Assignment 2: Lasso Regression

Ian Wallgren, Wanchang Zhang, Lavinia Hriscu, Victor Jimenez

Contents

In this assignment we will fit our data to a multiple linear regression model by means of a lasso regression. Considering n pairs of observations (x_i, y_i), where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$, the linear model is the following:

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \epsilon_i$$

where ϵ_i account for white noise and $\beta \in \mathbb{R}^{p+1}$ is the vector of coefficients of the model. By fitting the model, estimators for β and σ^2 will be obtained and the object variables y will be predicted from explanatory variables x as $\hat{y}_i = x_i^T \hat{\beta}$.

In general (OLS), the estimation of the coefficients is achieved by minimizing the square of the prediction error ϵ. It can be proven (Gauss-Markov) that this method provides the MVUE estimator of β. In the case of lasso (Least absolute shrinkage and selection operator) regression, the penalization term that imposes a shrinkage in the estimated parameter vector $\hat{\beta}$ is the L_1-norm of the vector:

$$\hat{\beta}_{lasso} = \arg\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

The greater the value of the penalization parameter $\lambda \geq 0$, the greater the amount of shrinkage of the estimator $\hat{\beta}_{lasso}$. Unlike ridge regression, where the penalty term was $\sum_{j=1}^{p} \beta_j^2$ and the optimal parameter vector was the one that minimised the magnitude of its coefficients, the minimization problem in lasso regression allows $\hat{\beta}_{lasso}$ to be sparse, that is, to have certain number of components reduced to zero, depending on the magnitude of the shrinkage parameter λ. This difference, which can be studied in general for penalization terms using the L_q-norm, comes from the fact that the feasible region in the parameter space for ridge regression is a centered hyper-sphere, whereas in lasso regression is a centered hyper-cube with vertex aligned with the axis, meaning that the intersection between level curves of the objective function and the border of the feasible region (where the optimal lays) happens when some components are zero. L_1-norm is the lowest-order norm that allows the minimization problem to be convex.

In some sense, the sparsity provided by the lasso estimation is more flexible than a possible OLS estimation with some coefficients set to zero, since the optimal obtained in lasso selects the best subset of variables to

nullify. However, it must be taken into account that lasso optimization shrinks the magnitude of the non-zero coefficients as well, and for that reason OLS presents higher flexibility in those, allowing for a more accurate result (in the sense of presenting a lower error). All in all, a flexibility trade-off takes place between lasso and OLS regression.

In contrast to ridge regression, the computation of $\hat{\beta}_{lasso}$ cannot be reduced to an explicit expression, and the cyclic coordinate descent algorithm is used instead. This algorithm converges to the global minimum by iterating:

$$\hat{\beta}_j^{new} = S_\lambda\left(\hat{\beta}_j + \frac{1}{n}\langle x_j, r\rangle\right), \qquad r^{new} = r - \left(\hat{\beta}_j^{new} - \hat{\beta}_j\right) x_j$$

where $S_\lambda(x) = sign(x)(|x| - \lambda)$ is the self-thresholding operator. The `glmnet` R package implements this method to solve a similar minimization problem that allows for a mixture between L_1- and L_2-norm penalization terms, thus balancing the shrinkage and sparsity of the solution. The elastic net-penalty parameter `alpha` controlling this mixture will be set to 1 (actually its default value) to obtain lasso regression.

1. Lasso for the Boston Housing data

1.1 Fitting a lasso regression model for `CMEDV`

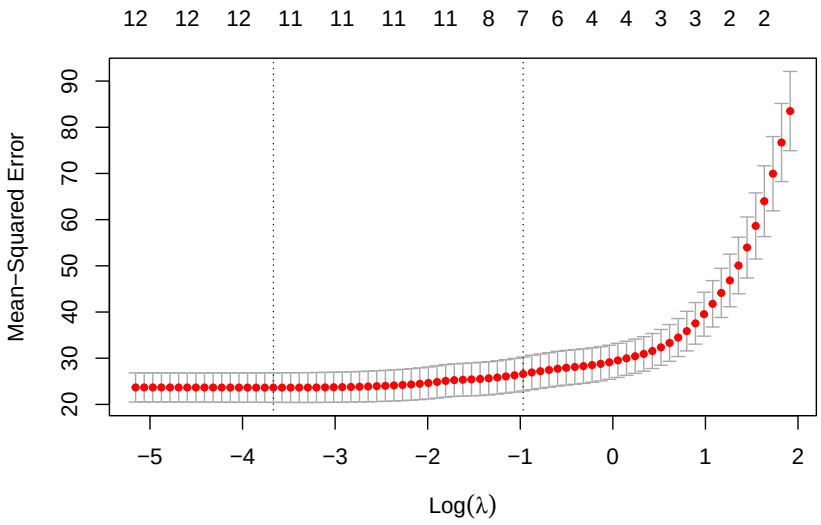
In this first exercise we will work with the `boston` dataset that was used in the previous assignment, which contains information about the housing market in the city of Boston. 13 explanatory variables will be assigned a coefficient in the parameter vector to predict `CMEDV`, the corrected version of the median value of owner-occupied homes. First, we will obtain the matrices of observations as follows:

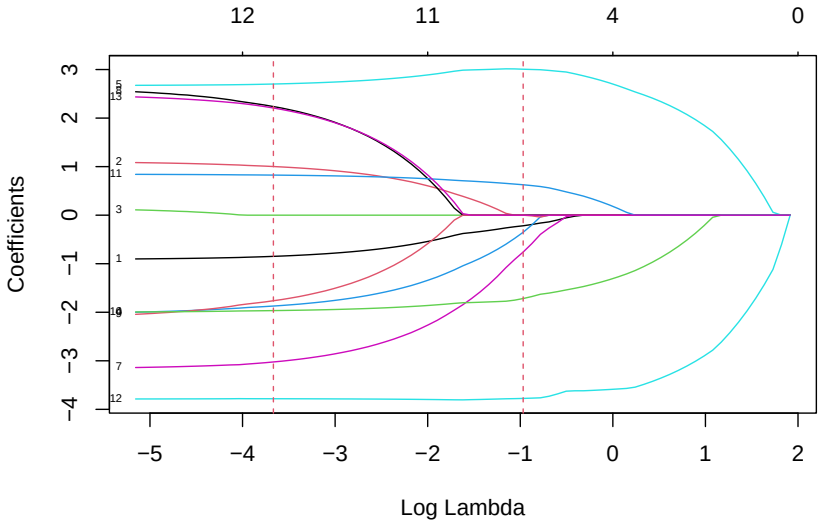
```
load("boston.Rdata")

#Select explanatory and response variables
boston = boston.c[c("CRIM","ZN","INDUS","CHAS","NOX","RM","AGE","DIS",
                    "RAD","TAX","PTRATIO","B","LSTAT","CMEDV")]

Y = scale(boston$CMEDV, center=TRUE, scale=FALSE) # scale target var
X1 = scale(boston[,c(-4,-14)], center=TRUE, scale=TRUE)  # remove CHAS and scale expl.
X = cbind(X1, as.numeric(boston$CHAS)-1) # add CHAS after scaling
colnames(X) = c(colnames(X1), "CHAS")
p = dim(X)[2] # 13 explanatory
n = dim(X)[1] # 506 observations
```

Notice that since `CHAS` is a factor (binary) variable, it has not been taken into account in the scale of the dataset. Now we can work with the regression model and perform K-fold cross-validation to tune the parameter λ. By default, `glmnet` considers $K = 10$ sets and 100 candidate λ values.

```
lasso.1 <- glmnet(X,Y, standardize=FALSE, intercept=FALSE)
cv.lasso.1 <- cv.glmnet(X,Y, standardize=FALSE, intercept=FALSE) # cross-validation
```

The choice of the optimal λ is aided by the routine by providing two significant values: `lambda.min` and `lambda.1se`. The first one corresponds to $\lambda_{min} = \arg\min_\lambda MSE(\lambda)$, the value that minimizes the mean-squared error, and the second one is λ_σ such that $MSE(\lambda_\sigma) = MSE(\lambda_{min}) + \sigma(MSE(\lambda_{min}))$; that is, the maximum value of λ for which the mean error lays on the range of errors found for λ_{min}. This second parameter is significant in lasso regression, since the shrinkage of the vector is not the main factor determining the suitability of the model, but the number of coefficients that are made zero. In this sense, providing a less sparse vector that behaves almost as good as the sparsest one might be adequate in certain cases.





We observe how the degrees of freedom of the model (i.e. the number of non-zero parameters, shown in the upper row of the $MSE(\lambda)$ plot) decrease significantly around λ_σ from 11 to 8 even if $df(\lambda_\sigma) = 11$ still.

```
pos.lambda.1se = cv.lasso.1$index[1]
lasso.1$df[pos.lambda.1se]
```

```
## [1] 11
```

For that reason, we will fit a regression model for this data using $\lambda_8 = \arg\min_\lambda\{df(\lambda) = 8\}$, from the default set of values of the model, so that we do not have to fit again.

```
lambda.8.pos = which(lasso.1$df == 8)[1]
beta.8 = lasso.1$beta[,lambda.8.pos]
data.frame(parameters = beta.8[which(abs(beta.8)>0)])
```

```
##          parameters
## CRIM    -0.16178896
```

```
## NOX     -0.03264912
## RM       2.99618587
## DIS     -0.39556380
## TAX     -0.03897197
## PTRATIO -1.62897506
## B        0.59015271
## LSTAT   -3.76234385
```

We observe how the model tends to preserve the most significant explanatory variables and set to zero those which do not affect the target in a substantial way. Attributes `LSTAT`, `RM` and `PTRATIO` are the ones that carry most of the weight, as it was obtained in the ridge regression assignment.

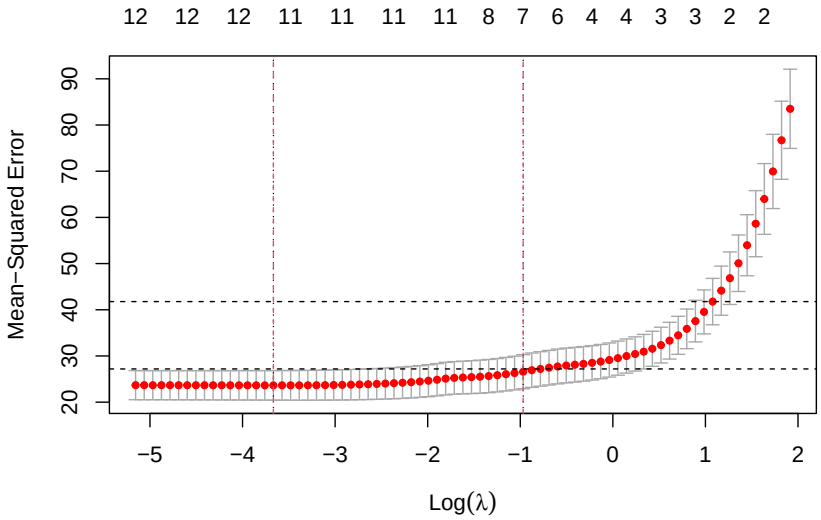
- `LSTAT`: It is negative and it measures the % of lower status of the population. If the value of `LSTAT` is high, it means the purchasing power is lower therefore the home price cost has to be smaller.
- `RM`: It is positive, which makes sense since it corresponds to the average number of rooms per dwelling. The higher is this number, the more likely the home cost will increase.
- `PTRATIO`: It is negative, which means that the higher the cost of the house, the less number of pupils per teacher, which is kind of a privilege.

In contrast with previous results using ridge regression, the `INDUS` variable (accounting for the number of non-retail business acres per town) has been set to zero and `B` (proportional to the percentage of African-american neighbors) carries a higher weight.

In genera, a lasso regression selects the best subset of variables to nullify and we observe in this particular case how `LSTAT`, `PTRATIO` and `RM` carry most of the weight of the vector. For that reason, it is intuitive to believe that a model with only three degrees of freedom will have similar accuracy in the result.

```
lambda.3.pos = which(lasso.1$df == 3)[1]
beta.3 = lasso.1$beta[,lambda.3.pos]
data.frame(parameters = beta.3[which(abs(beta.3)>0)])
```

```
##          parameters
## RM        1.72541011
## PTRATIO -0.04428615
## LSTAT   -2.78221644
```

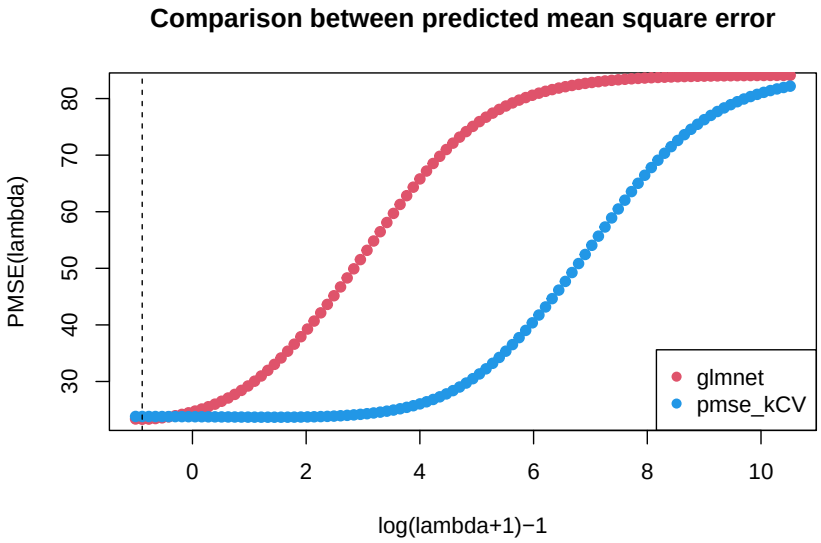


Indeed, the difference in the mean square error is not significant in this region, taking into account that we are able to reduce the degrees of freedom of the model from 8 to 3.

1.2 Fitting a ridge regression model for `CMEDV`

Now we will fit the previous model with ridge regression and compare the 10-fold cross-validation performed by the `glmnet` function with the `pmse_kCV` function we developed for the first assignment. We can perform a ridge regression calling `glmnet` with the parameter `alpha = 0`.

```
lambda.v = exp(seq(0,log(1e5+1),length=100))-1 # vector of lambdas to compare

ridge.1 = glmnet(X,Y, standardize=FALSE, intercept=FALSE, alpha = 0) # ridge regression
cv.ridge.1 = cv.glmnet(X,Y, standardize=FALSE, intercept=FALSE,
                       alpha = 0, type.measure = "mse", lambda = lambda.v)
pmse.lambda.1 = cv.ridge.1$cvm
pmse.lambda.2 = pmse_kCV(X, Y, lambda.v, 10) # using the same lambdas
```



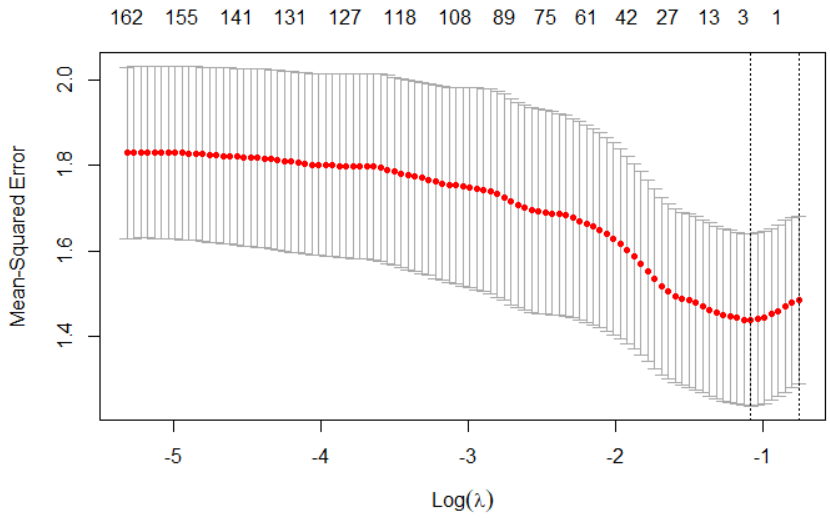
We see how the $PMSE(\lambda)$ is calculated in a different way between in both functions but its evolution for λ is similar, even though they do not reach the same minimum.

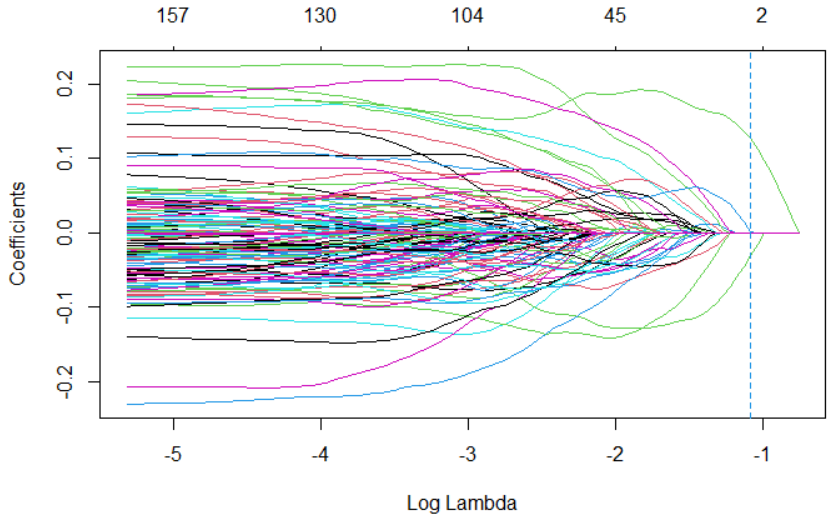
2. A regression model with $p \gg n$

Now we will use a different dataset, this one consisting of $n = 240$ samples from patients with diffuse large B-cell lymphoma (DLBCL), with gene expression measurements for $p = 7399$ genes. With such a big number of explanatory variables, we want to select a sparse parameter vector to build a regression model that predicts the survival time of the patient.

2.1 Lasso estimation for regressing `log.surv` against `expr`

```
lasso.2 = glmnet(X, Y, standardize=FALSE, intercept=FALSE, seed=123)
cv.lasso.2 = cv.glmnet(X, Y, standardize=FALSE, intercept=FALSE, seed = 123)
```





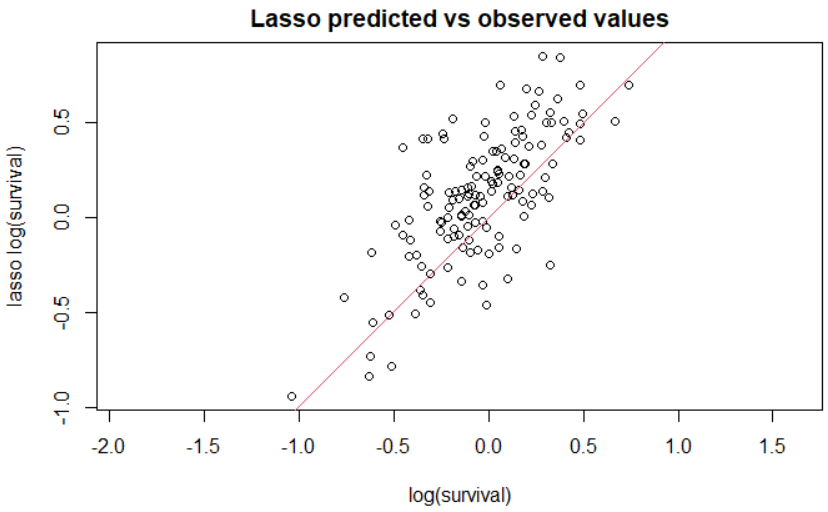
The solution $\hat{\beta}_{min}$ presenting the lowest MSE is considerably sparse, as only three parameters are different from zero. We see how $MSE(\lambda)$ achieves the lowest value in the range of 3 to 27 degrees of freedom, and thus a considerably sparse solution can be achieved without losing accuracy in the prediction.

```
beta.min = lasso.2$beta[,cv.lasso.2$index[1]]
df = data.frame(parameters=beta.min[which(abs(beta.min)>0)])
df
```

```
##         parameters
## V2252 -0.05948317
## V3787  0.14514004
## V5352  0.02729031
```

2.2 Plot observed vs predicted values

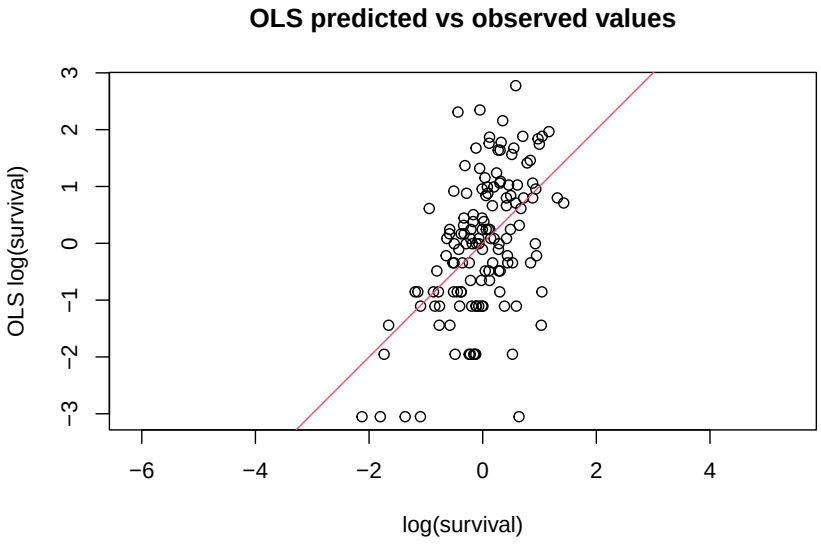
We can plot now the predicted values for $\hat{\beta}_{min}$ against the real values of the logarithm of the survival time.



2.3 OLS regression

As explained before, there is a flexibility trade-off between lasso and OLS estimations. In this exercise we will fit a OLS regression model using only the set of non-zero parameters selected by the previous lasso routine.

```
s0 = which(abs(beta.min)>0)
X.ols = X[, s0]
ols = glmnet(X.ols,Y, standardize = FALSE, intercept = FALSE, lambda=0)
Y.ols.pred = predict(ols, newx=X.ols, s=0)

plot(Y.ols.pred, Y, asp=1, main = 'OLS predicted vs observed values',
     xlab='log(survival)', ylab='OLS log(survival)')
abline(a=0,b=1,col=2)
```



As we can see, the predictions in this case are slightly better than in the lasso estimation, as now the minimization problem only considers the estimation error and not the sparsity of the parameter vector, and then the optimal is achieved when the error is minimum.

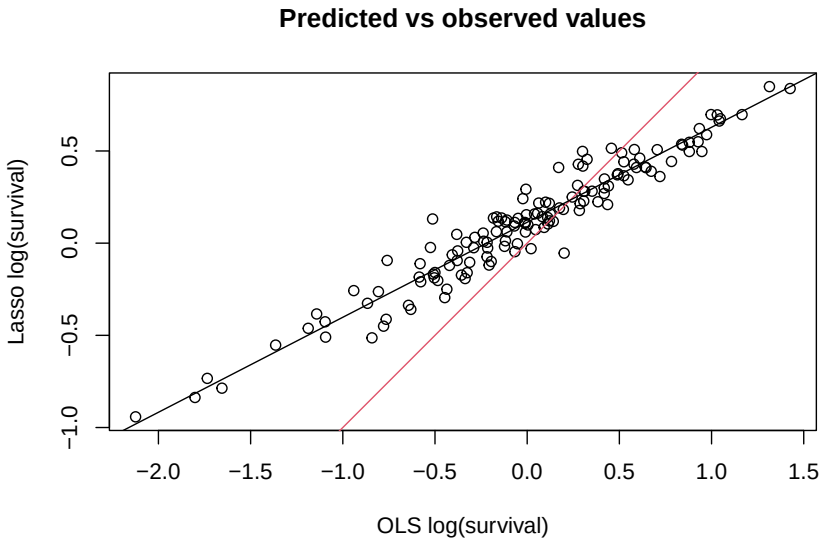
2.4 Comparison OLS and Lasso estimated coefficients

The fact that the parameter vector is no longer required to be sparse makes the parameter coefficients achieve higher values. In this case, almost an order of magnitude separates them. The relative weights between them are also no longer the same.

```
olsvslasso
```

```
##           OLS       Lasso
## V1 -0.3616023 -0.05948317
## V2  0.3325150  0.14514004
## V3  0.2849229  0.02729031
```

```
plot(Y.ols.pred, mean(log.surv) + Y.val.hat, asp=1, main='Predicted vs observed values',
     xlab='OLS log(survival)', ylab='Lasso log(survival)')
abline(a=0,b=1,col=2)
lm = lm(mean(log.surv) + Y.val.hat ~ Y.ols.pred)
abline(a=lm$coefficients[1], b = lm$coefficients[2])
```



We see how there is indeed a linear relation between the two predictions, which was expected given the linear nature of the predictors$\hat{y} = x^T \hat{\beta}$.