Assignment-03

Sumanth Donthula

2023-03-03

```
library(tinytex)
```

Recitation Exercises

chapter - 6

1.) a.) Understanding how Best Subset and other step-wise selections are performed allows us to intuitively realize that the best subset pick will have less training RSS than the other two.

The reason for this is that the Best Subset Selection approach creates 2k models with each predictor and each subset of predictors. At each phase, it would choose the best model with the least amount of training RSS.

Once we follow a step-wise (forward or backward) order (and observe just 1+ k(k+1)/2 models), we will not evaluate additional models that may provide less training error.

b.)

It's hard to determine which of the following models gives the lowest test RSS because all the above techniques employ algorithms based on reducing training RSS to select the best models when comparing models of the same size. We next utilize cross validation error/Mallow's Cp/AIC/BIC, and so on, to choose the final model among k+1 models. Nevertheless, the initial decision is made with the least training error in mind, which does not ensure the lowest test error.

c.)

i.) TRUE: The k-variable model has k variables (predictors). Therefore a k variable model identified by forward step-wise selection will undoubtedly be a subset of a (k+1) variable model identified by forward step-wise selection since it simply requires adding one more variable and employing forward step-wise selection again.

ii.) TRUE: The same explanation as previously applies here as well. The k variable model is already included in the collection of models evaluated backwards and forwards. By adding one more variable and using the same procedure, predictors in the k variable model become a subset of predictors in the (k+1) model.

iii.) FALSE: This is occasionally true, but not always. Backward step-by-step examination of k predictors and formation of model, then removal of one least relevant predictor and formation of model with k-1 predictors, and so on. In the forward direction, we consider a model that starts with one predictor and subsequently increases one predictor at a time. As a result, we cannot be positive that the predictors in the k= variable model identified via backward step-wise selection are a subset of the predictors in the k+1 variable model.

iv.) FALSE: The same explanation (as given in (iii)) may be used to understand why it is untrue.

v.) FALSE : The predictors found by best subset selection in the k-variable model are a subset of the predictors identified by best subset selection in the (k + 1)-variable model.

Because the best subset selects the best model for each k, the preceding assumption is false.

2.) a.) (iii) option

We know that when the lambda value grows, several of the predictor coefficients in Lasso become absolutely zero, reducing the model's flexibility (bias variance trade off actually take place – meaning it brings substantial decrease in variance though there is a slight increase in bias of the model).

b.) (iii) Option

The same idea that we used for Lasso before applies here. The ridge penalty makes the model less flexible but significantly reduces variation. One downside of ridge regression is that it uses all predictors and the values of predictor coefficients reduce but do not reach zero as in Lasso.

c.) (ii) option

We know that non-linear models are more flexible, with higher variance and lower bias. When we consider the estimated MSE equation, which comprises the elements square of bias, variance, and irreducible error, we may intuitively conclude that the prediction accuracy of a non-linear model will be high when the decrease in bias is greater than the increase in variance.

3.) a.) (iv) option

This is easily described by the contour plots of the coefficients (of predictors) and the value of s. (square for Lasso). As we do the lasso, we are attempting to discover the set of coefficient estimates that result in the minimum RSS, subject to the constraint that there is a maximum size s. If s is big enough, the coefficients of Lasso will be similar to those of Least squares. The oval shapes in the graph represent the same RSS at any position along the curve.

In addition, the outer ellipse has more RSS than the inner ellipse. Therefore, when s, the constraint, climbs from 0 to some positive values, the square size increases and it moves towards the inner ellipses, implying that as s increases, the value of training RSS decreases and so the answer is (iv)

b.) (ii) option

Except for the intercept, the only possible coefficient values for s=0 are zeroes. That means that at s=0, we have a null model that is independent of all predictors. The model's flexibility grows as the value of s increases and approaches towards least square estimates. We know from our basics that when model flexibility rises, bias lowers, variance increases, and test MSE drops to a certain amount before increasing at a certain point. As a result, we picked (ii) as the best alternative.

c.) (iii)Steadily increase

The approach used to explain question (b) may also be applied here. When s grows, the model's flexibility improves, and therefore the value of variance climbs steadily until s reaches a value where the coefficients match the least square estimate.

d.) (iv) Steadily decrease

The same idea as above may be applied here. When s grows from 0, the model's flexibility improves, and so the bias decreases continuously until the least square estimate meets the limitations of s.

e.) (v) Remains constant

The irreducible error occurs regardless of how good the model is, and it is caused by noise in the system (for not considering unknown element). Hence we don't notice a commensurate increase/decrease in irreducible error and thus the solution dependent on the model's flexibility or coefficient values.

4.) a.) (iii)Steadily increase

It is worth noting that when $\lambda = 0$, the above equation equals LSE (Least Squares Estimate). The model developed using least squares coefficients yields the smallest training RSS. The level curves of coefficients shift away from the LSE as the value of λ grows. As a result, the training error continues to rise.

b.) (ii) Decrease at first, then gradually increase in a U shape. To mitigate the over-fitting problem that LSE generally produces, we add penalty to the Least Square estimate (referred to as ridge or Lasso depending on the penalty selected). When λ rises, the penalty increases, and the model built in this manner tends to under-fit the model and improve on reducing test error. Nevertheless, this cannot continue indefinitely, and after a certain amount, the rise in bias surpasses the decrease in variance, resulting in an increase in test error.

c.) (iv)Steadily decrease

When the λ value grows, the coefficient value shrinks towards zero, making the model less flexible than the initial model. We employ shrinkage ideas to reduce variation as much as feasible (till we get almost zero variance). When the coefficients approach 0, the model approaches zero variance.

d.) (iii) Steadily increase

The same intuition that was employed in the last response may be applied here. The model becomes less flexible as λ grows, and variation tends to diminish. When the value of λ increases, the bias tends to rise as the coefficients go to zero.

e.) (v) Remains constant

Irreducible error is unaffected by the coefficients or penalty amount employed in the preceding equation. It arises as a result of system noise and is hence unrelated to the value of λ. As a result, the solution.

5.) Hand written and attached separately.

Chapter - 7

2.) a.)

The idea that $\int [g^{(m)}(x)]^2 dx$ is a summation of $[g^{(m)}(x)]^2$ throughout the range of x, thus ĝ will be the g that minimizes We can understand why large λ values drive the penalty term $\lambda \int [g^{(m)}(x)]^2 dx$ towards zero. At the opposite extreme, a $\lambda = 0$ eliminates this factor altogether from the equation, leaving us free to select any g that minimizes the loss function $\sum n_i = 1(y_i - g(x_i))^2$

b.) In this case ,as $\lambda = \infty$, $g^{!}(x)$ tends to zero and it is possible when g(x)is some constant c. Therefore $\hat{g}(x) = c$ (some constant).(So g(x) will be a straight line drawn parallel to X-axis) The value of constant c, which minimizes RSS is $c = \frac{1}{n} \sum_{i=1}^{n} y_i$

c.) As $\lambda = \infty$, it forces the 2nd derivative of g(x) to zero, i.e., $g^{!!}(x) = 0$. This is feasible if g(x) is a linear equation of the type g(x) = ax+b, and this is the line obtained by least squares since it has the smallest RSS (albeit for all linear equations, the second derivative is zero).

d.)

Using the same idea as before, we may assert that $\lambda = \infty$, $g^{!!!}(x) = 0$ and hence $\hat{g}(x)$ will be a quadratic of the form ax2+bx+c obtained from least squares (as it will have minimum RSS)

e.) From the equation $\hat{g}$ given in the question, We can see that as $\lambda = 0$, the penalty term becomes zero and just the loss term remains. Therefore, in order to reduce RSS to zero, g(x) can adopt any shape that passes through all points in the training data and can be too flexible or over-fitting.

3.) Let $Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \varepsilon$ be Equation 1

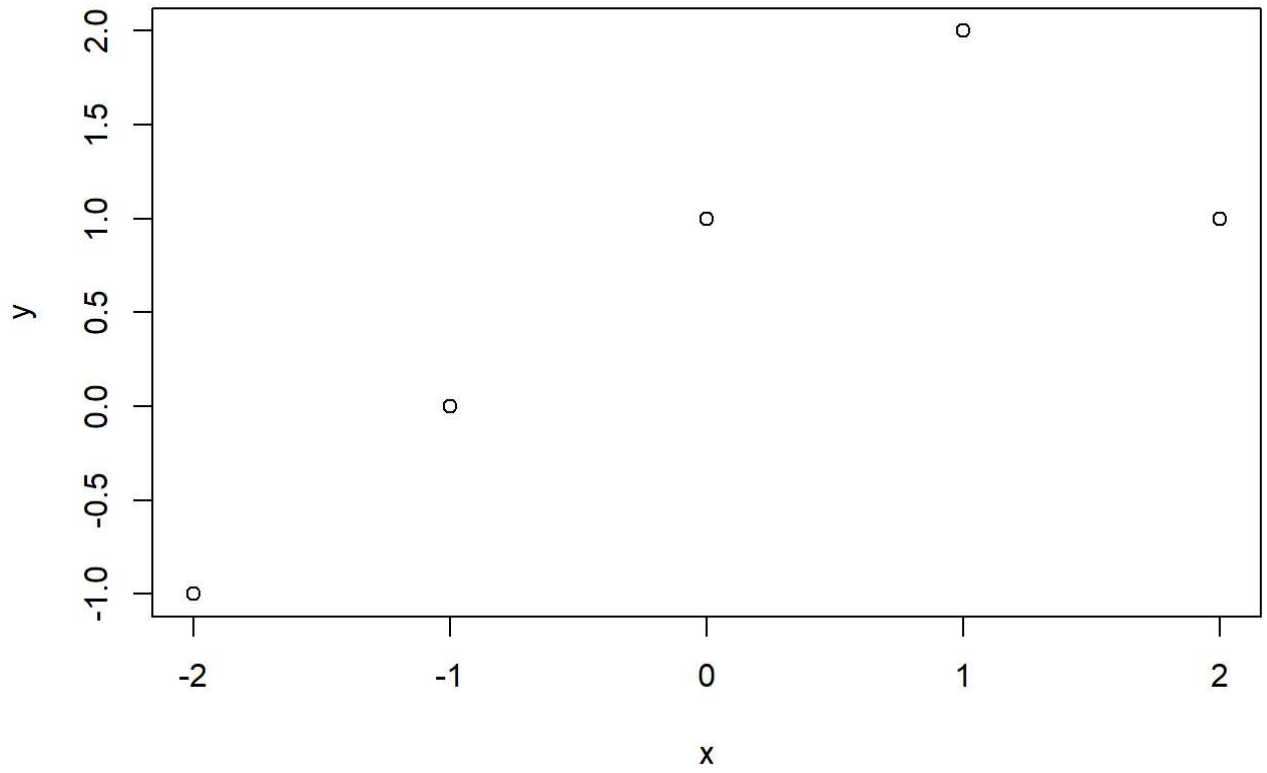
Let us find the equations when X<1 and X>=1 separately. We get two different functions.

Applying all the given data into Equation 1 when When X<1: $\hat{y} = 1 + X$ -> Let this be equation 2

When X>=1: $\hat{y} = 1 + X - 2(X-1)^2$ =>$\hat{Y} = 1 + X - 2X^2 + 4X - 2$ =>$\hat{y} = -2X^2 + 5X - 1$ -> let this be ation 3

Therefore the curve will be a straight line from X=-2 to X=1 and then a quadratic curve from X=1 to X=2 in the region between X=-2 and X=2. The critical point is obtained by calculating the first order derivative of Eq3 and equating the result to zero.

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



4.) Let us substitute all of the supplied values for the regions in the above equation of y.

-2 < X < 0: $\hat{y} = 1$;

0 ≤ X < 1: $\hat{y} = 2$

1 ≤ X ≤ 2: $\hat{y} = 2 - X + 1 = 3 - X$

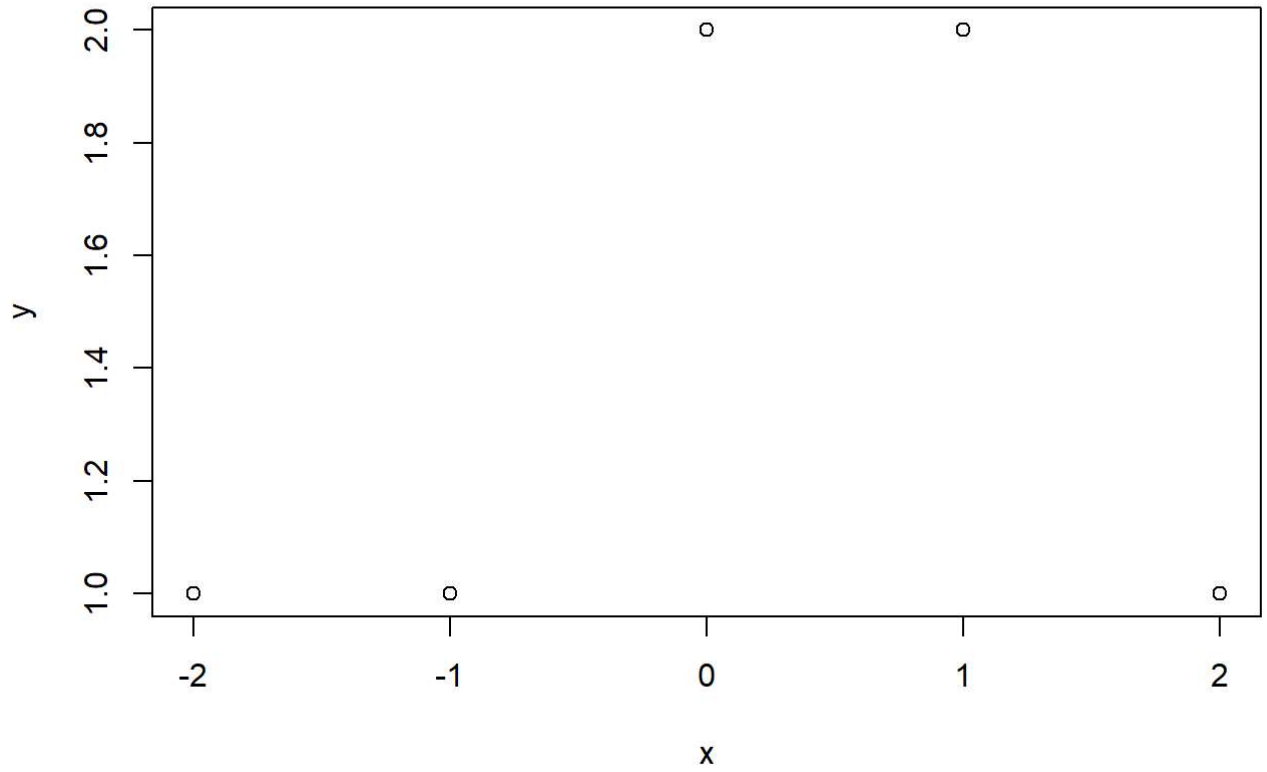
The asked is just for the region between X=-2 and X=2, therefore we need not check what the equation would be for X>2. So,

for -2≤ X < 0, the slope is zero and intercept is 1

for 0 ≤ X < 1, the slope is zero and intercept is 2

For 1 ≤ X ≤ 2, the slope is -1 and intercept is 3

```
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 above equations clearly show that when the value of λ approaches infinity, the penalty term becomes increasingly relevant.

When $\lambda \rightarrow \infty$, for $\hat{g}_1$, $\hat{g}_3$ (x) $\rightarrow$ 0, This indicates that the highest order polynomial that meets this condition will be of the form g(x) = ax2 + bx + c. (because the 3rd order derivative is zero). As a result, g^1 will be a quadratic that reduces training RSS.

When $\lambda \to \infty$, for $\hat{g}_2$, $\hat{g}_4$ (x) → 0, This indicates that the highest order polynomial that meets this condition will be of the type g(x) = ax3 + bx2 + cx + d (because the 4th order derivative is zero). As a result, g^2 will be a cubic that reduces training RSS.

As $\hat{g}_2$ is more flexible compared to $\hat{g}_1$, $\hat{g}_2$ will have less RSS compared to $\hat{g}_1$ for a given high value of RSS.

b.) We cant say which of the above have smaller test RSS. It solely depends on the true relationship of the predictors and the order of such relationship. Based on the true relation, $\hat{g}_1$ can be under-fit and $\hat{g}_2$ can be over-fit. So we can say for sure if $\hat{g}_1$ or $\hat{g}_2$ have the small test RSS.

c.) $\lambda = 0$ implies the penalty term totally becomes zero. And that means both $\hat{g}_1$ and $\hat{g}_2$ would have the same training RSS (of zero if all the xi are unique). We may just have any function that interpolates all of the training observations with no limitations on g.

In terms of test RSS, we cannot be certain of having a low test RSS since a model that covers all training points and has a training RSS of zero would be horribly over-fit and have a high test RSS. Assume that the same interpolating function was used for both $\hat{g}_1$ & $\hat{g}_1$ (e.g. Both would have the same test RSS if they were a linear spline with knots at each unique xi).

Practicum Problems

Problem 1

Loading the mtcars data frame

```
##       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
```

performing test/train split

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
set.seed(0)

trainIndex = createDataPartition(mtcars$am,times=1,p=0.8,list = F)
trainset = mtcars[trainIndex,]
testset = mtcars[-trainIndex,]
```

fitting a linear model and displaying coef values, mean squared error of Linear Model

wt is a predictor based on the statistics as it has the lowest p-value as per the resulting t-statistic above.

```
linearModel = lm(mpg~.,data=trainset)

mean((predict(linearModel,testset)-testset$mpg)^2)
```

```
## [1] 20.01153
```

```
summary(linearModel)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = trainset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8257 -0.8323  0.0080  1.0310  3.3446
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 27.975931  18.508625   1.512   0.1514
## cyl         -0.463054   1.087553  -0.426   0.6763
## disp         0.007147   0.016278   0.439   0.6669
## hp          -0.015572   0.023256  -0.670   0.5133
## drat         0.043647   1.958208   0.022   0.9825
## wt          -3.841354   1.998665  -1.922   0.0738 .
## qsec         0.514469   0.716008   0.719   0.4835
## vs           0.724096   1.988433   0.364   0.7208
## am           1.903646   2.082678   0.914   0.3752
## gear        -0.046622   1.463656  -0.032   0.9750
## carb        -0.788125   0.945297  -0.834   0.4175
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.338 on 15 degrees of freedom
## Multiple R-squared:  0.9223, Adjusted R-squared:  0.8704
## F-statistic:  17.8 on 10 and 15 DF,  p-value: 1.464e-06
```

```
coef(linearModel)
```

```
##  (Intercept)          cyl         disp           hp         drat           wt
## 27.975931216 -0.463053862  0.007147382 -0.015572021  0.043646659 -3.841354354
##         qsec           vs           am         gear         carb
##  0.514468609  0.724096147  1.903646030 -0.046621655 -0.788124585
```

performing ridge regression initializing lambda and plotting MSE vs λ

```
x = model.matrix(mpg~.,trainset)[,-1]

y = trainset$mpg

library(glmnet)
```

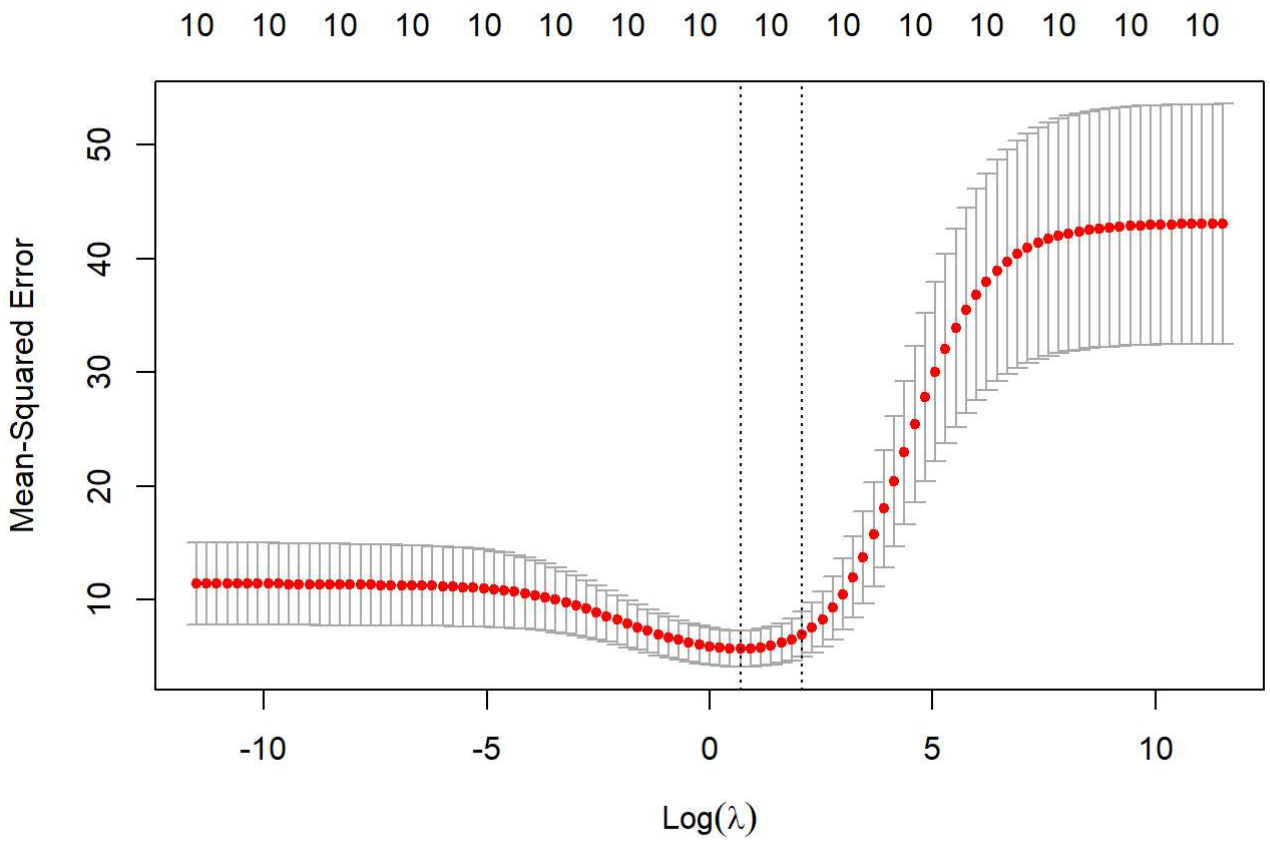
```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-6
```

```
lambdaSeq=10^seq(5,-5,by = -.1)


ridgeCrossVal = cv.glmnet(x, y, alpha = 0,lambda = lambdaSeq)
```

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

```
plot(ridgeCrossVal)
```



finding optimal lambda

```
#Finding the best lambda value:
lambdaOptimal = ridgeCrossVal$lambda.min
print(lambdaOptimal)
```

```
## [1] 1.995262
```

Building a Ridge Regression Model using GLMNET

```
ridgeModel = glmnet(x, y, alpha = 0, lambda  = lambdaOptimal)
```

Model Summary

```
summary(ridgeModel)
```

```
##           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
```

displaying coef values

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

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) 26.924970803
## cyl         -0.377105798
## disp        -0.005679121
## hp          -0.013252204
## drat         0.707244544
## wt          -1.526324786
## qsec         0.120147827
## vs           0.931779218
## am           1.659858645
## gear         0.372245497
## carb        -1.204900675
```

```
X =  model.matrix(mpg~.,testset)[,-1]
modelPredict = predict(ridgeModel,s =,newx = X, type = "response")

mean((modelPredict-testset$mpg)^2)
```

```
## [1] 15.83657
```

We can observe that doing Ridge Regression reduces MSE on test data from 10.71 to 1.18. The shift in coefficients is visible after doing Ridge Regression. The coefficients have decreased and become closer to zero, but none are precisely zero. As a result, we may argue that Ridge regression conducted shrinkage but not variable selection.

Problem 2:

Loading libraries and data set

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

swissData = data.frame(swiss)
```

splitting data into test and train set

```
set.seed(150)
trainIndex = createDataPartition(swissData$Fertility,p=0.8,list = F)
trainset = swissData[trainIndex,]
testset = swissData[-trainIndex,]
```

fitting a linear model

```
linearModel = lm(Fertility~.,trainset)

summary(linearModel)
```

```
##
## Call:
## lm(formula = Fertility ~ ., data = trainset)
##
## 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 feature with coefficients as -0.17497, -0.05176,0.11713, 1.03247

calculating test mse

```
mean((testset$Fertility-predict(linearModel,testset))^2)
```

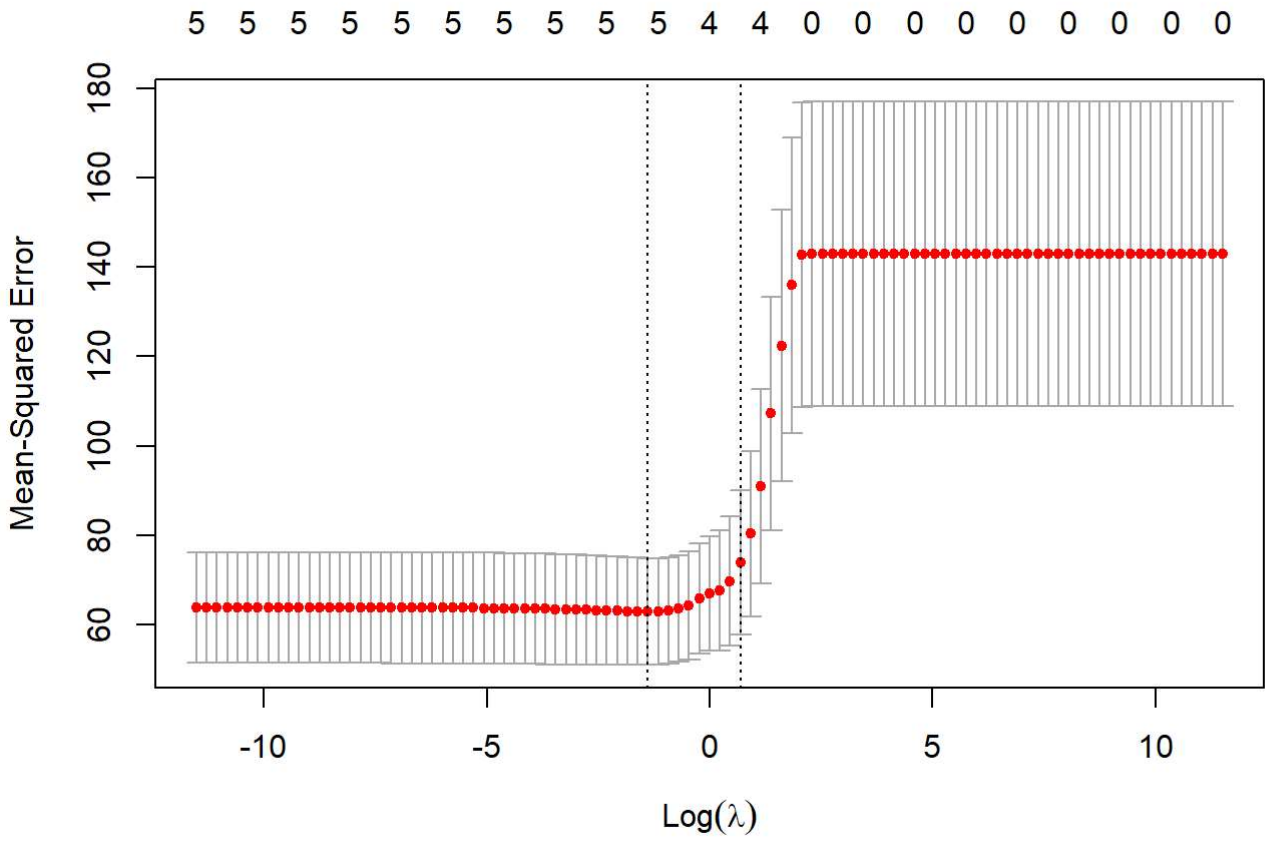
```
## [1] 59.91027
```

performing Lasso Regression

```
library(Matrix)
library(foreach)
library(glmnet)

x = model.matrix(Fertility~.,trainset)[,-1]

y = trainset$Fertility
```

Cross Validation Lasso GLMNET

```
# Setting the range of lambda values
lambdaSeq = 10^seq(5,-5,by = -.1)

# Using cross validation glmnet
lassoCrossVal = cv.glmnet(x, y, alpha = 1,lambda = lambdaSeq)
plot(lassoCrossVal)
```



finding optimal lambda value

```
lambdaOptimal = lassoCrossVal$lambda.min
lambdaOptimal
```

```
## [1] 0.2511886
```

Using glmnet function to build the ridge regression model

```
fit = glmnet(x, y, alpha = 1, lambda  = lambdaOptimal)

summary(fit)
```

```
##           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
```

```
X =  model.matrix(Fertility~.,testset)[,-1]
modelPredict = predict(fit,newx = X, type = "response")
```

MSE on test data

```
mean((modelPredict-testset$Fertility)^2)
```

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

comparing coesficients of Linear model and Lasso models

```
coef(linearModel)
```

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

```
coef(lassoCrossVal)
```

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

We can clearly see that, when compared to the linear fit, our Lasso Regularization has shrunk the coefficients, and we can also see that two of them have shrunk to zero. We may conclude that the Lasso obviously conducted shrinkage and variable selection.

Problem 3:

```
library(readxl)
ConcreteData = read_excel("C:/Users/SRINU/Desktop/Spring 23/DPA/Concrete_Data.xls")

summary(ConcreteData)
```

```
##  Cement (component 1)(kg in a m^3 mixture)
##  Min.   :102.0
##  1st Qu.:192.4
##  Median :272.9
##  Mean   :281.2
##  3rd Qu.:350.0
##  Max.   :540.0
##  Blast Furnace Slag (component 2)(kg in a m^3 mixture)
##  Min.   :  0.0
##  1st Qu.:  0.0
##  Median : 22.0
##  Mean   : 73.9
##  3rd Qu.:142.9
##  Max.   :359.4
##  Fly Ash (component 3)(kg in a m^3 mixture)
##  Min.   :  0.00
##  1st Qu.:  0.00
##  Median :  0.00
##  Mean   : 54.19
##  3rd Qu.:118.27
##  Max.   :200.10
##  Water  (component 4)(kg in a m^3 mixture)
##  Min.   :121.8
##  1st Qu.:164.9
##  Median :185.0
##  Mean   :181.6
##  3rd Qu.:192.0
##  Max.   :247.0
##  Superplasticizer (component 5)(kg in a m^3 mixture)
##  Min.   : 0.000
##  1st Qu.: 0.000
##  Median : 6.350
##  Mean   : 6.203
##  3rd Qu.:10.160
##  Max.   :32.200
##  Coarse Aggregate  (component 6)(kg in a m^3 mixture)
##  Min.   : 801.0
##  1st Qu.: 932.0
##  Median : 968.0
```

```
##  Mean   : 972.9
##  3rd Qu.:1029.4
##  Max.   :1145.0
##  Fine Aggregate (component 7)(kg in a m^3 mixture)   Age (day)
##  Min.   :594.0                                      Min.   :  1.00
##  1st Qu.:731.0                                      1st Qu.:  7.00
##  Median :779.5                                      Median : 28.00
##  Mean   :773.6                                      Mean   : 45.66
##  3rd Qu.:824.0                                      3rd Qu.: 56.00
##  Max.   :992.6                                      Max.   :365.00
##  Concrete compressive strength(MPa, megapascals)
##  Min.   : 2.332
##  1st Qu.:23.707
##  Median :34.443
##  Mean   :35.818
##  3rd Qu.:46.136
##  Max.   :82.599
```

Changing the Column names Taking Columns first 6 columns

```
colnames(ConcreteData) = c("cem", "bfs", "fa", "water", "sp", "cagg", "fagg", "age", "ccs")
keep = c("cem", "bfs", "fa", "water", "sp", "cagg", "ccs")
ConcreteData = ConcreteData[keep]
summary(ConcreteData)
```

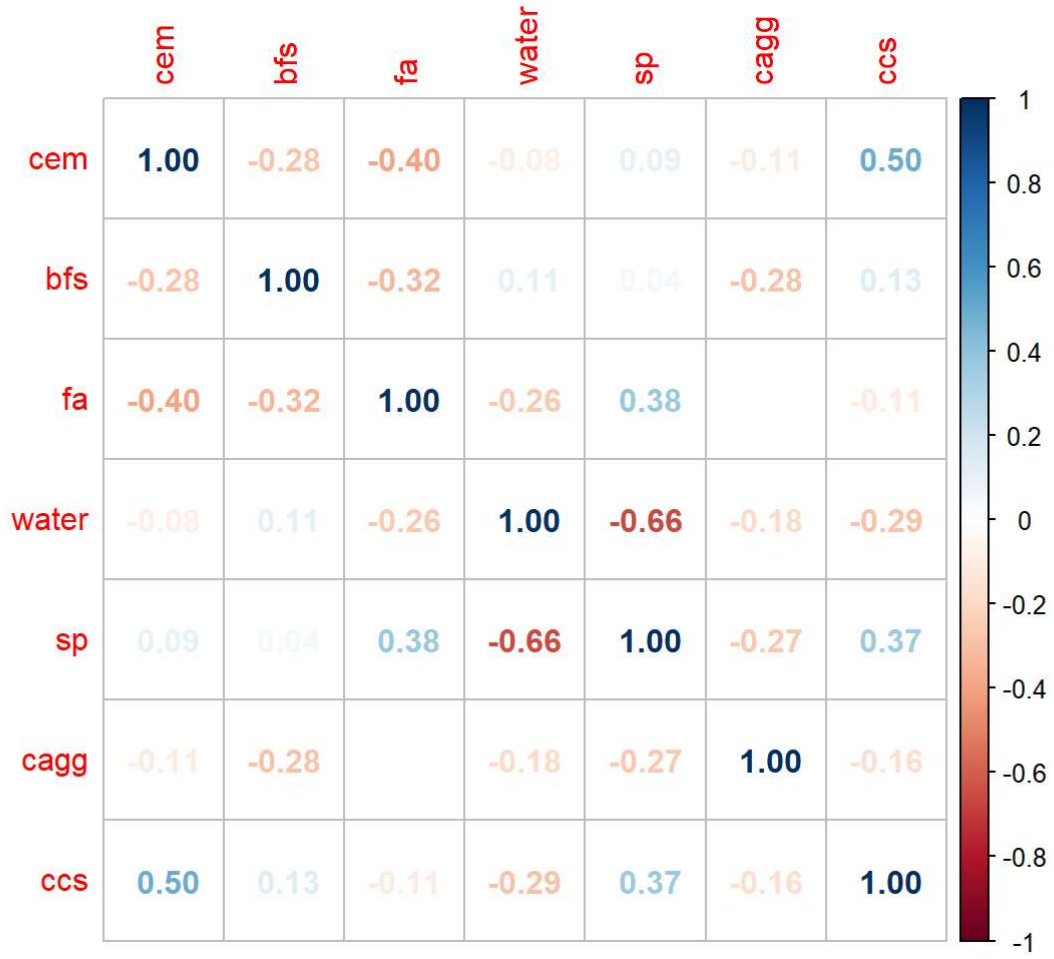
```
##       cem             bfs              fa             water      
##  Min.   :102.0   Min.   :  0.0   Min.   :  0.00   Min.   :121.8  
##  1st Qu.:192.4   1st Qu.:  0.0   1st Qu.:  0.00   1st Qu.:164.9  
##  Median :272.9   Median : 22.0   Median :  0.00   Median :185.0  
##  Mean   :281.2   Mean   : 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              cagg             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
```

plotting correlation

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

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



fitting GAM model

```
library(mgcv)
```

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

```
## This is mgcv 1.8-41. For overview type 'help("mgcv-package")'.
```

```
gamModel = gam(ccs ~ cem + bfs + fa + water + sp + cagg , data=ConcreteData)
summary(gamModel)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## ccs ~ cem + bfs + fa + water + sp + cagg
##
## 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 **
## cagg         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
```

It appear to have statistical effects for CEM and BFS, but not for CAGG, and the corrected R-squared shows that a significant percentage of the variation is explained.

Using Smoothing Function

```
gamModel2 = gam(ccs ~ s(cem) + s(bfs) + s(fa) + s(water) + s(sp) + s(cagg) , data=ConcreteData)
summary(gamModel2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## ccs ~ s(cem) + s(bfs) + s(fa) + s(water) + s(sp) + s(cagg)
##
## 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(cagg)  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
```

We can also see that this model accounts for most of the variance in CCS , with an adjusted R-squared of .531 . So, it looks like the CEM is associated with CCS.

```
gamModel1.sse = sum(fitted(gamModel)-ConcreteData$ccs)^2
gamModel1.ssr = sum(fitted(gamModel) - mean(ConcreteData$ccs))^2
gamModel1.sst = gamModel1.sse + gamModel1.ssr

rsqr_main=1-(gamModel1.sse/gamModel1.sst)
print(rsqr_main)
```

```
## [1] 0.4967177
```

```
gamModel2.sse = sum(fitted(gamModel2)-ConcreteData$ccs)^2
gamModel2.ssr = sum(fitted(gamModel2) -mean(ConcreteData$ccs))^2
gamModel2.sst = gamModel2.sse + gamModel2.ssr

rsqr_sm=1-(gamModel2.sse/gamModel2.sst)
print(rsqr_sm)
```

```
## [1] 0.5000744
```

Comparison of Model

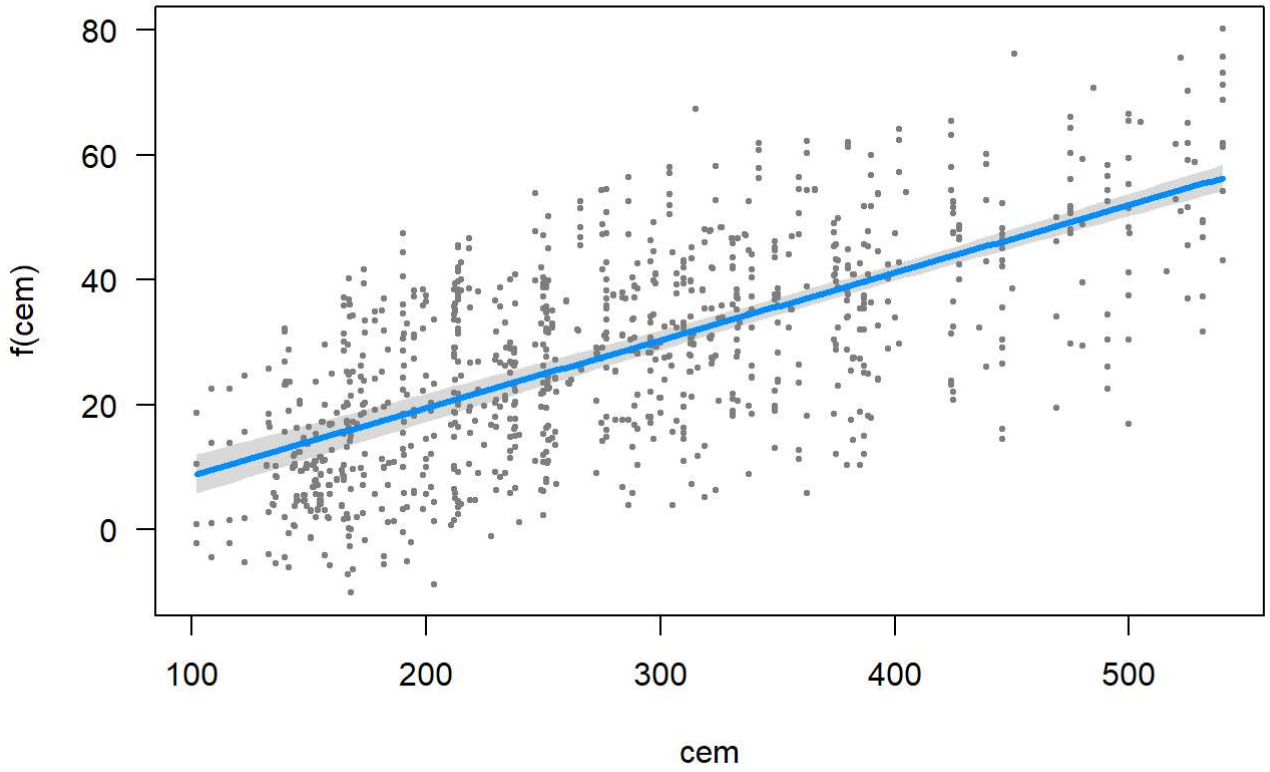
```
anova(gamModel, gamModel2, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: ccs ~ cem + bfs + fa + water + sp + cagg
## Model 2: ccs ~ s(cem) + s(bfs) + s(fa) + s(water) + s(sp) + s(cagg)
##   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
```

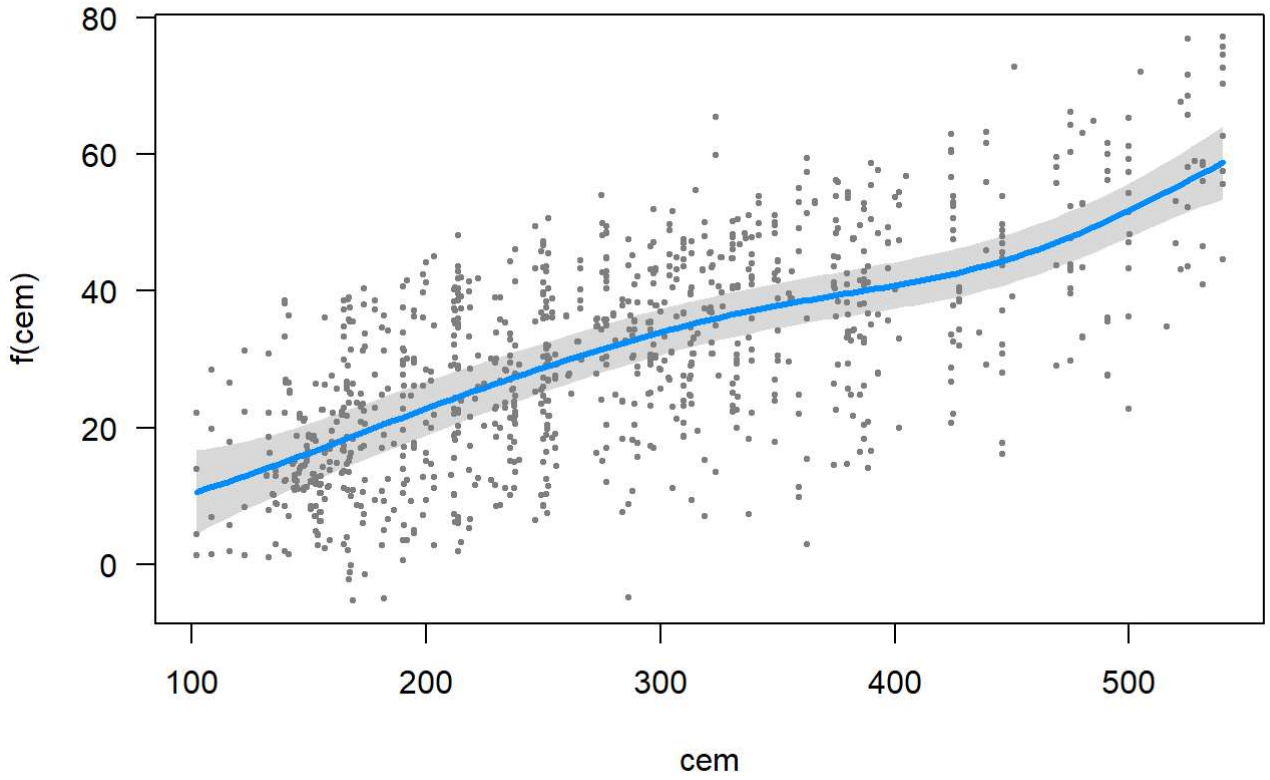
We couldn’t have assumed as much before, but now we have more statistical data to imply that integrating nonlinear covariate connections improves the model.

Visualizing with Visreg Library

```
library(visreg)
visreg(gamModel,'cem')
```



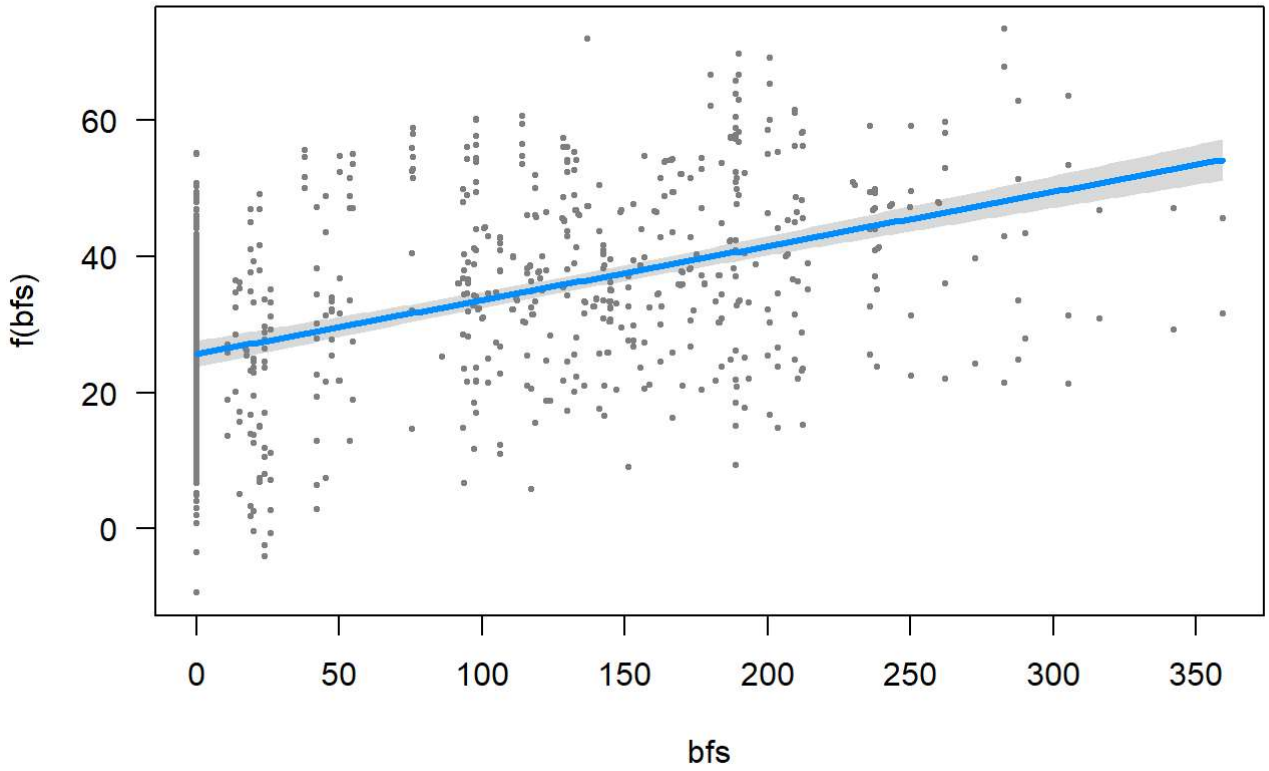

```
visreg(gamModel2,'cem')
```



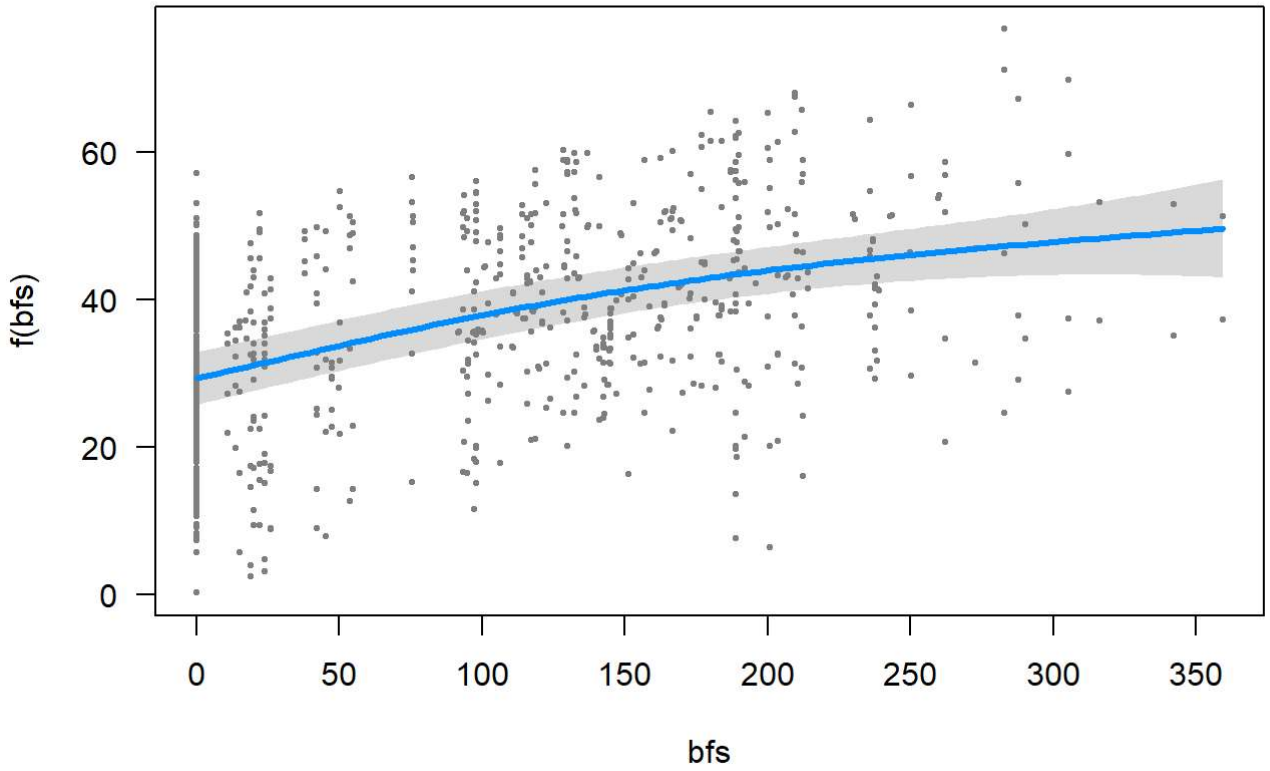
The end result is a plot of the

expected value of the CCS as a function of x (CEM), with all other variables in the model maintained constant. It contains the following elements: (1) the expected value (blue line) (2) a confidence interval for the expected value (gray band) and (3) partial residuals (dark gray dots)

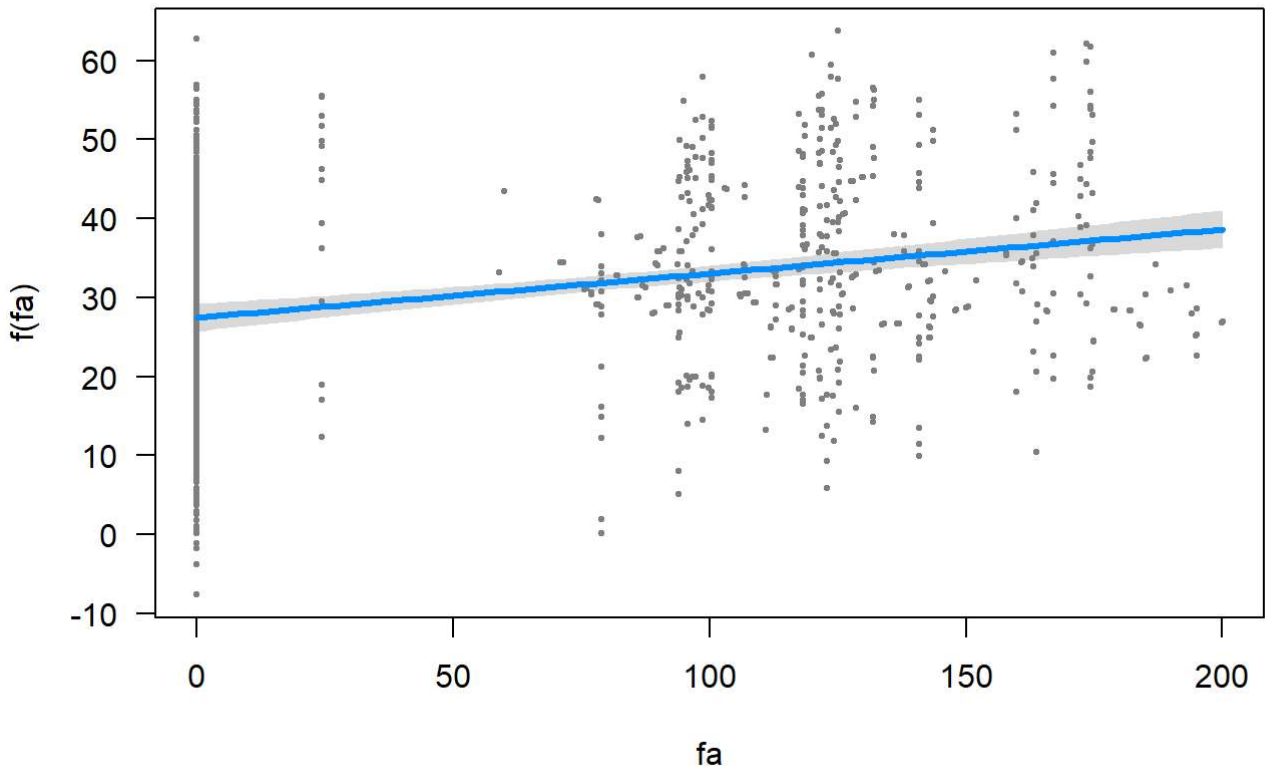
```
visreg(gamModel,'bfs')
```



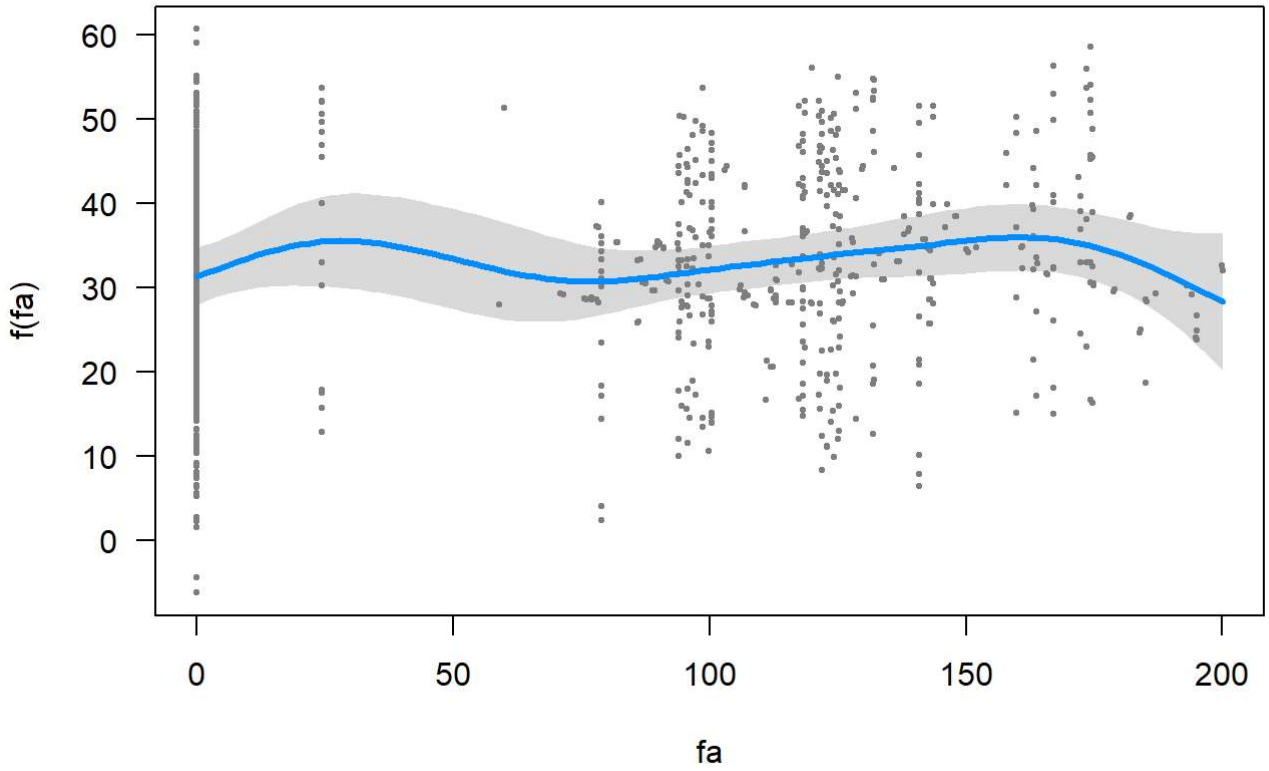

```
visreg(gamModel2,'bfs')
```



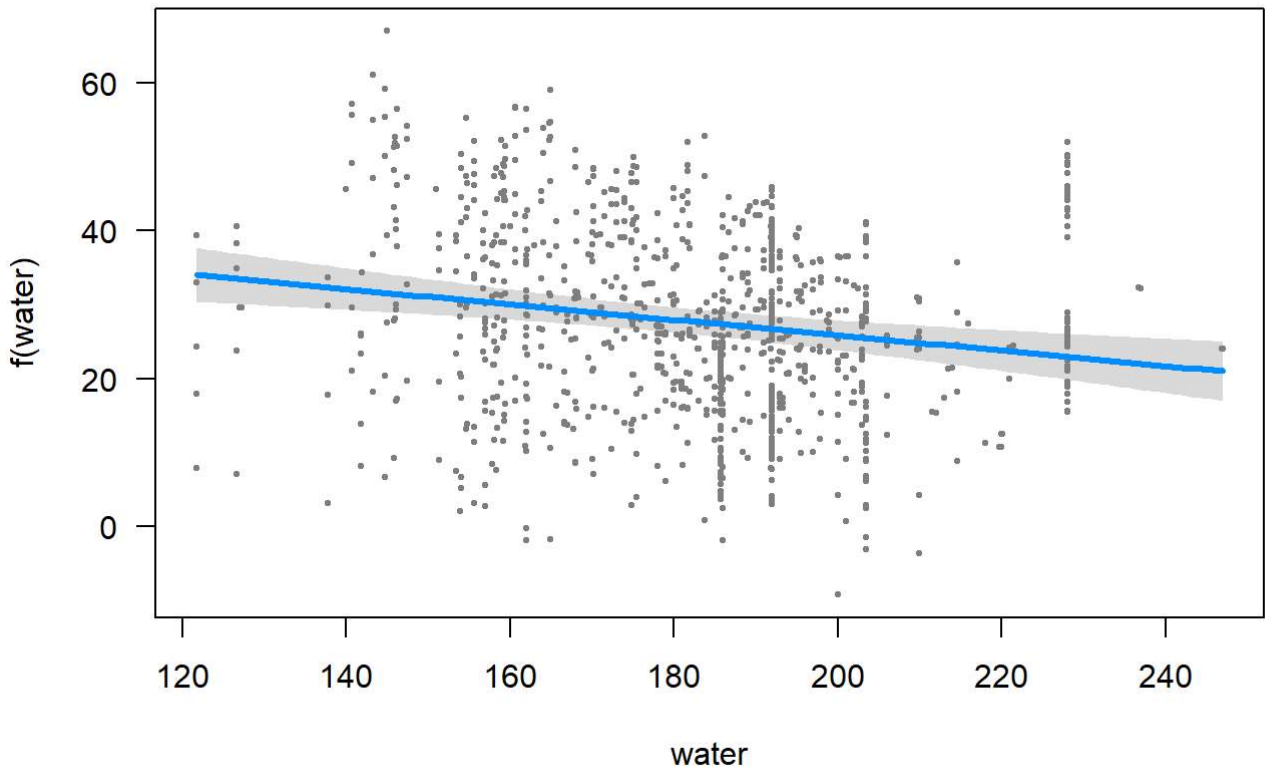

```
visreg(gamModel,'fa')
```



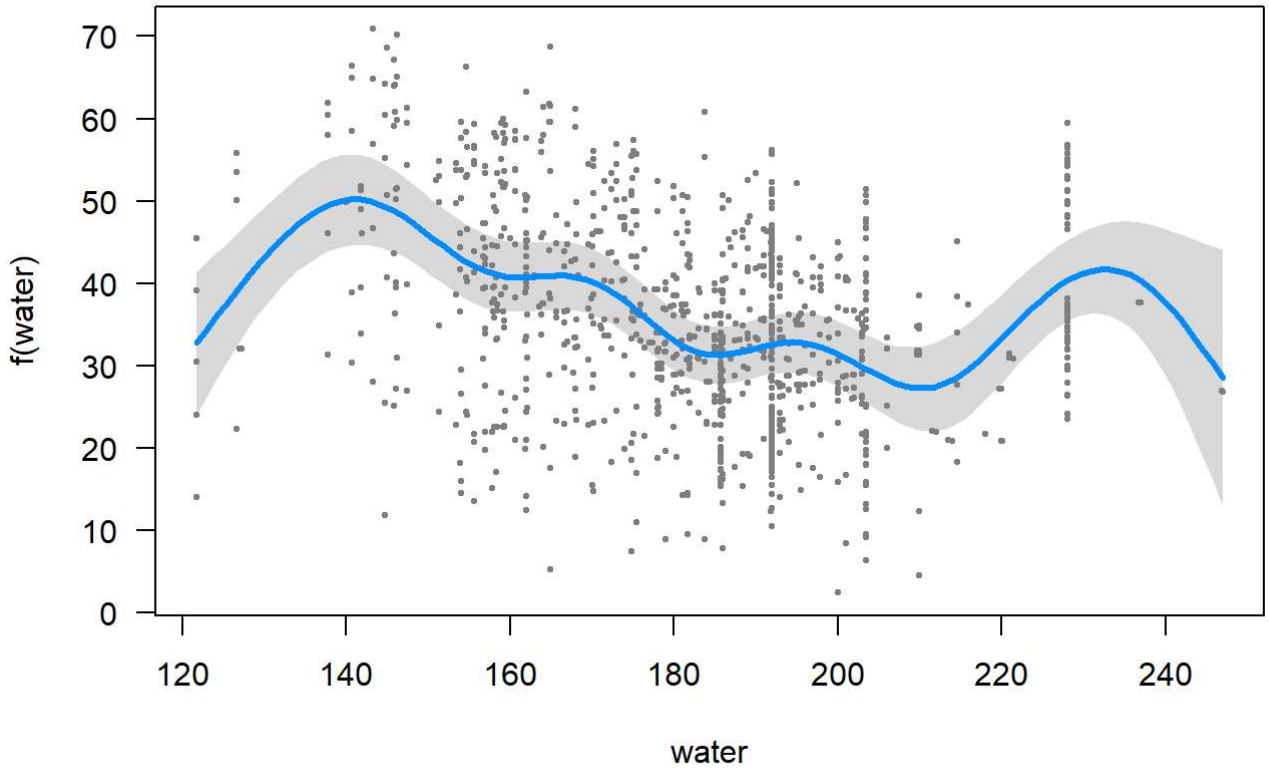

```
visreg(gamModel2,'fa')
```



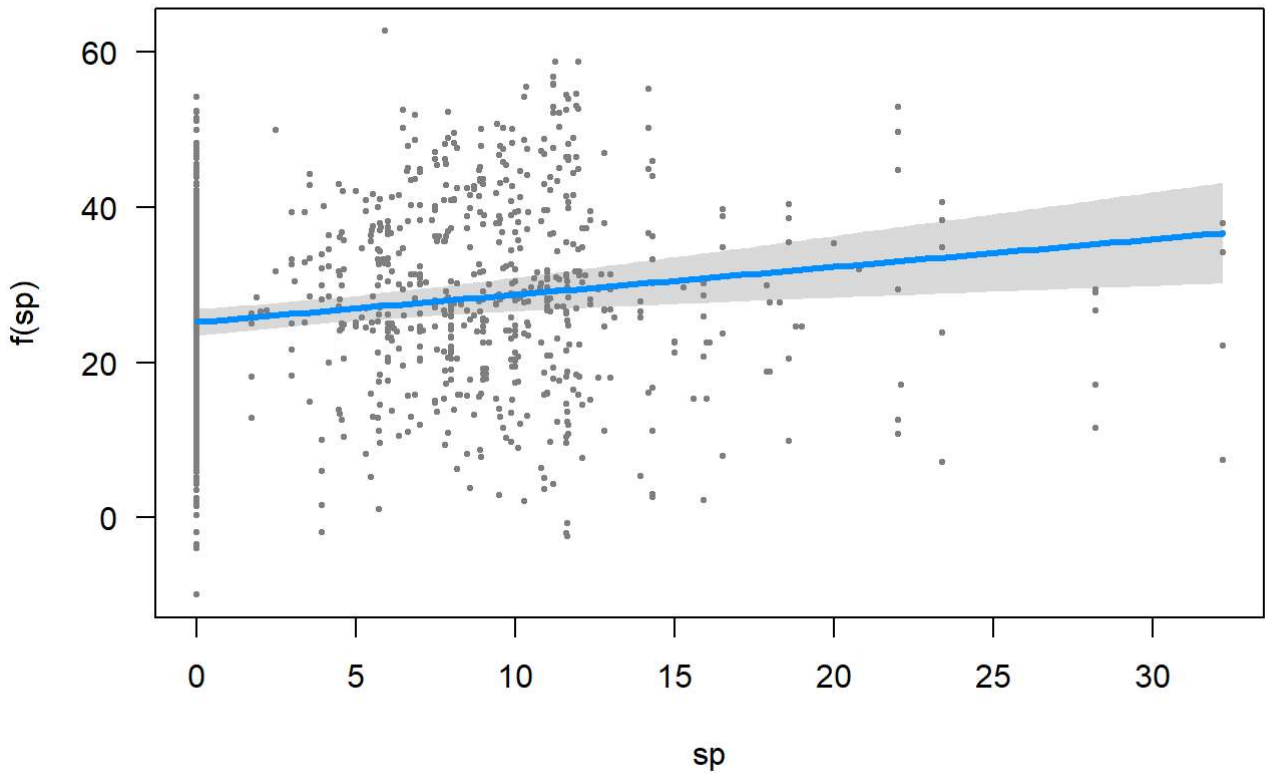

```
visreg(gamModel,'water')
```



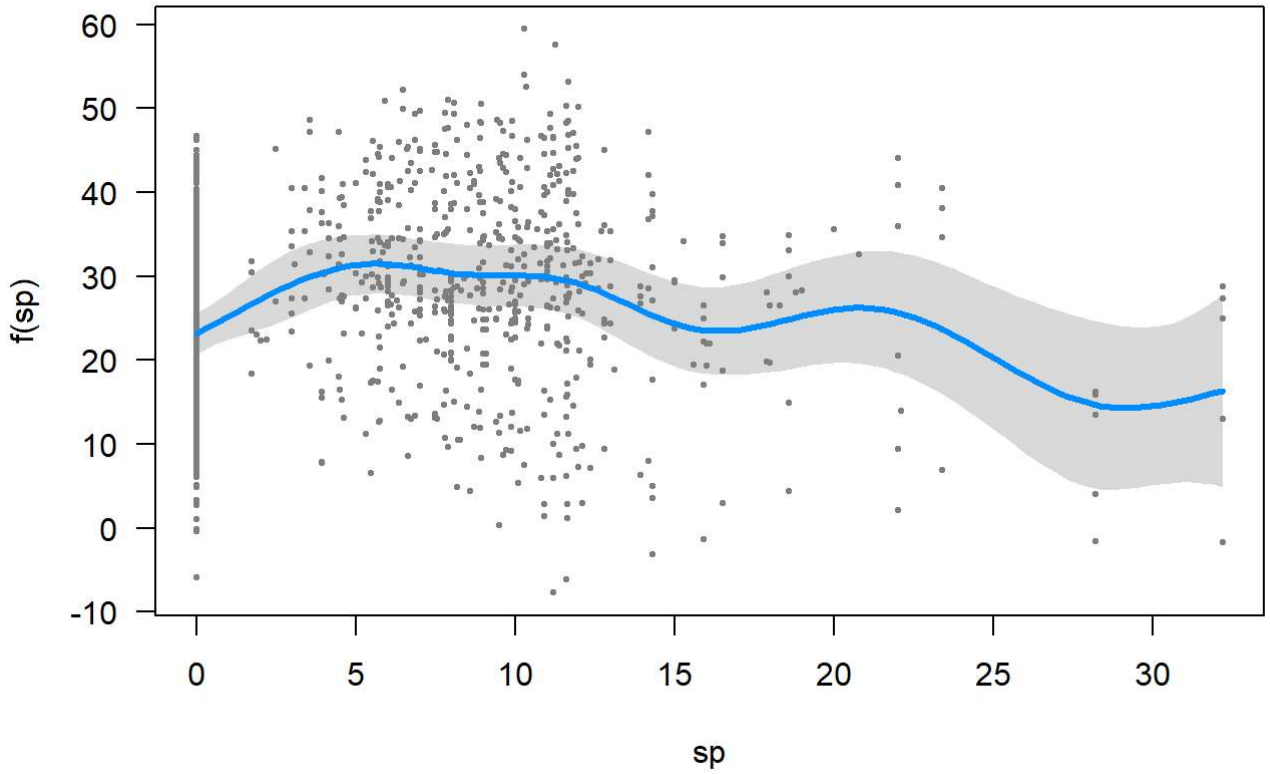

```
visreg(gamModel2,'water')
```



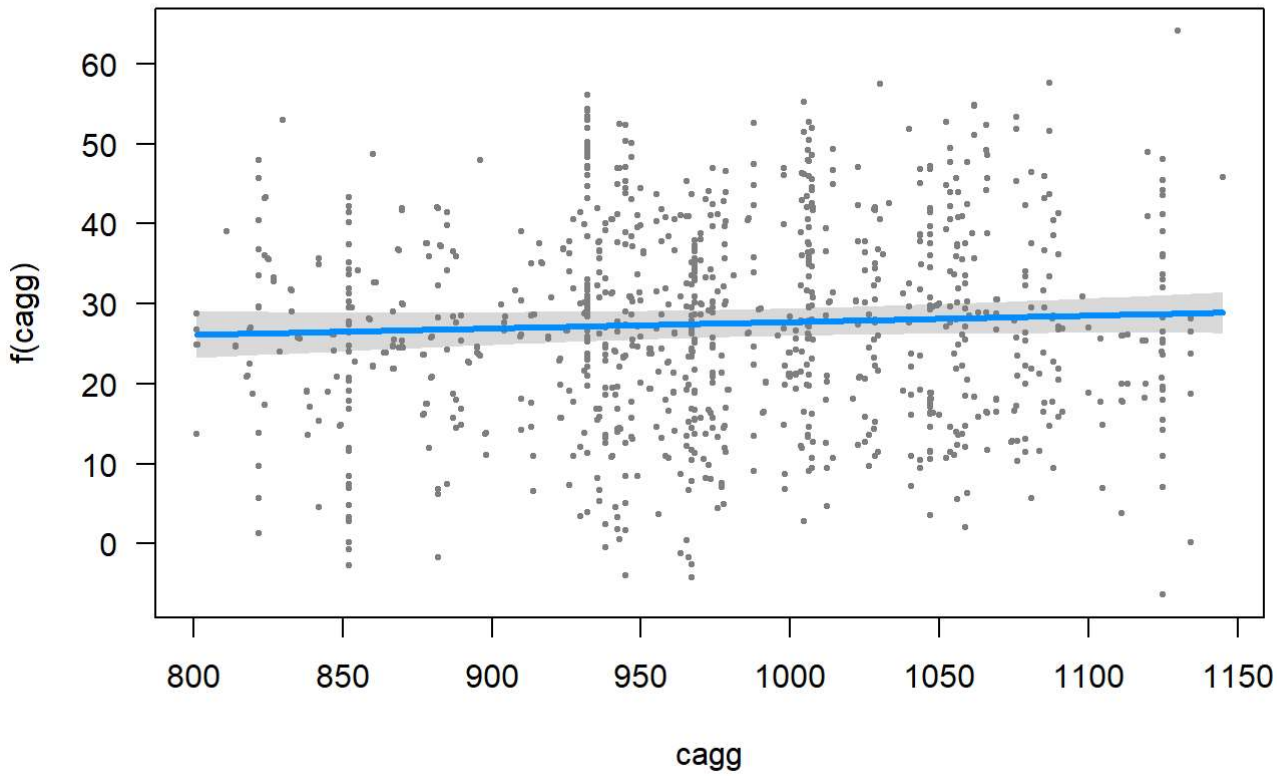

```
visreg(gamModel,'sp')
```



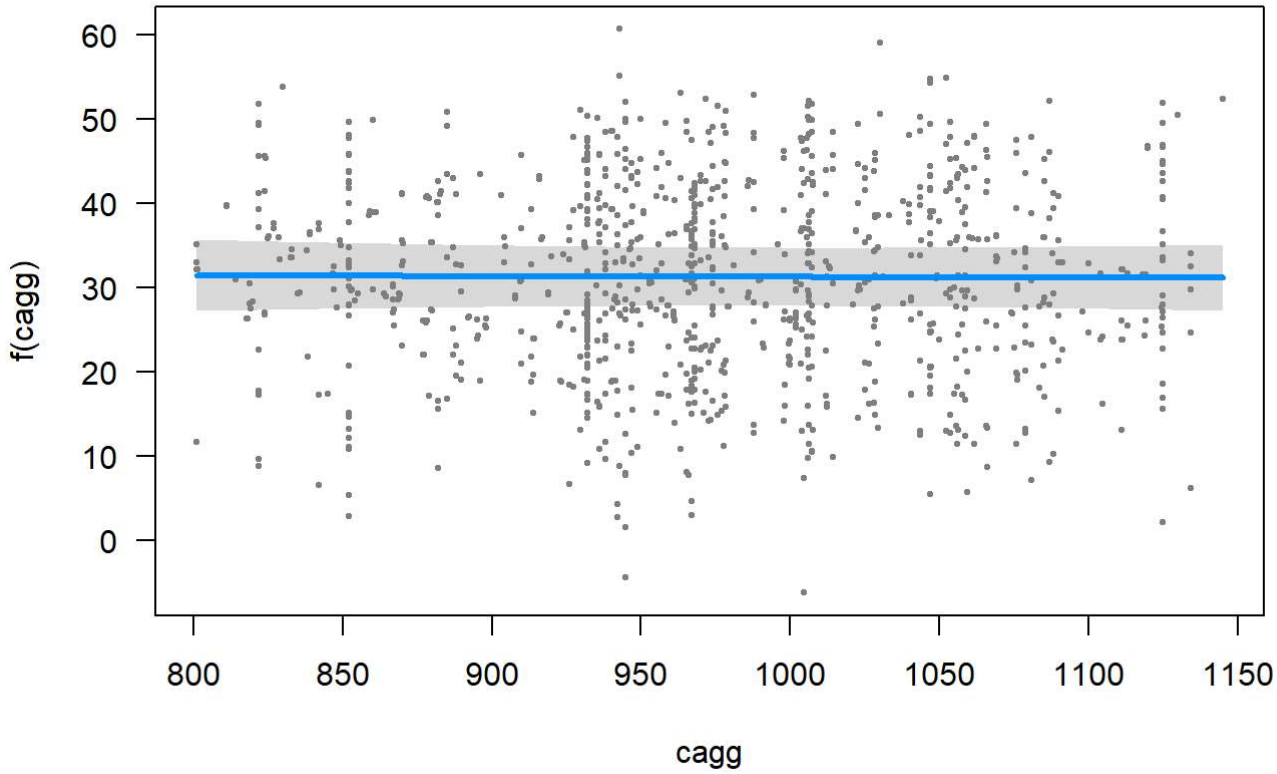

```
visreg(gamModel2,'sp')
```




```
visreg(gamModel,'cagg')
```




```
visreg(gamModel2,'cagg')
```



The CEM graph shows that the confidence interval after using the smoothing function has a higher value than the model before applying the smoothing function.