# Assignment 3

Abhiram Ravipati

2024-03-03

Chapter 6

1)

a) The model with the fewest k predictors will typically be the best subset model because best subset selection considers all possible predictor combinations for each k and selects the model that reduces the training RSS for that specific k.

b) It is impossible to determine with certainty which of the three models with k predictors has the lowest test RSS without testing the models on a test dataset. How well the model works on the test dataset depends on its generalizability to previously unknown data which may vary for each of the three selection strategies. Best subset selection increases computational complexity but is more likely to provide a model with better generalisation performance because it considers all possible subsets of predictors.

c)

i) True, There are k variables (predictors) in the k-variable model. Because it only involves including one more variable and applying forward step-wise selection once again, a k variable model determined through forward step-wise selection certainly will be a subset of a (k+1) variable model found by forward step-wise selection.

ii) True, The above reasoning can also be used to explain this. The k variable model is by default included in the list of models that are considered backwards and steps. When one additional variable is added and the same process is applied, the predictors in the k variable model become a subset of the predictors in the (k+1) model.

iii) False, This may not always be the case, but it might be occasionally. Initially, the backward step-wise considers k predictors and generates a model; it then eliminates the least significant predictor and generates a model with k-1 predictors, and so on. The approach will be completely different in the future; we will add predictors one at a time after starting with one. The predictors in the k+1 variable model are therefore not guaranteed to be a subset of the k= variable model ascertained through the use of backward step-wise.

iv) False, For an understanding of why iv is false we could consider the same explanation provided for iii.

v) True

2)

a) (iii) We understand that several predictor coefficients in Lasso become absolutely zero as the lambda value increases, decreasing the model's flexibility. The bias variance trade-off actually occurs, resulting in a considerable decrease in variance and a modest increase in model bias.

b) (iii) The same reasoning that we applied to Lasso earlier also remains true here. Ridge penalty lowers variance but decreases flexibility in the model. The disadvantage of ridge regression is that it uses all predictors, which causes predictor coefficient values to decline but not approach zero as in lasso.

c) (ii) We know that non-linear models usually have more variance and lower bias, and they are more flexible. If we take into consideration the estimated MSE equation, which consists of three components: square of bias, variance, and irreducible error, we may assert intuitively that the prediction accuracy of a non-linear model will be good when the decrease in bias is larger than the increase in variance.

3)

a) (iv) An interpretable explanation for this can be found in the contour plots of the coefficients (of predictors) and the value of s (square for Lasso). Although there is a maximum size that can be permitted, the goal of using the lasso is to find the set of coefficient estimations that produce the shortest RSS. If s is large enough, the Lasso coefficients will resemble the coefficients of least squares. Anywhere along the curve, the oval shapes on the graph reflect the same RSS. In addition, the RSS of the outer ellipse is higher than that of the inner one. The value of training RSS continues to decline as s grows, hence (iv) is the correct answer. The square size now grows and enters the inner ellipses when the constraint shifts from 0 to some positive values.

b) (ii) When s=0, the only potential coefficient values are zeroes, aside from the intercept. This indicates that our null model, which is independent of all variables, exists at s=0. As s grows and moves closer to least square estimates, the model's flexibility increases. From our foundational knowledge, we know that when model flexibility increases, bias decreases, variance rises, and test MSE first decreases before increasing at a specific point. Consequently, we determined that option (ii) was the best one.

c) (iii) Steadily Increase. This can also be explained by the same idea that was used to answer question b. The value of variance increases gradually as s increases because the model becomes more flexible. This process continues until s reaches a value where the coefficients agree with the least square estimate.

d) (iv) Steadily Increase. We can use the same concept as before in this case. The bias constantly drops as s increases from 0 because of the improved flexibility of the model, until the least square estimate is limited by s.

e) (v) Remains Constant. Regardless of the model's quality, noise in the system (from overlooking the unknown element) is the cause of the irreducible mistake. As a result, we do not observe a corresponding rise or fall in irreducible error, making the solution reliant on the flexibility of the model or the values of its coefficients.

4)

a) (iii) Steadily Increase. It should be noted that the above equation equals LSE (Least Squares Estimate) when lambda=0. The model with the shortest training RSS is the one constructed with least squares coefficients. As lambda increases, the coefficient level curves move further from the LSE. Consequently, the training error keeps getting higher.

b) (ii) Decreases First, then rise gradually in a U-shape. We add penalty to the Least Square estimate (referred to as ridge or Lasso depending on the penalty chosen) in order to offset the over-fitting problem that LSE normally produces. The model constructed in this way tends to underfit the model and gets better at reducing test error as lambda rises. This, however, is not sustainable, and eventually, test error increases when bias increases and variance decrease more than it does.

c) (iv) Steadily Decreases, The model becomes less flexible as the value increases since the coefficient value decreases towards zero. To minimise variation, we use shrinkage techniques (until it is very negligible). The model's variance approaches zero as the coefficients go closer to zero.

d) (iii) Steadily Increases, we can use the same intuition that you used to answer the last question in this situation. As λ increases, the model's flexibility decreases and its variation tends to decrease. The bias tends to increase as the coefficients approach zero as λ increases.

e) (v) Remains Constant, The coefficients or penalty amount used in the previous equation have no bearing on irreducible mistakes. It is independent of the value of λ since it results from system noise. Thus, the solution.

5) Ridge regression and lasso regression can be explored in this basic scenario with n = 2, p = 2, and particular constraints on the data:

a) Ridge Regression Optimization Problem: The main goal is to minimize the below cost function:

$$L(\beta) = \sum_{i=1}^{n}\left(y_i - \beta_o - \sum_{j=1}^{P} x_{ij}\,\beta_j\right)^2 + \lambda\sum_{j=1}^{P}\beta_j^2$$

where, n is the number of observations (Here it is 2) p is the number of predictors (Here it is 2) $y_i$ is the response variable $x_{ij}$ is the ith observation of the jth predictor. $\beta_j$ are the coefficients to be estimated. $\beta_o$ is the intercept $\lambda$ is the regularization parameter

Given that the $\beta_o$ and $y_1 + y_2 = 0$, the ridge regression problem can be simplified to

$$L(\beta) = \sum_{i=1}^{2}\left(y_1 - \sum_{j=1}^{2} x_{ij}\,\beta_j\right)^2 + \lambda\sum_{j=1}^{2}\beta_j^2$$

b) Ridge Regression Coefficient Estimates: In this scenario, where $\beta_o = 0$ and $y_1 + y_2 = 0$ it can be said that the ridge coefficient estimates $\beta_1$ and $\beta_2$ are equal. This is due to the fact that when predictors are associated, ridge regression tends to reduce the coefficient estimates towards each other. Ridge regression will result in comparable coefficient values for correlated variables since $x_{11} = x_{12}$ and $x_{21} = x_{22}$.

c) Lazzo Optimization Problem: In this Lazzo Regression, the main goal is to minimize the below cost fucntion:

$$L(\beta) = \sum_{i=1}^{n}\left(y_i - \beta_o - \sum_{j=1}^{P} x_{ij}\,\beta_j\right)^2 + \lambda \sum_{j=1}^{P}|\beta_j|$$

d) Lasso Coefficient Estimates: In this particular scenario, the lasso coefficients $\beta_1$ and $\beta_2$ are not unique. This is because the lasso penalty $\left(\lambda \sum_{j=1}^{P}|\beta_j|\right)$ sets some coefficients to exactly zero, which tends to generate sparsity. The lasso may set either $\beta_1$ or $\beta_2$ or both to zero while estimating the other coefficient in this situation, where $y_1 + y_2 = 0$ and $\beta = 0$. The lasso optimisation issue has several alternative solutions, and depending on the particular optimisation path and the value of the regularisation parameter $\lambda$, it may select different coefficients to be zero. The particular values of $\lambda$, $x_ij$, and $y_i$ will determine the precise answers.

Chapter 7

2)

a) Since $\int \left[g^{(m)}(x)\right]^2 dx$ may be seen as the sum of $\left[g^{(m)}(x)\right]^2$ throughout the whole x range, the g that minimizes is ĝ. It is obvious why the penalty term,$\lambda \int \left[g^{(m)}(x)\right]^2 dx$, will approach 0 for large lambda values.Conversely, if λ = 0, then this term is entirely removed from the equation, allowing us to choose any value of g that minimizes the loss function $\sum n_i = 1\left(y_i - g(x_i)\right)^2$.

b) It is possible that in this case, as $\lambda = \infty$, $g^{!}(x)$ will trend towards zero when g(x) is some constant c. Therefore, $\hat{g}(x) = c$, a constant.Consequently, (G(x) will be a straight line parallel to the X-axis.) $c = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the value of the constant c that lowers RSS.

c) Since $\lambda = \infty$, $g^{!!}(x) = 0$, the second derivative of g(x) is compelled to zero. While the second derivative of all linear equations is zero, this is possible if g(x) is a linear equation of the form $g(x) = ax + b$, which is the line we obtain using least squares.

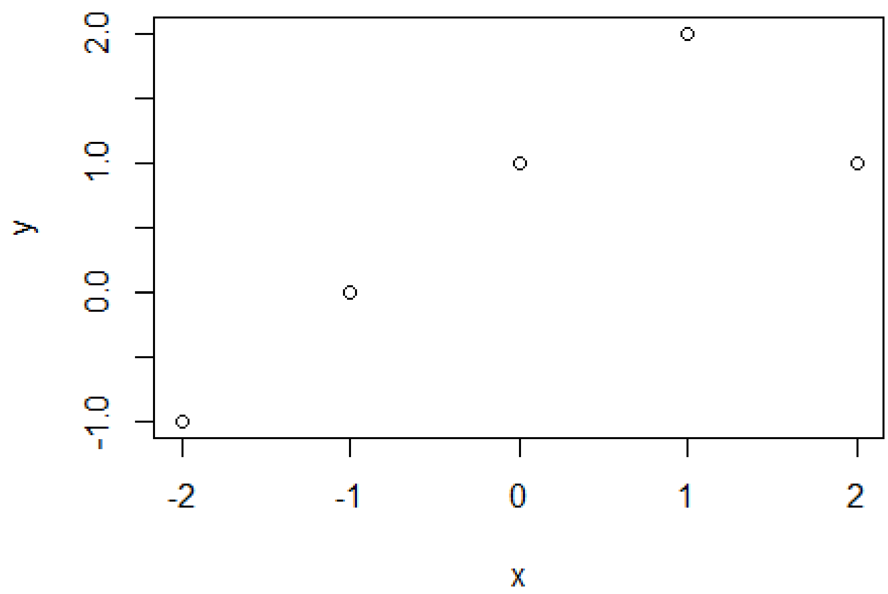
d) We can express that as $g^{!!!}(x) = 0$, $\lambda = \infty$ using the same intuition as before.It will therefore be a quadratic of the type $ax2 + bx + c$ since $\hat{g}(x)$ will have the least RSS.

e) The $\hat{g}$ equation in the question shows us that the penalty term disappears as $\lambda = 0$, leaving only the loss term. Therefore, g(x) can take any shape that passes over every point in the training data in order to reduce RSS to zero. It can also be extremely flexible or over-fitting.

3) Since $Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \varepsilon$ is Eq1, let's suppose that Now let's determine the equations for the cases where X>=1 and X<1 separately. Two distinct functions result from doing this. Utilising all available data, we apply eq1 to obtain Since X<1: $\hat{y} = 1 + X$ -> Suppose that it is Eq2. When X>=1, then $\hat{y} = 1 + X - 2(X-1)^2$ =>$\hat{Y} = 1 + X - 2X^2 + 4X - 2$ =>$\hat{y} = -2X^2 + 5X - 1$ -> Let us assume this as Eq3. By solving the above equation, the region between X = -2 and X = 2 the curve will be a straight

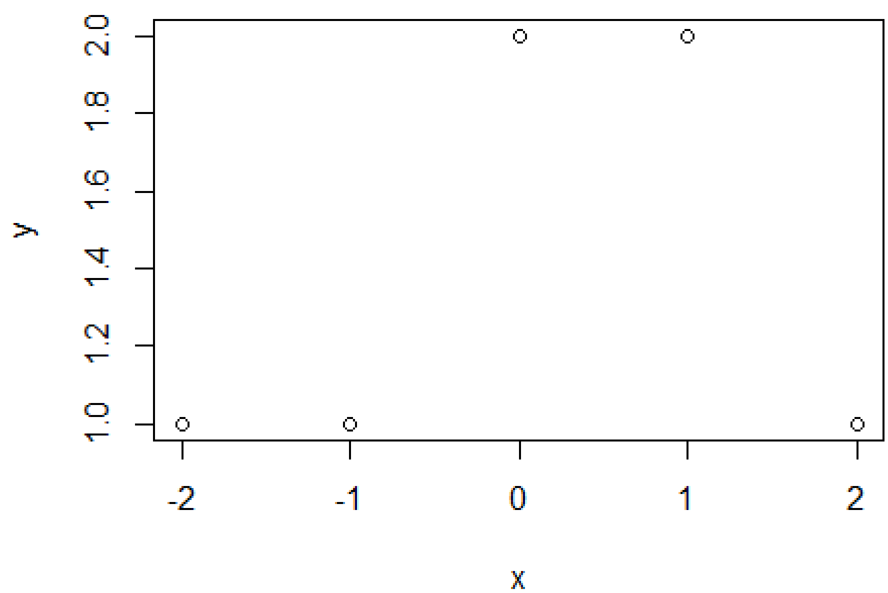
line, from X = -2 to X = 1 and from X =1 to X = 2 will be a quadratic curve. At X = 1.25 the critical point will happen which can be found by taking the first order derivative of Equation 3 and equating the obtained result with Zero.

```
x = -2:2
y = 1 + x+ -2 * (x - 1)^2 * I(x>1)
plot(x,y)
```




4) Let's replace all of the given values for each region in the given equation with y values. $\hat{y} = 1$ for -2 < X < 0 and $\hat{y} = 2$ for 0 ≤ X < 1. Since 1 ≤ X ≤ 2, $\hat{y} = 2 - X + 1 = 3 - X$ We don't need to check the equation for X > 2 because the provided question is just for the region between X = -2 and X = 2. Therefore, the intercept is 1 and the slope is zero for -2≤ X < 0. The intercept is two and the slope is zero for 0 ≤ X < 1. When 1 ≤ X < 2, the intercept is 3 and the slope is -1.

```
x = -2:2
y = c(1 + 0 + 0, # x = -2
      1 + 0 + 0, # x = -1
      1 + 1 + 0, # x = 0
      1 + (1 - 0) + 0, # x = 1
      1 + (1 - 1) + 0 # x = 2
)
plot(x,y)
```




5)

a) The previous formulas demonstrate unequivocally that the penalty term gets more significant as λ approaches infinity.

Since the third order derivative is zero, the highest order polynomial that satisfies this condition for $\lambda \to \infty$, for $\hat{g}_1$ and $\hat{g}_3$ (x) → 0 is g(x) = ax2 + bx + c. $\hat{g}_1$ will therefore be a quadratic that lowers training RSS. The greatest degree polynomial that satisfies this condition when $\lambda \to \infty$ for $\hat{g}_2$ and $\hat{g}_4$ (x) → 0 is of the form g(x) = ax3 + bx2 + cx + d (because the 4th order derivative is zero). $\hat{g}_2$ will therefore be a cubic that lowers training RSS. For a given large quantity of RSS, $\hat{g}_2$ will have less RSS than $\hat{g}_1$ because it is more flexible than $\hat{g}_1$.

b) It is unclear which of the above mentioned has a lower test RSS. The true relationship between the predictors and the order in which that relationship exists are the only factors that define it. In accordance with the true relationship, $\hat{g}_1$ and $\hat{g}_2$ may be over- or under-fit. In order to determine if $\hat{g}_1$ or $\hat{g}_2$ has the smaller test RSS, we can do this.

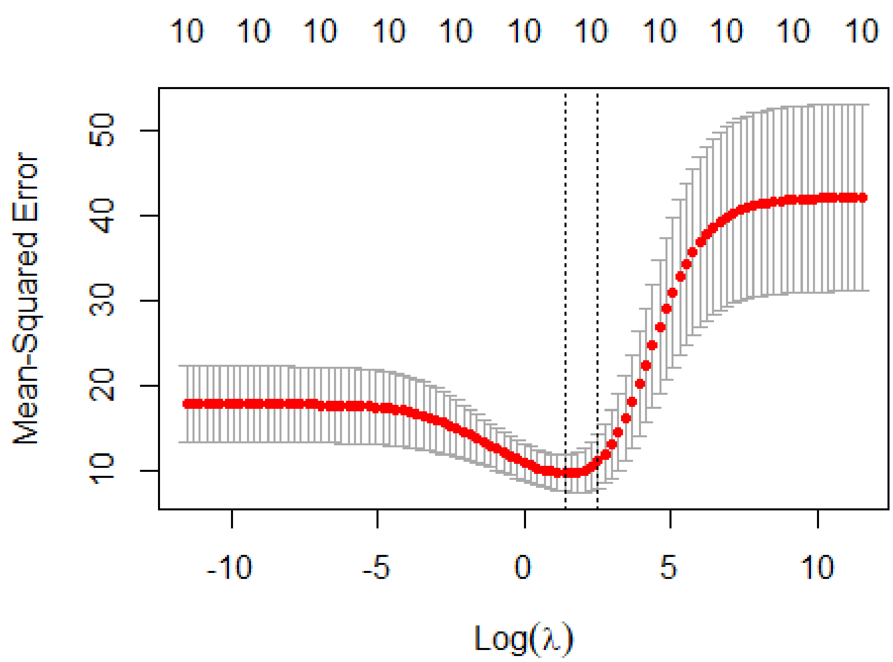
c) A value of λ equal to zero eliminates the penalty term completely. Consequently, if every xi is unique, then $\hat{g}_1$ and $\hat{g}_2$ would have the same training RSS, 0. There is no restriction on g, thus we could use any function to interpolate all of the training data. Since a model with a training RSS of zero that covers all training points would be incredibly over-fit and have a high test RSS, we are unable to be positive that the test RSS will be low. The test RSS for $\hat{g}_1$ and $\hat{g}_1$ would be the same if we assume that the same interpolating function was used for both. For example, if they were both linear splines with knots at each unique xi).

Problem 1:

```
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

library(ggplot2)
mt_cars_dataset <- data.frame(mtcars)
summary(mt_cars_dataset)

##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
##       drat             wt             qsec             vs        
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
##        am              gear            carb      
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000  
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
##  Median :0.0000   Median :4.000   Median :2.000  
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812  
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000  

str(mt_cars_dataset)

## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...

set.seed(200)
train_test_split_mt_cars <- createDataPartition(mt_cars_dataset$am, times =
1, p=0.8, list = FALSE)
```

```
train_mt_cars <- mt_cars_dataset[train_test_split_mt_cars,]
test_mt_cars <- mt_cars_dataset[-train_test_split_mt_cars,]

model_linear_mt_cars <- lm(mpg~., data= train_mt_cars)

mean((predict(model_linear_mt_cars, test_mt_cars)-test_mt_cars$mpg)^2)

## [1] 10.71549

summary(model_linear_mt_cars)

##
## Call:
## lm(formula = mpg ~ ., data = train_mt_cars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0200 -2.0955 -0.2192  1.3621  4.6315
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -17.79527   34.31617  -0.519   0.6116
## cyl          -0.10885    1.24904  -0.087   0.9317
## disp          0.02193    0.02167   1.012   0.3276
## hp           -0.01242    0.03012  -0.413   0.6858
## drat          0.65269    2.24277   0.291   0.7750
## wt           -5.30058    2.52253  -2.101   0.0529 .
## qsec          2.46523    1.61141   1.530   0.1469
## vs           -2.59087    3.43564  -0.754   0.4625
## am            2.71842    2.76117   0.985   0.3405
## gear          1.63422    2.10387   0.777   0.4494
## carb          0.07967    1.04162   0.076   0.9400
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.966 on 15 degrees of freedom
## Multiple R-squared:  0.8704, Adjusted R-squared:  0.7841
## F-statistic: 10.08 on 10 and 15 DF,  p-value: 5.523e-05

coef(model_linear_mt_cars)

##  (Intercept)          cyl         disp           hp         drat
wt
## -17.79526837  -0.10885352   0.02193177  -0.01242459   0.65268664  -
5.30057738
##         qsec           vs           am         gear         carb
##   2.46523037  -2.59087201   2.71842115   1.63421704   0.07966846

library(glmnet)

## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8

x <- model.matrix(mpg~., train_mt_cars)[,-1]
y <- train_mt_cars$mpg

sequence_lambda <- 10^seq(5, -5, by = -.1)
crossvalidation_ridge <- cv.glmnet(x,y, alpha= 0, lambda = sequence_lambda)

## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3
observations per
## fold

plot(crossvalidation_ridge)
```




```
lambda_bestvalue <- crossvalidation_ridge$lambda.min
lambda_bestvalue

## [1] 3.981072

model_ridge_mt_cars <- glmnet(x,y, alpha = 0, lambda = lambda_bestvalue)
summary(model_ridge_mt_cars)

##           Length Class     Mode
## a0         1     -none-    numeric
## beta      10     dgCMatrix S4
## df         1     -none-    numeric
## dim        2     -none-    numeric
## lambda     1     -none-    numeric
```

```
## dev.ratio  1      -none-     numeric
## nulldev    1      -none-     numeric
## npasses    1      -none-     numeric
## jerr       1      -none-     numeric
## offset     1      -none-     logical
## call       5      -none-     call
## nobs       1      -none-     numeric
```

```
coef(crossvalidation_ridge, s= "lambda.min")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) 19.533705869
## cyl         -0.368008786
## disp        -0.005720897
## hp          -0.011099008
## drat         1.156418468
## wt          -1.109528763
## qsec         0.203566030
## vs           0.804978288
## am           1.520934064
## gear         0.588710051
## carb        -0.497348516
```

```
x1 = model.matrix(mpg~., test_mt_cars)[,-1]
predict_model <- predict(model_ridge_mt_cars, newx = x1, type = "response")
mean((predict_model- test_mt_cars$mpg)^2)
```

```
## [1] 1.184656
```

Ridge Regression, as we can see, lowers the mean square error (MSE) on test data from 10.71 to 1.18. After completing Ridge Regression, the shift in coefficients is apparent. None of the coefficients are absolutely zero, but they have all decreased and are getting closer to zero. Therefore, we may contend that shrinkage rather than variable selection was carried out by Ridge regression.

Problem 2:

```
library(ggplot2)
library(lattice)
library(caret)

swiss_dataset <- data.frame(swiss)

set.seed(150)
test_train_split_swiss_dataset <-
createDataPartition(swiss_dataset$Fertility, p=0.8, list = FALSE)
train_data_swiss <- swiss_dataset[test_train_split_swiss_dataset,]
test_data_swiss <- swiss_dataset[-test_train_split_swiss_dataset,]
```

```
linear_model_swiss <- lm(Fertility~., train_data_swiss)
summary(linear_model_swiss)

##
## Call:
## lm(formula = Fertility ~ ., data = train_data_swiss)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.014  -5.942   1.329   3.491  15.717
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      66.16966   11.76082   5.626 2.90e-06 ***
## Agriculture      -0.17497    0.07982  -2.192  0.03552 *
## Examination      -0.05176    0.29772  -0.174  0.86303
## Education        -1.06932    0.23606  -4.530 7.32e-05 ***
## Catholic          0.11713    0.03946   2.969  0.00554 **
## Infant.Mortality  1.03247    0.41295   2.500  0.01756 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.167 on 33 degrees of freedom
## Multiple R-squared:  0.6893, Adjusted R-squared:  0.6422
## F-statistic: 14.64 on 5 and 33 DF,  p-value: 1.406e-07
```

Agriculture, Examination, Catholic and Infant Mortality are relevant features with coeffiecients -0.17497, -0.05176, 0.11713, 1.03247

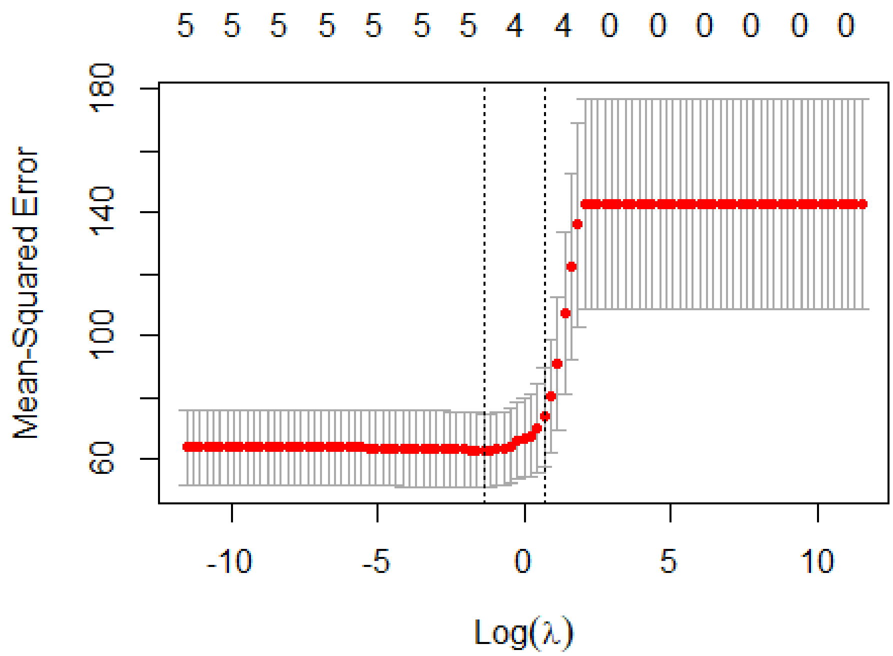
```
predict_mean <- mean((test_data_swiss$Fertility - predict(linear_model_swiss,
test_data_swiss))^2)
predict_mean

## [1] 59.91027
```

Lasso Regression:

```
library(Matrix)
library(foreach)
library(glmnet)
x <- model.matrix(Fertility~.,train_data_swiss)[,-1]
y <- train_data_swiss$Fertility
```

Cross Validation Lasso GLMNET:

```
sequence_lambda <- 10^seq(5, -5, by = -.1)
crossvalidation_lasso <- cv.glmnet(x, y, alpha = 1, lambda = sequence_lambda)
plot(crossvalidation_lasso)
```




```
lambda_bestvalue <- crossvalidation_lasso$lambda.min
lambda_bestvalue

## [1] 0.2511886

model_ridge_regression <- glmnet(x, y, alpha = 1, lambda = lambda_bestvalue)
summary(model_ridge_regression)

##           Length Class     Mode
## a0        1      -none-    numeric
## beta      5      dgCMatrix S4
## df        1      -none-    numeric
## dim       2      -none-    numeric
## lambda    1      -none-    numeric
## dev.ratio 1      -none-    numeric
## nulldev   1      -none-    numeric
## npasses   1      -none-    numeric
## jerr      1      -none-    numeric
## offset    1      -none-    logical
## call      5      -none-    call
## nobs      1      -none-    numeric

x2 <- model.matrix(Fertility~., test_data_swiss)[,-1]
predict_model <- predict(model_ridge_regression, s=, newx = x2, type =
"response")

mean((predict_model-test_data_swiss$Fertility)^2)
```

```
## [1] 57.83554
```

```
coef(linear_model_swiss)
```

```
##      (Intercept)      Agriculture      Examination        Education
##      66.16965921      -0.17497395      -0.05176448      -1.06932048
##         Catholic Infant.Mortality
##       0.11713319       1.03247401
```

```
coef(crossvalidation_lasso)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)      60.59242105
## Agriculture       .
## Examination       .
## Education        -0.62205775
## Catholic          0.06463855
## Infant.Mortality  0.69070657
```

As we can see from the above, our Lasso Regularization has shrunk the coefficients in comparison to the linear fit, and two of them have shrunk to zero. This indicates that the Lasso has successfully executed variable selection as well as shrinkage.

Problem 3:

```
library(readxl)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(mgcv)
```

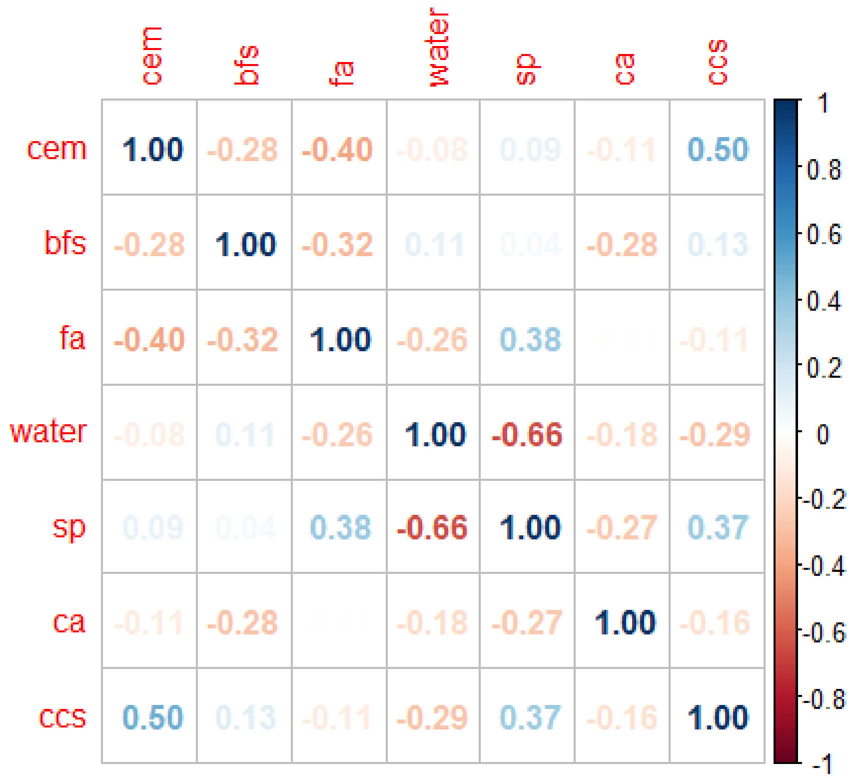
```
## Loading required package: nlme
```

```
## This is mgcv 1.9-0. For overview type 'help("mgcv-package")'.
```

```
concrete_dataset <-
read_excel("C:/Users/Abhiram/Downloads/Concrete_Data.xls")
head(concrete_dataset)
```

```
## # A tibble: 6 × 9
##   Cement (component 1)(kg in a m…¹ Blast Furnace Slag (…² Fly Ash
(component 3…³
##                                <dbl>                  <dbl>
<dbl>
## 1                               540                      0
0
## 2                               540                      0
0
## 3                               332.                   142.
0
## 4                               332.                   142.
```

```
0
## 5                            199.                    132.
0
## 6                            266                     114
0
## # ℹ abbreviated names: ¹`Cement (component 1)(kg in a m^3 mixture)`,
## #   ²`Blast Furnace Slag (component 2)(kg in a m^3 mixture)`,
## #   ³`Fly Ash (component 3)(kg in a m^3 mixture)`
## # ℹ 6 more variables: `Water  (component 4)(kg in a m^3 mixture)` <dbl>,
## #   `Superplasticizer (component 5)(kg in a m^3 mixture)` <dbl>,
## #   `Coarse Aggregate  (component 6)(kg in a m^3 mixture)` <dbl>,
## #   `Fine Aggregate (component 7)(kg in a m^3 mixture)` <dbl>, …

col_names <- c("cem", "bfs", "fa", "water", "sp", "ca", "fa", "age", "ccs")
colnames(concrete_dataset) <- col_names
keeps <- c("cem", "bfs", "fa", "water", "sp", "ca", "ccs")
concrete_dataset <- concrete_dataset[keeps]

summary(concrete_dataset)

##       cem              bfs               fa              water      
##  Min.   :102.0   Min.   :  0.0   Min.   :  0.00   Min.   :121.8  
##  1st Qu.:192.4   1st Qu.:  0.0   1st Qu.:  0.00   1st Qu.:164.9  
##  Median :272.9   Median : 22.0   Median :  0.00   Median :185.0  
##  Mean   :281.2   Mean   : 73.9   Mean   : 54.19   Mean   :181.6  
##  3rd Qu.:350.0   3rd Qu.:142.9   3rd Qu.:118.27   3rd Qu.:192.0  
##  Max.   :540.0   Max.   :359.4   Max.   :200.10   Max.   :247.0  
##        sp               ca              ccs        
##  Min.   : 0.000   Min.   : 801.0   Min.   : 2.332  
##  1st Qu.: 0.000   1st Qu.: 932.0   1st Qu.:23.707  
##  Median : 6.350   Median : 968.0   Median :34.443  
##  Mean   : 6.203   Mean   : 972.9   Mean   :35.818  
##  3rd Qu.:10.160   3rd Qu.:1029.4   3rd Qu.:46.136  
##  Max.   :32.200   Max.   :1145.0   Max.   :82.599  

corrplot(cor(concrete_dataset), method = "number")
```




```
model_gam <- gam(ccs ~ cem + bfs + fa + water + sp + ca, data =
concrete_dataset)
summary(model_gam)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## ccs ~ cem + bfs + fa + water + sp + ca
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.326997  10.510518   0.507 0.612387
## cem          0.108256   0.005214  20.761  < 2e-16 ***
## bfs          0.079357   0.006193  12.814  < 2e-16 ***
## fa           0.055928   0.009287   6.022  2.4e-09 ***
## water       -0.103871   0.027796  -3.737 0.000197 ***
## sp           0.356016   0.110251   3.229 0.001281 **
## ca           0.008027   0.006272   1.280 0.200940
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.445   Deviance explained = 44.9%
## GCV = 155.83  Scale est. = 154.77    n = 1030
```

For CEM and BFS, it seems that we have statistical effects, but not for CAGG, and the corrected R-squared indicates that a significant portion of the variation is present.

Using Smoothing Function:

```
model_gam2 <- gam(ccs ~ s(cem) + s(bfs) +s(fa) + s(water) + s(sp) + s(ca),
data = concrete_dataset)
summary(model_gam2)

##
## Family: gaussian
## Link function: identity
##
## Formula:
## ccs ~ s(cem) + s(bfs) + s(fa) + s(water) + s(sp) + s(ca)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  35.8178     0.3566   100.4   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##            edf Ref.df      F  p-value
## s(cem)   4.464  5.513 69.530  < 2e-16 ***
## s(bfs)   2.088  2.578 48.091  < 2e-16 ***
## s(fa)    5.332  6.404  1.784    0.101
## s(water) 8.567  8.936 13.504  < 2e-16 ***
## s(sp)    7.133  8.143  5.498 1.22e-06 ***
## s(ca)    1.000  1.000  0.018    0.892
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.531   Deviance explained = 54.4%
## GCV = 134.84  Scale est. = 130.96    n = 1030
```

It is also notable that this model, with an adjusted R-squared of.531, explains a large portion of the variance in CCS. In summary, it appears that CCS and CEM are related.

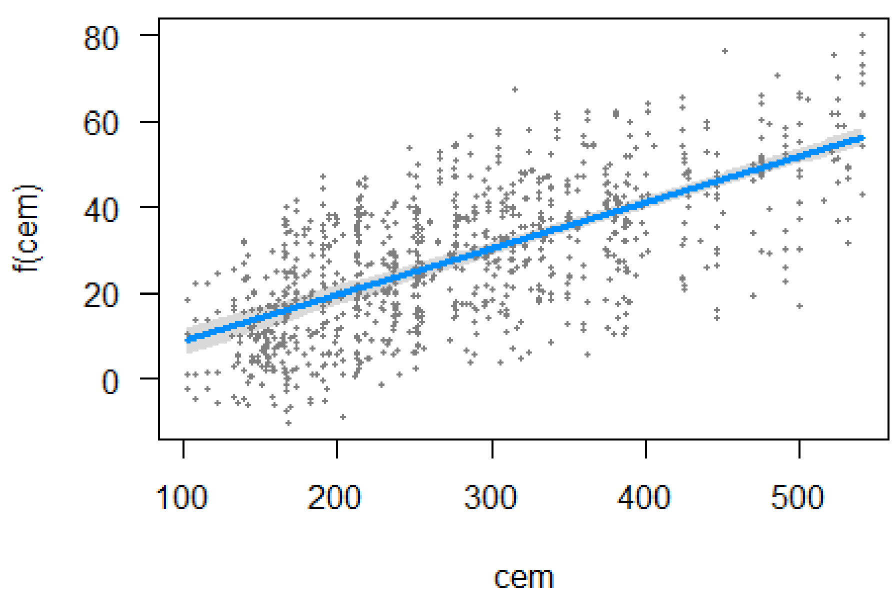
```
model_gam.SSE <- sum(fitted(model_gam)-concrete_dataset$ccs)^2
model_gam.SSR <- sum (fitted(model_gam)-mean(concrete_dataset$ccs))^2
model_gam.SST <- model_gam.SSE + model_gam.SSR

sm_RSQUARE <- 1-(model_gam.SSE/model_gam.SST)
sm_RSQUARE

## [1] 0.4968293

model_gam2.SSE <- sum(fitted(model_gam2)-concrete_dataset$ccs)^2
model_gam2.SSR <- sum(fitted(model_gam2)-mean(concrete_dataset$ccs))^2
model_gam2.SST <- model_gam2.SSE+model_gam2.SSR
```

```r
sm_RSQUARE = 1-(model_gam2.SSE/model_gam2.SST)
sm_RSQUARE

## [1] 0.4999546

anova(model_gam, model_gam2, test = "Chisq")

## Analysis of Deviance Table
##
## Model 1: ccs ~ cem + bfs + fa + water + sp + ca
## Model 2: ccs ~ s(cem) + s(bfs) + s(fa) + s(water) + s(sp) + s(ca)
##   Resid. Df Resid. Dev     Df Deviance  Pr(>Chi)
## 1   1023.00     158334
## 2    996.43     131019 26.574    27315 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

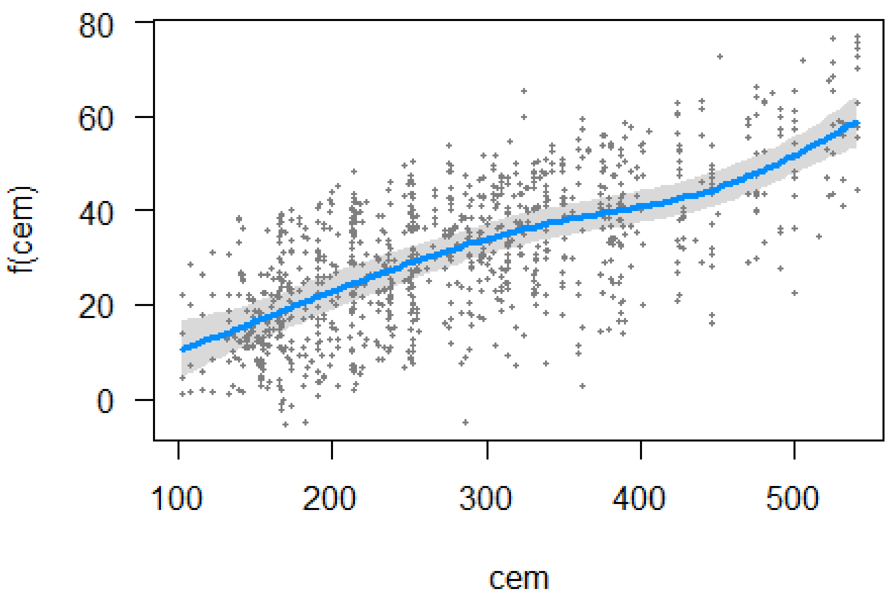
The inclusion of nonlinear correlations among the covariates appears to enhance the model, however this was not something we could have assumed previously based on additional statistical data.

Visualizing with visreg library

```r
library(visreg)
visreg(model_gam, 'cem')
```

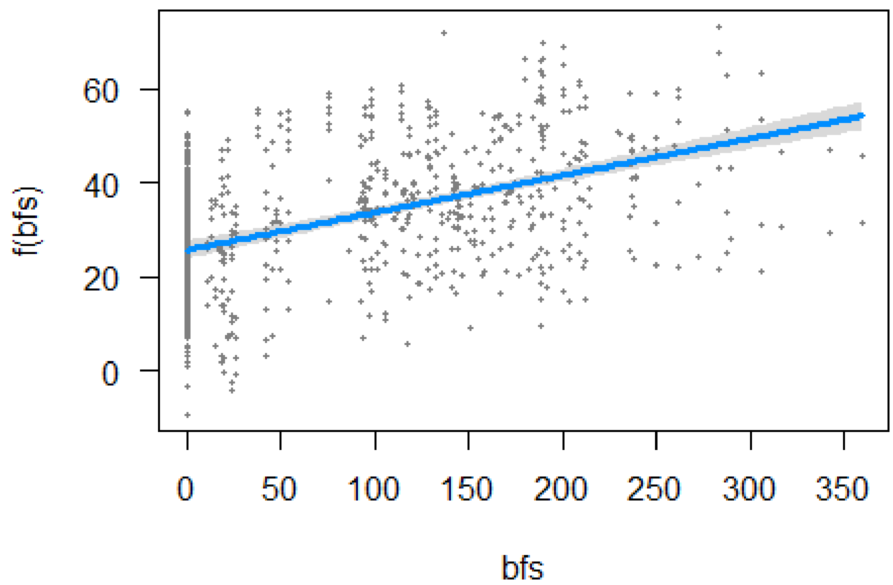



```r
visreg(model_gam2, 'cem')
```

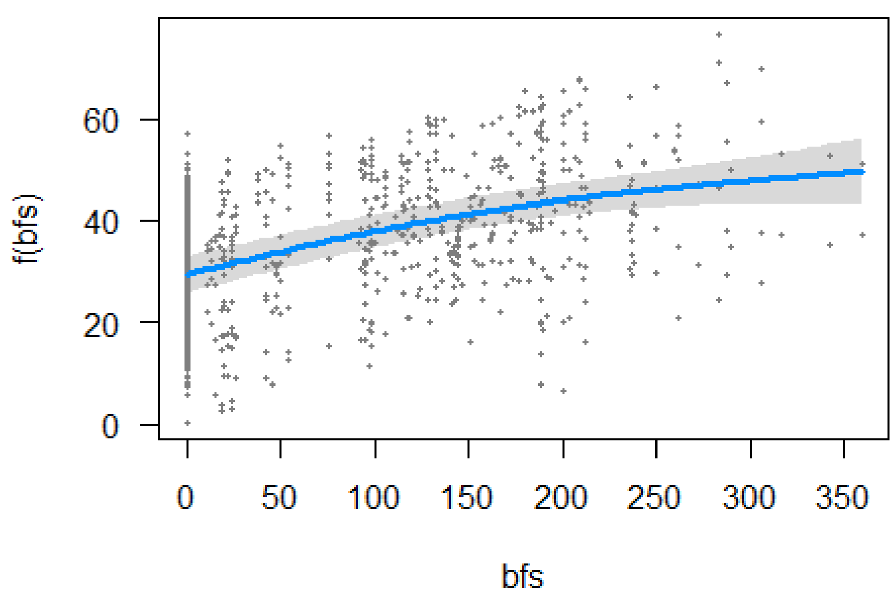



With all other model variables held constant, the outcome is a plot showing the expected value of the CCS changing as a function of x (CEM). The expected value (blue line), the confidence interval for the expected value (grey band), and the partial residuals (dark grey dots) are all included. The model is improved by analyzing the covariates' nonlinear relationships.

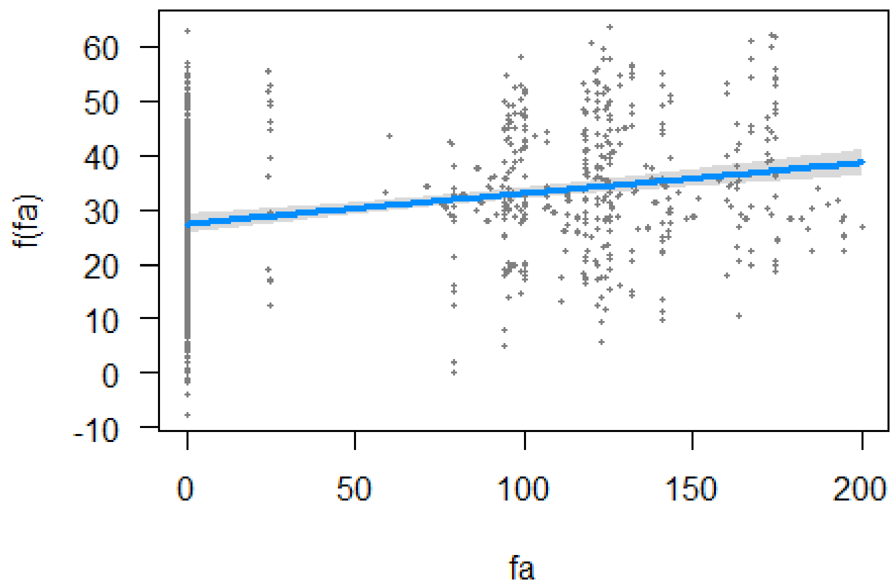
```
visreg(model_gam, 'bfs')
```



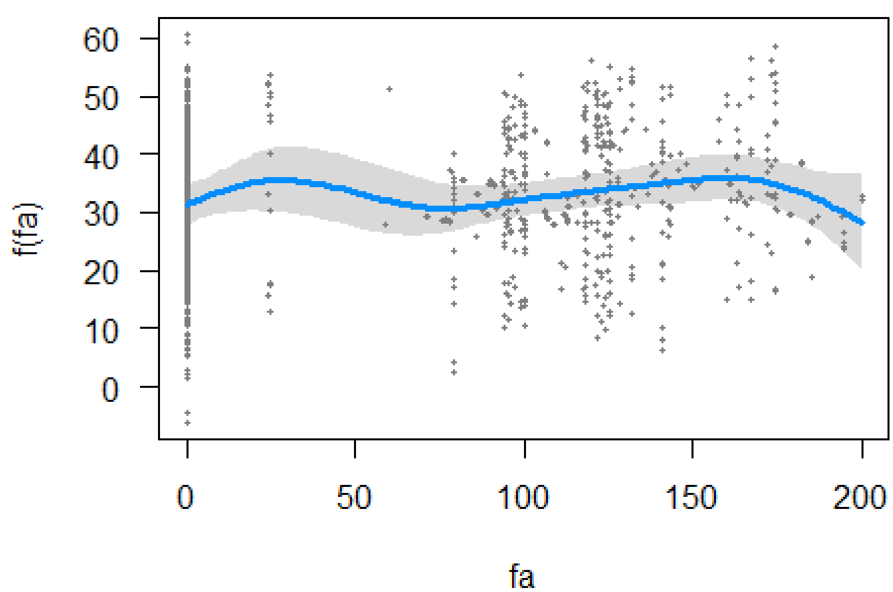

```r
visreg(model_gam2, 'bfs')
```



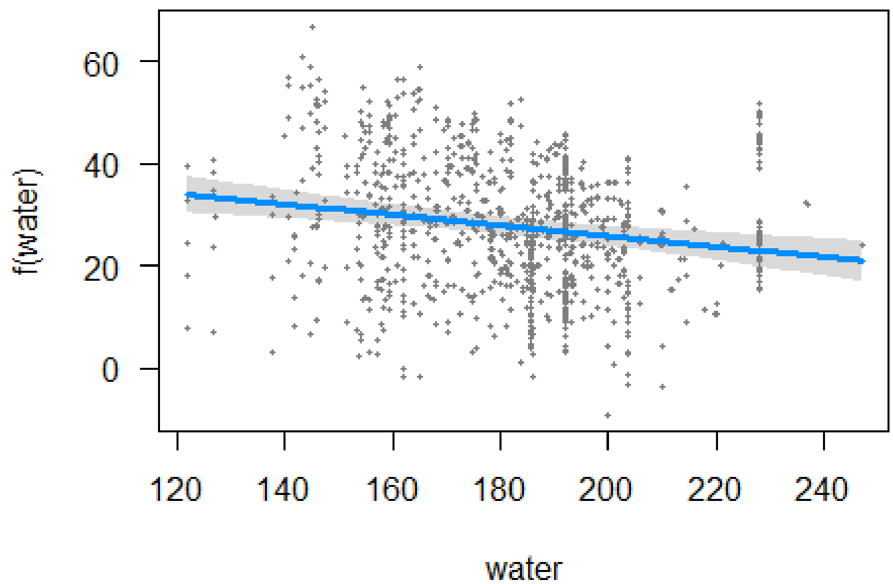

```r
visreg(model_gam, 'fa')
```



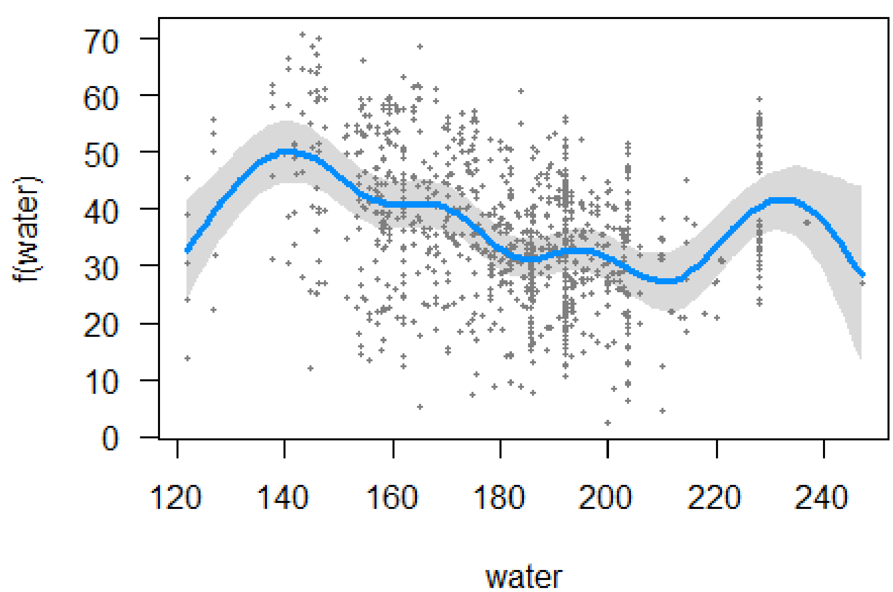

```r
visreg(model_gam2, 'fa')
```



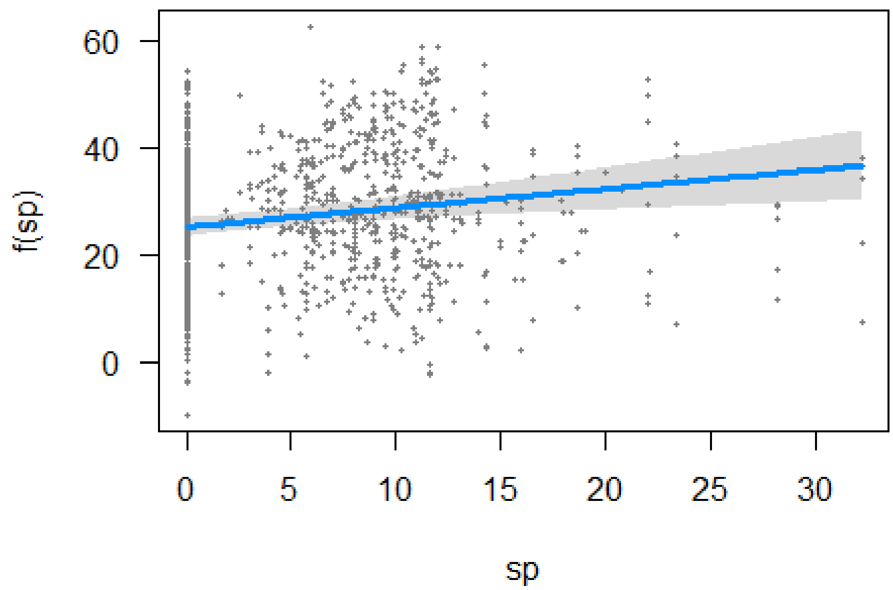

```r
visreg(model_gam, 'water')
```



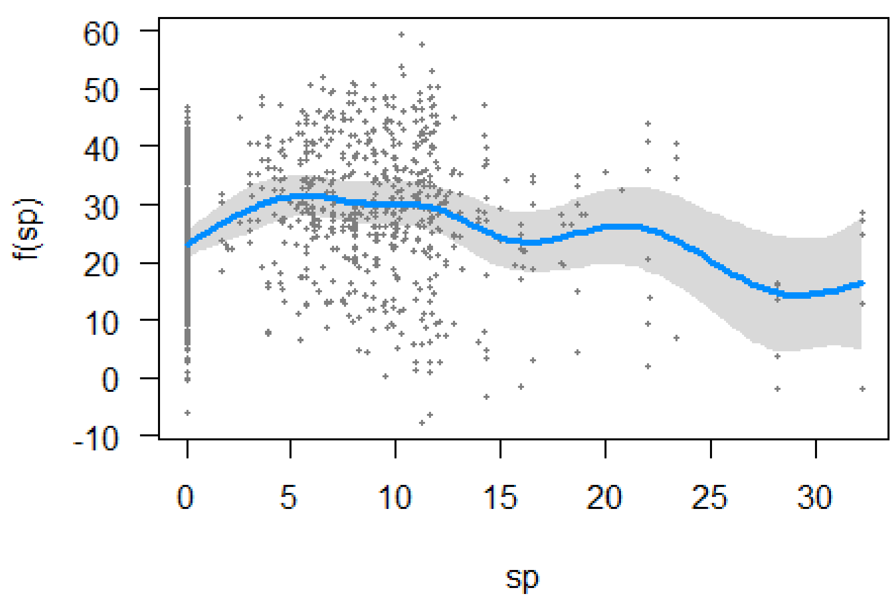

```r
visreg(model_gam2, 'water')
```



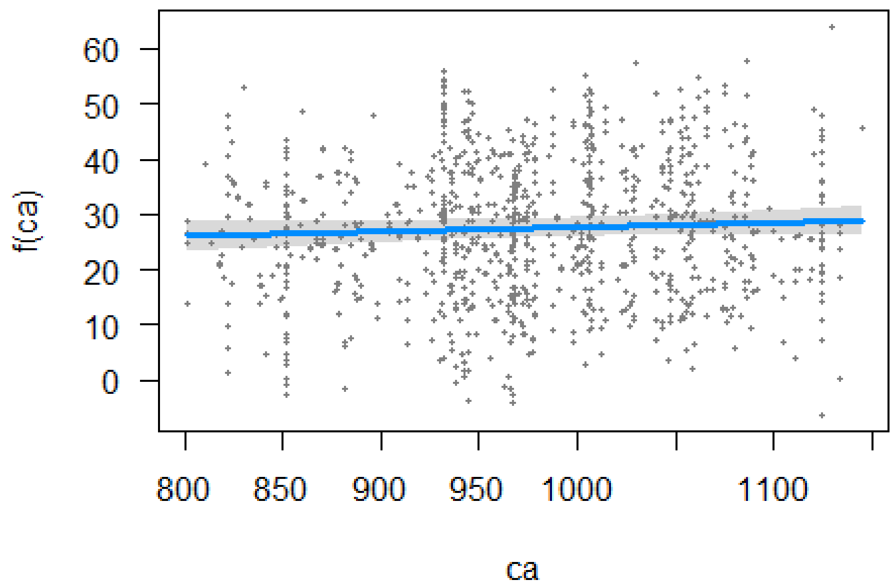

```r
visreg(model_gam, 'sp')
```



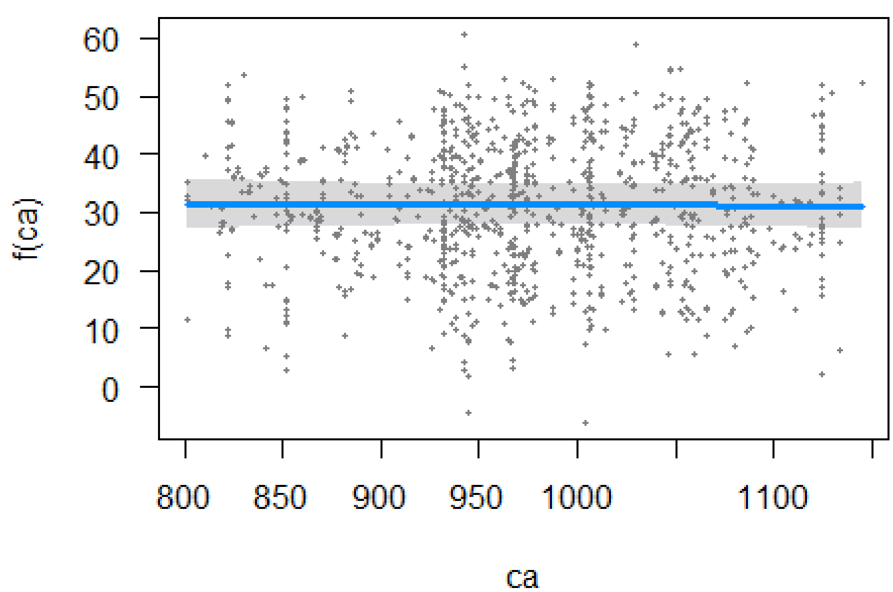

```
visreg(model_gam2, 'sp')
```




```
visreg(model_gam, 'ca')
```




```
visreg(model_gam2, 'ca')
```



The CEM graph indicates that the confidence interval has increased in value following the application of the smoothing function when compared to the model prior to the smoothing function.