Lab 10

Attila Lazar

09.12.2020

data

We load the dataset, and define a train and a test set.

```
data(Auto, package="ISLR")

set.seed(123)
n <- nrow(Auto)
train <- sample(1:n, round(n*2/3))
test <- (1:n) [-train]
str(Auto)
```

```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 
```

1)

We train a linear model using *lm* and *natural cubic splines*. Since cylinders and origin are categorical variables, they enter the model linearly.

```
library(splines)
model1 <- lm(mpg~ns(displacement, 4) + ns(horsepower, 4) + ns(acceleration, 4) + ns(weight, 4) + origin
```

a)

we interpret the model using *summary*

significantly contributig variables are *horsepower*, *year*, *weight* and 'acceleration

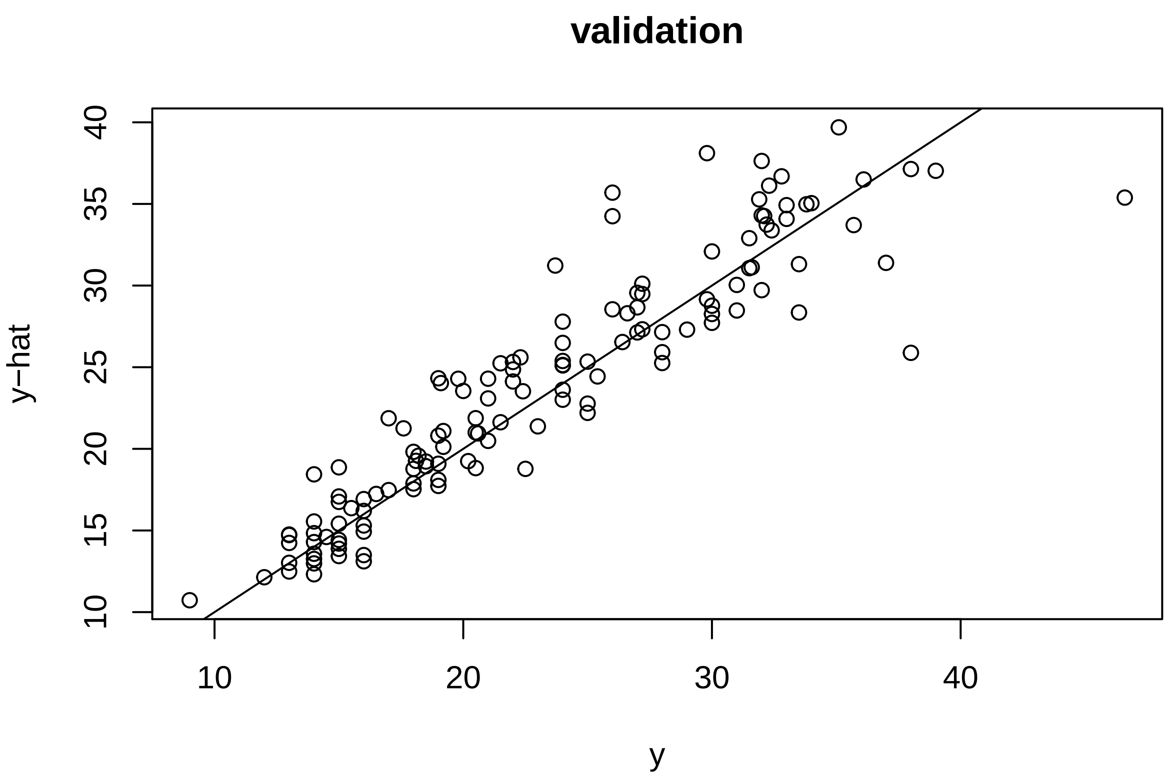
```
summary(model1)
```

```
##
## Call:
## lm(formula = mpg ~ ns(displacement, 4) + ns(horsepower, 4) +
##     ns(acceleration, 4) + ns(weight, 4) + origin + ns(year, 4) +
##     cylinders, data = Auto, subset = train)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -7.8917 -1.4840  0.1298  1.4435  7.4804
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             40.1461     3.8229  10.501  < 2e-16 ***
## ns(displacement, 4)1  -2.3655     2.2176  -1.067 0.287185
## ns(displacement, 4)2  -3.1748     2.9829  -1.064 0.288268
## ns(displacement, 4)3  -2.3071     4.2553  -0.542 0.588212
## ns(displacement, 4)4  -5.3432     3.4005  -1.571 0.117445
## ns(horsepower, 4)1    -8.1527     1.9244  -4.237 3.25e-05 ***
## ns(horsepower, 4)2   -12.6926     2.4476  -5.186 4.60e-07 ***
## ns(horsepower, 4)3   -22.5621     4.2721  -5.281 2.89e-07 ***
## ns(horsepower, 4)4   -11.9723     2.6830  -4.462 1.25e-05 ***
## ns(acceleration, 4)1  -6.2436     2.7551  -2.266 0.024340 *
## ns(acceleration, 4)2  -5.5801     1.9391  -2.878 0.004370 **
## ns(acceleration, 4)3 -11.2939     5.7682  -1.958 0.051403 .
## ns(acceleration, 4)4  -4.0390     2.5383  -1.591 0.112889
## ns(weight, 4)1        -7.9846     2.1543  -3.706 0.000261 ***
## ns(weight, 4)2        -7.4575     2.5580  -2.915 0.003892 **
## ns(weight, 4)3       -11.4807     4.4505  -2.580 0.010491 *
## ns(weight, 4)4        -6.2676     2.9929  -2.094 0.037304 *
## origin                 0.5120     0.3126   1.638 0.102748
## ns(year, 4)1          -0.3904     0.8922  -0.438 0.662109
## ns(year, 4)2           6.5663     0.8607   7.629 5.67e-13 ***
## ns(year, 4)3           6.1018     1.6303   3.743 0.000228 ***
## ns(year, 4)4           7.9807     0.7106  11.232  < 2e-16 ***
## cylinders              0.4917     0.4438   1.108 0.269070
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.631 on 238 degrees of freedom
## Multiple R-squared:  0.9008, Adjusted R-squared:  0.8916
## F-statistic: 98.25 on 22 and 238 DF,  p-value: < 2.2e-16
```

We see on the validation plot that the model predicts the test data quite well

```r
pred1 <- predict(model1, Auto[test,])
plot(Auto[test,'mpg'], pred1, xlab='y' ,ylab='y-hat', main="validation")
abline(c(0,1))
```



with RMSE of

```
sqrt(mean((Auto[test, 'mpg'] - predict(model1, Auto[test,]))^2))
```

```
## [1] 2.996651
```

b)

Now we use stepwise reduction of the model.

```
model2 <- step(lm(mpg~ns(displacement, 4) + ns(horsepower, 4) + ns(acceleration, 4) + ns(weight, 4) + o
summary(model2)
```

```
##
## Call:
## lm(formula = mpg ~ ns(horsepower, 4) + ns(acceleration, 4) +
##     ns(weight, 4) + origin + ns(year, 4), data = Auto, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.0631 -1.3062  0.1386  1.4229  7.5354
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           39.8651     2.9659  13.441  < 2e-16 ***
## ns(horsepower, 4)1    -7.9741     1.8850  -4.230 3.31e-05 ***
## ns(horsepower, 4)2   -11.0185     2.2895  -4.813 2.62e-06 ***
## ns(horsepower, 4)3   -20.3118     4.1708  -4.870 2.01e-06 ***
## ns(horsepower, 4)4   -11.3916     2.6121  -4.361 1.91e-05 ***
```

```
## ns(acceleration, 4)1  -4.9883     2.3906  -2.087 0.037968 *
## ns(acceleration, 4)2  -4.8724     1.7464  -2.790 0.005689 **
## ns(acceleration, 4)3  -8.7794     5.1606  -1.701 0.090176 .
## ns(acceleration, 4)4  -2.4111     2.3575  -1.023 0.307463
## ns(weight, 4)1        -8.4890     1.7382  -4.884 1.89e-06 ***
## ns(weight, 4)2        -9.5785     1.8445  -5.193 4.37e-07 ***
## ns(weight, 4)3       -12.4559     3.7075  -3.360 0.000906 ***
## ns(weight, 4)4        -9.0271     2.1770  -4.147 4.67e-05 ***
## origin                 0.7628     0.2743   2.781 0.005851 **
## ns(year, 4)1          -0.1018     0.8745  -0.116 0.907416
## ns(year, 4)2           6.5648     0.8400   7.816 1.65e-13 ***
## ns(year, 4)3           6.1643     1.5999   3.853 0.000149 ***
## ns(year, 4)4           8.1951     0.6958  11.777  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.639 on 243 degrees of freedom
## Multiple R-squared:  0.8981, Adjusted R-squared:  0.891
## F-statistic:   126 on 17 and 243 DF,  p-value: < 2.2e-16
```

This eliminates the variable *displacement* and gives us a slightly better RMSE

```r
sqrt(mean((Auto[test, 'mpg'] - predict(model2, Auto[test,]))^2))
```

```
## [1] 2.961918
```

c)

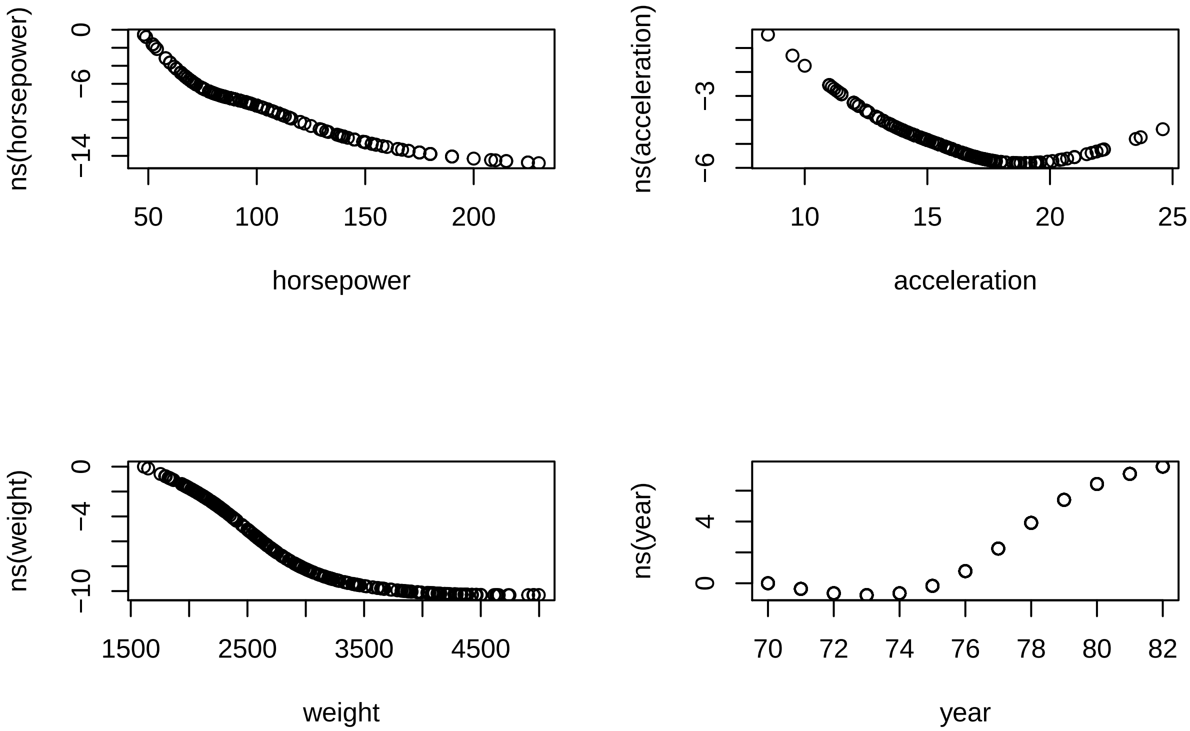
The model consists of the Intercept, the coefficients for each spline, and a coefficient for linearly modeled variables.

```r
model2$coefficients
```

```
##          (Intercept)   ns(horsepower, 4)1   ns(horsepower, 4)2
##           39.8651495           -7.9740653          -11.0185477
##   ns(horsepower, 4)3   ns(horsepower, 4)4 ns(acceleration, 4)1
##          -20.3117557          -11.3915600           -4.9882998
## ns(acceleration, 4)2 ns(acceleration, 4)3 ns(acceleration, 4)4
##           -4.8723536           -8.7794202           -2.4110593
##       ns(weight, 4)1       ns(weight, 4)2       ns(weight, 4)3
##           -8.4889941           -9.5785026          -12.4559214
##       ns(weight, 4)4               origin         ns(year, 4)1
##           -9.0270585            0.7627804           -0.1018131
##         ns(year, 4)2         ns(year, 4)3         ns(year, 4)4
##            6.5648195            6.1643067            8.1951039
```

We plot the calculated value of the splines in the model against the original variable

```r
par(mfrow=c(2,2))
plot(Auto$horsepower[train], model2$model$`ns(horsepower, 4)` %*% model2$coefficients[2:5], xlab='horsep
plot(Auto$acceleration[train], model2$model$`ns(acceleration, 4)` %*% model2$coefficients[6:9], xlab='ac
plot(Auto$weight[train], model2$model$`ns(weight, 4)` %*% model2$coefficients[10:13], xlab='weight', yla
plot(Auto$year[train], model2$model$`ns(year, 4)` %*% model2$coefficients[15:18], xlab='year', ylab='ns
```



In these plots we see how the variable enters the model. For *horsepower* and *weight* we see a near linear, negative trend which is expected.

Interestingly *acceleration* over 20 positively affects *mpg* reversing the trend. This may be becouse there are only few datapoints which may affect the model

Lastly *year* affects *mpg* negatively until 73, after that *year* strongly increases *mpg*. This may be attributed to the 1973 oil crisis.

2)

a)

We use *gam* to compute *Generalized Additive Models*. As in Ex1, we do not construct splines for *origin* and *cylinders*, since these are categorical variables.

```
library(mgcv)
```

```
## Loading required package: nlme

## This is mgcv 1.8-24. For overview type 'help("mgcv-package")'.
```

```
model3 <- gam(mpg~s(displacement) + s(horsepower) + s(acceleration) + s(weight) + origin + s(year) + cyl
```

b)

```
summary(model3)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## mpg ~ s(displacement) + s(horsepower) + s(acceleration) + s(weight) +
##     origin + s(year) + cylinders
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  21.4749     1.9928  10.776   <2e-16 ***
## origin        0.6412     0.2845   2.254   0.0251 *
## cylinders     0.1965     0.3611   0.544   0.5868
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df      F  p-value
## s(displacement) 1.315  1.543  0.549 0.385032
## s(horsepower)   2.863  3.648  5.946 0.000329 ***
## s(acceleration) 2.144  2.756  2.078 0.083199 .
## s(weight)       2.376  3.037 10.204 2.23e-06 ***
## s(year)         8.529  8.931 37.053  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.899   Deviance explained = 90.6%
## GCV = 7.0162  Scale est. = 6.4725    n = 261
```

We see that the model attributes no significance to *displacement* and only little significance to *acceleration*

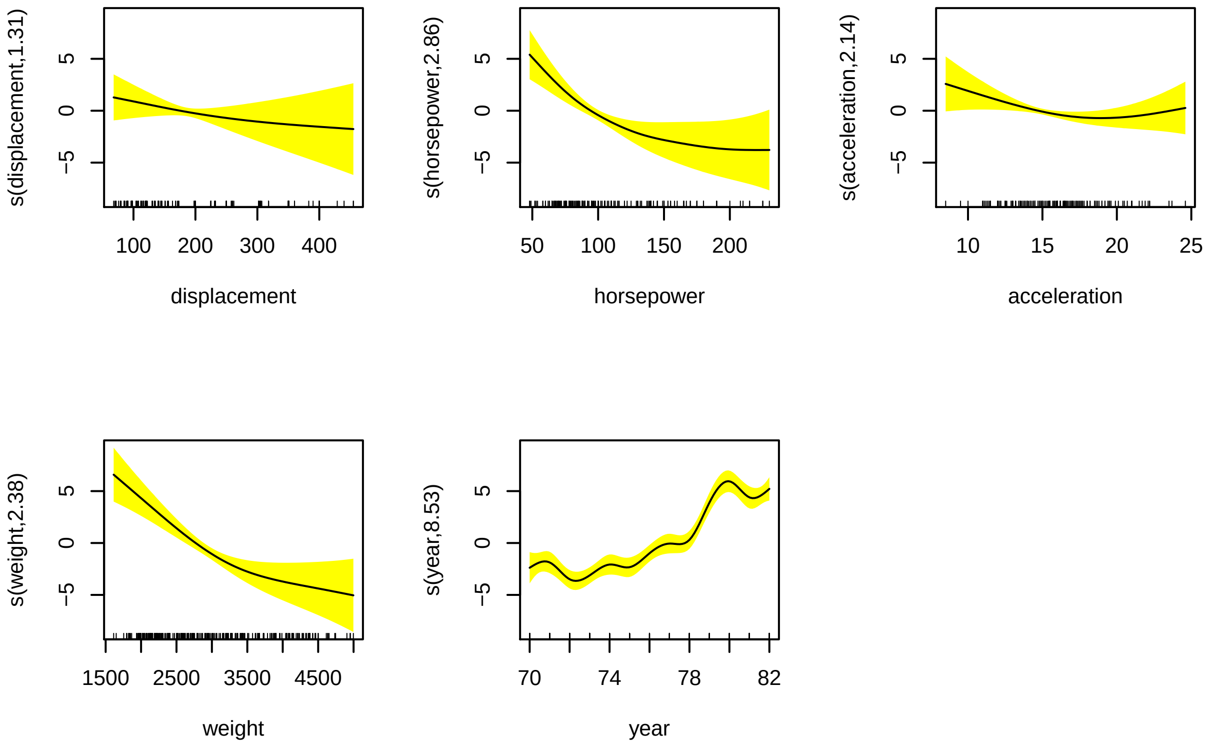
Also we see that the smooth function for *displacement* is near linear with edf = 1.3. In constrast it is quite complex for *year* with edf = 8.5 .

c)

We plot the smmoth functions. We can see how the variable enters the model and how it affects the predicted variable.

The smooth function for *year* seems to be to complex which might lead to overfitting.

```
plot(model3, page=1,shade=TRUE,shade.col = "yellow")
```



d)

```r
sqrt(mean((Auto[test, 'mpg'] - predict(model3, Auto[test,]))^2))
```

```
## [1] 2.90203
```

e)

first we try to enchance our model by manually restricting the choice of k value for *year*. We see a good improvement of the RMSE and also the complexity of the smooth function is reduced.

```r
model5 <- gam(mpg~s(displacement) + s(horsepower) + s(acceleration) + s(weight) + origin + s(year, k=3)
#plot(model5, page=1,shade=TRUE,shade.col = "yellow")
summary(model5)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## mpg ~ s(displacement) + s(horsepower) + s(acceleration) + s(weight) +
##     origin + s(year, k = 3) + cylinders
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  20.7398     2.0464   10.13   <2e-16 ***
## origin        0.6953     0.2959    2.35   0.0196 *
```

```
## cylinders     0.3154     0.3712    0.85   0.3964
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df       F  p-value
## s(displacement) 1.187  1.331   0.628 0.363472
## s(horsepower)   2.837  3.613   6.753 0.000103 ***
## s(acceleration) 2.254  2.899   2.626 0.041460 *
## s(weight)       2.571  3.282  10.035 1.83e-06 ***
## s(year)         1.962  1.998 124.006  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.885   Deviance explained = 89.1%
## GCV = 7.7478  Scale est. = 7.3378    n = 261
```

```r
sqrt(mean((Auto[test, 'mpg'] - predict(model5, Auto[test,]))^2))
```

```
## [1] 2.838914
```

Next we try the option bs=ts. This results in simular model than our ouriginal smooth model.

```r
model7 <- gam(mpg~s(displacement,bs='ts') + s(horsepower,bs='ts') + s(acceleration,bs='ts') + s(weight,b
#plot(model7, page=1,shade=TRUE,shade.col = "yellow")
summary(model7)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## mpg ~ s(displacement, bs = "ts") + s(horsepower, bs = "ts") +
##     s(acceleration, bs = "ts") + s(weight, bs = "ts") + origin +
##     s(year, bs = "ts") + cylinders
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 22.81788    1.63261  13.976  < 2e-16 ***
## origin       0.73922    0.27140   2.724  0.00693 **
## cylinders   -0.07633    0.27534  -0.277  0.78185
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                     edf Ref.df      F  p-value
## s(displacement) 0.07847      9  0.008   0.3226
## s(horsepower)   3.12305      9  2.469 7.68e-06 ***
## s(acceleration) 2.09404      9  0.674   0.0302 *
## s(weight)       2.52872      9  5.199 3.62e-12 ***
## s(year)         8.52977      9 36.514  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.899   Deviance explained = 90.6%
## GCV =  7.001  Scale est. = 6.4818    n = 261
```

```r
sqrt(mean((Auto[test, 'mpg'] - predict(model7, Auto[test,]))^2))
```

```
## [1] 2.907251
```

We also try the option bs='cr'. The smooth function complexities are reduced slightly and we improve the RMSE

```r
model8 <- gam(mpg~s(displacement,bs='cr') + s(horsepower,bs='cr') + s(acceleration,bs='cr') + s(weight,b
#plot(model8, page=1,shade=TRUE,shade.col = "yellow")
summary(model8)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## mpg ~ s(displacement, bs = "cr") + s(horsepower, bs = "cr") +
##     s(acceleration, bs = "cr") + s(weight, bs = "cr") + origin +
##     s(year, bs = "cr") + cylinders
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  21.7625     2.1100  10.314   <2e-16 ***
## origin        0.5881     0.2933   2.005   0.0461 *
## cylinders     0.1591     0.3824   0.416   0.6777
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df      F  p-value
## s(displacement) 1.427  1.728  1.008 0.436540
## s(horsepower)   3.276  4.132  5.192 0.000423 ***
## s(acceleration) 2.749  3.515  2.823 0.037482 *
## s(weight)       3.399  4.297  7.328 8.98e-06 ***
## s(year)         5.099  6.159 47.091  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.894   Deviance explained = 90.1%
## -REML = 624.76  Scale est. = 6.79      n = 261
```

```r
sqrt(mean((Auto[test, 'mpg'] - predict(model8, Auto[test,]))^2))
```

```
## [1] 2.856505
```

Now set the option select = TRUE. We arrive at our best RMSE.

```r
model9 <- gam(mpg~s(displacement,bs='cr') + s(horsepower,bs='cr') + s(acceleration,bs='cr') + s(weight,b
#plot(model9, page=1,shade=TRUE,shade.col = "yellow")
summary(model9)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## mpg ~ s(displacement, bs = "cr") + s(horsepower, bs = "cr") +
```

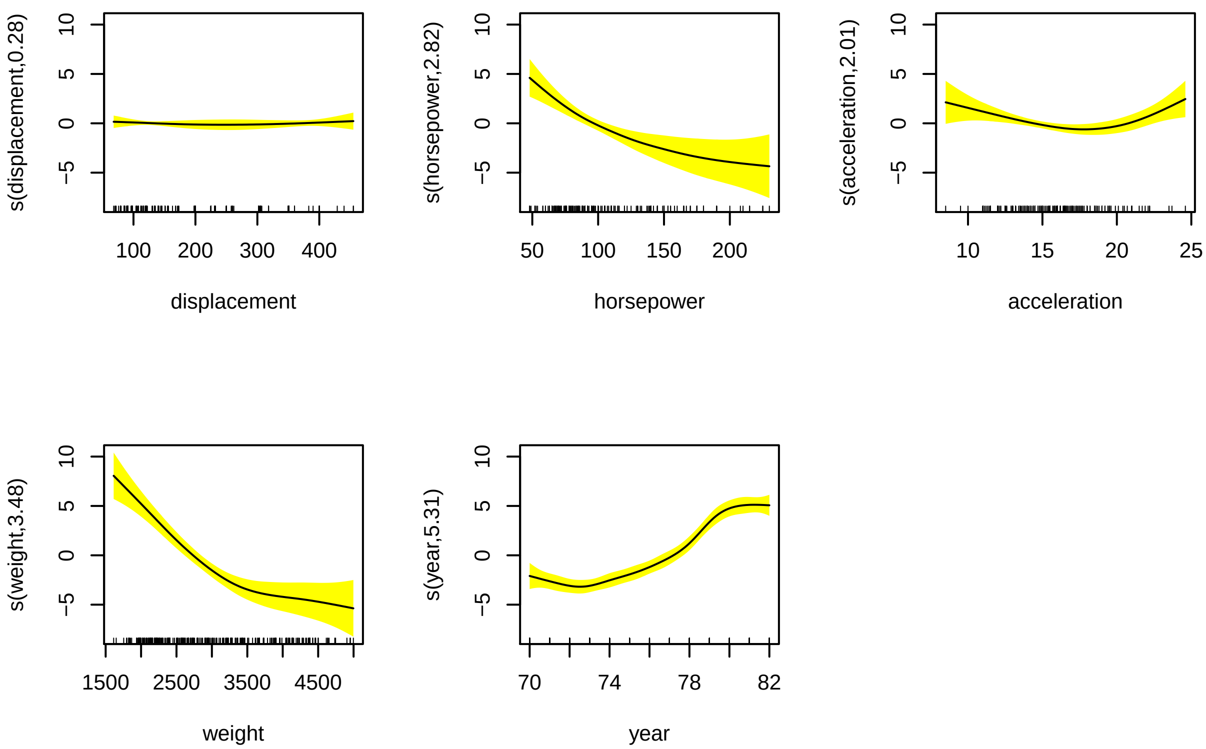
```
##     s(acceleration, bs = "cr") + s(weight, bs = "cr") + origin +
##     s(year, bs = "cr") + cylinders
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 22.96481    1.69648  13.537   <2e-16 ***
## origin       0.66607    0.27716   2.403    0.017 *
## cylinders   -0.08243    0.28540  -0.289    0.773
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                   edf Ref.df      F  p-value
## s(displacement) 0.281      9  0.037  0.23046
## s(horsepower)   2.821      9  2.943 1.35e-07 ***
## s(acceleration) 2.010      9  0.961  0.00291 **
## s(weight)       3.476      9  7.708  < 2e-16 ***
## s(year)         5.314      9 33.370  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.894   Deviance explained =   90%
## -REML = 640.45  Scale est. = 6.7912    n = 261
```

```
sqrt(mean((Auto[test, 'mpg'] - predict(model9, Auto[test,]))^2))
```

```
## [1] 2.83679
```

We plot the smooth functions.

```
plot(model9, page=1,shade=TRUE,shade.col = "yellow")
```



```r
plot(Auto[test,'mpg'], predict(model9, Auto[test,]), xlab='y' ,ylab='y-hat', main="validation")
abline(c(0,1))
```