Interpretability and Explainability in Machine Learning

Group 1: Pariente Antonio, Bosch Guillem, Ebner Lena

25/12/2023

Concrete Dataset

UC Irvine Machine Learning Repository, Concrete Dataset.

Abstract: Concrete is the most important material in civil engineering. The concrete compressive strength is a highly nonlinear function of age and ingredients. These ingredients include cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, and fine aggregate.

Data Characteristics: The actual concrete compressive strength (MPa) for a given mixture under a specific age (concretes) was determined from laboratory. Data is in raw form (not scaled).

Summary Statistics:

- Number of instances (observations): 1030
- Number of Attributes: 9
- Attribute breakdown: 8 quantitative input variables, and 1 quantitative output variable
- Missing Attribute Values: None

Variable Information: Given is the variable name, variable type, the measurement unit and a brief description.

Input Variables:

- Cement (component 1) – quantitative – kg in a m3 mixture
- Blast Furnace Slag (component 2) – quantitative – kg in a m3 mixture
- Fly Ash (component 3) – quantitative – kg in a m3 mixture
- Water (component 4) – quantitative – kg in a m3 mixture
- Superplasticizer (component 5) – quantitative – kg in a m3 mixture
- Coarse Aggregate (component 6) – quantitative – kg in a m3 mixture
- Fine Aggregate (component 7) – quantitative – kg in a m3 mixture
- Age – quantitative – concrete (1~365)

Response variable: - Concrete compressive strength – quantitative – MPa – Output Variable

```r
library(readxl)

concrete <- as.data.frame(read_excel("Concrete_Data.xls"))
DescVars <- names(concrete)
names(concrete) <- c("Cement","Slag","FlyAsh","Water","Superplast", "CoarseAggr","FineAggr","Age","Stre
```

Data processing: Creating training and test sets

Create a training sample choosing 700 data at random. The non-chosen data will be the test set.

```r
set.seed(123)

training_indices <- sort(sample(1:nrow(concrete), 700))
concrete_train <- concrete[training_indices, ]

concrete_test <- concrete[-training_indices, ]

dim(concrete_train)
```

```
## [1] 700   9
```

```r
dim(concrete_test)
```

```
## [1] 330   9
```

1. Fit a Random Forest

a. Compute the Variable Importance by the reduction of the **impurity** at the splits defined by each variable.
b. Compute the Variable Importance by out-of-bag random permutations.
c. Do a graphical representation of both Variable Importance measures.
d. Compute the Variable Importance of each variable by Shapley Values.

```r
library(ranger)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ranger':
##
##     importance
```

```r
library(caret)
```

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

```
##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:randomForest':
##
##     margin

## Loading required package: lattice
```

```r
library(vip)
```

```
##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##     vi
```

```r
library(DALEX)
```

```
## Welcome to DALEX (version: 2.4.3).
## Find examples and detailed introduction at: http://ema.drwhy.ai/
## Additional features will be available after installation of: ggpubr.
## Use 'install_dependencies()' to get all suggested dependencies

##
## Attaching package: 'DALEX'

## The following object is masked from 'package:vip':
##
##     titanic
```

```r
library(DALEXtra)
library(lime)
```

```
##
## Attaching package: 'lime'

## The following object is masked from 'package:DALEX':
##
##     explain
```

```r
library(iml)
library(localModel)
library(fastshap) # Attention! It re-define "explain" from DALEX
```

```
##
## Attaching package: 'fastshap'
```

```
## The following object is masked from 'package:lime':
##
##     explain
```

```
## The following objects are masked from 'package:DALEX':
##
##     explain, titanic
```

```
## The following object is masked from 'package:vip':
##
##     gen_friedman
```

```r
# if (require(ghostvar)){library(ghostvar)}
library(mgcv)
```

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

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

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:randomForest':
##
##     combine
```

a. Variable Importance by reduction of the impurity at splits

```r
model_rf_imp <- ranger(
  Strength ~ .,
  data = concrete_train,
  importance='impurity'
)
print(model_rf_imp)
```

```
## Ranger result
##
## Call:
##  ranger(Strength ~ ., data = concrete_train, importance = "impurity")
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      700
## Number of independent variables:  8
## Mtry:                             2
## Target node size:                 5
## Variable importance mode:         impurity
## Splitrule:                        variance
## OOB prediction error (MSE):       33.45219
## R squared (OOB):                  0.8764007
```

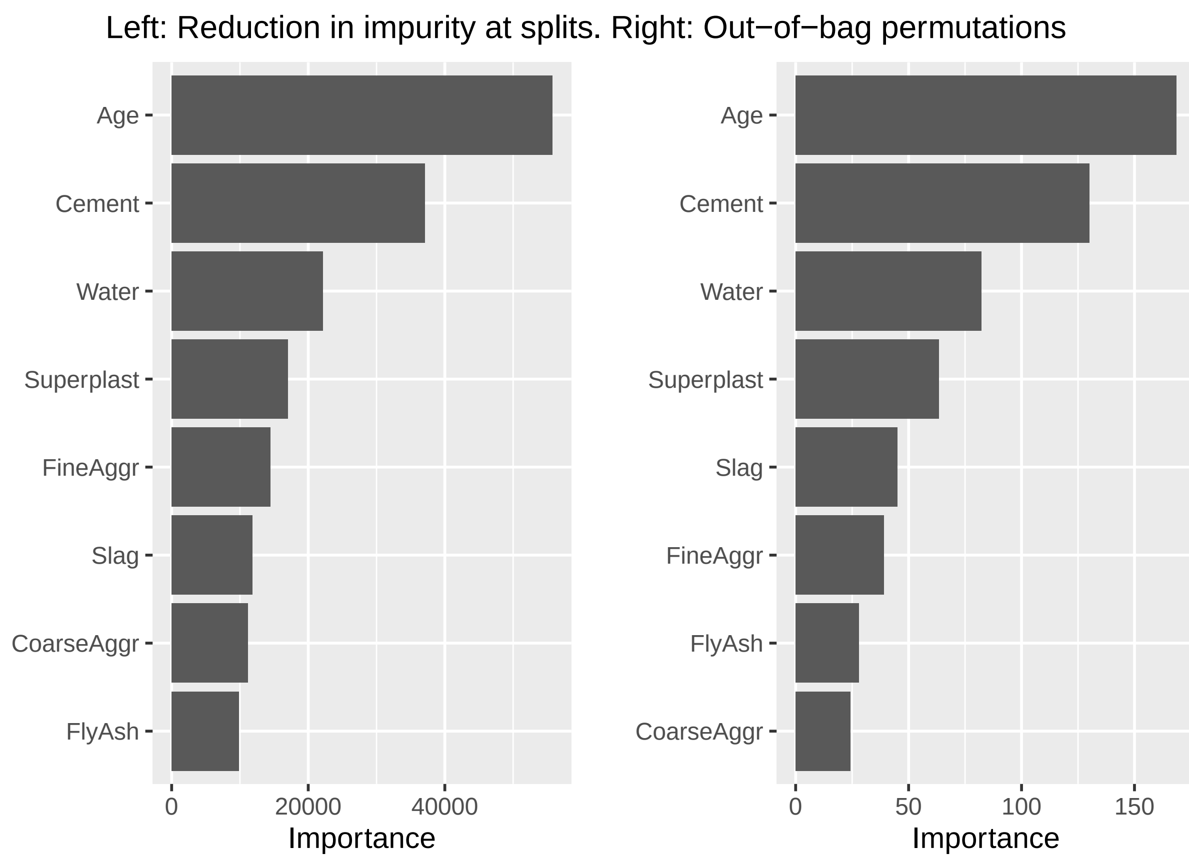
b. Variable Importance by out-of-bag random permutations

```
model_rf_perm <- ranger(
  Strength ~ .,
  data = concrete_train,
  importance='permutation'
)
print(model_rf_perm)
```

```
## Ranger result
##
## Call:
##  ranger(Strength ~ ., data = concrete_train, importance = "permutation")
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      700
## Number of independent variables:  8
## Mtry:                             2
## Target node size:                 5
## Variable importance mode:         permutation
## Splitrule:                        variance
## OOB prediction error (MSE):       33.13239
## R squared (OOB):                  0.8775822
```

c. Graphical Representation of Variable Importances

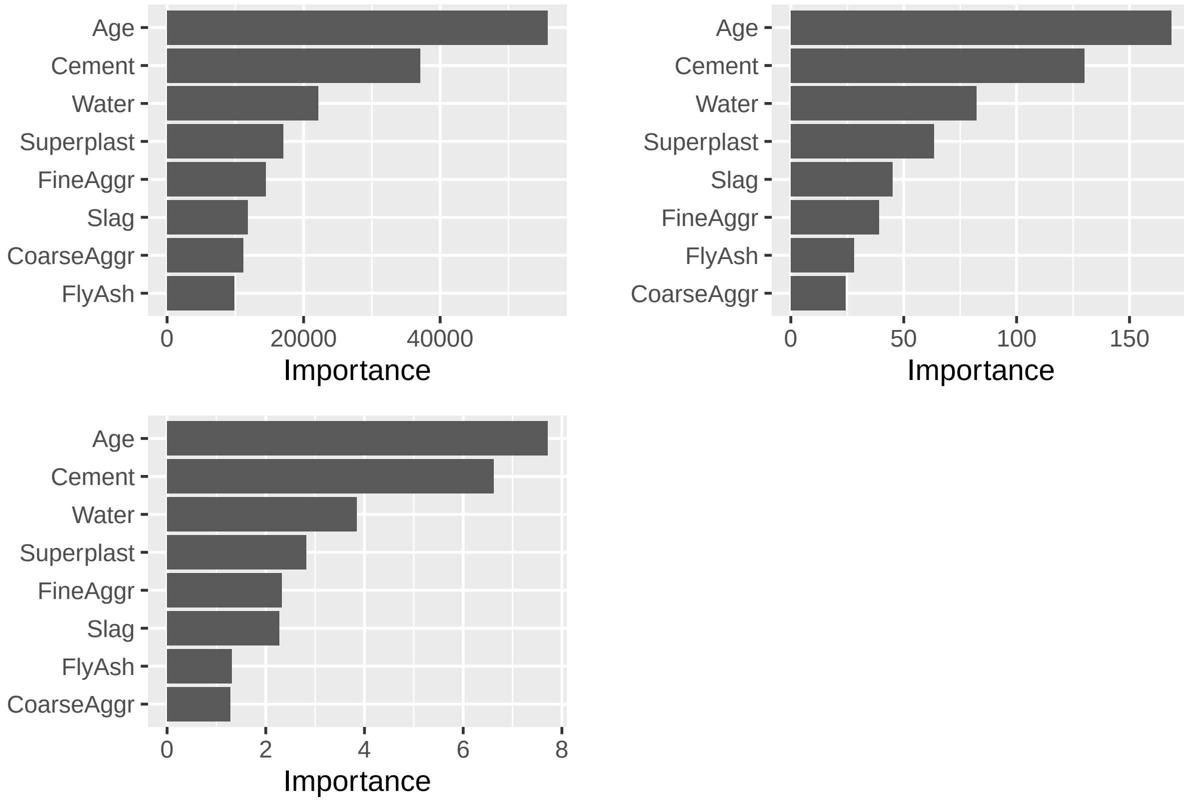
```
rf_imp_vip <- vip(model_rf_imp, num_features = 8)
rf_perm_vip <- vip(model_rf_perm, num_features = 8)
grid.arrange(rf_imp_vip, rf_perm_vip, ncol=2, top="Left: Reduction in impurity at splits. Right: Out-of-
```



d. Variable Importance by Shapley Values

```
rf_imp_vip <- vip(model_rf_imp, num_features = 8)
rf_perm_vip <- vip(model_rf_perm, num_features = 8)

rf_shapley <- vip(model_rf_imp, method="shap",
                  pred_wrapper=yhat, num_features = 8,
                  train = concrete_train,
                  newdata=concrete_test[,-9])

grid.arrange(rf_imp_vip, rf_perm_vip, rf_shapley,
             ncol=2, nrow=2,
             top="Top left: Impurity. Top Right: OOB Permutations. Bottom left: Shapley Values"
            )
```

Top left: Impurity. Top Right: OOB Permutations. Bottom left: Shapley Values



By looking at the plot we can see that all 3 ways of computing variable importance give similar results. We can see that the most important variables are, in order, Age, Cement and Water. Differences in the order of the importance of variables do not appear until the fifth position. None of the three orders are the same but the differences in the order of the predictors are mild.

2. Fit a linear model and a gam model.

a. Summarize, numerically and graphically, the fitted models.
b. Compute the Variable Importance by Shapley values in the linear and gam fitted models. Compare your results with what you have learned before.

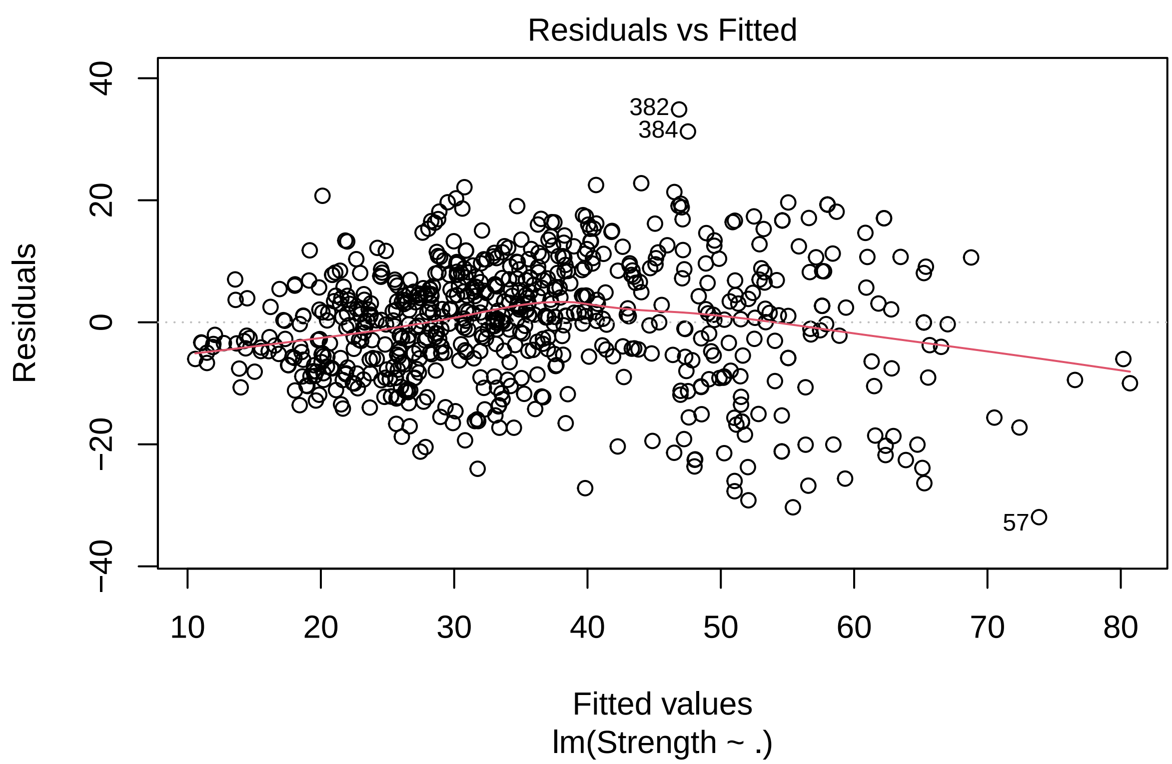
a. Linear Model

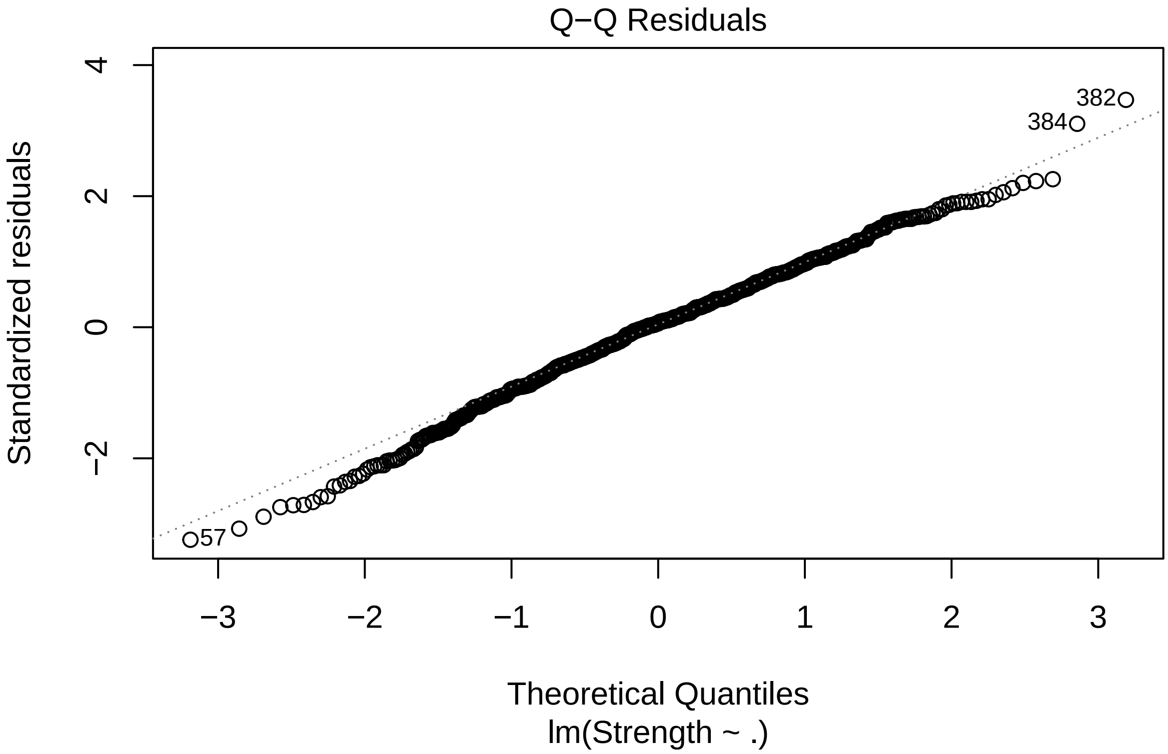
```
lm_concrete <- lm(Strength ~ ., data = concrete_train)
(summ_lm_concrete <- summary(lm_concrete))
```

```
##
## Call:
## lm(formula = Strength ~ ., data = concrete_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -31.934   -5.998    0.602    6.881   34.880
```

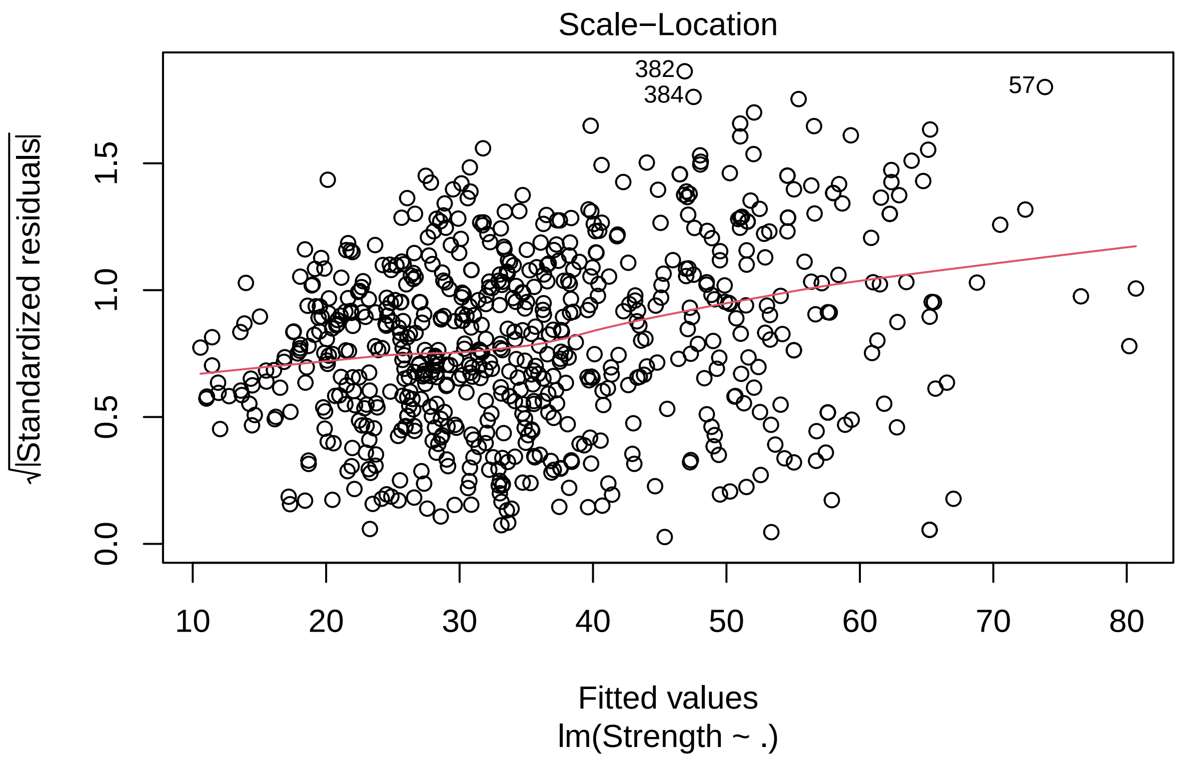
```
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -22.163227  30.941577  -0.716    0.474
## Cement        0.113761   0.010020  11.353  < 2e-16 ***
## Slag          0.097243   0.011870   8.193 1.24e-15 ***
## FlyAsh        0.073444   0.014957   4.911 1.13e-06 ***
## Water        -0.120030   0.046478  -2.583    0.010 *
## Superplast    0.517791   0.116083   4.461 9.54e-06 ***
## CoarseAggr    0.018545   0.010992   1.687    0.092 .
## FineAggr      0.013261   0.012532   1.058    0.290
## Age           0.121099   0.006842  17.700  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.13 on 691 degrees of freedom
## Multiple R-squared:  0.6252, Adjusted R-squared:  0.6209
## F-statistic: 144.1 on 8 and 691 DF,  p-value: < 2.2e-16
```

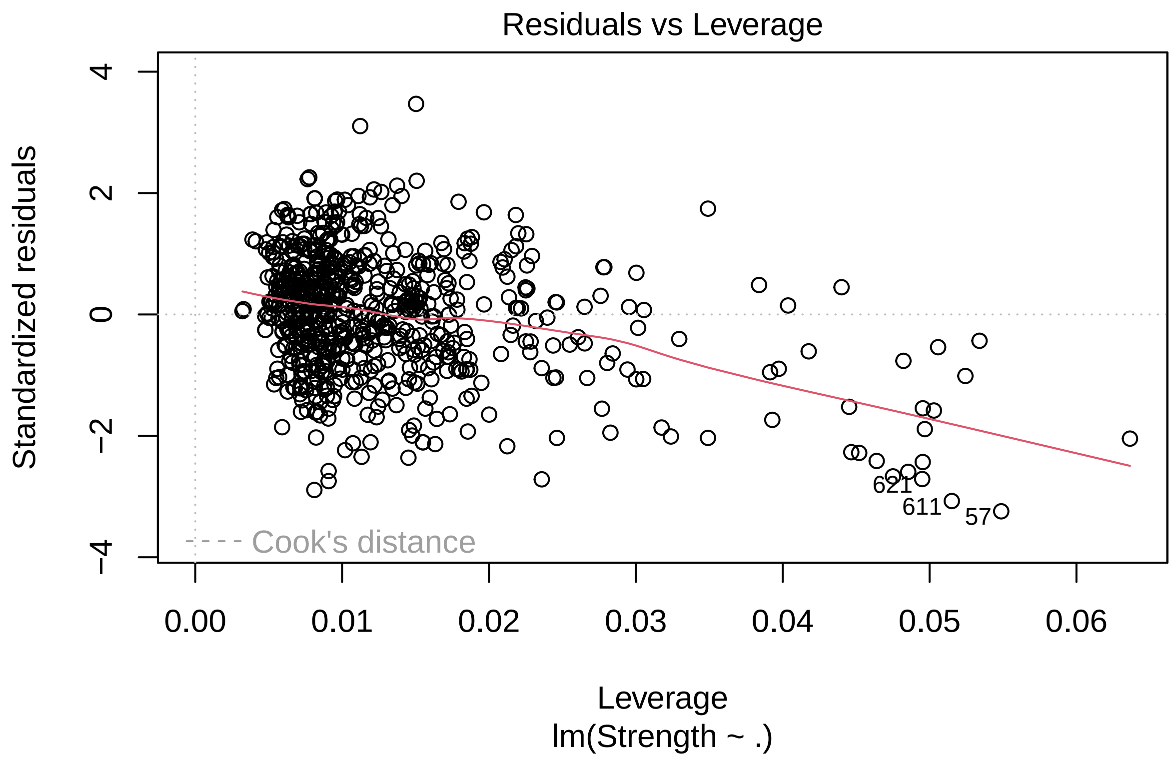
```r
plot(lm_concrete)
```





Q−Q Residuals
Standardized residuals
4
2
0
−2
382
384
57
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(Strength ~ .)


Scale−Location
382
384
57
√|Standardized residuals|
0.0
0.5
1.0
1.5
10
20
30
40
50
60
70
80
Fitted values
lm(Strength ~ .)



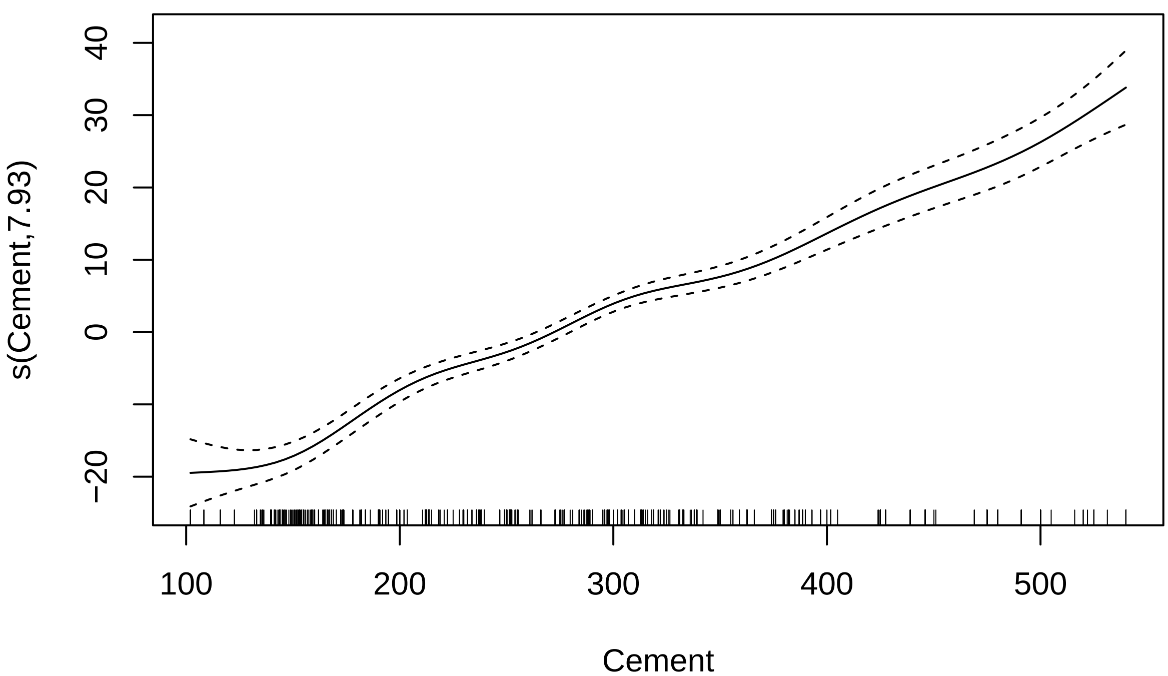
a. GAM Model

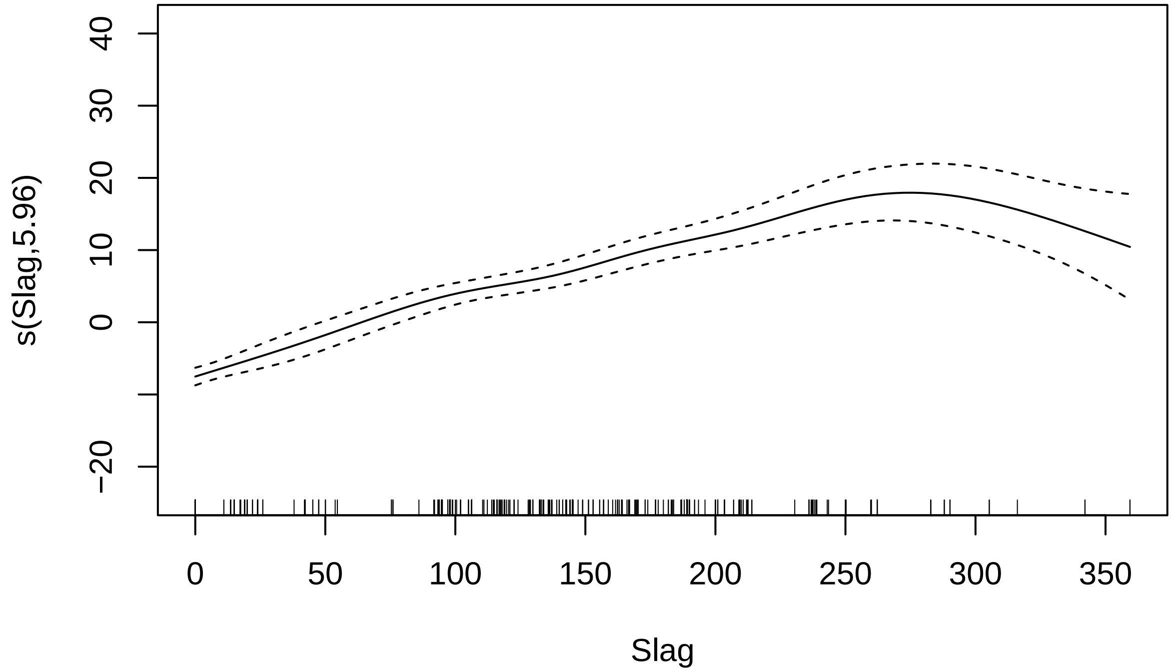
```
gam_concrete <- gam(Strength ~ s(Cement) + s(Slag) + s(FlyAsh) + s(Water) + s(Superplast) + s(CoarseAgg
(summary_gam_concrete <- summary(gam_concrete))
```

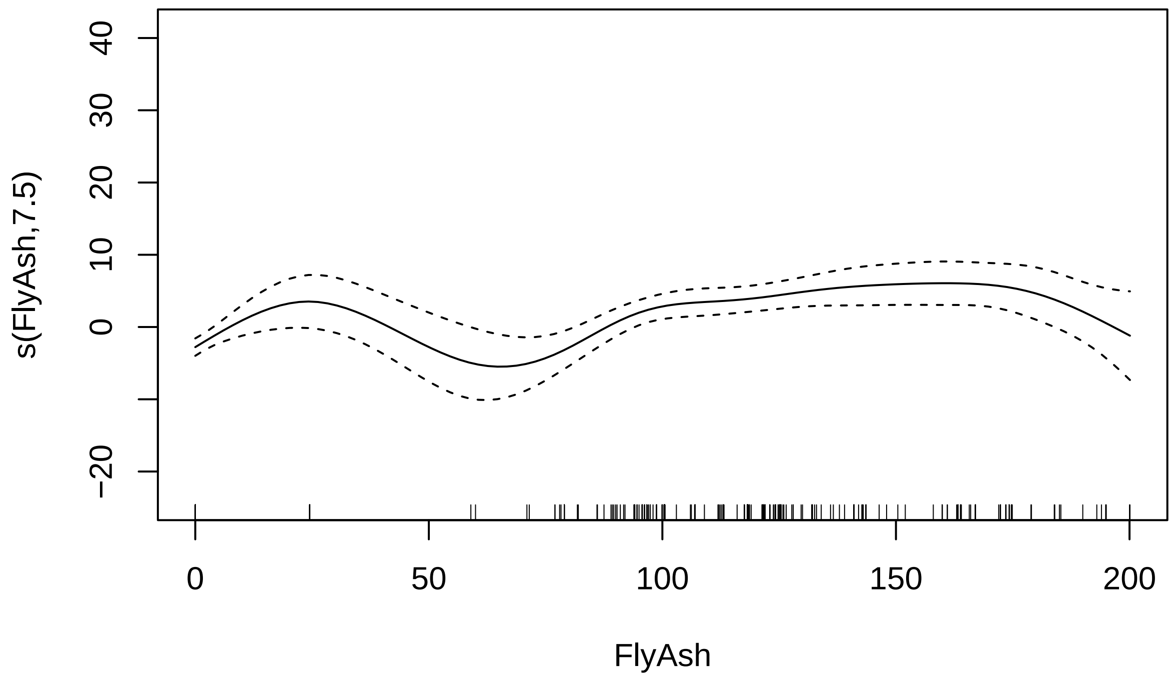
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## Strength ~ s(Cement) + s(Slag) + s(FlyAsh) + s(Water) + s(Superplast) +
##     s(CoarseAggr) + s(FineAggr) + s(Age)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  35.5892     0.2062   172.6   <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(Cement)     7.934  8.692 38.512  < 2e-16 ***
## s(Slag)       5.958  7.063 23.305  < 2e-16 ***
## s(FlyAsh)     7.503  8.368  6.926  < 2e-16 ***
```

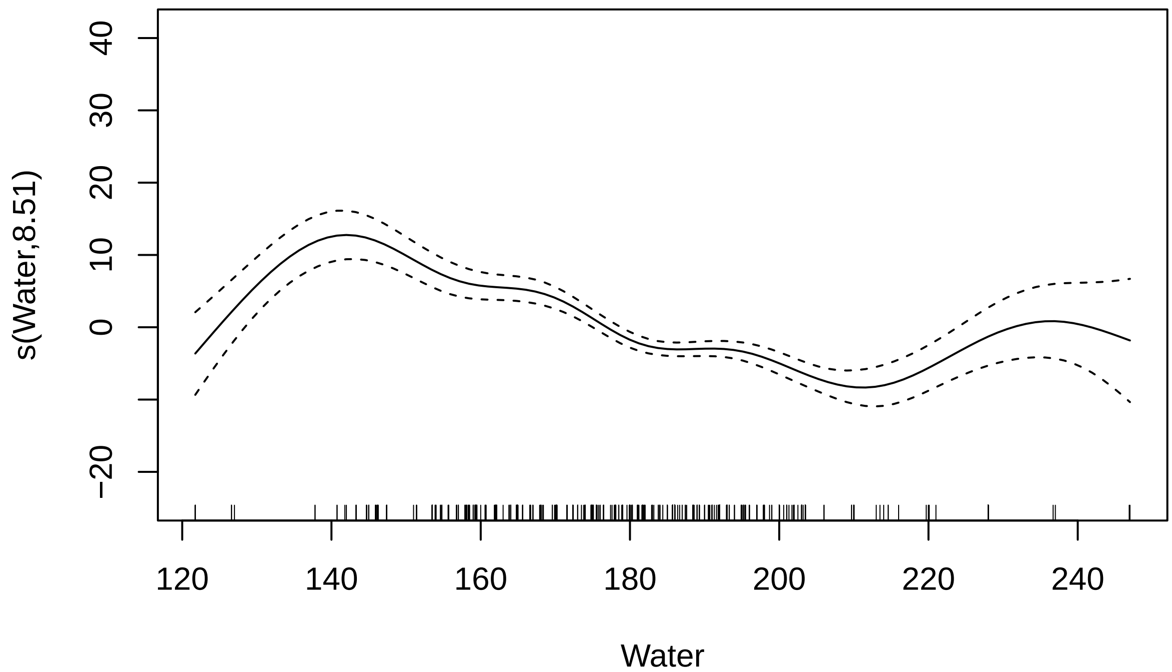
```
## s(Water)       8.512  8.916  17.078  < 2e-16 ***
## s(Superplast) 6.869  7.889   4.166 7.63e-05 ***
## s(CoarseAggr) 1.000  1.000   0.458    0.499
## s(FineAggr)   8.572  8.935   9.130  < 2e-16 ***
## s(Age)        8.400  8.810 250.570  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.89   Deviance explained = 89.9%
## GCV =  32.35  Scale est. = 29.774    n = 700
```

```
plot(gam_concrete)
```

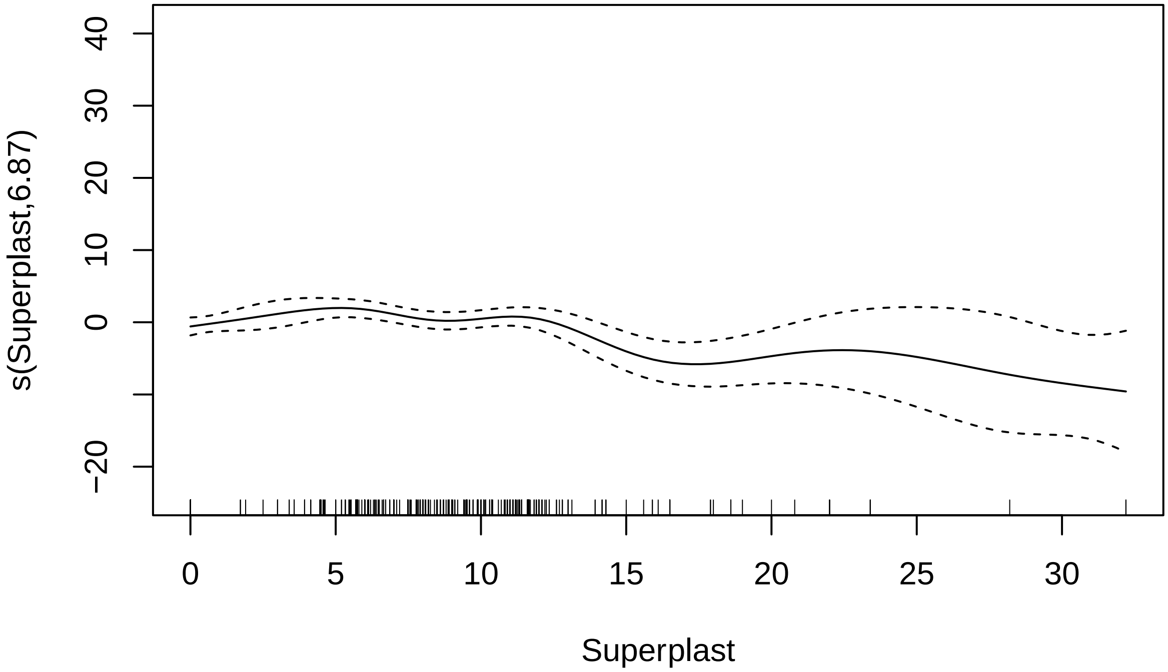



s(Slag,5.96)
40
30
20
10
0
−20
0
50
100
150
200
250
300
350
Slag

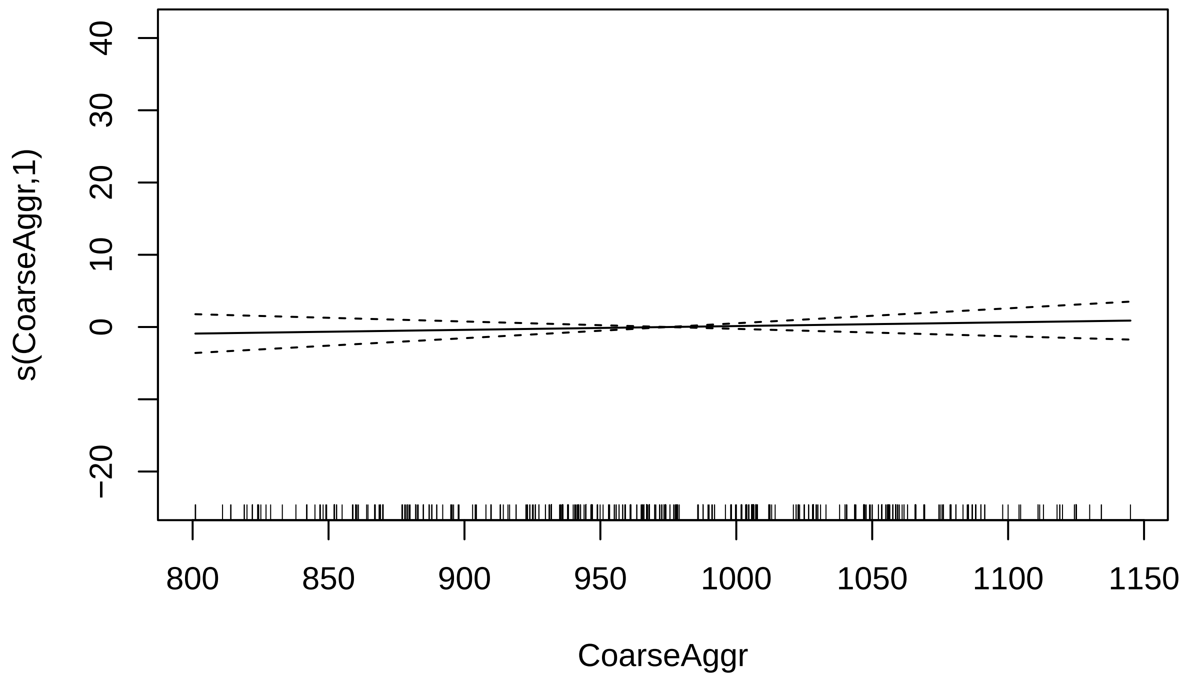

s(FlyAsh,7.5)
40
30
20
10
0
−20
0
50
100
150
200
FlyAsh

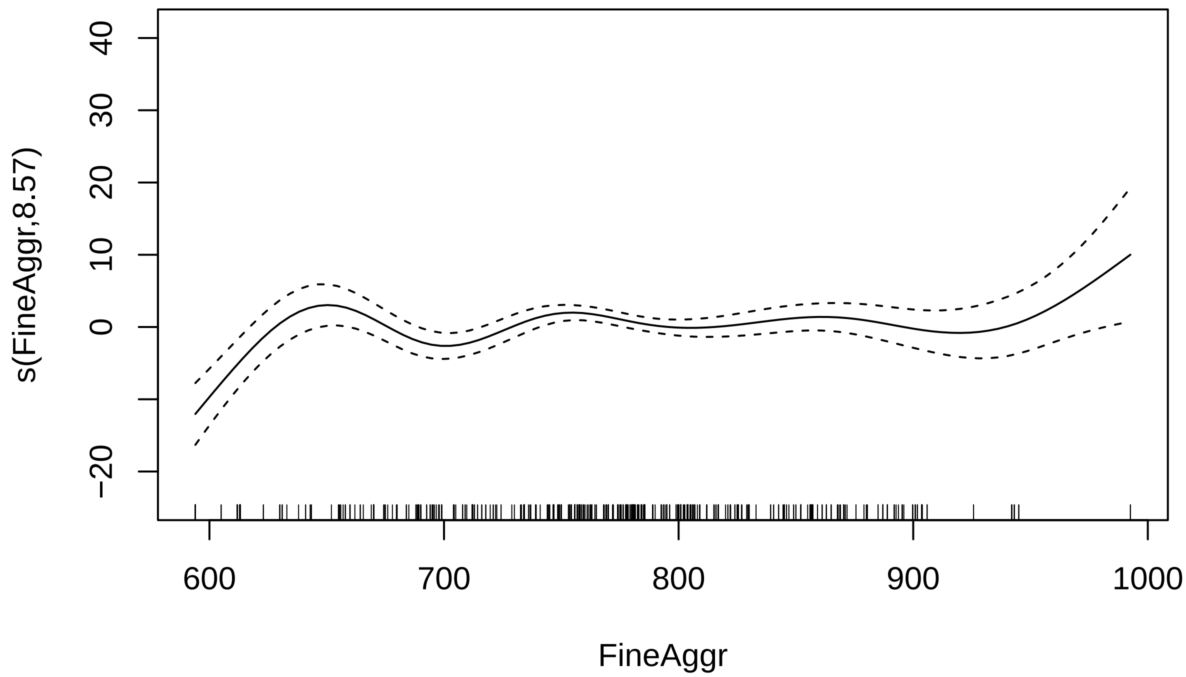


s(Water,8.51)
40
30
20
10
0
−20
120
140
160
180
200
220
240
Water

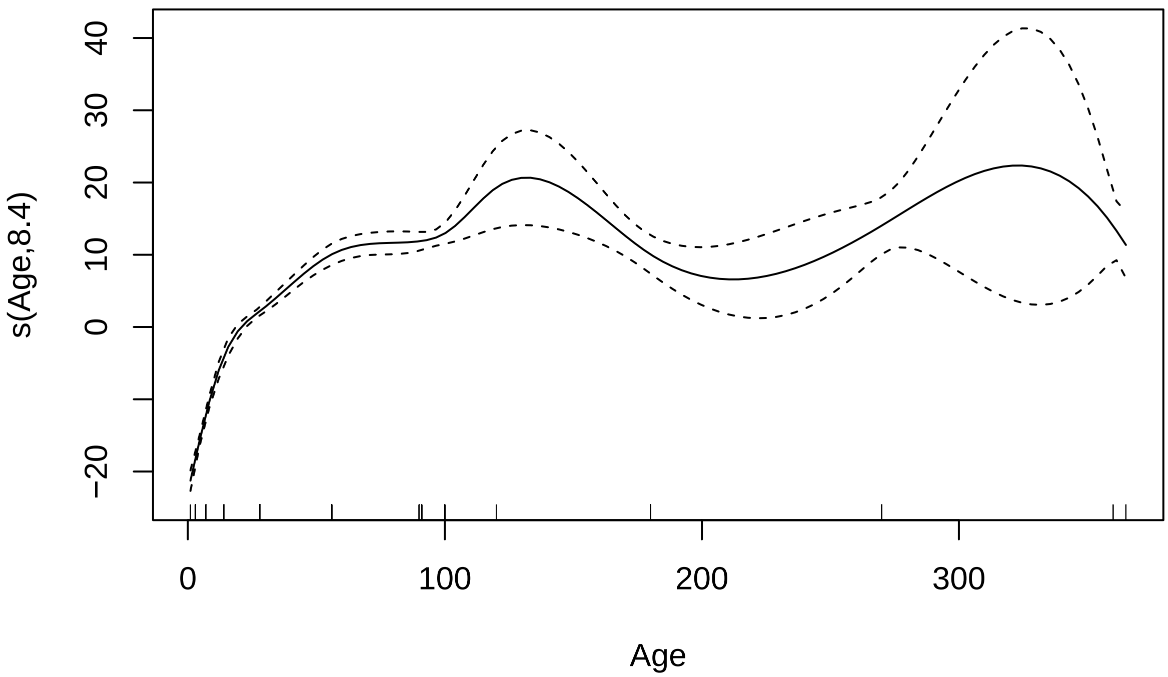


s(Superplast,6.87)
40
30
20
10
0
−20
0
5
10
15
20
25
30
Superplast


s(CoarseAggr,1)
40
30
20
10
0
−20
800
850
900
950
1000
1050
1100
1150
CoarseAggr



s(FineAggr,8.57)
40
30
20
10
0
−20
600
700
800
900
1000
FineAggr



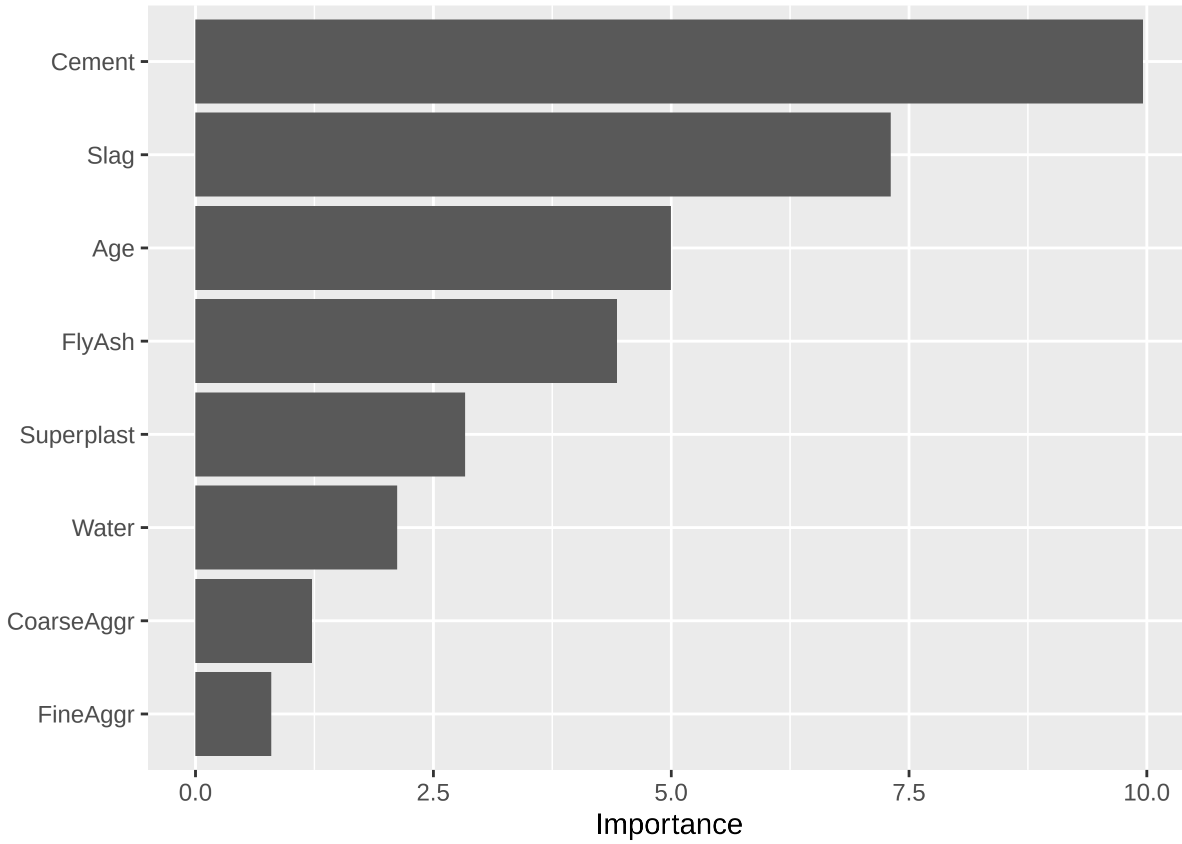
The linear model is able to explain 62% of the variability in the data, the GAM model achieves 89.9%.

A difference between the two models is that the GAM model also considers the Water as significant parameter for predicting the Strength, due to the effect of this variable to the Strength of concrete is highly non-linear. The same phenomenon is present with FineAggr. The best linear approximation is a constant but if non-linear dependence is allowed, better predictors appear.

b. Variable Importance by Shapley Comparision

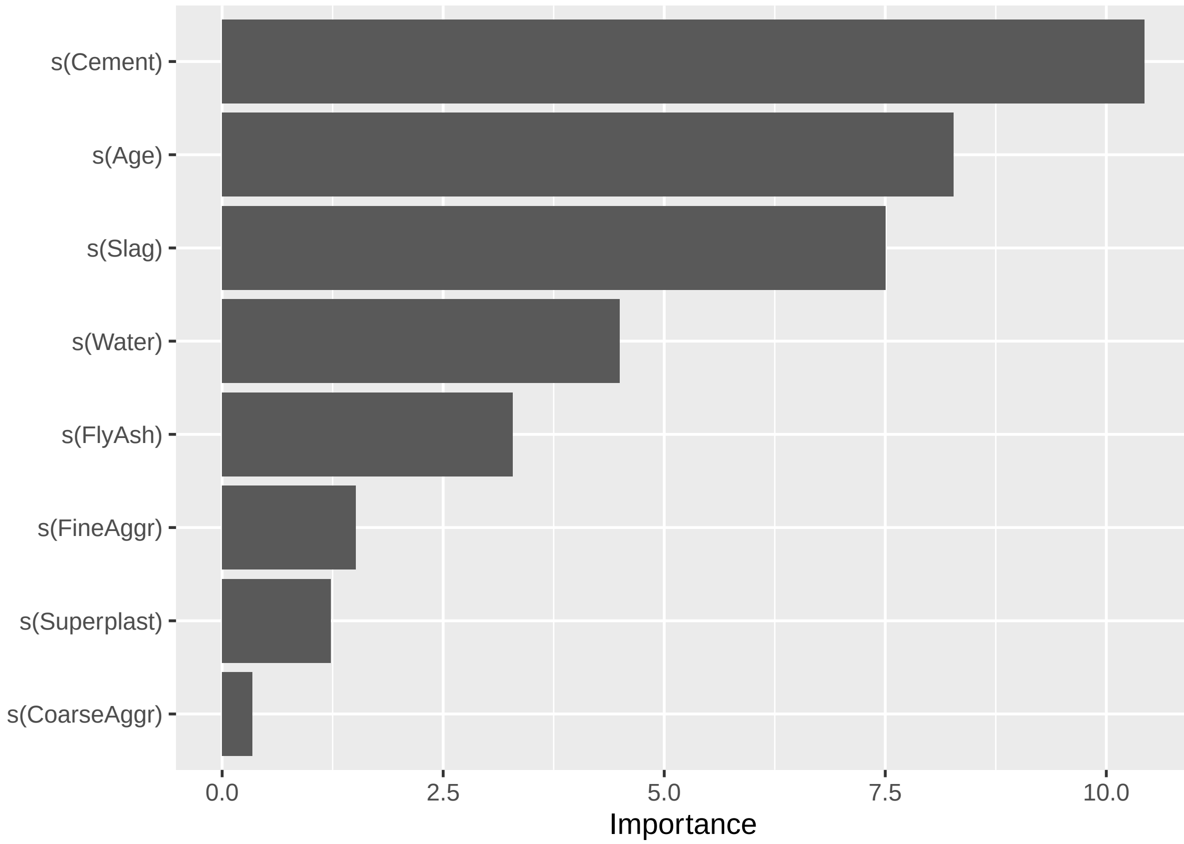
Linear Model

```
lm_concrete_shapley <- vip(lm_concrete, method="shap",
                  pred_wrapper=predict.lm,
                  train=concrete_train,
                  newdata=concrete_test,
                  num_features = 8,
                  exact=TRUE)
plot(lm_concrete_shapley)
```

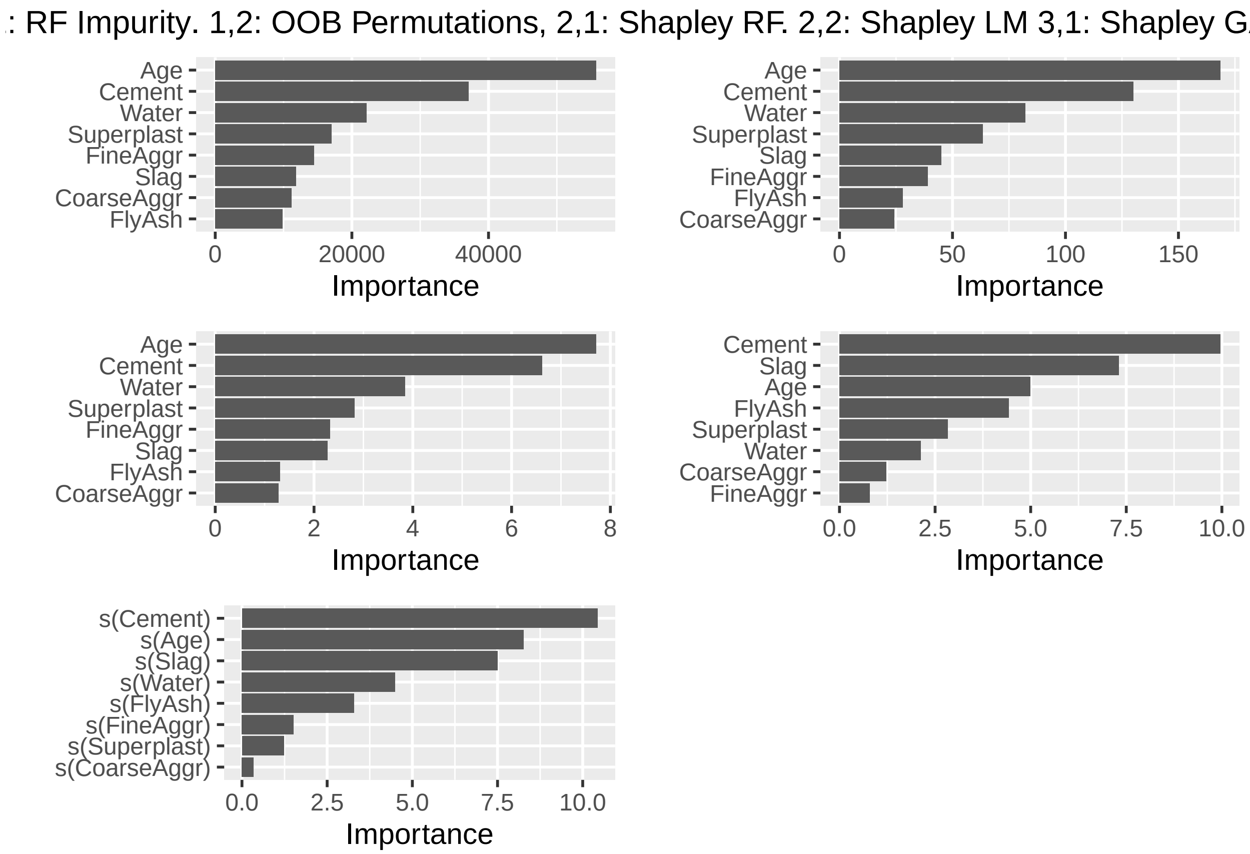


GAM Model

```
gam_concrete_shapley <- vip(gam_concrete, method="shap",
                  pred_wrapper=predict.gam,
                  train=concrete_train,
                  newdata=concrete_test,
                  num_features = 8,
                  exact=TRUE)

plot(gam_concrete_shapley)
```




```
grid.arrange(rf_imp_vip, rf_perm_vip, rf_shapley,
             lm_concrete_shapley, gam_concrete_shapley,
             ncol=2, nrow=3,
             top="1,1: RF Impurity. 1,2: OOB Permutations, 2,1: Shapley RF. 2,2: Shapley LM 3,1: Shaple
)
```



Comparing the variable importance from the Linear and GAM models to the previously obtained results, we can see some differences. Compared to the random forest models, in the LM and GAM model Cement is the most important variable. In the linear model, Slag is way more important and ranked as 2nd most important variable, far away from the fifth or sixth position in the RF. The GAM models also ranks the Slag variable higher (in third Position) than in the random forest models. Superplast does not seem as important as in random forest models. Also the variable FlyAsh seems more important in the LM and GAM model than in the RF models.

We attribute these differences to non-smooth changes of the Strength with respect to the predictors; those changes are better modeled with random forests, so variables with such changes will appear important in one model but non-important in the linear models or GAMs.

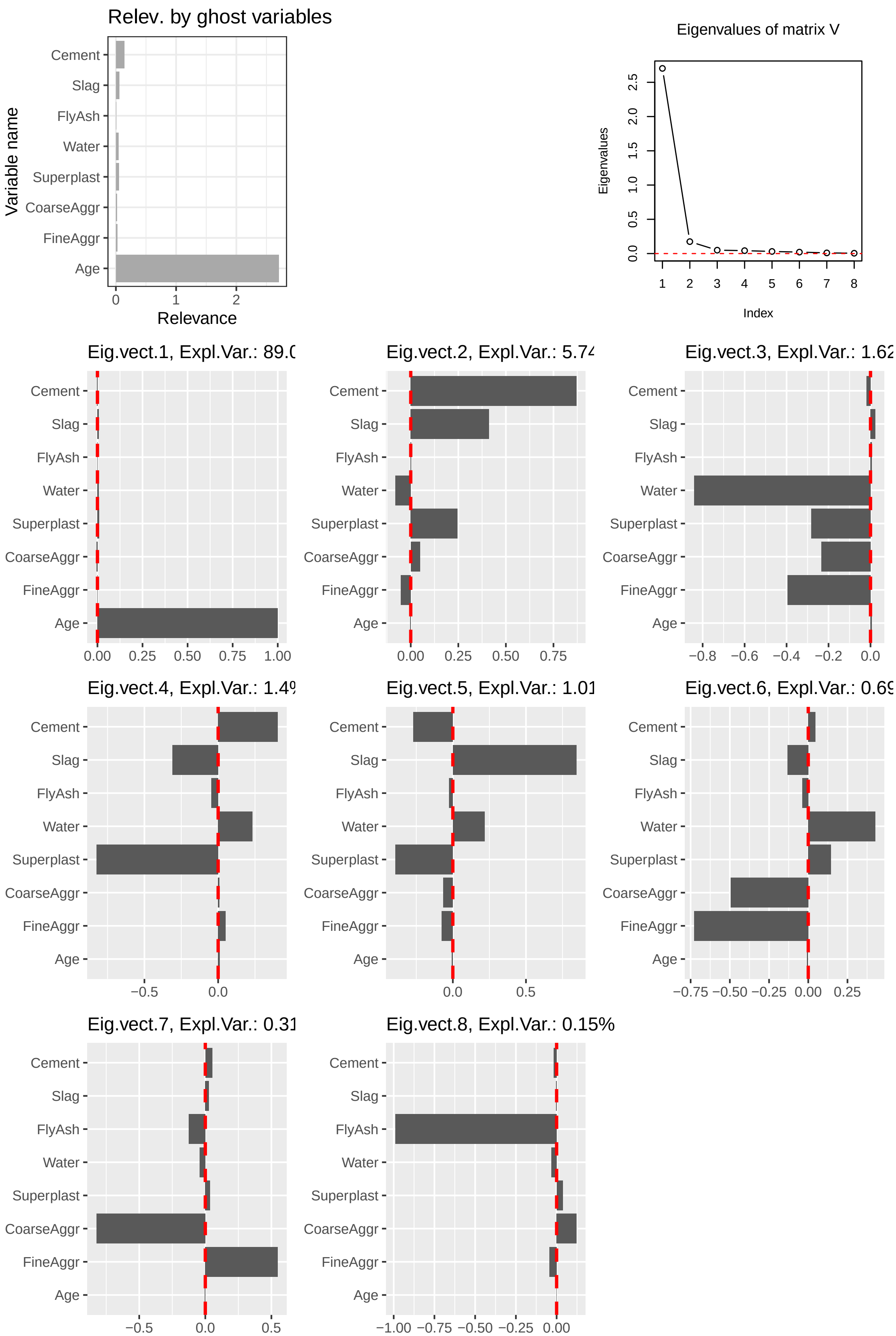
3. Relevance by Ghost Variables

Compute the relevance by ghost variables in the three fitted models.

Ghost Variables of RF Model

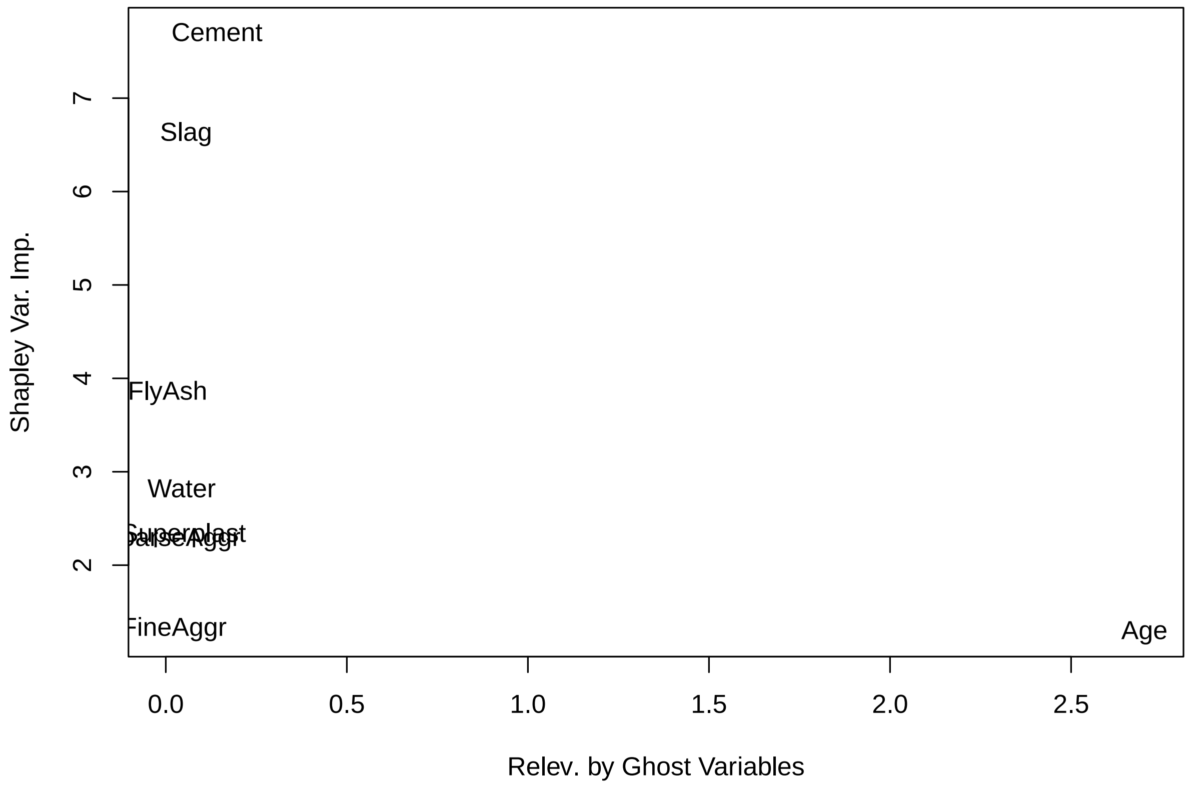
```
library(grid)
source("relev.ghost.var.R")

rf_model <- randomForest(Strength ~ ., data = concrete_train)

Rel_Gh_Var_Rf <- relev.ghost.var(model=rf_model,
```

```
                               newdata = concrete_test[,-9],
                               y.ts = concrete_test[,9],
                               func.model.ghost.var=gam
)
plot.relev.ghost.var(Rel_Gh_Var_Rf,n1=500,ncols.plot = 3)
```



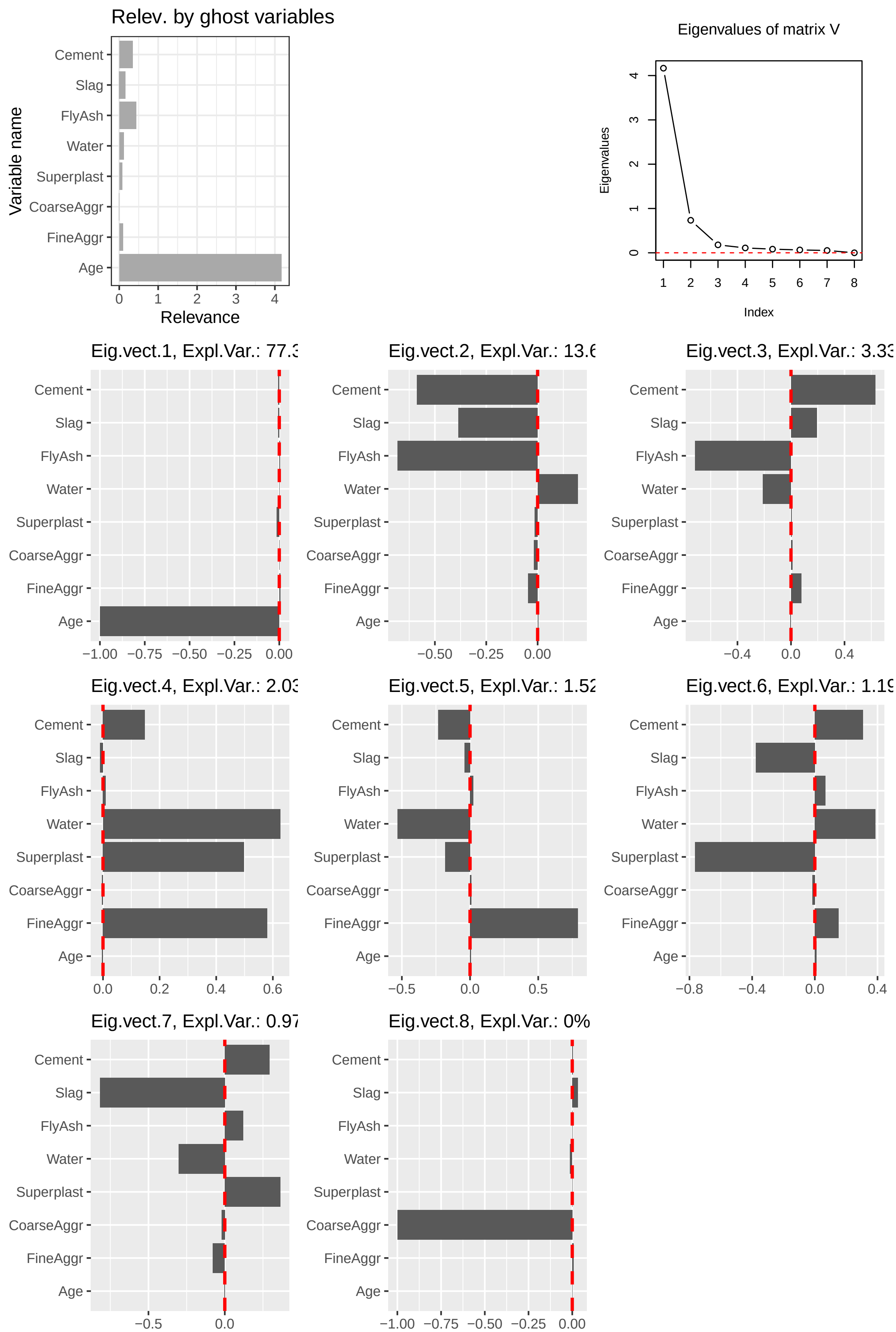
Relev. by ghost variables
Variable name
Cement
Slag
FlyAsh
Water
Superplast
CoarseAggr
FineAggr
Age
0
1
2
Relevance
Eigenvalues of matrix V
Eigenvalues
0.0
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7
8
Index
Eig.vect.1, Expl.Var.: 89.0
0.00
0.25
0.50
0.75
1.00
Eig.vect.2, Expl.Var.: 5.74
0.00
0.25
0.50
0.75
Eig.vect.3, Expl.Var.: 1.62
−0.8
−0.6
−0.4
−0.2
0.0
Eig.vect.4, Expl.Var.: 1.49
−0.5
0.0
Eig.vect.5, Expl.Var.: 1.01
0.0
0.5
Eig.vect.6, Expl.Var.: 0.69
−0.75
−0.50
−0.25
0.00
0.25
Eig.vect.7, Expl.Var.: 0.31
−0.5
0.0
0.5
Eig.vect.8, Expl.Var.: 0.15%
−1.00
−0.75
−0.50
−0.25
0.00

```r
aux <- cbind(Rel_Gh_Var_Rf$relev.ghost,rf_shapley$data$Importance)
plot(aux[,1],aux[,2],col=0,xlab="Relev. by Ghost Variables",ylab="Shapley Var. Imp.")
text(aux[,1],aux[,2],row.names(aux))
```

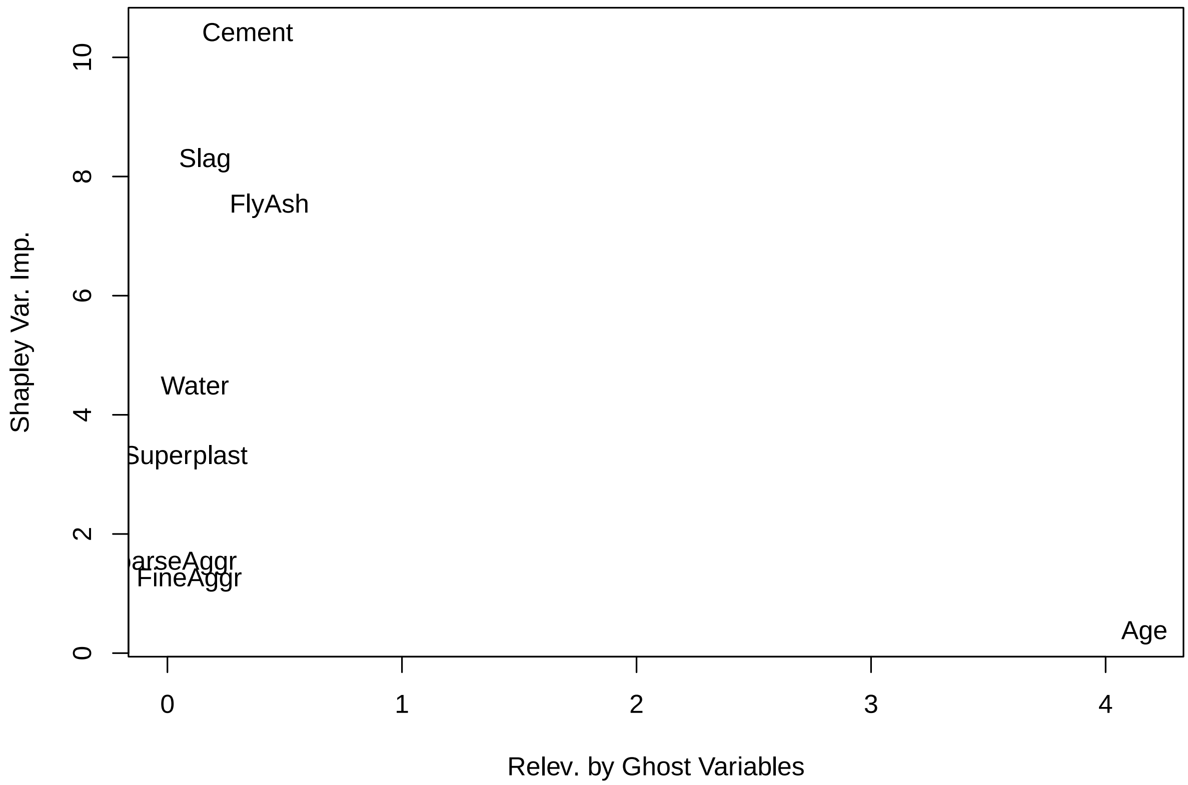


Ghost Variables of GAM Model

```r
Rel_Gh_Var_Gam <- relev.ghost.var(model=gam_concrete,
                              newdata = concrete_test[,-9],
                              y.ts = concrete_test[,9],
                              func.model.ghost.var=gam
)
plot.relev.ghost.var(Rel_Gh_Var_Gam,n1=500,ncols.plot = 3)
```

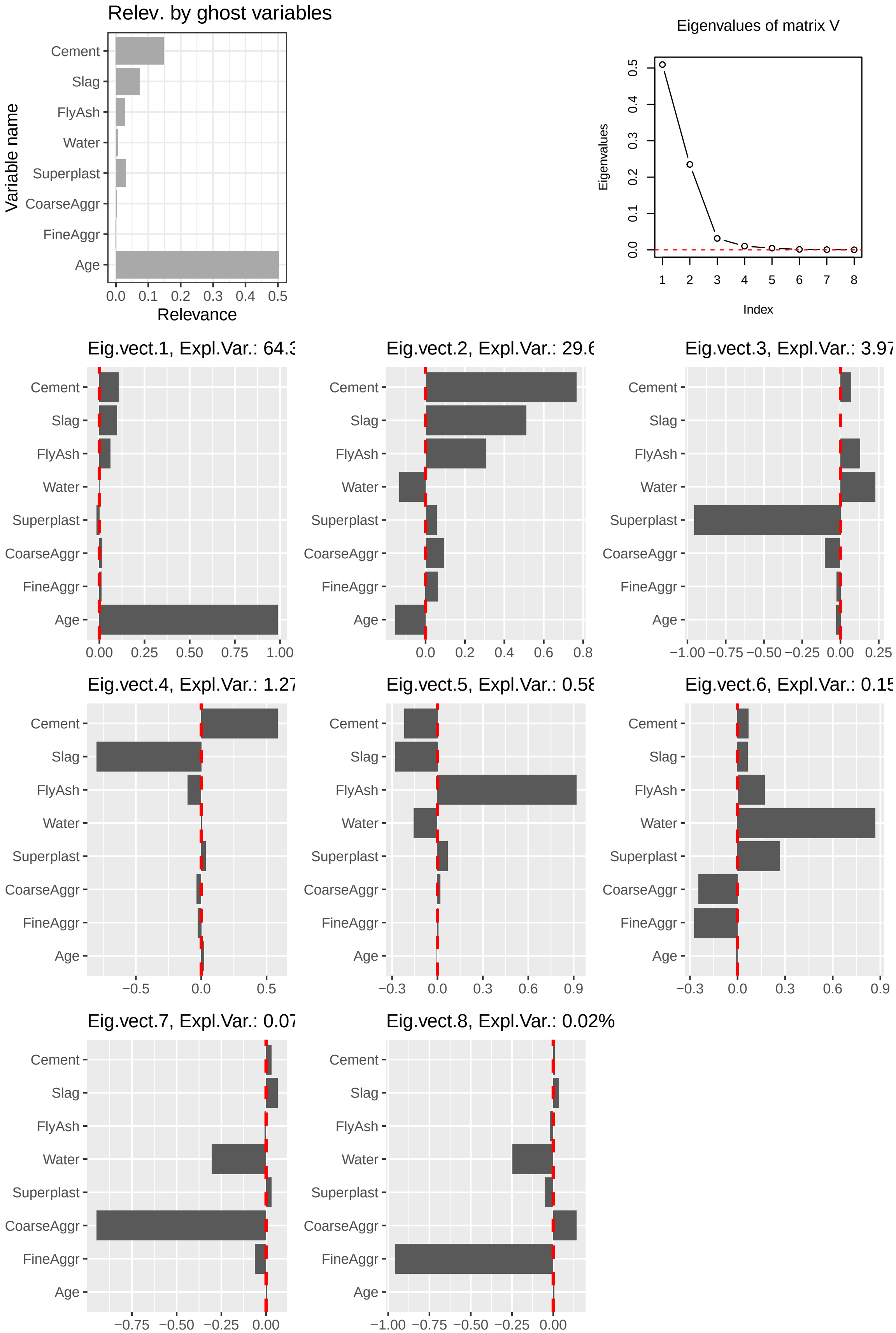

Relev. by ghost variables
Variable name
Relevance
Cement
Slag
FlyAsh
Water
Superplast
CoarseAggr
FineAggr
Age
Eigenvalues of matrix V
Eigenvalues
Index
Eig.vect.1, Expl.Var.: 77.3
Eig.vect.2, Expl.Var.: 13.6
Eig.vect.3, Expl.Var.: 3.33
Eig.vect.4, Expl.Var.: 2.03
Eig.vect.5, Expl.Var.: 1.52
Eig.vect.6, Expl.Var.: 1.19
Eig.vect.7, Expl.Var.: 0.97
Eig.vect.8, Expl.Var.: 0%

```
aux <- cbind(Rel_Gh_Var_Gam$relev.ghost,gam_concrete_shapley$data$Importance)
plot(aux[,1],aux[,2],col=0,xlab="Relev. by Ghost Variables",ylab="Shapley Var. Imp.")
text(aux[,1],aux[,2],row.names(aux))
```



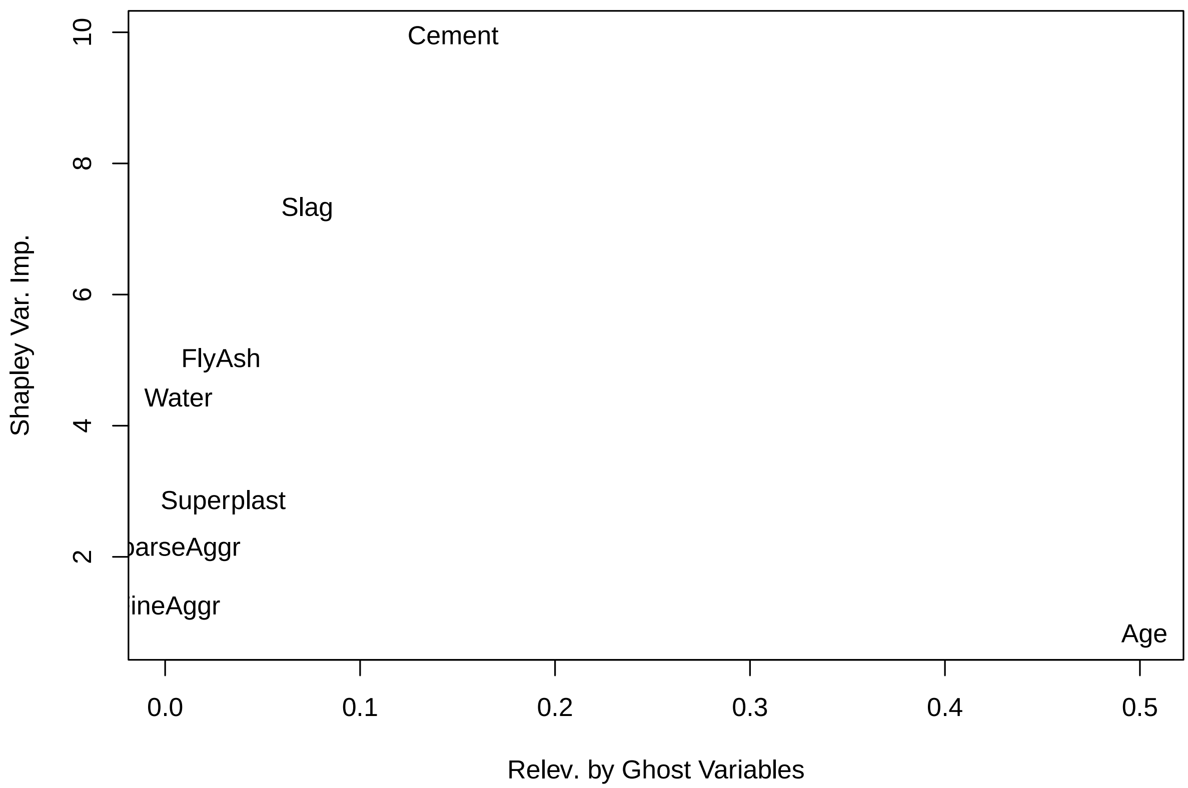
Ghost Variables of LM Model

```
Rel_Gh_Var_Lm <- relev.ghost.var(model=lm_concrete,
                                 newdata = concrete_test[,-9],
                                 y.ts = concrete_test[,9],
                                 func.model.ghost.var = lm
)
plot.relev.ghost.var(Rel_Gh_Var_Lm,n1=500,ncols.plot = 3)
```



Relev. by ghost variables
Variable name
Relevance
Eigenvalues of matrix V
Eigenvalues
Index
Eig.vect.1, Expl.Var.: 64.3
Eig.vect.2, Expl.Var.: 29.6
Eig.vect.3, Expl.Var.: 3.97
Eig.vect.4, Expl.Var.: 1.27
Eig.vect.5, Expl.Var.: 0.58
Eig.vect.6, Expl.Var.: 0.15
Eig.vect.7, Expl.Var.: 0.07
Eig.vect.8, Expl.Var.: 0.02%
Cement
Slag
FlyAsh
Water
Superplast
CoarseAggr
FineAggr
Age

```r
aux <- cbind(Rel_Gh_Var_Lm$relev.ghost,lm_concrete_shapley$data$Importance)
plot(aux[,1],aux[,2],col=0,xlab="Relev. by Ghost Variables",ylab="Shapley Var. Imp.")
text(aux[,1],aux[,2],row.names(aux))
```



In all three models Cement, Slag and FlyAsh are identified as important variables by the Shapley values method but not important by the Ghost variable method. Instead Age is the most important value by far in the ghost variable method axis.

In the three models the first eigenvector of the results of the ghost variable method corresponds exactly to the Age variable and explains a high fraction of the total variability. The variables Cement, Slag and FlyAsh appear relevant in less important eigenvectors.

The linear model is the model in which the variables Cement, Slag and FlyAsh appear to be more relevant in the ghost variable axis too. These variables are also part of the second eigenvector, with around 30% of explained variance, higher than in the other methods.

We attribute the fact that the variable Age appears to be important only by the ghost variable method to the fact that this variable cannot be well predicted by the other variables. This makes sense with the definition of the variable because Age is something independent to physical composition of the material.

4. Global Importance Measures and Plots using the library DALEX

a. Compute Variable Importance by Random Permutations
b. Do the Partial Dependence Plot for each explanatory variable.
c. Do the Local (or Conditional) Dependence Plot for each explanatory variable.

a. Variable Importance by Random Permutations

Random forest

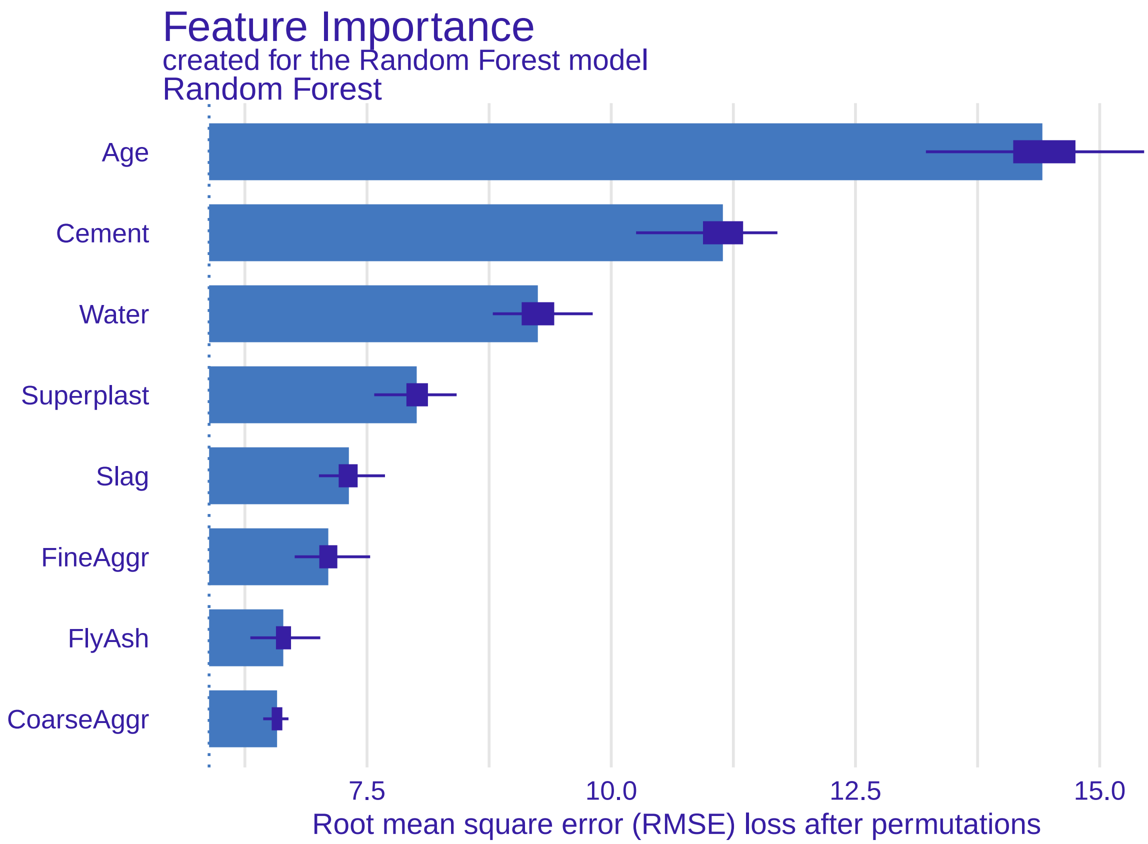
```r
explainer_rf <- DALEX::explain.default(model = model_rf_imp,
                                       data = concrete_test[, -9],
                                       y = concrete_test[, 9],
                                       label = "Random Forest")
```

```
## Preparation of a new explainer is initiated
##   -> model label       :  Random Forest
##   -> data              :  330  rows  8  cols
##   -> target variable   :  330  values
##   -> predict function  :  yhat.ranger  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package ranger , ver. 0.16.0 , task regression (  default  )
##   -> predicted values  :  numerical, min =  7.801456 , mean =  36.32307 , max =  75.43056
##   -> residual function :  difference between y and yhat (  default  )
##   -> residuals         :  numerical, min =  -21.47266 , mean =  -0.02022834 , max =  24.29542
##   A new explainer has been created!
```

```r
Rnd_Perm_rf <- model_parts(
  explainer_rf,
  N = NULL, # All available data are used
  B = 100   # number of permutations to be used, with B = 10 used by default
)

Rnd_Perm_rf
```

```
##        variable mean_dropout_loss         label
## 1  _full_model_          5.883609 Random Forest
## 2    CoarseAggr          6.579350 Random Forest
## 3        FlyAsh          6.642866 Random Forest
## 4      FineAggr          7.103662 Random Forest
## 5          Slag          7.314364 Random Forest
## 6    Superplast          8.008878 Random Forest
## 7         Water          9.248729 Random Forest
## 8        Cement         11.142732 Random Forest
## 9           Age         14.413091 Random Forest
## 10   _baseline_         22.098074 Random Forest
```

```r
plot(Rnd_Perm_rf)
```



Linear model

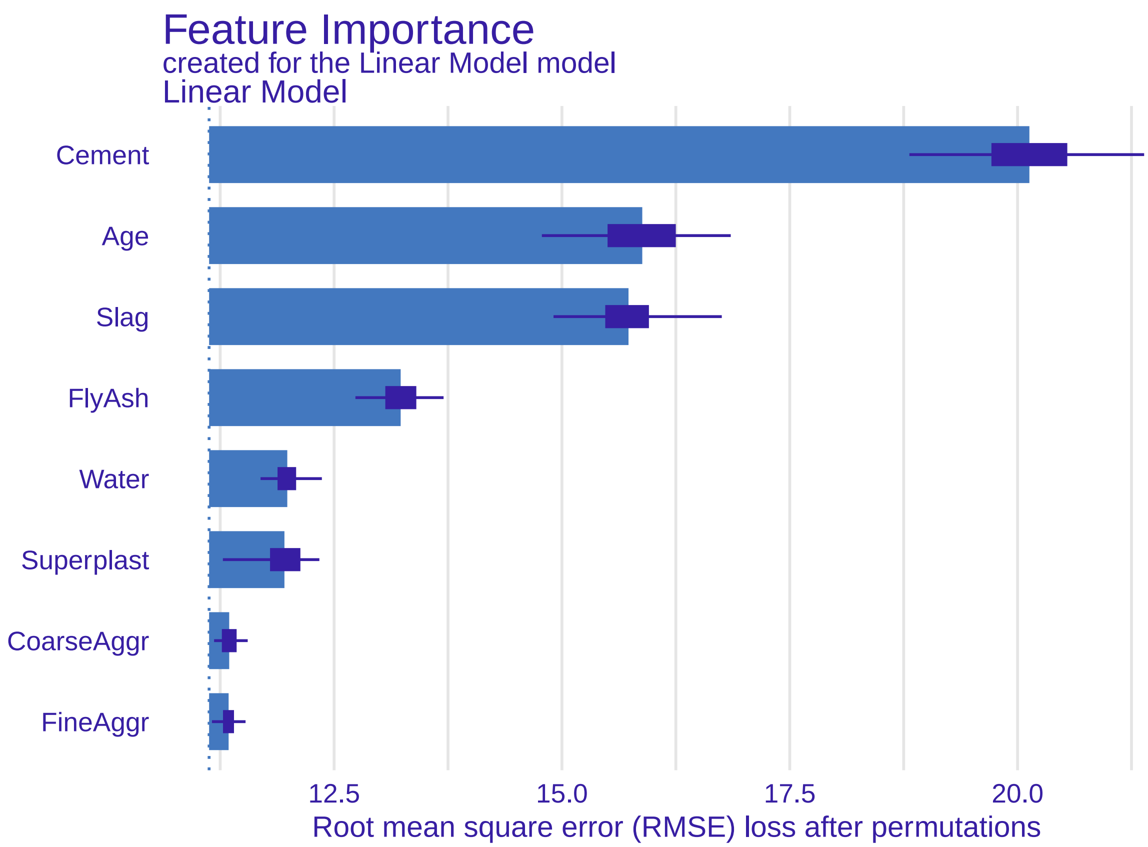
```
explainer_lm <- DALEX::explain.default(model = lm_concrete,
                              data = concrete_test[, -9],
                              y = concrete_test[, 9],
                              label = "Linear Model")
```

```
## Preparation of a new explainer is initiated
##   -> model label       :  Linear Model
##   -> data              :  330  rows  8  cols
##   -> target variable   :  330  values
##   -> predict function  :  yhat.lm  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package stats , ver. 4.3.1 , task regression (  default  )
##   -> predicted values  :  numerical, min =  10.54471 , mean =  37.61402 , max =  76.45037
##   -> residual function :  difference between y and yhat (  default  )
##   -> residuals         :  numerical, min =  -30.95065 , mean =  -1.311177 , max =  27.22631
##   A new explainer has been created!
```

```
Rnd_Perm_lm <- model_parts(
  explainer_lm,
  N = NULL, # All available data are used
  B = 100   # number of permutations to be used, with B = 10 used by default
)

Rnd_Perm_lm
```

```
##         variable mean_dropout_loss        label
## 1   _full_model_          11.12826 Linear Model
## 2       FineAggr          11.34244 Linear Model
## 3     CoarseAggr          11.34878 Linear Model
## 4     Superplast          11.95479 Linear Model
## 5          Water          11.98609 Linear Model
## 6         FlyAsh          13.23035 Linear Model
## 7           Slag          15.73062 Linear Model
## 8            Age          15.88152 Linear Model
## 9         Cement          20.12944 Linear Model
## 10    _baseline_          22.41574 Linear Model
```

```r
plot(Rnd_Perm_lm)
```



GAM model

```r
explainer_gam <- DALEX::explain.default(model = gam_concrete,
                               data = concrete_test[, -9],
                               y = concrete_test[, 9],
                               label = "GAM Model")
```

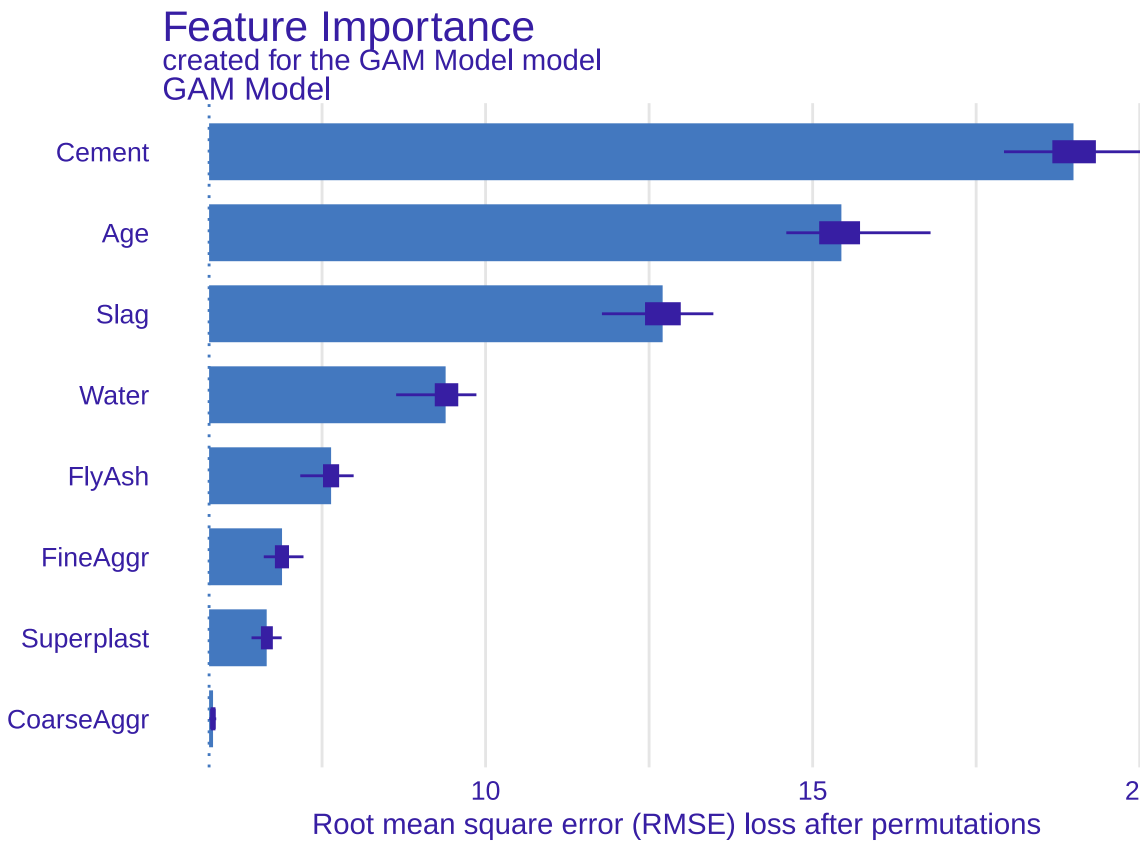
```
## Preparation of a new explainer is initiated
##   -> model label       :  GAM Model
##   -> data              :  330  rows  8  cols
##   -> target variable   :  330  values
```

```
##   -> predict function  :  yhat.glm  will be used (  default  )
##   -> predicted values  :  No value for predict function target column. (  default  )
##   -> model_info        :  package stats , ver. 4.3.1 , task regression (  default  )
##   -> predicted values  :  predict function returns multiple columns:  NA  (  default  )
##   -> residual function :  difference between y and yhat (  default  )
##   -> residuals         :  numerical, min =  -23.12587 , mean =  -0.2906517 , max =  16.36284
##   A new explainer has been created!
```

```
Rnd_Perm_gam <- model_parts(
  explainer_gam,
  N = NULL, # All available data are used
  B = 100   # number of permutations to be used, with B = 10 used by default
)

Rnd_Perm_gam
```

```
##        variable mean_dropout_loss     label
## 1  _full_model_          5.773026 GAM Model
## 2    CoarseAggr          5.833527 GAM Model
## 3    Superplast          6.653762 GAM Model
## 4      FineAggr          6.888126 GAM Model
## 5        FlyAsh          7.638170 GAM Model
## 6         Water          9.389089 GAM Model
## 7          Slag         12.706973 GAM Model
## 8           Age         15.439821 GAM Model
## 9        Cement         18.988126 GAM Model
## 10   _baseline_         23.570589 GAM Model
```

```
plot(Rnd_Perm_gam)
```



In the Random forest the importances obtained are similar to the ones obtained initially by OOB permutations, but the numbers are not actually the same. In contrast to the ghost variable method, the importance is more equally distributed among all explanatory variables.

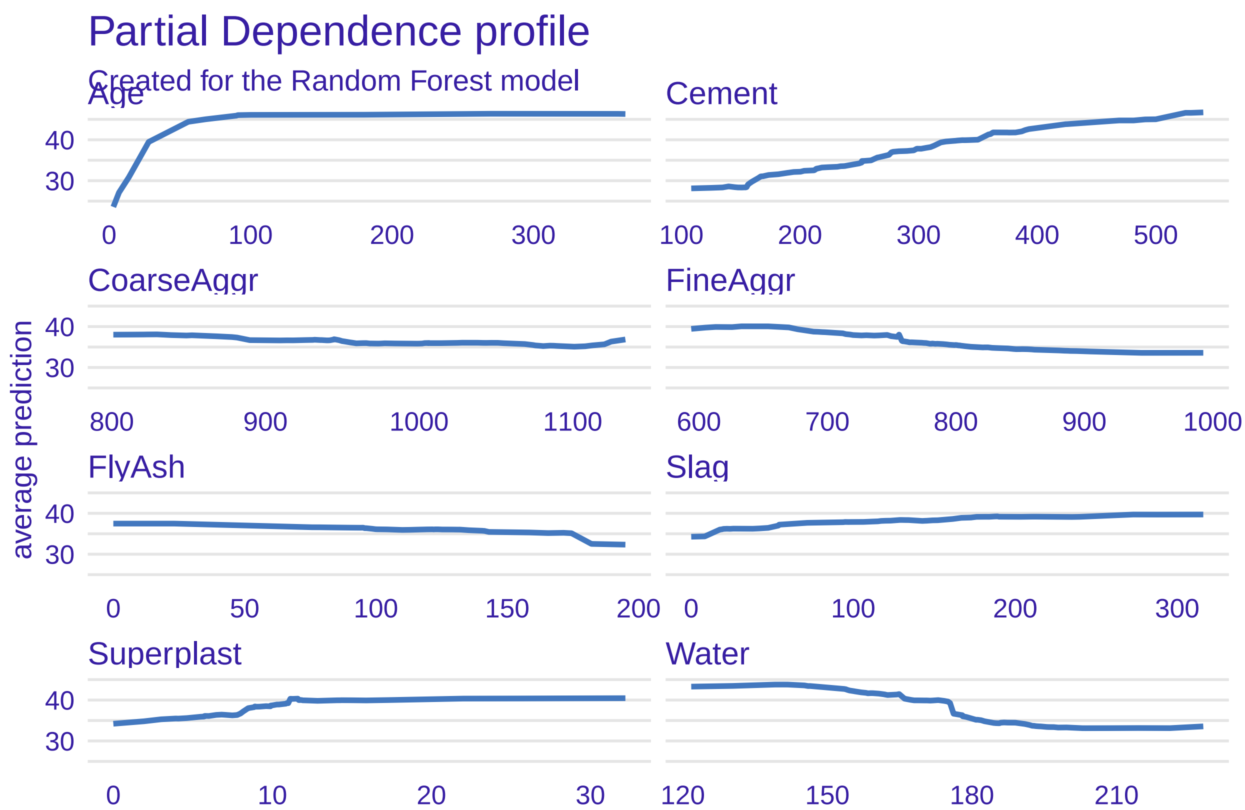
In the linear model the order of the variables by importance is different but the differences are not too severe. The values for the importance are completely different.

In the GAM the order of the variables is exactly the same and the barplots seem to be very similar.

b. Partial Dependence Plots for each explanatory variable

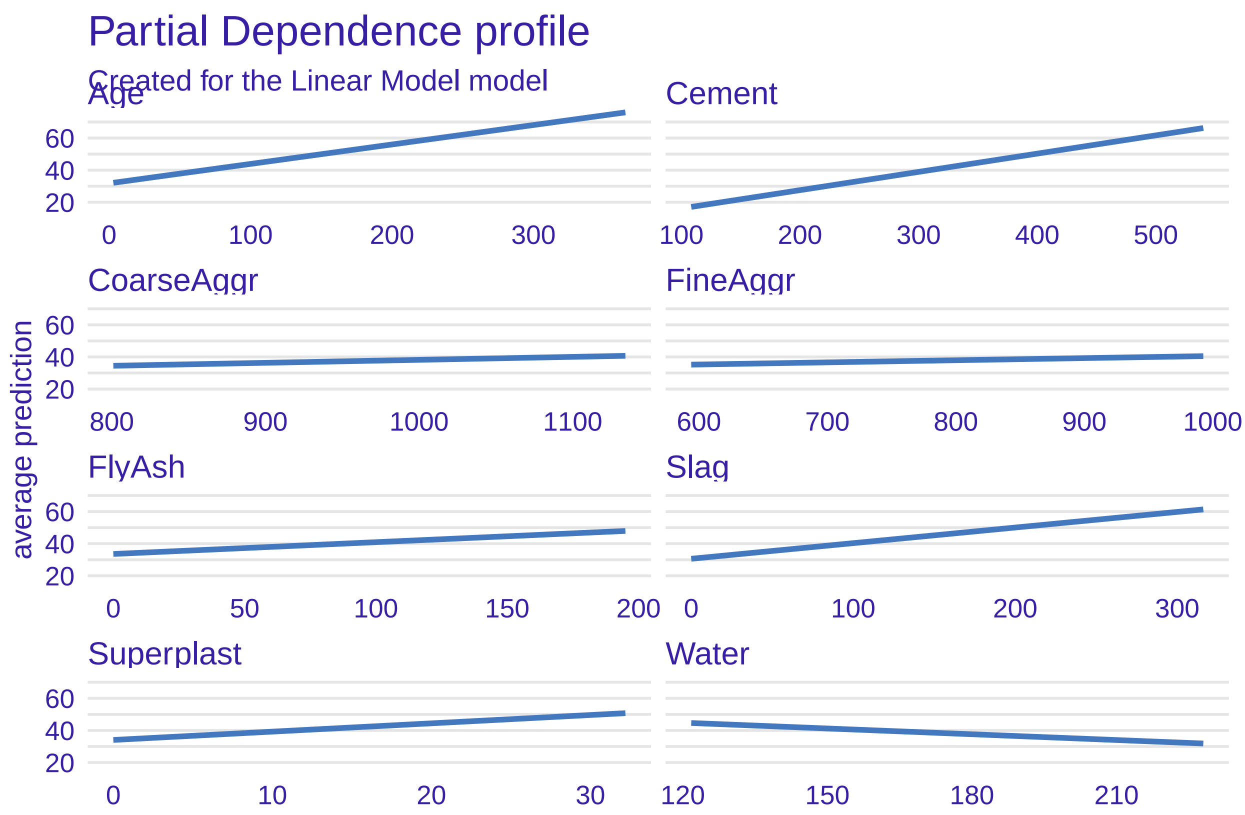
Random Forest

```
PDP_rf <- model_profile(
  explainer=explainer_rf,
  variables = NULL,  # All variables are used
  N = NULL, # All available data are used
  groups = NULL,
  k = NULL,
  center = TRUE,
  type = "partial" #  partial, conditional or accumulated
)

plot(PDP_rf, facet_ncol=2)
```



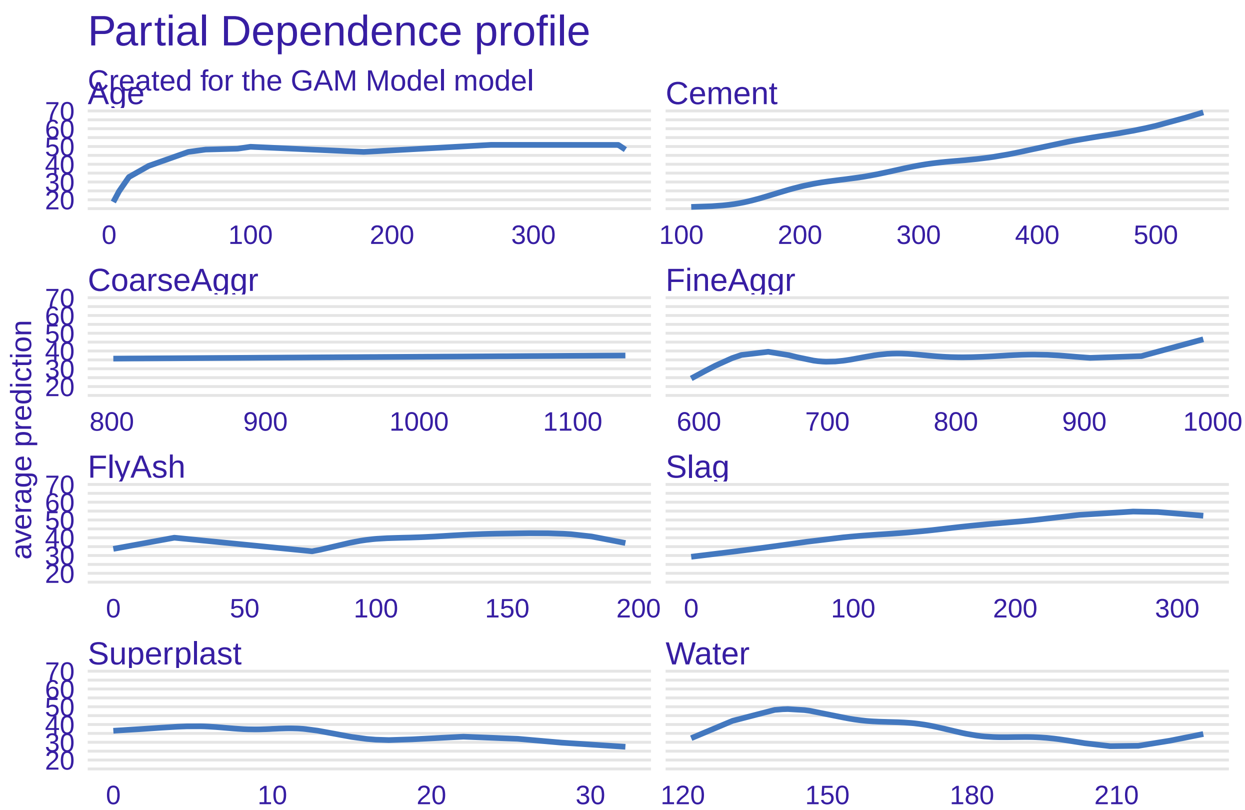
Linear model

```
PDP_lm <- model_profile(
  explainer=explainer_lm,
  variables = NULL,  # All variables are used
  N = NULL, # All available data are used
  groups = NULL,
  k = NULL,
  center = TRUE,
  type = "partial" #  partial, conditional or accumulated
)

plot(PDP_lm, facet_ncol=2)
```



GAM model

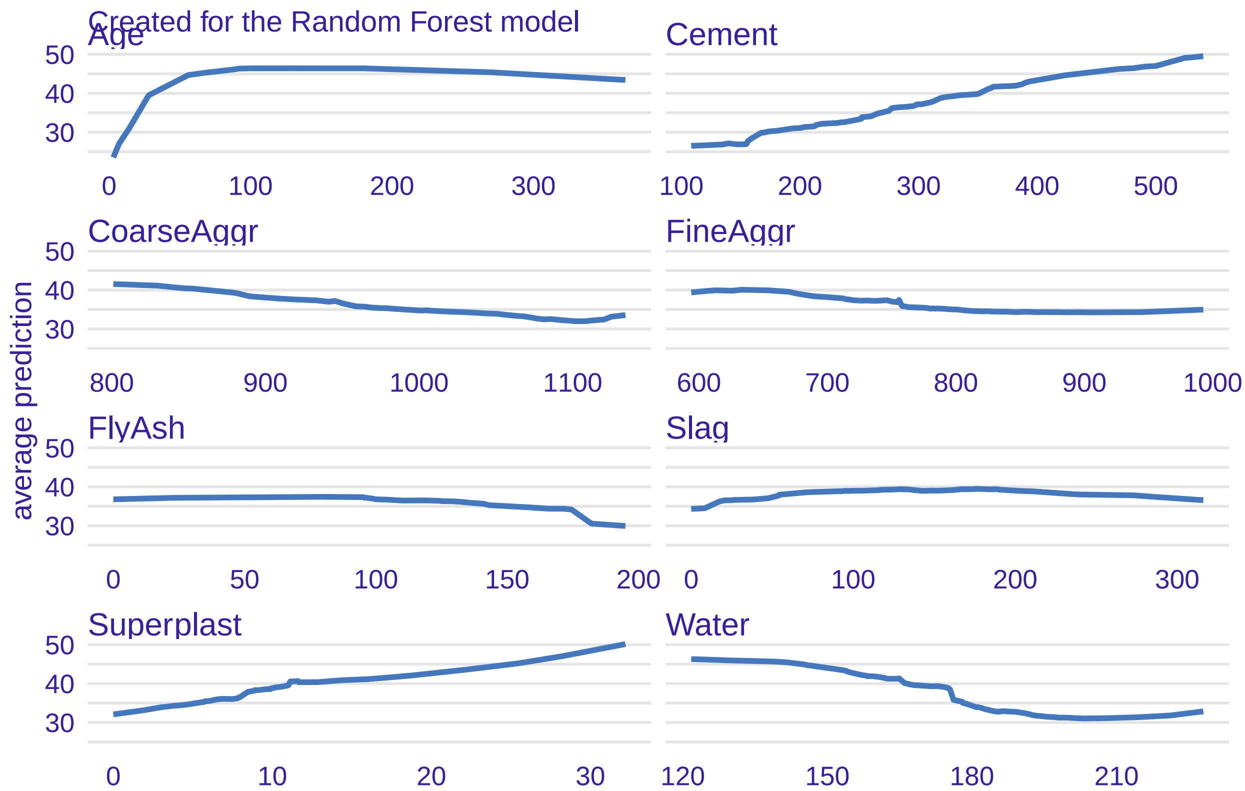
```
PDP_gam <- model_profile(
  explainer=explainer_gam,
  variables = NULL,  # All variables are used
  N = NULL, # All available data are used
  groups = NULL,
  k = NULL,
  center = TRUE,
  type = "partial" #  partial, conditional or accumulated
)

plot(PDP_gam, facet_ncol=2)
```



c. Local (or Conditional) Dependence Plots for each explanatory variable

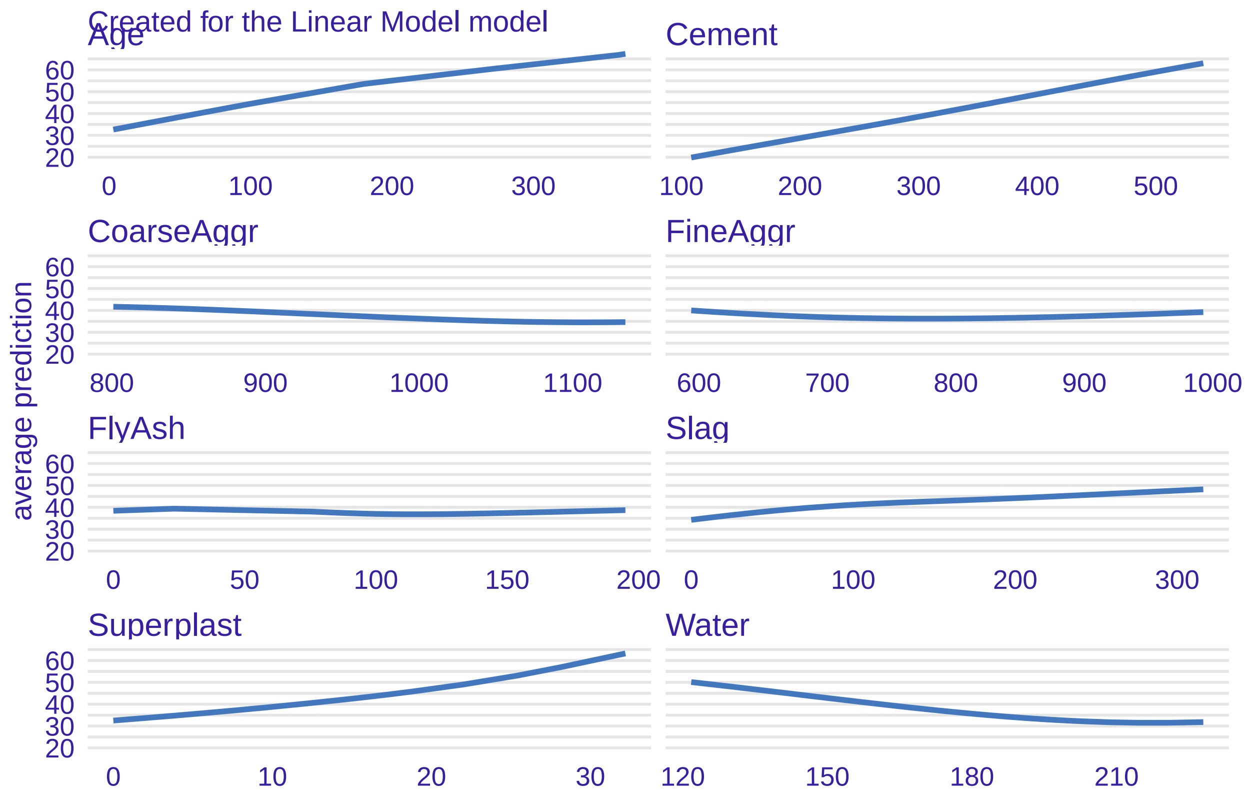
Random Forest

```
CDP_rf <- model_profile(
  explainer=explainer_rf,
  variables = NULL,  # All variables are used
  N = NULL, # All available data are used
  groups = NULL,
  k = NULL,
  center = TRUE,
  type = "conditional" #  partial, conditional or accumulated
)

plot(CDP_rf, facet_ncol=2)
```



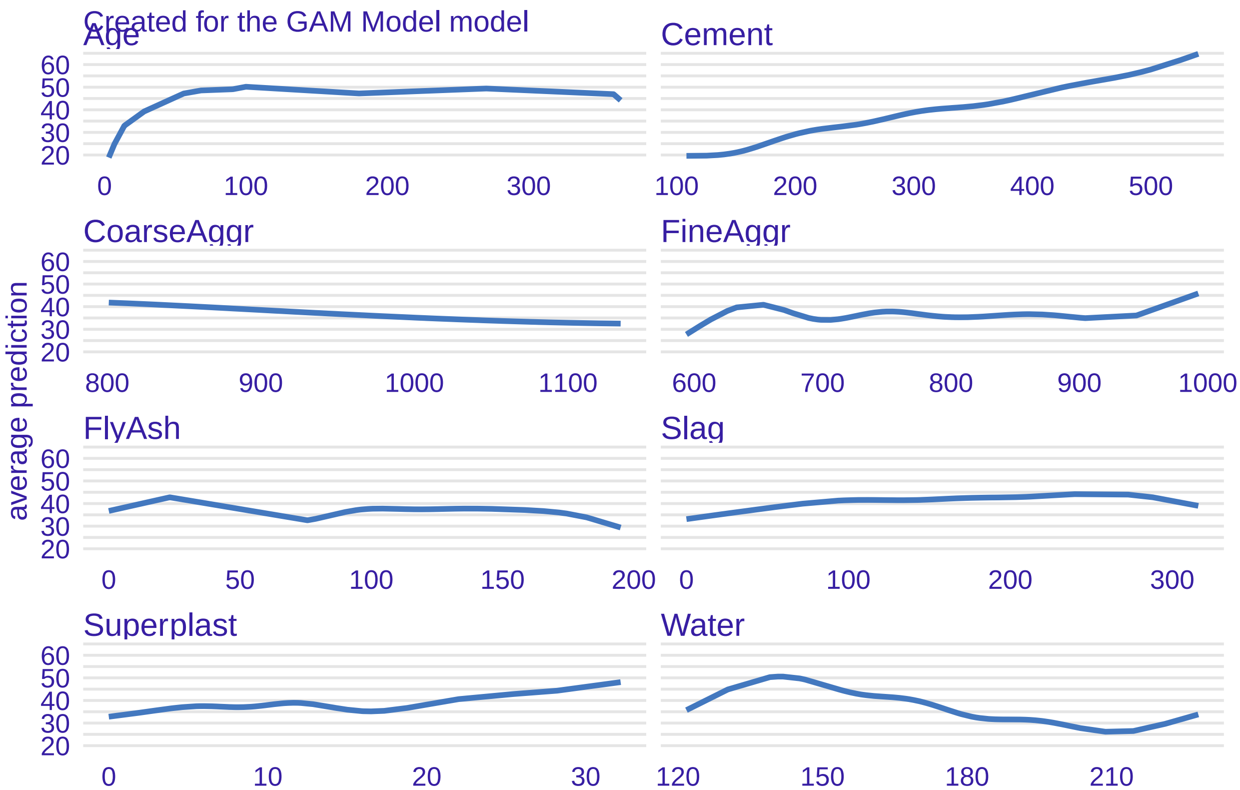
Linear Model

```r
CDP_lm <- model_profile(
  explainer=explainer_lm,
  variables = NULL,  # All variables are used
  N = NULL, # All available data are used
  groups = NULL,
  k = NULL,
  center = TRUE,
  type = "conditional" #  partial, conditional or accumulated
)

plot(CDP_lm, facet_ncol=2)
```



GAM model

```
CDP_gam <- model_profile(
  explainer=explainer_gam,
  variables = NULL,  # All variables are used
  N = NULL, # All available data are used
  groups = NULL,
  k = NULL,
  center = TRUE,
  type = "conditional" #  partial, conditional or accumulated
)

plot(CDP_gam, facet_ncol=2)
```



5. Local explainers with library DALEX

Choose two instances in the the test set, the prediction for which we want to explain:

- The data with the lowest value in Strength.
- The data with the largest value in Strength.

For these two instances, do the following tasks for the fitted random forest.

a. Explain the predictions using SHAP.

b. Explain the predictions using Break-down plots.

c. Explain the predictions using LIME.

d. Do the Individual conditional expectation (ICE) plot, or ceteris paribus plot

e. Plot in one graphic the Individual conditional expectation (ICE) plot for variable **Age** for each case in the test sample. Add the global Partial Dependence Plot.

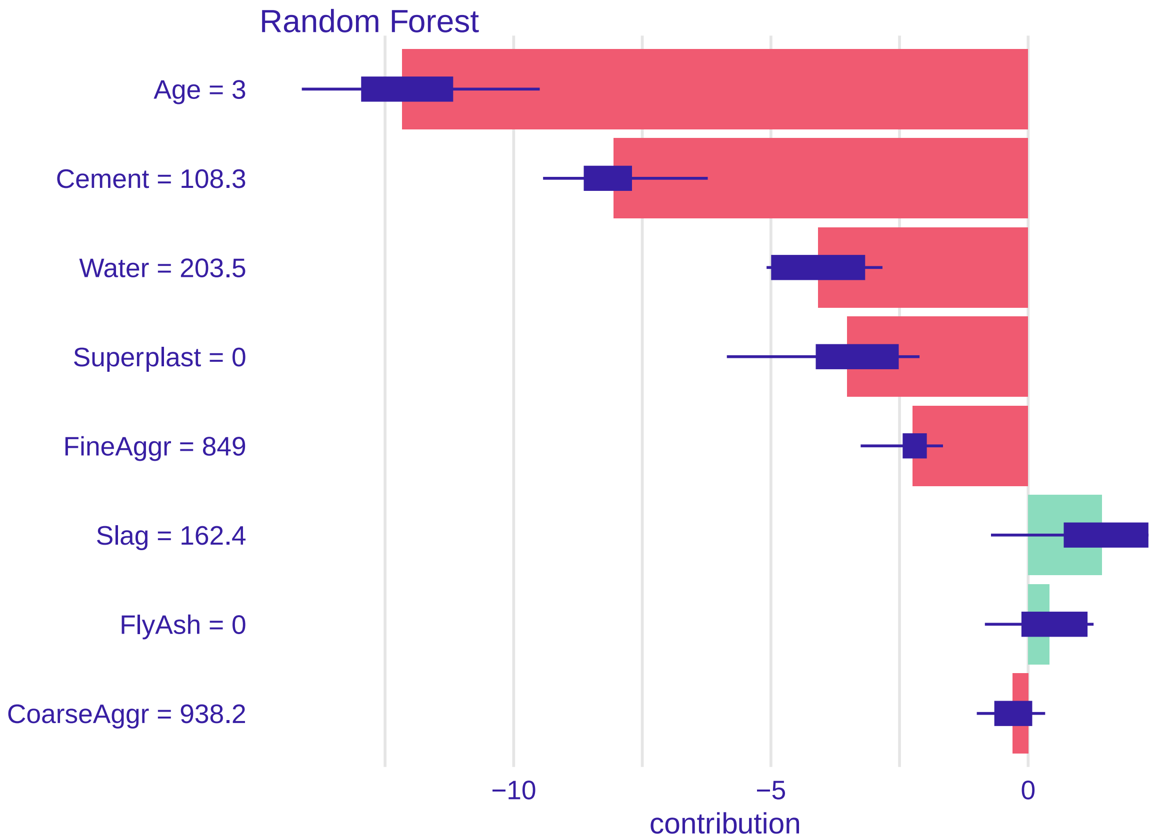
```
instance1 <- concrete_test[which.min(concrete_test$Strength),]
instance2 <- concrete_test[which.max(concrete_test$Strength),]
```

a. Explain the predictions using SHAP

```r
bd1_rf_shap <- predict_parts(explainer = explainer_rf,
                 new_observation = instance1,
                             type = "shap")
bd2_rf_shap <- predict_parts(explainer = explainer_rf,
                 new_observation = instance2,
                             type = "shap")

bd1_rf_shap
```

```
##                                              min          q1      median
## Random Forest: Age = 3             -14.1180353 -12.9371090 -12.4266835
## Random Forest: Cement = 108.3       -9.4282986  -8.6103402  -8.1410799
## Random Forest: CoarseAggr = 938.2   -0.9978235  -0.6301933  -0.2494229
## Random Forest: FineAggr = 849       -3.2554939  -2.4115014  -2.2318796
## Random Forest: FlyAsh = 0           -0.8411610  -0.1029513   0.4206067
## Random Forest: Slag = 162.4         -0.7223355   0.7192818   1.8450264
## Random Forest: Superplast = 0       -5.8561482  -4.1007625  -3.1553449
## Random Forest: Water = 203.5        -5.0843228  -4.9676995  -4.0095873
##                                           mean           q3         max
## Random Forest: Age = 3             -12.1664736 -11.20360465 -9.4935327
## Random Forest: Cement = 108.3       -8.0526665  -7.72783679 -6.2271854
## Random Forest: CoarseAggr = 938.2   -0.3050038   0.05168216  0.3306349
## Random Forest: FineAggr = 849       -2.2426705  -1.99760438 -1.6554586
## Random Forest: FlyAsh = 0            0.4095634   1.12656822  1.2717588
## Random Forest: Slag = 162.4          1.4340497   2.30989347  2.3401692
## Random Forest: Superplast = 0       -3.5174278  -2.54306643 -2.1115806
## Random Forest: Water = 203.5        -4.0809820  -3.19609295 -2.8318249
```

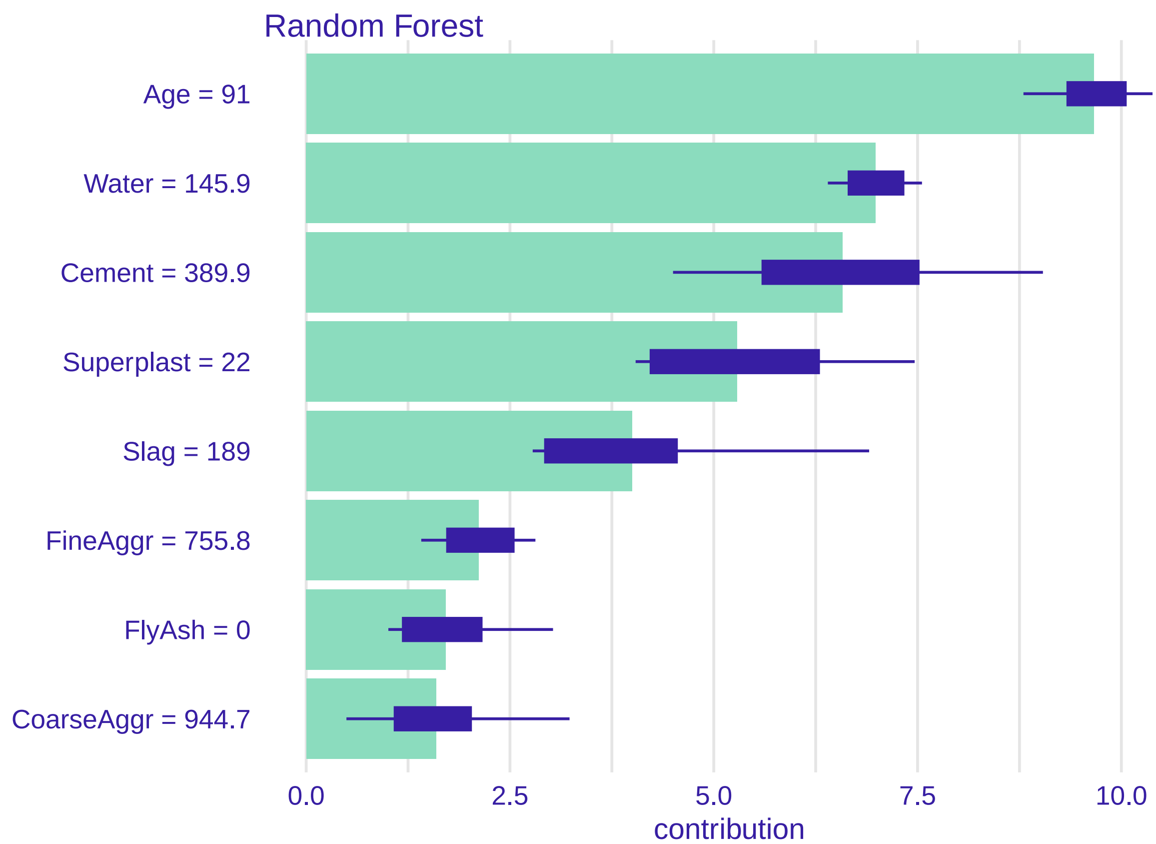
```r
plot(bd1_rf_shap)
```




```r
bd2_rf_shap
```

```
##                                            min        q1   median     mean
## Random Forest: Age = 91              8.7986567 9.343611 9.709325 9.660564
## Random Forest: Cement = 389.9        4.5002276 5.603113 6.542393 6.578458
## Random Forest: CoarseAggr = 944.7 0.4938351 1.091370 1.508825 1.593811
## Random Forest: FineAggr = 755.8      1.4118120 1.734763 2.011204 2.119263
## Random Forest: FlyAsh = 0            1.0083154 1.192000 1.579270 1.712936
## Random Forest: Slag = 189            2.7786728 2.936385 3.447459 3.996077
## Random Forest: Superplast = 22       4.0410776 4.230723 4.632055 5.288286
## Random Forest: Water = 145.9         6.3995013 6.659594 6.949934 6.985484
##                                             q3       max
## Random Forest: Age = 91              10.047633 10.382364
## Random Forest: Cement = 389.9         7.506589  9.037668
## Random Forest: CoarseAggr = 944.7    2.015647  3.230706
## Random Forest: FineAggr = 755.8       2.539157  2.812248
## Random Forest: FlyAsh = 0             2.146140  3.028230
## Random Forest: Slag = 189             4.541481  6.905890
## Random Forest: Superplast = 22        6.284630  7.464180
## Random Forest: Water = 145.9          7.320715  7.553828
```

```r
plot(bd2_rf_shap)
```

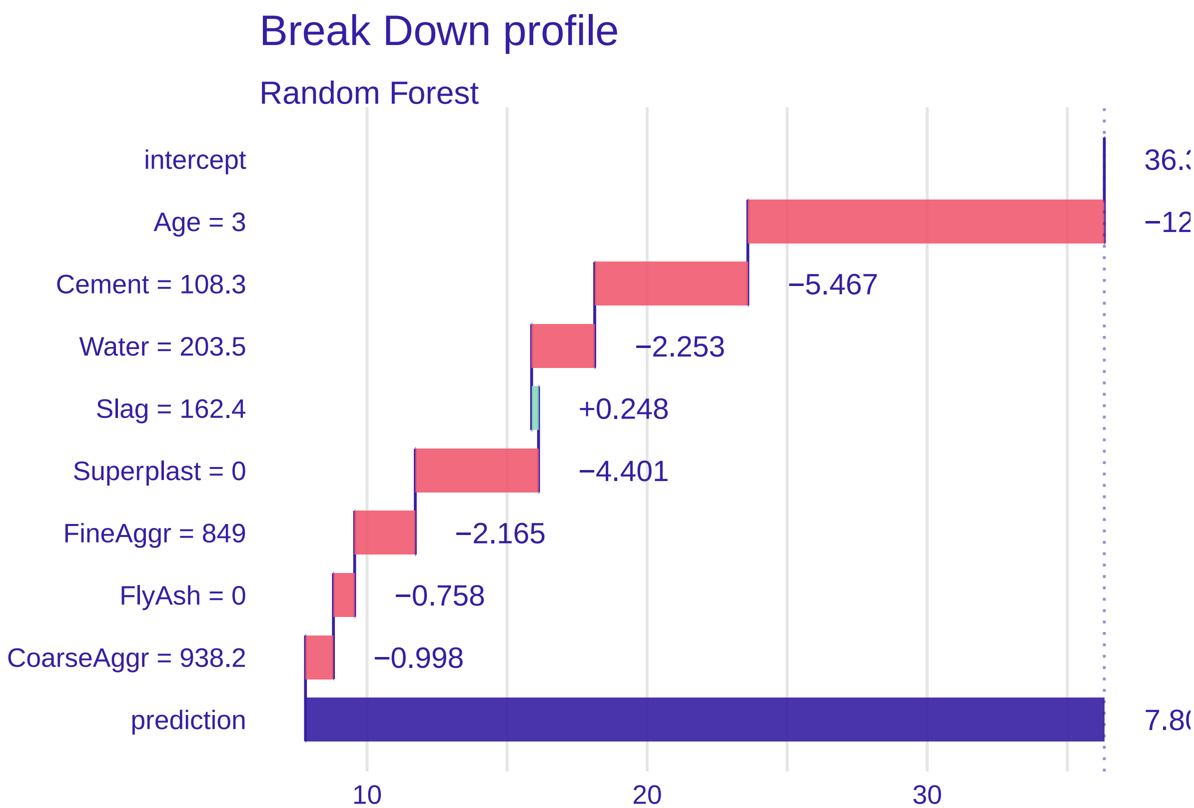


b. Explain the predictions using Break-down plots

```
bd1_rf_bd <- predict_parts(explainer = explainer_rf,
                   new_observation = instance1,
                              type = "break_down")
bd2_rf_bd <- predict_parts(explainer = explainer_rf,
                   new_observation = instance2,
                              type = "break_down")

bd1_rf_bd
```

```
##                                    contribution
## Random Forest: intercept                 36.323
## Random Forest: Age = 3                  -12.728
## Random Forest: Cement = 108.3            -5.467
## Random Forest: Water = 203.5             -2.253
## Random Forest: Slag = 162.4               0.248
## Random Forest: Superplast = 0            -4.401
## Random Forest: FineAggr = 849            -2.165
## Random Forest: FlyAsh = 0                -0.758
## Random Forest: CoarseAggr = 938.2        -0.998
## Random Forest: prediction                 7.801
```

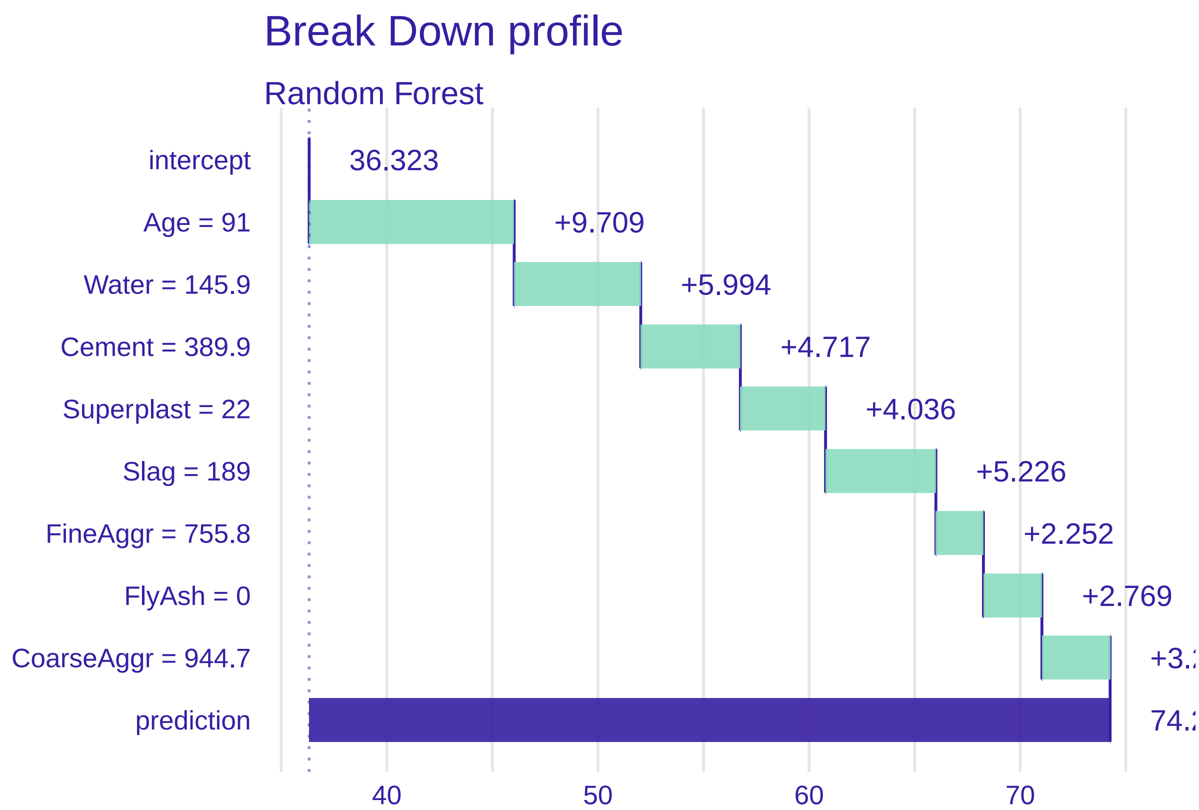
```r
plot(bd1_rf_bd)
```




```r
bd2_rf_bd
```

```
##                                         contribution
## Random Forest: intercept                      36.323
## Random Forest: Age = 91                        9.709
## Random Forest: Water = 145.9                   5.994
## Random Forest: Cement = 389.9                  4.717
## Random Forest: Superplast = 22                 4.036
## Random Forest: Slag = 189                      5.226
## Random Forest: FineAggr = 755.8                2.252
## Random Forest: FlyAsh = 0                      2.769
## Random Forest: CoarseAggr = 944.7              3.231
## Random Forest: prediction                     74.258
```

```r
plot(bd2_rf_bd)
```



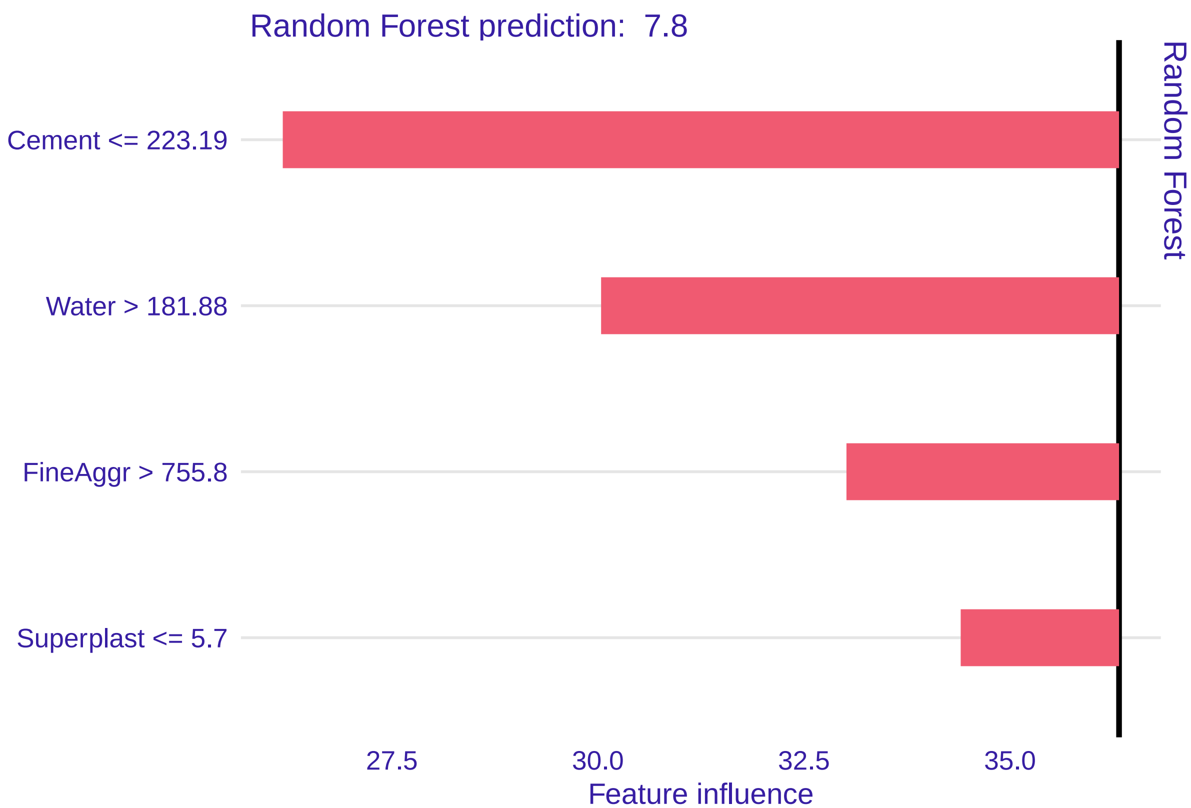
c. Explain the predictions using LIME

```
lime1_rf <- predict_surrogate(explainer = explainer_rf,
                  new_observation = instance1[,-9],
                  type = "localModel")
lime2_rf <- predict_surrogate(explainer = explainer_rf,
                  new_observation = instance2[,-9],
                  type = "localModel")


lime1_rf
```

```
##    estimated          variable original_variable dev_ratio response
## 1  36.323067      (Model mean)                   0.4728502
## 2  44.896012       (Intercept)                   0.4728502
## 3 -10.146481   Cement <= 223.19           Cement 0.4728502
## 4   0.000000   FlyAsh <= 118.27           FlyAsh 0.4728502
## 5  -6.284263     Water > 181.88            Water 0.4728502
## 6  -1.921495 Superplast <= 5.7        Superplast 0.4728502
##   predicted_value         model
## 1        7.801456 Random Forest
## 2        7.801456 Random Forest
## 3        7.801456 Random Forest
## 4        7.801456 Random Forest
```

```
## 5         7.801456 Random Forest
## 6         7.801456 Random Forest
```

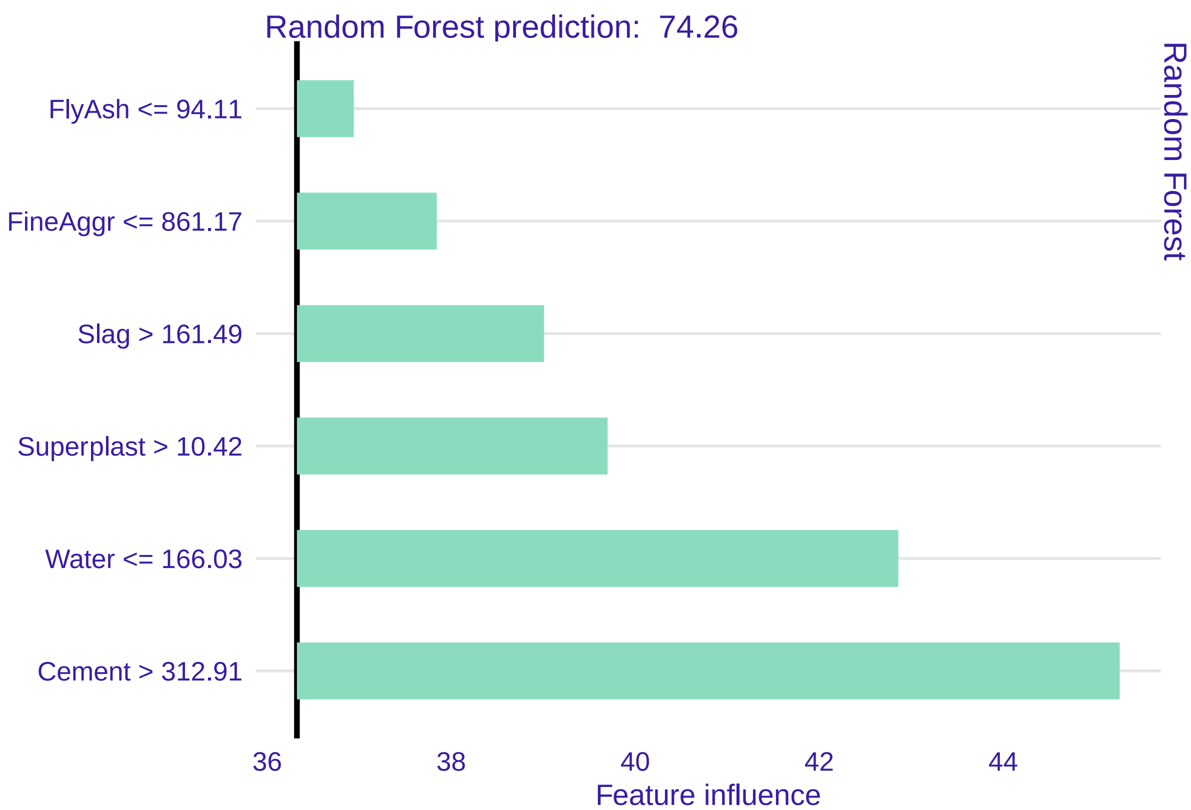
```r
plot(lime1_rf)
```




```r
lime2_rf
```

```
##    estimated        variable original_variable dev_ratio response
## 1 36.3230673    (Model mean)                   0.3948498
## 2 31.2424859     (Intercept)                   0.3948498
## 3  8.9436271 Cement > 312.91            Cement 0.3948498
## 4  2.6861177   Slag > 161.49              Slag 0.3948498
## 5  0.6187084 FlyAsh <= 94.11            FlyAsh 0.3948498
## 6  6.5371155 Water <= 166.03             Water 0.3948498
##   predicted_value         model
## 1        74.25795 Random Forest
## 2        74.25795 Random Forest
## 3        74.25795 Random Forest
## 4        74.25795 Random Forest
## 5        74.25795 Random Forest
## 6        74.25795 Random Forest
```

```r
plot(lime2_rf)
```



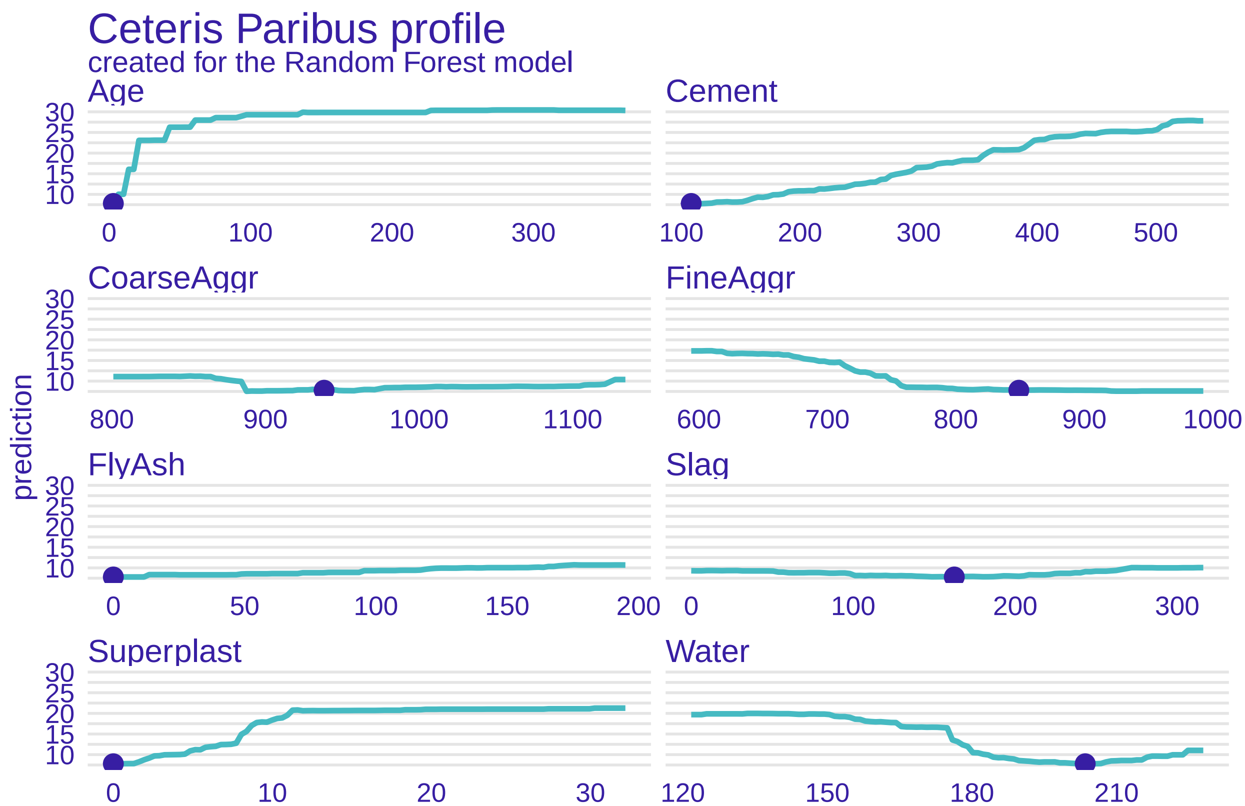
d. Individual conditional expectation (ICE) plot, or ceteris paribus plot

```
cp1_rf <- predict_profile(explainer = explainer_rf,
                          new_observation = instance1)
cp2_rf <- predict_profile(explainer = explainer_rf,
                          new_observation = instance2)
cp1_rf
```

```
## Top profiles    :
##         Cement  Slag FlyAsh Water Superplast CoarseAggr FineAggr Age    _yhat_
## 689    108.300 162.4      0 203.5          0      938.2      849   3 7.801456
## 689.1  112.617 162.4      0 203.5          0      938.2      849   3 7.769953
## 689.2  116.934 162.4      0 203.5          0      938.2      849   3 7.743863
## 689.3  121.251 162.4      0 203.5          0      938.2      849   3 7.809065
## 689.4  125.568 162.4      0 203.5          0      938.2      849   3 7.873280
## 689.5  129.885 162.4      0 203.5          0      938.2      849   3 8.137848
##        _vname_ _ids_        _label_
## 689     Cement   689 Random Forest
## 689.1   Cement   689 Random Forest
## 689.2   Cement   689 Random Forest
## 689.3   Cement   689 Random Forest
## 689.4   Cement   689 Random Forest
## 689.5   Cement   689 Random Forest
##
```

```
##
## Top observations:
##     Cement  Slag FlyAsh Water Superplast CoarseAggr FineAggr Age    _yhat_
## 689  108.3 162.4      0 203.5          0      938.2      849   3 7.801456
##          _label_ _ids_
## 689 Random Forest     1
```

```r
plot(cp1_rf, facet_ncol=2)
```

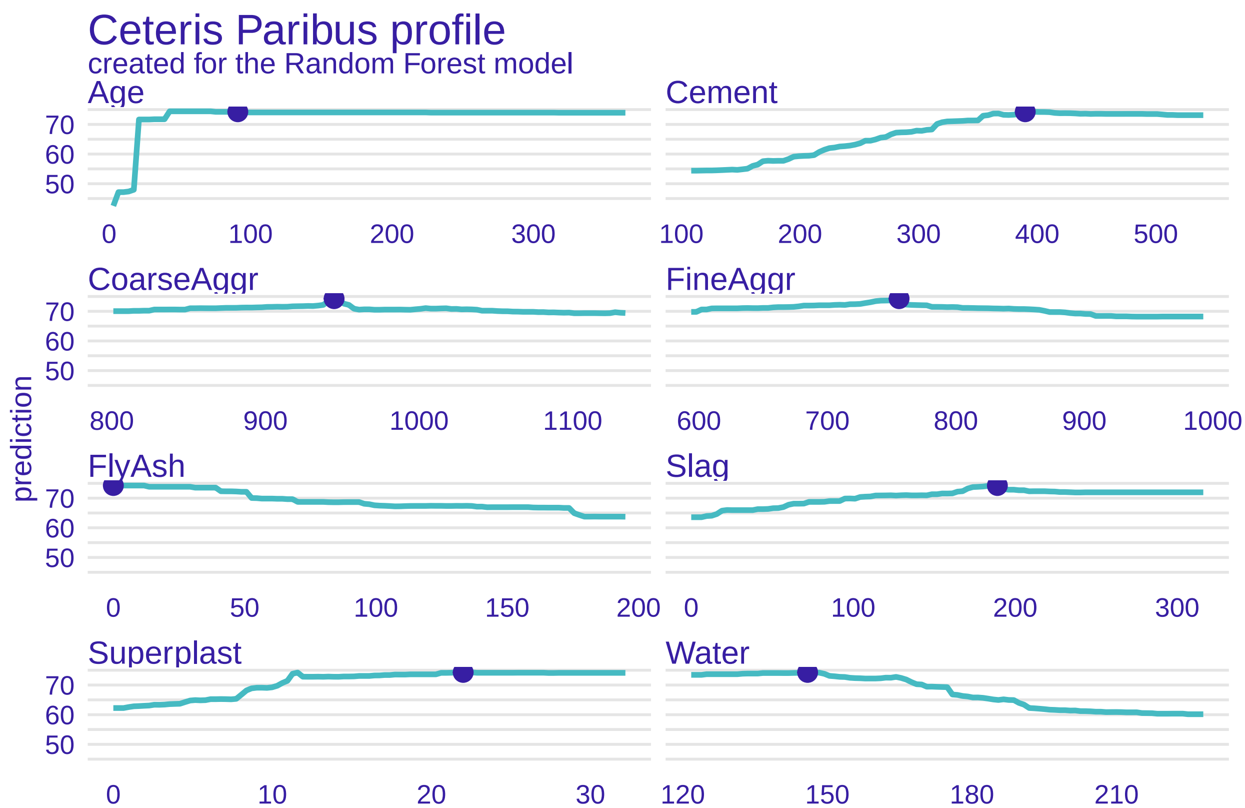



```
cp2_rf
```

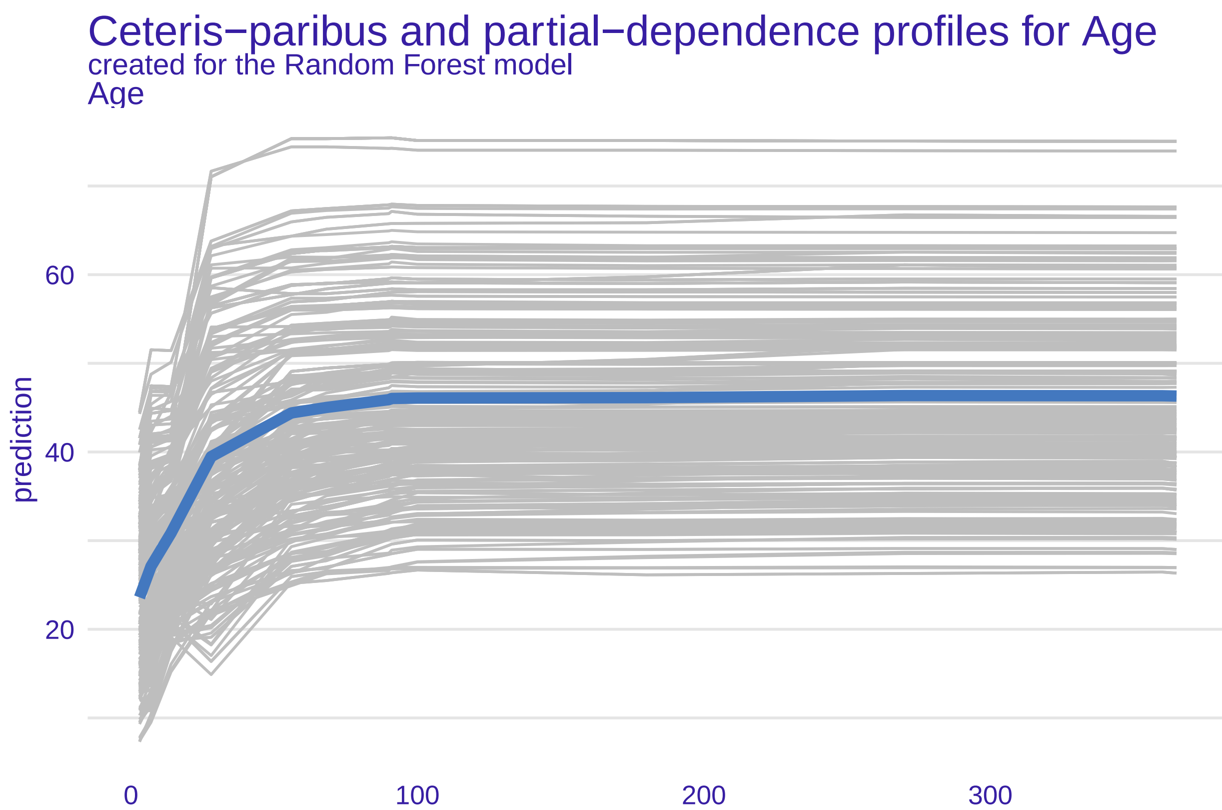
```
## Top profiles    :
##         Cement Slag FlyAsh Water Superplast CoarseAggr FineAggr Age    _yhat_
## 182    108.300  189      0 145.9         22      944.7    755.8  91 54.39946
## 182.1 112.617  189      0 145.9         22      944.7    755.8  91 54.39946
## 182.2 116.934  189      0 145.9         22      944.7    755.8  91 54.43597
## 182.3 121.251  189      0 145.9         22      944.7    755.8  91 54.46867
## 182.4 125.568  189      0 145.9         22      944.7    755.8  91 54.46867
## 182.5 129.885  189      0 145.9         22      944.7    755.8  91 54.52146
##        _vname_ _ids_        _label_
## 182     Cement   182 Random Forest
## 182.1   Cement   182 Random Forest
## 182.2   Cement   182 Random Forest
## 182.3   Cement   182 Random Forest
## 182.4   Cement   182 Random Forest
```

```
## 182.5   Cement    182 Random Forest
##
##
## Top observations:
##     Cement Slag FlyAsh Water Superplast CoarseAggr FineAggr Age    _yhat_
## 182  389.9  189      0 145.9         22      944.7    755.8  91 74.25795
##            _label_ _ids_
## 182 Random Forest     1
```

```r
plot(cp2_rf, facet_ncol=2)
```




```r
mp_rf <- model_profile(explainer = explainer_rf,
  variables = "Age",
  N = NULL,
  type = "partial"
)

plot(mp_rf, geom = "profiles") +
  ggtitle("Ceteris-paribus and partial-dependence profiles for Age")
```



Ceteris−paribus and partial−dependence profiles for Age
created for the Random Forest model
Age
prediction
20
40
60
0
100
200
300