Problem 1

```
#Importing the dataset
car<-data.frame(mtcars)
```

```
#Check Structure of dataset
str(car)
```

```
## '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 ...
```

```
library(caret)
```

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

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

```
set.seed(200)
partition <- createDataPartition(car$am,times=1,p=0.8,list = F)
train <- car[partition,]
test <- car[-partition,]
```

```
#fiitting a linear model
model <- lm(mpg~.,data=train)

#MSE on test set
mean((predict(model,test)-test$mpg)^2)
```

```
## [1] 10.71549
```

```
summary(model)
```

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

```
##  (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
```

Only Attribute “wt” is relevant

Ridge Regression

```
# Loading the Library
library(glmnet)
```

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

```
## Loaded glmnet 4.1-2
```

```
# Getting the independent variable
x <- model.matrix(mpg~.,train)[,-1]


# Getting the dependent variable
y <- train$mpg
```

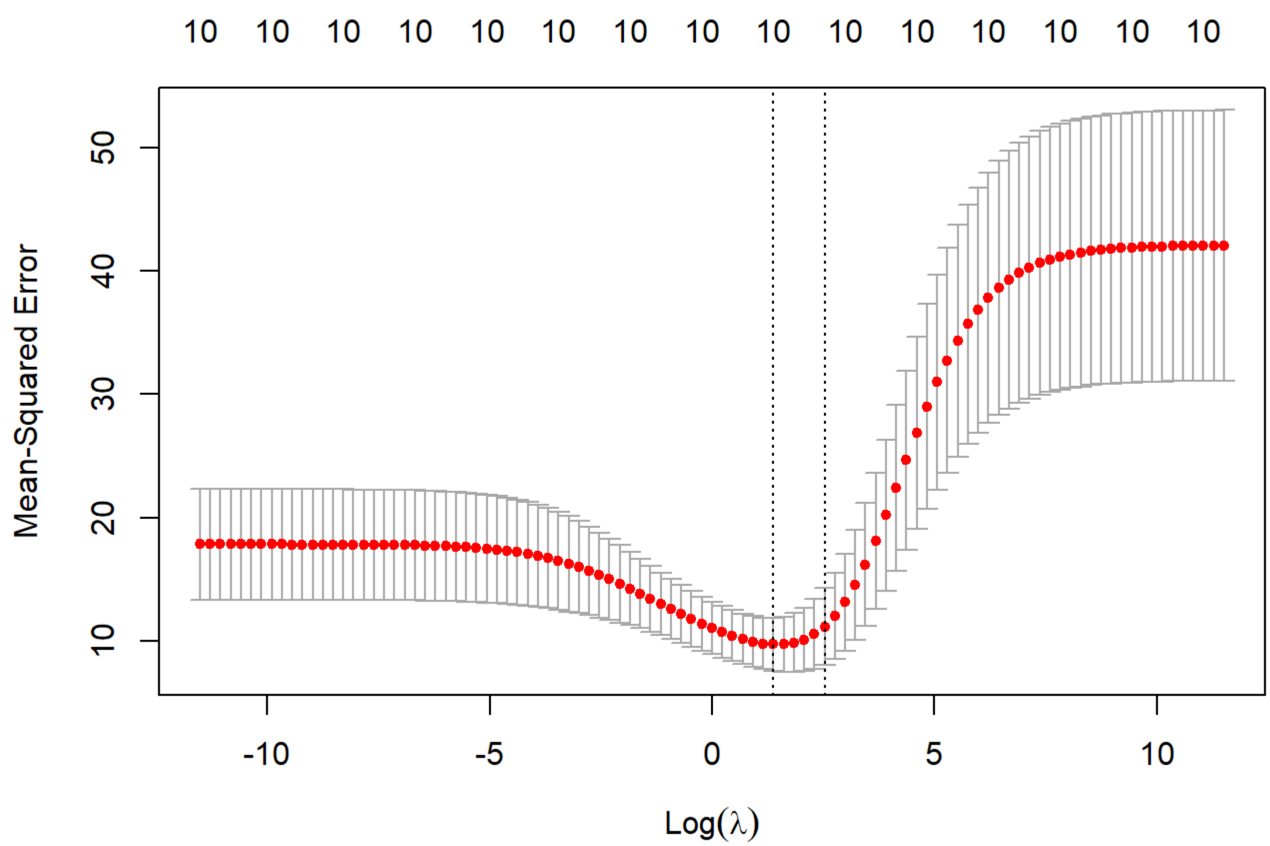
Cross Validation using GLMNET

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

# Using cross validation glmnet
ridge_cv <- cv.glmnet(x, y, alpha = 0,lambda = lambda_seq)
```

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

```
plot(ridge_cv)
```




```
#Best lambda value
best_lambda <- ridge_cv$lambda.min
best_lambda
```

```
## [1] 3.981072
```

```
# Building the Ridge Regression Model using GLMNET
fit <- glmnet(x, y, alpha = 0, lambda  = best_lambda)
```

```
summary(fit)
```

```
##           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(ridge_cv,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
```

```
# Test Dataset
x1 =  model.matrix(mpg~.,test)[,-1]
model_predict <- predict(fit,s =,newx = x1, type = "response")

#MSE on test data
mean((model_predict-test$mpg)^2)
```

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

We can see that MSE on test data will decreases from 10.71 to 1.18 by performing Ridge Regression. As we can see after Ridge Regression the coefficients have shrunk and are more close to zero but none of them are perfect zero. Hence Ridge Regression has performed shrinkage.

Problem 2

```
library(ggplot2)
library(lattice)
library(caret)
#Importing the dataset
data <- data.frame(swiss)
```

```
#80-20 split using createDataPartition
set.seed(150)
partition <- createDataPartition(data$Fertility,p=0.8,list = F)
train <- data[partition,]
test <- data[-partition,]
```

```
#fitting a linear fit
model <- lm(Fertility~.,train)

summary(model)
```

```
##
## Call:
## lm(formula = Fertility ~ ., data = train)
##
## 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((test$Fertility-predict(model,test))^2)
```

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

Lasso Regression

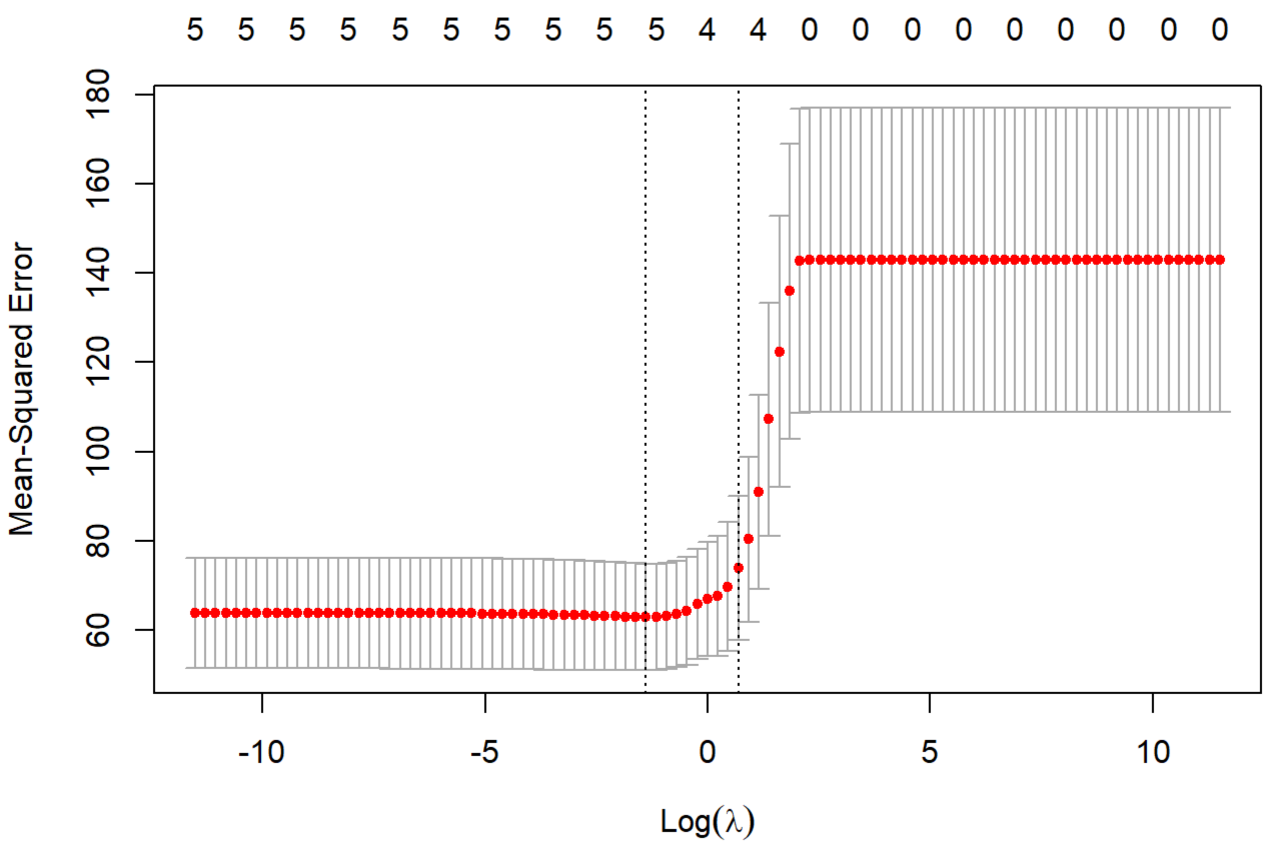
```
# Loaging the library
library(Matrix)
library(foreach)
library(glmnet)
# Getting the independent variable
x <- model.matrix(Fertility~.,train)[,-1]

# Getting the dependent variable
y <- train$Fertility
```

Cross Validation Lasso GLMNET

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

# Using cross validation glmnet
lasso_cv <- cv.glmnet(x, y, alpha = 1,lambda = lambda_seq)
plot(lasso_cv)
```



```
#Best lambda value
best_lambda <- lasso_cv$lambda.min
best_lambda
```

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

```
# Using glmnet function to build the ridge regression model
fit <- glmnet(x, y, alpha = 1, lambda  = best_lambda)

# Checking the model
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
```

```
#for testdata
x2 =  model.matrix(Fertility~.,test)[,-1]
model_predict <- predict(fit,s =,newx = x2, type = "response")
```

```
#MSE on test data
mean((model_predict-test$Fertility)^2)
```

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

```
#coefficients
coef(model)
```

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

```
coef(lasso_cv)
```

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

Compared to Linear fit Lasso Regularization has shrinked the coefficients and two of them are shrinked to zero.

Problem 3

```
concrete <- read.csv("D:\\Temp\\Concrete_Data.csv")
summary(concrete)
```

```
##    ï..Cement     Blast.Furnace.Slag    Fly.Ash           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.:118.30   3rd Qu.:192.0
##  Max.   :540.0   Max.   :359.4      Max.   :200.10   Max.   :247.0
##  Superplasticizer Course.Aggregate Fine.Aggregate       Age
##  Min.   : 0.000   Min.   : 801.0   Min.   :594.0   Min.   :  1.00
##  1st Qu.: 0.000   1st Qu.: 932.0   1st Qu.:731.0   1st Qu.:  7.00
##  Median : 6.400   Median : 968.0   Median :779.5   Median : 28.00
##  Mean   : 6.205   Mean   : 972.9   Mean   :773.6   Mean   : 45.66
##  3rd Qu.:10.200   3rd Qu.:1029.4   3rd Qu.:824.0   3rd Qu.: 56.00
##  Max.   :32.200   Max.   :1145.0   Max.   :992.6   Max.   :365.00
##     Strength
##  Min.   : 2.33
##  1st Qu.:23.71
##  Median :34.45
##  Mean   :35.82
##  3rd Qu.:46.13
##  Max.   :82.60
```

Changing the Column names Taking Columns C1-C6

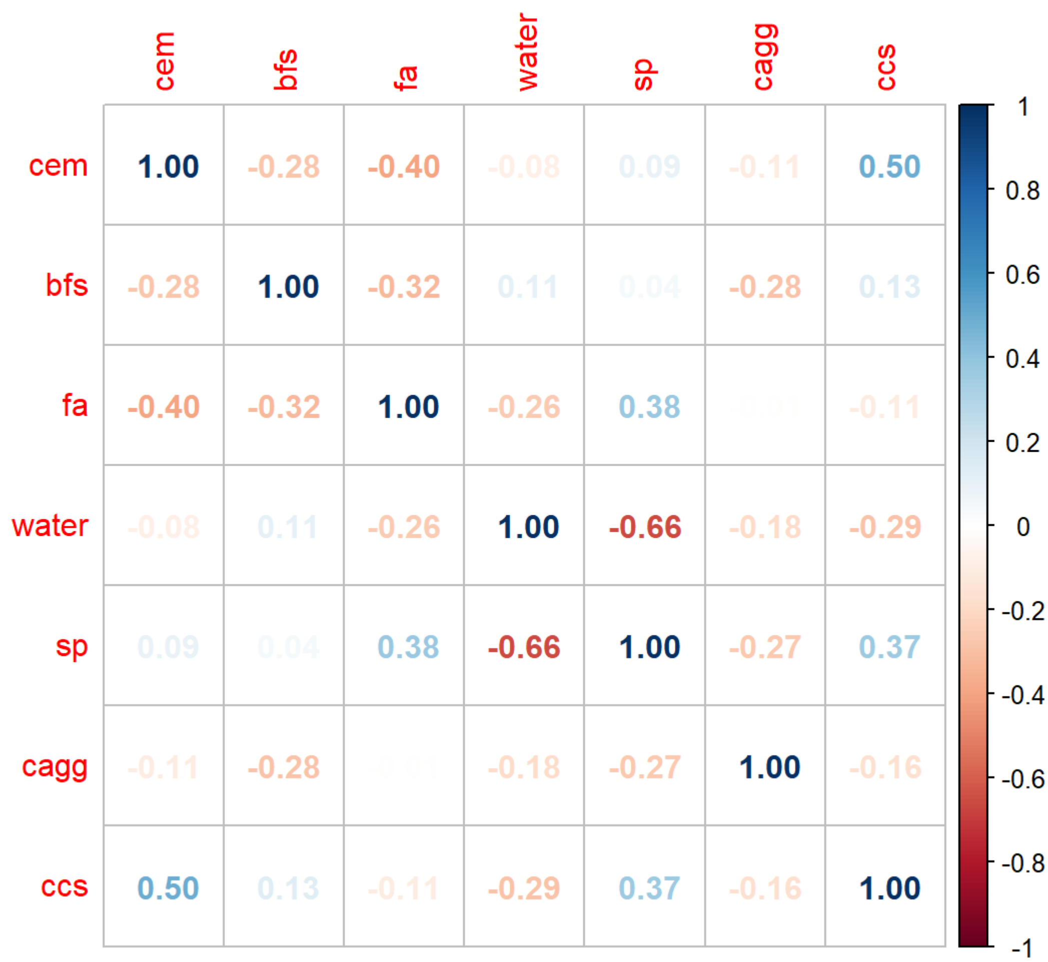
```
colnames(concrete) = c("cem", "bfs", "fa", "water", "sp", "cagg", "fagg", "age", "ccs")
keeps = c("cem", "bfs", "fa", "water", "sp", "cagg", "ccs")
concrete = concrete[keeps]
summary(concrete)
```

```
##       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.:118.30   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.   : 2.33  
##  1st Qu.: 0.000   1st Qu.: 932.0   1st Qu.:23.71  
##  Median : 6.400   Median : 968.0   Median :34.45  
##  Mean   : 6.205   Mean   : 972.9   Mean   :35.82  
##  3rd Qu.:10.200   3rd Qu.:1029.4   3rd Qu.:46.13  
##  Max.   :32.200   Max.   :1145.0   Max.   :82.60
```

```
library(corrplot)
```

```
## corrplot 0.90 loaded
```

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




```
library(mgcv)
```

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

```
## This is mgcv 1.8-36. For overview type 'help("mgcv-package")'.
```

```
model1 <- gam(ccs ~ cem + bfs + fa + water + sp + cagg , data=concrete)
summary(model1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## ccs ~ cem + bfs + fa + water + sp + cagg
##
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.231362  10.509324   0.498 0.618744
## cem          0.108250   0.005213  20.764  < 2e-16 ***
## bfs          0.079345   0.006192  12.814  < 2e-16 ***
## fa           0.055881   0.009285   6.018 2.46e-09 ***
## water       -0.103562   0.027795  -3.726 0.000205 ***
## sp           0.357695   0.110211   3.246 0.001210 **
## cagg         0.008061   0.006271   1.285 0.198930
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.445   Deviance explained = 44.9%
## GCV = 155.82  Scale est. = 154.76    n = 1030
```

It appears we have statistical effects for CEM, BFS, but not for CAGG and the adjusted R-squared suggests a notable amount of the variance.

Using Smoothing Function

```
model2 <- gam(ccs ~ s(cem) + s(bfs) + s(fa) + s(water) + s(sp) + s(cagg) , data=concrete)
summary(model2)
```

```
##
## 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)  35.8180     0.3564   100.5   <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)   4.448  5.494 69.935  < 2e-16 ***
## s(bfs)   2.088  2.578 47.990  < 2e-16 ***
## s(fa)    5.592  6.646  1.954   0.0686 .
## s(water) 8.567  8.936 13.394  < 2e-16 ***
## s(sp)    7.200  8.174  5.383 1.56e-06 ***
## s(cagg)  1.000  1.000  0.035   0.8512
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.531   Deviance explained = 54.4%
## GCV = 134.76  Scale est. = 130.85    n = 1030
```

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

```
model1.sse <- sum(fitted(model1)-concrete$ccs)^2
model1.ssr <- sum(fitted(model1) -mean(concrete$ccs))^2
model1.sst = model1.sse + model1.ssr

rsqr_main=1-(model1.sse/model1.sst)
print(rsqr_main)
```

```
## [1] 0.4994171
```

```
model2.sse <- sum(fitted(model2)-concrete$ccs)^2
model2.ssr <- sum(fitted(model2) -mean(concrete$ccs))^2
model2.sst = model2.sse + model2.ssr

rsqr_sm=1-(model2.sse/model2.sst)
print(rsqr_sm)
```

```
## [1] 0.5022629
```

Comparison of Model

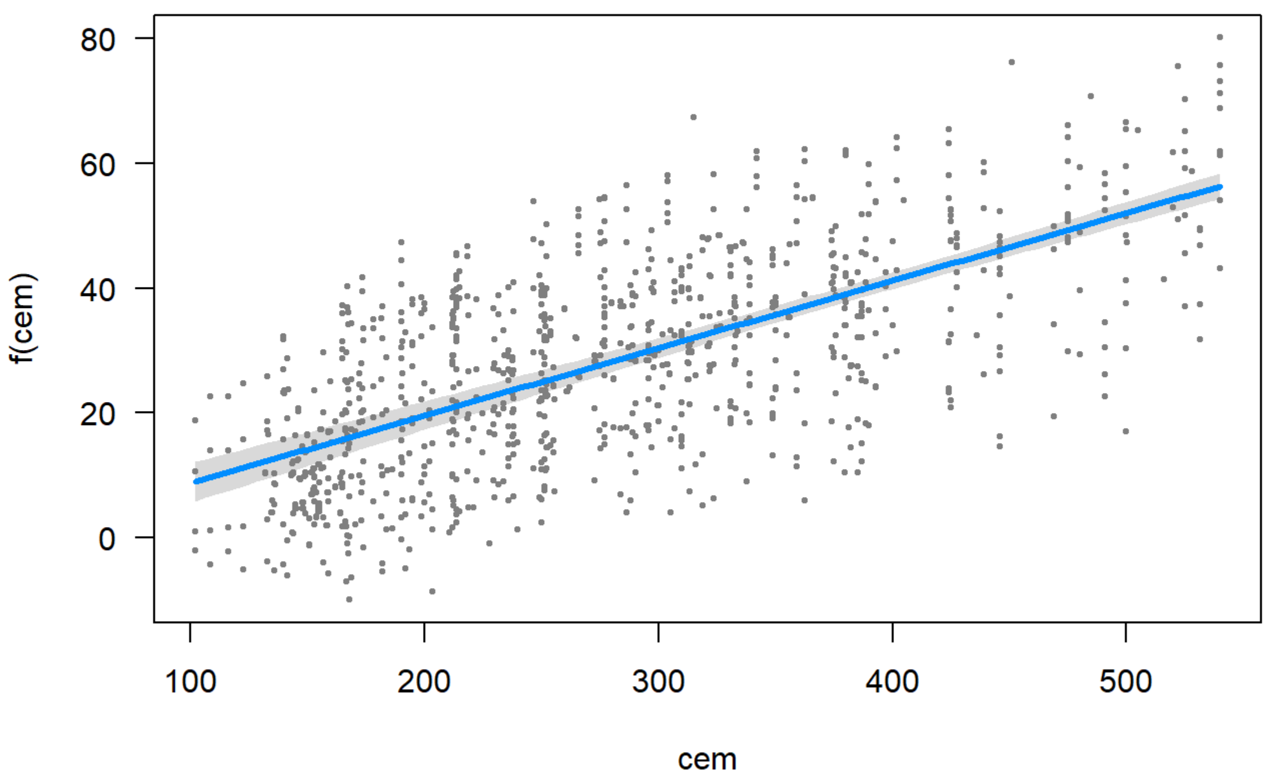
```
anova(model1, model2, test="Chisq")
```

```
## 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     158316
## 2    996.17     130865 26.828    27451 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

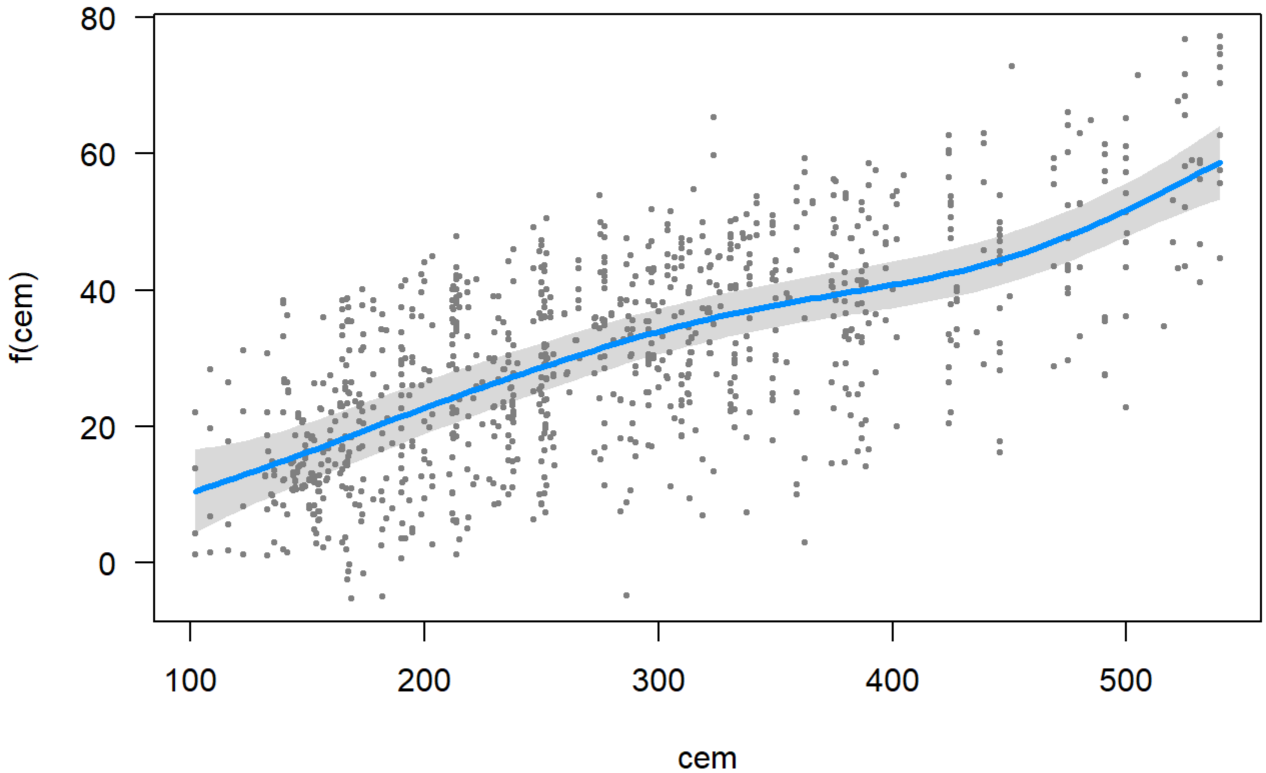
We couldn’t have assumed as such already, but now we have additional statistical evidence to suggest that incorporating nonlinear relationships of the covariates improves the model.

Visualizing with Visreg Library

```
library(visreg)
visreg(model1,'cem')
```

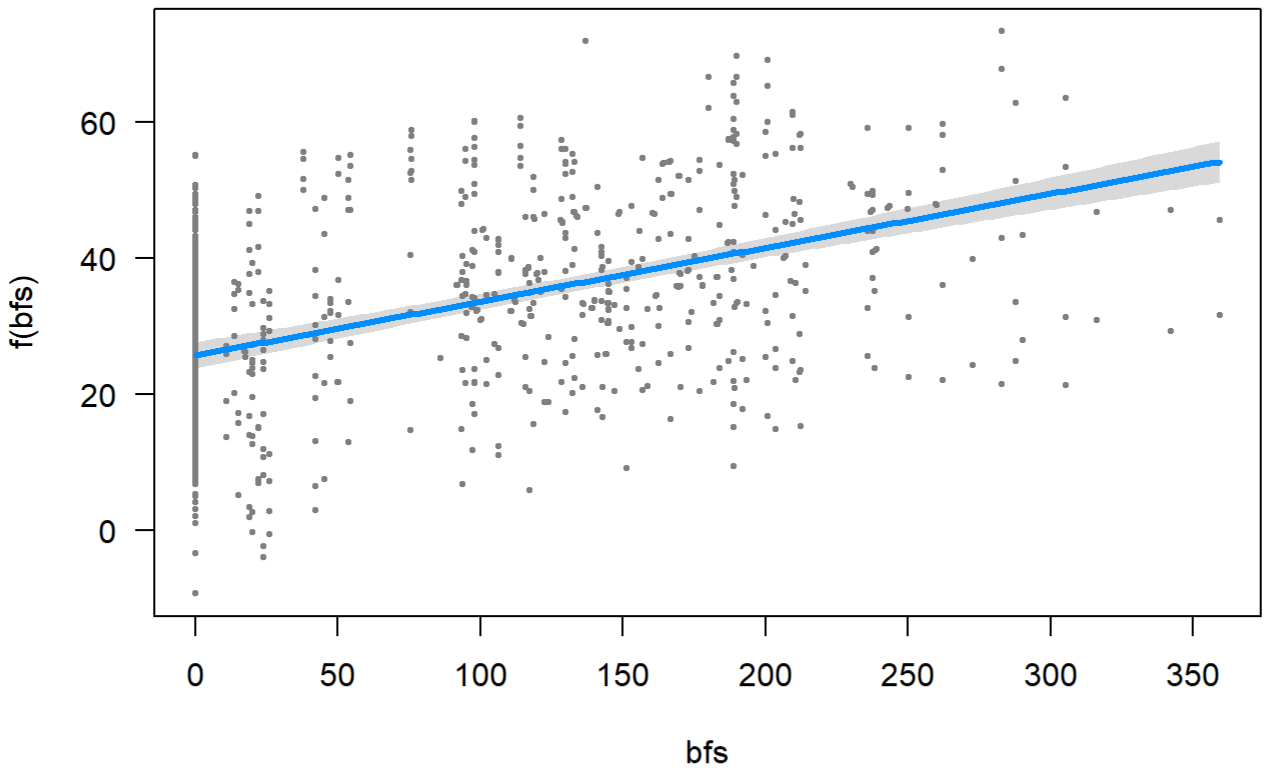



```
visreg(model2,'cem')
```

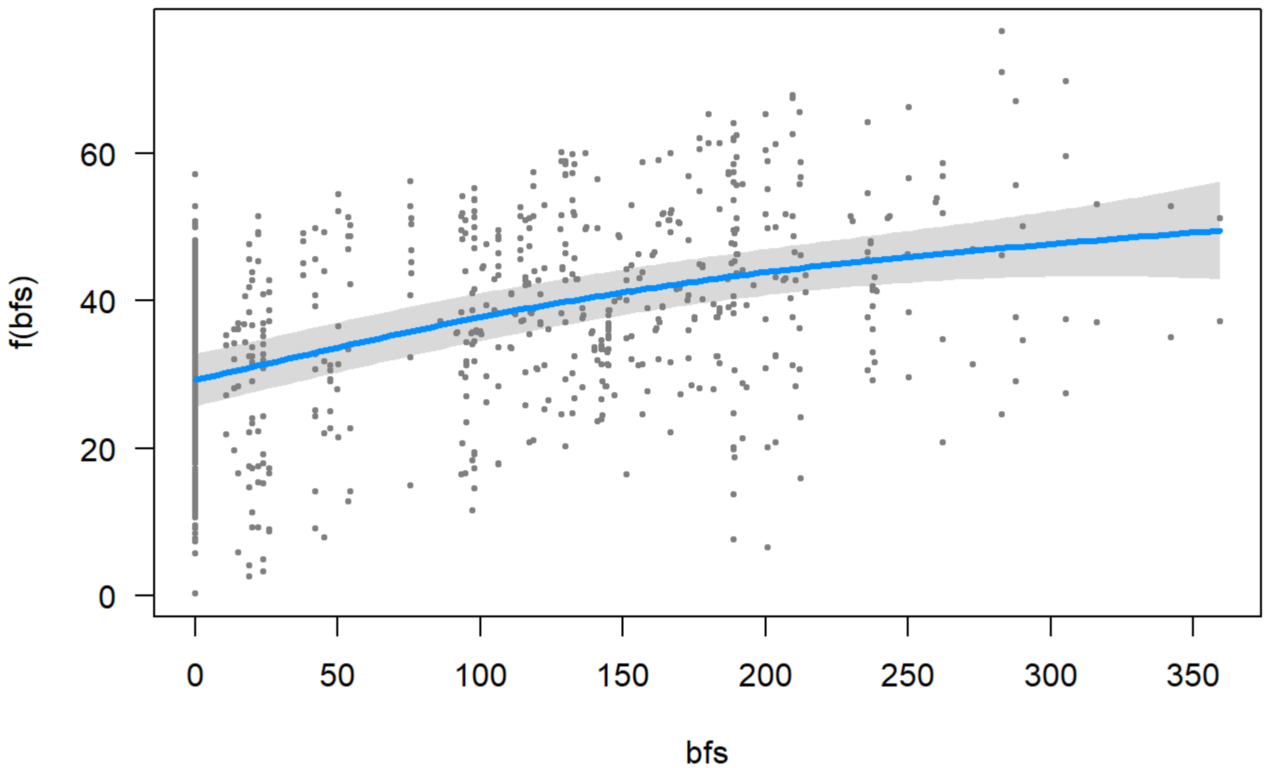


The result is a plot of how the expected value of the CCS changes as a function of x (CEM), with all other variables in the model held fixed. It includes (1) the expected value (blue line) (2) a confidence interval for the expected value (gray band) (3) partial residuals (dark gray dots).

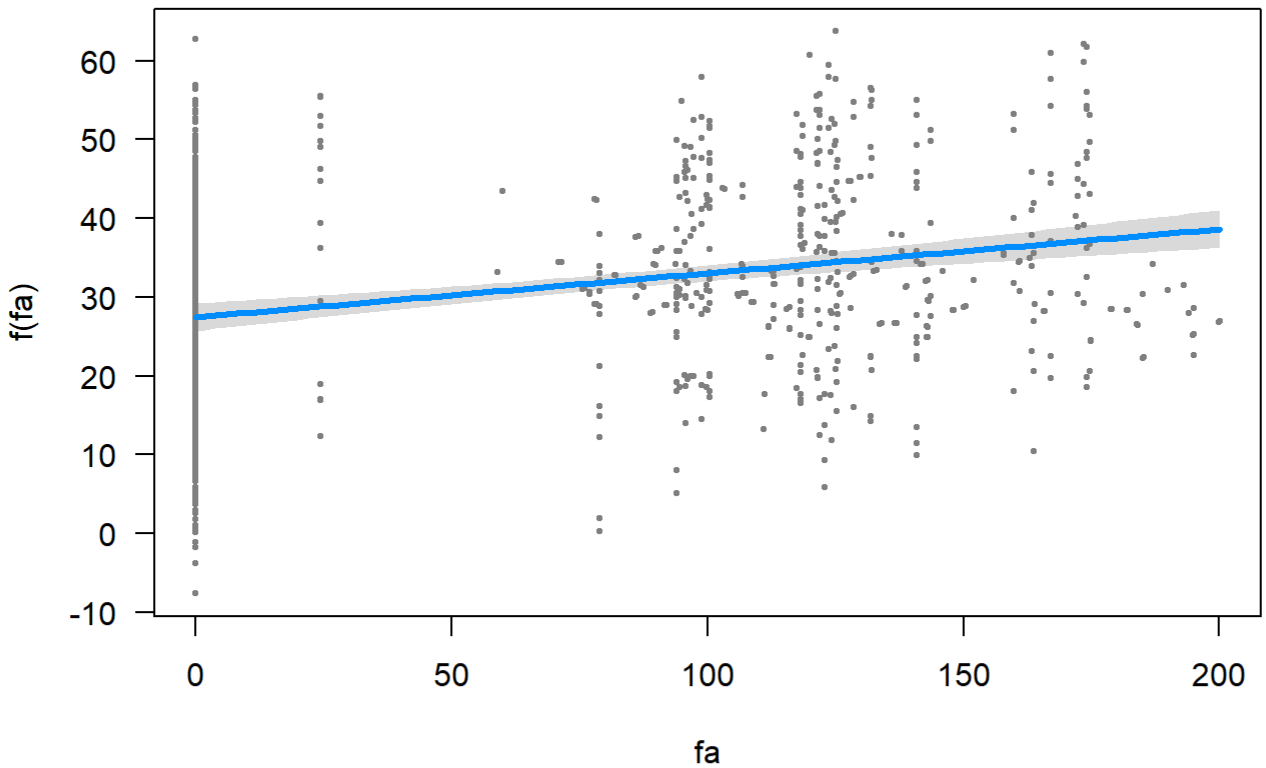
```
visreg(model1,'bfs')
```



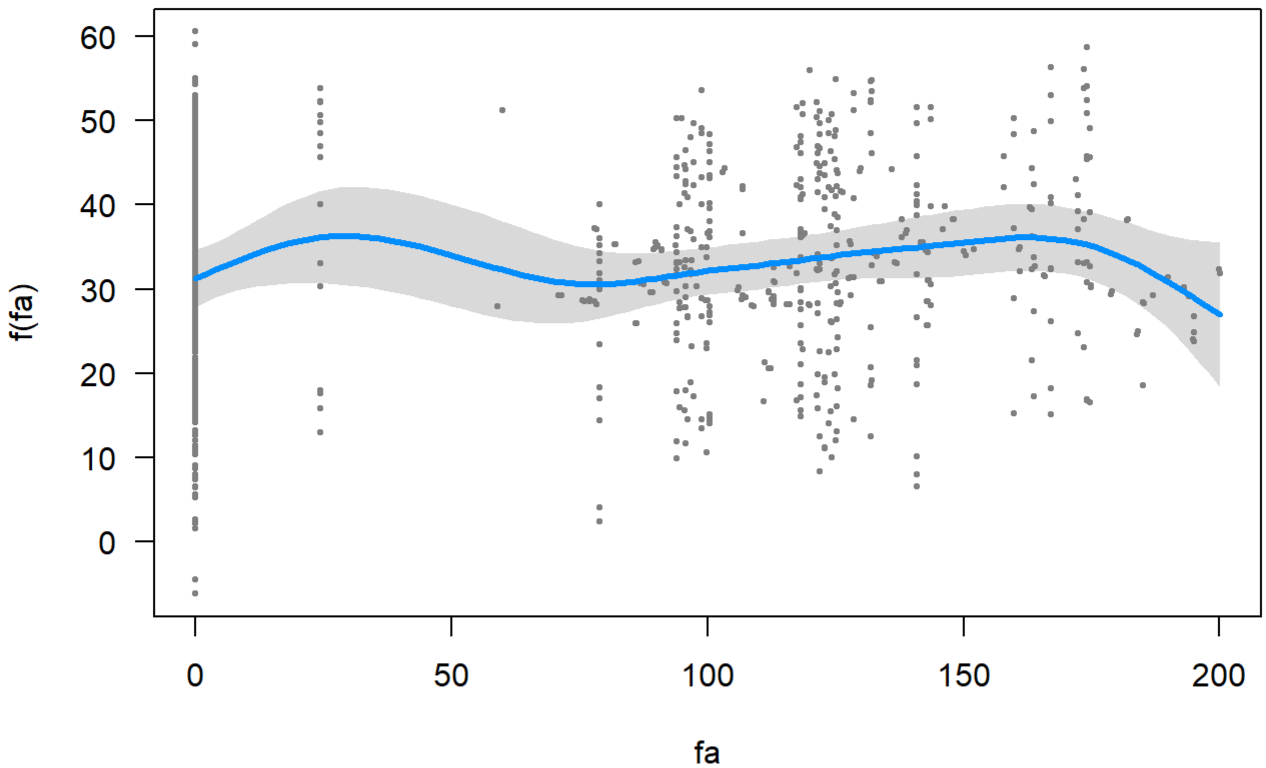

```
visreg(model2,'bfs')
```



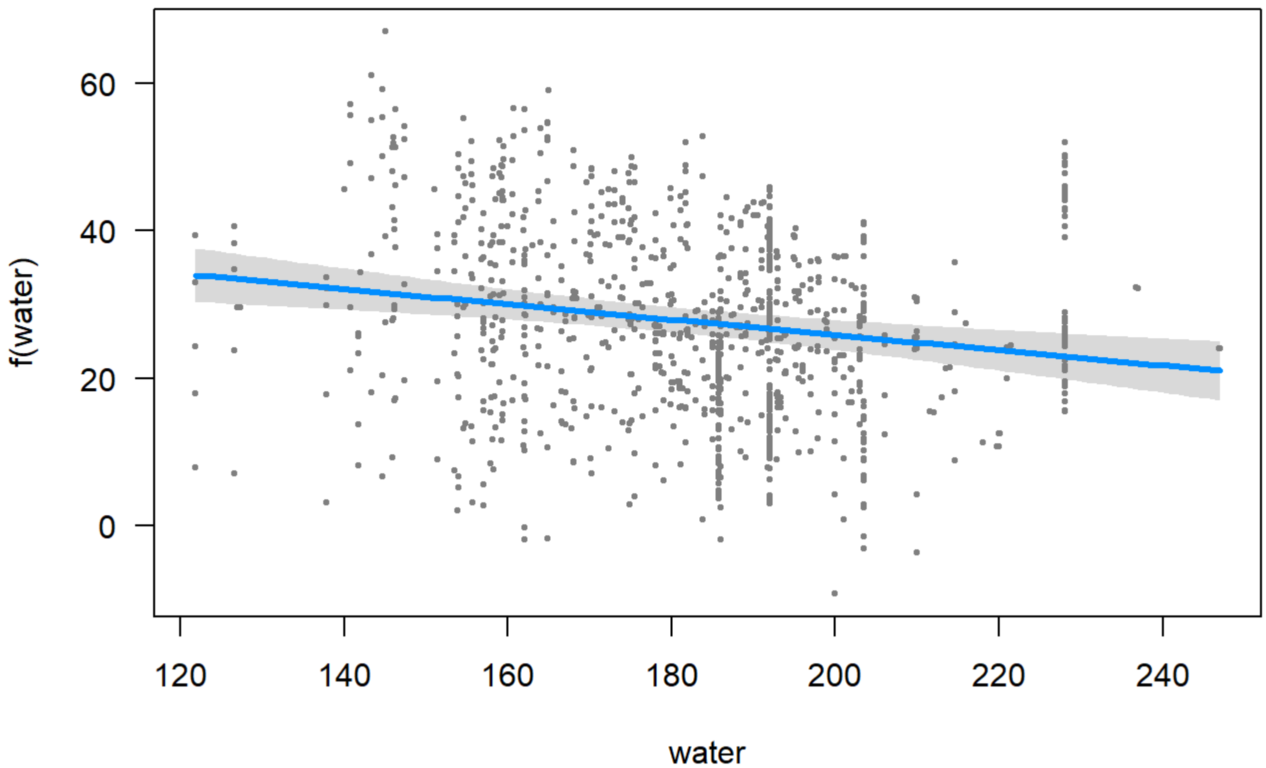

```
visreg(model1,'fa')
```



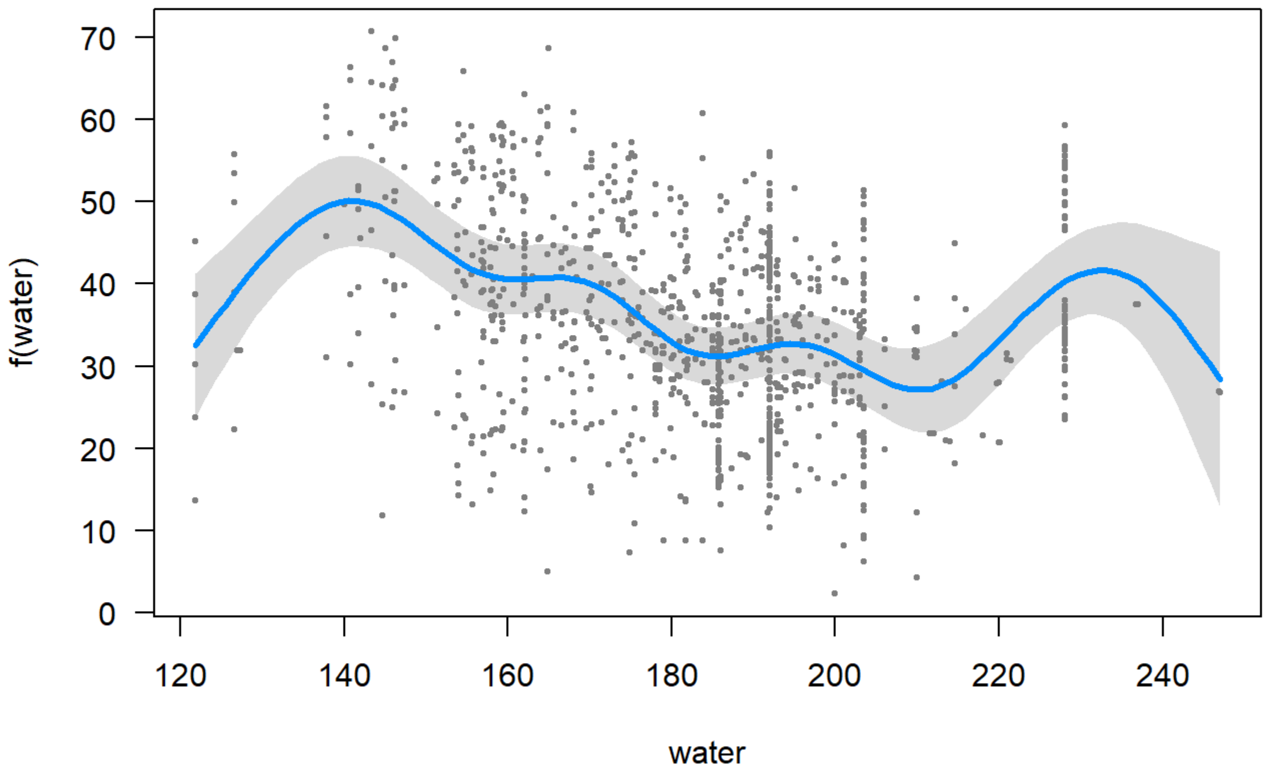

```
visreg(model2,'fa')
```



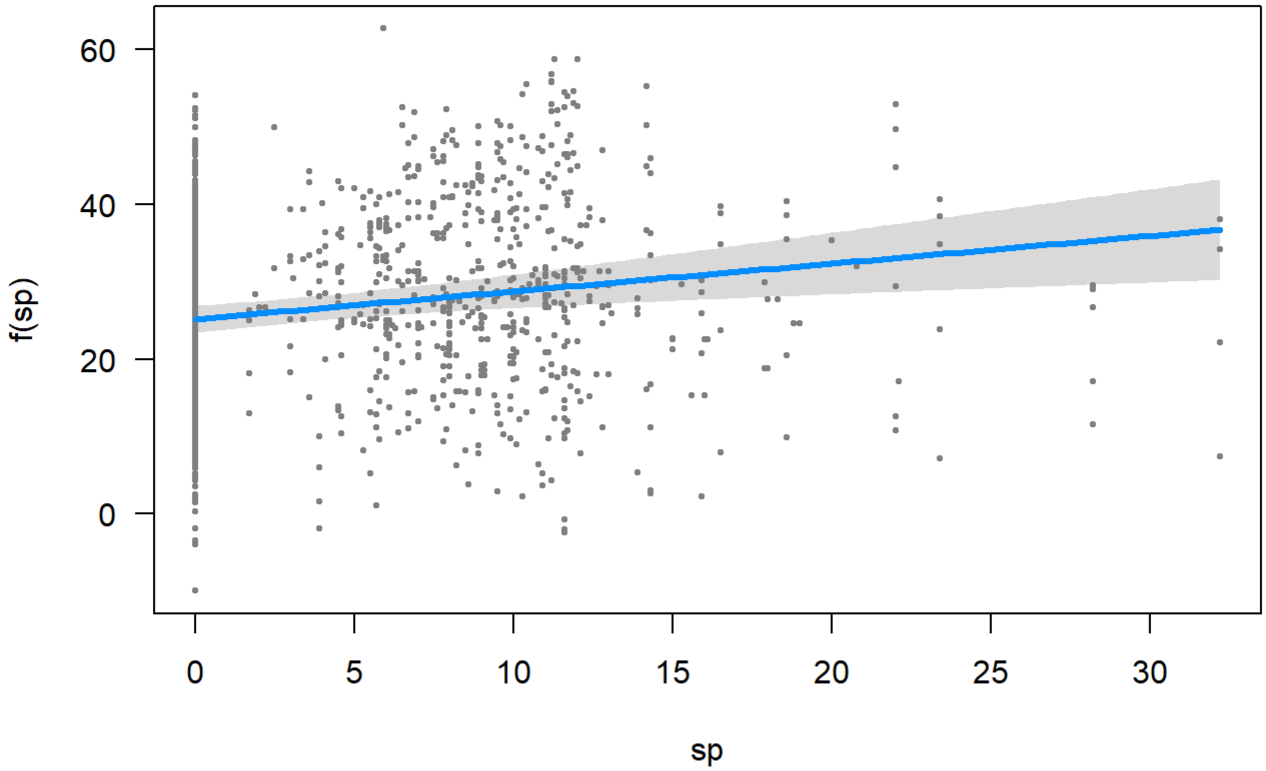

```
visreg(model1,'water')
```



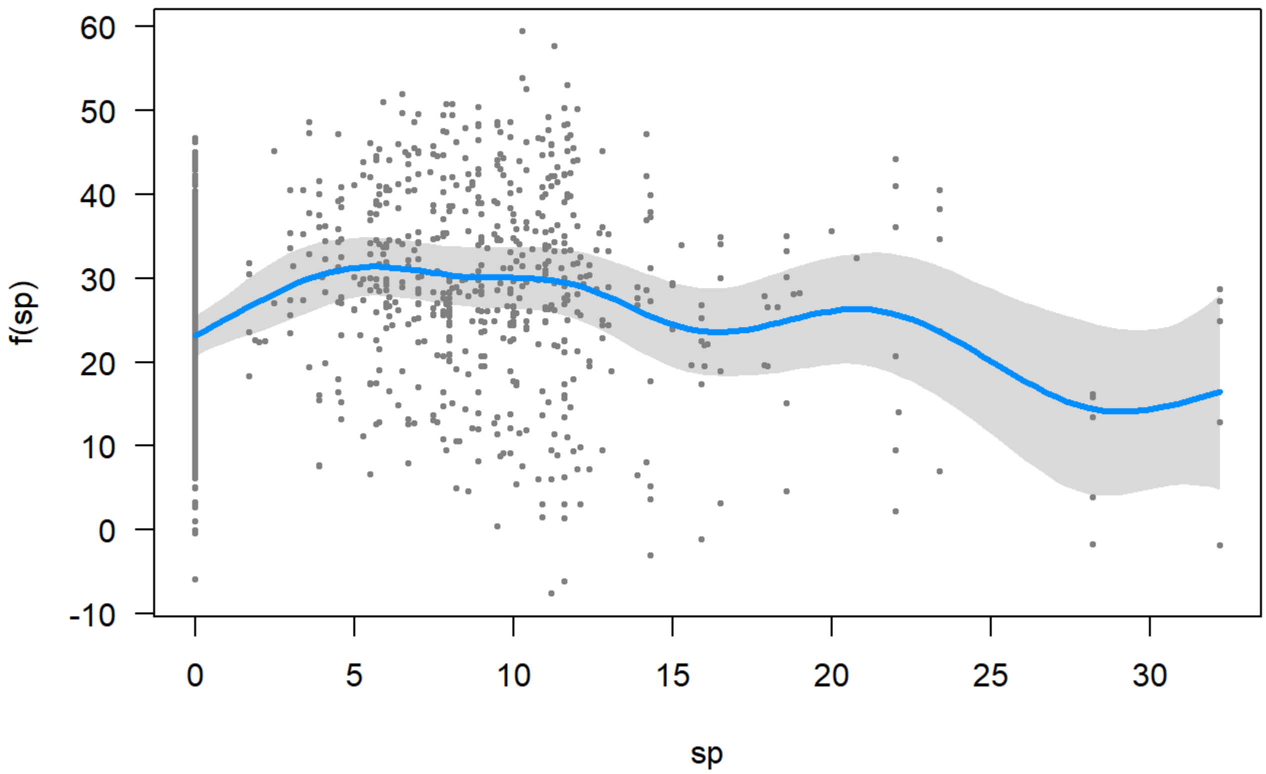

```
visreg(model2,'water')
```



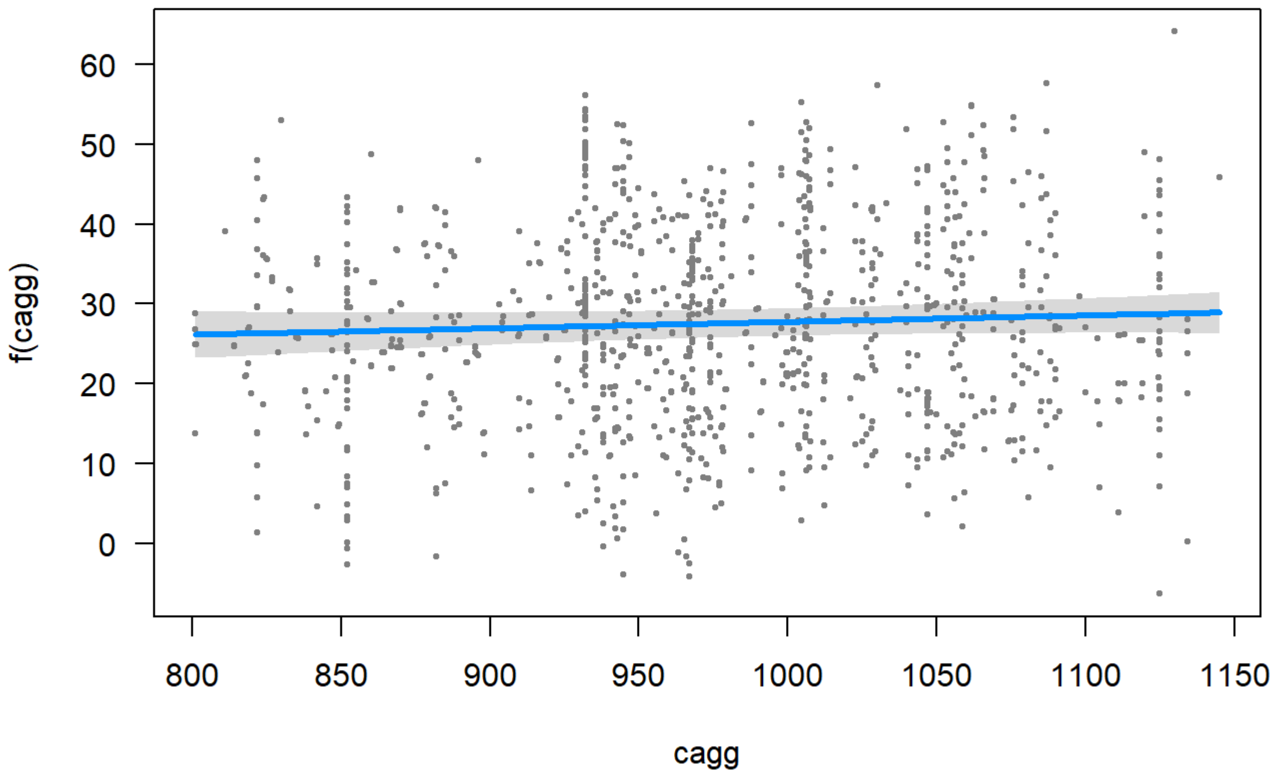

```
visreg(model1,'sp')
```



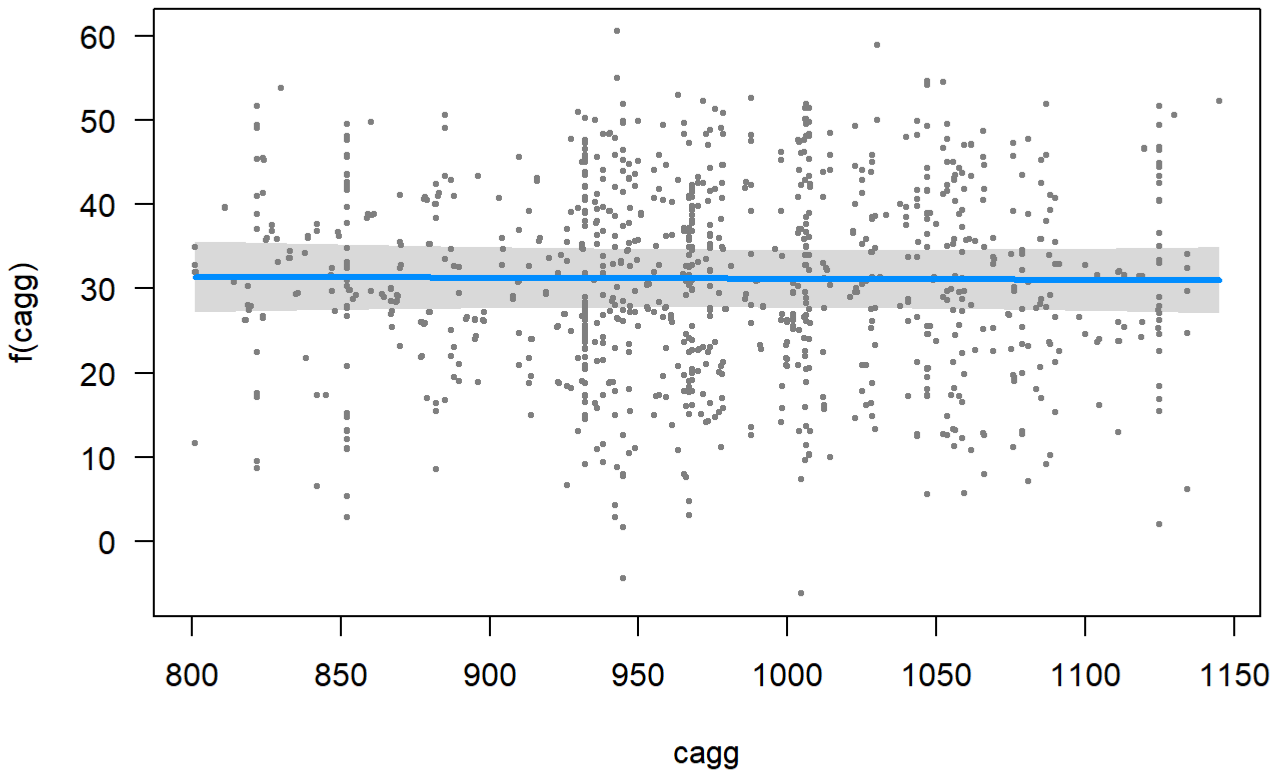

```
visreg(model2,'sp')
```




```
visreg(model1,'cagg')
```




```
visreg(model2,'cagg')
```



From CEM graph we can see that, the confidence interval after applying smoothing function has greater value as compared to the model before smoothing function. After applying the smoothing function, the confidence interval gets better.