Shiva Sankar Modala | A20517528 | CSP 571 Spring 2023

Data Preparation and Analysis Assignment 3

Recitation Exercises:
Chapter 6
Exercise 1 Answers:

a. The ideal subset to choose for choosing the best model would be the one with the minimum RSS. Both stepwise forward and stepwise backward may produce the same model. There is no subset of predictors that could be identified by the forward stepwise and backward stepwise selection that would not be identified by the best selection because best subset selection trains all 2p possible models using p predictors and because the selection criteria is to get the smallest training RSS.

b. As the number of observations (n) is not provided, choosing between the three models that are presented is challenging. The model will be over fitted in the case of the best subset if n is comparatively less than p, and the other two models may by chance choose the model that performs best on the test set.

c.

i. TRUE :- because the (k+1) variable model has one more predictor in addition to all of the predictors chosen for the k-variable model in the forward stepwise selection.

ii. TRUE :- because in the backward stepwise selection, k-variable model is obtained by removing one predictor from (k+1)-variable model which will reduce the RSS of the model.

iii. FALSE :- because both the models follow different criteria.

iv. FALSE :- because both the models follow different criteria. Also, there is no link between the models obtained from forward and backward model.

v. FALSE :- because the best subset approach selects the model with (k+1) predictors from among all feasible models with (k+1) predictors. As a result, it does not ensure that the same predictors will be used for the k predictor model.

Exercise 2 Answers:

a. Option (iii) is correct

Justification: Since the shrinkage penalty is lower for p near to 0, estimates tend to be almost zero. As a result of term significance being higher than RSS, there is a bigger shrinking. At the expense of a slight increase in bias, this shrinking lowers the variance of the predictions. While least squares estimate have large volatility, the Lasso solution can reduce variance at the cost of a minor increase in bias. Lasso can reduce coefficient estimates by removing unimportant factors for more bias and less variance.

b. Option (iii) is correct

Justification: The goal function for ridge is RSS + |i2|pi=1. In this instance, the ridge's phrase for shrinkage is slightly different from Lasso's, but the remaining factors are the same as part of (a). Although the least common variable coefficients will not be shrunk to zero by the ridge regression, the remaining reasoning is still valid. Ridge, like Lasso, has the ability to reduce coefficient estimates, decreasing variance while increasing bias. Compared to least squares, ridge is less forgiving.

c. Option (ii) is correct

Justification: The non-linear technique is more adaptable because we are assuming less about the function form of f. The non-linear relationship between predictors and response is preferable for obtaining an estimated f because it produces less bias, which exceeds any increase in variance, and higher prediction accuracy. In comparison to least squares, nonlinear models are more adaptable and less biased.

Exercise 3 Answers:

a. Option (iv) is correct

Justification: The model becomes more flexible as we continue to increase λ from 0, which causes the training RSS to steadily drop. We are least constraining the coefficients j as we increase s from 0.

b. Option (ii) is correct

Justification: The coefficients βj are least constrained as we increase λ from 0, which makes the model more flexible and causes the test RSS to initially decline before growing again in a U shape.

c. Option (iii) is correct

Justification: The coefficients βj are least constrained as we continue to increase λ from 0, making the model more flexible and causing a steady rise in variance.

d. Option (iv) is correct

Justification: The model gets more flexible as we continue to increase λ from 0, which leads to a continuous decrease in bias because we are no longer placing as much restriction on the coefficients βj.

e. Option (v) is correct

Justification: The concept of irreducible mistake states that it is independent of both the model and the value of λ.

Exercise 4 Answers:

a. Option (iii) is correct

Justification: The coefficients βj are increasingly tightly constrained as we increase from 0, which makes the model less flexible and causes the training RSS to steadily rise.

b. Option (ii) is correct

Justification: The model becomes less flexible as we continue to increase from 0, further constraining the coefficients βj, initially causing a reduction in the test RSS before climbing again later in a U shape.

c. Option (iv) is correct

Justification: The model gets more flexible as we continue to increase λ from 0, which leads to a continuous decrease in bias because we are no longer placing as much restriction on the coefficients βj.

d. Option (iii) is correct

Justification: As we continue to increase from 0, we are more severely limiting the coefficients βj, which causes the model to become less flexible and causes bias to steadily rise.

e. Option (v) is correct

Justification: The concept of irreducible mistake states that it is independent of both the model and the value of λ.

Exercise 5 Answers:

a. Given that $x_{11} = x_{12} = x_1$ and $x_{21} = x_{22} = x_2$

From this the ridge regression optimization is:

$$(y_1 - \hat{\beta}_1 x_1 - \hat{\beta}_2 x_1)^2 + (y_2 - \hat{\beta}_1 x_2 - \hat{\beta}_2 x_2)^2 + \lambda(\hat{\beta}^2_1 + \hat{\beta}^2_2)$$

b. By taking the derivatives on the given equation with respect to $\hat{\beta}_1$ and $\hat{\beta}_2$ and setting them equal to 0, we get two equations

$$\hat{\beta}_1(x_1^2 + x_2^2 + \lambda) + \hat{\beta}_2(x_1^2 + x_2^2) = y_1x_1 + y_2x_2$$

and

$$\hat{\beta}_1(x_1^2 + x_2^2) + \hat{\beta}_2(x_1^2 + x_2^2 + \lambda) = y_1x_1 + y_2x_2$$

Then by subtracting the two expressions above we get $\hat{\beta}_1 = \hat{\beta}_2$

c. Given that $x_{11} = x_{12} = x_1$ and $x_{21} = x_{22} = x_2$

From this the ridge regression optimization is:

$$(y_1 - \hat{\beta}_1 x_1 - \hat{\beta}_2 x_1)^2 + (y_2 - \hat{\beta}_1 x_2 - \hat{\beta}_2 x_2)^2 + \lambda(|\hat{\beta}_1| + |\hat{\beta}_2|)$$

d. We can use the alternate form of the lasso optimization problem

$$(y_1 - \hat{\beta}_1 x_1 - \hat{\beta}_2 x_1)^2 + (y_2 - \hat{\beta}_1 x_2 - \hat{\beta}_2 x_2)^2 \text{ subject to } (|\hat{\beta}_1| + |\hat{\beta}_2|) \leq s$$

The lasso constraint is represented geometrically by a diamond with a center at the origin of the plane $(\hat{\beta}_1, \hat{\beta}_2)$ and an intersection with the axes at s from the origin. We must minimize the following expression by using the information from this issue
($x_{11} = x_{12} = x_1$ and $x_{21} = x_{22} = x_2$ and $x_1 + x_2 = 0$, and $y_1 + y_2 = 0$)
Into the following expression:
$2[y_1 - (\hat{\beta}_1 + \hat{\beta}_2)x_1]^2 \geq 0$
The answer to this optimization issue is straightforward: $\hat{\beta}_1 + \hat{\beta}_2 = (y_1/x_1)$
This is a line that is geometrically parallel to the edge of the constraints' diamond. The contours of the function $[y_1 - (\hat{\beta}_1 + \hat{\beta}_2)x_1]^2$ that intersect the diamond of the constraints are the solutions to the lasso optimization issue at this point. This means that the complete edge $\hat{\beta}_1 + \hat{\beta}_2 = s$ (as is the case for the edge $\hat{\beta}_1 + \hat{\beta}_2 = -s$) could be a solution to the lasso optimization problem. As a result, there are numerous possible solutions to the lasso optimization problem rather than just one:

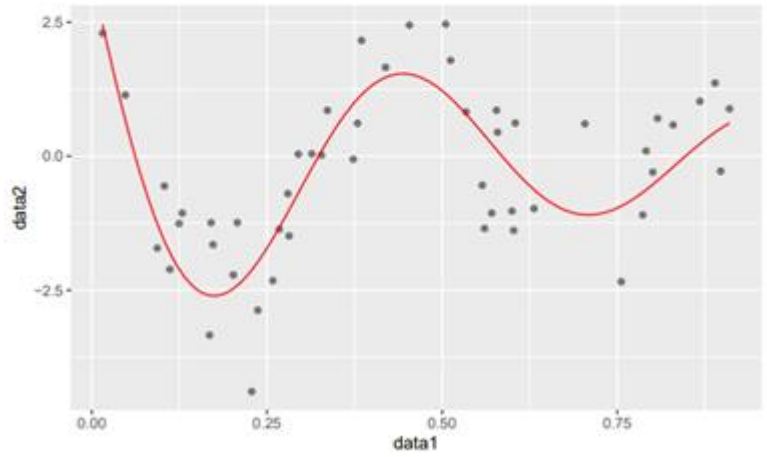
$$\{(\hat{\beta}_1, \hat{\beta}_2) : \hat{\beta}_1 + \hat{\beta}_2 = s \text{ with } \hat{\beta}_1, \hat{\beta}_2 \geq 0 \text{ and } \hat{\beta}_1 + \hat{\beta}_2 = -s \text{ with } \hat{\beta}_1, \hat{\beta}_2 \leq 0\}$$

Chapter 7
Exercise 2 Answers:

Code:

```
library(ggplot2)
set.seed(3)

data1 = runif(50)
eps = rnorm(50)
data2 = sin(12*(data1 + 0.2)) / (data1 + 0.2) + eps
generating_fn = function(data1) {sin(12*(data1 + 0.2)) / (data1 + 0.2)}
dframe <- data.frame(data1, data2)

ggplot(dframe, aes(x = data1, y = data2)) +
geom_point(alpha = 0.5) +
stat_function(fun = generating_fn, aes(col = "Curve")) +
scale_color_manual(values = "red") +
theme(legend.position = "bottom", legend.title = element_blank())
```

```
library(ggplot2)
set.seed(3)

data1 = runif(50)
eps = rnorm(50)
data2 = sin(12*(data1 + 0.2)) / (data1 + 0.2) + eps
generating_fn = function(data1) {sin(12*(data1 + 0.2)) / (data1 + 0.2)}
dframe <- data.frame(data1, data2)

ggplot(dframe, aes(x = data1, y = data2)) +
  geom_point(alpha = 0.5) +
  stat_function(fun = generating_fn, aes(col = "Curve")) +
  scale_color_manual(values = "red") +
  theme(legend.position = "bottom", legend.title = element_blank())
```



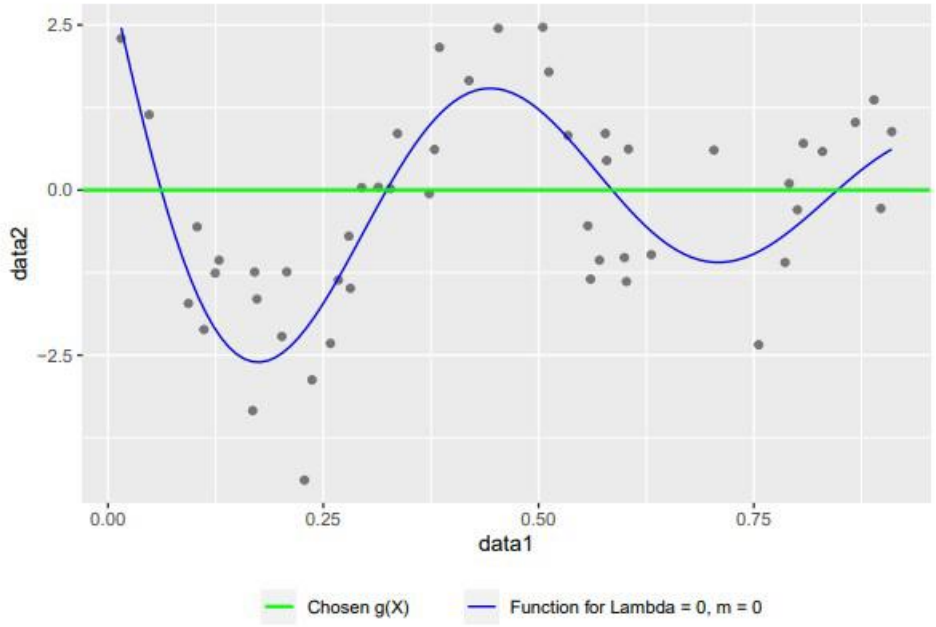
a. λ = ∞, m = 0.

For m=0, we get: g^=argming(Σi=1n(yi−g(xi))2+λ∫[g(x)]2dx) As λ increases, the penalty term becomes more and more important in the above equation. As λ→∞, this forces g(x)→0. We therefore get g^(x)=0:

Code:

```
ggplot(dframe, aes(x = data1, y = data2)) +
  geom_point(alpha = 0.5) +
  stat_function(fun = generating_fn, aes(col = "Function")) +
  geom_hline(aes(yintercept = 0, linetype = "Chosen g(X)"), col = "green", size = 0.8) +
  scale_color_manual(values = "blue") +  theme(legend.position = "bottom", legend.title =
element_blank())
```

```r
library(ggplot2)
set.seed(3)

data1 = runif(50)
eps = rnorm(50)
data2 = sin(12*(data1 + 0.2)) / (data1 + 0.2) + eps
generating_fn = function(data1) {sin(12*(data1 + 0.2)) / (data1 + 0.2)}
dframe <- data.frame(data1, data2)

ggplot(dframe, aes(x = data1, y = data2)) +
  geom_point(alpha = 0.5) +
  stat_function(fun = generating_fn, aes(col = "Function for Lambda = 0, m = 0")) +
  geom_hline(aes(yintercept = 0, linetype = "Chosen g(X)"), col = "green", size = 0.8) +
  scale_color_manual(values = "blue") +  theme(legend.position = "bottom", legend.title = element_blank
```



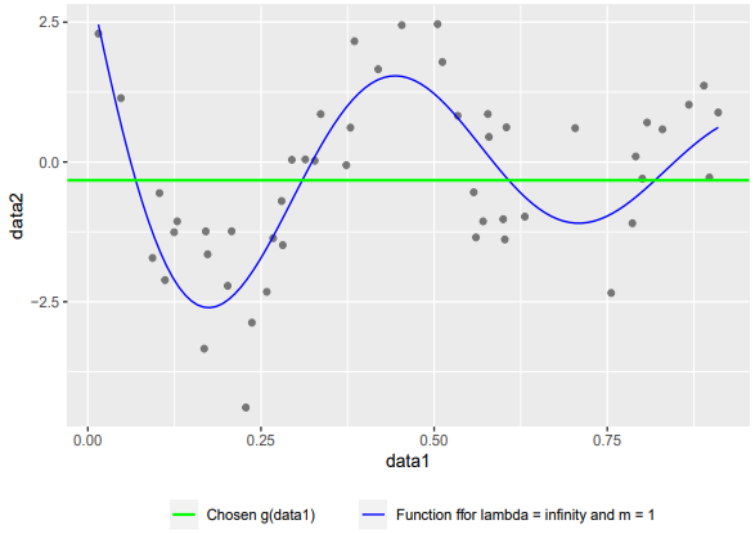
b. $\lambda = \infty$, m = 1.

For m=1, we will get: $\hat{g}=\text{argming}(\sum_{i=1}^{n}(y_i-g(x_i))^2+\lambda\int[g'(x)]^2dx)$ As a result $\lambda\to\infty$, this forces $g'(x)\to 0$. As a result, we would obtain $\hat{g}(x)=c$. More specifically, we can use $\hat{g}(x)=c=\frac{1}{n}\sum_{i=1}^{n}y_i$ because all constant functions have a first derivative of zero, yet this one will also reduce the RSS.

Code:

```r
ggplot(dframe, aes(x = data1, y = data2)) +
  geom_point(alpha = 0.5) +
  stat_function(fun = generating_fn, aes(col = "Function for lambda = infinity and m = 1"))
+
  geom_hline(aes(yintercept = mean(data2), linetype = "Chosen g(data1)"), col = "green",
size = 0.8) +
  scale_color_manual(values = "blue") +
  theme(legend.position = "bottom", legend.title = element_blank())
```

```r
library(ggplot2)
set.seed(3)

data1 = runif(50)
eps = rnorm(50)
data2 = sin(12*(data1 + 0.2)) / (data1 + 0.2) + eps
generating_fn = function(data1) {sin(12*(data1 + 0.2)) / (data1 + 0.2)}
dframe <- data.frame(data1, data2)

ggplot(dframe, aes(x = data1, y = data2)) +
  geom_point(alpha = 0.5) +
  stat_function(fun = generating_fn, aes(col = "Function ffor lambda = infinity and m = 1")) +
  geom_hline(aes(yintercept = mean(data2), linetype = "Chosen g(data1)"), col = "green", size = 0.8) +
  scale_color_manual(values = "blue") +
  theme(legend.position = "bottom", legend.title = element_blank())
```



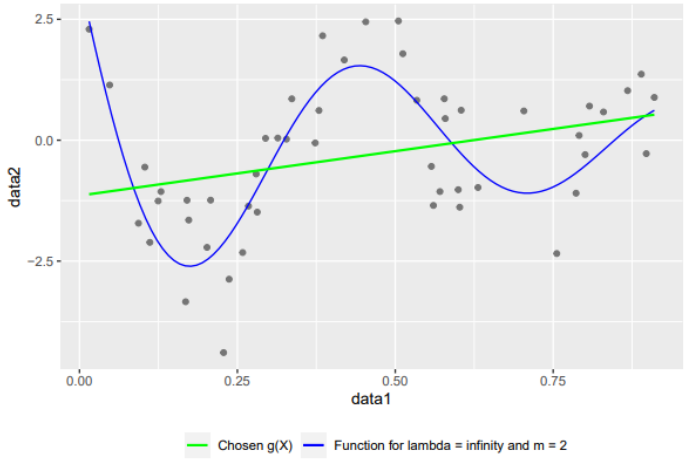
c. λ = ∞, m = 2.

For m=2, we will get: g^=argming(∑i=1n(yi−g(xi))2+λ∫[g″(x)]2dx) As a result λ→∞, this forces g″(x)→0. This implies we would still get g^(x)=ax+b. More precisely, g^(x) is the linear least squares line, because all linear functions do have a second derivative of zero, yet this one also reduces the RSS.

Code:

```r
ggplot(dframe, aes(x = data1, y = data2)) + geom_point(alpha = 0.5) +
  stat_function(fun = generating_fn, aes(col = "Function for lambda = infinity and m = 2")) +
  geom_smooth(method = "lm", formula = "y ~ x", se = F, size = 0.8, aes(col = "Chosen g(X)"))+   scale_color_manual(values = c("green", "blue")) +
  theme(legend.position = "bottom", legend.title = element_blank())
```

```r
library(ggplot2)
set.seed(3)

data1 = runif(50)
eps = rnorm(50)
data2 = sin(12*(data1 + 0.2)) / (data1 + 0.2) + eps
generating_fn = function(data1) {sin(12*(data1 + 0.2)) / (data1 + 0.2)}
dframe <- data.frame(data1, data2)

ggplot(dframe, aes(x = data1, y = data2)) + geom_point(alpha = 0.5) +
  stat_function(fun = generating_fn, aes(col = "Function for lambda = infinity and m = 2")) +
  geom_smooth(method = "lm", formula = "y ~ x", se = F, size = 0.8, aes(col = "Chosen g(X)")) +
  scale_color_manual(values = c("green", "blue")) +
  theme(legend.position = "bottom", legend.title = element_blank())
```



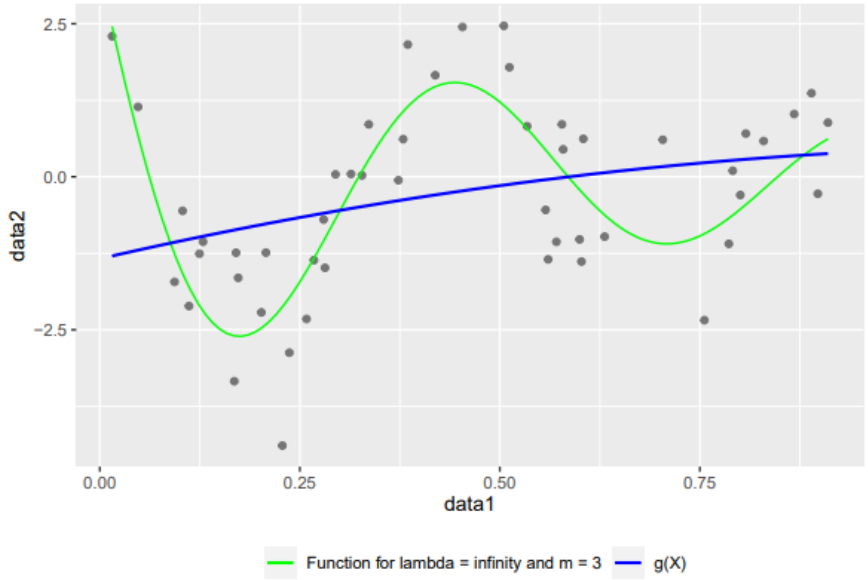
d. λ = ∞, m = 3.

For m=3, we will get: g^=argming(∑i=1n(yi−g(xi))2+λ∫[g(3)(x)]2dx) As a result λ→∞, this forces g(3)(x)→0. This implies that we would still get g^(x)=ax2+bx+c. More precisely we can say that g^(x) is the quadratic least squares line, because all quadratic functions have a third derivative of Zero(0), this one will also reduce the RSS.

Code:

```
ggplot(dframe, aes(x = data1, y = data2)) +
  geom_point(alpha = 0.5) +
  stat_function(fun = generating_fn, aes(col = "Function for lambda = infinity and m = 3"))
+
  geom_smooth(method = "lm", formula = "y ~ x + I(x^2)", se = F,
              size = 0.8, aes(col = "g(X)")) +
  scale_color_manual(values = c("green", "blue")) +
  theme(legend.position = "bottom", legend.title = element_blank())
```

```
library(ggplot2)
set.seed(3)

data1 = runif(50)
eps = rnorm(50)
data2 = sin(12*(data1 + 0.2)) / (data1 + 0.2) + eps
generating_fn = function(data1) {sin(12*(data1 + 0.2)) / (data1 + 0.2)}
dframe <- data.frame(data1, data2)

ggplot(dframe, aes(x = data1, y = data2)) +
  geom_point(alpha = 0.5) +
  stat_function(fun = generating_fn, aes(col = "Function for lambda = infinity and m = 3")) +
  geom_smooth(method = "lm", formula = "y ~ x + I(x^2)", se = F,
              size = 0.8, aes(col = "g(X)")) +
  scale_color_manual(values = c("green", "blue")) +
  theme(legend.position = "bottom", legend.title = element_blank())
```



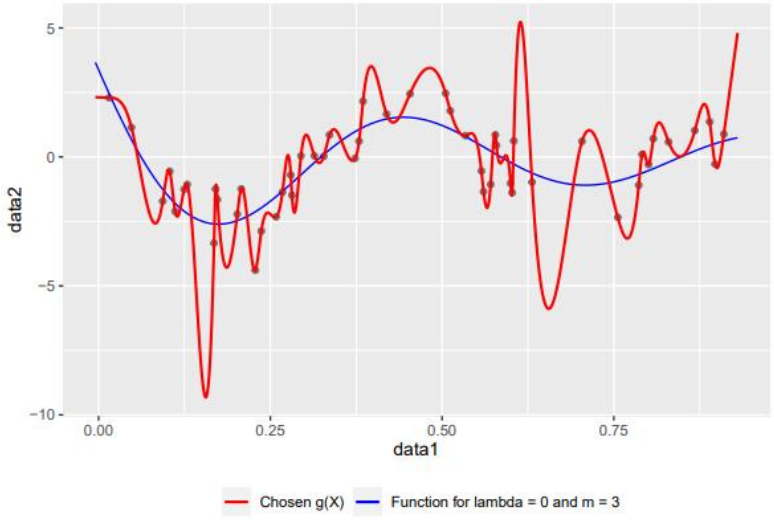
e. λ = 0, m = 3.

For m=3, we will get the same as: g^=argming(∑i=1n(yi−g(xi))2+λ∫[g(3)(x)]2dx) Although, since λ=0, the penalty term has no effect on the selection of g^(x). As a result, RSS = 0 can be obtained with any g(x) that calculates all of the observations! Trying to take a cubic smoothing spline (without a smoothing) as an instance:

Code:

```
spline1 = smooth.spline(x = dframe$data1, y = dframe$data2, all.knots = T, lambda = 0.0000000000001)
fitted = predict(spline1, x = seq(min(data1) - 0.02, max(data1) + 0.02, by = 0.0001))
fitted = data.frame(x = fitted$x, fitted_y = fitted$y)
ggplot(dframe, aes(x = data1, y = data2)) +
  geom_point(alpha = 0.5) +
  stat_function(fun = generating_fn, aes(col = "Function for lambda = 0 and m = 3")) +
  geom_line(data = fitted,  aes(x = x, y = fitted_y, col = "Chosen g(X)"), size = 0.8) +
  scale_color_manual(values = c("red", "blue")) +
  theme(legend.position = "bottom", legend.title = element_blank())
```

```r
library(ggplot2)
set.seed(3)

data1 = runif(50)
eps = rnorm(50)
data2 = sin(12*(data1 + 0.2)) / (data1 + 0.2) + eps
generating_fn = function(data1) {sin(12*(data1 + 0.2)) / (data1 + 0.2)}
dframe <- data.frame(data1, data2)

spline1 = smooth.spline(x = dframe$data1, y = dframe$data2, all.knots = T, lambda = 0.0000000000001)
fitted = predict(spline1, x = seq(min(data1) - 0.02, max(data1) + 0.02, by = 0.0001))
fitted = data.frame(x = fitted$x, fitted_y = fitted$y)
ggplot(dframe, aes(x = data1, y = data2)) +
  geom_point(alpha = 0.5) +
  stat_function(fun = generating_fn, aes(col = "Function for lambda = 0 and m = 3")) +
  geom_line(data = fitted,  aes(x = x, y = fitted_y, col = "Chosen g(X)"), size = 0.8) +
  scale_color_manual(values = c("red", "blue")) +
  theme(legend.position = "bottom", legend.title = element_blank())
```



Exercise 3 Answers:

The curve for model and the fitted curve f^(X)=1+X−2(X−1)2I(X≥1):

Code:

```
X = seq(-2, 2, 0.01)
Y = 1 + X + -2 * (X - 1)^2 * (X >= 1)
df <- data.frame(X, Y)

ggplot(df, aes(x = X, y = Y)) +
  geom_vline(xintercept = 0) +
  geom_vline(xintercept = 1, col = "red") +
  geom_hline(yintercept = 0) +
  geom_line(size = 1)
```

For $x<1$, we have $\hat{f}(x) = 1+ x$,

As a result, the slope and intercept are both 1. As a result of the indicator variable $I(x \geq 1)$, the quadratic term becomes relevant only for $x \geq 1$, and that's why the curve is linear before that point. When we have $x \geq 1$, the fitted equation is:

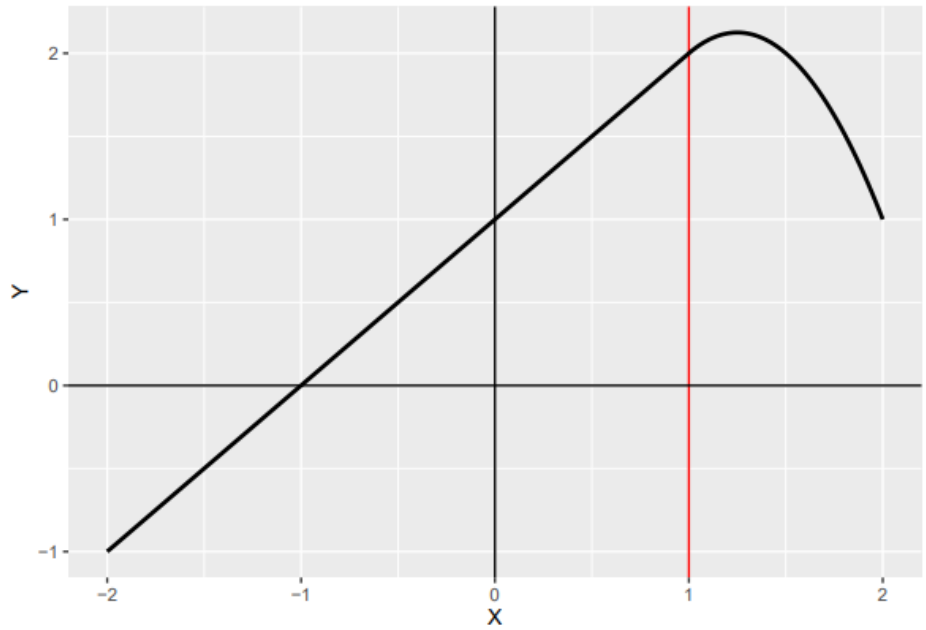
$\hat{f}(x)= 1+ x -2(x -1)^2$

$= 1+ x -2(x^2-2 x +1)$

$= -2x^2+5 x -1$

Using the derivative, we can observe that the slope for $x \geq 1$ varies - $\hat{f}'(x)= -4x+5$ Applying this derivative to zero explains why the graph's critical point occurs at $X=\frac{5}{4}$.

```
library(ggplot2)
X = seq(-2, 2, 0.01)
Y = 1 + X + -2 * (X - 1)^2 * (X >= 1)
df <- data.frame(X, Y)

ggplot(df, aes(x = X, y = Y)) +
  geom_vline(xintercept = 0) +
  geom_vline(xintercept = 1, col = "red") +
  geom_hline(yintercept = 0) +
  geom_line(size = 1)
```



Exercise 4 Answers:

Using these basis functions and coefficient estimates,
We have obtain the fitted model: $\hat{f}(x)=1 + I(0\leq X\leq 2) - (X-1)\, I(1\leq X\leq 2) + 3(X-3)\, I(3\leq X\leq 4) + 3I(4<X\leq 5)$
Because we are only willing to get involved with in the function over the range $[-2,2]$, we can simplify $\hat{f}(x)$ over this range: $\hat{f}(x) = 1+I(0\leq X\leq 2)-(X-1)I(1\leq X\leq 2)$

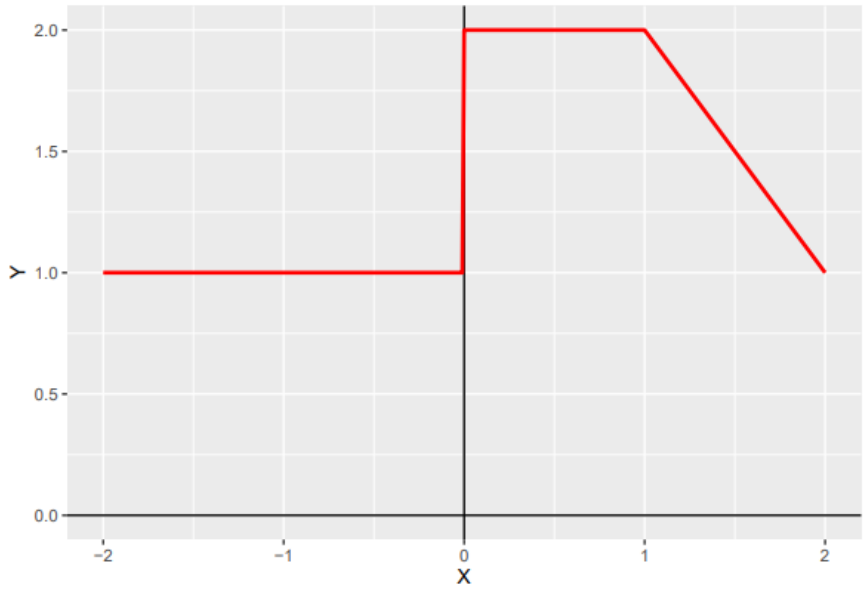
Code

```
library(ggplot2)
X = seq(-2, 2, 0.01)
Y = 1 + (X >= 0 & X <= 2) - (X - 1)*(X >= 1 & X <= 2) +
  3*(X - 3)*(X >= 3 & X <= 4) + 3*(X > 4 & X <= 5)
df <- data.frame(X, Y)

ggplot(df, aes(x = X, y = Y)) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +
  geom_line(size = 1, col = "red")
```

Because none of the indicator variables are true for $X<0$, so $\hat{f}(X)=1$ and has a slope of 0. For $0\leq X<1$, the first indicator variable is true, so $\hat{f}(X)=1+1=2$, indicating that this is the intercept with a slope of 0. For $1\leq X\leq 2$, both indicator variables are true, so

$\hat{f}(X) = 1 + 1 - (X-1) \cdot 1 = 3 - X$, indicating that the slope is 1.

```
library(ggplot2)
X = seq(-2, 2, 0.01)
Y = 1 + (X >= 0 & X <= 2) - (X - 1)*(X >= 1 & X <= 2) +
  3*(X - 3)*(X >= 3 & X <= 4) + 3*(X > 4 & X <= 5)
df <- data.frame(X, Y)

ggplot(df, aes(x = X, y = Y)) +
  geom_vline(xintercept = 0) +
  geom_hline(yintercept = 0) +
  geom_line(size = 1, col = "red")
```



Exercise 5 Answers:

a. G(3)(x)0 is forced for $\hat{g}1$ because G(x)=ax2+bx+c is the highest order (most adaptable) polynomial that really can meet this requirement (as the third derivative would be zero). Therefore, $\hat{g}1$ will be the quadratic that reduces the training RSS.
G(4)(x)0 is forced for $\hat{g}_2$ by G(x)=ax3+bx2+cx+d seems to be the highest order (most adaptable) polynomial to satisfy this condition (as the third derivative would be zero). As a result, $\hat{g}2$ will be the cubic that reduces the training RSS.
As the higher order polynomial, $\hat{g}2$ would be more adaptable, and hence $\hat{g}2$ has a smaller training RSS.

b. We don't know; it varies depends on whether a cubic or quadratic relationship is a better approximation of the underlying relationship between x and y. Either $\hat{g}2$ or $\hat{g}1$ may be overfit or underfit and have bigger test RSSs, depending on the situation.

c. For $\lambda=0$, there is no restriction at all on g(x), so both $\hat{g}1$ and $\hat{g}2$ would have the exact same training RSS (of zero if all the xi are unique). With no restrictions on g, we can simply have any function that interpolates all of the training observations; for an instance, see question 2) e).
Both these functions will have high test RSS as well as be highly overfit. They would also undoubtedly have the same test RSS. If we assume that the same interpolating function was chosen for $\hat{g}1$ and $\hat{g}2$ (e.g., both were linear splines with knots at each distinct xi) they would also have the same test RSS.

2. Practicum Problems

(I’m attaching the pdf markdown files I created from r script.)

mtcars

Shiva Sankar Modala

2023-03-01

```
library(caret)
```

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

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

```
library(glmnet)
```

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

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

```
# Creating 80-20 Training Testing Split, createDataPartition() returns the
indices
# Perform a basic 80/20 test-train split on the data (you may use caret, the
sample method, or manually)
initial_train = createDataPartition(mtcars$mpg,times=1,p=0.8,list=FALSE)
# Training data
training_data= mtcars[initial_train, ]
# Testing data (note the minus sign)
testing_data= mtcars[-initial_train, ]
training_data$am = factor(training_data$am)
is.factor(training_data$am)
```

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

```
# Fitting linear model
# Fit a linear model with mpg as the target response,
testing_data$am = factor(testing_data$am)
lm.fit = lm(mpg~.,data=training_data)
#MSE on test set
mean((predict(lm.fit,testing_data)-testing_data$mpg)^2)
```

```
## [1] 11.26835
```

```
# What features are selected as relevant based on resulting t-statistics?
# Analyze the t-stat and p-values to select relevant features
summary(lm.fit)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = training_data)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -2.9424 -1.7282 -0.2225  1.0956  5.4001
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24.62378   19.65034   1.253    0.227
## cyl         -0.41449    1.09482  -0.379    0.710
## disp         0.01090    0.01814   0.601    0.556
## hp          -0.03299    0.02539  -1.299    0.211
## drat         0.88507    1.76755   0.501    0.623
## wt          -2.73163    2.07093  -1.319    0.205
## qsec         0.18021    0.86946   0.207    0.838
## vs           0.08982    2.36188   0.038    0.970
## am1          1.15988    2.43176   0.477    0.639
## gear         0.85259    1.54799   0.551    0.589
## carb        -0.41727    0.91617  -0.455    0.655
##
## Residual standard error: 2.632 on 17 degrees of freedom
## Multiple R-squared:  0.8699, Adjusted R-squared:  0.7934
## F-statistic: 11.37 on 10 and 17 DF,  p-value: 1.079e-05
```

```
cat(" We will select wt as a predictor based on the statistics as it has the
lowest p value.")
```

```
##  We will select wt as a predictor based on the statistics as it has the
lowest p value.
```

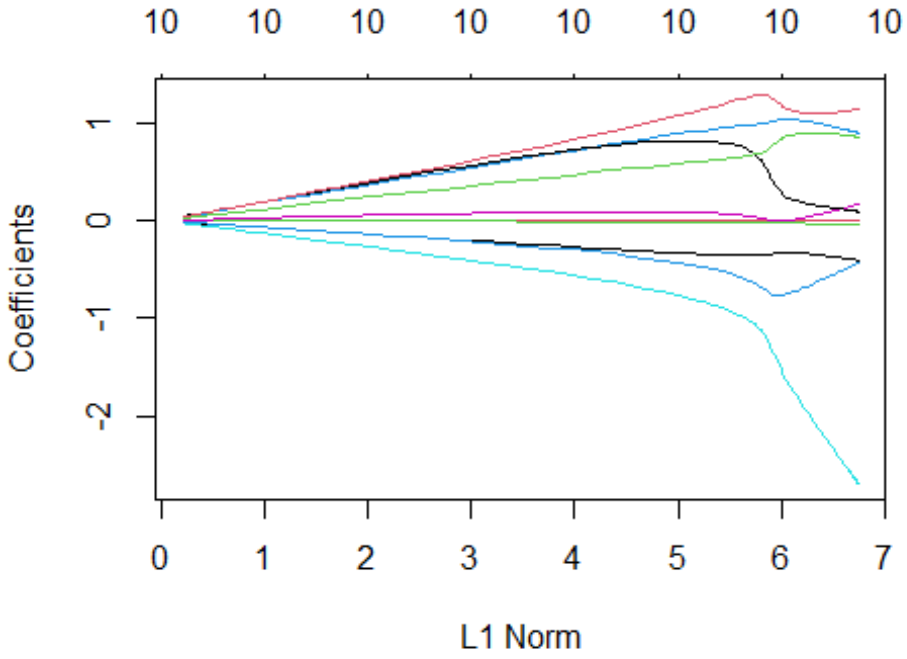
```
# coefficient values for relevant features
lm.fit$coefficients
```

```
## (Intercept)         cyl        disp          hp        drat          wt
## 24.62378182 -0.41448777  0.01090413 -0.03298694  0.88506840 -2.73162674
##        qsec          vs          am1        gear        carb
##  0.18021450  0.08982164  1.15987939  0.85259465 -0.41726838
```

```
lambda_seq = 10^seq(3, -3, by= -.06)
# Perform a ridge regression using the glmnet package
ridge_regression<-glmnet(model.matrix(training_data$mpg~.,data =
training_data)[, - 1],training_data$mpg,alpha=0,lambda=lambda_seq)
summary(ridge_regression)
```

```
##           Length Class     Mode
## a0         101   -none-    numeric
## beta      1010   dgCMatrix S4
## df         101   -none-    numeric
## dim          2   -none-    numeric
## lambda     101   -none-    numeric
## dev.ratio  101   -none-    numeric
## nulldev      1   -none-    numeric
## npasses      1   -none-    numeric
## jerr         1   -none-    numeric
```

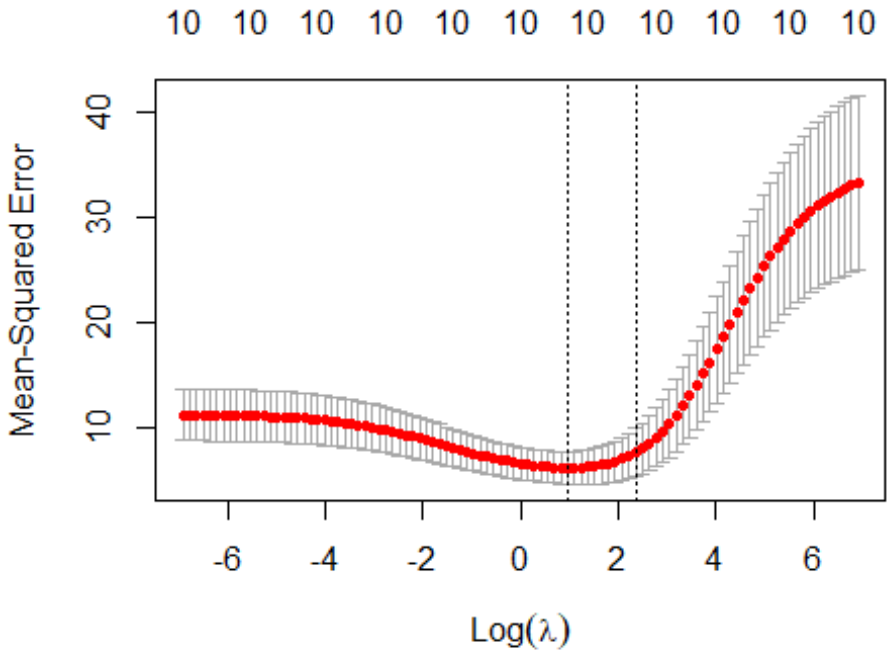
```
## offset       1   -none-    logical
## call         5   -none-    call
## nobs         1   -none-    numeric

plot(ridge_regression)
```




```
# Use cross-validation (via cv.glmnet) to determine the minimum value for
lambda - what do you obtain
cross_validation<-cv.glmnet(model.matrix(training_data$mpg~.,data =
training_data)[,- 1],training_data$mpg,alpha=0,lambda = lambda_seq,grouped =
FALSE)
cat("\n The best lambda: %s",cross_validation$lambda.min)

##
##  The best lambda: %s 2.630268

lambda_bst<-cross_validation$lambda.min
summary(cross_validation)

##             Length Class  Mode
## lambda      101    -none- numeric
## cvm         101    -none- numeric
## cvsd        101    -none- numeric
## cvup        101    -none- numeric
## cvlo        101    -none- numeric
## nzero       101    -none- numeric
## call          6    -none- call
## name          1    -none- character
```

```
## glmnet.fit  12    elnet  list
## lambda.min   1    -none- numeric
## lambda.1se   1    -none- numeric
## index        2    -none- numeric
```

```
# Plot training MSE as a function of lambda
plot(cross_validation)
```




```
# What is out-of-sample test set performance (using predict)
testing_predict<-predict(ridge_regression,s=lambda_bst,newx =
model.matrix(testing_data$mpg~.,data = testing_data)[, -1])
mean((testing_data$mpg-testing_predict)^2)
```

```
## [1] 11.50495
```

```
coef(cross_validation)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept) 20.376767880
## cyl         -0.321695828
## disp        -0.004734022
## hp          -0.010568861
## drat         0.866619485
## wt          -0.732029668
## qsec         0.091940966
## vs           0.816459140
## am1          1.039344279
```

```
## gear         0.570440867
## carb        -0.405025502

# Has ridge regression performed shrinkage, variable selection, or both?
cat("\n As we can see that new coefficients are smaller, we can say that the
ridge regression performs shrinkage.")

##
##  As we can see that new coefficients are smaller, we can say that the
ridge regression performs shrinkage.
```

swiss

Shiva Sankar Modala

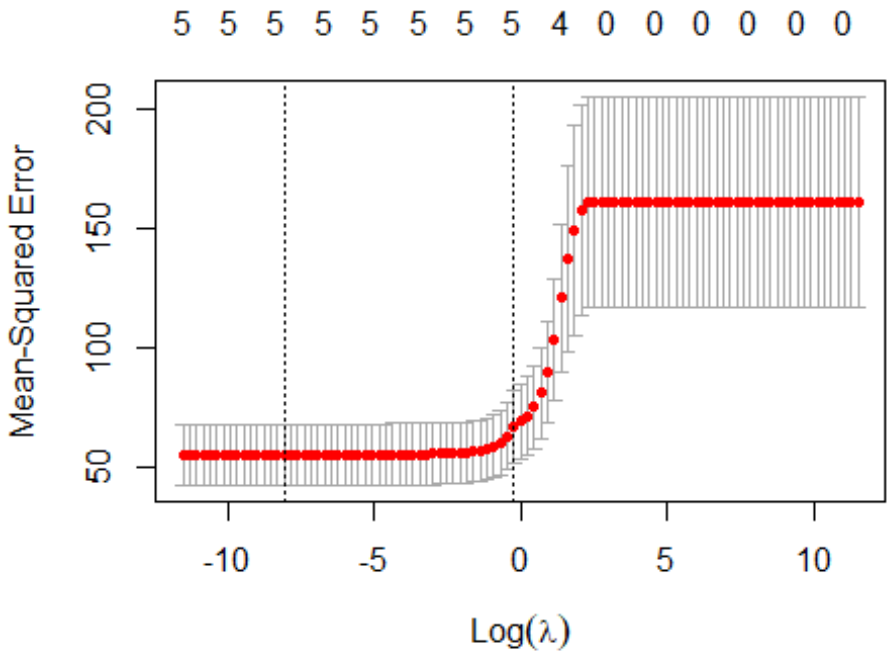
2023-03-01

```
# Load the swiss sample dataset from the built-in datasets (data(swiss))
data("swiss")

# Perform a basic 80/20 test-train split on the data
# Creating 80-20 Training Testing Split, createDataPartition() returns the
indices
sampleSize <- floor(0.8 * nrow(swiss))

# Setting the seed to make your partition reproducible
set.seed(123)
training_index <- sample(seq_len(nrow(swiss)), size = sampleSize)

# Training data
training_data = swiss[training_index, ]

# Testing data (note the minus sign)
testing_data = swiss[-training_index, ]

# Fitting linear model
# model_fit a linear model with Fertility as the target response,
linear_model_1 = lm(Fertility ~ ., training_data)

# What features are selected as relevant based on resulting t-statistics?
# Analyze the t-stat and p-values to select relevant features
summary(linear_model_1)
```

```
##
## Call:
## lm(formula = Fertility ~ ., data = training_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.8850  -3.0226   0.1069   3.3241  14.3459
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      67.96084   10.55947   6.436 3.57e-07 ***
## Agriculture      -0.22216    0.08167  -2.720 0.010593 *
## Examination      -0.22362    0.27124  -0.824 0.416003
## Education        -0.89779    0.18977  -4.731 4.64e-05 ***
## Catholic          0.13664    0.03446   3.965 0.000402 ***
## Infant.Mortality  1.13267    0.38546   2.939 0.006177 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.54 on 31 degrees of freedom
## Multiple R-squared:  0.7656, Adjusted R-squared:  0.7278
## F-statistic: 20.25 on 5 and 31 DF,  p-value: 6.076e-09

#  What are the associated coefficient values for relevant features?
# coefficient values for relevant features
linear_model_1$coefficients

##      (Intercept)      Agriculture      Examination        Education
##       67.9608389       -0.2221646       -0.2236157       -0.8977904
##         Catholic Infant.Mortality
##        0.1366393        1.1326696

# Predict out-of-sample
predict_out_of = predict(linear_model_1, testing_data, type = "response")

# Evaluate error
actual_data = testing_data[, "Fertility"]
cat("Out-of-Sample test MSE for regular linear model = ",
mean((predict_out_of - actual_data)^2))

## Out-of-Sample test MSE for regular linear model =  93.27207

library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-6

# Lambda vector of 101 elements Ranging from 0 - 100000
lambda_seq = 10^seq(5, -5, by = -.1)

# Extract x and y from training data
y = training_data$Fertility
x = model.matrix(Fertility~. ,training_data)[,-1]

#  Use cross-validation (via cv.glmnet) to determine the minimum value for
lambda - what do you obtain?
# Cross-validation to perform minimum lambda
cross_validation_fit = cv.glmnet(x, y, alpha = 1, lambda = lambda_seq)
optimal_lambda = cross_validation_fit$lambda.min
cat("Optimal Lambda = ",optimal_lambda)

## Optimal Lambda =  0.0003162278

# Perform a lasso regression using the glmnet package
# Fitting Lasso Regression with optimal lambda
model_fit = glmnet(x, y, alpha = 1, lambda = optimal_lambda)
```

```
# Plot training MSE as a function of lambda
# Plot the model
plot(cross_validation_fit)
```




```
# Coeff. of Lasso Regression
coef(model_fit)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)      67.9556030
## Agriculture      -0.2221546
## Examination      -0.2229399
## Education        -0.8981031
## Catholic          0.1366884
## Infant.Mortality  1.1324147
```

```
LassoReg_x = model.matrix(Fertility~. ,testing_data)[,-1]

# Predicting on out-of-sample test data
LassoPredict = predict(model_fit, s = optimal_lambda, newx = LassoReg_x)

# Evaluate error
actual_data = testing_data[, "Fertility"]
cat("Out-of-Sample test MSE with Lasso Regression = ", mean((LassoPredict -
actual_data)^2))
```

```
## Out-of-Sample test MSE with Lasso Regression =  93.27388
```

```
cat ("After the Lasso, we are supposed to get some coefficient perfectly
equal to zero, however we aren't getting such results, rather the
coefficients have shrunk to some extent and the out-of-sample MSE has raised
a little bit from 93.27207 to 93.27388. Lasso usually performs variable
selection, but in this case it is performing shrinkage.")
```

```
## After the Lasso, we are supposed to get some coefficient perfectly equal
to zero, however we aren't getting such results, rather the coefficients have
shrunk to some extent and the out-of-sample MSE has raised a little bit from
93.27207 to 93.27388. Lasso usually performs variable selection, but in this
case it is performing shrinkage.
```

concrete_data

Shiva Sankar Modala

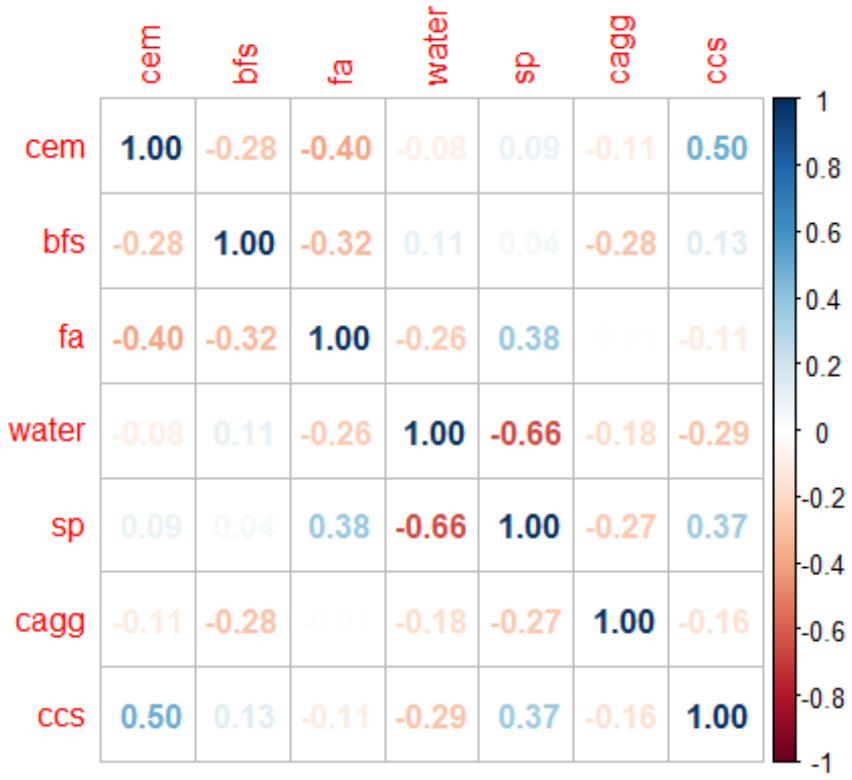
2023-03-01

```
#install.packages("tidyverse")

# readxl packages to load Excel data
#install.packages("readxl")
#install.packages("magrittr")
#install.packages("corrplot")

# Use the mgcv package to create a generalized additive model
#install.packages("mgcv")

# Visualize the regression using the visreg package,
#install.packages("visreg")

library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.0     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.1     ✓ tibble    3.1.8
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force
all conflicts to become errors

library(readxl)
library(magrittr)

##
## Attaching package: 'magrittr'
##
## The following object is masked from 'package:purrr':
##
##     set_names
##
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(corrplot)

## corrplot 0.92 loaded

library(mgcv)

## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## This is mgcv 1.8-41. For overview type 'help("mgcv-package")'.

library(visreg)

# Load the Concrete Compressive Strength sample dataset
concrete_data <- read_excel("C:/Users/shiva/OneDrive/Desktop/dpa
Assignments/Assignment3/Concrete_Data.xls")
summary(concrete_data)

##  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

colnames(concrete_data) = c("cem", "bfs", "fa", "water", "sp", "cagg",
"fagg", "age", "ccs")
column_names = c("cem", "bfs", "fa", "water", "sp", "cagg", "ccs")
concrete_data = concrete_data[column_names]
summary(concrete_data)

##       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
```

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




```
# gam function) to predict the Concrete Compressive Strength
dataModel1 <- gam(ccs ~ cem + bfs + fa + water + sp + cagg ,
data=concrete_data)
summary(dataModel1)

##
## 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
```

```
# compare the R2 value for a GAM with linear terms as well as smoothed terms
cat("The corrected R-squared + shows that a sizable portion of the variation
is present, and it appears that we have statistical effects for CEM and BFS
but not for CAGG.")
```

```
## The corrected R-squared + shows that a sizable portion of the variation is
present, and it appears that we have statistical effects for CEM and BFS but
not for CAGG.
```

```
# Use the s() function to apply smoothing using the default bs of tp).
dataModel2 <- gam(ccs ~ s(cem) + s(bfs) + s(fa) + s(water) + s(sp) + s(cagg)
, data=concrete_data)
summary(dataModel2)
```

```
##
## 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
```

```
cat("We should also remark that this model, with an adjusted R-squared
of.531, explains a large portion of the variance in CCS. In summary, it
appears that the CEM and CCS are connected.")
```

```
## We should also remark that this model, with an adjusted R-squared of.531,
explains a large portion of the variance in CCS. In summary, it appears that
the CEM and CCS are connected.
```

```r
#  showing the fit as a function of each predictor
dataModel1.sse <- sum(fitted(dataModel1)-concrete_data$ccs)^2
dataModel1.ssr <- sum(fitted(dataModel1) -mean(concrete_data$ccs))^2
dataModel1.sst = dataModel1.sse + dataModel1.ssr
Rsquared=1-(dataModel1.sse/dataModel1.sst)
cat(Rsquared)
```

```
## 0.4967177
```

```r
dataModel2.sse <- sum(fitted(dataModel2)-concrete_data$ccs)^2
dataModel2.ssr <- sum(fitted(dataModel2) -mean(concrete_data$ccs))^2
dataModel2.sst = dataModel2.sse + dataModel2.ssr
Rsquared_sm=1-(dataModel2.sse/dataModel2.sst)
cat(Rsquared_sm)
```

```
## 0.5000744
```

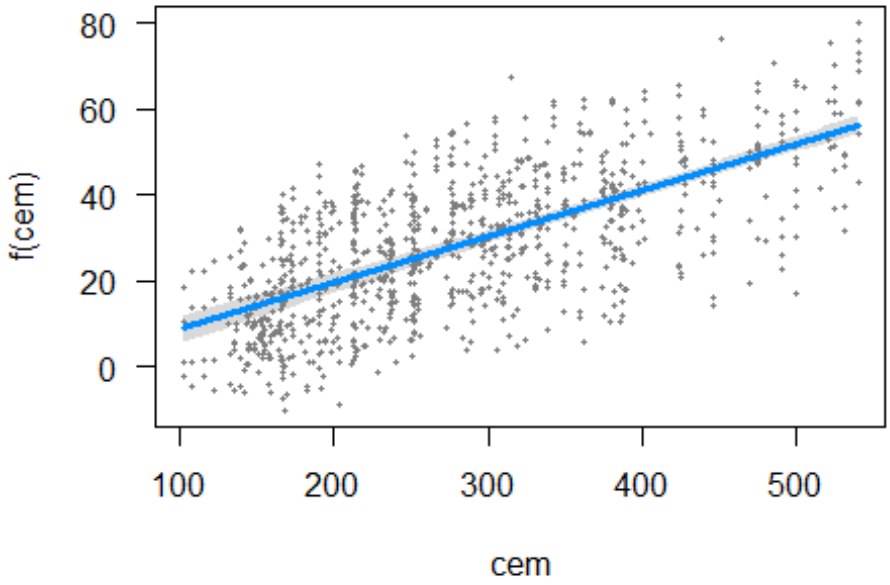
```r
anova(dataModel1, dataModel2, test="Chisq")
```

```
## 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
```

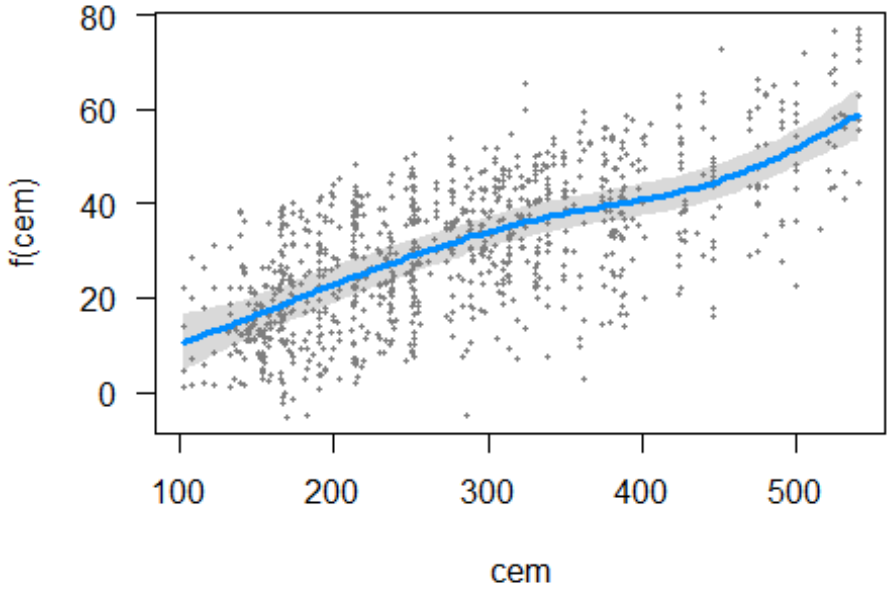
```r
cat("Although we couldn't have known as much already, new statistical
evidence suggests that adding in the variables' nonlinear correlations
enhances the model.")
```

```
## Although we couldn't have known as much already, new statistical evidence
suggests that adding in the variables' nonlinear correlations enhances the
model.
```

```r
visreg(dataModel1,'cem')
```



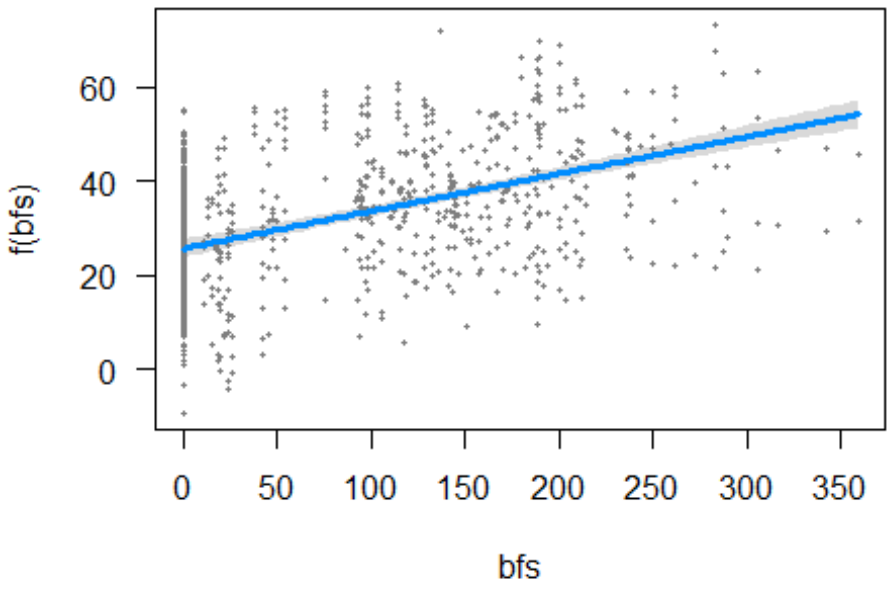

```
visreg(dataModel2,'cem')
```



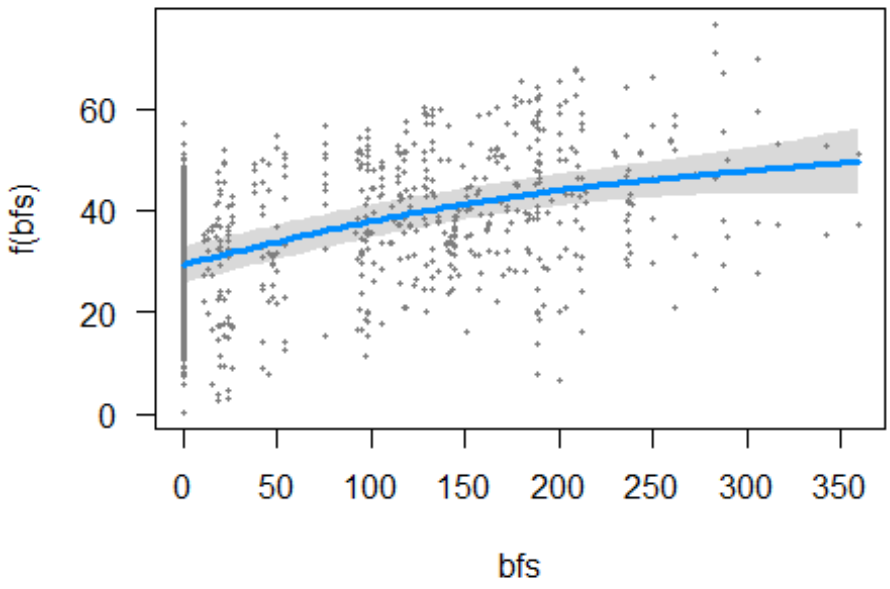
```
cat("The end result, with all other model variables maintained constant,
    is a plot showing how the expected value of the CCS changes as a function
of x (CEM).
    It contains the following information: (1) the expected value (blue
line),
    (2) a confidence interval for the expected value (gray band), and
    (3) partial residuals (dark gray dots).")
```

```
## The end result, with all other model variables maintained constant,
##     is a plot showing how the expected value of the CCS changes as a
function of x (CEM).
##     It contains the following information: (1) the expected value (blue
line),
##     (2) a confidence interval for the expected value (gray band), and
##     (3) partial residuals (dark gray dots).
```

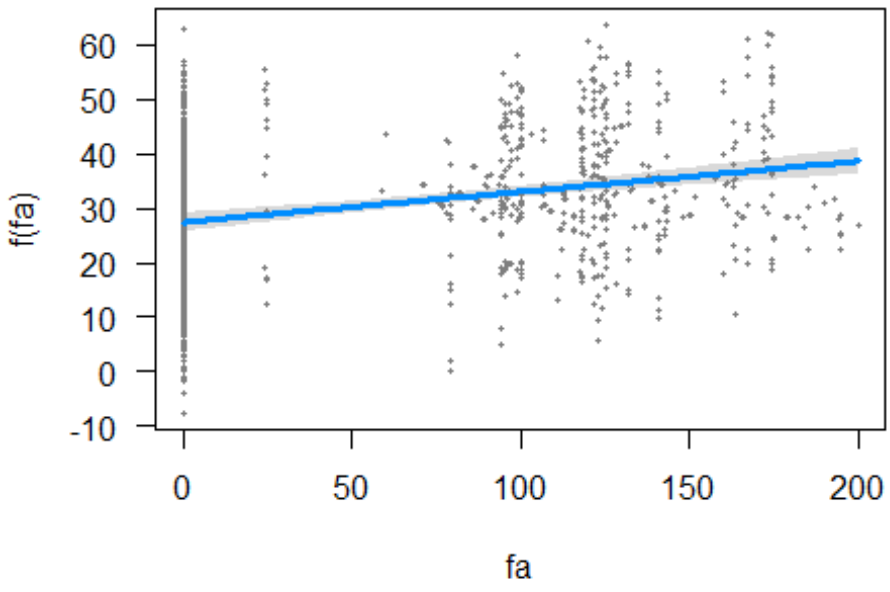
```
# Visualizing the feature with the function of their feature
visreg(dataModel1,'bfs')
```



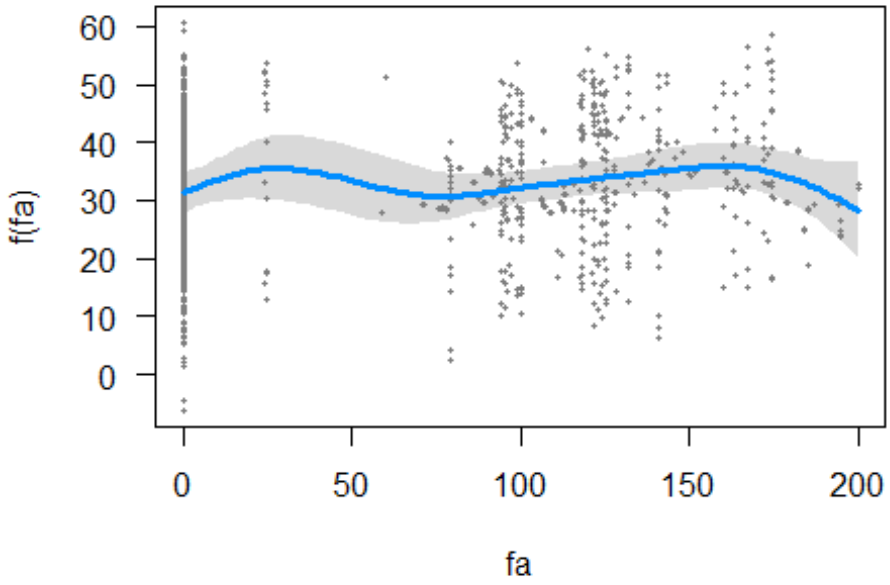

```
visreg(dataModel2,'bfs')
```



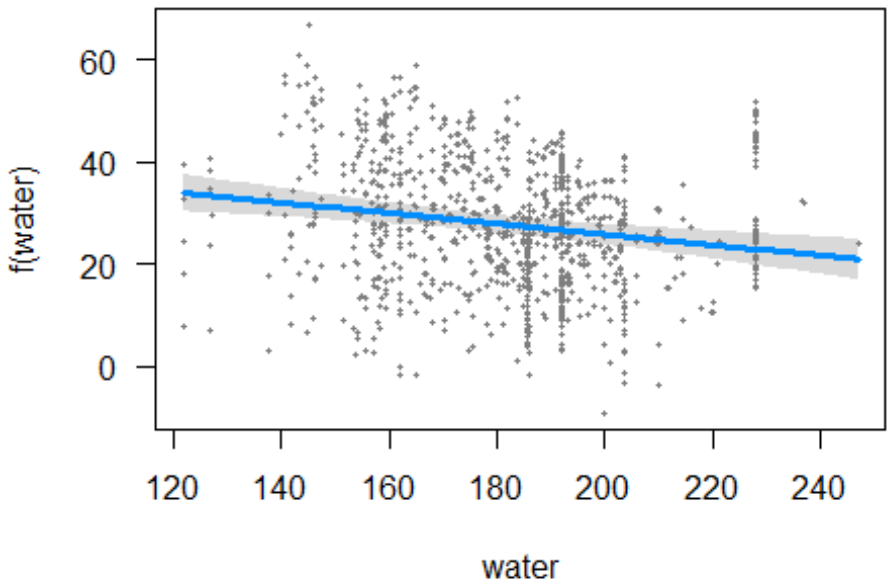

```
visreg(dataModel1,'fa')
```



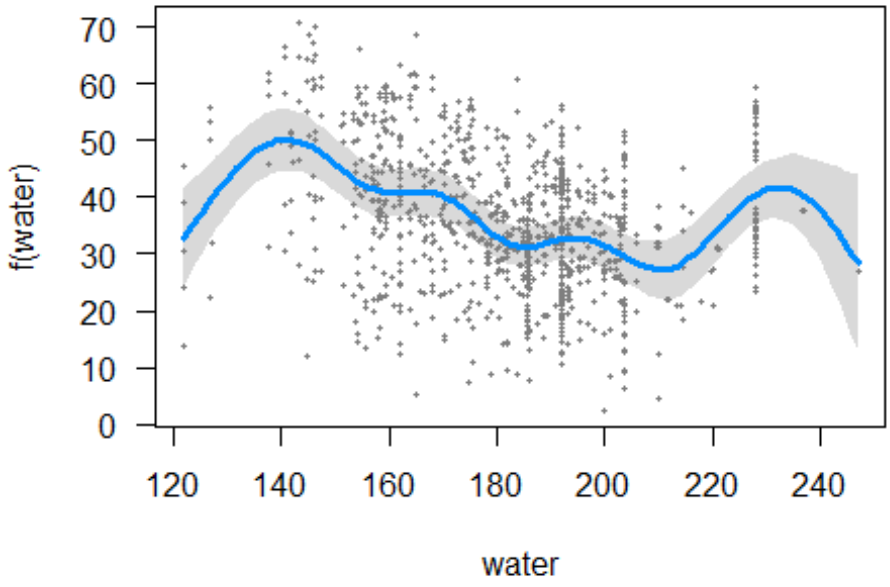

```
visreg(dataModel2,'fa')
```



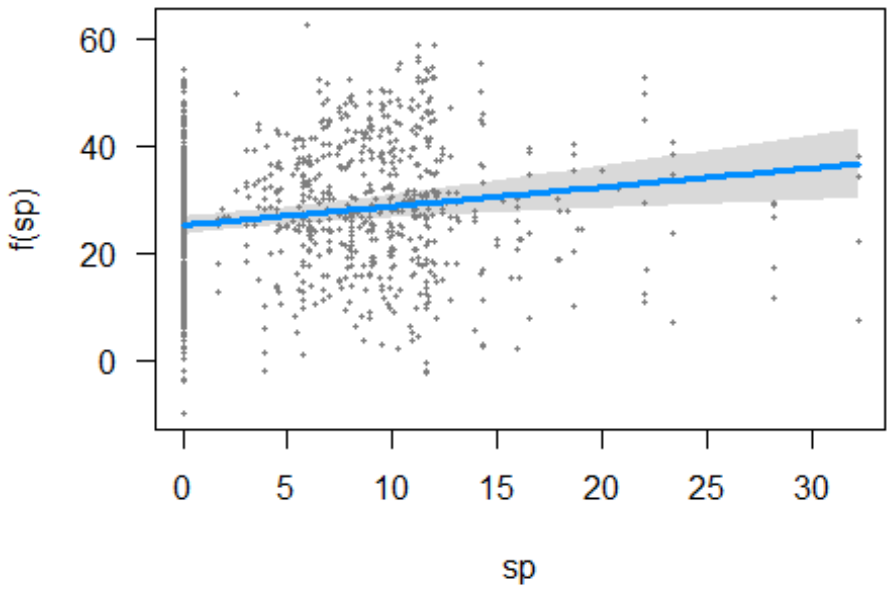

```
visreg(dataModel1,'water')
```



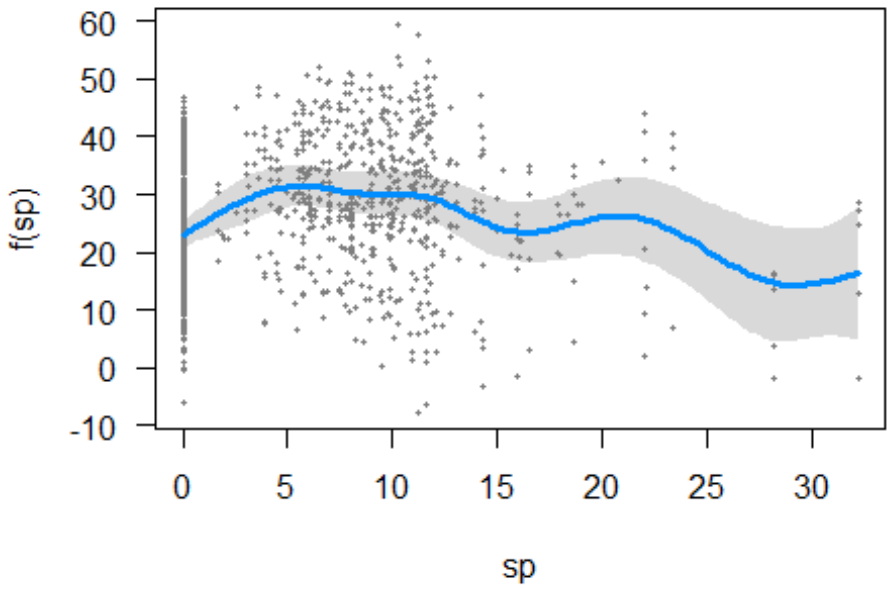

```
visreg(dataModel2,'water')
```



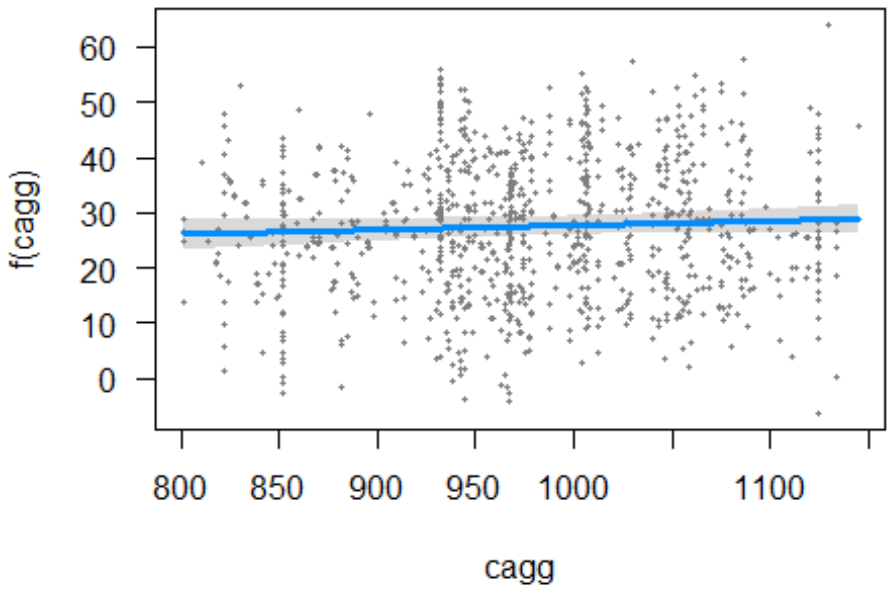

```
visreg(dataModel1,'sp')
```



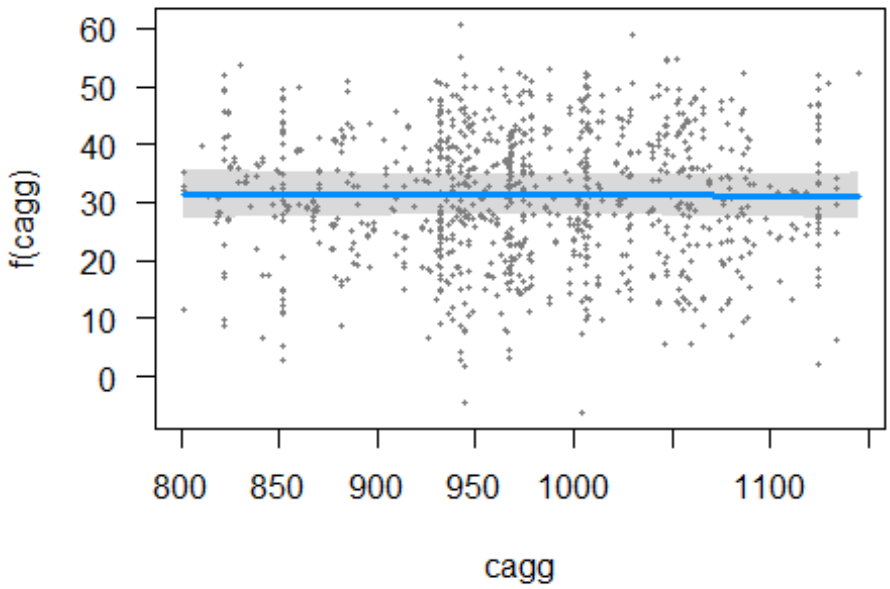

```
visreg(dataModel2,'sp')
```




```
visreg(dataModel1,'cagg')
```




```
visreg(dataModel2,'cagg')
```




```
cat("We can see from the CEM graph that the confidence interval has a higher
value after adding the smoothing function than the model had without it.
Using the smoothing function improves the confidence interval.")
```

```
## We can see from the CEM graph that the confidence interval has a higher
value after adding the smoothing function than the model had without it.
Using the smoothing function improves the confidence interval.
```