Assignment 1: Linear model selection

Thomas Vanpoucke, Boris Shilov

Contents

1 Variable comparison and evaluation

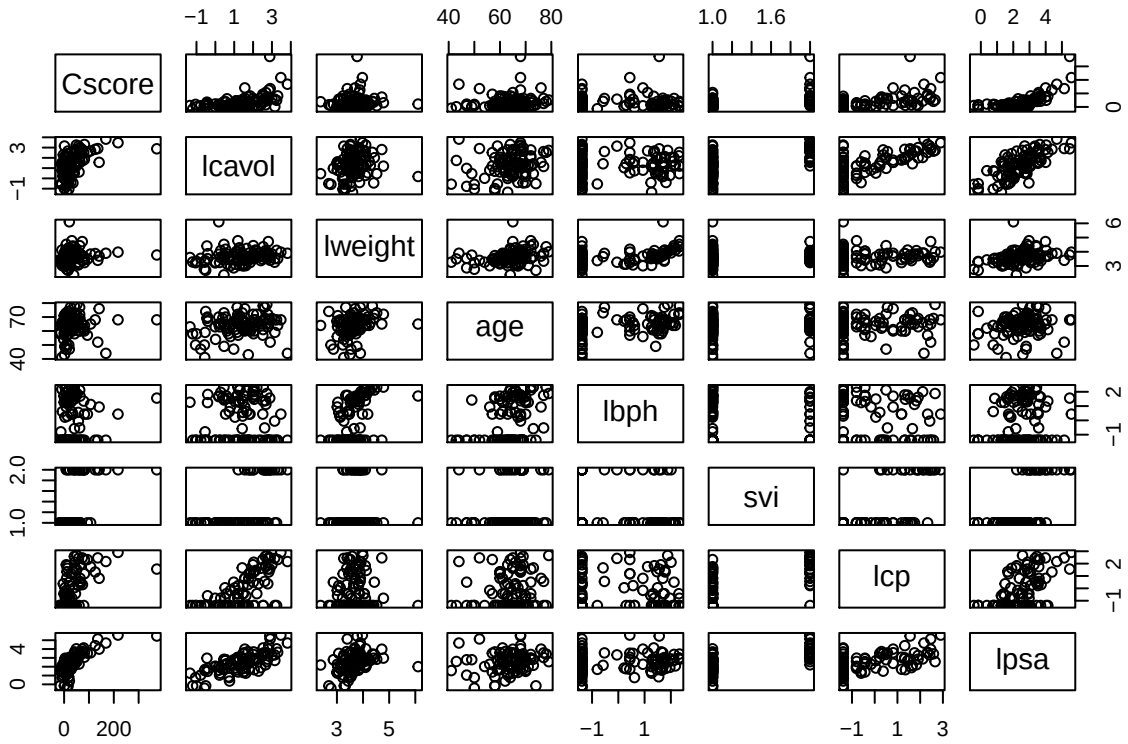
There are 97 data points, seven predictor variables and one response variable, Cscore. There is one binary variable, svi. Looking at the scatterplots below, there seem to be a few strong positive correlations present, such as those between Cscore and lpsa as well as lcavol and lpsa.

When looking at the first data points, we noticed that there were some recurring numbers, specifically within the lbph and lcp predictor variables.

```
summary(prostate)
```

```
##      Cscore        lcavol         lweight          age
##  Min.   :-19   Min.   :-1.35   Min.   :2.37   Min.   :41.0
##  1st Qu.:  9   1st Qu.: 0.51   1st Qu.:3.38   1st Qu.:60.0
##  Median : 21   Median : 1.45   Median :3.62   Median :65.0
##  Mean   : 36   Mean   : 1.35   Mean   :3.65   Mean   :63.9
##  3rd Qu.: 49   3rd Qu.: 2.13   3rd Qu.:3.88   3rd Qu.:68.0
##  Max.   :373   Max.   : 3.82   Max.   :6.11   Max.   :79.0
##       lbph       svi         lcp              lpsa
##  Min.   :-1.39   0:76   Min.   :-1.386   Min.   :-0.43
##  1st Qu.:-1.39   1:21   1st Qu.:-1.386   1st Qu.: 1.73
##  Median : 0.30          Median :-0.799   Median : 2.59
##  Mean   : 0.10          Mean   :-0.179   Mean   : 2.48
##  3rd Qu.: 1.56          3rd Qu.: 1.179   3rd Qu.: 3.06
##  Max.   : 2.33          Max.   : 2.904   Max.   : 5.58
```

```
plot(prostate)
```



2 Forward stepwise selection

We implemented algorithm 6.2 for forward stepwise selection. To choose which parameter to add in one round of forward stepwise selection, we used the model with the highest R^2 value, as per the algorithm. To select the single best model, we used BIC. The entire function code can be found in the source file for this document.

```
##
## Call:
## lm(formula = paste(data_vars[1], "~", paste(predictor_list, collapse = "+",
##     "+", paste(data_vars[i]))), data = data)
##
## Coefficients:
## (Intercept)         lpsa         svi1
##       -37.0         26.9         29.8
```

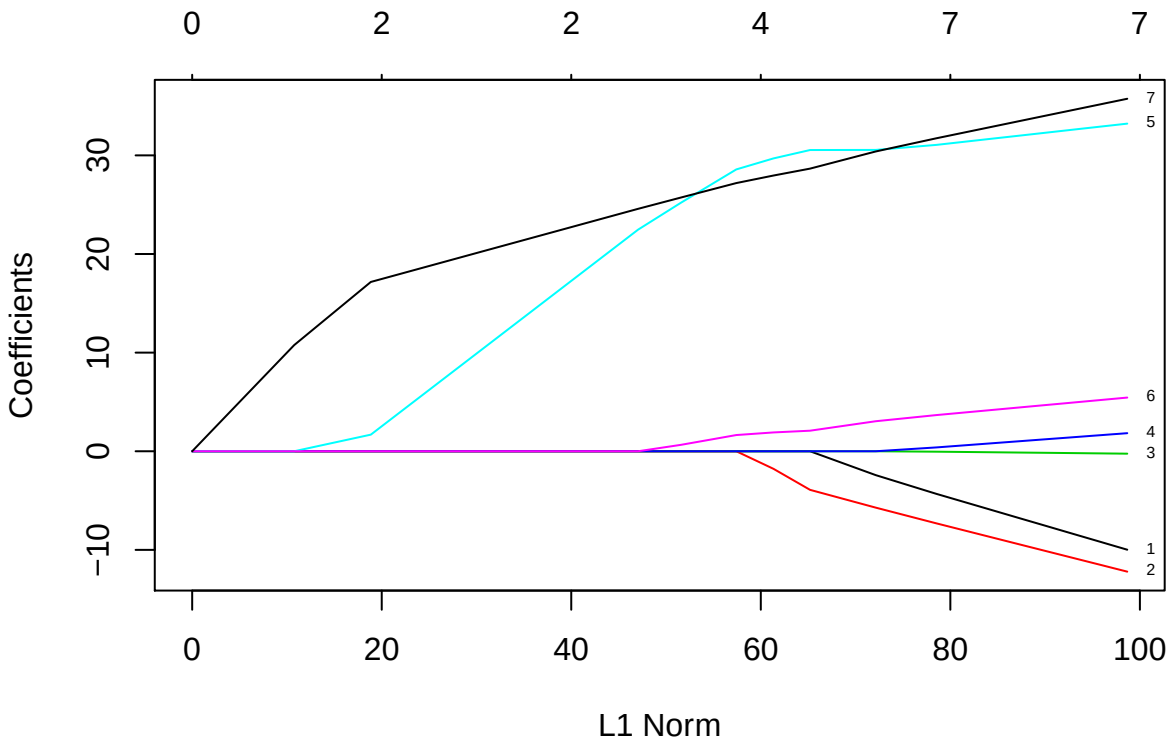
Thus, the model returned by our algorithm as the best was one with only two predictor variables, lpsa and svi.

$$Cscore = -36.977 + 26.903 \times lpsa + 29.81 \times svi1$$

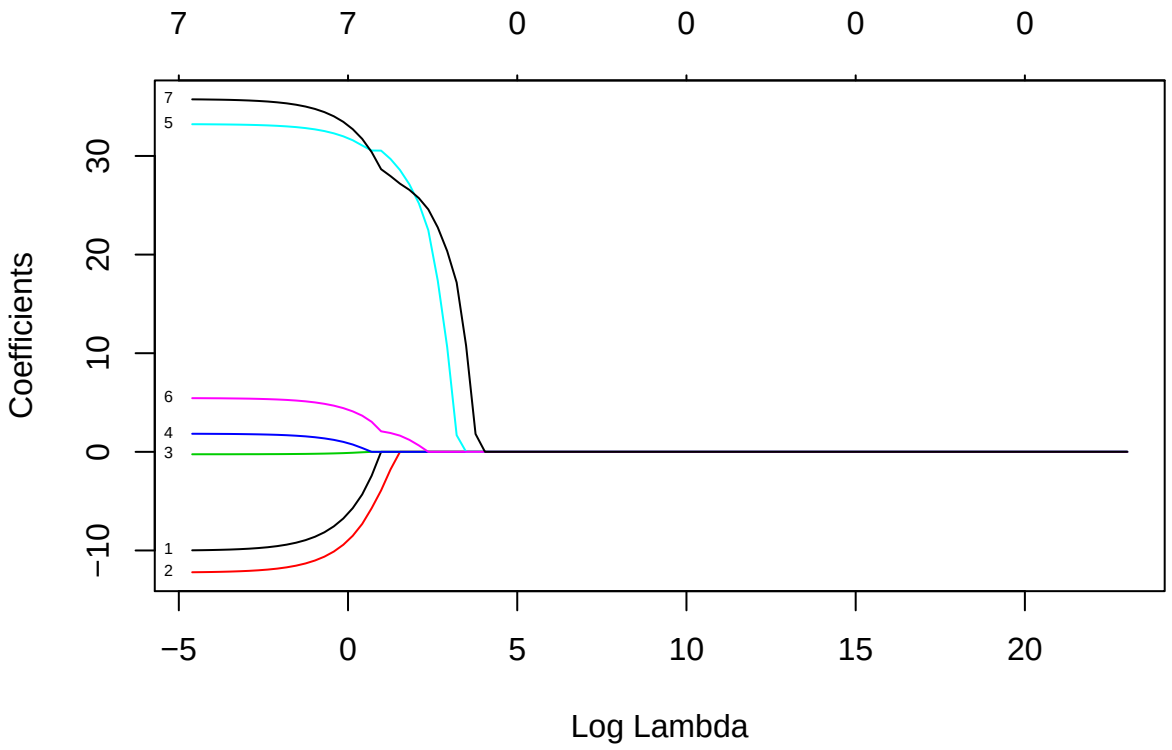
3 Lasso

To do a cross-validated lasso and ridge, we first split up the dataset into two pieces, a training and a test dataset. We made a model, based on the training data, that was crossvalidated within this training data. The test dataset was then used to check the mean squared test error for the best lambda (which is the lowest mean squared training error). To finish off, we used the entire dataset and the best lambda value to construct our lasso model.

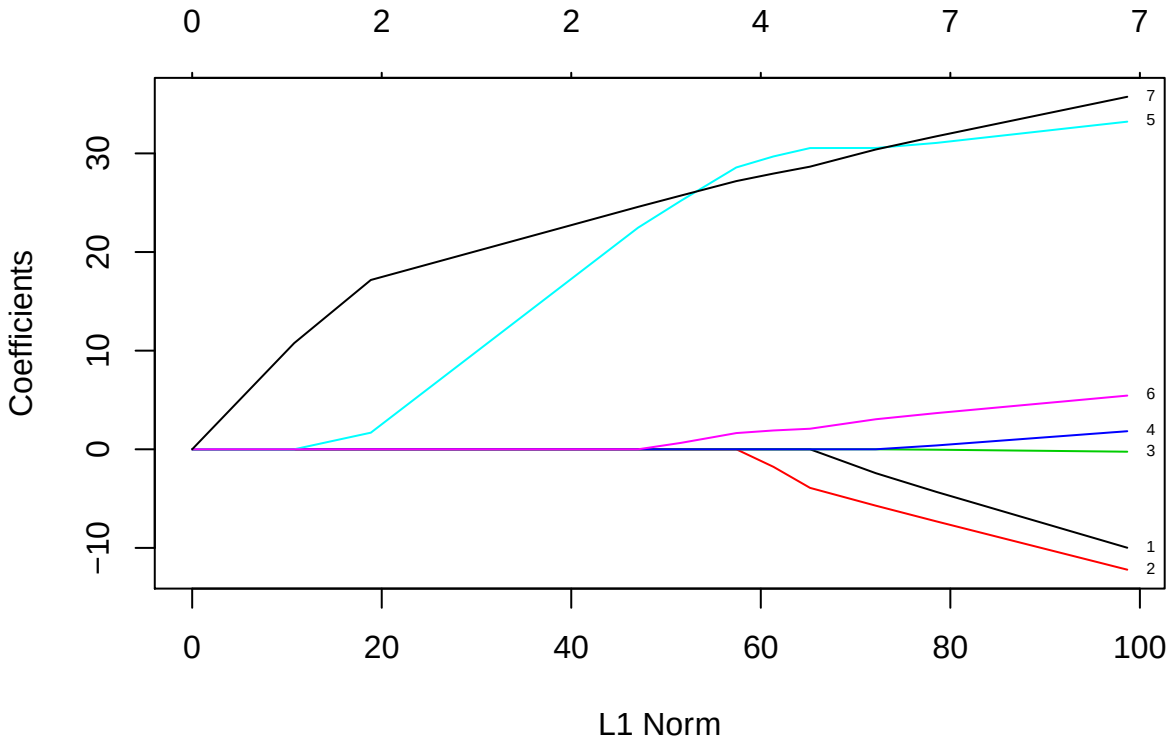
```
x = model.matrix(Cscore~.,prostate)[,-1]
y = prostate$Cscore
train=sample(1:nrow(x),nrow(x)/2)
test=(-train)
y.test=y[test]

grid=10^seq(10,-2,length=100)
lasso.model=glmnet(x[train,],y[train],alpha=1,lambda=grid, thresh = 1e-12)
plot(lasso.model, label = TRUE)
```



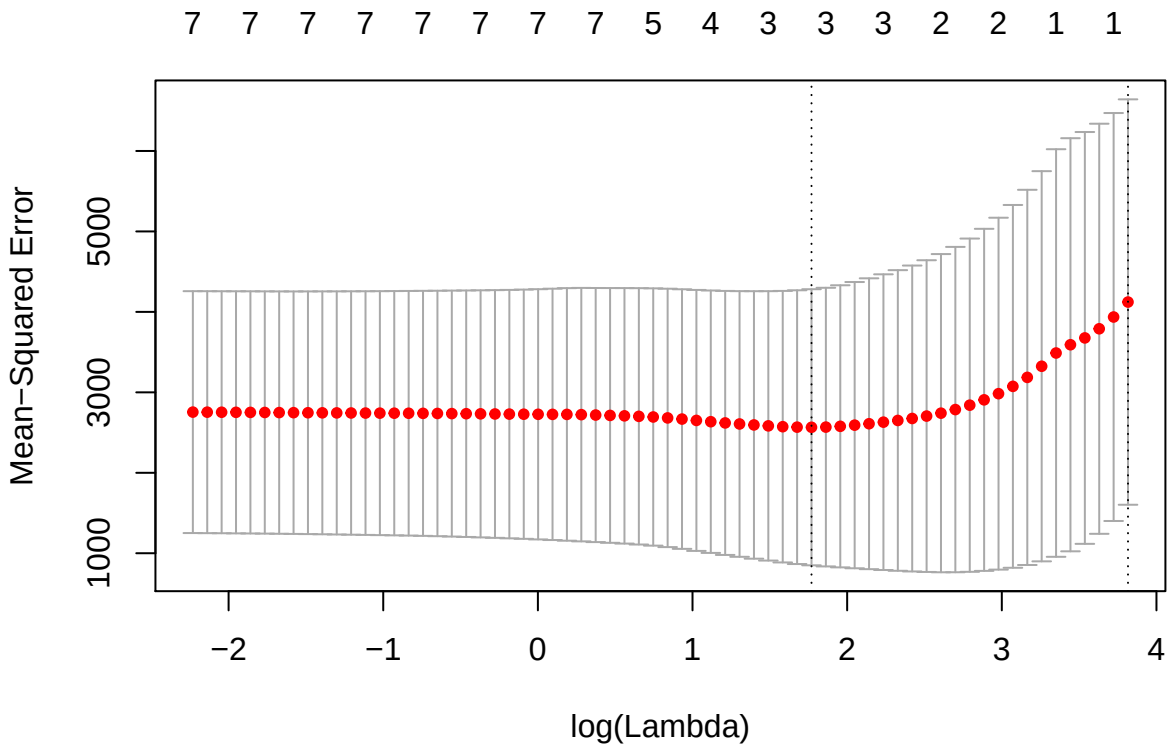

```
plot(lasso.model,xvar="lambda",label=TRUE)
```




```
lasso.model=glmnet(x[train,],y[train],alpha=1,lambda=grid, thresh = 1e-12)
OLS.model=glmnet(x[train,],y[train],alpha=1,lambda=0, thresh = 1e-12)
plot(lasso.model, label = TRUE)
```




```
cv.lasso.out=cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.lasso.out)
```




```
bestlambda.lasso=cv.lasso.out$lambda.min
out=glmnet(x,y,alpha=1,lambda=grid)

lasso.pred = predict(lasso.model,s=bestlambda.lasso,newx=x[test,])
lasso.mean = mean((lasso.pred-y.test)^2)
lasso.coef = predict(out,type="coefficients",s=bestlambda.lasso)[1:8,]

OLS.pred = predict(OLS.model, s=0, newx=x[test,])
OLS.mean = mean((OLS.pred-y.test)^2)
OLS.coef = predict(out, type="coefficients", s=0)[1:8,]

lasso.mean
```

```
## [1] 664
```

```
OLS.mean
```

```
## [1] 736
```

```
lasso.coef
```

```
## (Intercept)      lcavol     lweight         age        lbph        svi1
##      -23.09        0.00        0.00        0.00        0.00       15.47
##         lcp        lpsa
##        2.92       22.76
```

```
OLS.coef
```

```
## (Intercept)      lcavol     lweight         age        lbph        svi1
##      -4.705     -10.071     -11.895       0.201      -0.526      18.435
##         lcp        lpsa
##       7.790      33.289
```

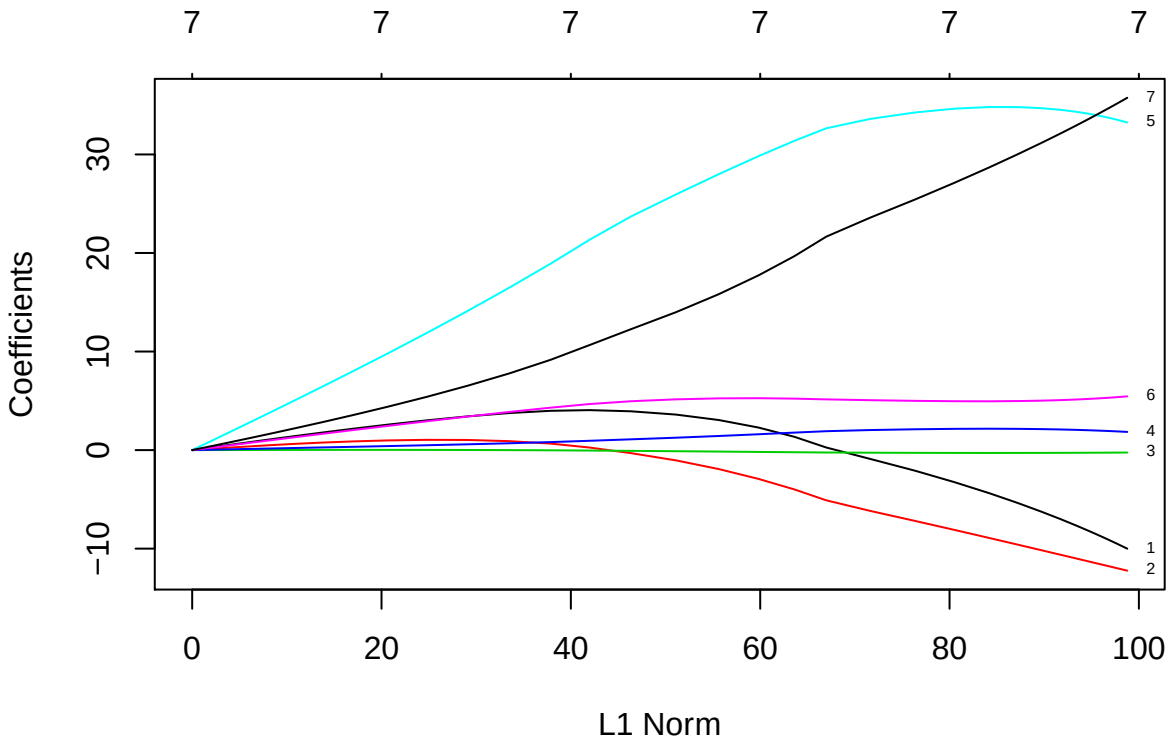
We can clearly see that Lasso zeroes out a large number of our variables, which is a massive reduction of

complexity. Two of the variables agree with our previous best forward selection. Thus Lasso is clearly superior to ordinary least squares in this instance. The resulting Lasso model becomes:

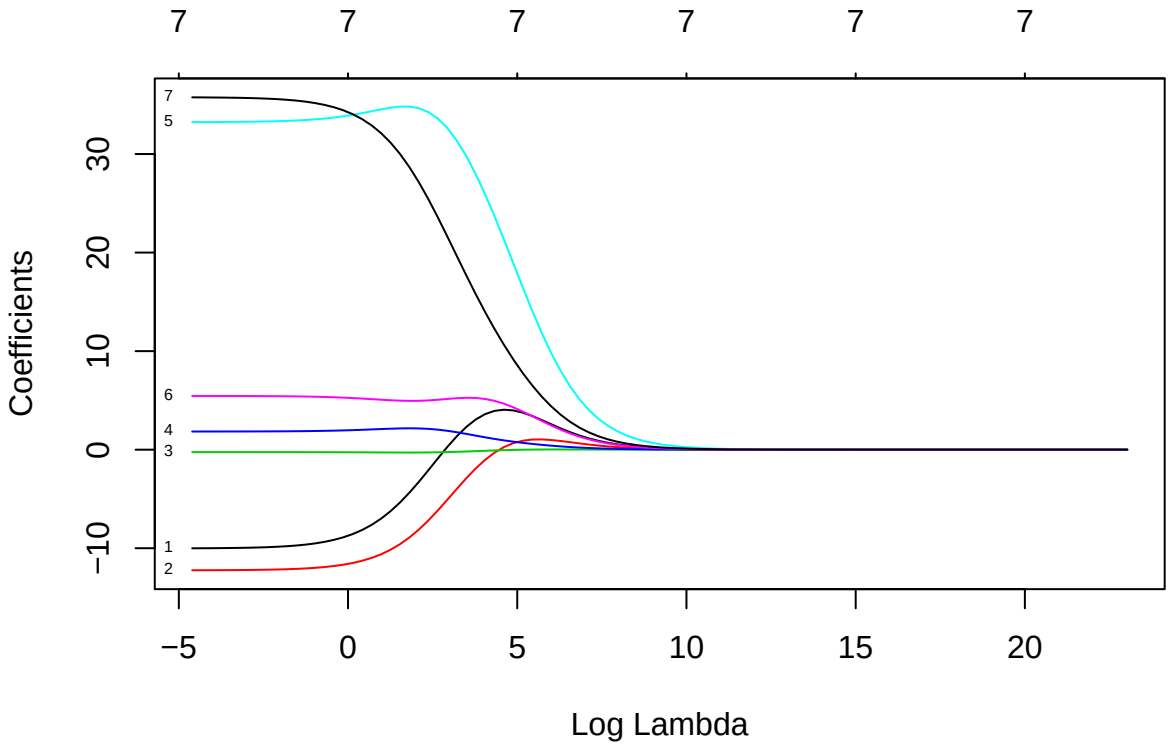
$$Cscore = -23.089 + 0 \times lweight + 15.467 \times svi1 + 2.923 \times lcp + 22.763 \times lpsa$$

4 Ridge

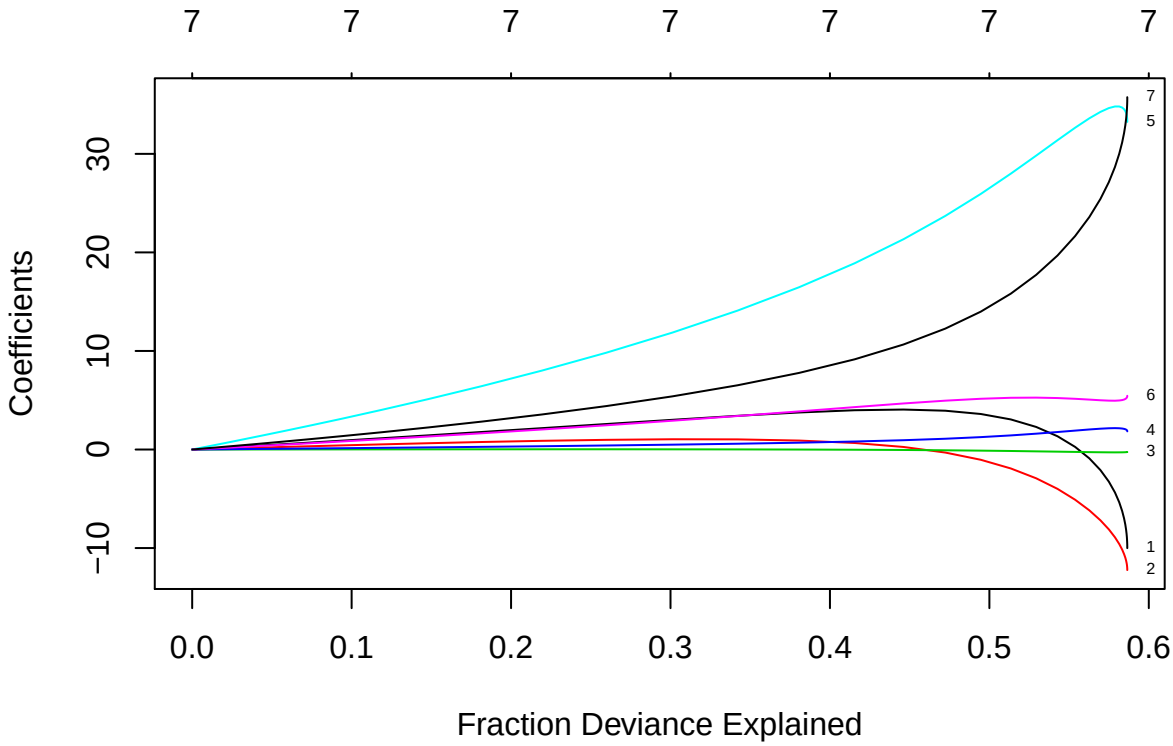
```
grid=10^seq(10,-2,length=100)
ridge.model=glmnet(x[train,],y[train],alpha=0,lambda=grid, thresh = 1e-12)
OLS.model.ridge=glmnet(x[train,],y[train],alpha=0,lambda=0, thresh = 1e-12)
plot(ridge.model, label = TRUE)
```



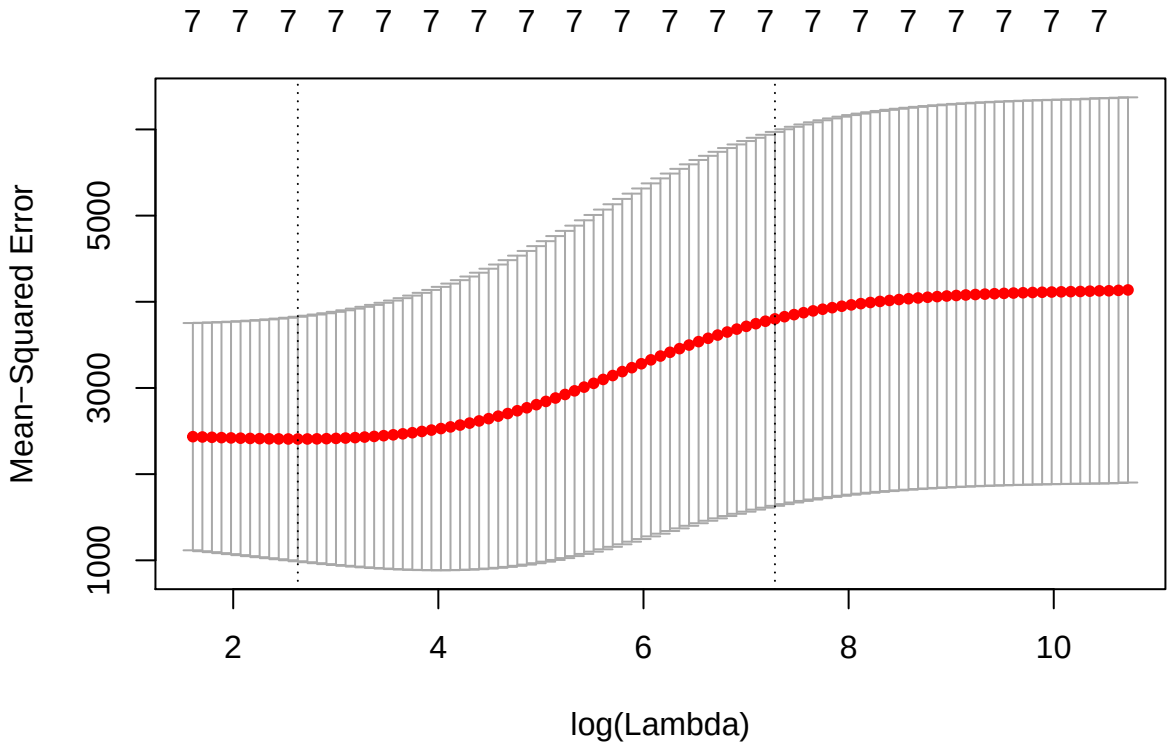

```
plot(ridge.model,xvar="lambda",label=TRUE)#behaviour coef plotted against log lambda
```




```
plot(ridge.model,xvar="dev",label=TRUE)#behaviour coef plotted against percentage deviance explained
```




```
cv.ridge.out=cv.glmnet(x[train,],y[train],alpha=0)
plot(cv.ridge.out)
```




```
bestlambda.ridge=cv.ridge.out$lambda.min
```

Lowest mean squared error here corresponds to lowest cross validation error. We can conclude that the best λ value equals 13.872.

```
bestlambda.ridge=cv.ridge.out$lambda.min
out.ridge=glmnet(x,y,alpha=1,lambda=grid)

ridge.pred = predict(ridge.model,s=bestlambda.ridge,newx=x[test,])
ridge.mean = mean((ridge.pred-y.test)^2)
ridge.coef = predict(out.ridge,type="coefficients",s=bestlambda.ridge)[1:8,]

OLS.pred.ridge = predict(OLS.model.ridge, s=0, newx=x[test,])
OLS.mean.ridge = mean((OLS.pred.ridge-y.test)^2)
OLS.coef.ridge = predict(out.ridge, type="coefficients", s=0)[1:8,]

ridge.mean
```

```
## [1] 675
```

```
OLS.mean.ridge
```

```
## [1] 736
```

```
ridge.coef
```

```
## (Intercept)      lcavol     lweight         age        lbph        svi1
##      -12.65        0.00        0.00        0.00        0.00        7.56
##         lcp        lpsa
##        0.42       19.06
```

```
OLS.coef
```

```
## (Intercept)      lcavol     lweight         age        lbph        svi1
```

```
##      -4.705     -10.071     -11.895       0.201      -0.526      18.435
##         lcp        lpsa
##       7.790      33.289
```

There is still a subsantial improvement over OLS in Ridge as most coefficients are zeroed out. The resulting Ridge model is thus:

$$Cscore = -12.651 + 7.561 \times svi1 + 0.42 \times lcp + 19.061 \times lpsa$$

5 Principal Component Regression

5.1 How many principal components would you select for yout PCR model?

We would select 3 principal components, because the mean squared error compared to the number of components is the lowest there. Of course, taking 7 principal components is still lower, beacause all portions of all variables are taken into account there. (see graph below) Also, 3 components already explain approximately 78% of the variation in the data.

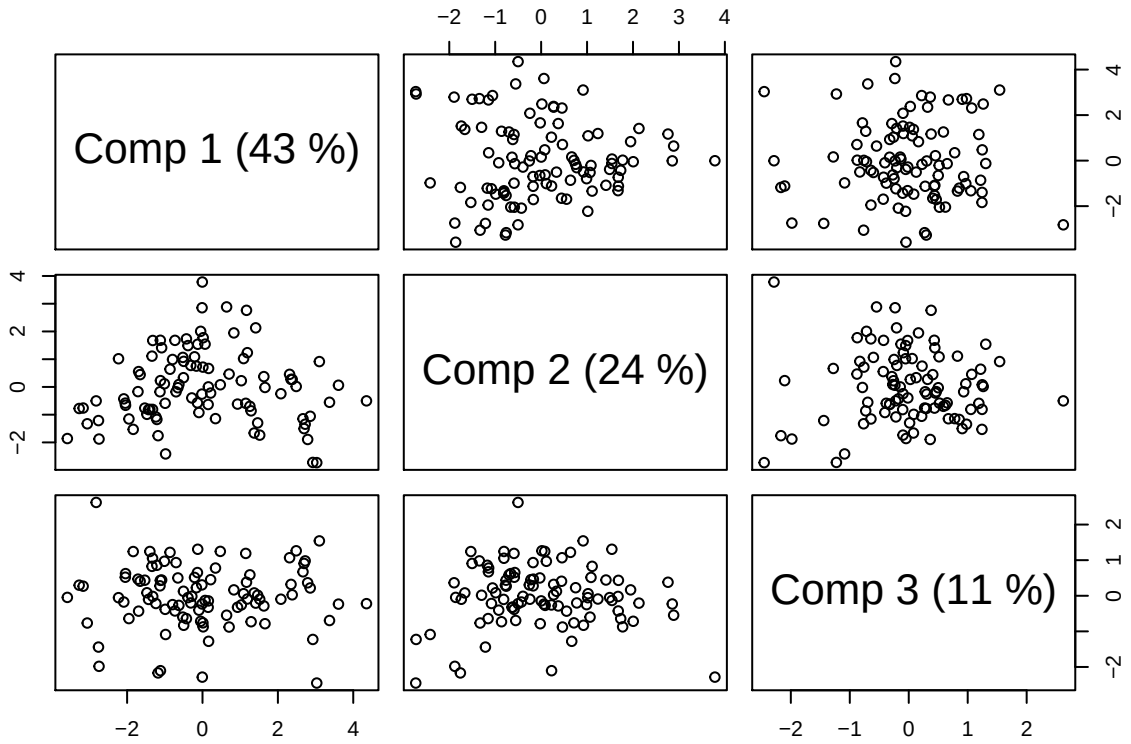
```
set.seed(10)
pcr_model=pcr(Cscore~.,data=prostate,scale=TRUE,
        validation="CV")
summary(pcr_model)
```

```
## Data:    X dimension: 97 7
##  Y dimension: 97 1
## Fit method: svdpc
## Number of components considered: 7
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           52.99    40.71    40.06    39.94    39.85    40.56    39.85
## adjCV        52.99    40.61    39.95    39.83    39.71    40.43    39.66
##        7 comps
## CV       37.47
## adjCV    37.26
##
## TRAINING: % variance explained
##         1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X         43.44    66.95    77.62    85.37    92.39    97.42   100.00
## Cscore    44.91    46.93    47.77    48.25    48.28    52.84    59.22
```

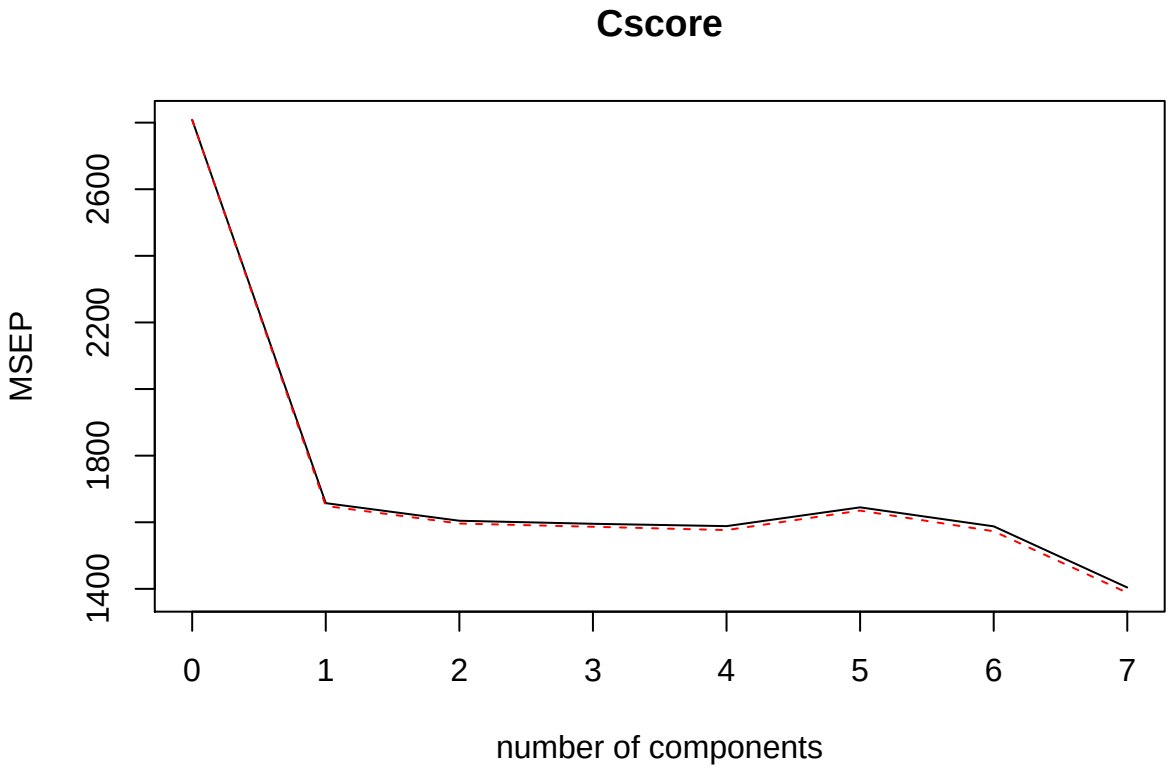
```
explvar(pcr_model)
```

```
## Comp 1 Comp 2 Comp 3 Comp 4 Comp 5 Comp 6 Comp 7
##  43.44  23.51  10.67   7.75   7.02   5.03   2.58
```

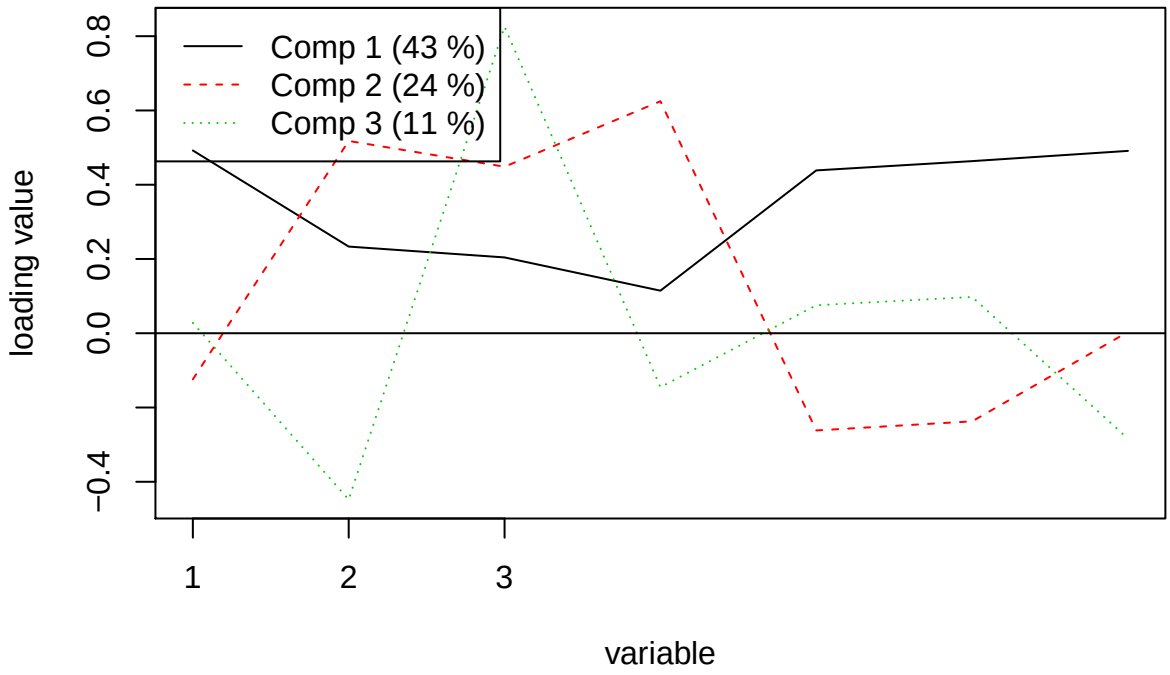
```
plot(pcr_model, plottype = "scores", comps = 1:3)
```



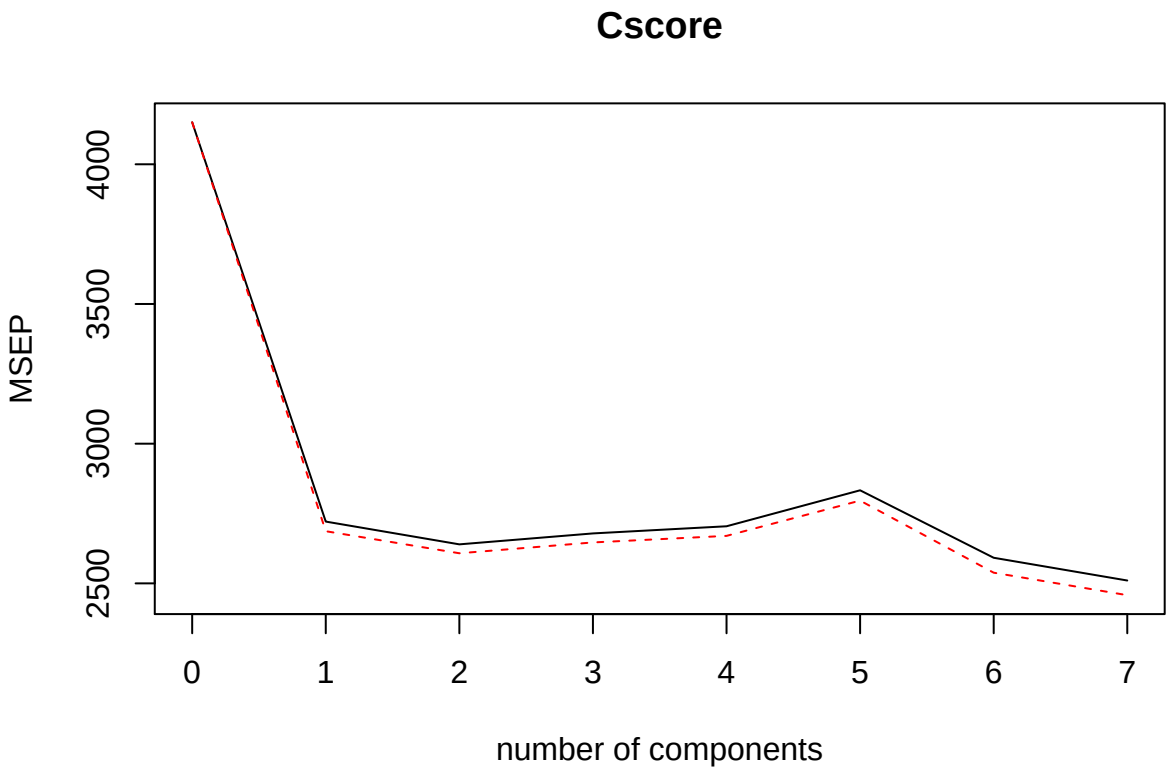

```
validationplot(pcr_model,val.type="MSEP")
```



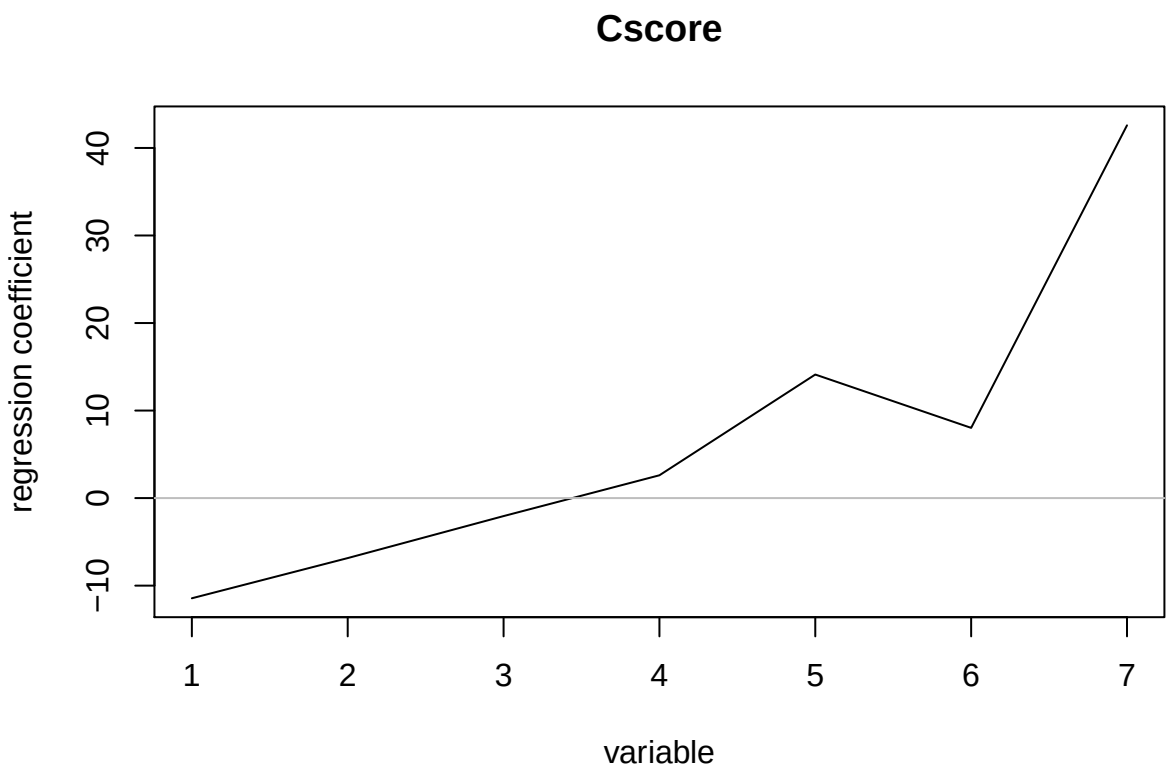

```
plot(pcr_model, "loadings", comps = 1:3, legendpos = "topleft",labels = 1:3)
abline(h = 0)
```




```
pcr_model=pcr(Cscore~.,data=prostate,subset=train,scale=TRUE,
        validation="CV")
validationplot(pcr_model,val.type="MSEP")
```




```
coefplot(pcr_model)
```




```
pcr_pred=predict(pcr_model,x[test,],ncomp=3)
mean((pcr_pred-y.test)^2)
```

```
## [1] 873
```

```
pcr_model=pcr(y~x,scale=TRUE,ncomp=3)
summary(pcr_model)
```

```
## Data:    X dimension: 97 7
##  Y dimension: 97 1
## Fit method: svdpc
## Number of components considered: 3
## TRAINING: % variance explained
##    1 comps  2 comps  3 comps
## X    43.44    66.95    77.62
## y    44.91    46.93    47.77
```

```
linreg = lm(Cscore~., data = prostate[train,])
lm_pred = predict(linreg, prostate[test,], type="response")
mean((lm_pred-y.test)^2)
```

```
## [1] 736
```

5.2 How appropriate is PCR for this dataset?

When we look at the analysis we did, we can certainly say that it is beneficial to use PCR for this dataset, since we can explain a lot of the variance with only 3 principal components. However, when we use cross-validation for our 3-PC model, we can see that the mean squared error of the PCR model is higher than the mean squared error of a normal linear regression model (see below). So, we do not think that PCR is really beneficial here.