MRR_TP1_Zeyu_CHEN_Clement_VEYSSIERE

Zeyu CHEN et Clement_VEYSSIERE

2018/10/22

IV. The Boston Housing data set

Linear models

```
rm(list = ls())
library(mlbench)
data(BostonHousing)

sub <- sample(nrow(BostonHousing), 0.6 * nrow(BostonHousing))
TabTrain <- BostonHousing[sub,]
TabTest <- BostonHousing[-sub,]
# model with TabTrain
modBH <- lm(medv~.,data=TabTrain)
summary(modBH)
```

```
##
## Call:
## lm(formula = medv ~ ., data = TabTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.5920 -2.8358 -0.7694  1.6698 24.1616
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.835189   7.016245   5.393 1.45e-07 ***
## crim         -0.147017   0.046600  -3.155 0.001775 **
## zn            0.052480   0.018071   2.904 0.003968 **
## indus         0.070128   0.091712   0.765 0.445103
## chas1         3.506756   1.249643   2.806 0.005353 **
## nox         -22.266630   5.671013  -3.926 0.000108 ***
## rm            3.955928   0.570023   6.940 2.58e-11 ***
## age           0.008393   0.018127   0.463 0.643695
## dis          -1.778762   0.278569  -6.385 6.80e-10 ***
## rad           0.463375   0.094327   4.912 1.51e-06 ***
## tax          -0.018580   0.005725  -3.245 0.001312 **
## ptratio      -0.882055   0.175151  -5.036 8.38e-07 ***
## b             0.011372   0.003700   3.074 0.002316 **
## lstat        -0.559829   0.072405  -7.732 1.77e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.975 on 289 degrees of freedom
## Multiple R-squared:  0.7375, Adjusted R-squared:  0.7257
## F-statistic: 62.47 on 13 and 289 DF,  p-value: < 2.2e-16
```

```r
RSSlm <- sum((TabTest$medv - predict(modBH,newdata = TabTest))^2)/length(TabTest$medv)

#As we can see in the print ,there are some coefs whose p-value is larger than 0.1 #which means it is n
#model based on a small subset of vars

modBHboth <- step(modBH,direction = 'both')
```

```
## Start:  AIC=985.99
## medv ~ crim + zn + indus + chas + nox + rm + age + dis + rad +
##     tax + ptratio + b + lstat
##
##           Df Sum of Sq    RSS     AIC
## - age      1      5.31 7159.3  984.22
## - indus    1     14.47 7168.5  984.61
## <none>                 7154.0  985.99
## - chas     1    194.94 7349.0  992.14
## - zn       1    208.77 7362.8  992.71
## - b        1    233.87 7387.9  993.74
## - crim     1    246.39 7400.4  994.25
## - tax      1    260.70 7414.7  994.84
## - nox      1    381.63 7535.7  999.74
## - rad      1    597.37 7751.4 1008.29
## - ptratio  1    627.80 7781.8 1009.48
## - dis      1   1009.31 8163.3 1023.98
## - rm       1   1192.25 8346.3 1030.70
## - lstat    1   1479.87 8633.9 1040.97
##
## Step:  AIC=984.22
## medv ~ crim + zn + indus + chas + nox + rm + dis + rad + tax +
##     ptratio + b + lstat
##
##           Df Sum of Sq    RSS     AIC
## - indus    1     13.70 7173.0  982.80
## <none>                 7159.3  984.22
## + age      1      5.31 7154.0  985.99
## - chas     1    200.96 7360.3  990.61
## - zn       1    203.46 7362.8  990.71
## - b        1    240.70 7400.0  992.24
## - crim     1    247.75 7407.1  992.53
## - tax      1    257.13 7416.5  992.91
## - nox      1    386.16 7545.5  998.14
## - rad      1    592.07 7751.4 1006.30
## - ptratio  1    624.52 7783.9 1007.56
## - dis      1   1120.05 8279.4 1026.26
## - rm       1   1290.55 8449.9 1032.44
## - lstat    1   1706.15 8865.5 1046.99
##
## Step:  AIC=982.8
## medv ~ crim + zn + chas + nox + rm + dis + rad + tax + ptratio +
##     b + lstat
##
##           Df Sum of Sq    RSS     AIC
## <none>                 7173.0  982.80
## + indus    1     13.70 7159.3  984.22
```

```
## + age      1      4.53 7168.5  984.61
## - zn       1    191.35 7364.4  988.78
## - chas     1    216.64 7389.7  989.81
## - b        1    240.52 7413.6  990.79
## - crim     1    251.99 7425.0  991.26
## - tax      1    277.63 7450.7  992.31
## - nox      1    373.02 7546.1  996.16
## - rad      1    604.08 7777.1 1005.30
## - ptratio  1    612.33 7785.4 1005.62
## - dis      1   1183.64 8356.7 1027.08
## - rm       1   1278.64 8451.7 1030.50
## - lstat    1   1695.15 8868.2 1045.08
```

```
summary(modBHboth)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + chas + nox + rm + dis + rad +
##     tax + ptratio + b + lstat, data = TabTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6173 -2.9560 -0.9745  1.6463 24.5946
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.039688   6.937045   5.339 1.88e-07 ***
## crim         -0.148571   0.046467  -3.197 0.001540 **
## zn            0.048842   0.017530   2.786 0.005683 **
## chas1         3.660317   1.234670   2.965 0.003281 **
## nox         -20.636201   5.304815  -3.890 0.000124 ***
## rm            3.986963   0.553571   7.202 5.09e-12 ***
## dis          -1.841580   0.265758  -6.930 2.72e-11 ***
## rad           0.434850   0.087841   4.950 1.26e-06 ***
## tax          -0.016130   0.004806  -3.356 0.000896 ***
## ptratio      -0.866304   0.173813  -4.984 1.07e-06 ***
## b             0.011498   0.003681   3.124 0.001966 **
## lstat        -0.539468   0.065053  -8.293 4.15e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.965 on 291 degrees of freedom
## Multiple R-squared:  0.7368, Adjusted R-squared:  0.7269
## F-statistic: 74.07 on 11 and 291 DF,  p-value: < 2.2e-16
```

```
RSSstep <- sum((TabTest$medv - predict(modBHboth,newdata = TabTest))^2)/length(TabTest$medv)
data.frame(RSSlm,RSSstep)
```

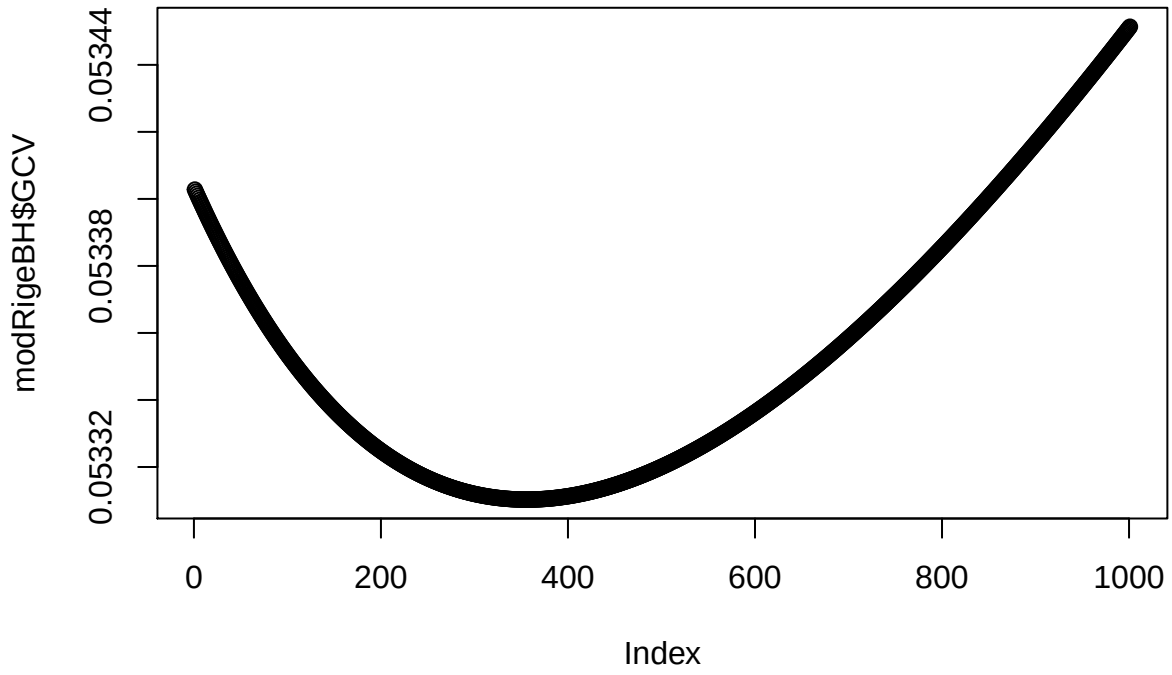
```
##      RSSlm  RSSstep
## 1 21.37756 21.09605
```

```
#We can find that the model which produced by stepwise regression has a smaller RSE
```

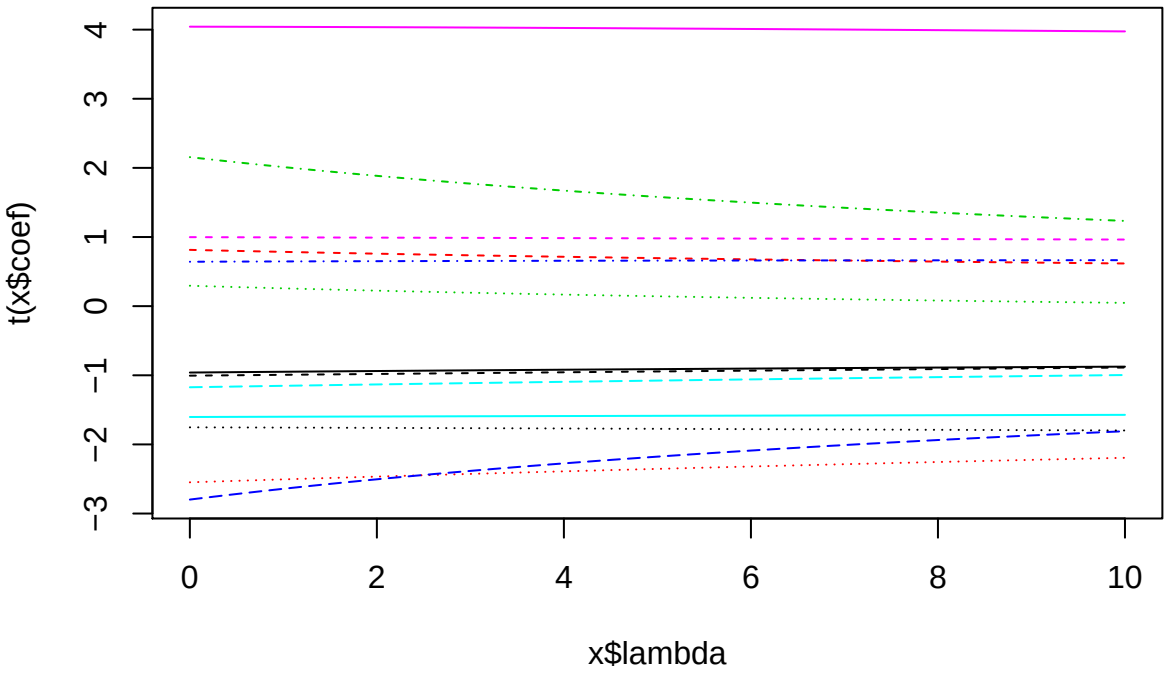
Ridge models

```
library(mlbench)
library(MASS)
data(BostonHousing)
BostonHousing$chas = as.numeric(BostonHousing$chas)
sub <- sample(nrow(BostonHousing), 0.6 * nrow(BostonHousing))
TabTrain <- BostonHousing[sub,]
TabTest <- BostonHousing[-sub,]
```

```
#Compute ridge regression model for different values of lambda starting from 0
#to 10 with an increment of 0.01 which is for finding the best lambda
modRigeBH <- lm.ridge(medv~.,TabTrain,lambda = seq(0,10,0.01))

plot(modRigeBH$GCV)
```




```
plot(modRigeBH)
```




```
#The index which correspond the smallest GCV is the best index of lambda
indexlambda <- which.min(modRigeBH$GCV)

coefridge <- coef(lm.ridge(medv~.,TabTrain,lambda = modRigeBH$GCV[indexlambda]))

#finnally , the coefs that we found by using Ridge is here
data.frame(coefridge)
```

```
##             coefridge
##           9.64751875
## crim     -0.10140009
## zn        0.03375273
## indus     0.04345168
## chas      2.52979112
## nox      -9.78300286
## rm        6.13868939
## age      -0.03603563
## dis      -1.17693448
## rad       0.24676862
## tax      -0.01693959
## ptratio -0.71700871
## b         0.01153654
## lstat    -0.25811384
```

Lasso models

```
library(mlbench)
library(lars)
```

```
## Loaded lars 1.2
```

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

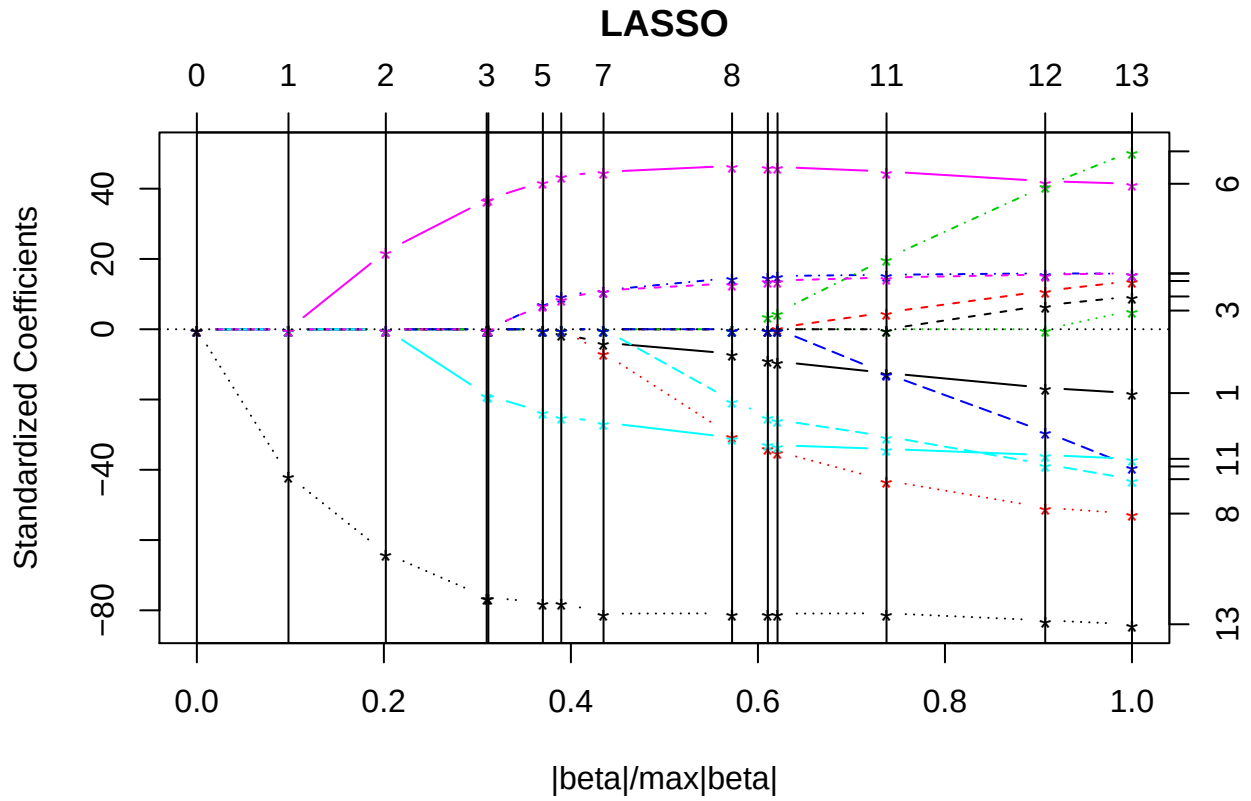
```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

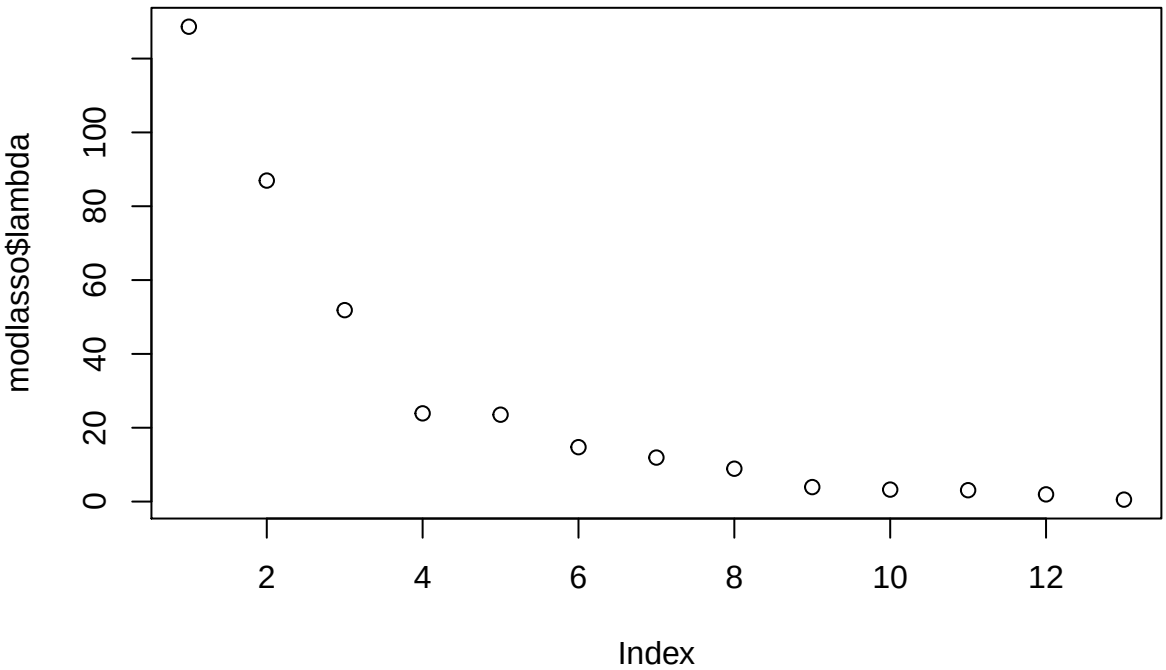
```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
data(BostonHousing)
BostonHousing$chas = as.numeric(BostonHousing$chas)

#split data
sub <- sample(nrow(BostonHousing), 0.75 * nrow(BostonHousing))

TabTrain <- BostonHousing[sub,]
TabTest <- BostonHousing[-sub,]
Ytrain = TabTrain$medv
Ytest = TabTest$medv

Xtrain = as.matrix(scale(select(TabTrain,-medv)))
Xtest = as.matrix(scale(select(TabTest,-medv)))

#Xtrain = as.matrix(select(TabTrain,-medv))
#Xtest =as.matrix(select(TabTest,-medv))

#Lasso regression
modlasso = lars(Xtrain, Ytrain, type = "lasso")
plot(modlasso)
```




```
plot(modlasso$lambda)
```




```
pY = predict.lars(modlasso, Xtest, type = "fit", mode = "lambda", s=modlasso$lambda[1])
MSElasso = mean((Ytest - pY$fit)^2)
lambda = 1
for (i in 2:length(modlasso$lambda)){
  pY = predict.lars(modlasso, Xtest, type = "fit", mode = "lambda", s=modlasso$lambda[i])
```

```
  newMSElasso = mean((Ytest - pY$fit)^2)
  if (MSElasso > newMSElasso){
    MSElasso = newMSElasso
    lambda = i
  }
}

lambda #index of lambda which gives the smallest mean square error
```

```
## [1] 13
```

```
MSElasso
```

```
## [1] 25.51854
```

```
modlasso$lambda[lambda]
```

```
## [1] 0.5282995
```

```
coef = predict.lars(modlasso, Xtest, type = "coefficients", mode = "lambda", s = modlasso$lambda[lambda]
coef$coefficients #coefficients obtained for this lambda
```

```
##       crim         zn      indus       chas        nox         rm
## -0.8585205  0.5616489  0.0000000  0.8180226 -1.9755440  2.1677278
##        age        dis        rad        tax    ptratio          b
##  0.3507786 -2.6131163  2.0963148 -1.5028339 -1.8415838  0.8004054
##      lstat
## -4.2618125
```

```
pY = predict.lars(modlasso, Xtest, type = "fit", mode = "lambda", s=modlasso$lambda[lambda])

RSS <- data.frame(sum((Ytest = TabTest$medv - pY$fit)^2)/14)
names(RSS) <- "RSS by Lasso"
row.names(RSS) <- "RSS"
RSS
```

```
##     RSS by Lasso
## RSS     231.4897
```