2. $X | Y=j \sim N(\mu_j, \sigma^2)$

$h^*(x) = \operatorname{argmax}_j \mathbb{P}(Y=j \mid X=x)$

$$\mathbb{P}(Y=j \mid X=x) = \frac{\mathbb{P}(X=x \mid Y=j)\,\mathbb{P}(Y=j)}{\mathbb{P}(X=x)}$$

$\mathbb{P}(X=x)$ is the same for all Y, so we want to find

$\operatorname{argmax}_j \mathbb{P}(X=x \mid Y=j)\,\mathbb{P}(Y=j)$

$$= \operatorname{argmax}_j \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu_j}{\sigma}\right)^2} \mathbb{P}(Y=j)$$

This is positive and $\frac{1}{\sigma\sqrt{2\pi}}$ constant, so we can ignore $\frac{1}{\sigma\sqrt{2\pi}}$ and take the log of the rest

$$= \operatorname{argmax}_j -\frac{1}{2}\frac{x^2 - 2x\mu_j + \mu_j^2}{\sigma^2} + \log \mathbb{P}(Y=j)$$

$$= \operatorname{argmax}_j x\sigma^{-2}\mu_j - \frac{1}{2}\mu_j(\sigma^2)^{-1}\mu_j + \log \pi_j.$$

(This agrees with lecture notes 17 Thm 5)

We can estimate π_j by $\hat{\pi}_j = \frac{1}{n}\sum_{i=1}^{n} I(y_i = j)$ where n is the sample size.

401 HW7

```r
#Problem 1
#install.packages("AmesHousing")
set.seed(36401)
library(AmesHousing)
ames = make_ames()
names(ames)
```

```
##  [1] "MS_SubClass"        "MS_Zoning"          "Lot_Frontage"
##  [4] "Lot_Area"           "Street"             "Alley"
##  [7] "Lot_Shape"          "Land_Contour"       "Utilities"
## [10] "Lot_Config"         "Land_Slope"         "Neighborhood"
## [13] "Condition_1"        "Condition_2"        "Bldg_Type"
## [16] "House_Style"        "Overall_Qual"       "Overall_Cond"
## [19] "Year_Built"         "Year_Remod_Add"     "Roof_Style"
## [22] "Roof_Matl"          "Exterior_1st"       "Exterior_2nd"
## [25] "Mas_Vnr_Type"       "Mas_Vnr_Area"       "Exter_Qual"
## [28] "Exter_Cond"         "Foundation"         "Bsmt_Qual"
## [31] "Bsmt_Cond"          "Bsmt_Exposure"      "BsmtFin_Type_1"
## [34] "BsmtFin_SF_1"       "BsmtFin_Type_2"     "BsmtFin_SF_2"
## [37] "Bsmt_Unf_SF"        "Total_Bsmt_SF"      "Heating"
## [40] "Heating_QC"         "Central_Air"        "Electrical"
## [43] "First_Flr_SF"       "Second_Flr_SF"      "Low_Qual_Fin_SF"
## [46] "Gr_Liv_Area"        "Bsmt_Full_Bath"     "Bsmt_Half_Bath"
## [49] "Full_Bath"          "Half_Bath"          "Bedroom_AbvGr"
## [52] "Kitchen_AbvGr"      "Kitchen_Qual"       "TotRms_AbvGrd"
## [55] "Functional"         "Fireplaces"         "Fireplace_Qu"
## [58] "Garage_Type"        "Garage_Finish"      "Garage_Cars"
## [61] "Garage_Area"        "Garage_Qual"        "Garage_Cond"
## [64] "Paved_Drive"        "Wood_Deck_SF"       "Open_Porch_SF"
## [67] "Enclosed_Porch"     "Three_season_porch" "Screen_Porch"
## [70] "Pool_Area"          "Pool_QC"            "Fence"
## [73] "Misc_Feature"       "Misc_Val"           "Mo_Sold"
## [76] "Year_Sold"          "Sale_Type"          "Sale_Condition"
## [79] "Sale_Price"         "Longitude"          "Latitude"
```

```r
X = ames[, -79]
Y = ames[, 79]
X = Filter(is.numeric, X)
dim(X)
```

```
## [1] 2930   34
```

```r
names(X)
```

```
##  [1] "Lot_Frontage"       "Lot_Area"           "Year_Built"
##  [4] "Year_Remod_Add"     "Mas_Vnr_Area"       "BsmtFin_SF_1"
##  [7] "BsmtFin_SF_2"       "Bsmt_Unf_SF"        "Total_Bsmt_SF"
## [10] "First_Flr_SF"       "Second_Flr_SF"      "Low_Qual_Fin_SF"
## [13] "Gr_Liv_Area"        "Bsmt_Full_Bath"     "Bsmt_Half_Bath"
## [16] "Full_Bath"          "Half_Bath"          "Bedroom_AbvGr"
## [19] "Kitchen_AbvGr"      "TotRms_AbvGrd"      "Fireplaces"
## [22] "Garage_Cars"        "Garage_Area"        "Wood_Deck_SF"
## [25] "Open_Porch_SF"      "Enclosed_Porch"     "Three_season_porch"
## [28] "Screen_Porch"       "Pool_Area"          "Misc_Val"
## [31] "Mo_Sold"            "Year_Sold"          "Longitude"
## [34] "Latitude"
```

```
#str(X)
Y = unlist(Y)
X = as.matrix(X)
n = nrow(X)
I = sample(1:n, size = 500, replace = FALSE)
Xtest = X[I, ]
Ytest = Y[I]
Xtrain = X[-I, ]
Ytrain = Y[-I]
```

```
#(a)
library(mgcv)
```

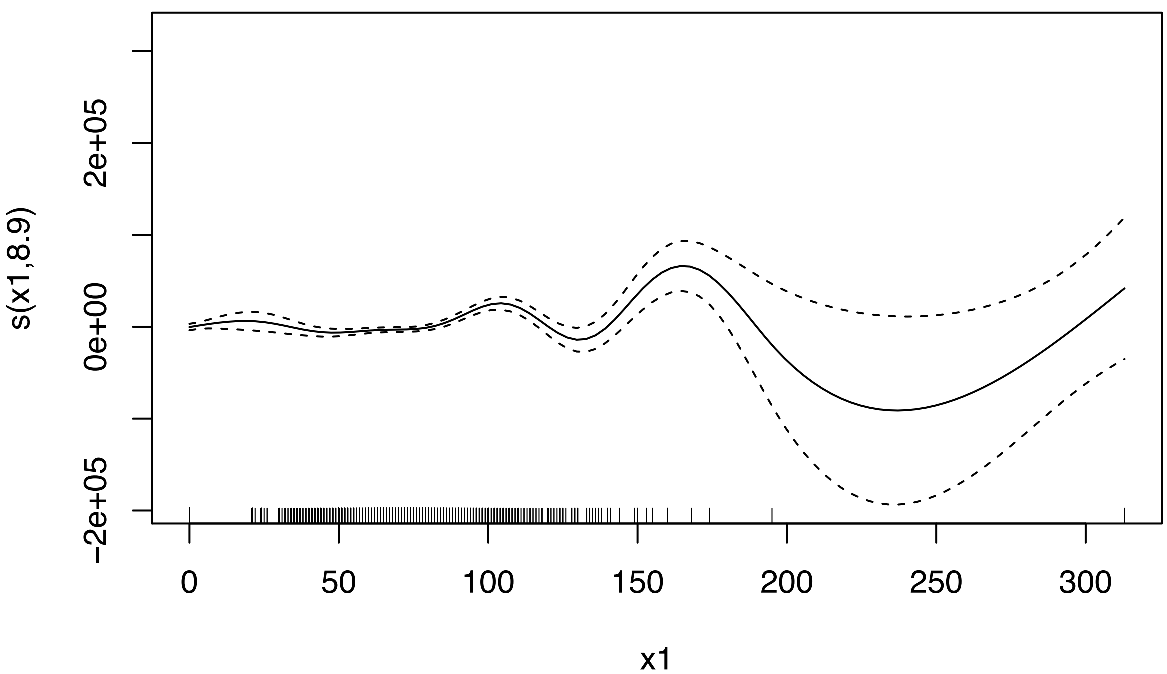
```
## Loading required package: nlme

## This is mgcv 1.8-31. For overview type 'help("mgcv-package")'.
```

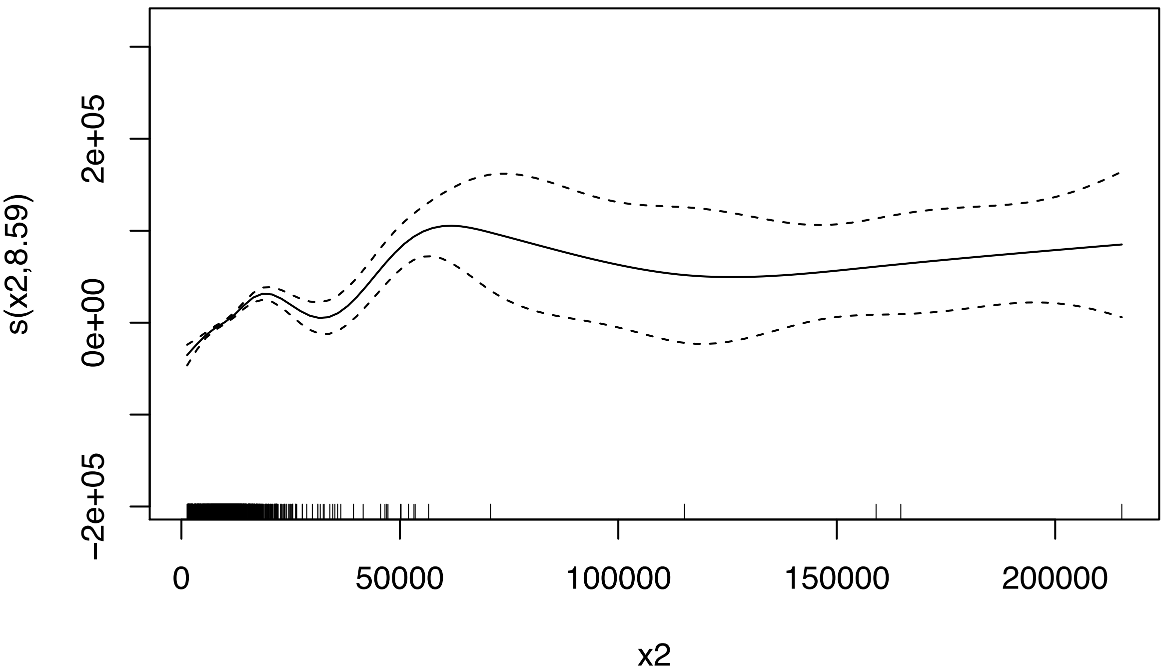
```
namesx = paste("x",1:10,sep="")
names = c("y",namesx)
df = data.frame(Ytrain,Xtrain)
names(df) = names
outa = gam(y ~ s(x1) + s(x2) + s(x3) + s(x4) + s(x5) + x6 + s(x7) + s(x8) + s(x9) + s(x10), data = df)
summary(outa)
```

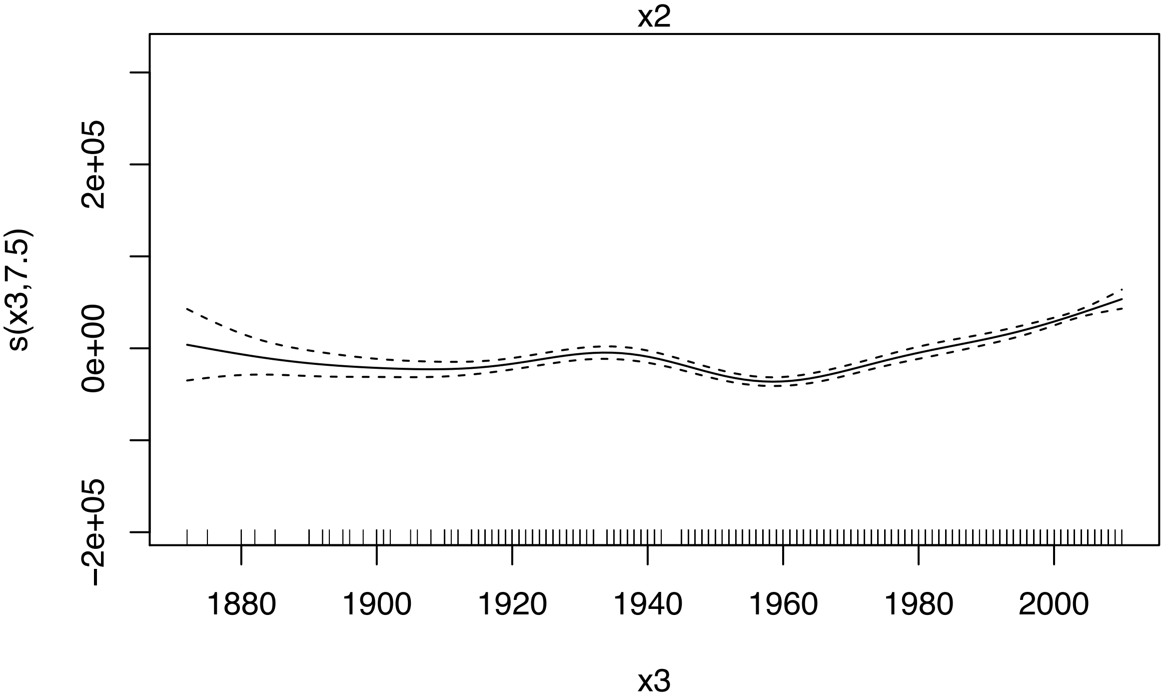
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ s(x1) + s(x2) + s(x3) + s(x4) + s(x5) + x6 + s(x7) + s(x8) +
##     s(x9) + s(x10)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 179714.4     2133.0  84.253   <2e-16 ***
## x6             127.4      481.7   0.264    0.791
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

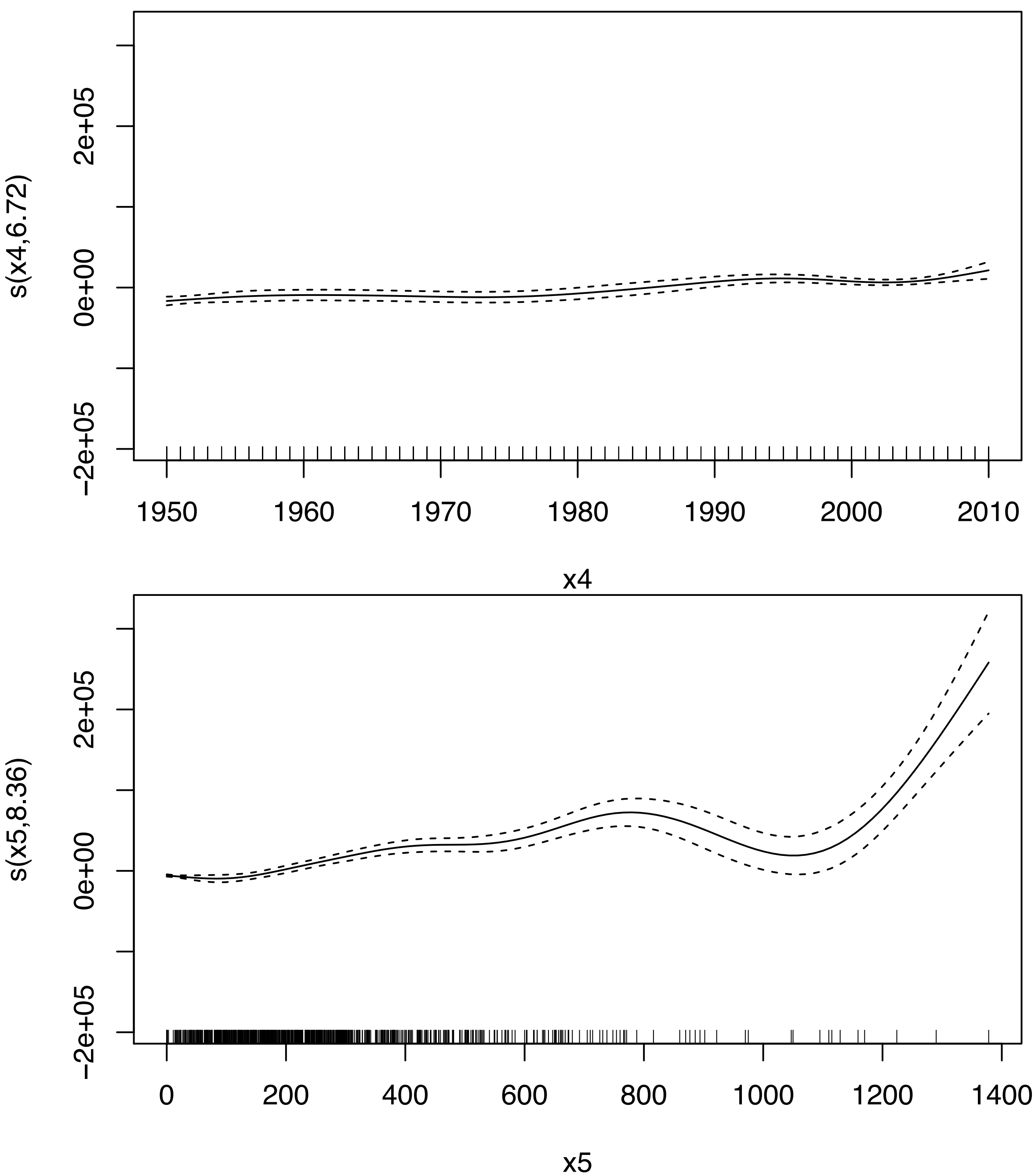
```
## Approximate significance of smooth terms:
##           edf Ref.df       F  p-value
## s(x1)  8.903  8.995   8.797 3.51e-13 ***
## s(x2)  8.591  8.947  23.531  < 2e-16 ***
## s(x3)  7.500  8.396  57.900  < 2e-16 ***
## s(x4)  6.725  7.863  13.717  < 2e-16 ***
## s(x5)  8.359  8.874  33.989  < 2e-16 ***
## s(x7)  3.149  3.904   2.875   0.0247 *
## s(x8)  7.561  8.455  17.191  < 2e-16 ***
## s(x9)  8.642  8.954  28.720  < 2e-16 ***
## s(x10) 1.000  1.000 112.013  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.78   Deviance explained = 78.6%
## GCV = 1.4246e+09  Scale est. = 1.388e+09  n = 2430
```

```
plot(outa)
```

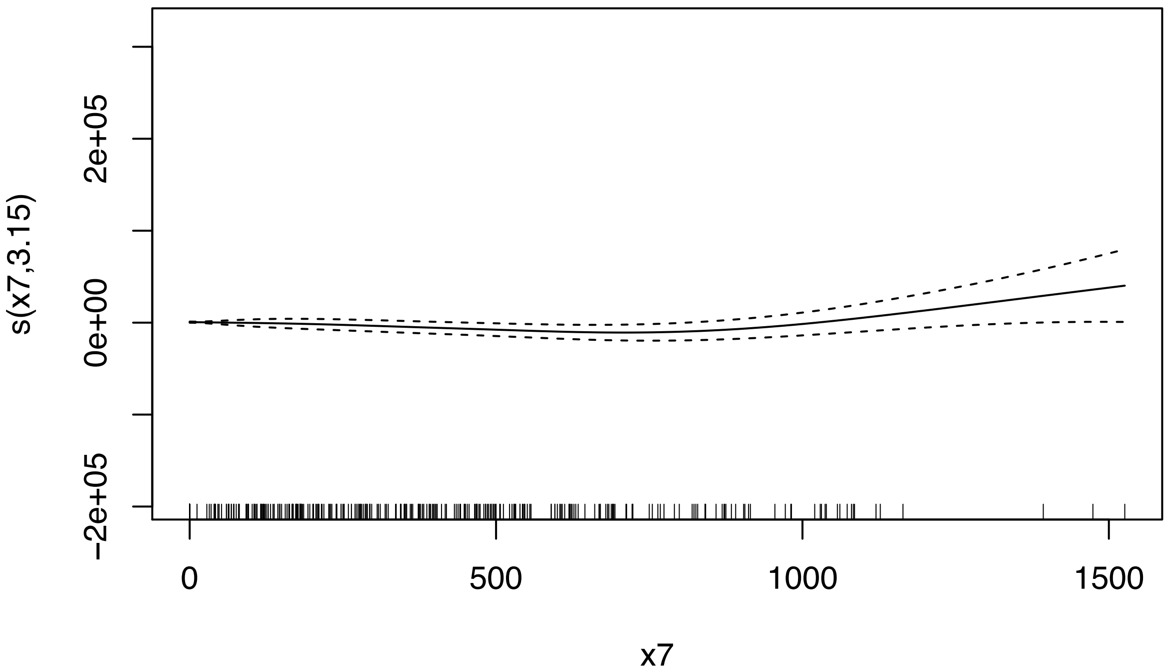



s(x2,8.59)
−2e+05
0e+00
2e+05
0
50000
100000
150000
200000
x2

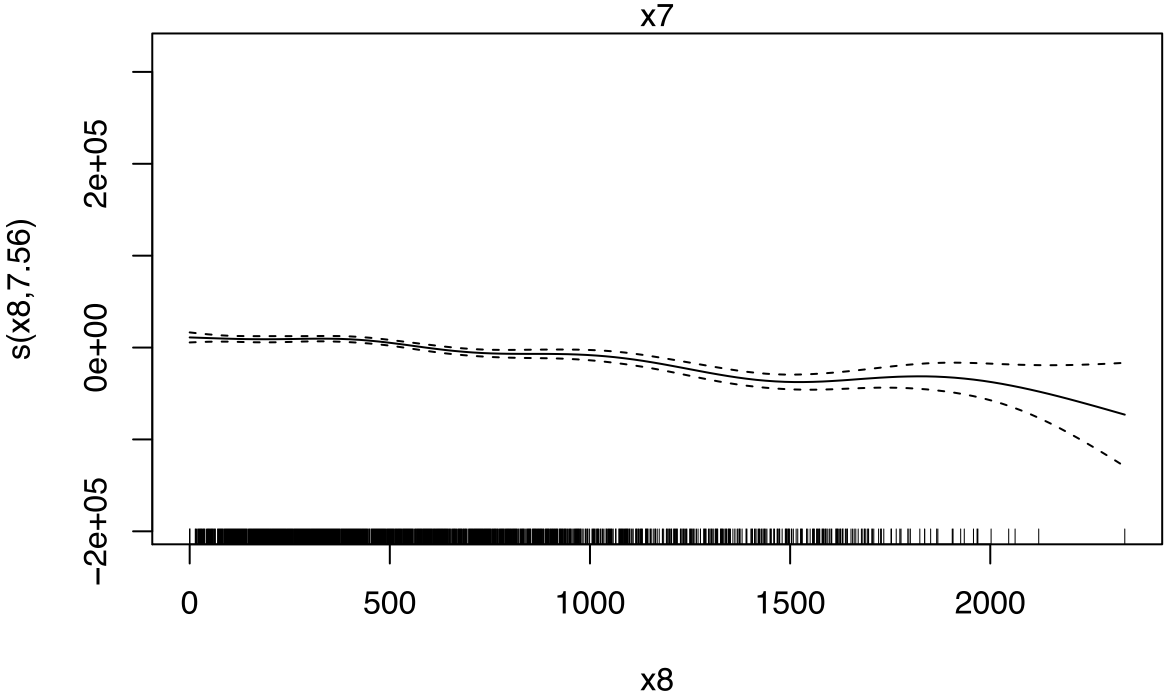

s(x3,7.5)
−2e+05
0e+00
2e+05
1880
1900
1920
1940
1960
1980
2000
x3



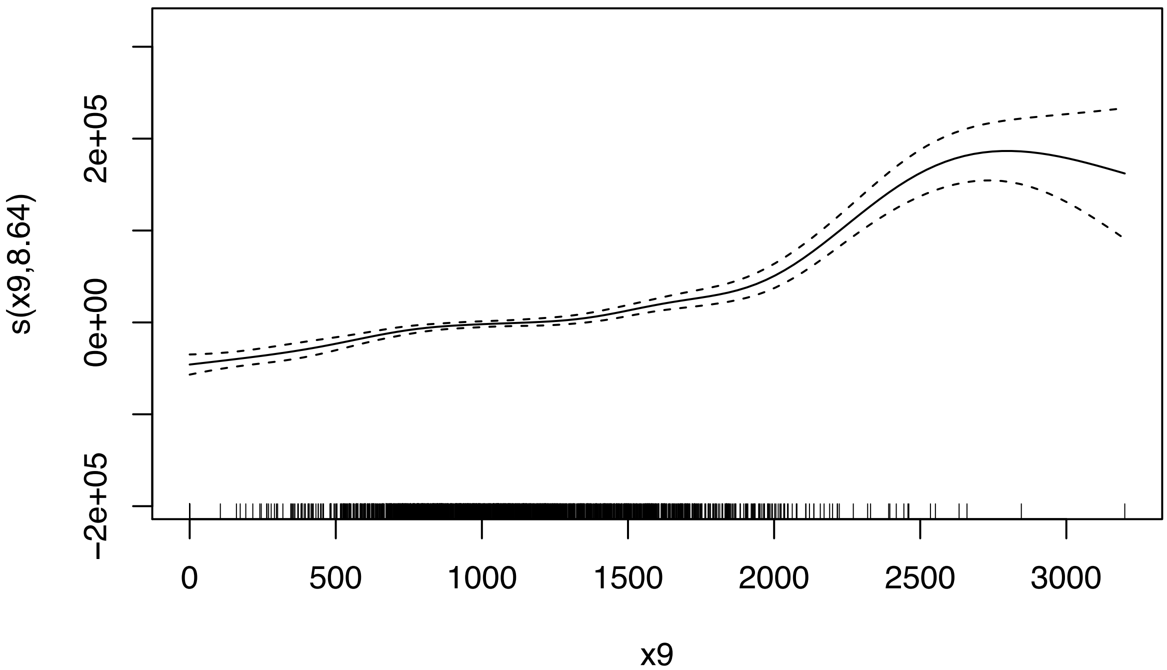
s(x4,6.72)
2e+05
0e+00
−2e+05
1950
1960
1970
1980
1990
2000
2010
x4
s(x5,8.36)
2e+05
0e+00
−2e+05
0
200
400
600
800
1000
1200
1400
x5

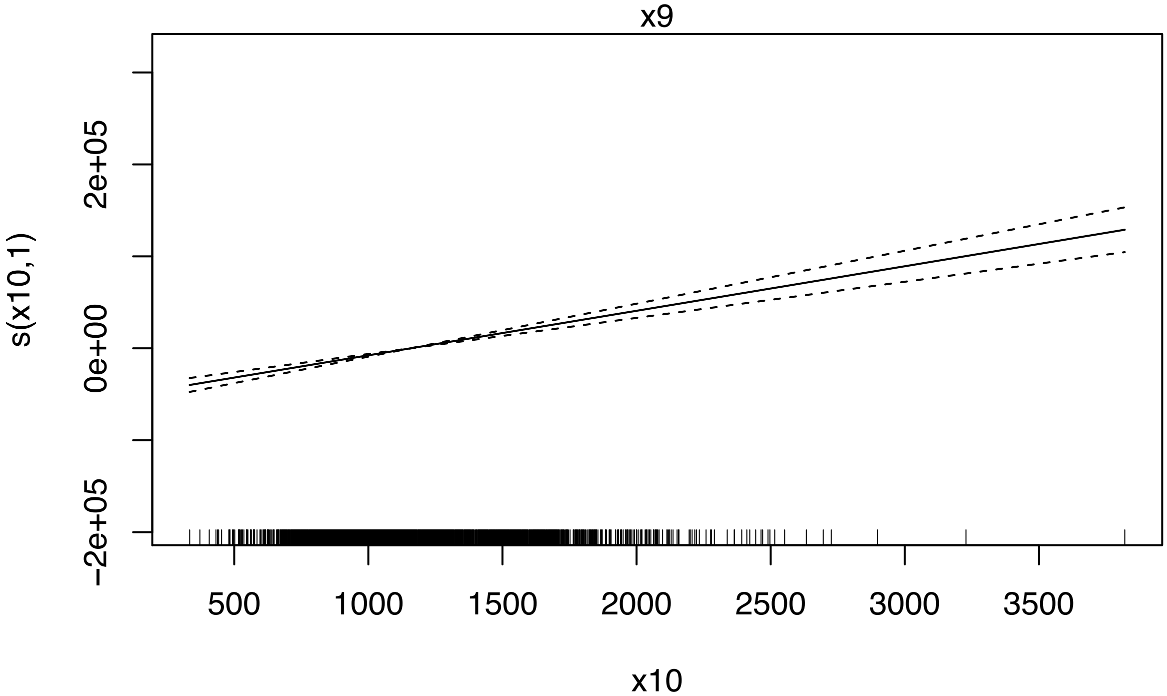


s(x7,3.15)
2e+05
0e+00
−2e+05
0
500
1000
1500
x7



s(x8,7.56)
2e+05
0e+00
−2e+05
0
500
1000
1500
2000
x8





x1, x2, …, x10 corresponds to "Lot_Frontage", "Lot_Area", "Year_Built", "Year_Remod_Add", "Mas_Vnr_Area", "BsmtFin_SF_1", "BsmtFin_SF_2", "Bsmt_Unf_SF", "Total_Bsmt_SF", and "First_Flr_SF" respectively.The estimates of the smooth functions and its confidence interval is shown in the plots. Based on the R^2, about 78.6% of the data can be explained by this model.

```r
#(b)
set.seed(36401)
library(randomForest)
```

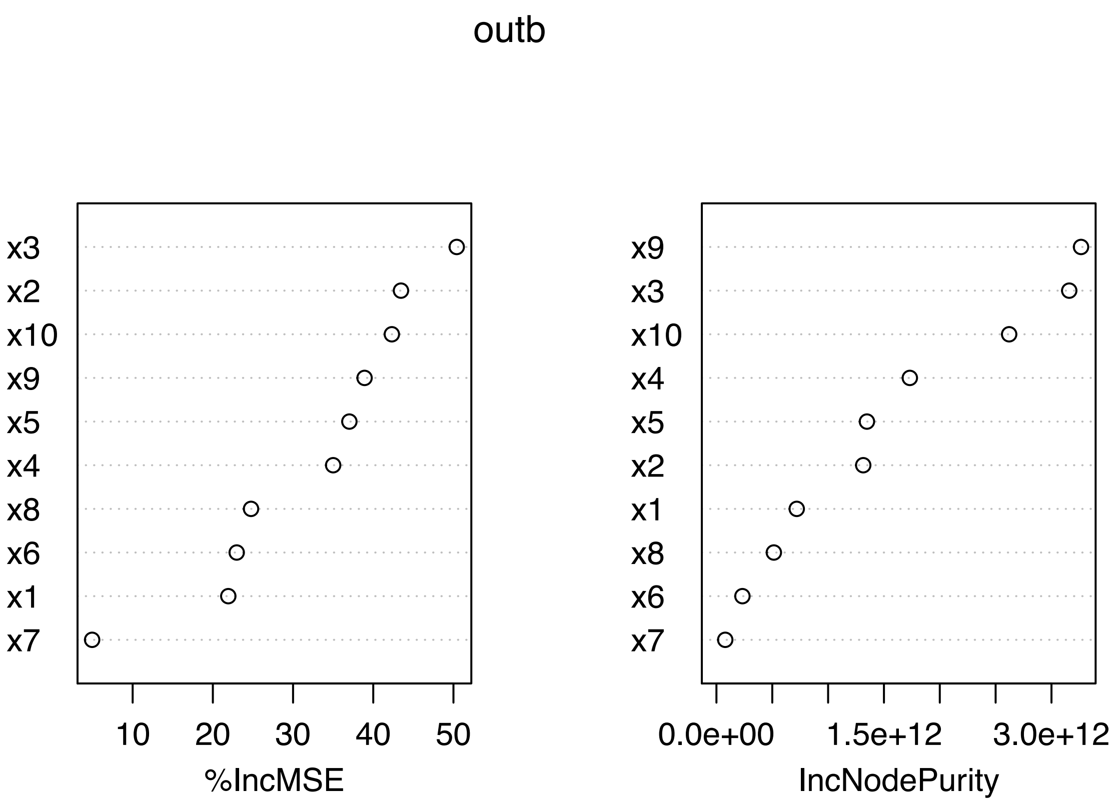
```
## randomForest 4.6-14
```

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

```r
f = paste(namesx, collapse=" + ",sep="")
f = paste("y ~ ",f)
f = as.formula(f)
outb = randomForest(formula=f, importance = TRUE, data=df)
print(outb$importance)
```

```
##        %IncMSE IncNodePurity
## x1   210251014  7.190341e+11
## x2   540834557  1.314396e+12
## x3  2006850021  3.159003e+12
## x4  1055687345  1.731692e+12
## x5   438234372  1.347651e+12
## x6   154198006  2.328639e+11
## x7     9261672  7.900568e+10
## x8   221267710  5.141190e+11
## x9  1362276635  3.264654e+12
## x10 1219842787  2.620966e+12
```

```r
varImpPlot(outb)
```



The random forest model suggests that Year_Built, First_Flr_SF, and First_Flr_SF (top 3 in both measures) are the most important variables. They are also important in the additive model. Both models suggest x7 is the least important.

```r
#(c)
dfnew = data.frame(Xtest)
names(dfnew) = namesx
```

```r
preda = predict(outa, newdata = dfnew)
predb = predict(outb, newdata = dfnew)
B1 = (Ytest-preda)^2
Rhat1 = (1/500)*(sum(B1))
print(Rhat1)
```

```
## [1] 2837224682
```

```r
s = sqrt((1/499)*sum((B1-Rhat1)^2))
z = -qnorm(0.025)
low1 = Rhat1 - (z*s)/sqrt(500)
print(low1)
```

```
## [1] 1534255795
```

```r
up1 = Rhat1 + (z*s)/sqrt(500)
print(up1)
```

```
## [1] 4140193570
```

```r
B2 = (Ytest-predb)^2
Rhat2 = (1/500)*(sum(B2))
print(Rhat2)
```

```
## [1] 2098587190
```

```r
s = sqrt((1/499)*sum((B2-Rhat2)^2))
z = -qnorm(0.025)
low2 = Rhat2 - (z*s)/sqrt(500)
print(low2)
```

```
## [1] 1066363752
```

```r
up2 = Rhat2 + (z*s)/sqrt(500)
print(up2)
```

```
## [1] 3130810629
```

```r
diff = Rhat1-Rhat2
print(diff)
```

```
## [1] 738637492
```

```r
s = sqrt((1/499)*sum((B1-B2-diff)^2))
z = -qnorm(0.025)
low = diff - (z*s)/sqrt(500)
print(low)
```

```
## [1] -171793959
```

```
up = diff + (z*s)/sqrt(500)
print(up)
```

```
## [1] 1649068943
```

The prediction error for the additive model is 2837224682, with a 95% CI of [1534255795, 4140193570]. The prediction error for the forest model is 2098587190, with a 95% CI of [1066363752, 3130810629]. The difference is 738637492, and the 95% confidence interval for their difference is [-171793959, 1649068943] (Additive-Forest), so the random forest is not necessarily better.

```
#d
set.seed(36401)
n = ncol(Xtrain)
namesx = paste("x",1:n,sep="")
names = c("y",namesx)
df = data.frame(Ytrain,Xtrain)
names(df) = names
outd = randomForest(y ~ ., importance = TRUE, data=df)
dfnew = data.frame(Xtest)
names(dfnew) = namesx
predd = predict(outd, newdata = dfnew)
B = (Ytest-predd)^2
Rhat = (1/500)*(sum(B))
print(Rhat)
```

```
## [1] 1363080982
```

```
s = sqrt((1/499)*sum((B-Rhat)^2))
z = -qnorm(0.025)
low = Rhat - (z*s)/sqrt(500)
print(low)
```

```
## [1] 473923667
```

```
up = Rhat + (z*s)/sqrt(500)
print(up)
```

```
## [1] 2252238296
```

```
diff = Rhat2-Rhat
print(diff)
```

```
## [1] 735506209
```

```
s = sqrt((1/499)*sum((B2-B-diff)^2))
z = -qnorm(0.025)
low = diff - (z*s)/sqrt(500)
print(low)
```

```
## [1] 356132745
```

```r
up = diff + (z*s)/sqrt(500)
print(up)
```

```
## [1] 1114879673
```

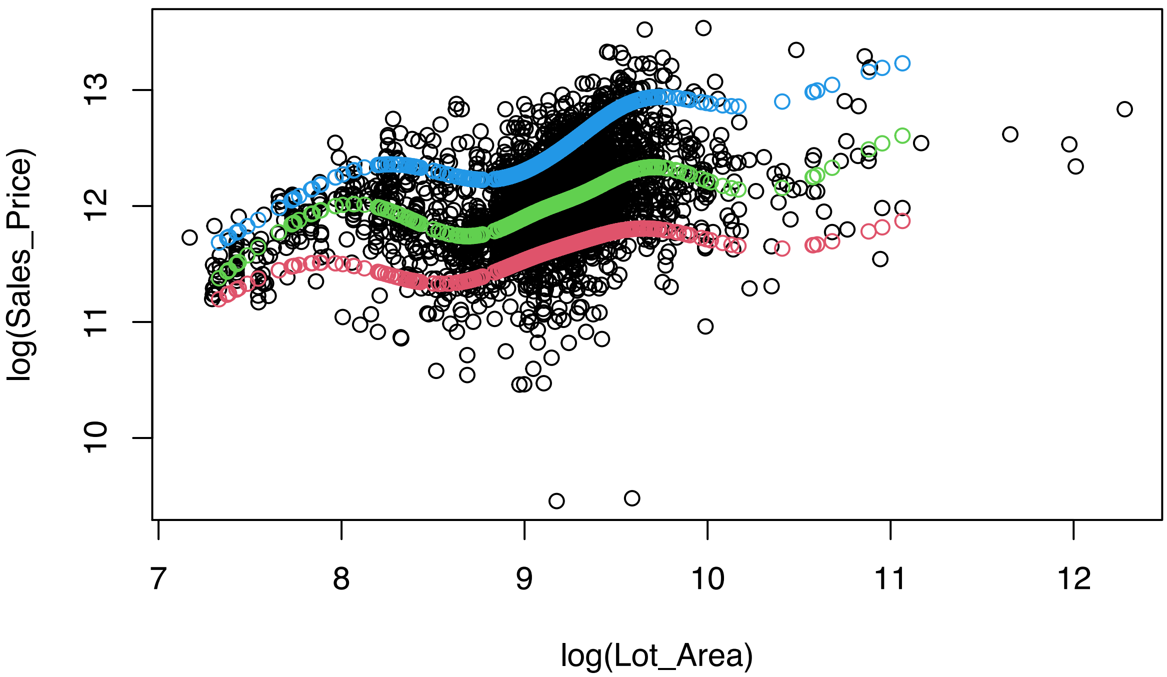
Now the prediction error is 1363080982 with a 95% CI of [473923667, 2252238296], which is smaller than the model using only the first 10 covariates. Their difference is 735506209 with a 95% CI of [356132745, 1114879673].

```r
#(e)
set.seed(36401)
library(qgam)
library(AmesHousing)
ames = make_ames()
names(ames)
```

```
##  [1] "MS_SubClass"        "MS_Zoning"          "Lot_Frontage"
##  [4] "Lot_Area"           "Street"             "Alley"
##  [7] "Lot_Shape"          "Land_Contour"       "Utilities"
## [10] "Lot_Config"         "Land_Slope"         "Neighborhood"
## [13] "Condition_1"        "Condition_2"        "Bldg_Type"
## [16] "House_Style"        "Overall_Qual"       "Overall_Cond"
## [19] "Year_Built"         "Year_Remod_Add"     "Roof_Style"
## [22] "Roof_Matl"          "Exterior_1st"       "Exterior_2nd"
## [25] "Mas_Vnr_Type"       "Mas_Vnr_Area"       "Exter_Qual"
## [28] "Exter_Cond"         "Foundation"         "Bsmt_Qual"
## [31] "Bsmt_Cond"          "Bsmt_Exposure"      "BsmtFin_Type_1"
## [34] "BsmtFin_SF_1"       "BsmtFin_Type_2"     "BsmtFin_SF_2"
## [37] "Bsmt_Unf_SF"        "Total_Bsmt_SF"      "Heating"
## [40] "Heating_QC"         "Central_Air"        "Electrical"
## [43] "First_Flr_SF"       "Second_Flr_SF"      "Low_Qual_Fin_SF"
## [46] "Gr_Liv_Area"        "Bsmt_Full_Bath"     "Bsmt_Half_Bath"
## [49] "Full_Bath"          "Half_Bath"          "Bedroom_AbvGr"
## [52] "Kitchen_AbvGr"      "Kitchen_Qual"       "TotRms_AbvGrd"
## [55] "Functional"         "Fireplaces"         "Fireplace_Qu"
## [58] "Garage_Type"        "Garage_Finish"      "Garage_Cars"
## [61] "Garage_Area"        "Garage_Qual"        "Garage_Cond"
## [64] "Paved_Drive"        "Wood_Deck_SF"       "Open_Porch_SF"
## [67] "Enclosed_Porch"     "Three_season_porch" "Screen_Porch"
## [70] "Pool_Area"          "Pool_QC"            "Fence"
## [73] "Misc_Feature"       "Misc_Val"           "Mo_Sold"
## [76] "Year_Sold"          "Sale_Type"          "Sale_Condition"
## [79] "Sale_Price"         "Longitude"          "Latitude"
```

```r
attach(ames)
I = sample(1:2930, size = 500, replace = FALSE)
test = ames[I,]
train = ames[-I,]
tau = c(0.1, 0.5, 0.9)
plot(log(Lot_Area), log(Sale_Price), xlab = "log(Lot_Area)", ylab = "log(Sales_Price)")
for (i in 1:3) {
  fit = qgam(log(Sale_Price) ~ s(log(Lot_Area)), data = train, qu = tau[i])
```

```
  pred = predict(fit, newdata = test, se = TRUE)
  points(log(test$Lot_Area), pred$fit, col = i+1)
}
```




```
## Estimating learning rate. Each dot corresponds to a loss evaluation.
## qu = 0.1........done
## Estimating learning rate. Each dot corresponds to a loss evaluation.
## qu = 0.5........done
## Estimating learning rate. Each dot corresponds to a loss evaluation.
## qu = 0.9........done
```

```
#Problem 3
#install.packages("ISLR")
library(ISLR)
attach(Auto)
names(Auto)
```

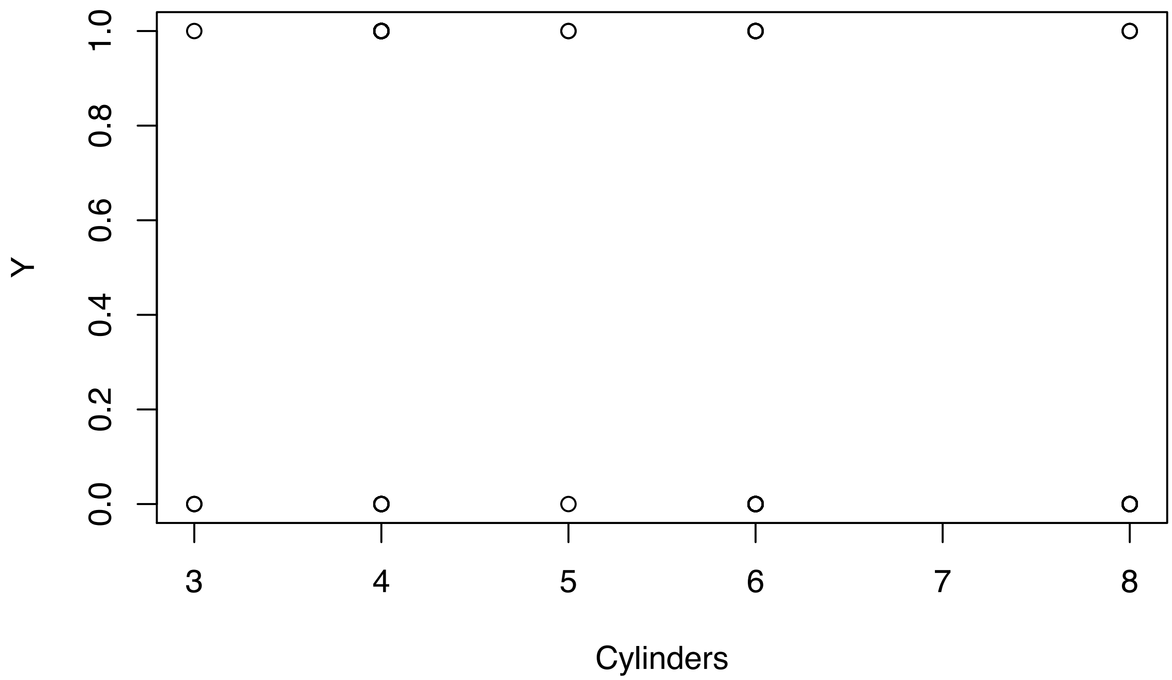
```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

```
str(Auto)
```

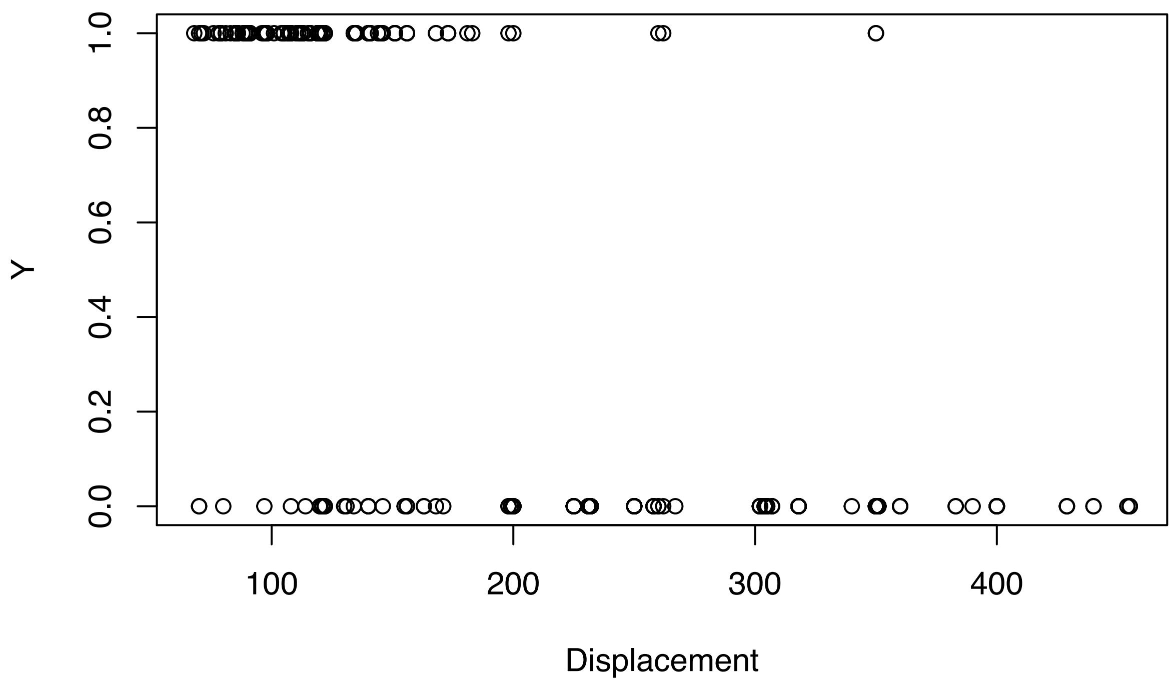
```
## 'data.frame':    392 obs. of  9 variables:
##  $ mpg         : num  18 15 18 16 17 15 14 14 14 15 ...
##  $ cylinders   : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
##  $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
##  $ weight      : num  3504 3693 3436 3433 3449 ...
##  $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
##  $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
```

```
##  $ origin      : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ name        : Factor w/ 304 levels "amc ambassador brougham",..: 49 36 231 14 161 141 54 223 241 :
```

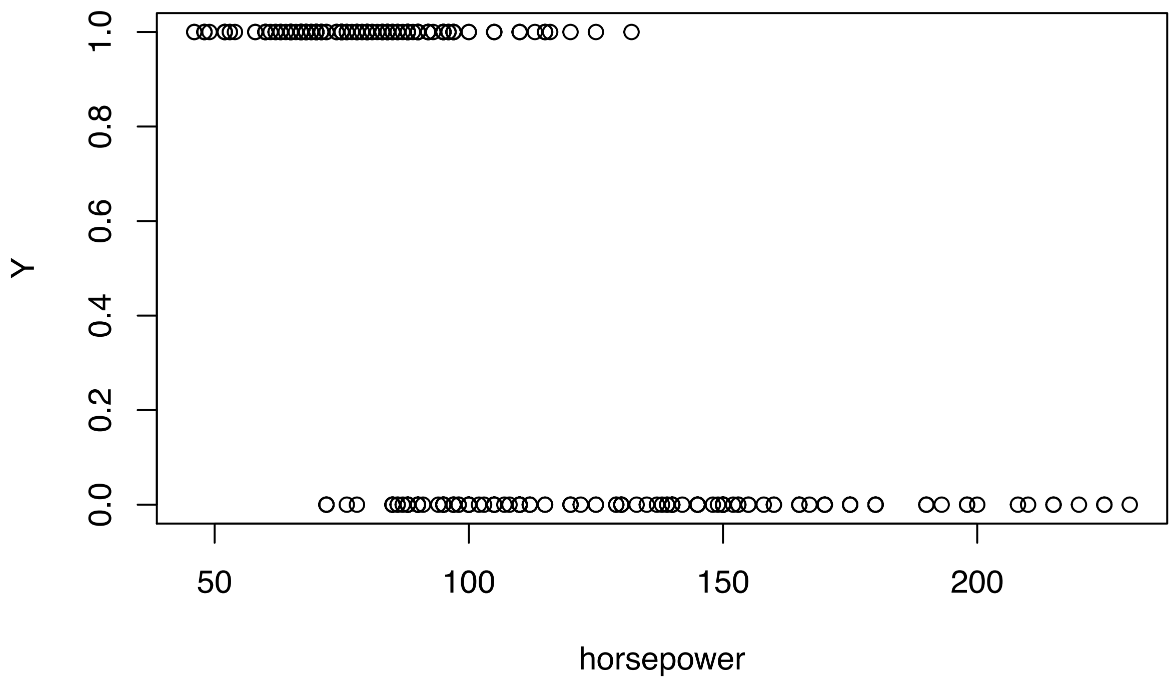
```
m = median(mpg)
Y = ifelse(mpg > m, 1, 0)
#(a)
plot(cylinders, Y, xlab = "Cylinders", ylab = "Y")
```



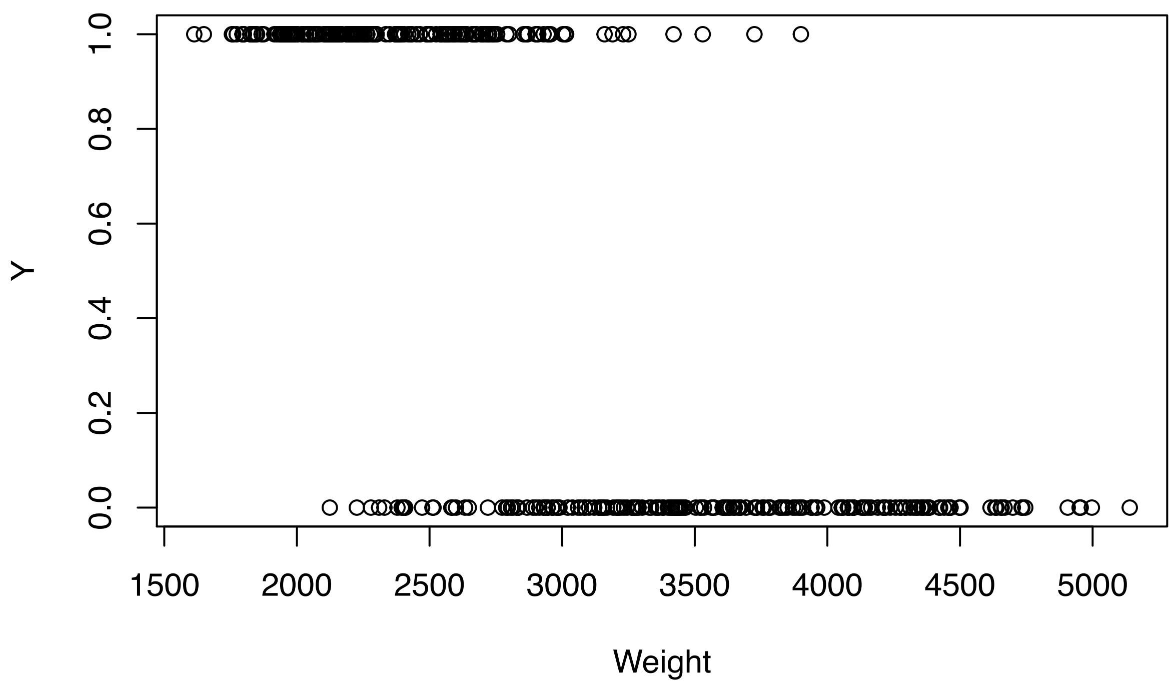

```
plot(displacement, Y, xlab = "Displacement", ylab = "Y")
```



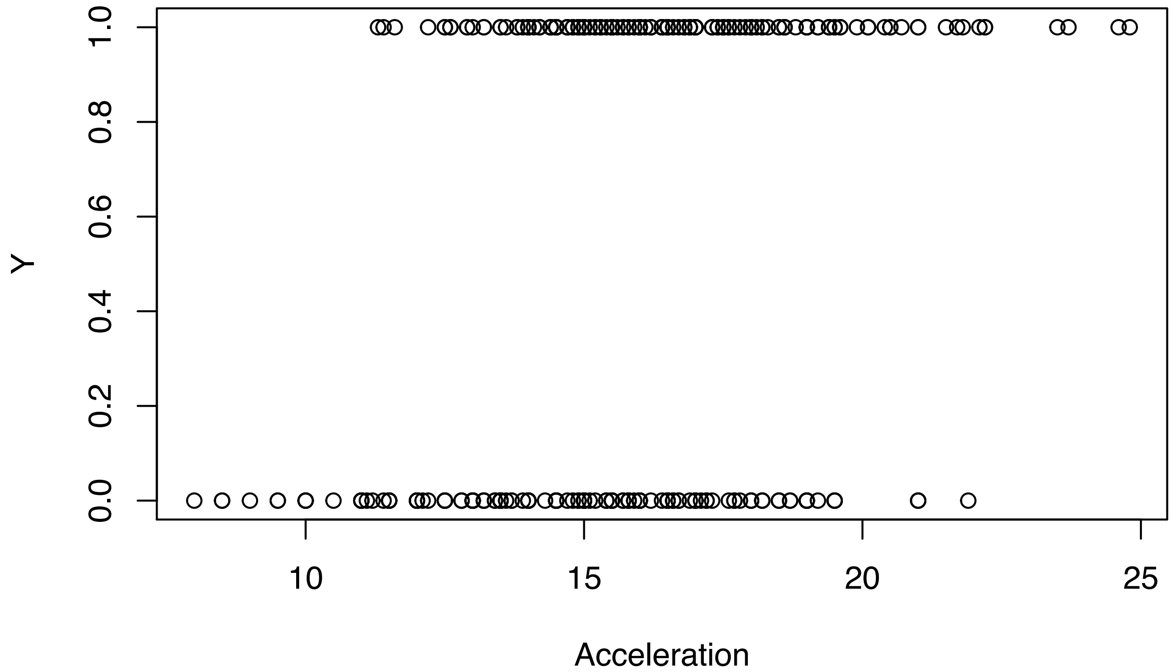
```
plot(horsepower, Y, xlab = "horsepower", ylab = "Y")
```



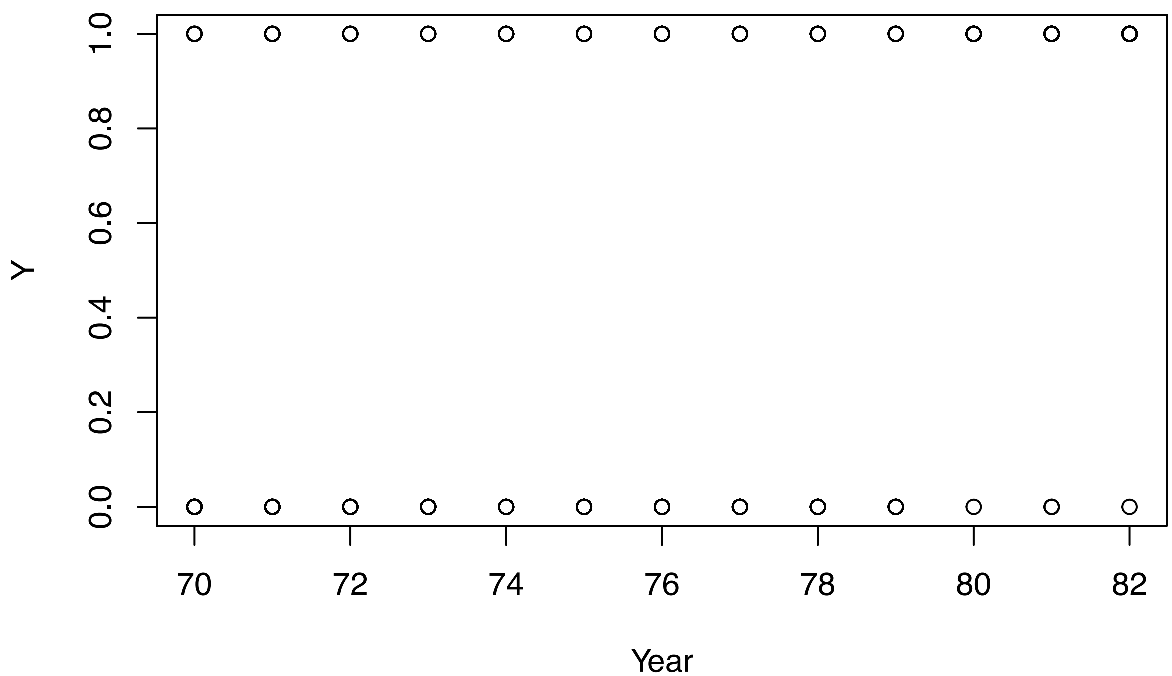

```
plot(weight, Y, xlab = "Weight", ylab = "Y")
```



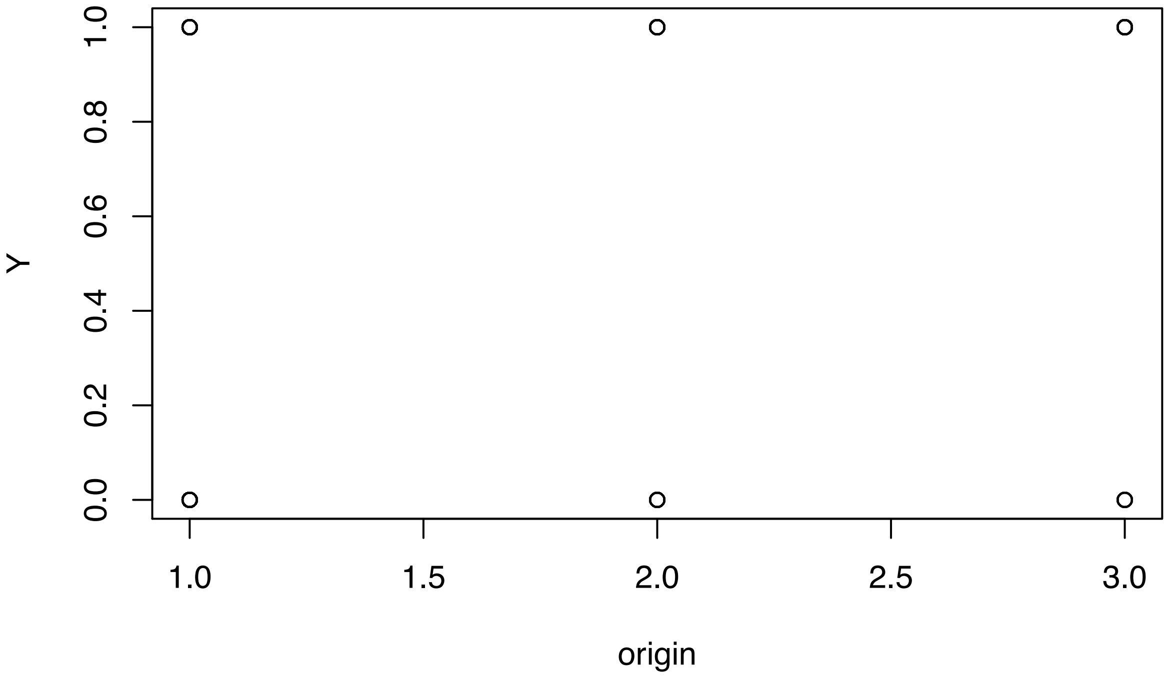

```
plot(acceleration, Y, xlab = "Acceleration", ylab = "Y")
```




```
plot(year, Y, xlab = "Year", ylab = "Y")
```




```
plot(origin, Y, xlab = "origin", ylab = "Y")
```



It seems that displacement, horsepower, weight, and accelaration are most relevant for predicting Y.

```
#(b)
set.seed(36401)
I = sample(1:392, size = 80, replace = FALSE)
test = Auto[I, ]
train = Auto[-I, ]

#(c)
#Logistic regression
out1 = glm(Y[-I] ~ cylinders+displacement+horsepower+weight+acceleration+year+origin, data = train, fami
summary(out1)
```

```
##
## Call:
## glm(formula = Y[-I] ~ cylinders + displacement + horsepower +
##     weight + acceleration + year + origin, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.88903  -0.10788   0.02024   0.19621   3.03997
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -17.682032   6.751443  -2.619 0.008819 **
## cylinders     -0.938843   0.502026  -1.870 0.061469 .
## displacement   0.018283   0.013923   1.313 0.189136
## horsepower    -0.026863   0.026140  -1.028 0.304107
## weight        -0.005022   0.001331  -3.772 0.000162 ***
## acceleration  -0.032373   0.173211  -0.187 0.851738
## year           0.467066   0.091913   5.082 3.74e-07 ***
```

```
## origin         0.791874   0.454048   1.744 0.081154 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 432.20  on 311  degrees of freedom
## Residual deviance: 117.59  on 304  degrees of freedom
## AIC: 133.59
##
## Number of Fisher Scoring iterations: 8
```

```
#Random forest
set.seed(36401)
library(randomForest)
Ytmp = factor(Y)
out2 = randomForest(Ytmp[-I] ~ cylinders+displacement+horsepower+weight+acceleration+year+origin, impor

#(d)
pred1 = predict(out1, newdata = test, type = "response")
n = length(Y[I])
yhat = rep(0, n)
yhat[pred1 >= 0.5] = 1
T = table(Y[I], yhat)
prediction_error = (T[1, 2] + T[2, 1])/sum(T)
print(1-prediction_error)
```

```
## [1] 0.8875
```

```
z = -qnorm(0.025)
phat = 1-prediction_error
low = phat - z*(sqrt(phat*(1-phat)/n))
print(low)
```

```
## [1] 0.818259
```

```
up = phat + z*(sqrt(phat*(1-phat)/n))
print(up)
```

```
## [1] 0.956741
```

```
pred2 = predict(out2, newdata = test)
T = table(Y[I], pred2)
prediction_error = (T[1, 2] + T[2, 1])/sum(T)
print(1-prediction_error)
```

```
## [1] 0.9125
```

```
z = -qnorm(0.025)
phat = 1-prediction_error
low = phat - z*(sqrt(phat*(1-phat)/n))
print(low)
```

```
## [1] 0.8505811
```

```r
up = phat + z*(sqrt(phat*(1-phat)/n))
print(up)
```

```
## [1] 0.9744189
```

The predictive accuracy of the logistic regression is 88.75% with a 95% CI of [81.8259%, 95.6741%]. The predictive accuracy of the random forest is 91.25% with a 95% CI of [85.05811%, 97.44189%]. It seems that the random forest works better.