Short solutions to Compulsory exercise 2

TMA4268 Statistical Learning V2018

Thea Roksvåg, thea.roksvag@ntnu.no and Mette Langaas

23 May, 2018

Last changes: (23.05: corrected Q13 on acceleration/cylinder from Bjørn, 20.05 corrected Q2 decrease-increase from Rasmus.)

Problem 1: Model selection and cross-validation

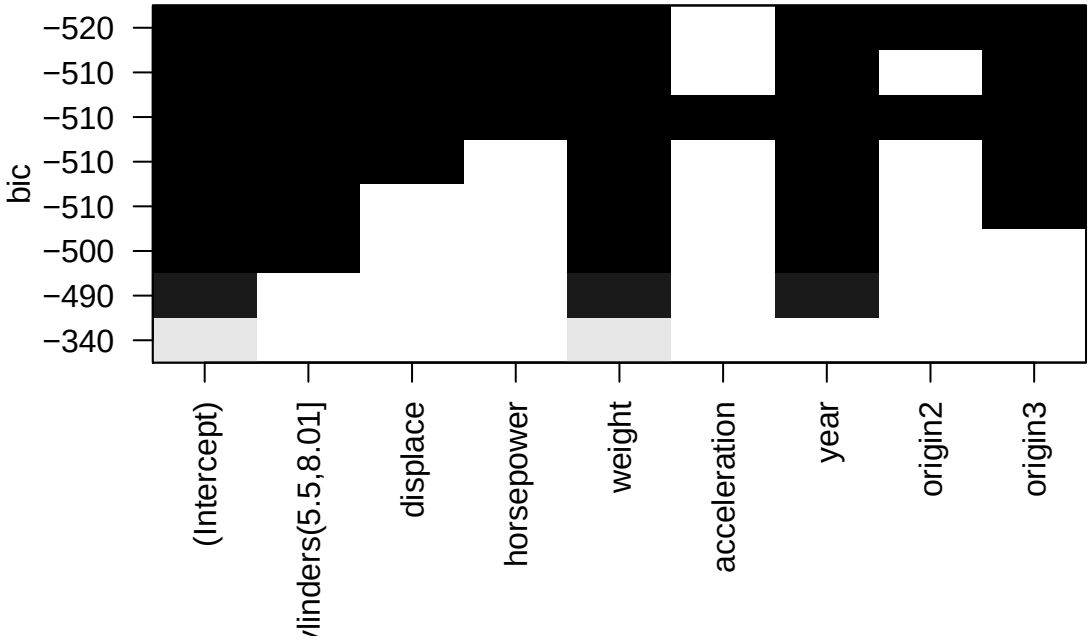
1a) Theory

Q1: How many possible models? 2^d possible models: each covariate can either be in or out (2 possibilities) and we look at d covariates.

Q2: How to choose best model

Use MSE to find the best model for each model complexity. Then choose the best model complexity from minimum BIC.

If R^2 is used the most complex model will always be selected. Remember that RSS on the training data cannot increase when new covariates are added to the model, even if the covariates are random noise. This means that $R^2 = 1 - \text{RSS/TSS}$ cannot decrease.



1b) Interpreting output

Q3: How to compare models

To compare models with the same model complexity (same number of covariate) we may use the RSS or the R^2. The best model with two covariates is `mpg~weight+year`.

```
fit=lm(mpg~.-acceleration,data=ourAutoTrain)
summary(fit)
library(nortest)
ad.test(rstudent(fit))
trainMSE=mean((fit$fitted.values-ourAutoTrain$mpg)^2)
trainMSE
pred=predict(fit,newdata=ourAutoTest)
Q5testMSE=mean((pred-ourAutoTest$mpg)^2)
Q5testMSE
```

```
##
## Call:
## lm(formula = mpg ~ . - acceleration, data = ourAutoTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7254 -2.0561 -0.3412  1.7122 12.9550
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -1.941e+01  4.589e+00  -4.230 3.09e-05 ***
## cylinders(5.5,8.01] -3.533e+00  7.469e-01  -4.730 3.43e-06 ***
## displace             3.279e-02  6.749e-03   4.858 1.90e-06 ***
## horsepower          -4.883e-02  1.184e-02  -4.124 4.80e-05 ***
## weight              -5.824e-03  6.185e-04  -9.415  < 2e-16 ***
## year                 7.849e-01  5.699e-02  13.771  < 2e-16 ***
## origin2              1.794e+00  6.204e-01   2.892  0.00411 **
## origin3              2.906e+00  5.943e-01   4.891 1.63e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.233 on 305 degrees of freedom
## Multiple R-squared:  0.8344, Adjusted R-squared:  0.8306
## F-statistic: 219.6 on 7 and 305 DF,  p-value: < 2.2e-16
##
##
##  Anderson-Darling normality test
##
## data:  rstudent(fit)
## A = 2.2684, p-value = 9.241e-06
##
## [1] 10.18351
## [1] 8.931018
```

Q4: Choose model, BIC, best model:

To compare models with different model complexity we could use penalized versions of the RSS, for example AIC, BIC and Mallow's Cp, or R^2adjusted - or use cross-validation to report MSE on test data. The smallest BIC is for the model with 7 coded covariates, `mpg~cylinders(5.5,8.01]+displace+horsepower+weight+year+origin2+origin3`. This models explains 83% of the variability in the data, and all covariates are significant at level 0.004. However, normality of the residuals is questionable. MSE on training set is 10.1835056.

Q5: Use this model fit to predict new values for `ourAutoTest` and report the MSE.

MSE on test set is 8.9310178.

1c) Cross-validation

Q6. Explain how k-fold cross-validation is performed.

Divide training part of data into k random divisions. Repeat for $j = 1, ..., k$: fit model(s) on all parts except j and then use part j for prediction - and then calculate a loss function. Sum over all parts and get a final MSE on the data left out. Use this to choose between different models - or to evaluate MSE.

For $k = 5$ this is in detail: we divide the training data randomly into 5 folds of size $n/5$ each and call the folds $j = 1$, to $j = 5$. For $j = 1, \ldots, 5$:

- use the $4n/5$ observations from the folds except fold j to fit the model
- the observations in the jth fold is left out and is the validation set, there are $n/5$ observations - and we denote them (x_{0j}, y_{0j}),
- we then predict $\hat{f}(x_{0j})$.
- We calculate the error in the jth fold of the validation set as $\sum_j (y_{0j} - \hat{f}(x_{0j}))^2$ where the j is for the validation fold

The total error on the validation set is thus the validation MSE=$\frac{1}{n}\sum_{m=1}^{5}\sum_j (y_{0j} - \hat{f}(x_{0j}))^2$.

Q7. Why may k-fold cross-validation be preferred to leave-one-out cross-validation?

The LOOCV would give a less biased estimate (than k-fold) of the MSE on a test set, since the sample size used $(n - 1)$ is close to the sample size to be used in the real world situation (n), but the LOOCV is time consuming (however, for linear regression simple formula exists) and will be variable (the variance of the validation MSE is high). On the other hand the k-fold CV would give a biased estimate of the MSE on a test set since the sample size used is (k-1)/k of the sample size that would be used in the real world situation. However, the variance will be low and the procedure will be less time consuming.

1d) Programming cross-validation

Q8. R-code for 10-fold CV.

```
library(caret)
library(leaps)

nmodels=8
nfolds=10
set.seed(4268)
folds <- createFolds(ourAutoTrain$mpg,k=nfolds)
folds[[1]] #just checking which observations is in fold 1
folds[[2]]
lapply(folds,length)
sum(unlist(lapply(folds,length)))

# smart to make predict.regsubset
predict.regsubsets=function(object,newdata,id,...){
form=as.formula(object$call[[2]])
mat=model.matrix(form,newdata)
coefi=coef(object,id=id)
xvars=names(coefi)
mat[,xvars]%*%coefi
```

```
}

cv.errors=rep(0,nmodels)
# cross validation loop
for(j in 1:nfolds)
{
  thisfit=regsubsets(mpg~.,data=ourAutoTrain[-folds[[j]],],nvmax=nmodels)
  for(i in 1:nmodels)
  {
  pred=predict(thisfit,ourAutoTrain[folds[[j]],],id=i)
  cv.errors[i]=cv.errors[i]+sum((ourAutoTrain$mpg[folds[[j]]]-pred)^2)
  }
}
which.min(cv.errors)
# all covariates in - is the best

cvbest=lm(mpg~.,data=ourAutoTrain)
summary(cvbest)
```

```
##  [1]  13  19  22  43  47  49  70  73  82  88  98 109 112 118 134 139 150
## [18] 153 159 170 183 194 210 224 256 261 265 271 276 290 300
##  [1]  20  32  46  54  57  64  85  89 106 110 128 142 149 155 156 191 198
## [18] 208 214 217 218 221 222 235 238 248 280 288 298 301 313
## $Fold01
## [1] 31
##
## $Fold02
## [1] 31
##
## $Fold03
## [1] 30
##
## $Fold04
## [1] 31
##
## $Fold05
## [1] 31
##
## $Fold06
## [1] 32
##
## $Fold07
## [1] 31
##
## $Fold08
## [1] 32
##
## $Fold09
## [1] 32
##
## $Fold10
## [1] 32
##
## [1] 313
```

```
## [1] 7
##
## Call:
## lm(formula = mpg ~ ., data = ourAutoTrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6520 -1.9959 -0.1845  1.7847 12.8464
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -2.189e+01  4.987e+00  -4.389 1.57e-05 ***
## cylinders(5.5,8.01] -3.509e+00  7.464e-01  -4.701 3.94e-06 ***
## displace             3.396e-02  6.807e-03   4.989 1.02e-06 ***
## horsepower          -3.722e-02  1.499e-02  -2.484  0.01354 *
## weight              -6.237e-03  6.996e-04  -8.916  < 2e-16 ***
## acceleration         1.337e-01  1.060e-01   1.261  0.20816
## year                 7.872e-01  5.697e-02  13.818  < 2e-16 ***
## origin2              1.796e+00  6.198e-01   2.897  0.00404 **
## origin3              2.932e+00  5.940e-01   4.936 1.32e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.23 on 304 degrees of freedom
## Multiple R-squared:  0.8353, Adjusted R-squared:  0.831
## F-statistic: 192.7 on 8 and 304 DF,  p-value: < 2.2e-16
```

Q9. What is the optimal model complexity (number of parameters) in your regression?

The optimal choice is 7 parameters. NB: this is with seed 4268.

```
# MSE on test set
pred=predict(cvbest,newdata=ourAutoTest)
Q10testMSE=mean((pred-ourAutoTest$mpg)^2)
Q10testMSE
Q5testMSE
```

```
## [1] 9.136218
## [1] 8.931018
```

Q10. Report on the best model.

Same model as in Q4 was chosen, so see Q5 for results.

Problem 2: Shrinkage methods

2a) Lasso and ridge regression

Q11: Which figure is lasso and which is ridge.

The L_1 penalty forces some of the coefficients to be zero. This is the case for the predictors in Figure 1, so this figure corresponds to the lasso. The L_2 penalty shrinks the coefficients to zero more smoothly. This is the case for the predictors in Figure 2, so this figure corresponds to ridge.

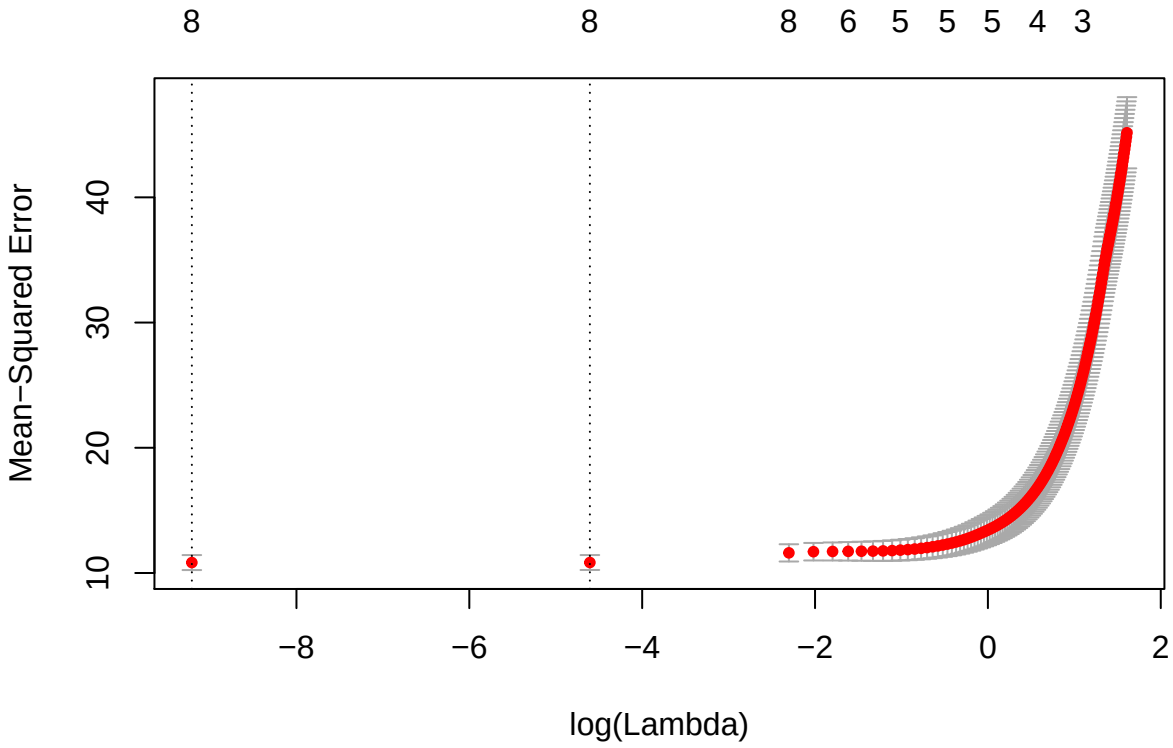
Q12: Explain effect of tuning parameter.

The tuning parameter puts a penalty on the coefficient estimates. As the tuning parameter increases, the flexibility of the regression fit decreases because the coefficients are shrinked towards zero. The variance decreases, but the bias increases. $\lambda = 0$: Least squares solution. $\lambda \to \infty$: A model with intercept only.

Q13: Model selection?

In ridge regression, all coefficients will shrink towards zero, but it will not set any of them exactly to zero (unless $\lambda = \infty$). Consequently, ridge regression will always generate a model involving all predictors. The lasso, however, forces some of the coefficient estimates to be exactly equal to zero when λ is sufficiently large. Hence, the lasso is more suitable for performing variable selection and is more similar to best subset selection.

In Figure 1 and Figure 2 we see that year and cylinders are the most important predictors. In 1b, we got that weight was the most important predictor, but year and cylinders were included in the best model with 3 covariates.



2b) Finding the optimal λ

Q14: What does `cv.glmnet`?

The function `cv.glmnet` uses cross validation to find the optimal λ, in this case 10-fold. Mean and standard deviation for validation set MSE is reported in a plot for each grid value of λ.

Q15: What do you see in the plot?

The plot shows the MSE with error estimates as a function of log(Lambda). The numbers on the top of the plot show the number of non-zero coefficients in the fitted model for each value of λ. One solution for optimal λ is to choose the one where MSE is at its minimum.

Our textbook authors suggests a slight variation to this solution, called `lambda.1se`, find the λ for the minimum MSE and then look at the standard deviation of the MSE at this point - and then move towards a more parsimonious model (increase λ) with mean MSE as the minimum MSE + one standard deviation. Read moe under `help(cv.glmnet)`.

Q16: Finding the optimal lambda:

```
lambdaopt=cv.out$lambda.1se
print(lambdaopt)
```

```
## [1] 0.01
```

3c) Prediction

Q17: Model fit

Fit model, coefficients, ..

```
lassofit=glmnet(x,y,alpha=1,lambda=lambdaopt,standardize=TRUE)
beta=coef(lassofit)
print(beta)
coef(lassofit)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                 s0
## (Intercept)         -21.475534414
## cylinders(5.5,8.01]  -3.383985180
## displace              0.031030021
## horsepower           -0.035228794
## weight               -0.006084397
## acceleration          0.124425729
## year                  0.782180280
## origin2               1.676410824
## origin3               2.832044860
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                                 s0
## (Intercept)         -21.475534414
## cylinders(5.5,8.01]  -3.383985180
## displace              0.031030021
## horsepower           -0.035228794
## weight               -0.006084397
## acceleration          0.124425729
## year                  0.782180280
## origin2               1.676410824
## origin3               2.832044860
```

Q18: Prediction

```
# 0 for cylinder, displace, horsepower, weight, acceleration, year, 1 for origin2 and 0 for origin3
newx=matrix(c(0,150,100,3000,10,82,1,0),nrow=1)
pred_x0=predict(lassofit,s=lambdaopt,newx=newx)
# or - alternatively
sum(c(1,newx)*coef(lassofit))
```

```
## [1] 28.46235
```

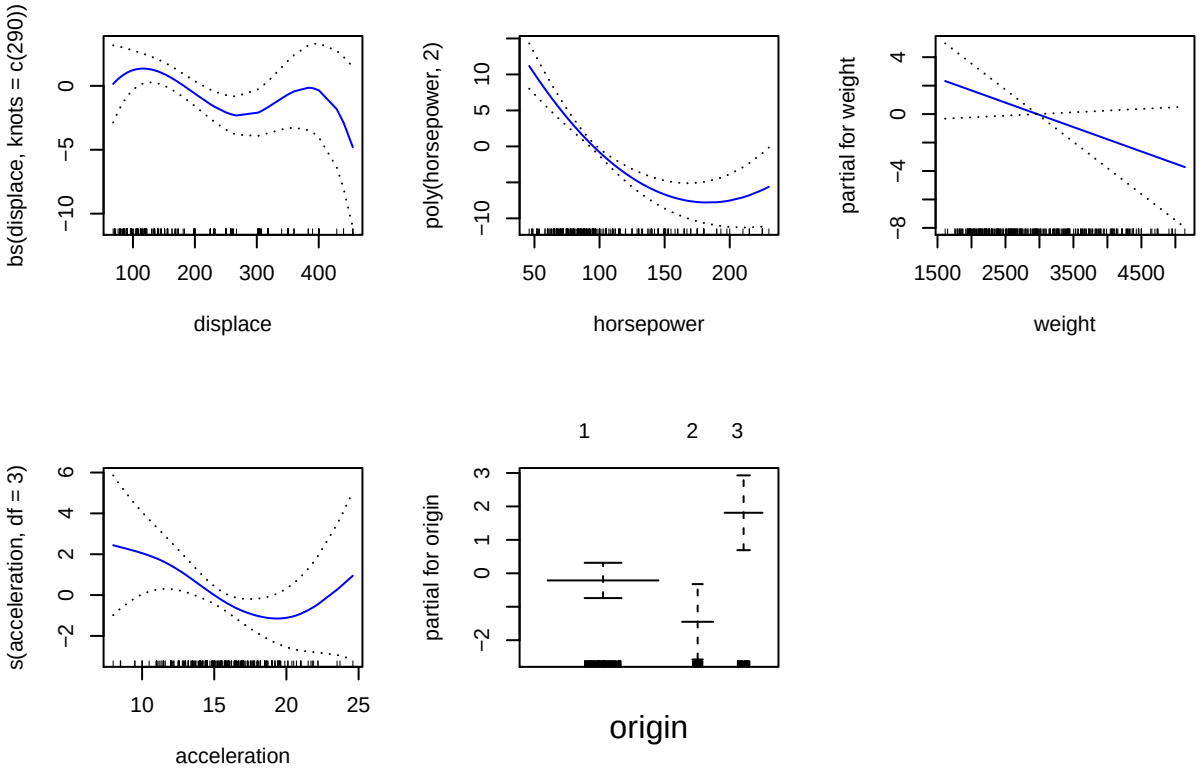
The predicted `mpg` 28.4623485.

Problem 3: Additiva non-linear regression

3a) Additive model

Q19: Fit additive model and comment.

```
library(gam)
fitgam=gam(mpg~bs(displace, knots=c(290))+
poly(horsepower,2)+weight+s(acceleration,df=3)+
origin,data=ourAutoTrain)
par(mfrow=c(2,3))
plot(fitgam,se=TRUE,col="blue")
summary(fitgam)
```




```
##
## Call: gam(formula = mpg ~ bs(displace, knots = c(290)) + poly(horsepower,
##     2) + weight + s(acceleration, df = 3) + origin, data = ourAutoTrain)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -10.7984  -2.2702  -0.3987   1.9718  15.8655
##
## (Dispersion Parameter for gaussian family taken to be 14.826)
##
##     Null Deviance: 19252.79 on 312 degrees of freedom
## Residual Deviance: 4447.809 on 299.9999 degrees of freedom
## AIC: 1746.946
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
```

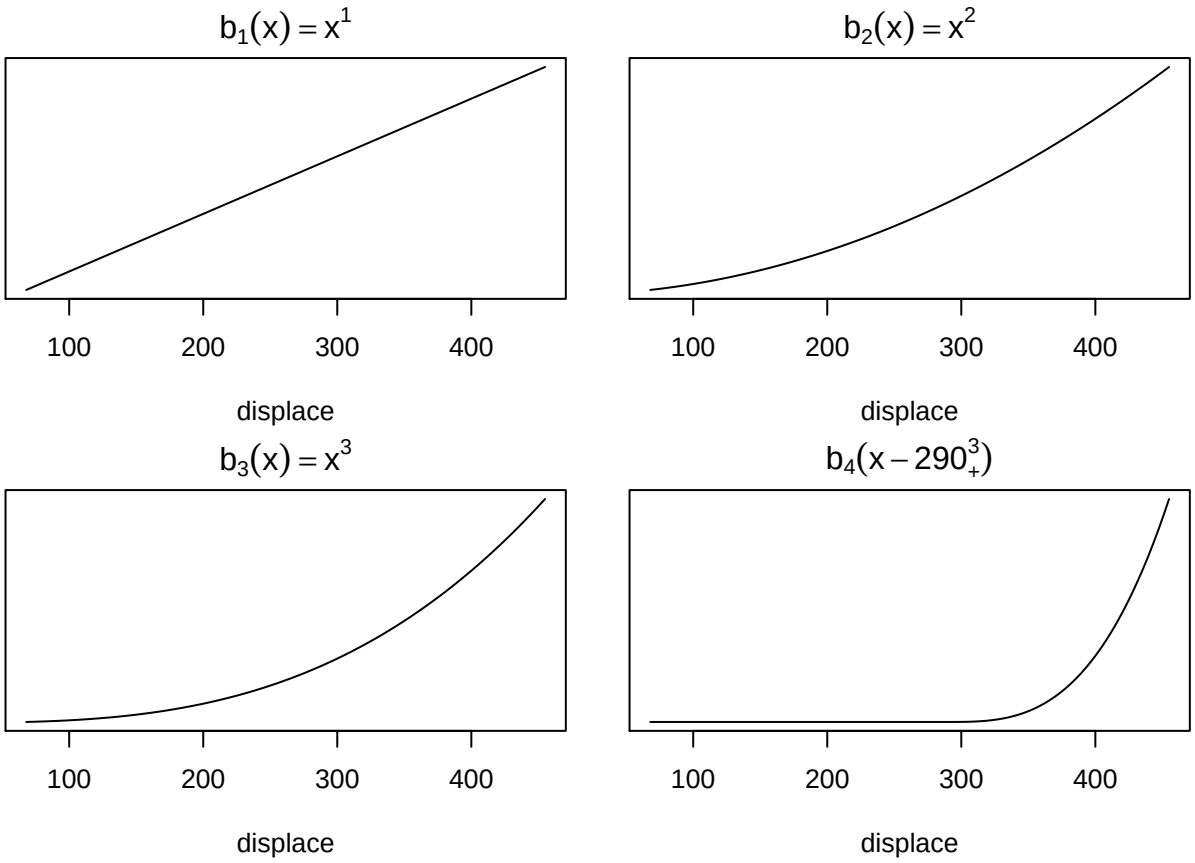
```
##                               Df  Sum Sq Mean Sq  F value    Pr(>F)
## bs(displace, knots = c(290))   4 13147.7  3286.9 221.6995 < 2.2e-16 ***
## poly(horsepower, 2)            2  1239.9   619.9  41.8141 < 2.2e-16 ***
## weight                         1   275.4   275.4  18.5740 2.219e-05 ***
## s(acceleration, df = 3)        1    61.2    61.2   4.1277   0.04307 *
## origin                         2   292.3   146.1   9.8562 7.150e-05 ***
## Residuals                    300  4447.8    14.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                               Npar Df Npar F   Pr(F)
## (Intercept)
## bs(displace, knots = c(290))
## poly(horsepower, 2)
## weight
## s(acceleration, df = 3)            2 3.4655 0.03251 *
## origin
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see `displace` has two peaks, `horsepower` has the smallest CI for low values, the linear function in `weight` is very variable for small and high values, `acceleration` looks rather like a cubic function and there is a clear effect of `origin`.

Q20: Basis for the cubic spline

The cubic spline (see Module 7 page): Here $M = 4$ for a cubic spline, and the basis is x, x^2,x^3, $(x - 290)^3_+$. In `gam` an orthogonal version of this is used.

```
M=4
cp=290
l=range(ourAutoTrain$displace)
par(mfrow=c(2,2),mar=c(4,1,2,1))
s=sapply(1:(M-1), function(y) curve(x^y, l[1],l[2],yaxt="n",
                                    xlab="displace",ylab="",main=bquote(b[.(y)](x)==x^.(y))))
s=sapply(1:1, function(y) curve(pmax(0,x-cp)^3,l[1],l[2],yaxt="n",
                                    xlab="displace",ylab="",main=bquote(b[.(y+M-1)](x-.(cp[y])["+"]^.(M
```


$b_1(x) = x^1$
100
200
300
400
displace
$b_2(x) = x^2$
100
200
300
400
displace
$b_3(x) = x^3$
100
200
300
400
displace
$b_4(x - 290^3_+)$
100
200
300
400
displace