Exercise set 3

50537

```r
candidate_number <- 50537
set.seed(candidate_number)
```

```r
#### Simulation exercise

### Data generation

set.seed(50537)
# the data set I have created will be assessing ship quality based on different aspects

n <- 2000
## Generate categorical predictors
wood <- c("oak", "walnut", "spruce")
wood_var <- factor(sample(wood, n, prob = c(7, 4, 2), replace = TRUE))

metal <- c("steel", "iron", "brass")
metal_var <- factor(sample(metal, n, prob = c(4, 5, 3), replace = TRUE))

propulsion <- c("wind", "coal", "electric")
prop_var <- factor(sample(propulsion, n, prob = c(3, 4, 1), replace = TRUE))

table(wood_var)
```

```
## wood_var
##    oak spruce walnut
##   1104    292    604
```

```r
table(metal_var)
```

```
## metal_var
## brass  iron steel
##   513   843   644
```

```r
table(prop_var)
```

```
## prop_var
##     coal electric     wind
##      990      264      746
```

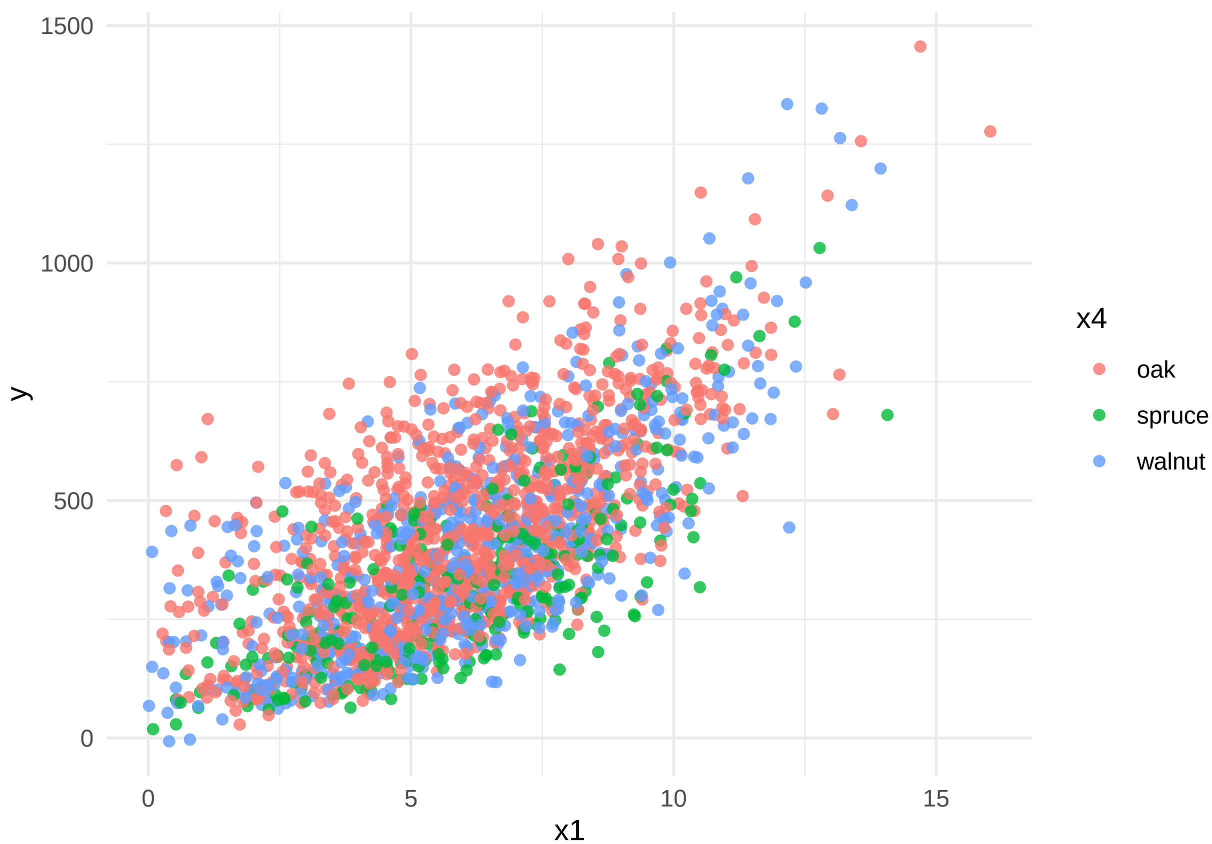
```r
set.seed(50537)
```

```r
## Generate numeric predictors
strength <- abs(rnorm(2000, mean=6, sd=2.5))
speed <- runif(2000, min=1, max=10)
agility <- 0.5 * runif(2000, min=1, max=10)

## Data frame version of predictors
predictors <- data.frame(x1 = strength, x2 = speed, x3 = agility,
                         x4 = wood_var, x5 = metal_var, x6 = prop_var)
head(predictors)
```

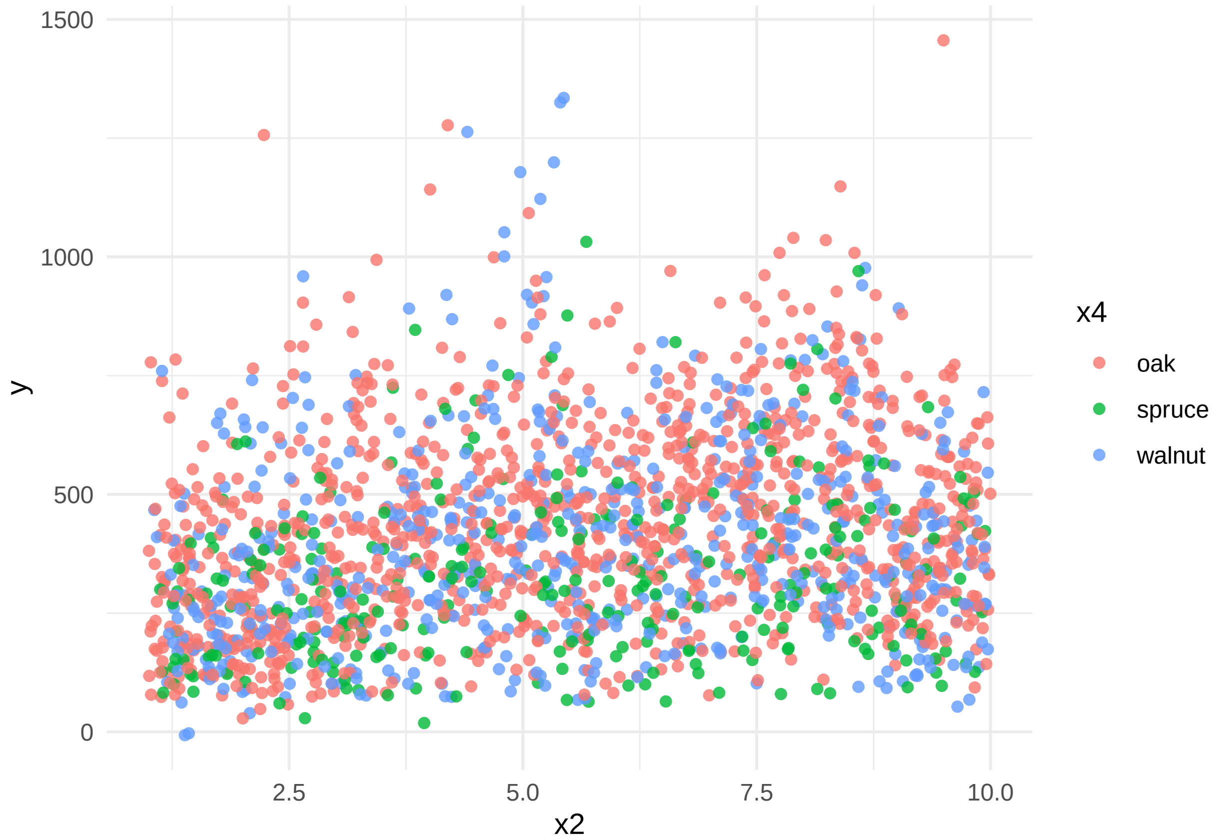
```
##         x1       x2       x3     x4    x5   x6
## 1 5.340353 4.932604 2.548004    oak brass coal
## 2 5.603661 4.199933 2.179384 spruce steel coal
## 3 5.825070 3.427621 1.863447    oak steel coal
## 4 9.715069 4.204016 3.070870 walnut steel coal
## 5 7.633863 1.613893 2.986042    oak  iron coal
## 6 7.323944 2.155077 3.308637    oak brass coal
```

```r
set.seed(50537)

### CEF
CEF <- function(x1, x2, x3, x4, x5, x6) {

  # oak > walnut > spruce
  # iron > steel > brass
  # coal > wind > electric

  case_when(
    x4=="oak" & x5=="iron" & x6 == "coal" ~ 6*x1^2 + 4*x2 + 20*x2*x3 + 5,
    x4=="oak" & x5=="iron" & x6 == "wind" ~ 5*x1^2 + 3*x2 + 16*x2*x3 + 7.5,
    x4=="oak" & x5=="iron" & x6 == "electric" ~ 4*x1^2 + 2*x2 + 12*x2*x3 + 10,
    x4=="oak" & x5=="steel" & x6 == "coal" ~ 5*x1^2 + 3.5*x2 + 18*x2*x3 + 5,
    x4=="oak" & x5=="steel" & x6 == "wind" ~ 4*x1^2 + 2.5*x2 + 14*x2*x3 + 7.5,
    x4=="oak" & x5=="steel" & x6 == "electric" ~ 3*x1^2 + 1.5*x2 + 10*x2*x3 + 10,
    x4=="oak" & x5=="brass" & x6 == "coal" ~ 4*x1^2 + 3*x2 + 16*x2*x3 + 5,
    x4=="oak" & x5=="brass" & x6 == "wind" ~ 3*x1^2 + 2*x2 + 12*x2*x3 + 7.5,
    x4=="oak" & x5=="brass" & x6 == "electric" ~ 2*x1^2 + x2 + 8*x2*x3 + 10,
    x4=="walnut" & x5=="iron" & x6 == "coal" ~ 5.5*x1^2 + 4*x2 + 18*x2*x3 + 2.5,
    x4=="walnut" & x5=="iron" & x6 == "wind" ~ 4.5*x1^2 + 3*x2 + 14*x2*x3 + 5,
    x4=="walnut" & x5=="iron" & x6 == "electric" ~ 3.5*x1^2 + 2*x2 + 10*x2*x3 + 7.5,
    x4=="walnut" & x5=="steel" & x6 == "coal" ~ 4.5*x1^2 + 3.5*x2 + 16*x2*x3 + 2.5,
    x4=="walnut" & x5=="steel" & x6 == "wind" ~ 3.5*x1^2 + 2.5*x2 + 12*x2*x3 + 5,
    x4=="walnut" & x5=="steel" & x6 == "electric" ~ 2.5*x1^2 + 1.5*x2 + 8*x2*x3 + 7.5,
    x4=="walnut" & x5=="brass" & x6 == "coal" ~ 3.5*x1^2 + 3*x2 + 14*x2*x3 + 2.5,
    x4=="walnut" & x5=="brass" & x6 == "wind" ~ 2.5*x1^2 + 2*x2 + 10*x2*x3 + 5,
    x4=="walnut" & x5=="brass" & x6 == "electric" ~ 1.5*x1^2 + x2 + 6*x2*x3 + 7.5,
    x4=="spruce" & x5=="iron" & x6 == "coal" ~ 5*x1^2 + 4*x2 + 16*x2*x3,
    x4=="spruce" & x5=="iron" & x6 == "wind" ~ 4*x1^2 + 3*x2 + 12*x2*x3 + 2.5,
    x4=="spruce" & x5=="iron" & x6 == "electric" ~ 3*x1^2 + 2*x2 + 8*x2*x3 + 5,
    x4=="spruce" & x5=="steel" & x6 == "coal" ~ 4*x1^2 + 3.5*x2 + 14*x2*x3,
    x4=="spruce" & x5=="steel" & x6 == "wind" ~ 3*x1^2 + 2.5*x2 + 10*x2*x3 + 2.5,
    x4=="spruce" & x5=="steel" & x6 == "electric" ~ 2*x1^2 + 1.5*x2 + 6*x2*x3 + 5,
    x4=="spruce" & x5=="brass" & x6 == "coal" ~ 3*x1^2 + 3*x2 + 12*x2*x3,
```

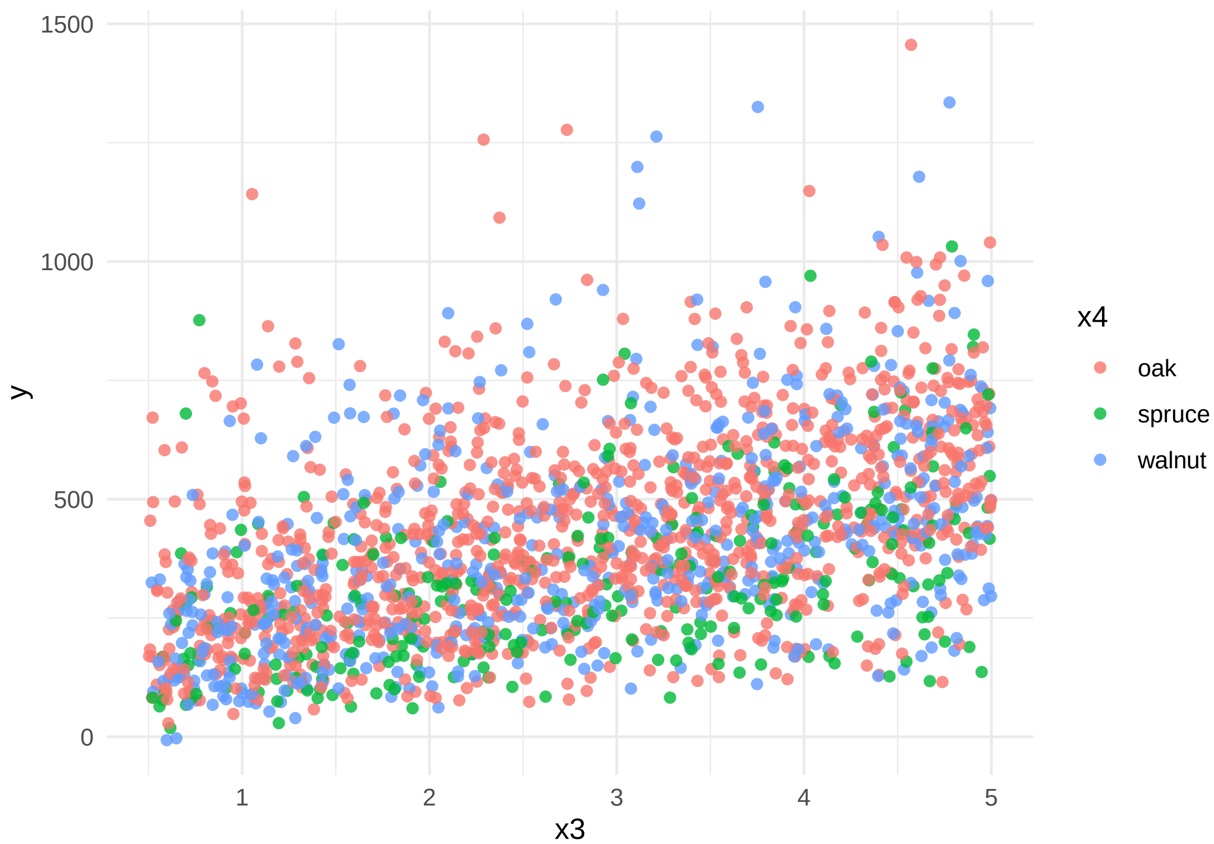
```
    x4=="spruce" & x5=="brass" & x6 == "wind" ~ 2*x1^2 + 2*x2 + 8*x2*x3 + 2.5,
    x4=="spruce" & x5=="brass" & x6 == "electric" ~ x1^2 + x2 + 4*x2*x3 + 5
  )

}

training_data <- predictors |> mutate(y = CEF(x1, x2, x3, x4, x5, x6) + rnorm(n, sd = 12)) # adding noi
# y represents "ship quality"

ggplot(training_data, aes(x1, y, colour = x4)) + geom_point(alpha = .8)
```



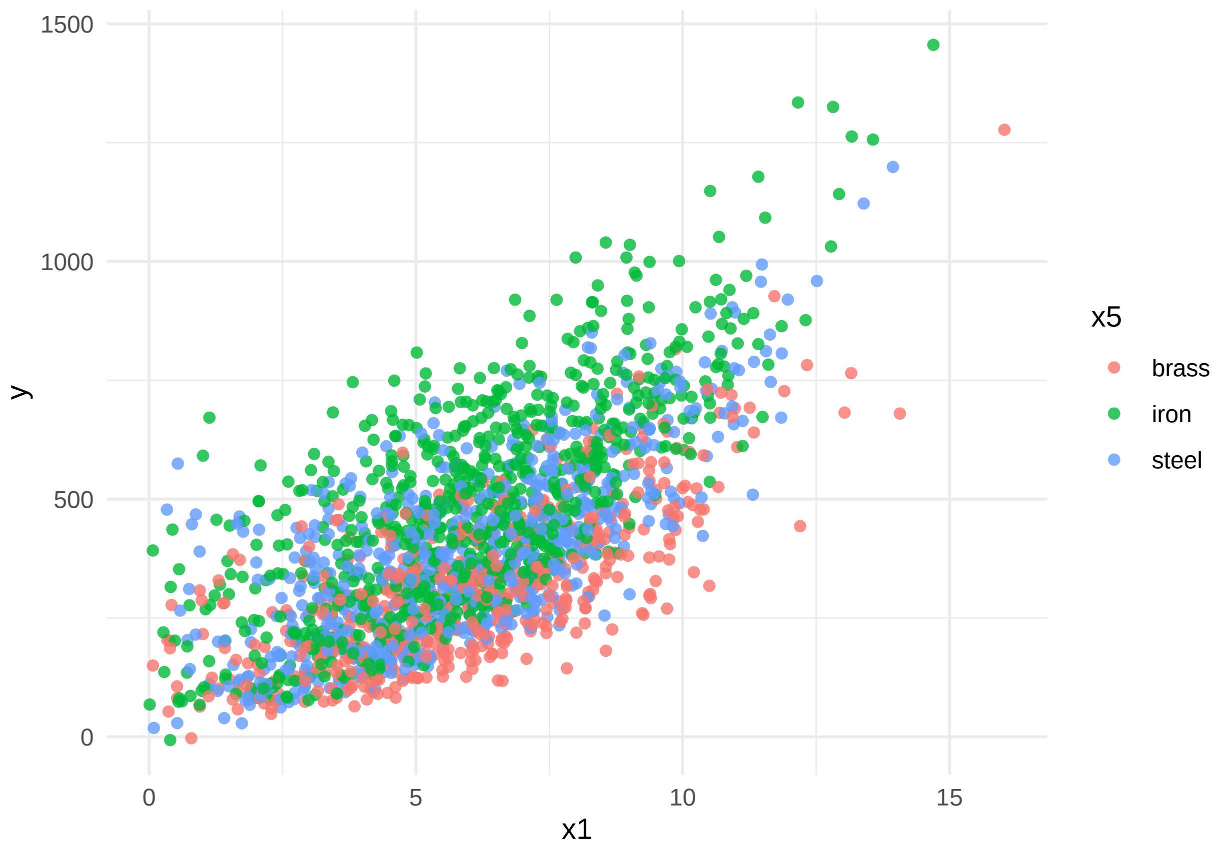

```
ggplot(training_data, aes(x2, y, colour = x4)) + geom_point(alpha = .8)
```



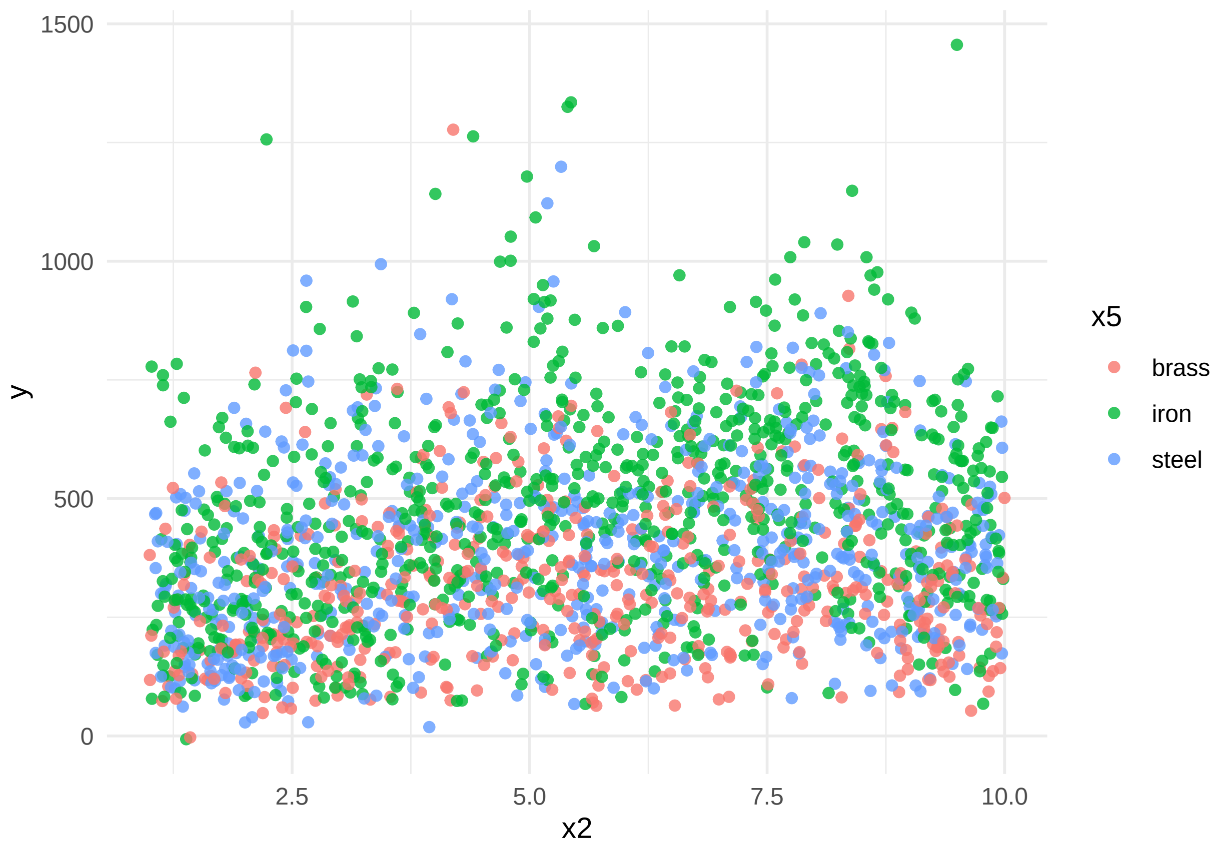

```r
ggplot(training_data, aes(x3, y, colour = x4)) + geom_point(alpha = .8)
```



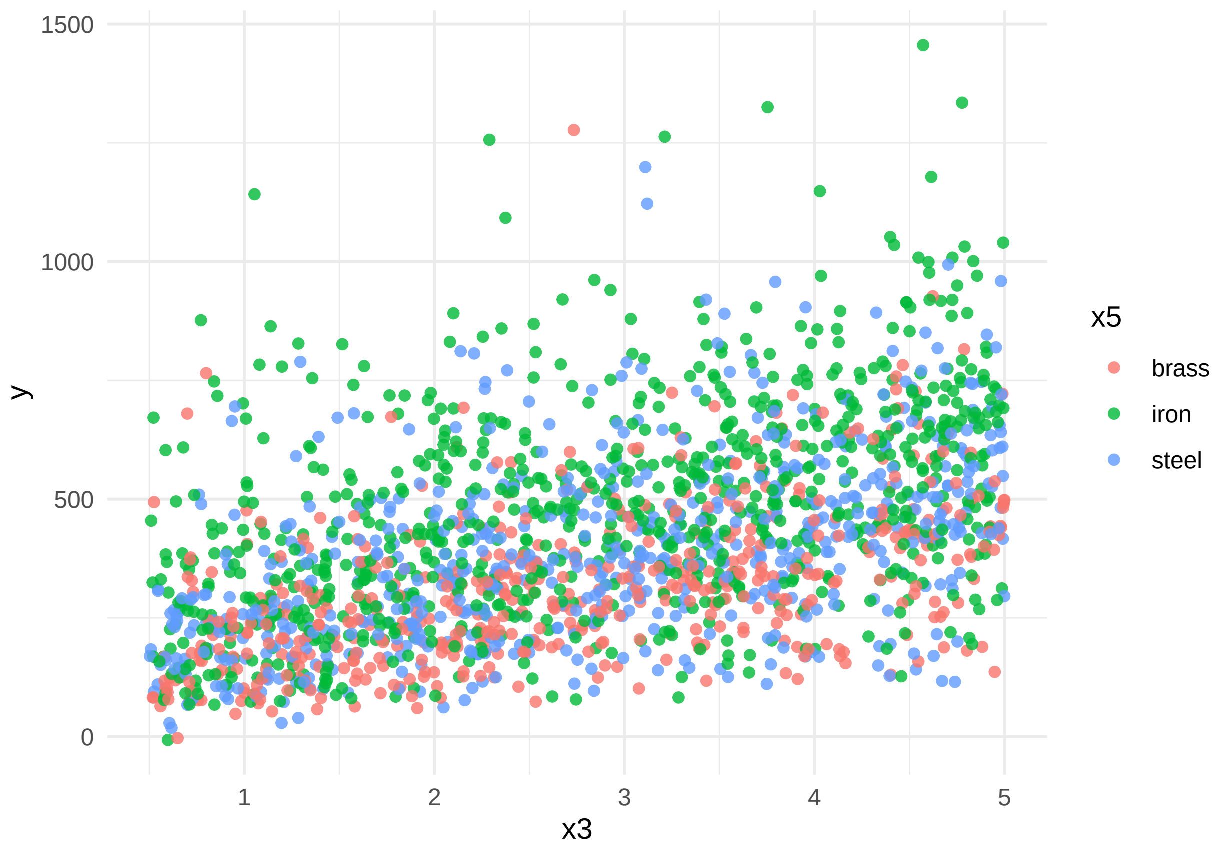

```r
ggplot(training_data, aes(x1, y, colour = x5)) + geom_point(alpha = .8)
```



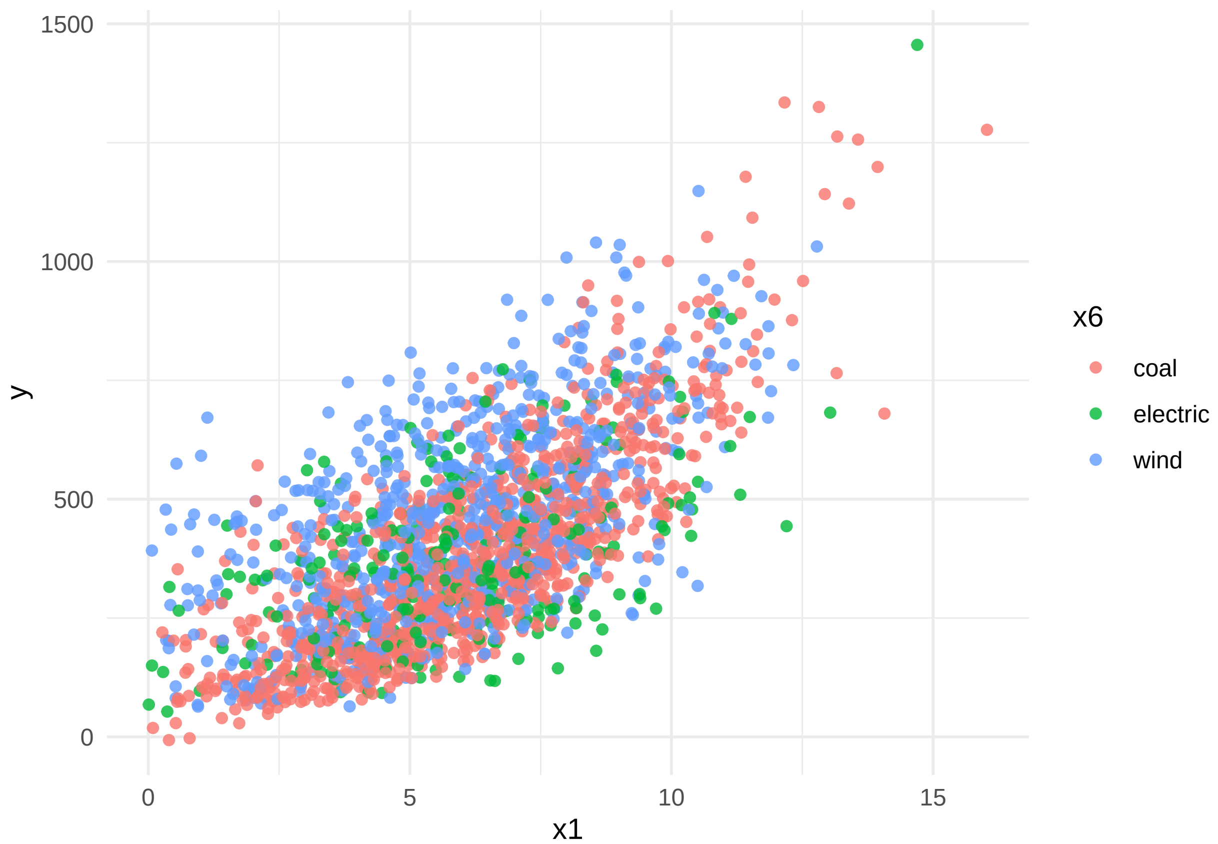

```
ggplot(training_data, aes(x2, y, colour = x5)) + geom_point(alpha = .8)
```



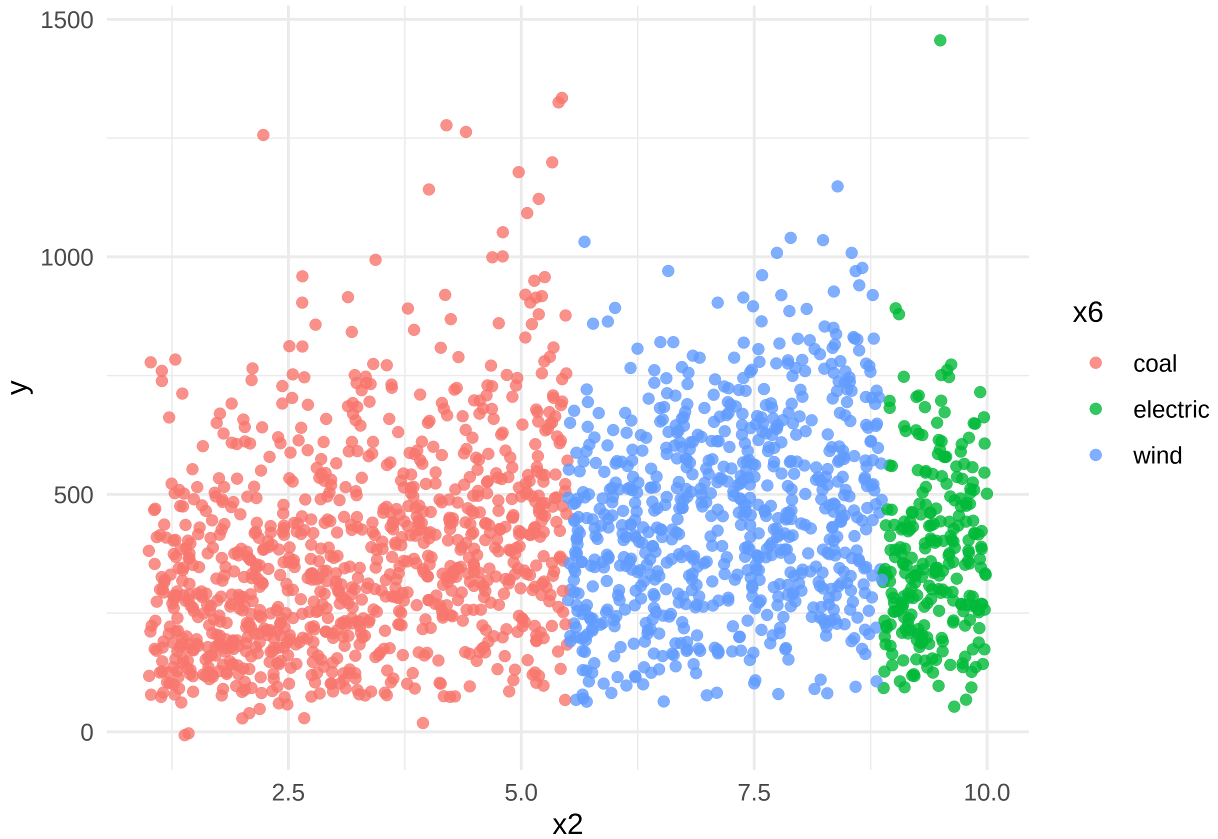

```
ggplot(training_data, aes(x3, y, colour = x5)) + geom_point(alpha = .8)
```



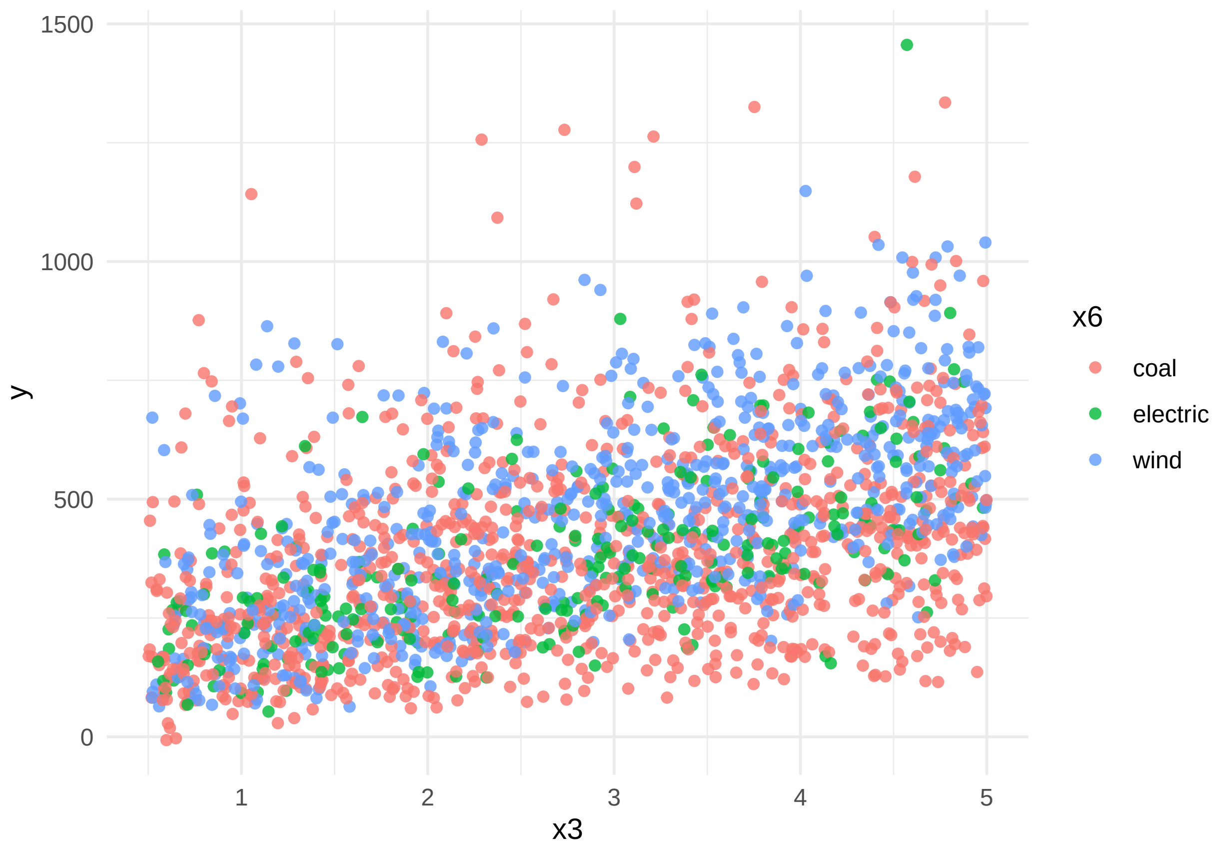

```r
ggplot(training_data, aes(x1, y, colour = x6)) + geom_point(alpha = .8)
```




```r
ggplot(training_data, aes(x2, y, colour = x6)) + geom_point(alpha = .8)
```




```r
ggplot(training_data, aes(x3, y, colour = x6)) + geom_point(alpha = .8)
```




```r
### additive models
library(gam)


gam_oracle <- gam(y ~ x1^2 + x2 + x2*x3 + as.factor(x4) + as.factor(x5) + as.factor(x6),
                  data=training_data)
gam_simple <- gam(y ~ ., data=training_data)


summary(gam_oracle) # all predictors significant
```

```
##
## Call: gam(formula = y ~ x1^2 + x2 + x2 * x3 + as.factor(x4) + as.factor(x5) +
##     as.factor(x6), data = training_data)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -250.989  -32.672   -6.401   23.365  433.246
##
## (Dispersion Parameter for gaussian family taken to be 3369.293)
##
##     Null Deviance: 87311399 on 1999 degrees of freedom
## Residual Deviance: 6701524 on 1989 degrees of freedom
## AIC: 21933.64
##
## Number of Local Scoring Iterations: 2
##
```

```
## Anova for Parametric Effects
##                 Df   Sum Sq  Mean Sq  F value    Pr(>F)
## x1               1 40569933 40569933 12041.08 < 2.2e-16 ***
## x2               1  4123033  4123033  1223.71 < 2.2e-16 ***
## x3               1 18792752 18792752  5577.65 < 2.2e-16 ***
## as.factor(x4)    2  2720018  1360009   403.65 < 2.2e-16 ***
## as.factor(x5)    2  7207848  3603924  1069.64 < 2.2e-16 ***
## as.factor(x6)    2  5418237  2709118   804.06 < 2.2e-16 ***
## x2:x3            1  1778054  1778054   527.72 < 2.2e-16 ***
## Residuals     1989  6701524     3369
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(gam_simple) # all predictors significant
```

```
##
## Call: gam(formula = y ~ ., data = training_data)
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -210.459  -39.236   -7.877   33.413  451.277
##
## (Dispersion Parameter for gaussian family taken to be 4261.094)
##
##     Null Deviance: 87311399 on 1999 degrees of freedom
## Residual Deviance: 8479578 on 1990 degrees of freedom
## AIC: 22402.29
##
## Number of Local Scoring Iterations: 2
##
## Anova for Parametric Effects
##             Df   Sum Sq  Mean Sq F value    Pr(>F)
## x1           1 40569933 40569933 9521.01 < 2.2e-16 ***
## x2           1  4123033  4123033  967.60 < 2.2e-16 ***
## x3           1 18792752 18792752 4410.31 < 2.2e-16 ***
## x4           2  2720018  1360009  319.17 < 2.2e-16 ***
## x5           2  7207848  3603924  845.77 < 2.2e-16 ***
## x6           2  5418237  2709118  635.78 < 2.2e-16 ***
## Residuals 1990  8479578     4261
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

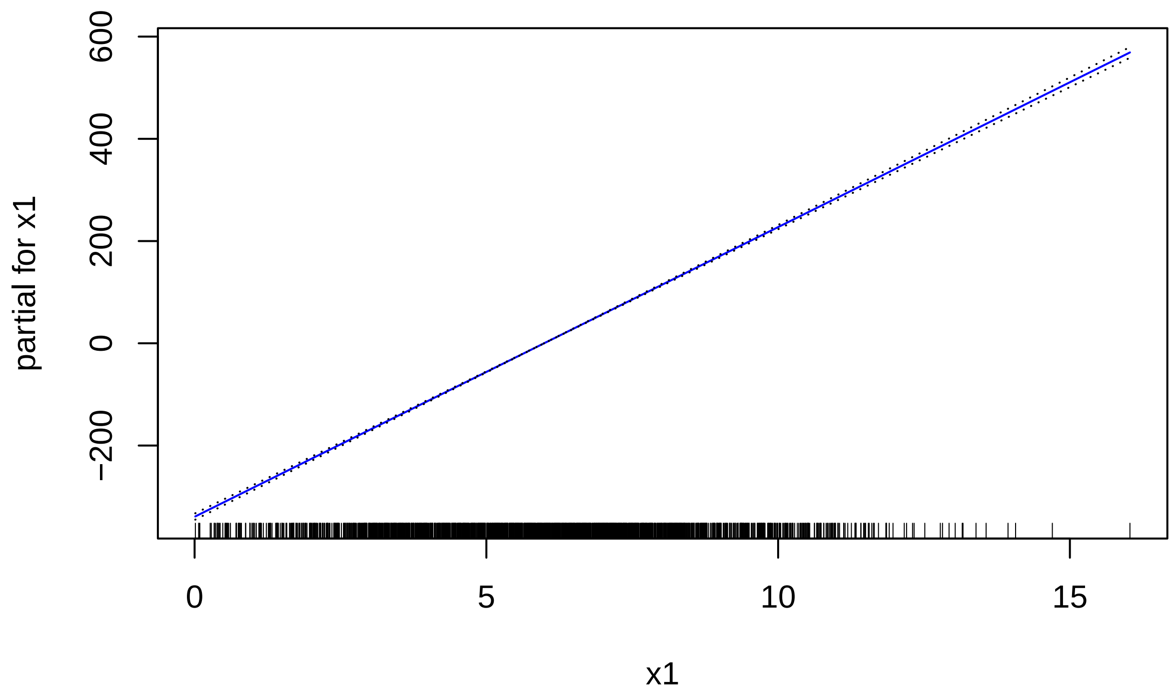
```r
preds_oracle <- predict(gam_oracle , newdata=training_data)
mean((preds_oracle - training_data$y)^2) # smaller and so better than 'simple' model
```

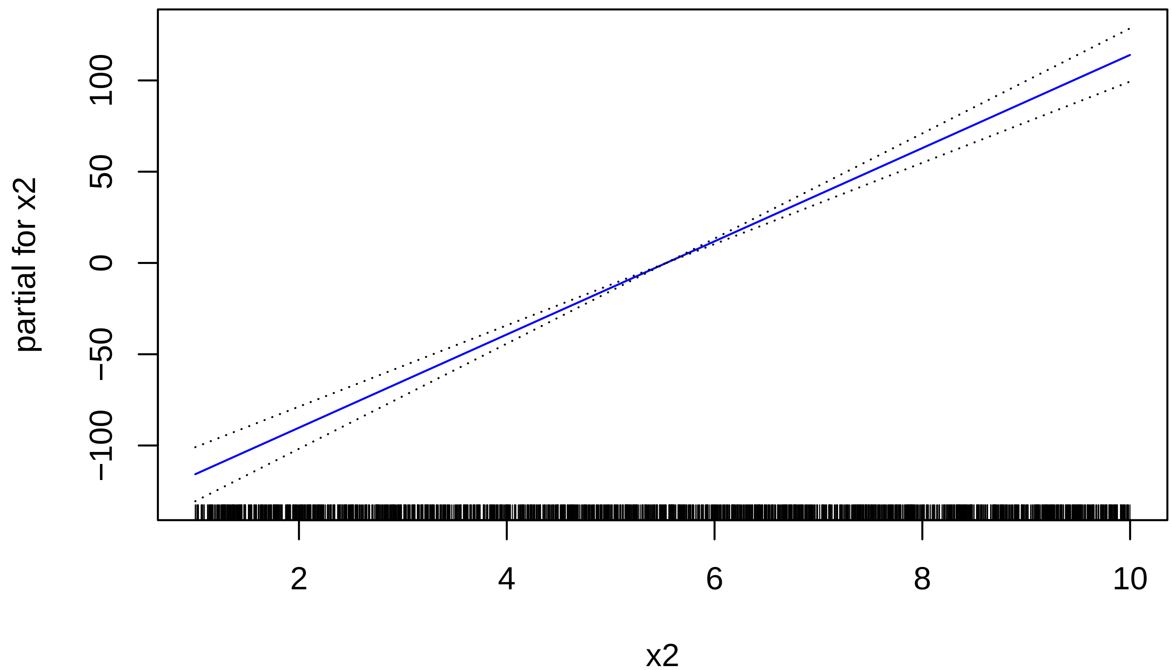
```
## [1] 3350.762
```

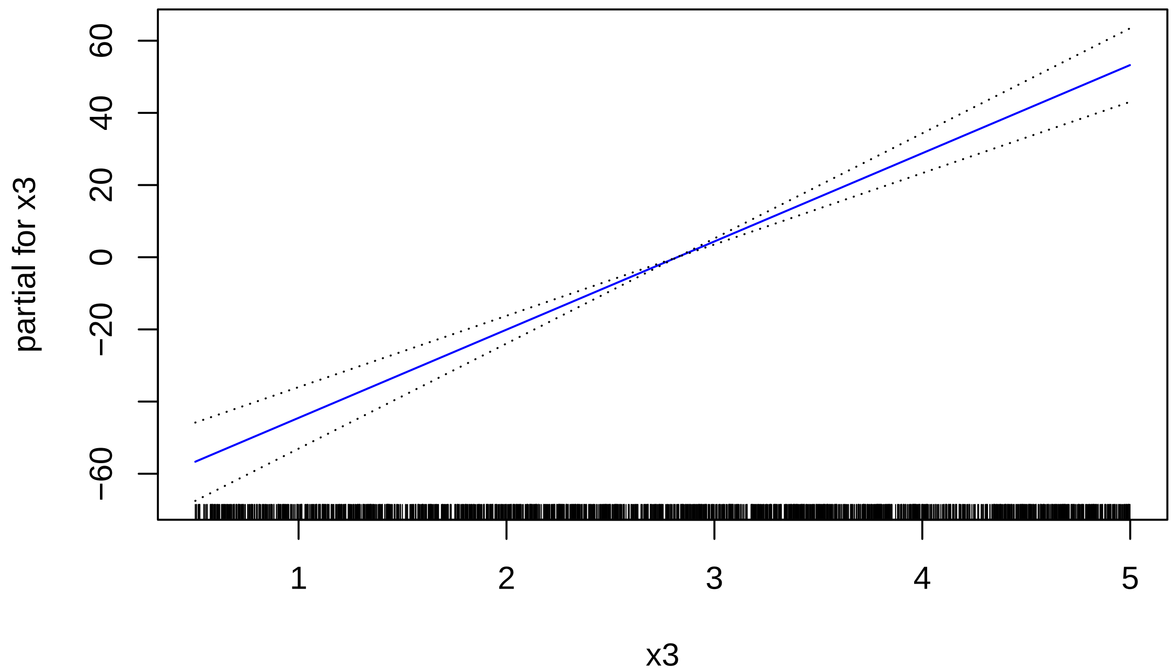
```r
preds_simple <- predict(gam_simple , newdata=training_data)
mean((preds_simple - training_data$y)^2) # larger and so worse than 'oracle' model
```

```
## [1] 4239.789
```

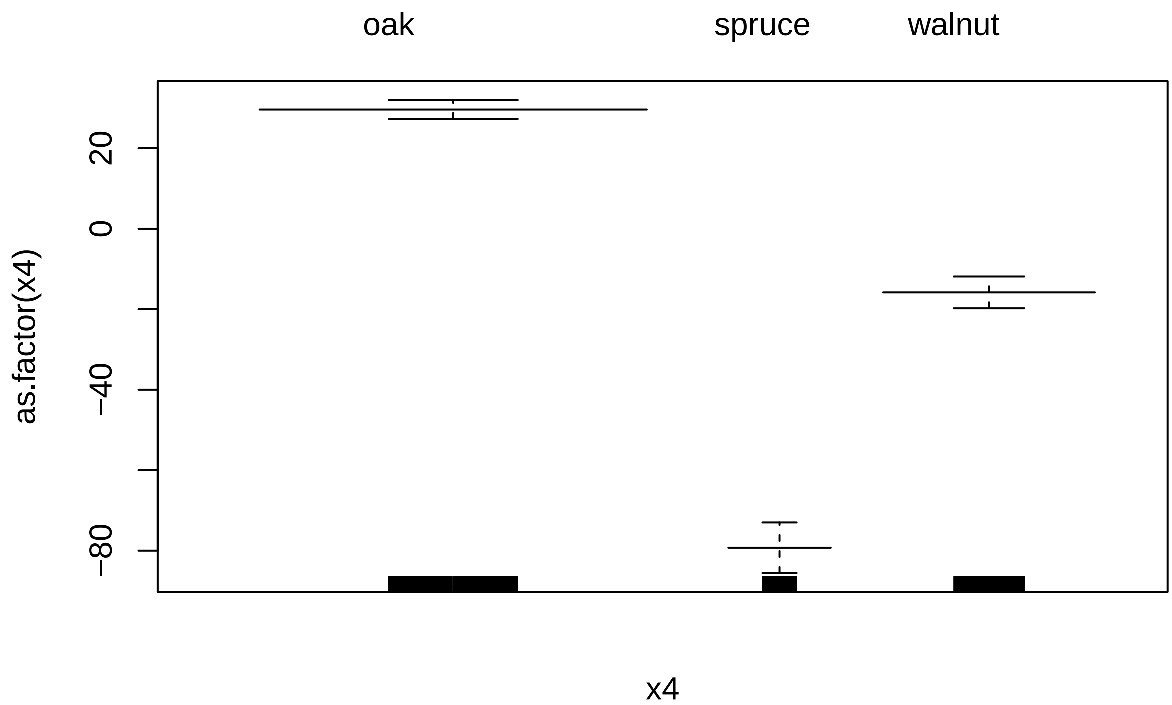
```r
plot.Gam(gam_oracle, se=T , col="blue") # smaller difference from true value than 'simple' model
```

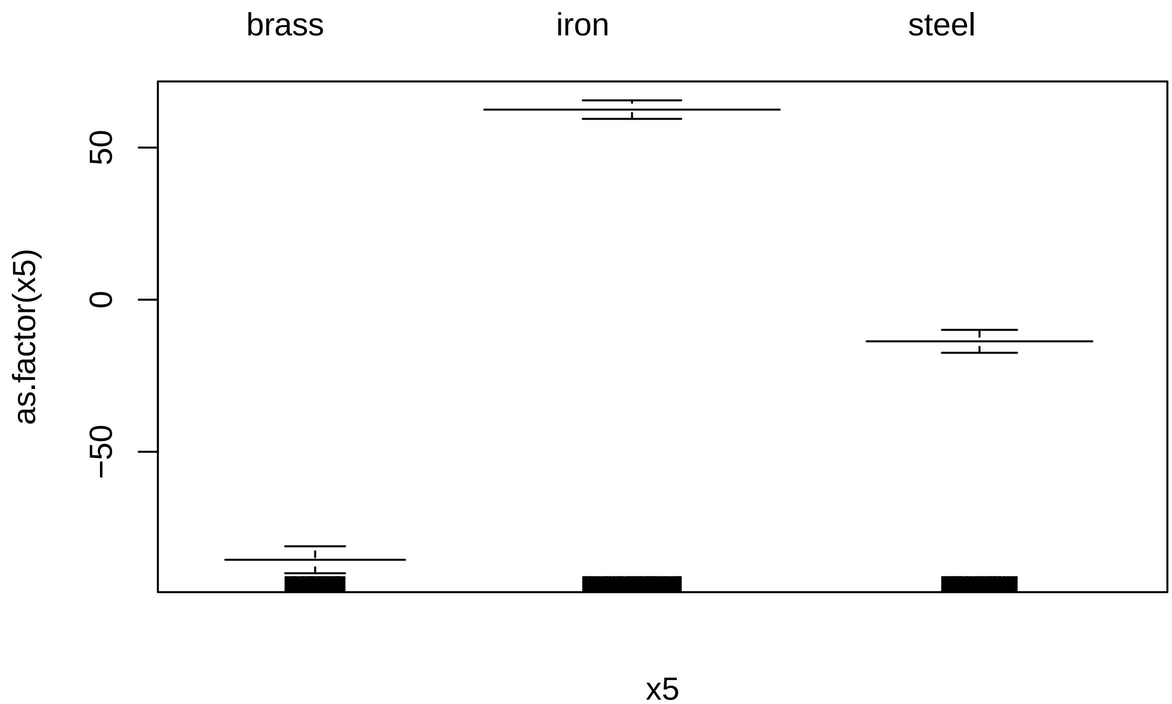


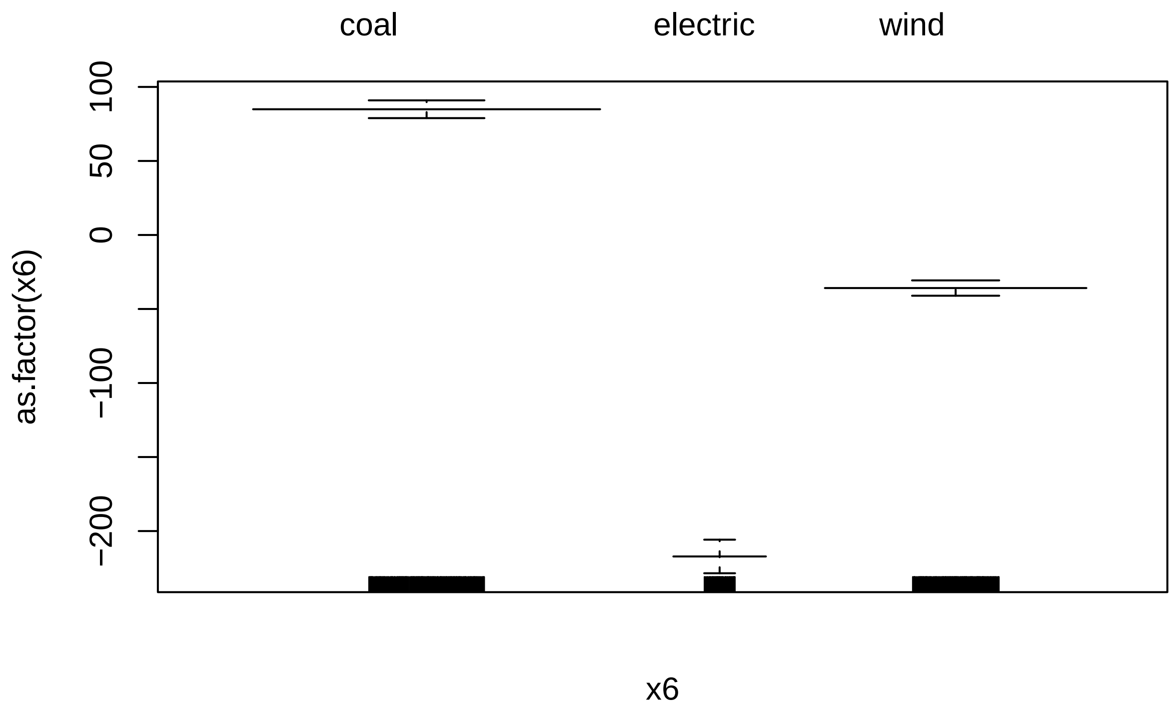

100
50
0
−50
−100
partial for x2
2
4
6
8
10
x2



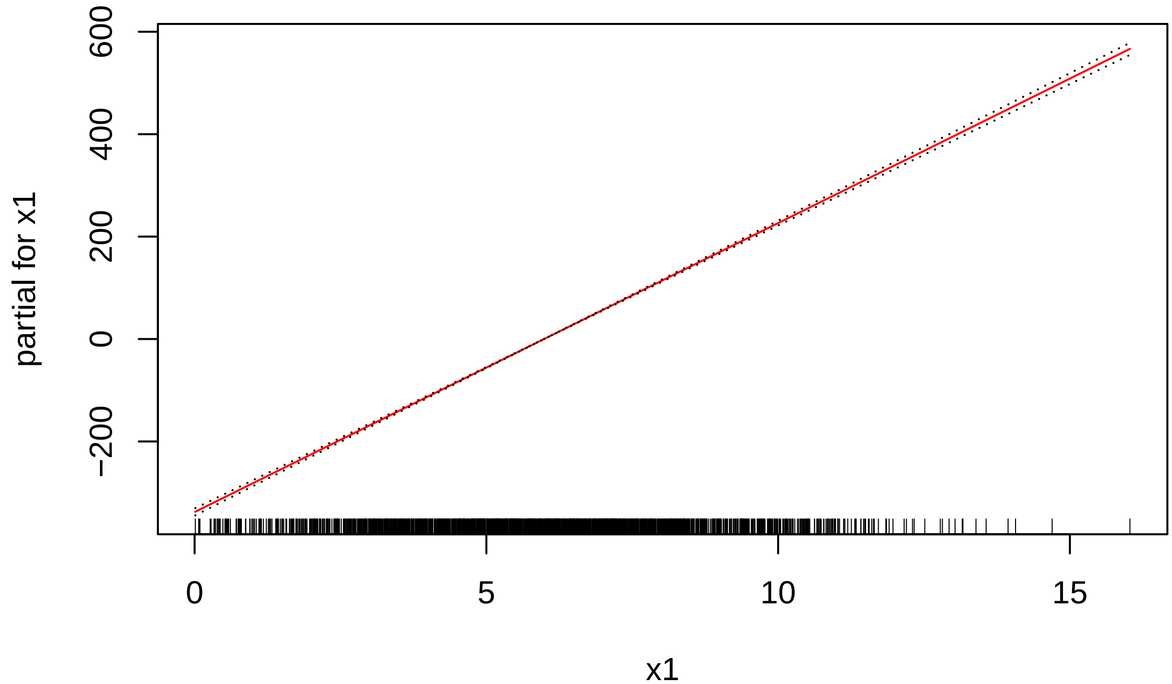
partial for x3
60
40
20
0
−20
−60
1
2
3
4
5
x3

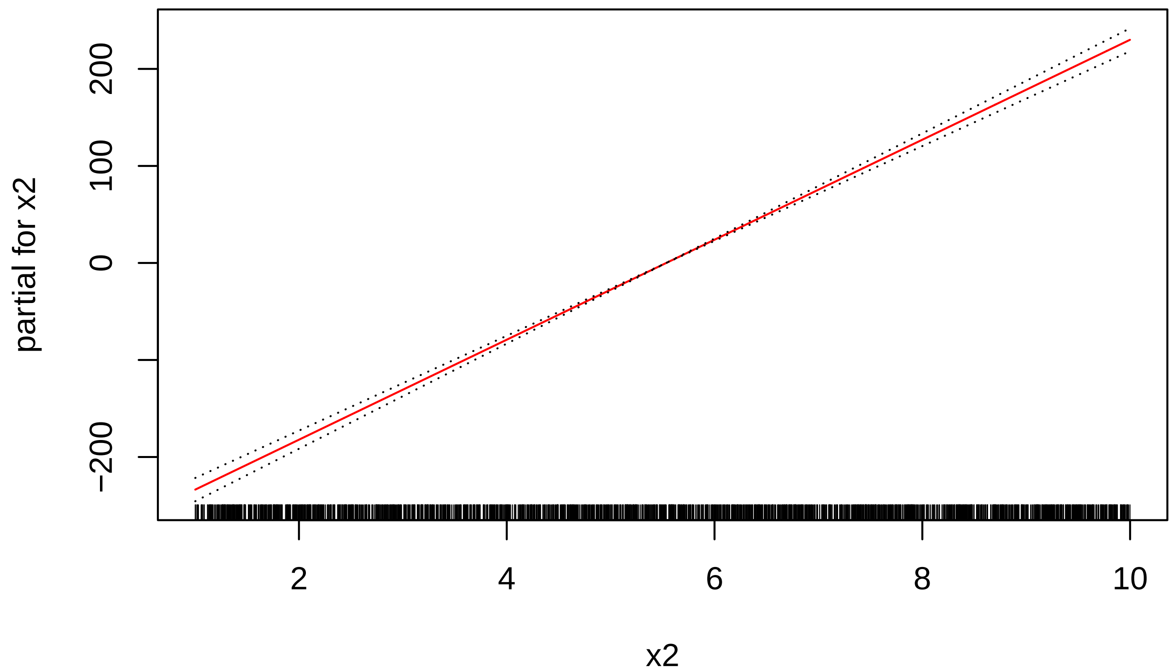

oak
spruce
walnut
as.factor(x4)
20
0
−40
−80
x4

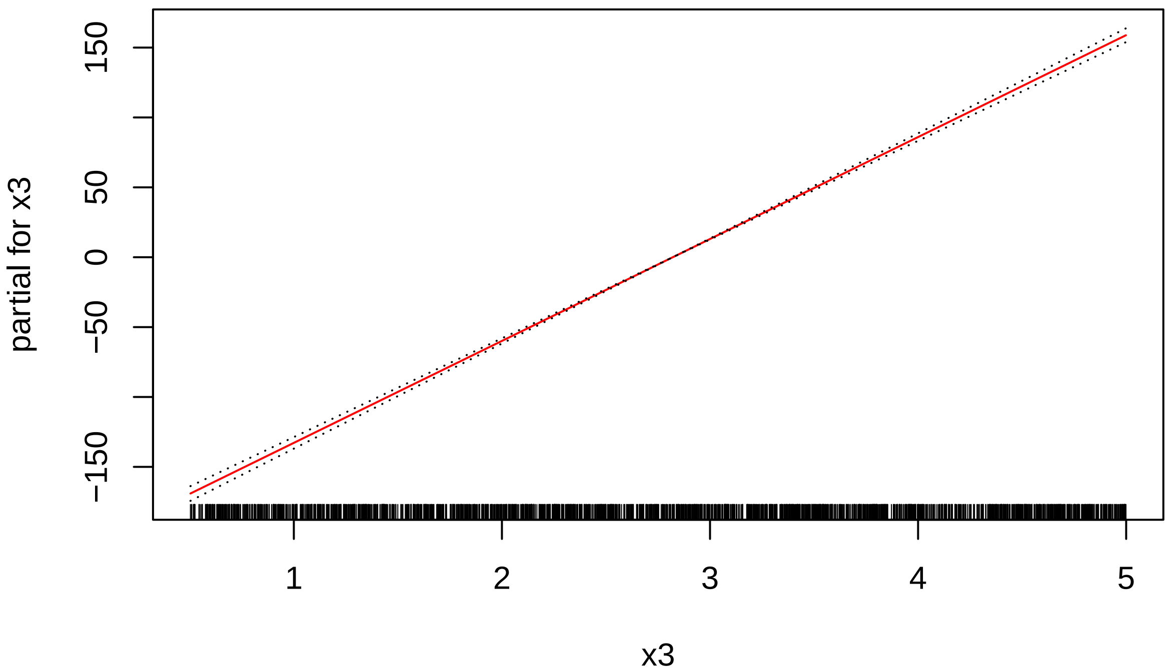

brass
iron
steel
50
0
−50
as.factor(x5)
x5



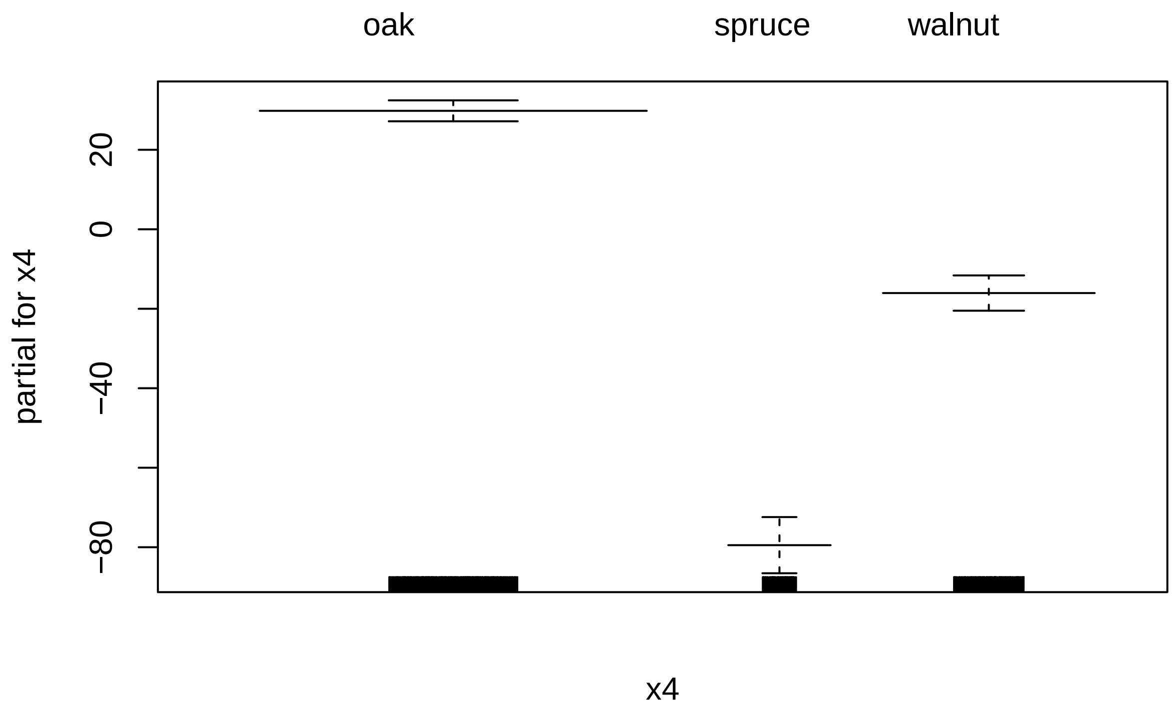

```
plot.Gam(gam_simple, se=T, col="red")
```

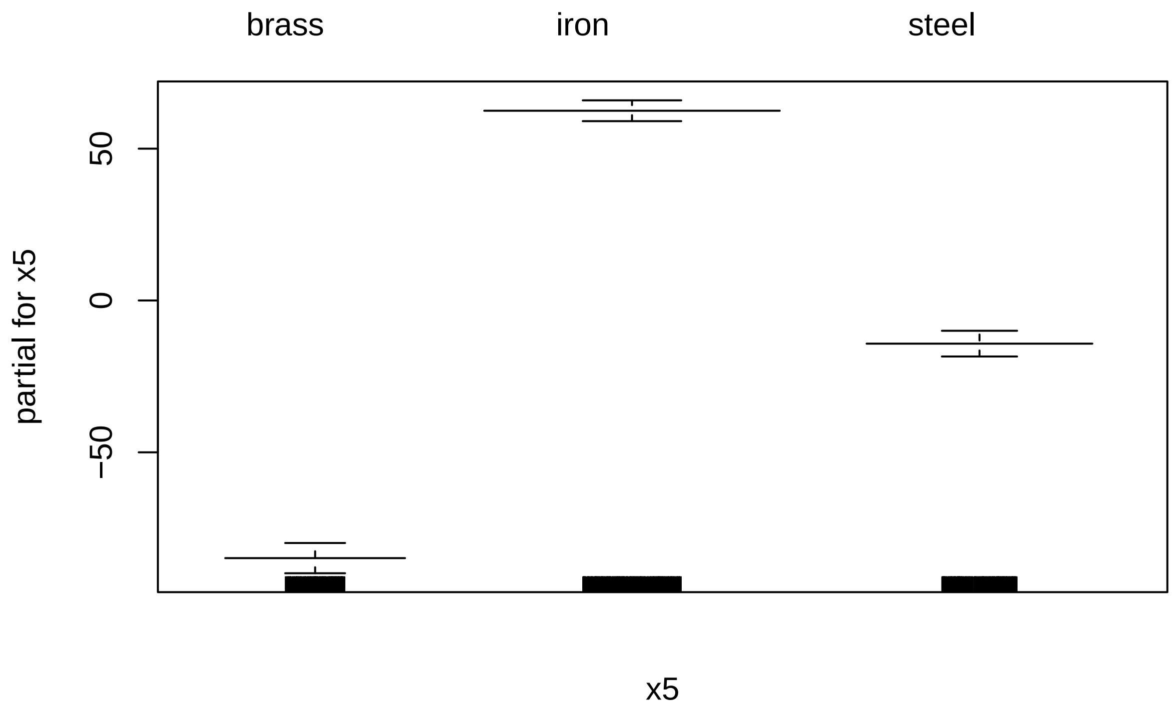

600
400
200
0
−200
partial for x1
0
5
10
15
x1

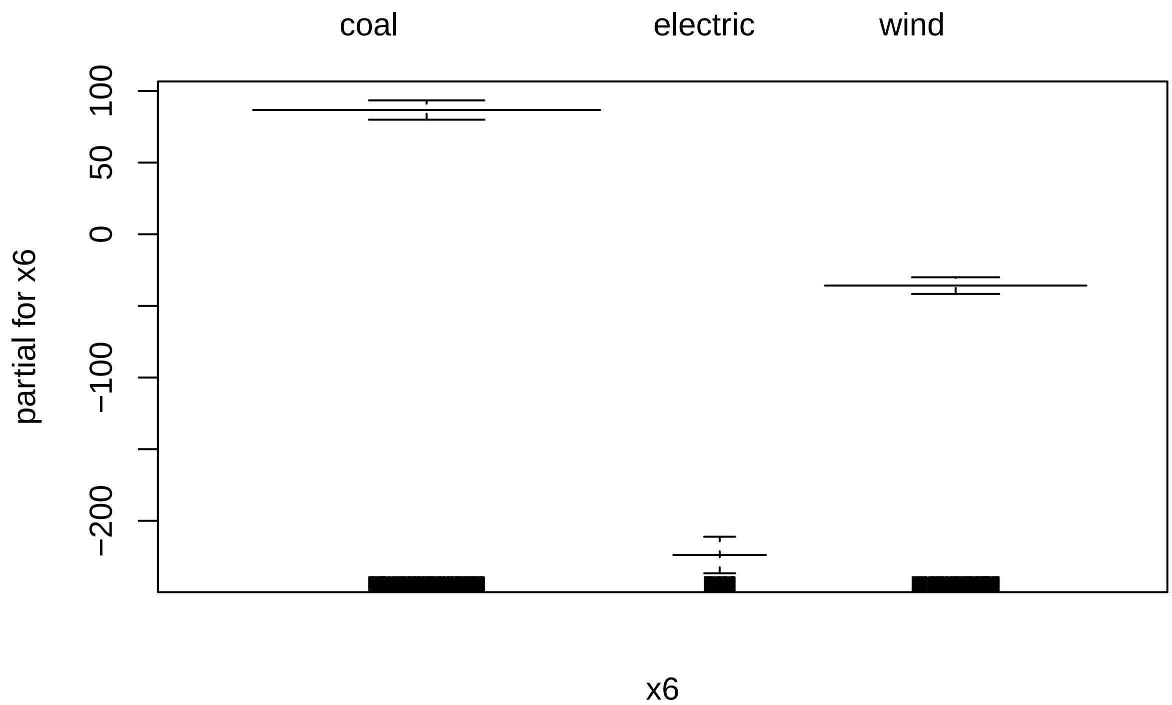

partial for x2
200
100
0
−200
2
4
6
8
10
x2



150
50
0
−50
−150
partial for x3
1
2
3
4
5
x3


oak
spruce
walnut
20
0
−40
−80
partial for x4
x4

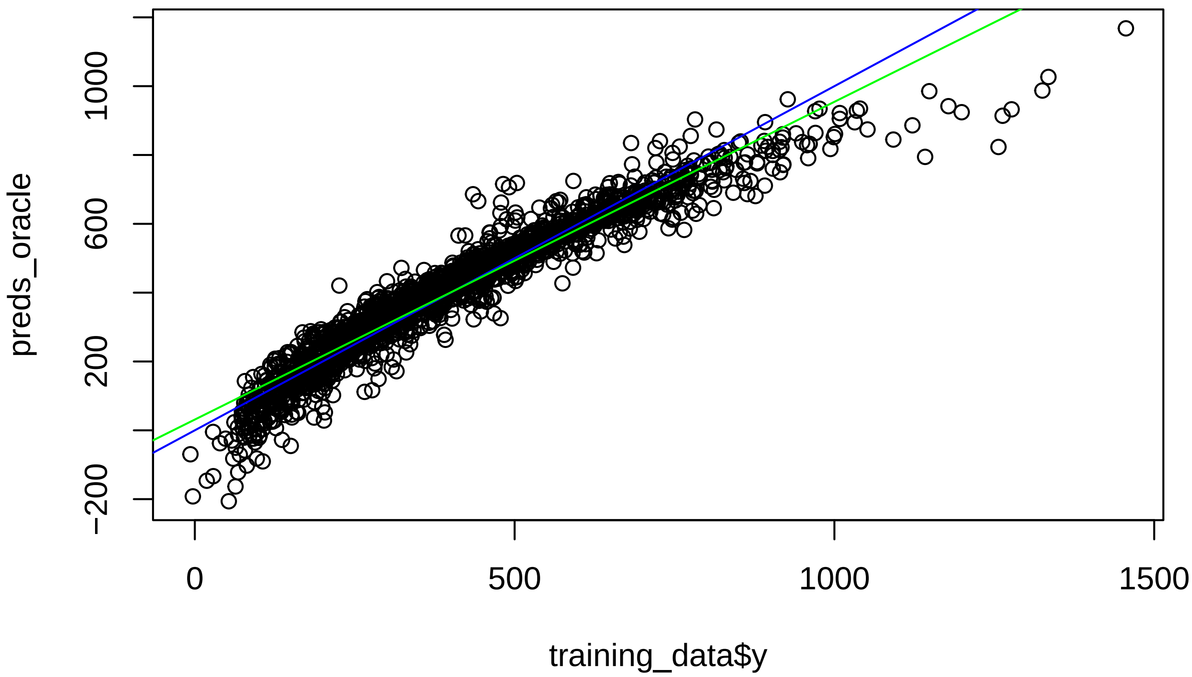

brass
iron
steel
50
0
−50
partial for x5
x5



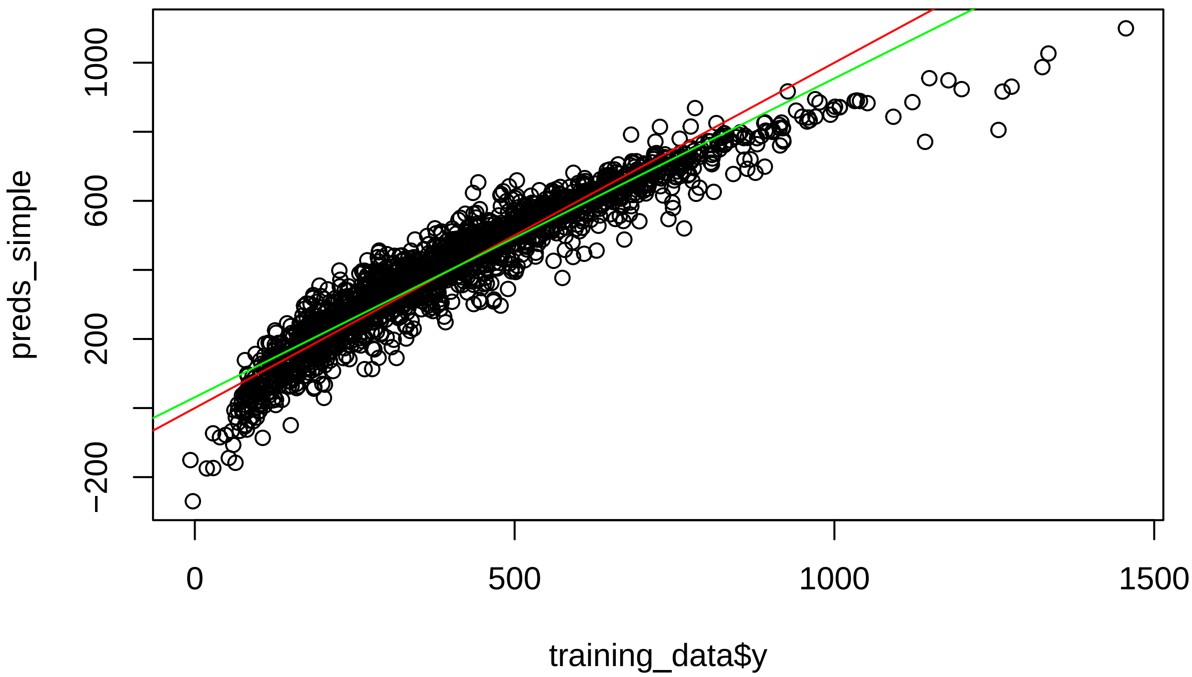

```r
# both show very little variance

anova(gam_oracle, gam_simple, test="F") # 'oracle' model is significantly better
```

```
## Analysis of Deviance Table
##
## Model 1: y ~ x1^2 + x2 + x2 * x3 + as.factor(x4) + as.factor(x5) + as.factor(x6)
## Model 2: y ~ x1 + x2 + x3 + x4 + x5 + x6
##   Resid. Df Resid. Dev Df Deviance      F    Pr(>F)
## 1      1989    6701524
## 2      1990    8479578 -1 -1778054 527.72 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

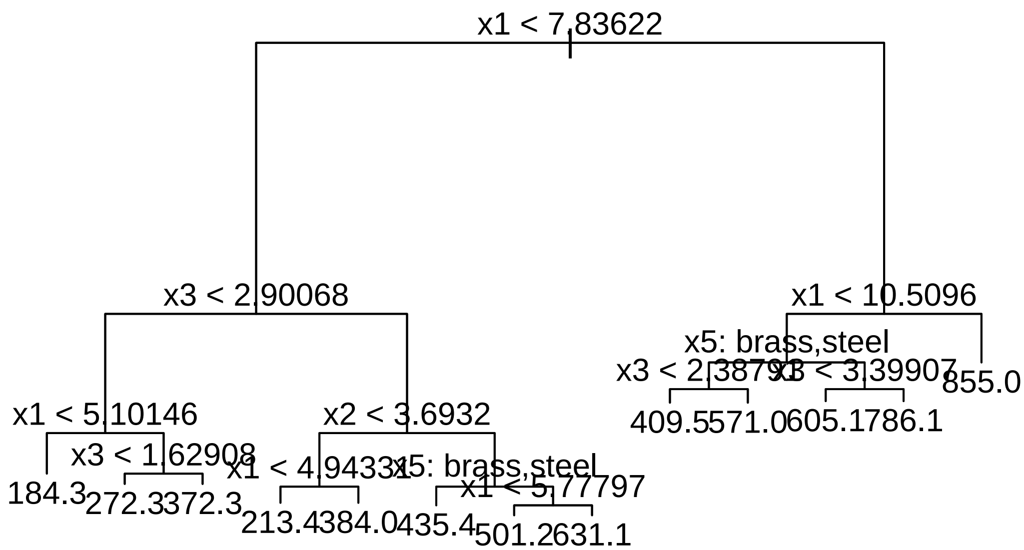
```r
plot(training_data$y, preds_oracle)
abline(0, 1, col="blue")
abline(lm(preds_oracle ~ training_data$y), col="green") # closer to true CEF
```



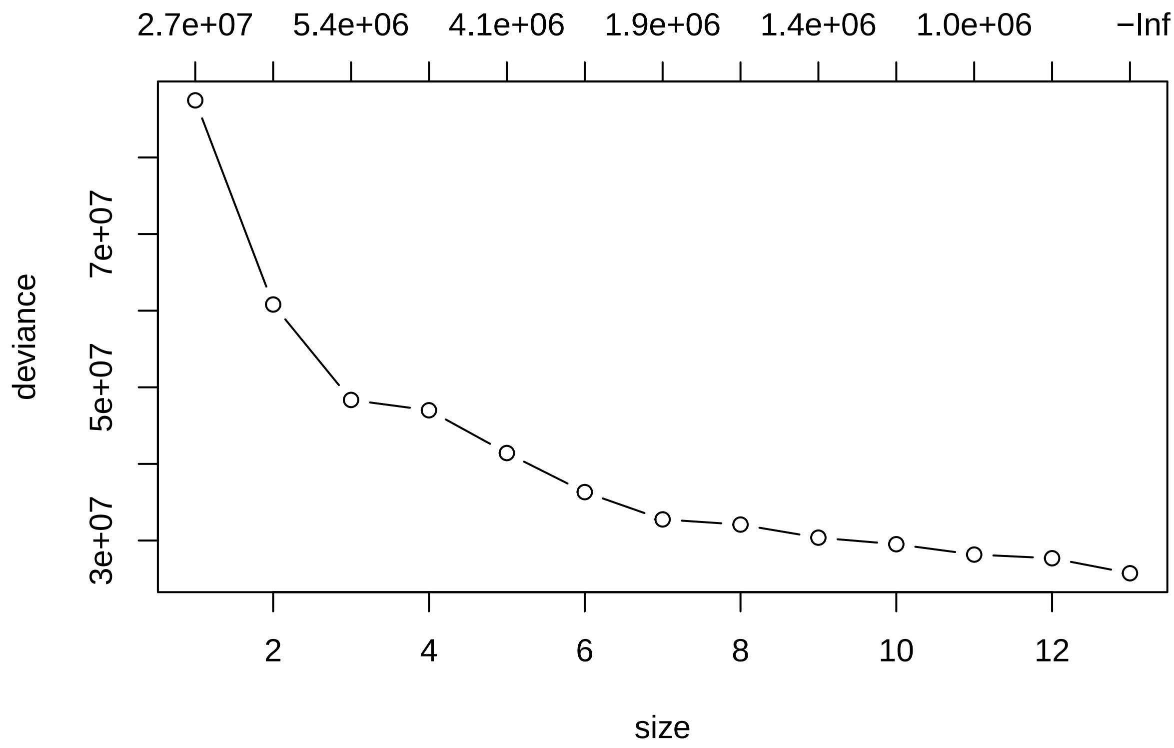

```
plot(training_data$y, preds_simple)
abline(0, 1, col="red")
abline(lm(preds_oracle ~ training_data$y), col="green")
```



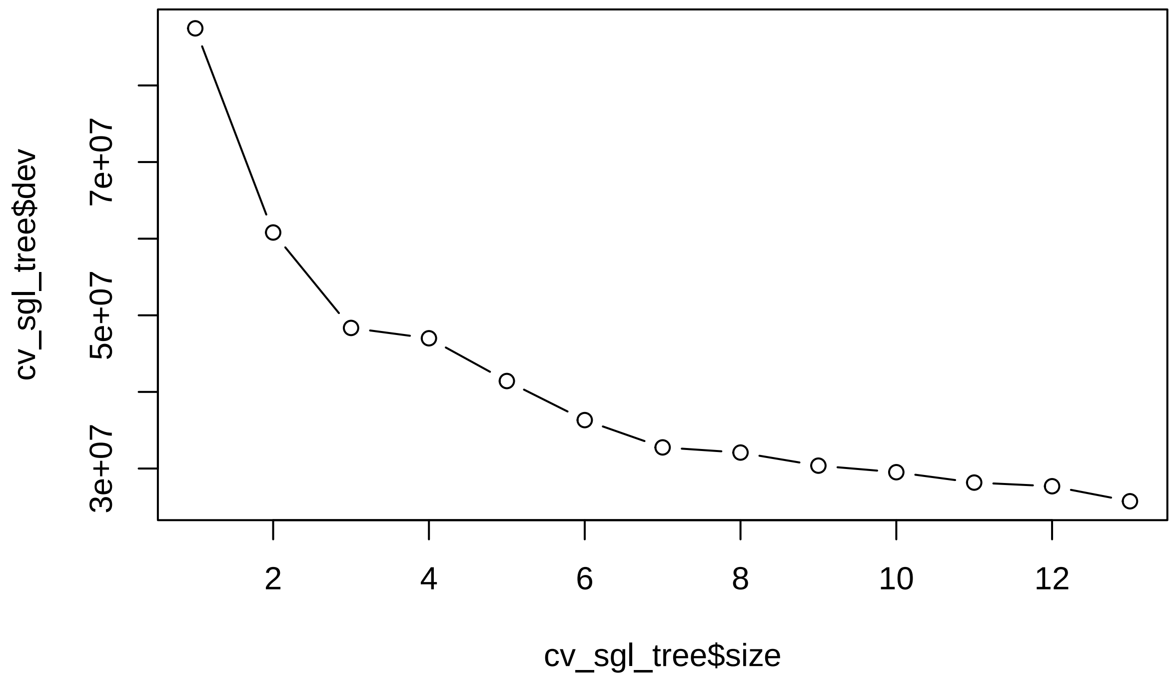

```
library(tree)
library(randomForest)
library(gbm)


### single tree
single_tree <- tree(y ~ ., training_data)

plot(single_tree)
text(single_tree , pretty = 0)
```



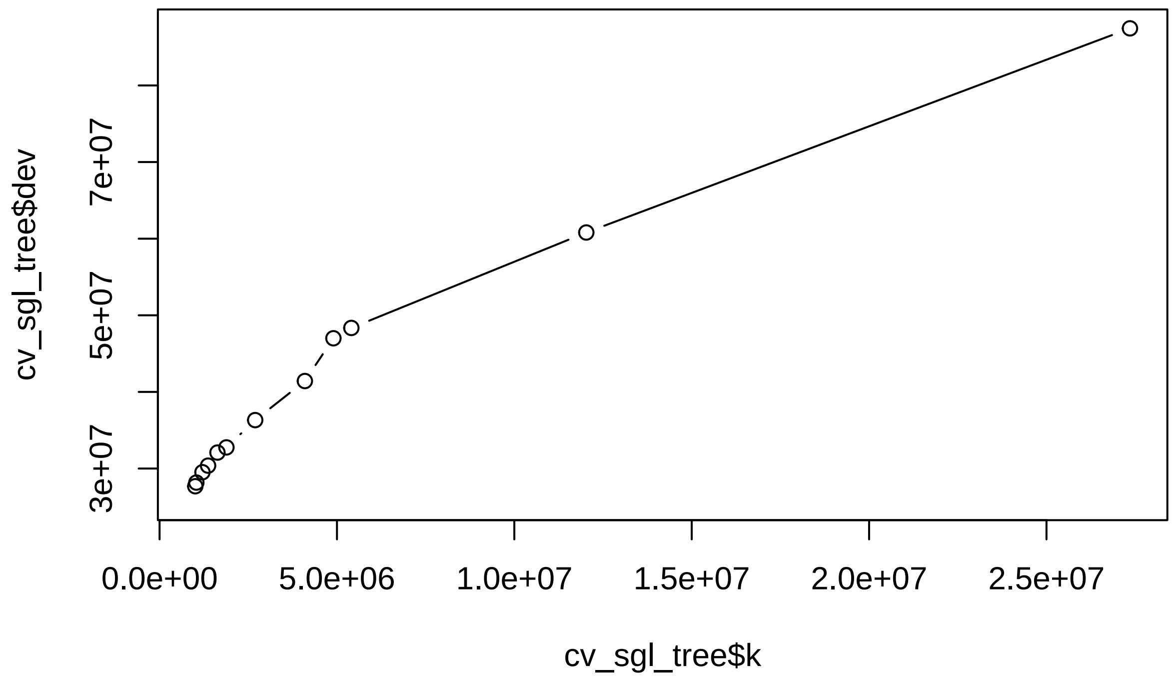

```r
cv_sgl_tree <- cv.tree(single_tree) # cross-validating to choose tuning parameters
plot(cv_sgl_tree, type='b')
```




```r
plot(cv_sgl_tree$size , cv_sgl_tree$dev , type = "b")
```




```r
plot(cv_sgl_tree$k, cv_sgl_tree$dev , type = "b")
```



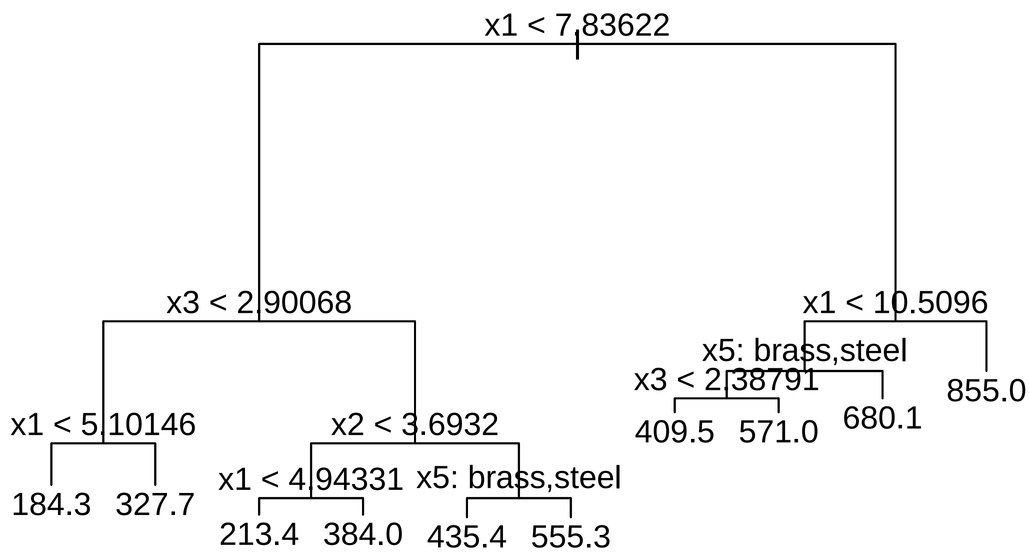

```r
names(cv_sgl_tree)
```

```
## [1] "size"   "dev"    "k"      "method"
```

```r
cv_sgl_tree
```

```
## $size
##  [1] 13 12 11 10  9  8  7  6  5  4  3  2  1
##
## $dev
##  [1] 25731544 27689534 28168083 29518963 30375955 32084541 32757859 36321628
##  [9] 41421621 47002175 48348129 60807044 87450502
##
## $k
##  [1]     -Inf  1005793  1037149  1208971  1367187  1631732  1884687  2694818
##  [9]  4095536  4900065  5405733 12028297 27353711
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

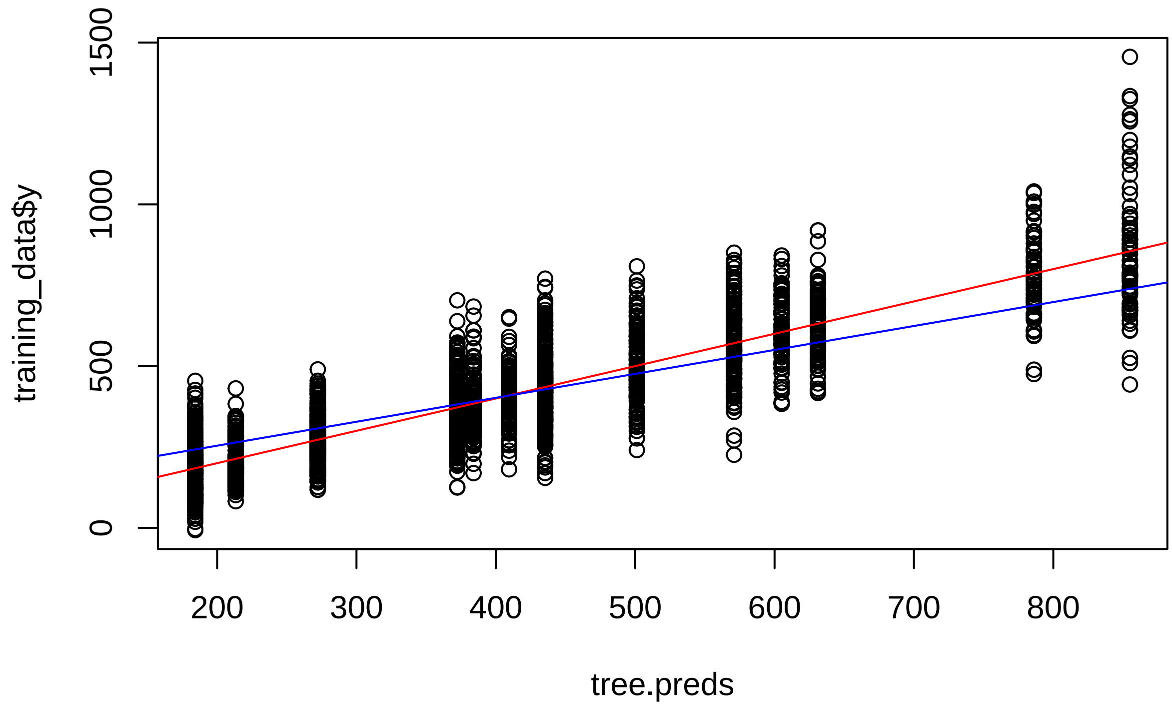
```r
prune_tree <- prune.tree(single_tree , best = 10)
plot(prune_tree)
text(prune_tree , pretty = 0)
```




```r
tree.preds <- predict(single_tree, newdata = training_data)
mean((tree.preds - training_data$y)^2)
```

```
## [1] 11348.86
```

```r
plot(tree.preds, training_data$y) # not close enough fit to training data or CEF
abline(0, 1, col="red")
abline(lm(tree.preds ~ training_data$y), col="blue")
```




```r
# very sensitive to change in data but very interpretable


### random forest
rand_forest <- randomForest(y ~ ., data=training_data, mtry = 2, importance = TRUE)

rand_forest # 6/3 = 2 vars tried at each split
```

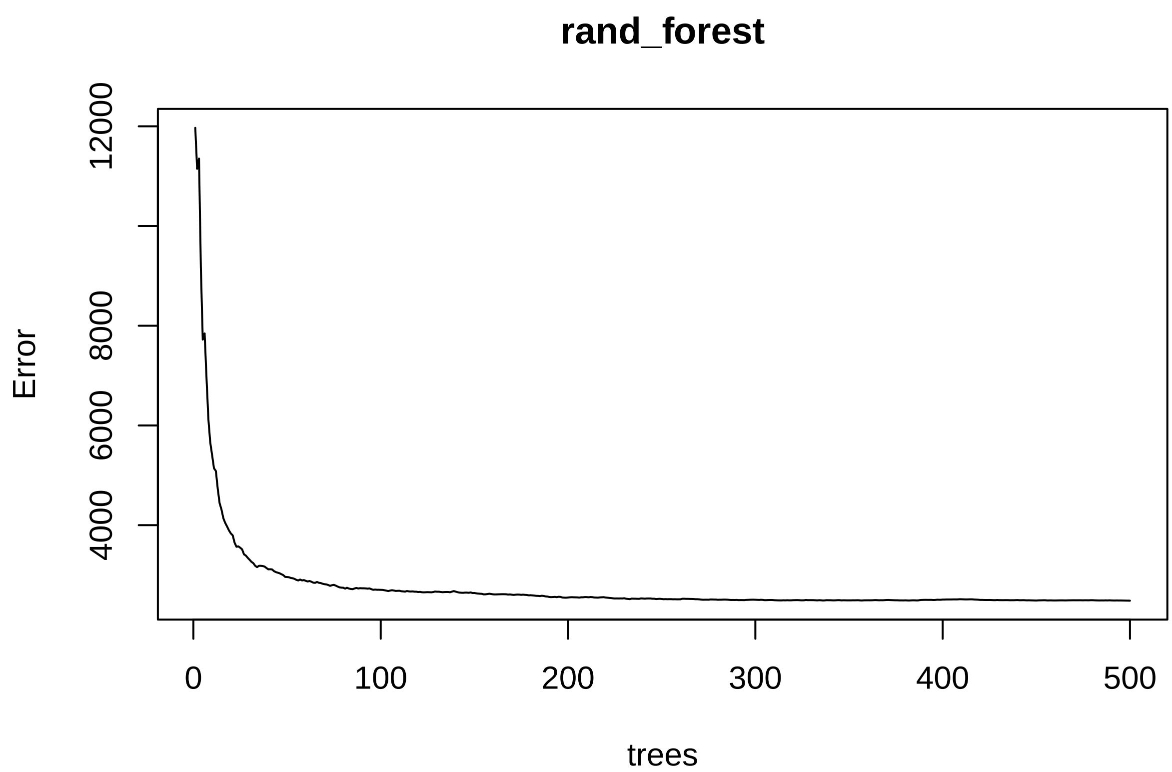
```
##
## Call:
##  randomForest(formula = y ~ ., data = training_data, mtry = 2,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 2484.914
##                     % Var explained: 94.31
```

```r
importance(rand_forest)
```

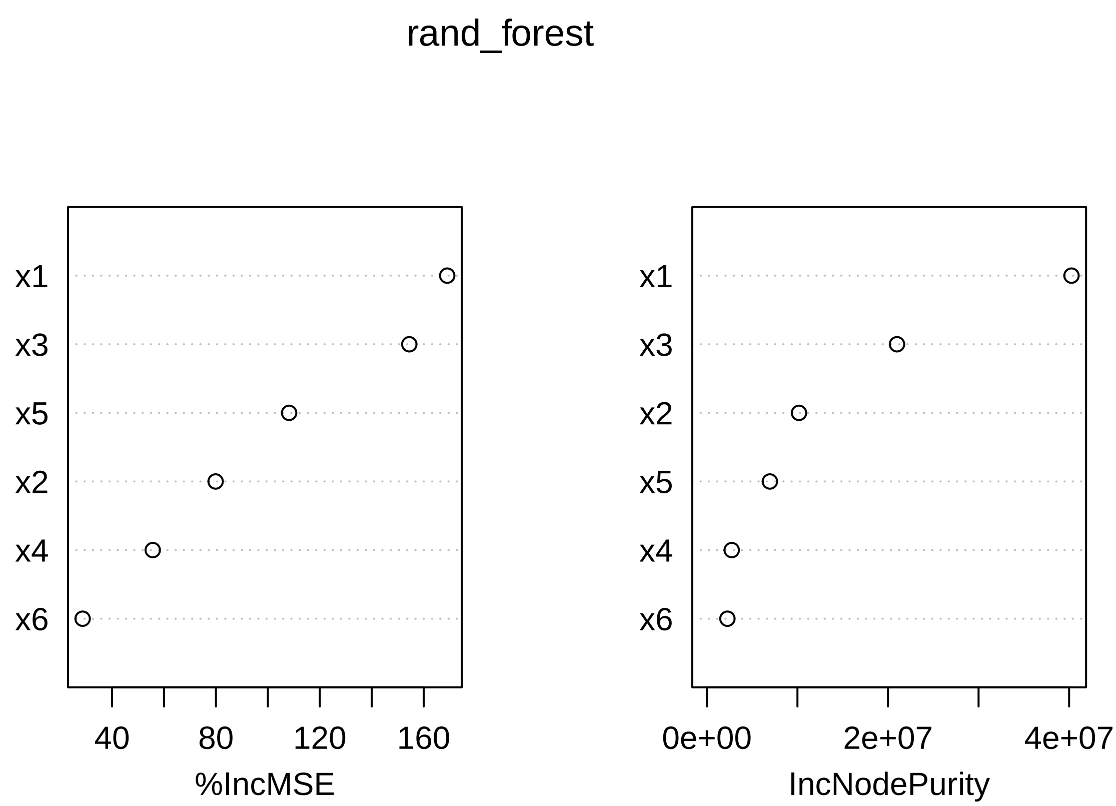
```
##      %IncMSE IncNodePurity
## x1 169.06988      40262961
## x2  79.87524      10169485
## x3 154.46211      20994543
## x4  55.66504       2740189
```

```
## x5 108.18398        6958076
## x6  28.65725        2266770
```

```r
plot(rand_forest)
```



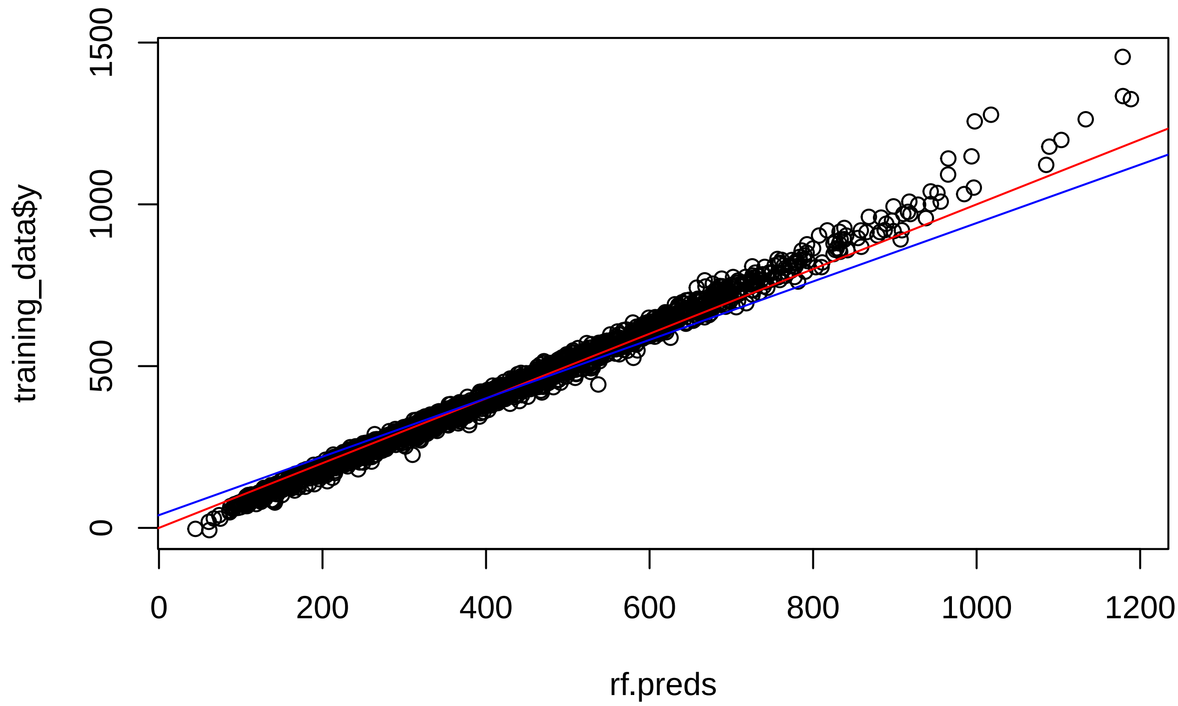

```r
varImpPlot(rand_forest)
```




```
rf.preds <- predict(rand_forest, newdata = training_data)
mean((rf.preds - training_data$y)^2)
```

```
## [1] 728.5933
```

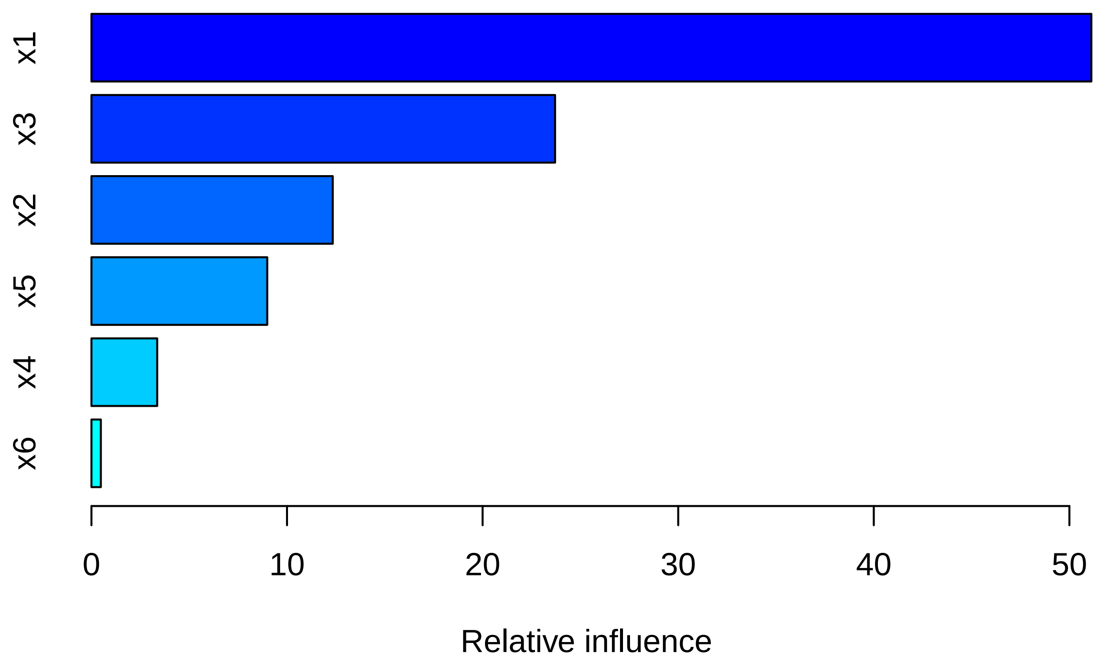
```
plot(rf.preds, training_data$y) # slightly off training data and CEF but still very close
abline(0, 1, col="red")
abline(lm(rf.preds ~ training_data$y), col="blue")
```




```r
# less interpretable and slower than single tree


### boosted tree
boost_tree <- gbm(y ~ ., data=training_data, distribution="gaussian",
                  n.trees=5000, interaction.depth=4, shrinkage=0.2)

boost_tree
```

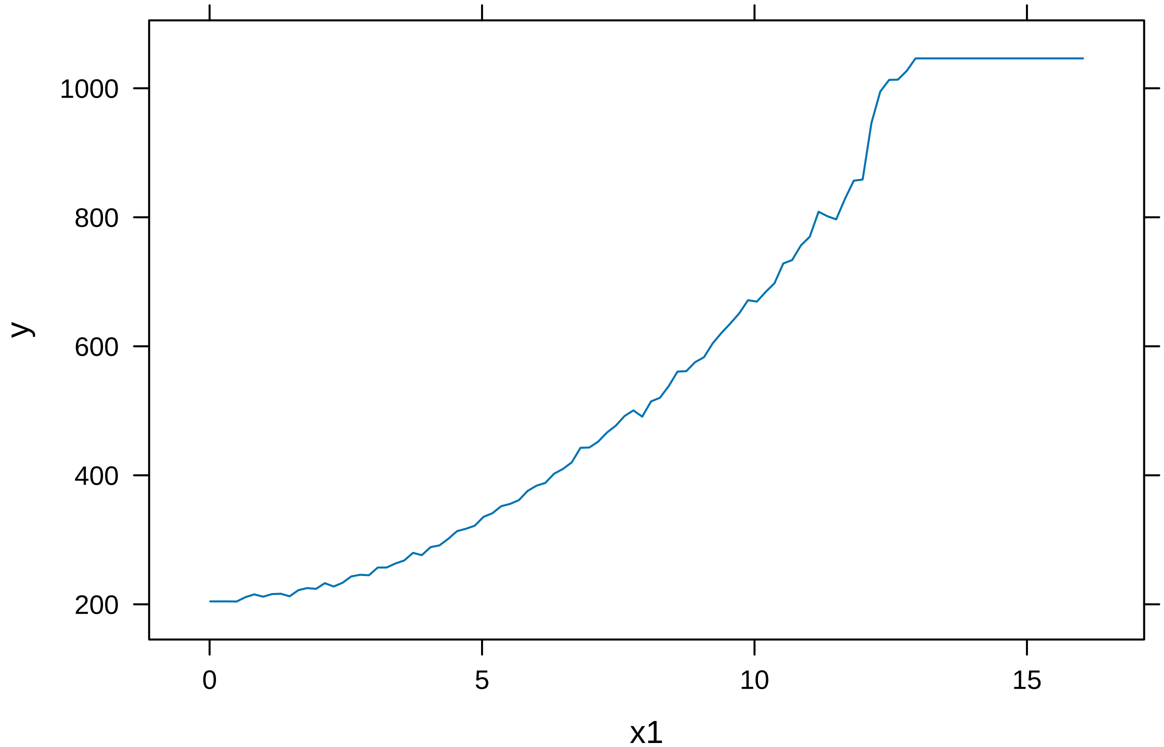
```
## gbm(formula = y ~ ., distribution = "gaussian", data = training_data,
##     n.trees = 5000, interaction.depth = 4, shrinkage = 0.2)
## A gradient boosted model with gaussian loss function.
## 5000 iterations were performed.
## There were 6 predictors of which 6 had non-zero influence.
```

```r
summary(boost_tree)
```

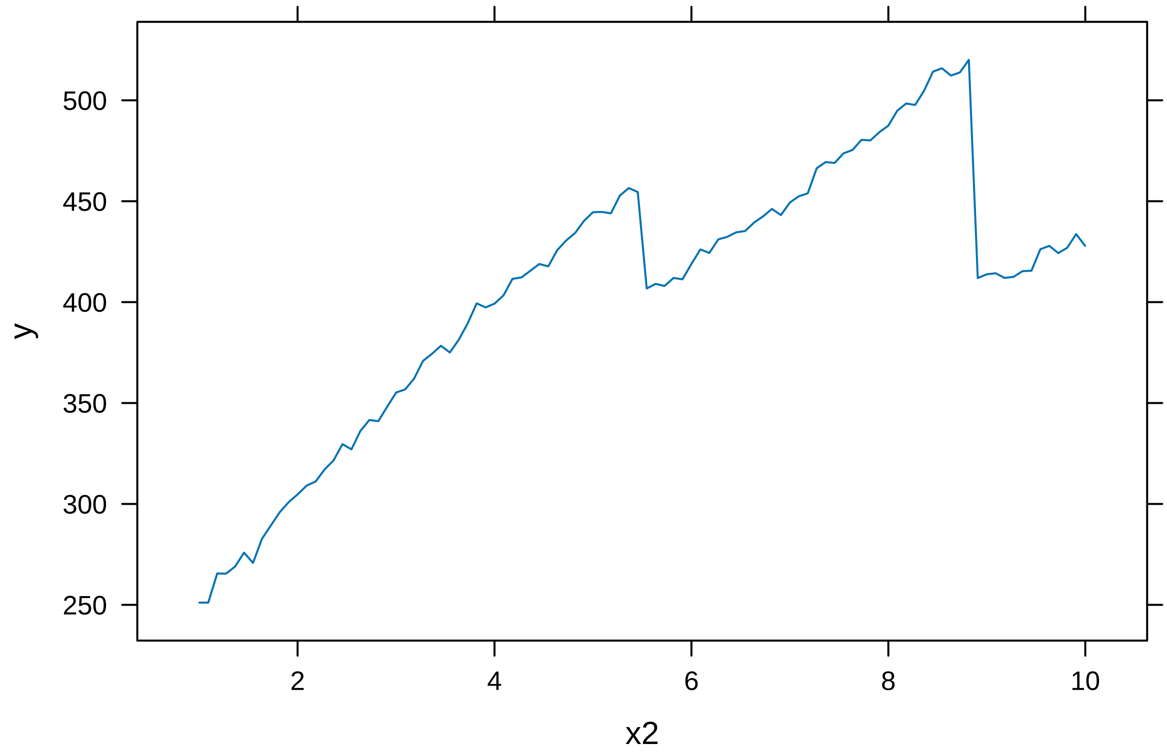



```
##     var    rel.inf
## x1  x1 51.1240127
## x3  x3 23.7058689
## x2  x2 12.3393906
## x5  x5  8.9883136
## x4  x4  3.3617303
## x6  x6  0.4806839
```

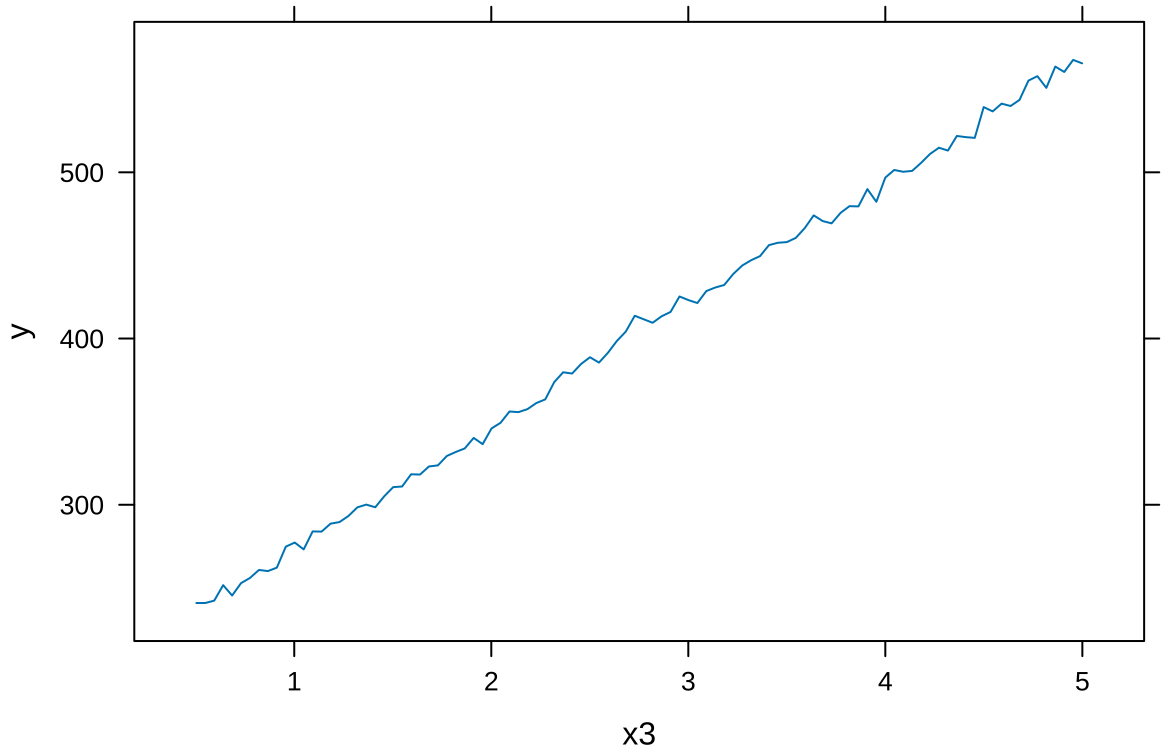
```
# partial dependence plots
plot(boost_tree, i='x1') # exponential relationship with y
```



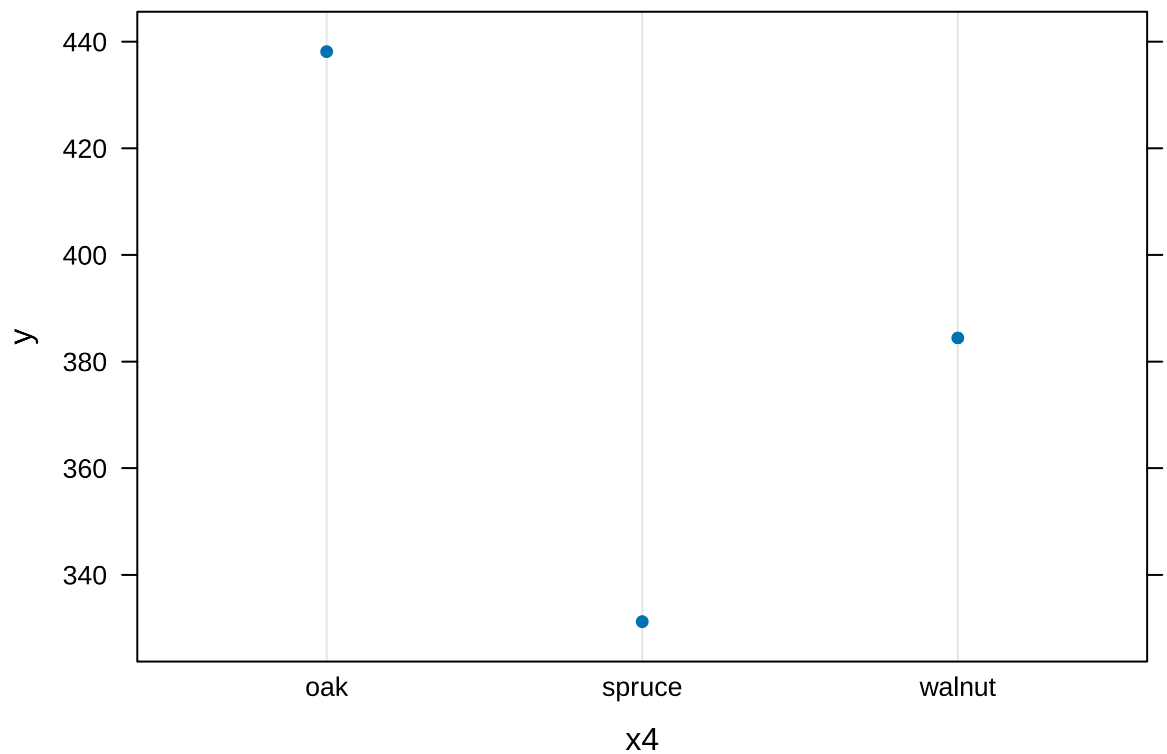

```
plot(boost_tree, i='x2') # linear relationship but with 2 sudden drops
```



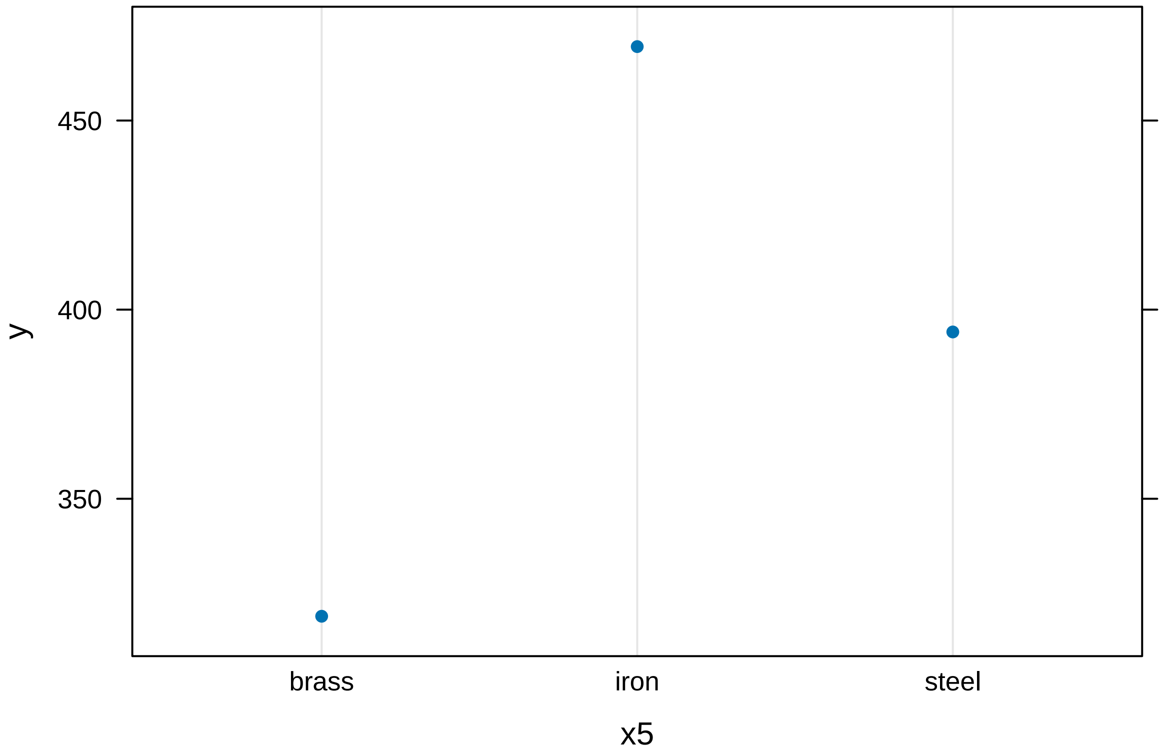

```
plot(boost_tree, i='x3') # very linearly dependent
```



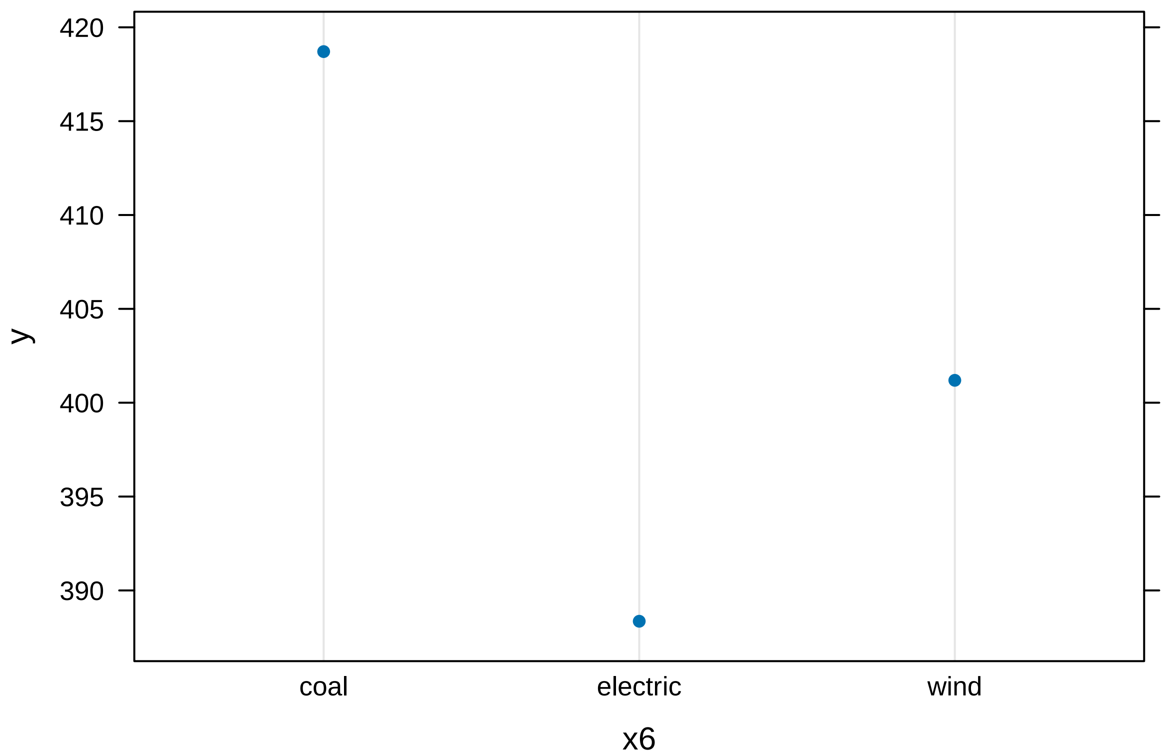

```r
plot(boost_tree, i='x4') # clear ranking of importance/ effect on y
```




```
plot(boost_tree, i='x5') # clear ranking of importance/ effect on y
```



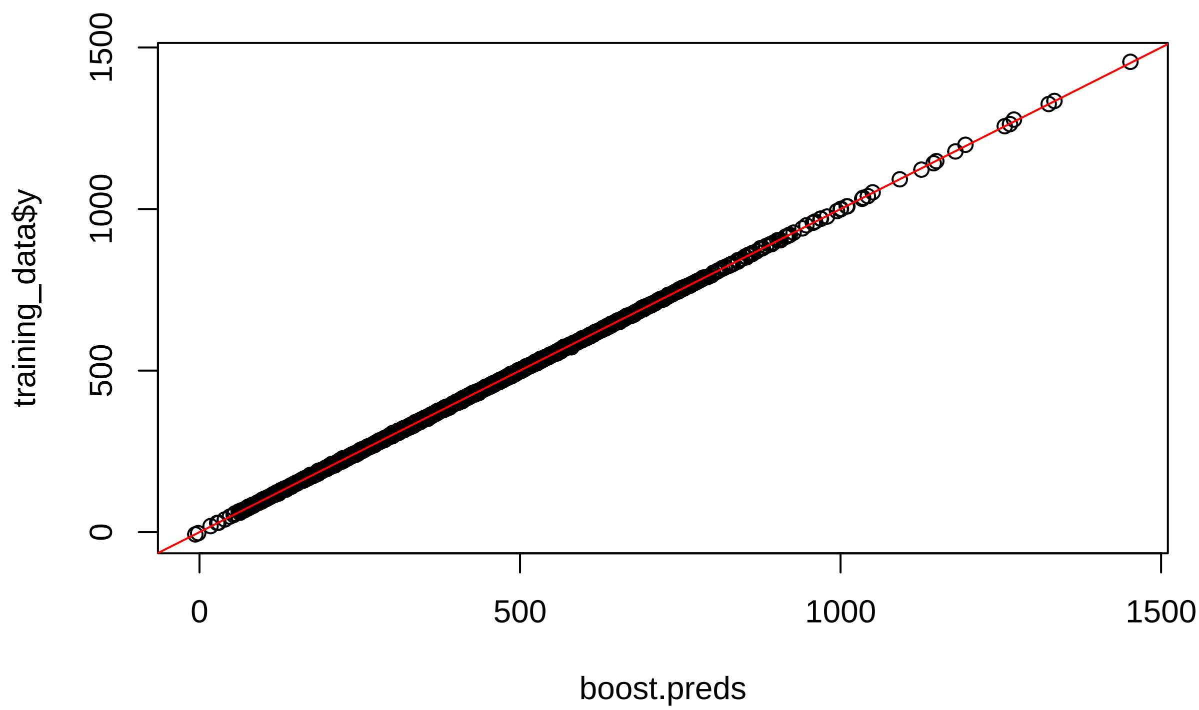

```
plot(boost_tree, i='x6') # clear ranking of importance/ effect on y
```




```
boost.preds <- predict(boost_tree, newdata = training_data)
mean((boost.preds - training_data$y)^2)
```

```
## [1] 2.613515
```

```
plot(boost.preds, training_data$y) # very close fit to training data and CEF (possibly over fitted)
abline(0, 1, col="red")
```




```
# least interpretable of the 3 models and slower than single tree
```

```
set.seed(50537)

### ID generalisation
test_data <- predictors |> mutate(y = CEF(x1, x2, x3, x4, x5, x6) + rnorm(n, sd = 12))
test <- sample(1: nrow(test_data), nrow(test_data) / 2)
test_data <- test_data[test, ]
head(test_data)
```

```
##             x1       x2        x3     x4    x5       x6        y
## 774   2.823146 5.658071 2.1019105    oak brass     wind 170.1908
## 1294 10.506369 9.677664 2.0585005 spruce  iron electric 536.5091
## 803   8.378568 9.015617 0.5859096    oak  iron electric 383.6380
## 375   8.477152 6.724260 2.5375892 walnut  iron     wind 599.3303
## 220   7.691171 2.453224 2.2060261    oak steel     coal 414.8883
## 1937  8.183691 3.273232 4.8018079    oak steel     coal 644.7158
```

```
tree.preds.test <- predict(single_tree, newdata = test_data)
mean((tree.preds.test - test_data$y)^2)
```

```
## [1] 11507.81
```

```r
prune.preds.test <- predict(prune_tree, newdata = test_data)
mean((prune.preds.test - test_data$y)^2)
```

```
## [1] 12990.16
```

```r
rf.preds.test <- predict(rand_forest, newdata = test_data)
mean((rf.preds.test - test_data$y)^2)
```

```
## [1] 865.2863
```

```r
boost.preds.test <- predict(boost_tree, newdata = test_data)
mean((boost.preds.test - test_data$y)^2)
```

```
## [1] 2.704527
```

```r
# in order of best test results: boost tree > random forest > single tree > pruned tree
# i choose to use boost tree from here as it's performed best on the test data
# (also see previous part for visualisation of tree model fits)


### OOD generalisation

## Concept shift
summary(training_data)
```

```
##        x1                  x2              x3              x4            x5
##  Min.   : 0.01142   Min.   :1.002   Min.   :0.503   oak   :1104   brass:513
##  1st Qu.: 4.29673   1st Qu.:3.213   1st Qu.:1.747   spruce: 292   iron :843
##  Median : 5.95779   Median :5.564   Median :2.850   walnut: 604   steel:644
##  Mean   : 5.98889   Mean   :5.535   Mean   :2.821
##  3rd Qu.: 7.60347   3rd Qu.:7.810   3rd Qu.:3.910
##  Max.   :16.02954   Max.   :9.999   Max.   :4.999
##         x6              y
##  coal    :990   Min.   :  -6.886
##  electric:264   1st Qu.: 252.185
##  wind    :746   Median : 381.512
##                 Mean   : 407.784
##                 3rd Qu.: 533.909
##                 Max.   :1455.801
```

```r
summary(test_data)
```

```
##        x1                  x2              x3              x4           x5
##  Min.   : 0.06984   Min.   :1.002   Min.   :0.508   oak   :542   brass:259
##  1st Qu.: 4.26921   1st Qu.:3.147   1st Qu.:1.772   spruce:145   iron :425
##  Median : 5.98840   Median :5.431   Median :2.828   walnut:313   steel:316
##  Mean   : 6.01774   Mean   :5.486   Mean   :2.818
##  3rd Qu.: 7.61929   3rd Qu.:7.870   3rd Qu.:3.896
##  Max.   :16.02954   Max.   :9.974   Max.   :4.999
##         x6              y
```

```
##  coal     :507   Min.   :  -6.886
##  electric:128   1st Qu.: 246.829
##  wind     :365   Median : 384.699
##                  Mean   : 410.730
##                  3rd Qu.: 536.601
##                  Max.   :1455.801
```

```r
mean((boost.preds.test - test_data$y)^2) # boost model on test data value
```

```
## [1] 2.704527
```

```r
concpt_btr <- test_data
concpt_btr$y <- concpt_btr$y - 0.001*(concpt_btr$x1)^2 + 0.01*(concpt_btr$x2) - 0.01*(concpt_btr$x3)
boost.concpt.btr.preds <- predict(boost_tree, newdata = concpt_btr)
mean((boost.concpt.btr.preds - concpt_btr$y)^2) # smaller number and so, more accurate
```

```
## [1] 2.703967
```

```r
# The training values of x1 and x3 are, on average, larger than the test values
# and so by decreasing x1 and x3's effect on y, it makes the model, that was made
# using the training data, fit the test data better.
# The training value of x2 is, on average, smaller than the test values and so,
# by increasing x2's effect on y, it makes the model, that was made using the
# training data, fit the test data better.



concpt_wrse <- test_data
concpt_wrse$y <- concpt_wrse$y + 5
boost.concpt.wrse.preds <- predict(boost_tree, newdata = concpt_wrse)
mean((boost.concpt.wrse.preds - concpt_wrse$y)^2) # larger number and so, less accurate
```

```
## [1] 28.03175
```

```r
# Linearly shifting the y (output) values by +5 causes the tree model to fit the data worse
```

```r
set.seed(50537)

## Covariate shift

mean((boost.preds.test - test_data$y)^2) # boost model on test data value
```

```
## [1] 2.704527
```

```r
summary(test_data) # viewing the data the comparison is being made to
```

```
##        x1                  x2              x3              x4           x5
##  Min.   : 0.06984   Min.   :1.002   Min.   :0.508   oak    :542   brass:259
##  1st Qu.: 4.26921   1st Qu.:3.147   1st Qu.:1.772   spruce:145   iron :425
##  Median : 5.98840   Median :5.431   Median :2.828   walnut:313   steel:316
```

```
## Mean    : 6.01774   Mean   :5.486   Mean   :2.818
## 3rd Qu.: 7.61929   3rd Qu.:7.870   3rd Qu.:3.896
## Max.   :16.02954   Max.   :9.974   Max.   :4.999
##        x6             y
## coal    :507   Min.   :  -6.886
## electric:128   1st Qu.: 246.829
## wind    :365   Median : 384.699
##                Mean   : 410.730
##                3rd Qu.: 536.601
##                Max.   :1455.801
```

```
summary(training_data) # viewing the data the model was trained on
```

```
##       x1                 x2              x3              x4           x5
## Min.   : 0.01142   Min.   :1.002   Min.   :0.503   oak   :1104   brass:513
## 1st Qu.: 4.29673   1st Qu.:3.213   1st Qu.:1.747   spruce: 292   iron :843
## Median : 5.95779   Median :5.564   Median :2.850   walnut: 604   steel:644
## Mean   : 5.98889   Mean   :5.535   Mean   :2.821
## 3rd Qu.: 7.60347   3rd Qu.:7.810   3rd Qu.:3.910
## Max.   :16.02954   Max.   :9.999   Max.   :4.999
##        x6             y
## coal    :990   Min.   :  -6.886
## electric:264   1st Qu.: 252.185
## wind    :746   Median : 381.512
##                Mean   : 407.784
##                3rd Qu.: 533.909
##                Max.   :1455.801
```

```
strength_covar_btr <- abs(rnorm(2000, mean=6.0001, sd=2.5)) # increased mean of x1 (strength) predictor
predictors_covar_btr <- data.frame(x1 = strength_covar_btr, x2 = speed, x3 = agility,
                         x4 = wood_var, x5 = metal_var, x6 = prop_var)
set.seed(50537)
covar_btr <- predictors_covar_btr |> mutate(y = (CEF(x1, x2, x3, x4, x5, x6)
                                            + rnorm(n, sd = 12)))
summary(covar_btr)
```

```
##       x1                 x2              x3              x4           x5
## Min.   : 0.01152   Min.   :1.002   Min.   :0.503   oak   :1104   brass:513
## 1st Qu.: 4.29683   1st Qu.:3.213   1st Qu.:1.747   spruce: 292   iron :843
## Median : 5.95789   Median :5.564   Median :2.850   walnut: 604   steel:644
## Mean   : 5.98898   Mean   :5.535   Mean   :2.821
## 3rd Qu.: 7.60357   3rd Qu.:7.810   3rd Qu.:3.910
## Max.   :16.02964   Max.   :9.999   Max.   :4.999
##        x6             y
## coal    :990   Min.   :  -6.887
## electric:264   1st Qu.: 252.189
## wind    :746   Median : 381.516
##                Mean   : 407.789
##                3rd Qu.: 533.917
##                Max.   :1455.813
```

```r
boost.covar.btr.preds <- predict(boost_tree, newdata = covar_btr)
mean((boost.covar.btr.preds - covar_btr$y)^2) # smaller number and so, more accurate
```

```
## [1] 2.700593
```

```r
# By slightly increasing the mean average value of x1 I have moved it closer to
# a value which best fits the model based off of the training data and so the prediction
# of this data set has a lower mean squared error / is better at predicting y.


set.seed(50537)
strength_covar_wrse <- abs(rnorm(2000, mean=6, sd=3)) # increased variance of x1 (strength) predictor
predictors_covar_wrse <- data.frame(x1 = strength_covar_wrse, x2 = speed, x3 = agility,
                         x4 = wood_var, x5 = metal_var, x6 = prop_var)
set.seed(50537)
covar_wrse <- predictors_covar_wrse |> mutate(y = (CEF(x1, x2, x3, x4, x5, x6)
                                                   + rnorm(n, sd = 12)))

boost.covar.wrse.preds <- predict(boost_tree, newdata = covar_wrse)
mean((boost.covar.wrse.preds - covar_wrse$y)^2) # larger number and so, less accurate
```

```
## [1] 560.7266
```

```r
# By increasing the variance of the normally distributed variable, x1 (strength),
# which is significantly influential on y, I have caused the randomly generated
# values of x1 to be less consistent and so the values of y (the outcome) to be
# less consistent without having an effect on the conditional distribution of y.
# And so the model is worse at predicting y from the predictors.
```