# Module 7: Solutions to recommended Exercises

TMA4268 Statistical Learning V2020

*Andreas Strand, Martina Hall, Michail Spitieris, Stefanie Muff, Department of Mathematical Sciences, NTNU*

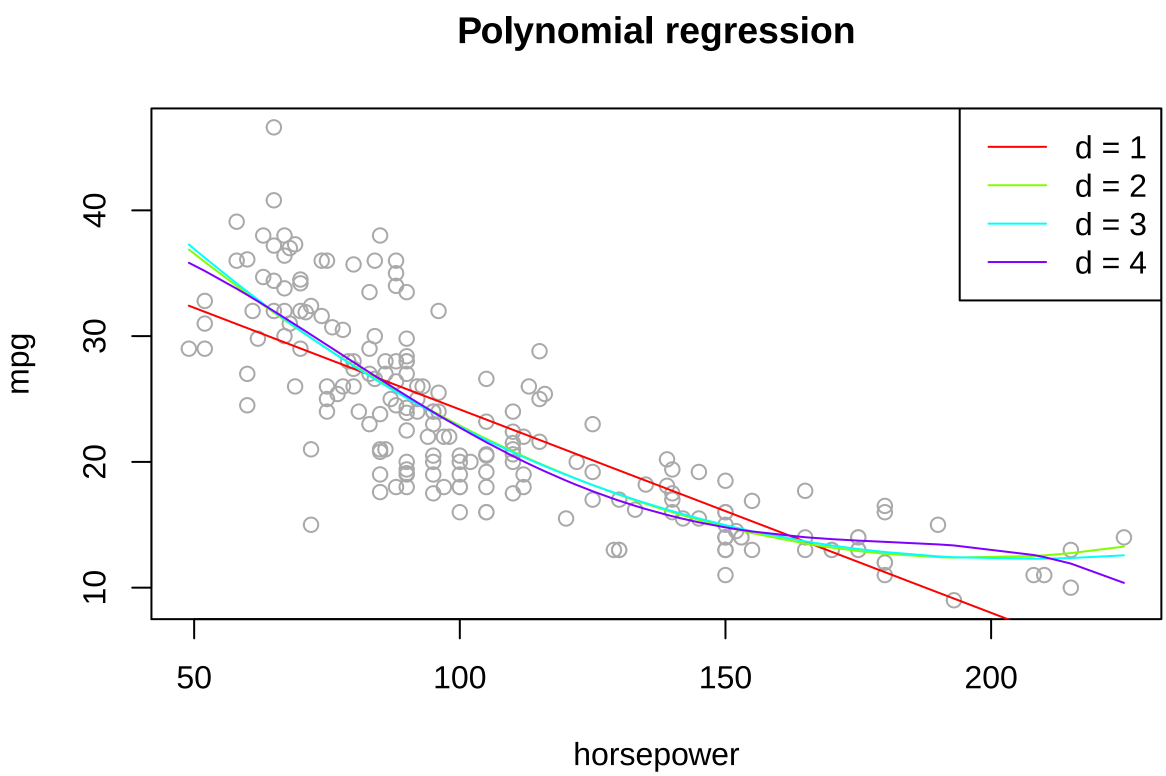
*February 27, 2020*

## Problem 1

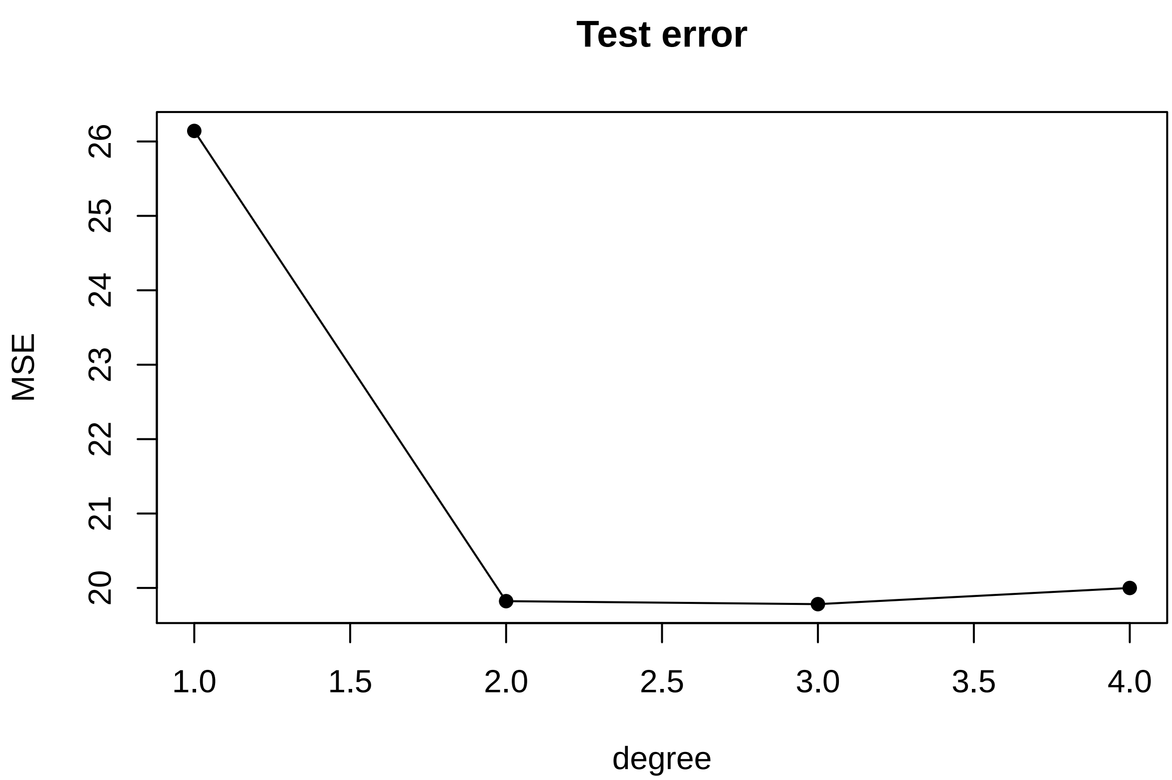
The code below performs polynomial regression of degree 1, 2, 3 and 4. The function `sapply()` is an efficient for loop. We iterate over all degrees to plot the fitted values and compute the test error. Finally we plot the test error by polynomial degree.

```
library(ISLR)
# extract only the two variables from Auto
ds = Auto[c("horsepower","mpg")]
n = nrow(ds)
#which degrees we will look at
deg = 1:4
set.seed(1)
#training ids for training set
tr = sample.int(n, n/2)
# plot of training data
plot(ds[tr,], col = "darkgrey", main = "Polynomial regression")

# which colors we will plot the lines with
co = rainbow(length(deg))
# iterate over all degrees (1:4) - could also use a for-loop here
MSE = sapply(deg, function(d){
  #fit model with this degree
  mod = lm(mpg ~ poly(horsepower,d),ds[tr,])
  #add lines to the plot - use fitted values (for mpg) and horsepower from training set
  lines(cbind(ds[tr,1],mod$fit)[order(ds[tr,1]),], col = co[d])
  #calculate mean MSE - this is returned in the MSE variable
  mean((predict(mod,ds[-tr,])- ds[-tr,2])^2)
})
#add legend to see which color corresponds to which line
legend("topright", legend = paste("d =",deg), lty = 1, col = co)
```

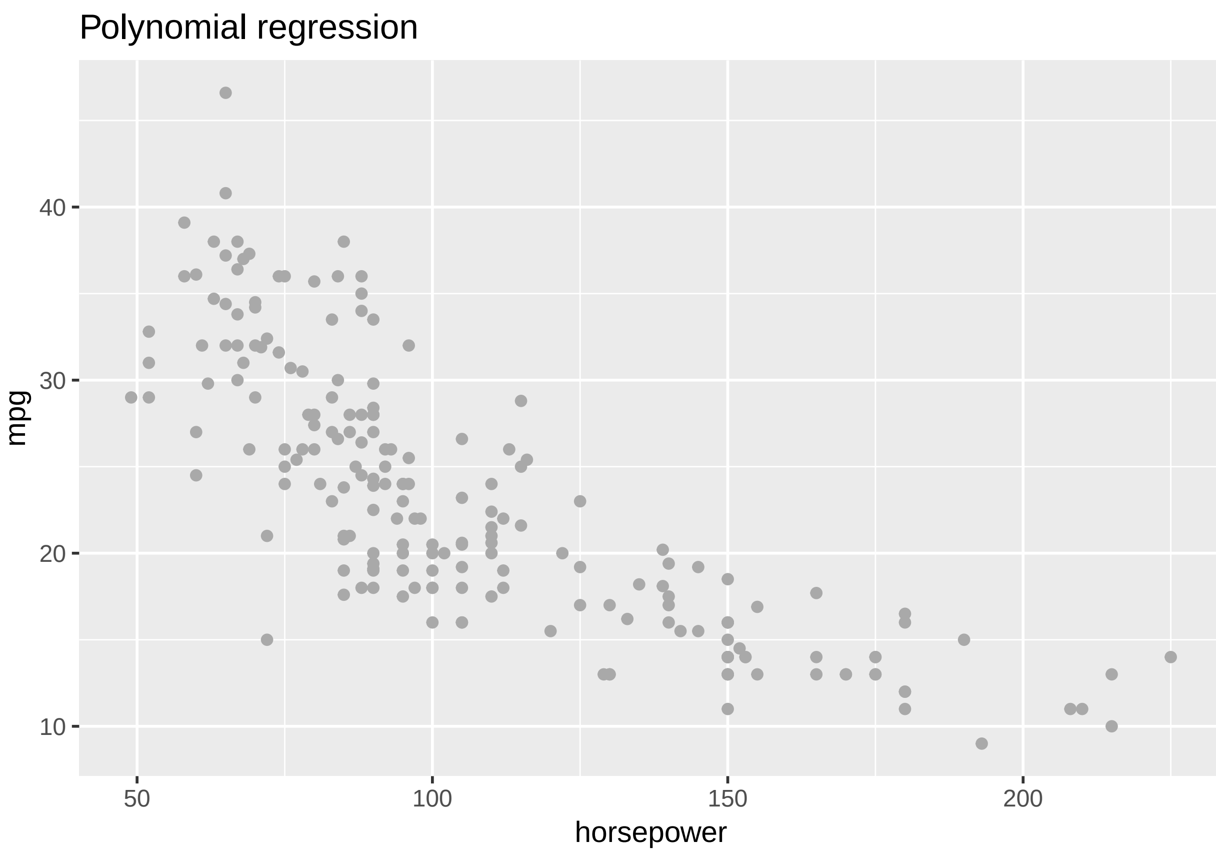



```
#plot MSE
plot(MSE, type="o", pch = 16, xlab = "degree", main = "Test error")
```



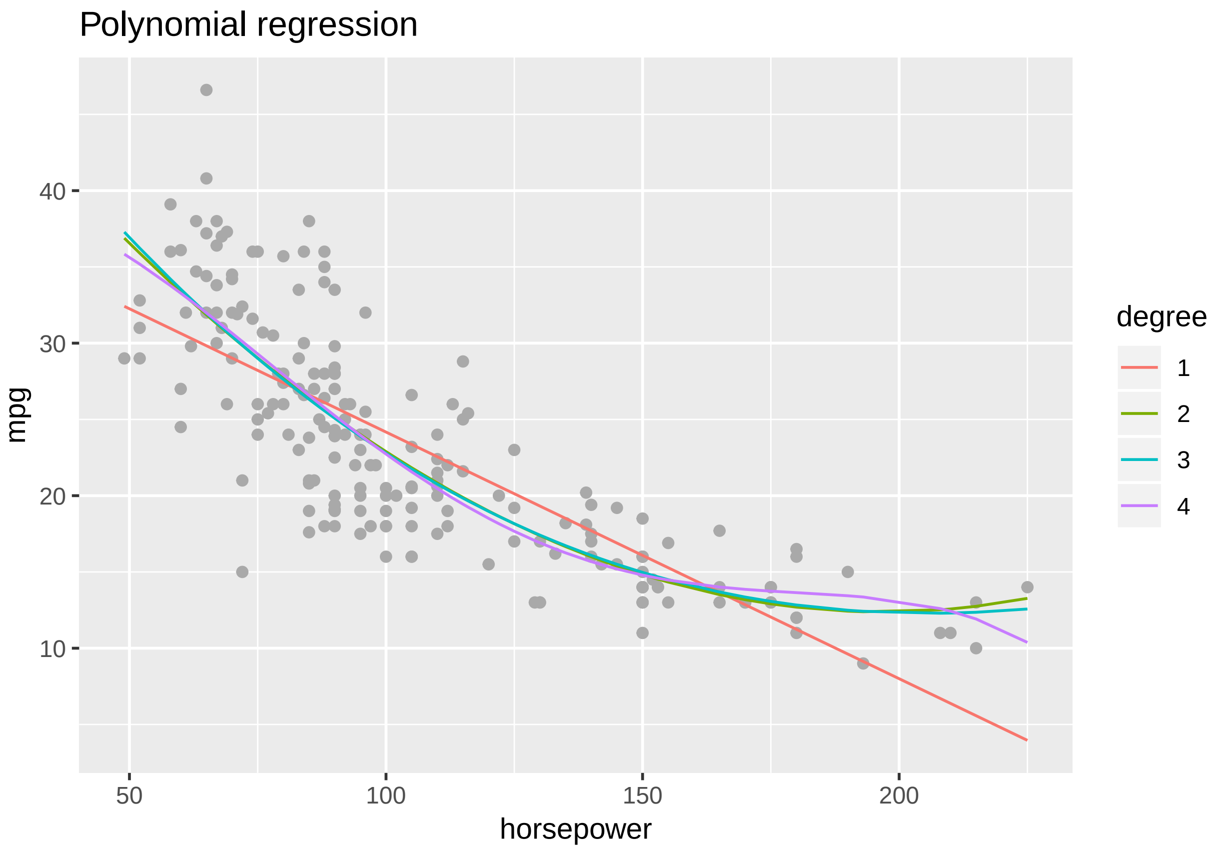

The same solution using `ggplot` is shown below.

```r
# solution with ggplot
library(ISLR)
library(ggplot2)
# extract only the two variables from Auto
ds = Auto[c("horsepower","mpg")]
n = nrow(ds)
#which degrees we will look at
deg = 1:4
set.seed(1)
#training ids for training set
tr = sample.int(n, n/2)
# plot of training data
ggplot(data = ds[tr,], aes(x=horsepower, y=mpg)) + geom_point(color = "darkgrey") +
  labs(title = "Polynomial regression")
```

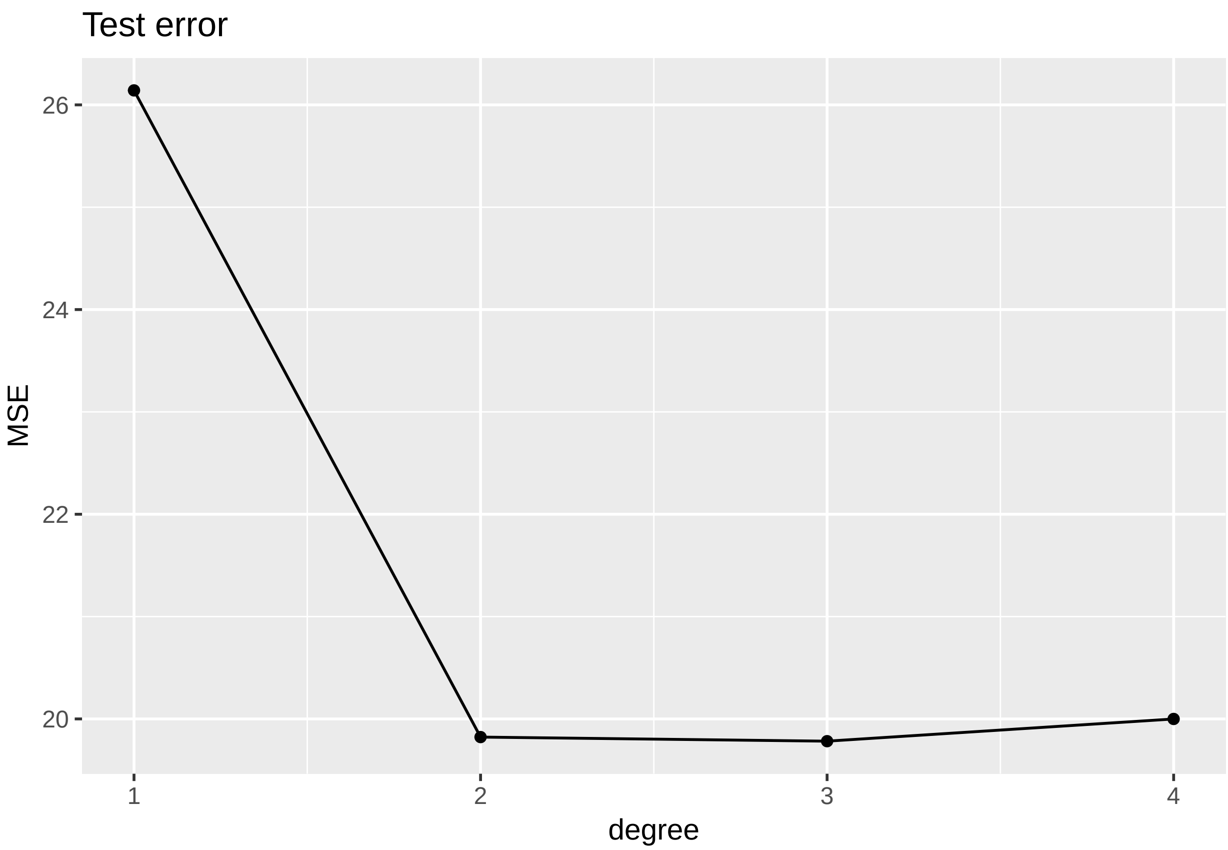



```r
# iterate over all degrees (1:4) - could also use a for-loop here
dat = c() #make a empty variable to store predicted values
for(d in deg){
  #fit model with this degree
  mod = lm(mpg ~ poly(horsepower,d),ds[tr,])
  #dataframe of predicted values - use fitted values (for mpg) and horsepower from
  #training set and add column (factor) for degree
  dat = rbind(dat, data.frame(horsepower = ds[tr,1], mpg = mod$fit,
                              degree = as.factor(rep(d, length(mod$fit)))))
  #calculate mean MSE - this is returned in the MSE variable
  MSE[d] = mean((predict(mod,ds[-tr,])- ds[-tr,2])^2)
}
```

```r
# plot fitted values for different degrees
ggplot(data = ds[tr,], aes(x=horsepower, y=mpg)) + geom_point(color = "darkgrey") +
  labs(title = "Polynomial regression") +
  geom_line(data = dat, aes(x=horsepower, y=mpg, color = degree))
```




```r
#plot MSE
MSEdata = data.frame(MSE = MSE, degree = 1:4)
ggplot(data = MSEdata, aes(x = degree, y = MSE)) + geom_line() +
  geom_point() + labs(title = "Test error")
```



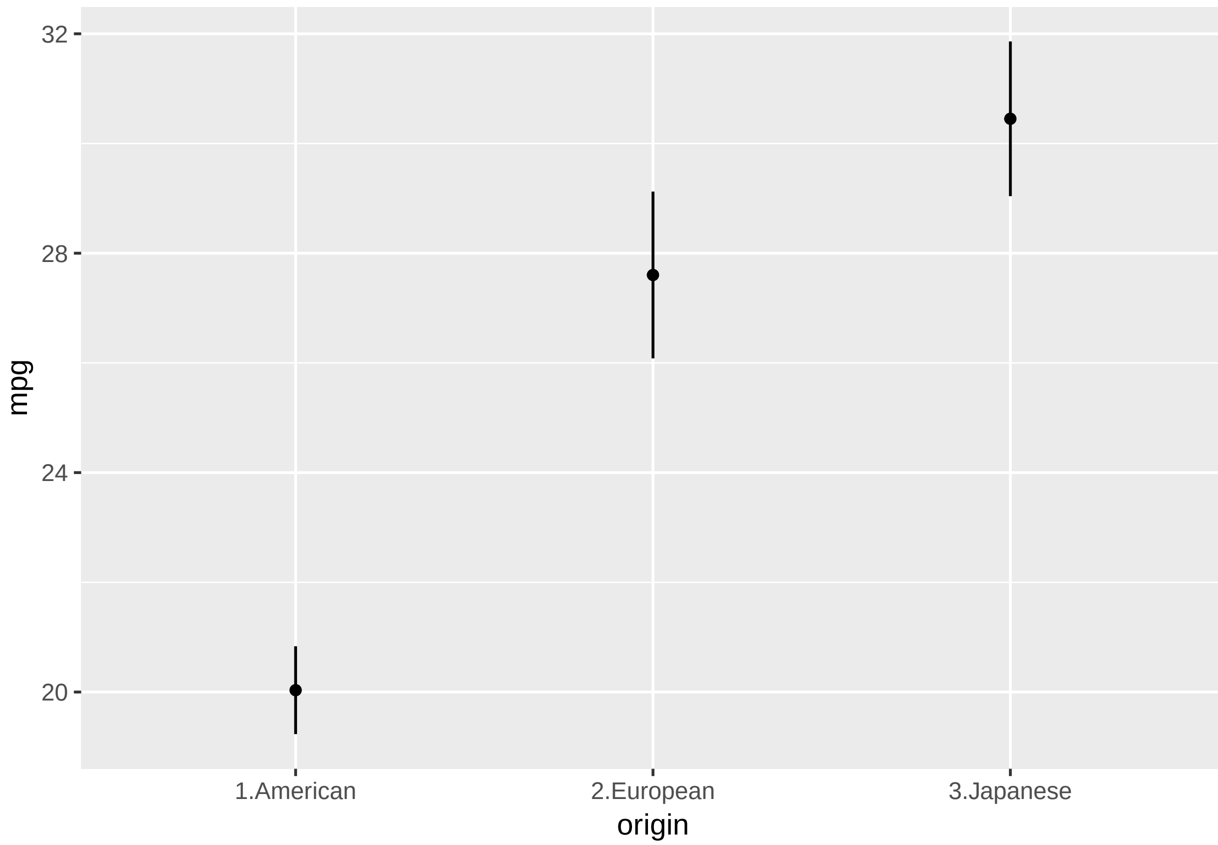

## Problem 2

We use `factor(origin)` for conversion to a factor variable. The function `predict(..., se = T)` gives fitted values with standard errors.

```
attach(Auto)
```

```
## The following object is masked from package:ggplot2:
##
##     mpg
```

```
#fit model
fit = lm(mpg ~ factor(origin))
#make a new dataset of the origins to predict the mpg for the different origins
new = data.frame(origin = as.factor(sort(unique(origin))))
# predicted values and standard errors
pred = predict(fit, new, se=T)
# dataframe including CI (z_alpha/2 = 1.96)
dat = data.frame(origin = new, mpg = pred$fit,
                 lwr = pred$fit - 1.96*pred$se.fit,
                 upr = pred$fit + 1.96*pred$se.fit)
# plot the fitted/predicted values and CI
ggplot(dat, aes(x=origin, y=mpg)) + geom_point() +
  geom_segment(aes(x=origin, y=lwr, xend = origin, yend=upr)) +
  scale_x_discrete(labels=c("1" = "1.American", "2" = "2.European","3" = "3.Japanese"))
```




## Problem 3

The request is a design matrix for a natural spline with $X =$ `year` and one knot $c_1 = 2006$. The boundary knots be the extreme values of `year`, that is $c_0 = 2003$ and $c_2 = 2009$. A general basis for a natural spline is

$$b_1(x_i) = x_i, \quad b_{k+2}(x_i) = d_k(x_i) - d_K(x_i),\ k = 0, \ldots, K-1,$$
$$d_k(x_i) = \frac{(x_i - c_k)_+^3 - (x_i - c_{K+1})_+^3}{c_{K+1} - c_k}.$$

In our case we have one internal knot, that is $K = 1$. Thus, $k$ takes only the value 0. The two basis functions are

$$\begin{aligned}
b_1(x_i) &= x_i, \\
b_2(x_i) &= d_0(x_i) - d_1(x_i) \\
&= \frac{(x_i - c_0)_+^3 - (x_i - c_2)_+^3}{c_2 - c_0} - \frac{(x_i - c_1)_+^3 - (x_i - c_2)_+^3}{c_2 - c_1} \\
&= \frac{1}{c_2 - c_0}(x_i - c_0)_+^3 - \frac{1}{c_2 - c_1}(x_i - c_1)_+^3 + \left(\frac{1}{c_2 - c_1} - \frac{1}{c_2 - c_0}\right)(x_i - c_2)_+^3 \\
&= \frac{1}{6}(x_i - 2003)_+^3 - \frac{1}{3}(x_i - 2006)_+^3 + \frac{1}{6}(x_i - 2009)_+^3.
\end{aligned}$$

The design matrix is obtained by $\{\mathbf{X}_2\}_{ij} = b_j(x_i)$. We can simplify the second basis function more by using the fact that the boundary knots are the extreme values of $x_i$, that is $2003 \leq x_i \leq 2009$. Thus,

$$b_2(x_i) = \frac{1}{6}(x_i - 2003)^3 - \frac{1}{3}(x_i - 2006)^3_+.$$

## Problem 4

The matrix $\mathbf{X}$ is obtained by using `cbind()` to join an intercept, a cubic spline, a natural cubic spline and a factor.

```
library(ISLR)
attach(Wage)
#install.packages("gam")
library(gam)
#X_1
mybs = function(x,knots) cbind(x,x^2,x^3,sapply(knots,function(y) pmax(0,x-y)^3))

d = function(c, cK, x) (pmax(0,x-c)^3-pmax(0,x-cK)^3)/(cK-c)
#X_2
myns = function(x,knots){
  kn = c(min(x), knots, max(x))
  K = length(kn)
  sub = d(kn[K-1],kn[K],x)
  cbind(x,sapply(kn[1:(K-2)],d,kn[K],x)-sub)
}

#X_3
myfactor = function(x) model.matrix(~x)[,-1]
# X = (1,X_1, X_2, X_3)
X = cbind(1,mybs(age,c(40,60)), myns(year, 2006), myfactor(education))
# fitted values with our X
myhat = lm(wage~X-1)$fit
# fitted values with gam
yhat = gam(wage ~ bs(age, knots = c(40,60)) + ns(year, knots = 2006) + education)$fit
all.equal(myhat,yhat)
```

```
## [1] TRUE
```

The fitted values `myhat` and `yhat` are equal. Both the design matrices $\mathbf{X}$ and the coefficients $\hat{\beta}$ differs, but $\mathbf{X}\hat{\beta}$ are the same.

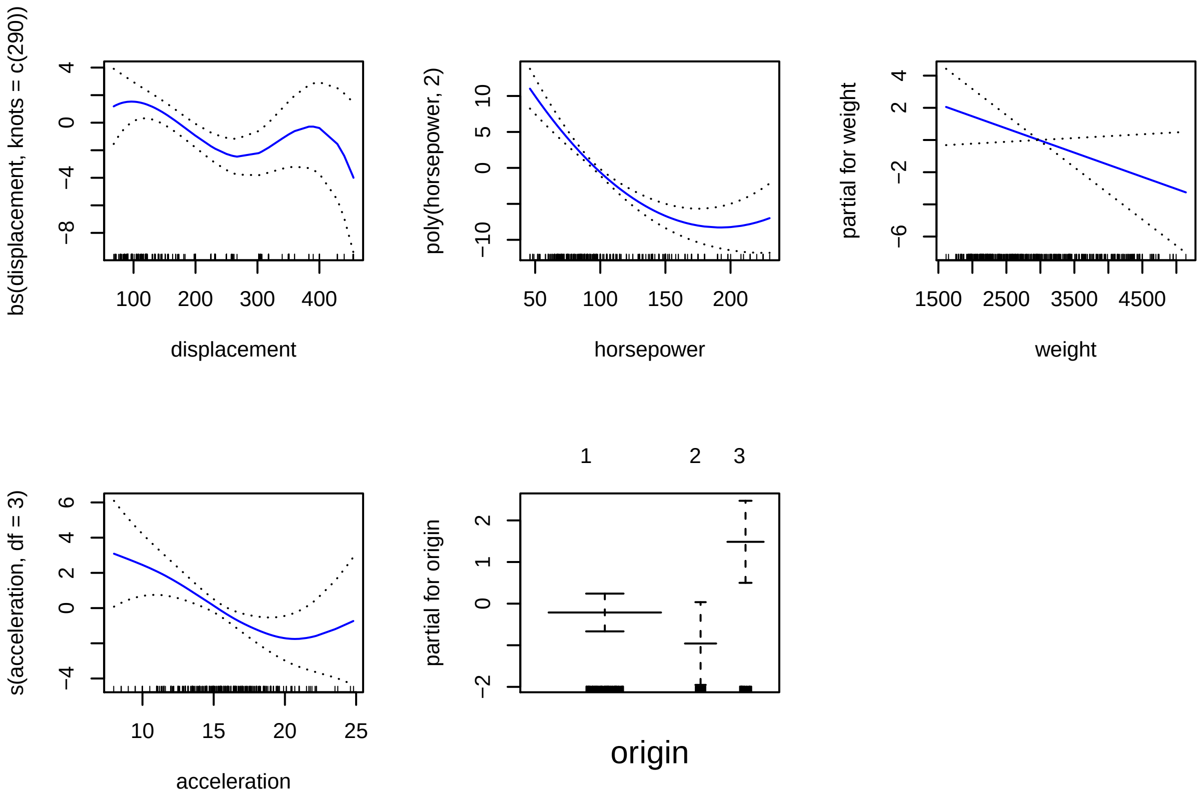
## Problem 5

Fit additive model and commenting:

```
library(gam)
#first set origin as a factor variable
Auto$origin = as.factor(Auto$origin)
#gam model
fitgam=gam(mpg~bs(displacement, knots=c(290))+
poly(horsepower,2)+weight+s(acceleration,df=3)+
origin,data=Auto)
#plot covariates
par(mfrow=c(2,3))
plot(fitgam,se=TRUE,col="blue")
```

```r
#summary of fitted model
summary(fitgam)
```

```
##
## Call: gam(formula = mpg ~ bs(displacement, knots = c(290)) + poly(horsepower,
##     2) + weight + s(acceleration, df = 3) + origin, data = Auto)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5172  -2.3774  -0.2538   1.7982  15.9994
##
## (Dispersion Parameter for gaussian family taken to be 14.1747)
##
##     Null Deviance: 23818.99 on 391 degrees of freedom
## Residual Deviance: 5372.203 on 378.9999 degrees of freedom
## AIC: 2166.599
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##                                   Df  Sum Sq Mean Sq  F value    Pr(>F)
## bs(displacement, knots = c(290))   4 16705.2  4176.3 294.6301 < 2.2e-16
## poly(horsepower, 2)                2  1283.6   641.8  45.2786 < 2.2e-16
## weight                             1   318.9   318.9  22.4970 2.985e-06
## s(acceleration, df = 3)            1   128.1   128.1   9.0362 0.0028231
## origin                             2   213.8   106.9   7.5422 0.0006137
## Residuals                        379  5372.2    14.2
##
## bs(displacement, knots = c(290)) ***
## poly(horsepower, 2)              ***
## weight                           ***
## s(acceleration, df = 3)          **
## origin                           ***
## Residuals
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Anova for Nonparametric Effects
##                                  Npar Df Npar F   Pr(F)
## (Intercept)
## bs(displacement, knots = c(290))
## poly(horsepower, 2)
## weight
## s(acceleration, df = 3)                2 2.9111 0.05563 .
## origin
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```




We see `displacement` 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`.

## Problem 6

The fitted values are obtained by $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$. In R this is `S%*%y`. The effective degrees of freedom is defined as

$$df_\lambda = \sum_{i=1}^{n} \frac{1}{1+\lambda d_i}.$$

This summation is done in one line in R.

```
K = function(x){
  xi = sort(unique(x))
  n = length(xi)
  h = xi[-1]-xi[-n]
  i = seq.int(n-2)
  D = diag(1/h[i], ncol = n)
  D[cbind(i,i+1)] = - 1/h[i] - 1/h[i+1]
  D[cbind(i,i+2)] = 1/h[i+1]
  W = diag(h[i]+h[i+1]/3)
  W[cbind(i[-1],i[-1]-1)] = h[i[-1]]/6
  W[cbind(i[-1]-1,i[-1])] = h[i[-1]]/6
  t(D)%*%solve(W)%*%D
}

lambda = 2000
```

```
x = sort(unique(age))
y = wage[order(age[!duplicated(age)])]
eig = eigen(K(x))
U = eig$vectors
d = eig$values
S = U%*%diag(1/(1+lambda*d))%*%t(U)

myhat = S%*%y
yhat = smooth.spline(x, y, df = sum(1/(1+lambda*d)))$y

plot(x,y, main = "Comparison of fitted values")
co = c("blue", "red")
w = c(5,2)
lines(x,myhat, lwd = w[1], col = co[1])
lines(x,yhat, lwd = w[2], col = co[2])
legend("topright", legend = c("myhat", "yhat"), col = co,
       lwd = w, inset=c(0, -0.19), xpd = T)
```