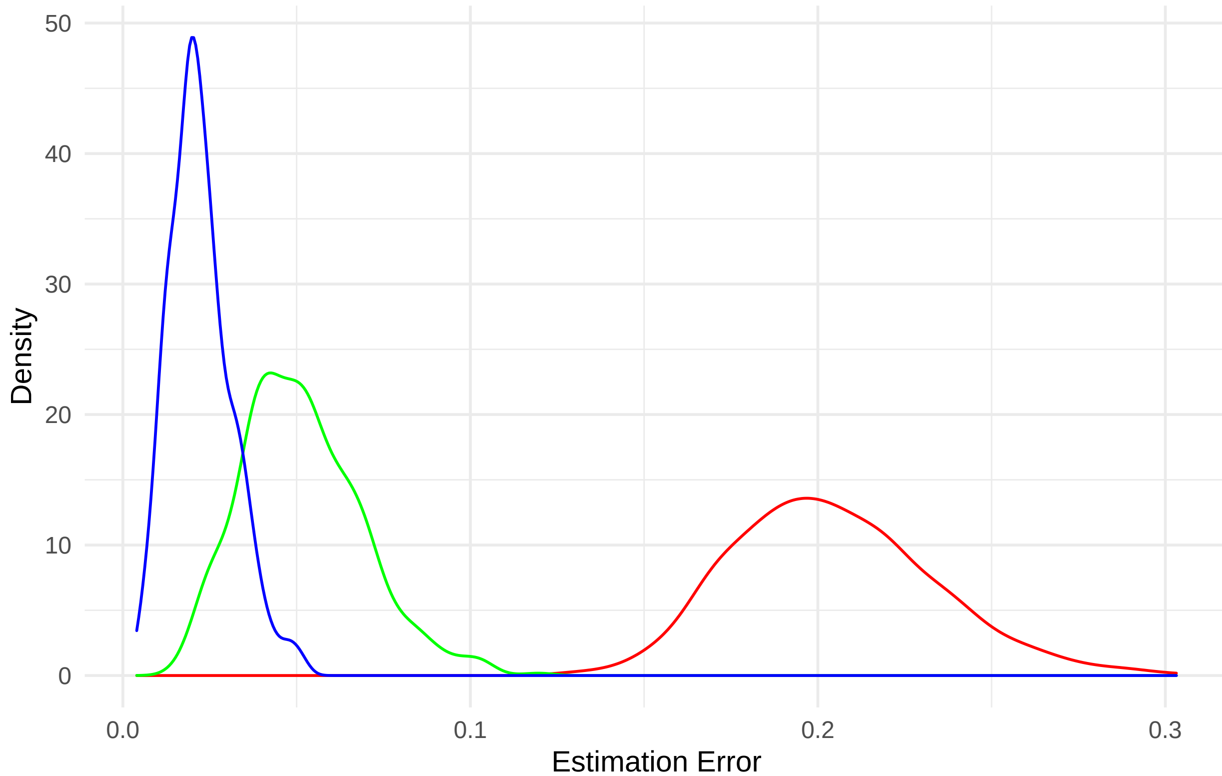
Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Part 2

```
p<- 101
rho<-0.5
beta<-rbind(beta_one,matrix(0,nrow=p-11,ncol=1))

est_err_ols<-matrix(,nrow=B,ncol=1)
```

```r
est_err_lasso<-matrix(,nrow=B,ncol=1)
best_lambdas<-matrix(,nrow=B,ncol=1)
for (i in 1:B) {

        Sigma <- outer(1:p,1:p,FUN=function(x,y) rho^(abs(x-y)))
        X<-matrix(rmvnorm(n,sigma=Sigma,method = "chol"),nrow=500,ncol= p)


        epsilon <-rnorm(n,mean=0,sd=1)

        Y <- X%*%beta +epsilon
        beta_hat_ols<-solve((t(X)%*%X)) %*% (t(X)%*%Y)
        lasso_model<-cv.glmnet(X,Y,alpha = 1,nfolds = 5,intercept=FALSE)

        #choose the lambda that minimizes MSE
        best_lambdas[i,1] <- lasso_model$lambda.min

        #Now I run LASSSO with this lambda

        beta_hat_lasso<- predict(lasso_model, s = best_lambdas[i,1],newx = X,
                                  type="coefficients",intercept=FALSE)

        est_err_ols[i,1]<- t(beta_hat_ols-beta)%*%cov(X)%*%(beta_hat_ols-beta)
        est_err_lasso[i,1]<- t(beta_hat_lasso[2:102,]-beta)%*%cov(X)%*%
                (beta_hat_lasso[2:102,]-beta)

}
```

```r
combined_err<-tibble(ten_dim=(est_err),hund_dim_ols=(est_err_ols)
                     ,hund_dim_lasso = (est_err_lasso))
p<-combined_err %>% ggplot()+
        geom_density(mapping = aes(hund_dim_ols),color="red")+
        geom_density(mapping = aes(hund_dim_lasso),color="green")+
        geom_density(mapping=aes(ten_dim),color="blue")
p+labs(x = "Estimation Error", y = "Density",caption = "red:OLS p=100,
       green:lasso, blue:OLS p=10")+theme_minimal()
```



red:OLS p=100,
green:lasso, blue:OLS p=10

From the graph above it seems that the low dimensional OLS performs best. It has lower bias and variance than the othwer two. However, the fact that LASSO has a lower bias and variance indicates that OLS (p=100) might be overfitting in this case.

Another explanation could be that since there is some correlation between features we can drop a few and improve performance. We can check this logic in part 4.

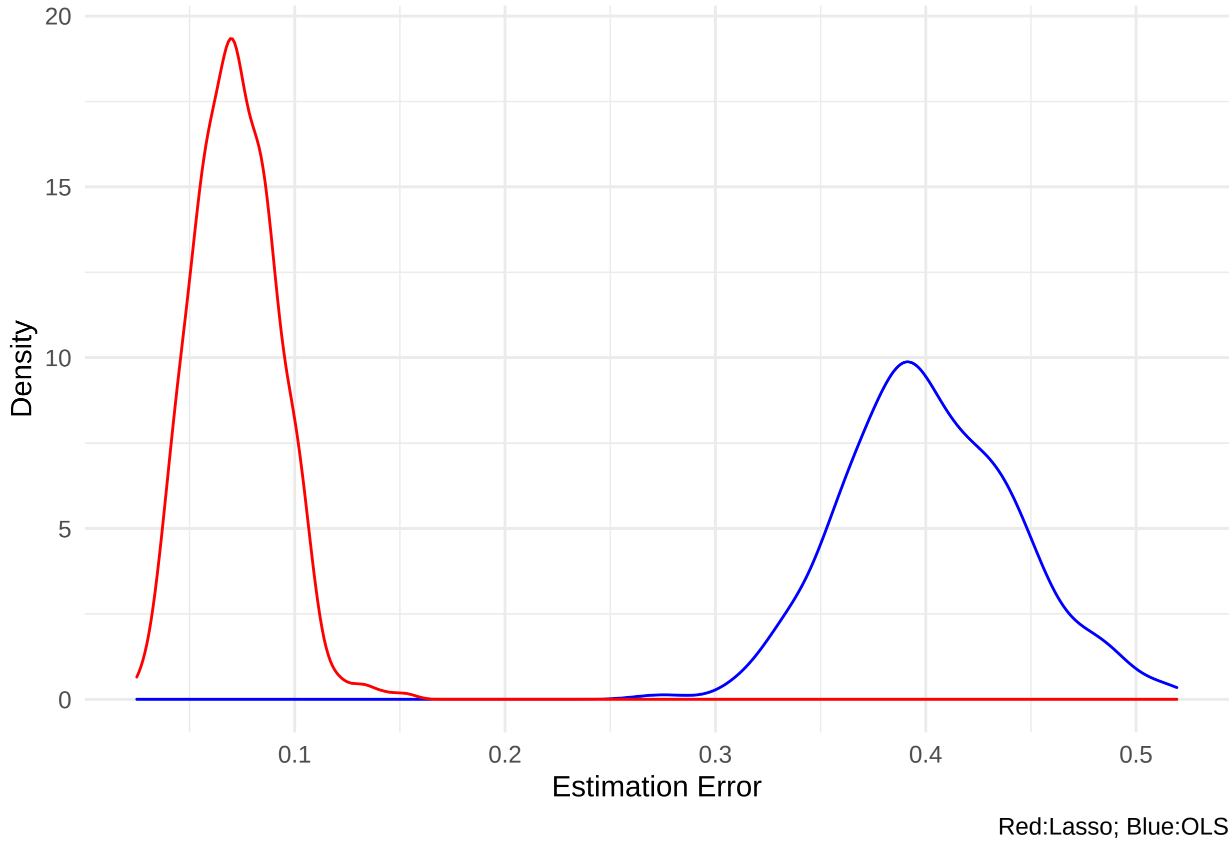
Part 3

Only the recaliberation part changes, rest stays the same as part 2

```
#Recaliberation
p=201
rho=0.5

# Creating the sequence
seq=matrix(,nrow = p-11,ncol=1)
for (i in 1:p-11) {seq[i,1]=(1/(2^i))}
beta_three<-rbind(beta_one,seq)

#
est_err_ols_three<-matrix(,nrow=B,ncol=1)
est_err_lasso_three<-matrix(,nrow=B,ncol=1)
best_lambdas_two<-matrix(,nrow=B,ncol=1)
for (i in 1:B) {

        Sigma <- outer(1:p,1:p,FUN=function(x,y) rho^(abs(x-y)))
        X<-matrix(rmvnorm(n,sigma=Sigma,method = "chol"),nrow=500,ncol=p)
```

```r
        epsilon <-rnorm(n,mean=0,sd=1)

        Y <- X%*%beta_three +epsilon
        beta_hat_ols<-solve((t(X)%*%X)) %*% (t(X)%*%Y)
        lasso_model<-cv.glmnet(X,Y,alpha = 1,nfolds = 5,intercept=FALSE)

        #choose the lambda that minimizes MSE
        best_lambdas_two[i,1] <- lasso_model$lambda.min

        #Now I run LASSSO with this lambda

        beta_hat_lasso<- predict(lasso_model, s = best_lambdas[i,1],newx = X,
                                 type="coefficients",intercept=FALSE)

        est_err_ols_three[i,1]<-
                t(beta_hat_ols-beta_three)%*%cov(X)%*%(beta_hat_ols-beta_three)
        est_err_lasso_three[i,1]<-
                t(beta_hat_lasso[2:202,]-beta_three)%*%cov(X)%*%
                (beta_hat_lasso[2:202,]-beta_three)

}
```

```r
combined_err_two<-tibble(hund_dim_ols=est_err_ols_three,hund_dim_lasso
                    =est_err_lasso_three)
p<-combined_err_two %>% ggplot()+geom_density(mapping=aes(hund_dim_ols),
                                              color="blue")+
        geom_density(mapping = aes(hund_dim_lasso),color="red")
p+labs(x = "Estimation Error", y = "Density",caption ="Red:Lasso; Blue:OLS" )+
        theme_minimal()
```



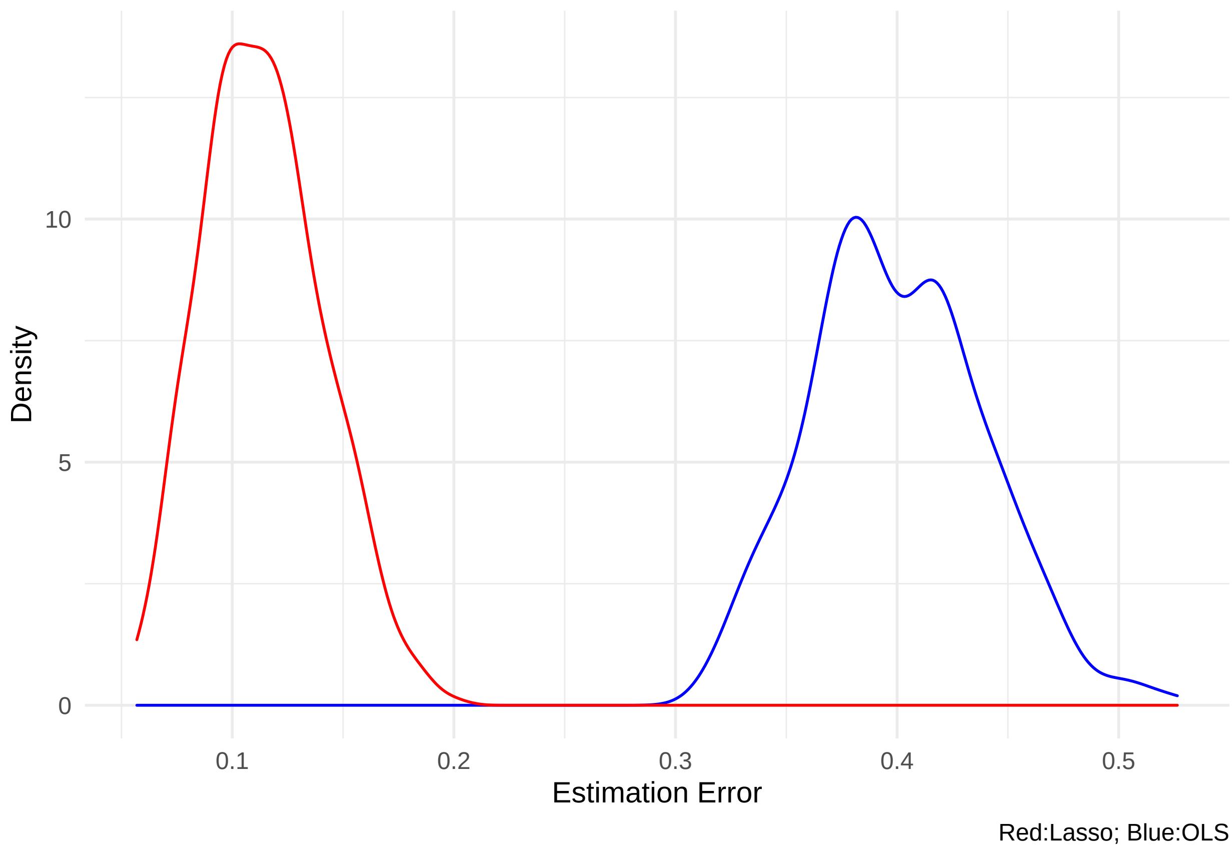
Again, the OLS performs worse than LASSO. I speculate it is for the same reason as before.

Part Four

```
#Recaliberation
p=201
rho=0

# Creating the sequence
seq=matrix(,nrow = p-11,ncol=1)
for (i in 1:p-11) {seq[i,1]=(1/(2^i))}
beta_three<-rbind(beta_one,seq)

#
est_err_ols_four<-matrix(,nrow=B,ncol=1)
est_err_lasso_four<-matrix(,nrow=B,ncol=1)
best_lambdas_three<-matrix(,nrow=B,ncol=1)
for (i in 1:B) {

        Sigma <- outer(1:p,1:p,FUN=function(x,y) rho^(abs(x-y)))
        X<-matrix(rmvnorm(n,sigma=Sigma,method = "chol"),nrow=500,ncol=p)


        epsilon <-rnorm(n,mean=0,sd=1)

        Y <- X%*%beta_three +epsilon
```

```
        beta_hat_ols<-solve((t(X)%*%X)) %*% (t(X)%*%Y)
        lasso_model<-cv.glmnet(X,Y,alpha = 1,nfolds = 5,intercept=FALSE)

        #choose the lambda that minimizes MSE
        best_lambdas_three[i,1] <- lasso_model$lambda.min

        #Now I run LASSSO with this lambda

        beta_hat_lasso<- predict(lasso_model, s = best_lambdas[i,1],newx = X,
                                 type="coefficients",intercept=FALSE)
        est_err_ols_four[i,1]<-
                t(beta_hat_ols-beta_three)%*%cov(X)%*%(beta_hat_ols-beta_three)

        est_err_lasso_four[i,1]<-
                t(beta_hat_lasso[2:202,]-beta_three)%*%cov(X)%*%
                (beta_hat_lasso[2:202,]-beta_three)

}
```

```
combined_err_three<-tibble(hund_dim_ols=est_err_ols_four,hund_dim_lasso
                    =est_err_lasso_four)
p<-combined_err_three %>% ggplot()+geom_density(mapping=aes(hund_dim_ols),
                                             color="blue")+
        geom_density(mapping = aes(hund_dim_lasso),color="red")
p+labs(x = "Estimation Error", y = "Density",caption ="Red:Lasso; Blue:OLS" )+
        theme_minimal()
```



What is perhaps a bit surprising is that LASSO performs better even in the 4th case when the features in X are not correlated. It maybe then that the penalization is reducing overfitting and not correcting for multicollinearity.