Homework Assignment 4

1. For parts (a) through (c), indicate which of i. through iv. is correct. Justify your answer.

i. More flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
ii. More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.
iii. Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance.
iv. Less flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias.

(a) The lasso, relative to least squares, is:

Lasso compared to least squares: Lasso is just an extended version of least squares where an additional penalty is added which is also called L1 or absolute penalty and this penalty restricts the flexibility of the linear model. Therefore, it becomes "less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance."

(b) Repeat (a) for ridge regression relative to least squares.

Ridge compared to least squares: Ridge is just an extended version of least squares where an additional penalty is added which is also called L2 or squared penalty and this penalty restricts the flexibility of the linear model. Therefore, it becomes "less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance."

(c) Repeat (a) for non-linear methods relative to least squares.

Non-Linear models compared to least squares: Non-Linear models offer more flexibility in comparison to linear models and thus "More flexibility will give improved prediction accuracy when its increase in variance is less than its decrease in bias."

2. We will now try to predict per capita crime rate in the Boston data set.

(a) Try out some of the regression methods explored in this chapter, such as best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

```
library(ISLR)
library(glmnet)
library(pls)
library(leaps)
library(MASS)
set.seed(123)
```

```
sum(is.na(Boston))
```

```
0
```

```
names(Boston)
```

```
'crim' · 'zn' · 'indus' · 'chas' · 'nox' · 'rm' · 'age' · 'dis' · 'rad' · 'tax' · 'ptratio' · 'black' · 'lstat' · 'medv'
```

```
round((nrow(Boston)*0.2),0)
```

```
101
```

Split Train Test Boston Dataset

```
test_index=sample(nrow(Boston),round((nrow(Boston)*0.2),0),replace=FALSE)
train_set=Boston[-test_index,]
test_set=Boston[test_index,]
```

```
c(nrow(train_set),nrow(test_set),nrow(train_set)+nrow(test_set),nrow(Boston))
```

```
405 · 101 · 506 · 506
```

```
# best subset selection method
model_bss=regsubsets(medv~.,data=Boston,nvmax=13)
model_bss_summary=summary(model_bss)
```

```
names(model_bss_summary)
```

```
'which' · 'rsq' · 'rss' · 'adjr2' · 'cp' · 'bic' · 'outmat' · 'obj'
```

```
model_bss_summary$rss
```

```
19472.3814183264 · 15439.3092013135 · 13727.9853137995 · 13228.9077026118 · 12469.3441508071 · 12141.0727358978 · 11868.2356073211 · 11678.2994702239 · 11526.1224460365 · 11308.5776061855 · 11081.3639524346 · 11078.8464123084 · 11078.784577955
```

```
min(model_bss_summary$rss)
```

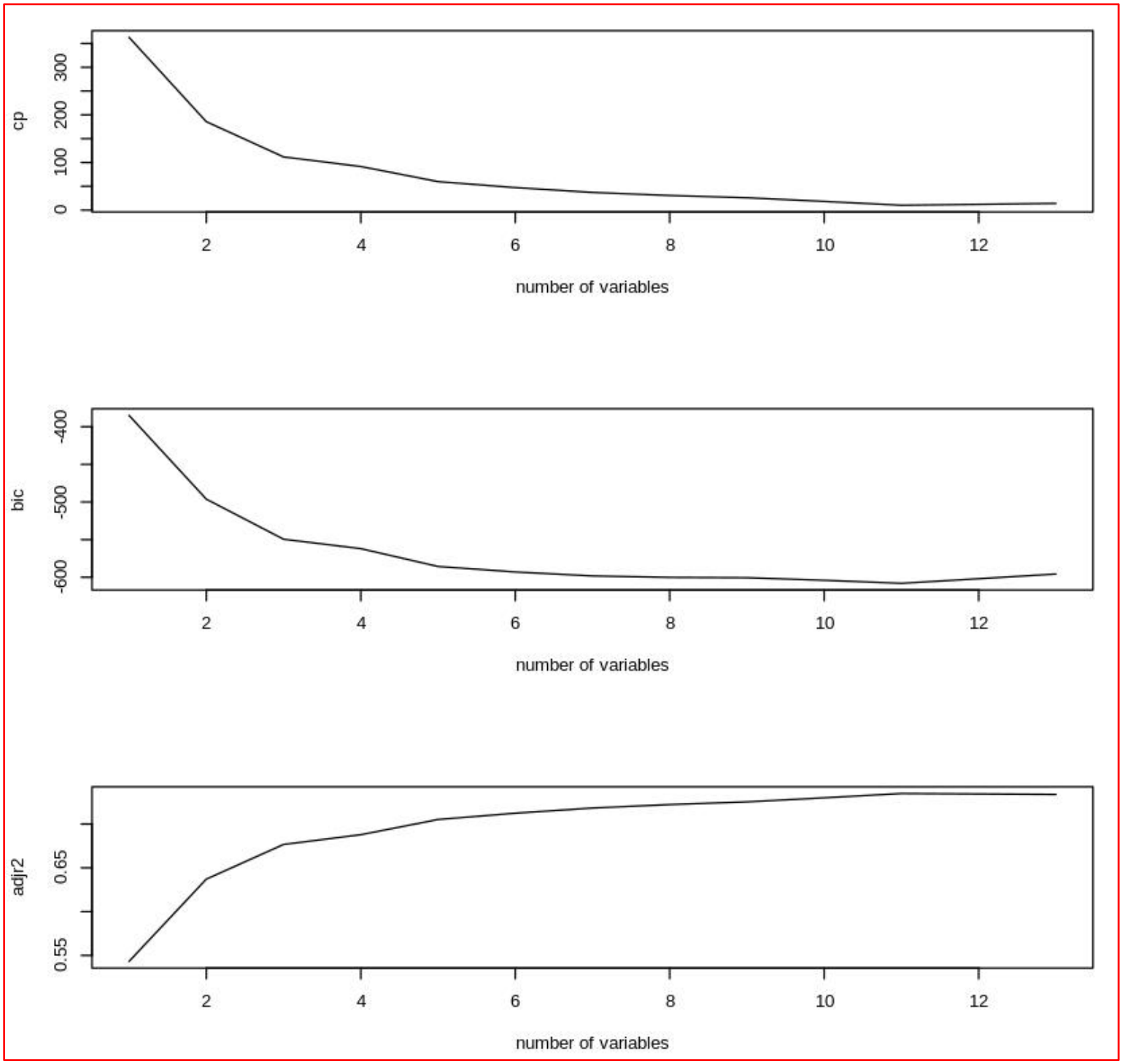
```
11078.784577955
```

```
bss_model_pred=mean(model_bss_summary$rss)
```

```
1  model_bss_summary
```

```
Subset selection object
Call: regsubsets.formula(medv ~ ., data = Boston, nvmax = 13)
13 Variables  (and intercept)
        Forced in Forced out
crim        FALSE      FALSE
zn          FALSE      FALSE
indus       FALSE      FALSE
chas        FALSE      FALSE
nox         FALSE      FALSE
rm          FALSE      FALSE
age         FALSE      FALSE
dis         FALSE      FALSE
rad         FALSE      FALSE
tax         FALSE      FALSE
ptratio     FALSE      FALSE
black       FALSE      FALSE
lstat       FALSE      FALSE
1 subsets of each size up to 13
Selection Algorithm: exhaustive
          crim zn  indus chas nox rm  age dis rad tax ptratio black lstat
1  ( 1 )  " "  " " " "   " "  " " " " " " " " " " " "     " "   "*"
2  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " " "     " "   "*"
3  ( 1 )  " "  " " " "   " "  " " "*" " " " " " " " "*"     " "   "*"
4  ( 1 )  " "  " " " "   " "  " " "*" " " "*" " " " " "*"     " "   "*"
5  ( 1 )  " "  " " " "   " "  "*" "*" " " "*" " " " " "*"     " "   "*"
6  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" " " " " "*"     " "   "*"
7  ( 1 )  " "  " " " "   "*"  "*" "*" " " "*" " " " " "*"     "*"   "*"
8  ( 1 )  " "  "*" " "   "*"  "*" "*" " " "*" " " " " "*"     "*"   "*"
9  ( 1 )  "*"  " " " "   "*"  "*" "*" " " "*" "*" " " "*"     "*"   "*"
10  ( 1 ) "*"  "*" " "   " "  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
11  ( 1 ) "*"  "*" " "   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
12  ( 1 ) "*"  "*" "*"   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
13  ( 1 ) "*"  "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"
```

```
par(mfrow=c(3,1))
plot(model_bss_summary$cp,xlab='number of variables',ylab='cp',type='l')
plot(model_bss_summary$bic,xlab='number of variables',ylab='bic',type='l')
plot(model_bss_summary$adjr2,xlab='number of variables',ylab='adjr2',type='l')
```


cp
0
100
200
300
2
4
6
8
10
12
number of variables
bic
-400
-500
-600
2
4
6
8
10
12
number of variables
adjr2
0.55
0.65
2
4
6
8
10
12
number of variables

```r
indices=c(which.min(model_bss_summary$cp),which.min(model_bss_summary$bic),which.max(model_bss_summary$adjr2))

for (index in indices) {
  cat("Number of variables:",index,"\n")
  cat("Coefficients:\n")
  print(coef(model_bss,id=index))
  cat("\n")
}
```

```
Number of variables: 11
Coefficients:
  (Intercept)          crim            zn          chas           nox
 36.341145004  -0.108413345   0.045844929   2.718716303 -17.376023429
           rm           dis           rad           tax       ptratio
  3.801578840  -1.492711460   0.299608454  -0.011777973  -0.946524570
        black         lstat
  0.009290845  -0.522553457

Number of variables: 11
Coefficients:
  (Intercept)          crim            zn          chas           nox
 36.341145004  -0.108413345   0.045844929   2.718716303 -17.376023429
           rm           dis           rad           tax       ptratio
  3.801578840  -1.492711460   0.299608454  -0.011777973  -0.946524570
        black         lstat
  0.009290845  -0.522553457

Number of variables: 11
Coefficients:
  (Intercept)          crim            zn          chas           nox
 36.341145004  -0.108413345   0.045844929   2.718716303 -17.376023429
           rm           dis           rad           tax       ptratio
  3.801578840  -1.492711460   0.299608454  -0.011777973  -0.946524570
        black         lstat
  0.009290845  -0.522553457
```

```r
# best_subset_predictors=
names(coef(model_bss,id=best_subset_model))
```

'(Intercept)' · 'crim' · 'zn' · 'chas' · 'nox' · 'rm' · 'dis' · 'rad' · 'tax' · 'ptratio' · 'black' · 'lstat'

```r
best_subset_model_pred=predict(lm(medv~crim+zn+chas+nox+rm+dis+rad+tax+ptratio+black+lstat,data=train_set),test_set)
```

LS

```r
lm_model=lm(medv~.,data=train_set)
lm_model_pred=predict(lm_model,test_set)
```

```r
train_xmat=model.matrix(medv~.,data=train_set)
test_xmat=model.matrix(medv~.,data=test_set)
```

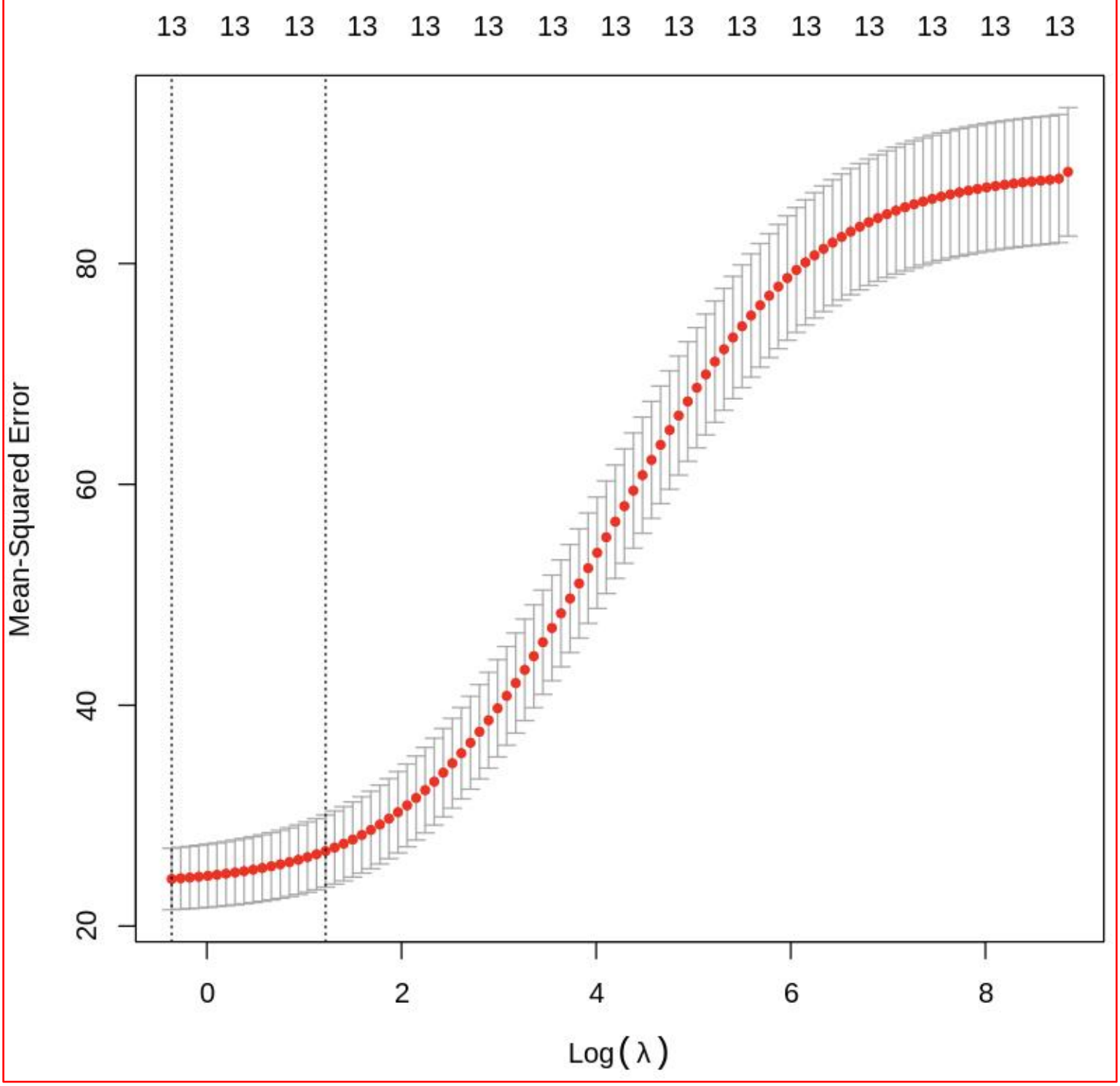
RIDGE

```r
# Ridge
model_ridge=cv.glmnet(train_xmat,train_set[,'medv'],alpha=0)
model_ridge_best_lambda=model_ridge$lambda.min
ridge_model_pred=predict(model_ridge,test_xmat,s=model_ridge_best_lambda)
```

```r
print(model_ridge_best_lambda)
```

```
[1] 0.6964206
```

```r
plot(model_ridge)
```



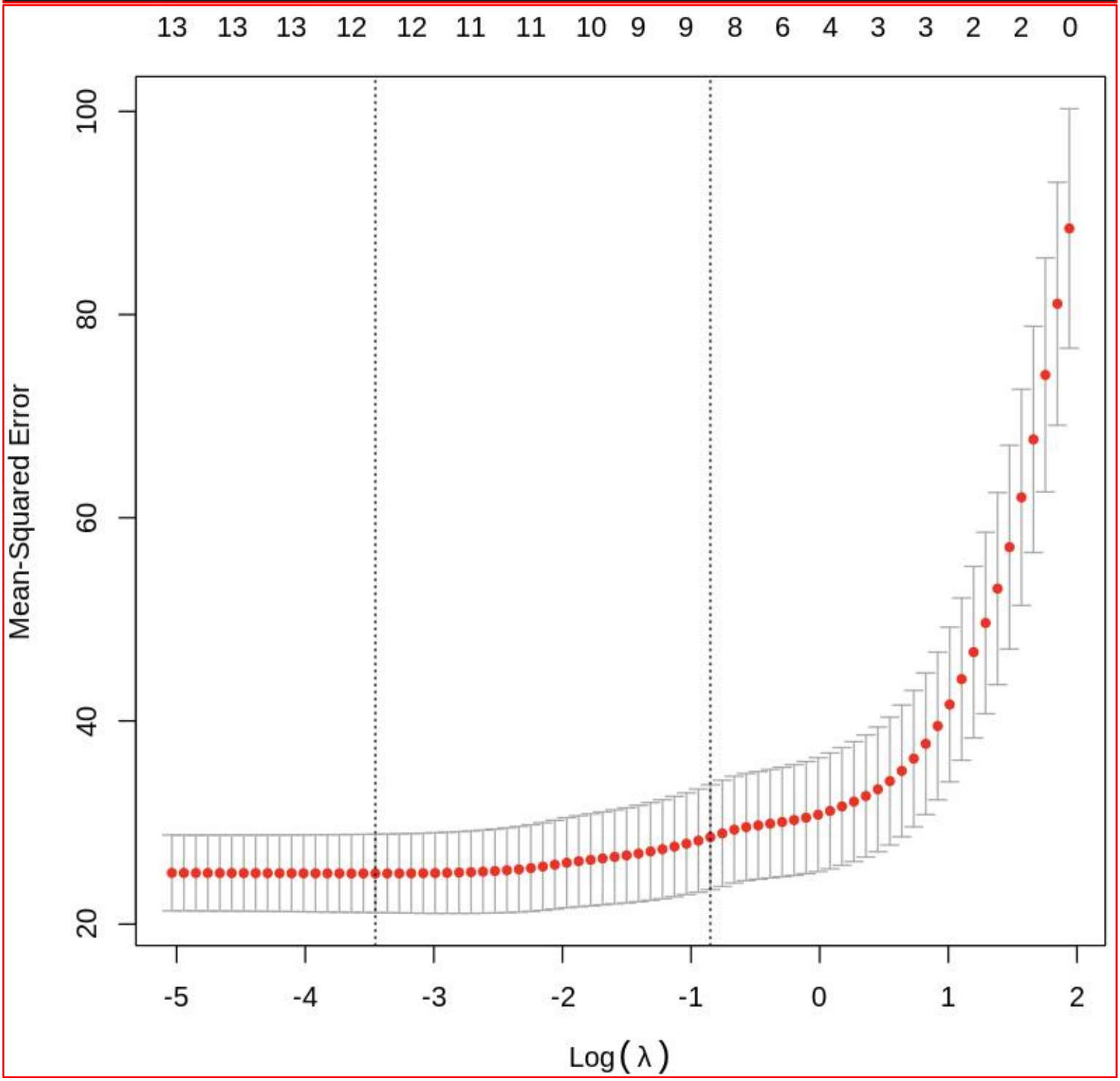
LASSO

```
1  # Lasso
2  model_lasso=cv.glmnet(train_xmat,train_set[,'medv'],alpha=1)
3  model_lasso_best_lambda=model_lasso$lambda.min
4  lasso_model_pred=predict(model_lasso,test_xmat,s=model_lasso_best_lambda)
```

```
1  print(model_lasso_best_lambda)
```

```
[1] 0.03158183
```

```
1  plot(model_lasso)
```



```
1  predict(model_lasso,test_xmat,s=model_lasso_best_lambda,type='coefficients')
```

```
15 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept)  35.832397215
(Intercept)   .
crim         -0.116087086
zn            0.047239333
indus         .
chas          2.116369879
nox         -17.488486754
rm            3.903641091
age           0.001890364
dis          -1.487045601
rad           0.253142520
tax          -0.010144150
ptratio      -0.962472578
black         0.009242004
lstat        -0.521470115
```

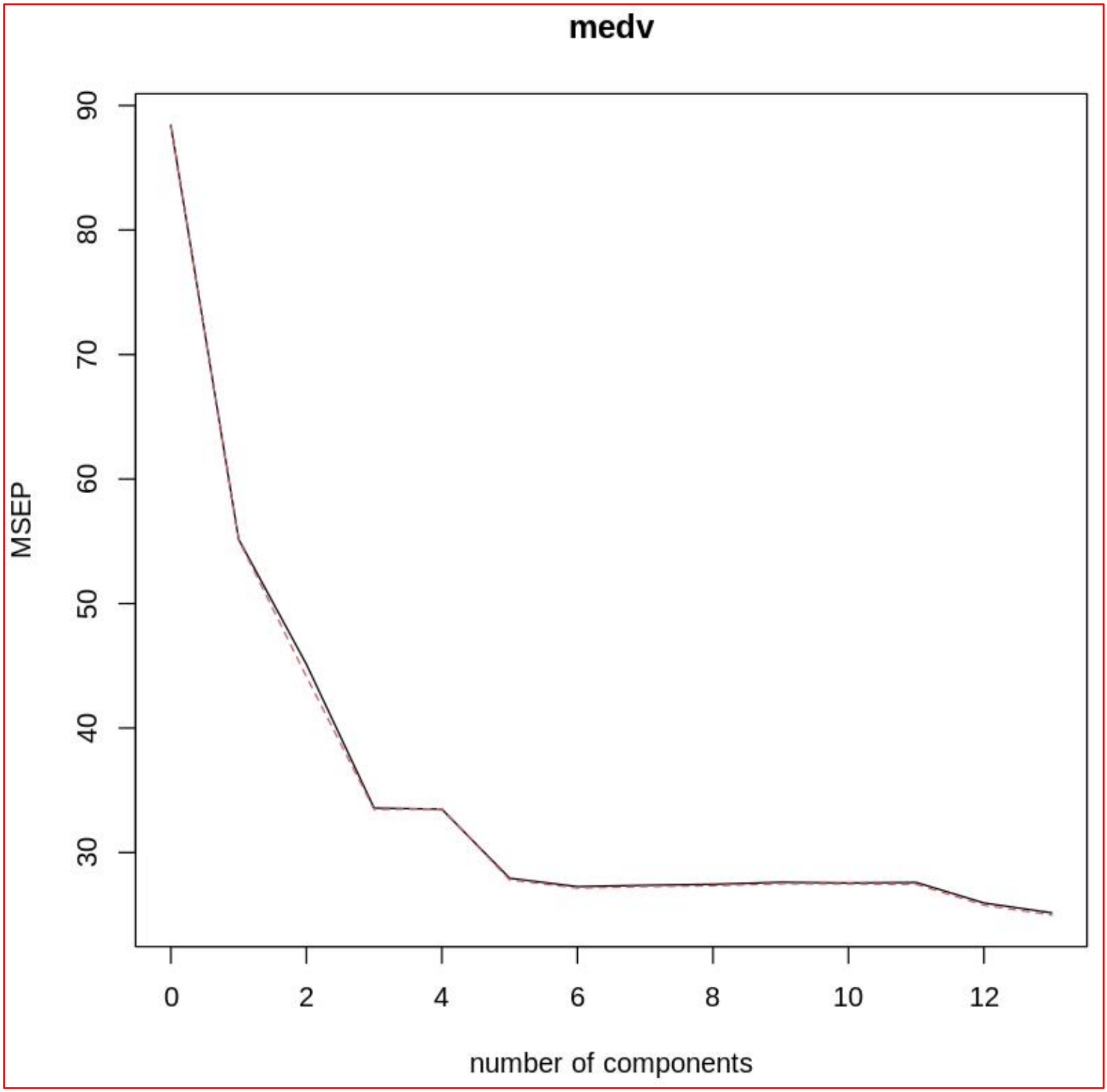
PCR

```
1  model_pcr=pcr(medv~.,data=train_set,scale=TRUE,validation="CV")
```

```
summary(model_pcr)
```

```
Data:   X dimension: 405 13
        Y dimension: 405 1
Fit method: svdpc
Number of components considered: 13

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           9.403    7.428    6.718    5.794    5.786    5.286    5.222
adjCV        9.403    7.425    6.644    5.785    5.789    5.274    5.211
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       5.232    5.239    5.255     5.249     5.254     5.094     5.016
adjCV    5.222    5.228    5.244     5.243     5.239     5.077     4.998

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X       46.55    57.48    67.47    74.33    80.54    85.84    89.98    93.03
medv    38.03    52.08    63.74    64.19    70.29    70.93    71.07    71.36
      9 comps  10 comps  11 comps  12 comps  13 comps
X       95.18     96.78     98.28     99.55    100.00
medv    71.36     71.58     72.13     74.14     75.08
```

```
validationplot(model_pcr,val.type="MSEP")
```


medv
MSEP
90
80
70
60
50
40
30
0
2
4
6
8
10
12
number of components

```
model_pcr_pred=predict(model_pcr,test_set,ncomp=6)
```

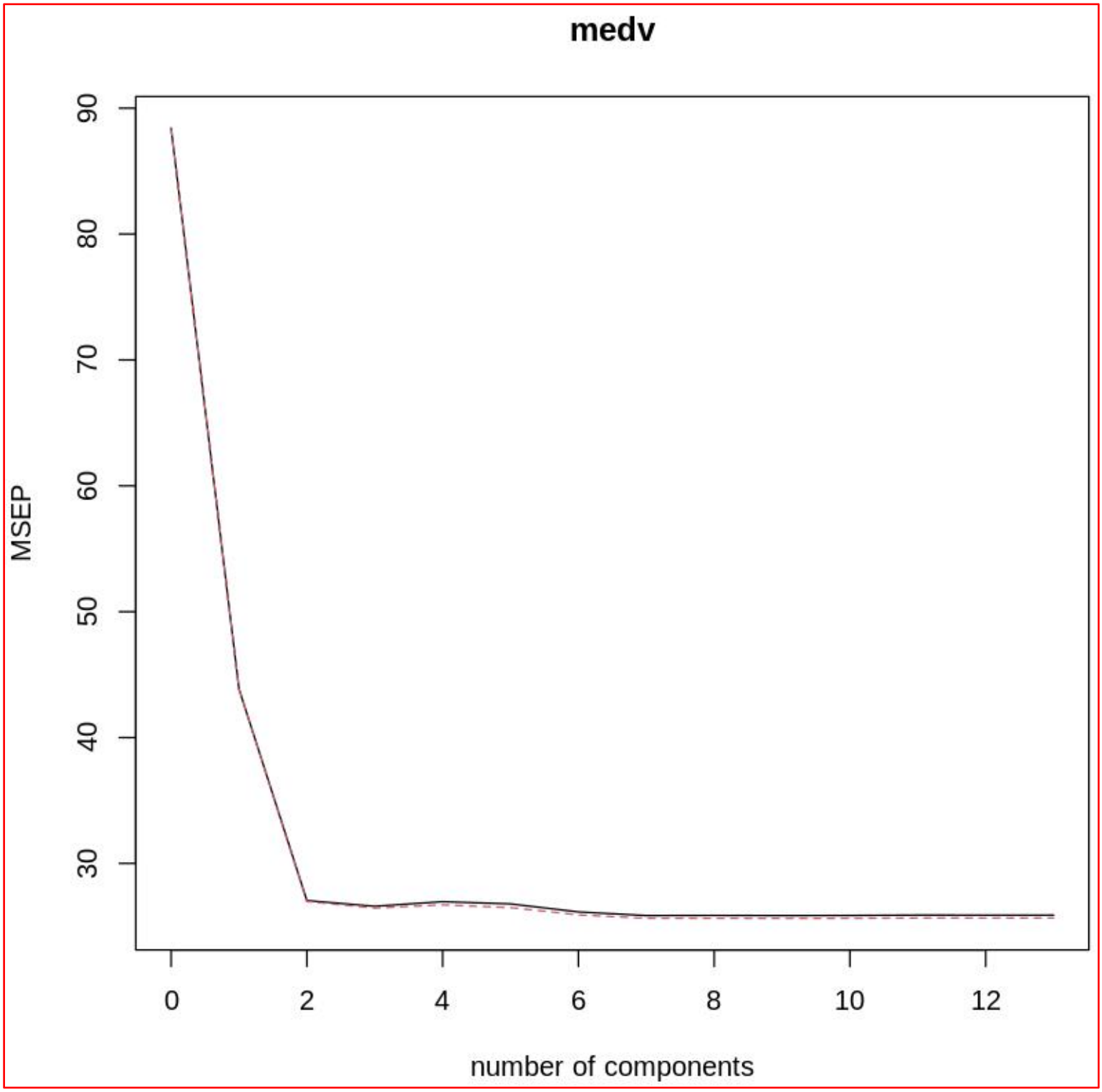
PLR

```
model_plsr=plsr(medv~.,data=train_set,scale=TRUE,validation="CV")
```

```
summary(model_plsr)
```

```
Data:   X dimension: 405 13
        Y dimension: 405 1
Fit method: kernelpls
Number of components considered: 13

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
CV           9.403    6.622    5.201    5.157    5.193    5.175    5.113
adjCV        9.403    6.619    5.193    5.143    5.168    5.147    5.090
       7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
CV       5.085    5.086    5.085     5.086     5.089     5.088     5.088
adjCV    5.064    5.064    5.063     5.064     5.066     5.066     5.066

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X       45.22    56.57    63.08    68.23    74.88    78.72    82.21    85.64
medv    51.09    71.28    73.22    74.31    74.76    74.90    75.00    75.07
      9 comps  10 comps  11 comps  12 comps  13 comps
X       90.27     93.47     96.30     98.01    100.00
medv    75.08     75.08     75.08     75.08     75.08
```

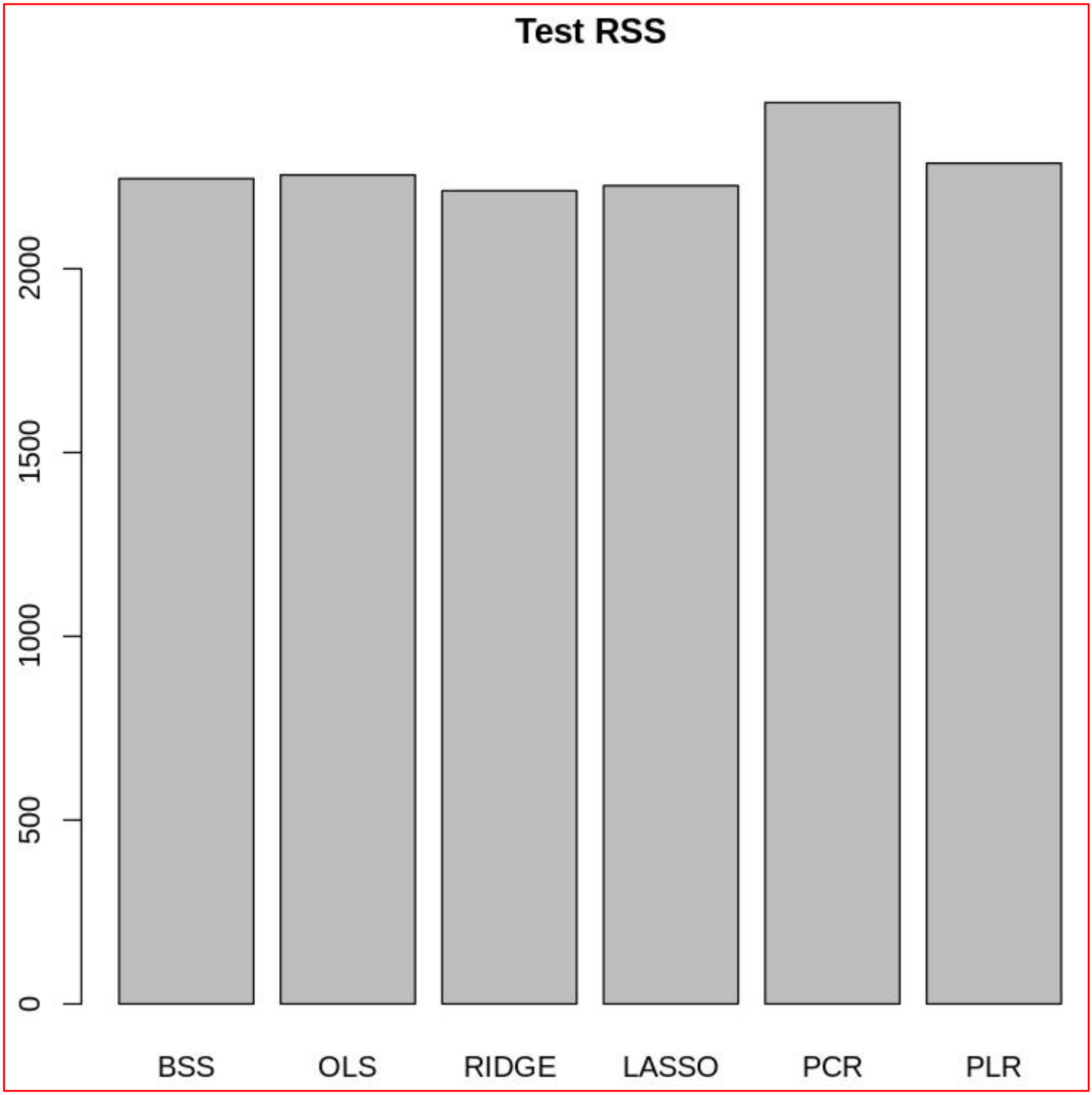
```
validationplot(model_plsr,val.type="MSEP")
```


medv
MSEP
90
80
70
60
50
40
30
0
2
4
6
8
10
12
number of components

```
model_plsr_pred=predict(model_plsr,test_set,ncomp=7)
```

```
# RSS FOR BEST SUBSET
print(round((mean(sum((test_set[,"medv"]-best_subset_model_pred)^2))),0))

# RSS FOR LM
print(round((mean(sum((test_set[,"medv"]-lm_model_pred)^2))),0))

# RSS FOR RIDGE
print(round((mean(sum((test_set[,"medv"]-ridge_model_pred)^2))),0))

# RSS FOR LASSO
print(round((mean(sum((test_set[,"medv"]-lasso_model_pred)^2))),0))

# RSS FOR PCR
print(round((mean(sum((test_set[,"medv"]-model_pcr_pred)^2))),0))

# RSS FOR PLR
print(round((mean(sum((test_set[,"medv"]-model_plsr_pred)^2))),0))
```

```
[1] 2245
[1] 2255
[1] 2212
[1] 2226
[1] 2452
[1] 2287
```



We divided your dataset into a training set with 405 records and a testing set with 101 records. This allows us to train our models on one set and evaluate their performance on another, ensuring a fair assessment. We have used several regression techniques, including best subset selection, least squares, ridge regression, lasso regression, Principal Components Regression (PCR), and Partial Least Squares regression. To identify the best subset of features for prediction, we employed the best subset selection method. This method evaluates different combinations of predictor variables to determine the most effective set for the model. By examining criteria such as the ***Cp*** score, ***BIC*** score, and ***Adjusted R-squared*** score, we identified a model with 11 features that yielded the most promising results. These features include ***'crim', 'zn', 'chas', 'nox', 'rm', 'dis', 'rad', 'tax', 'ptratio', 'black', and 'lstat'***. Then using features, we fitted the regression model using the chosen subset of 11 variables and saved it as the best subset model. Additionally, we applied least squares, ridge regression, lasso regression, PCR, and Partial Least Squares regression methods to the dataset using all features. Finally, we calculated the Residual Sum of Squares (RSS) for each model and represented the RSS results using a bar plot, allowing for easy comparison between the different regression techniques.

(b) Propose a model (or set of models) that seem to perform well on this data set, and justify your answer. Make sure that you are evaluating model performance using validation set error, cross-validation, or some other reasonable alternative, as opposed to using training error.

To determine which models perform well, we calculated the Residual Sum of Squares (RSS) for each method. The RSS is a measure of how well the model fits the data, with lower values indicating better performance.

Based on our analysis, the models performed as follows in terms of RSS:

- ✓ Ridge Regression RSS = 2212
- ✓ Lasso Regression RSS = 2226
- ✓ Best Subset Selection (BSS) RSS = 2245
- ✓ Principal Components Regression (PCR) RSS = 2452
- ✓ Least Squares Regression (LM) RSS = 2255
- ✓ Partial Least Squares Regression (PLR) RSS = 2287

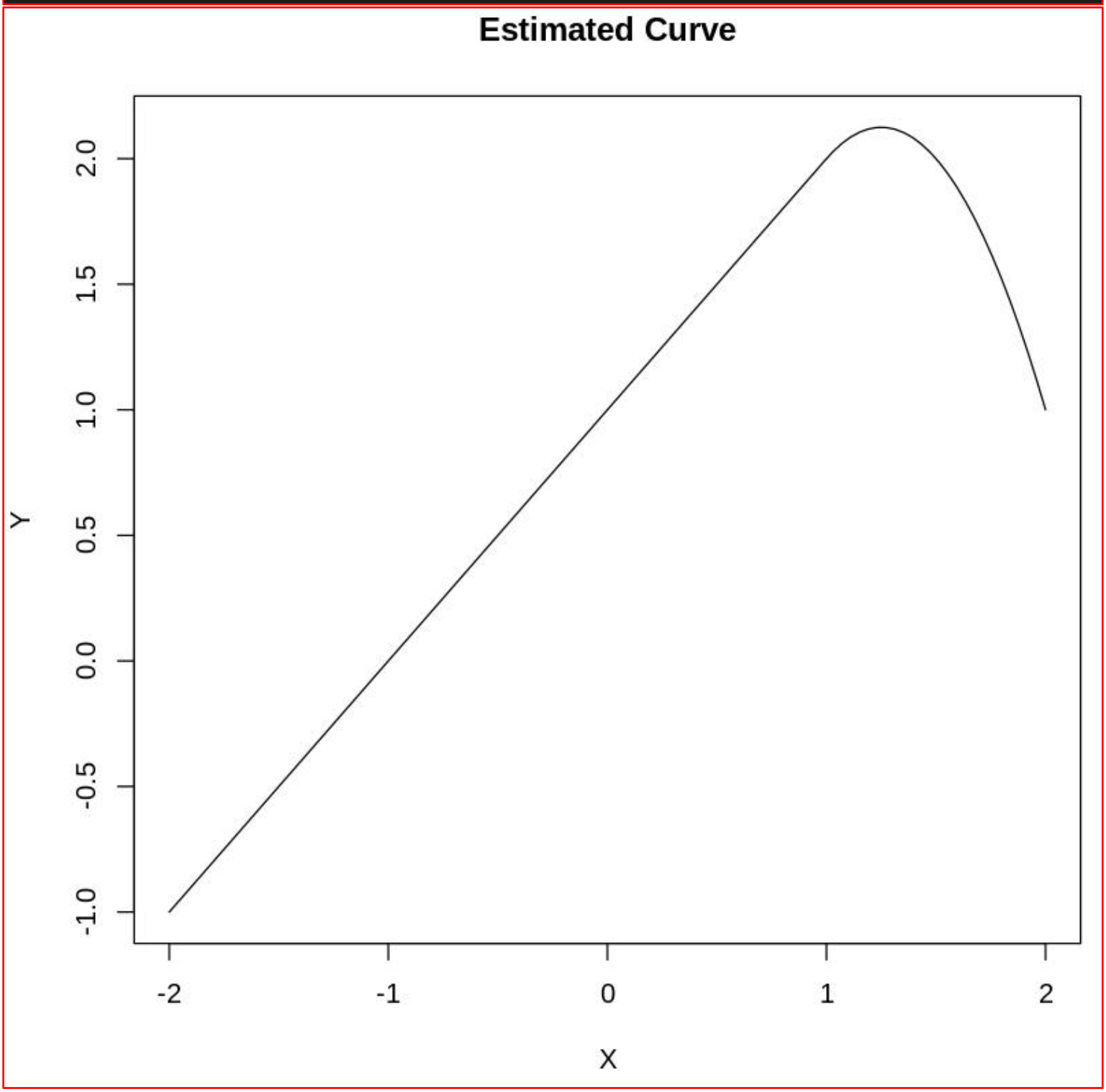
Among these, Ridge Regression produced the lowest RSS at 2212, suggesting its superior fit compared to other methods. However, Lasso Regression followed closely with an RSS of 2226, implying comparable performance. Meanwhile, BSS, LM, and PLR displayed higher RSS values, indicating relatively poorer fits, with PCR exhibiting the highest RSS at 2452, suggesting the least favorable performance among the methods assessed.

(c) Does your chosen model involve all of the features in the data set? Why or why not?

The Ridge model demonstrated the lowest test Residual Sum of Squares (RSS) score among the models evaluated. By selecting the Ridge model, we are opting for a regularization technique that penalizes the magnitude of the coefficients in the regression model. This helps prevent overfitting and improves the model's ability to generalize to unseen data. The Ridge model typically involves regularization, which shrinks the coefficients of less important features towards zero. This means that not all features may have significant coefficients in the final model. Instead, the Ridge model selects and emphasizes the most relevant features while mitigating the influence of less informative variables. The chosen model includes all the features.

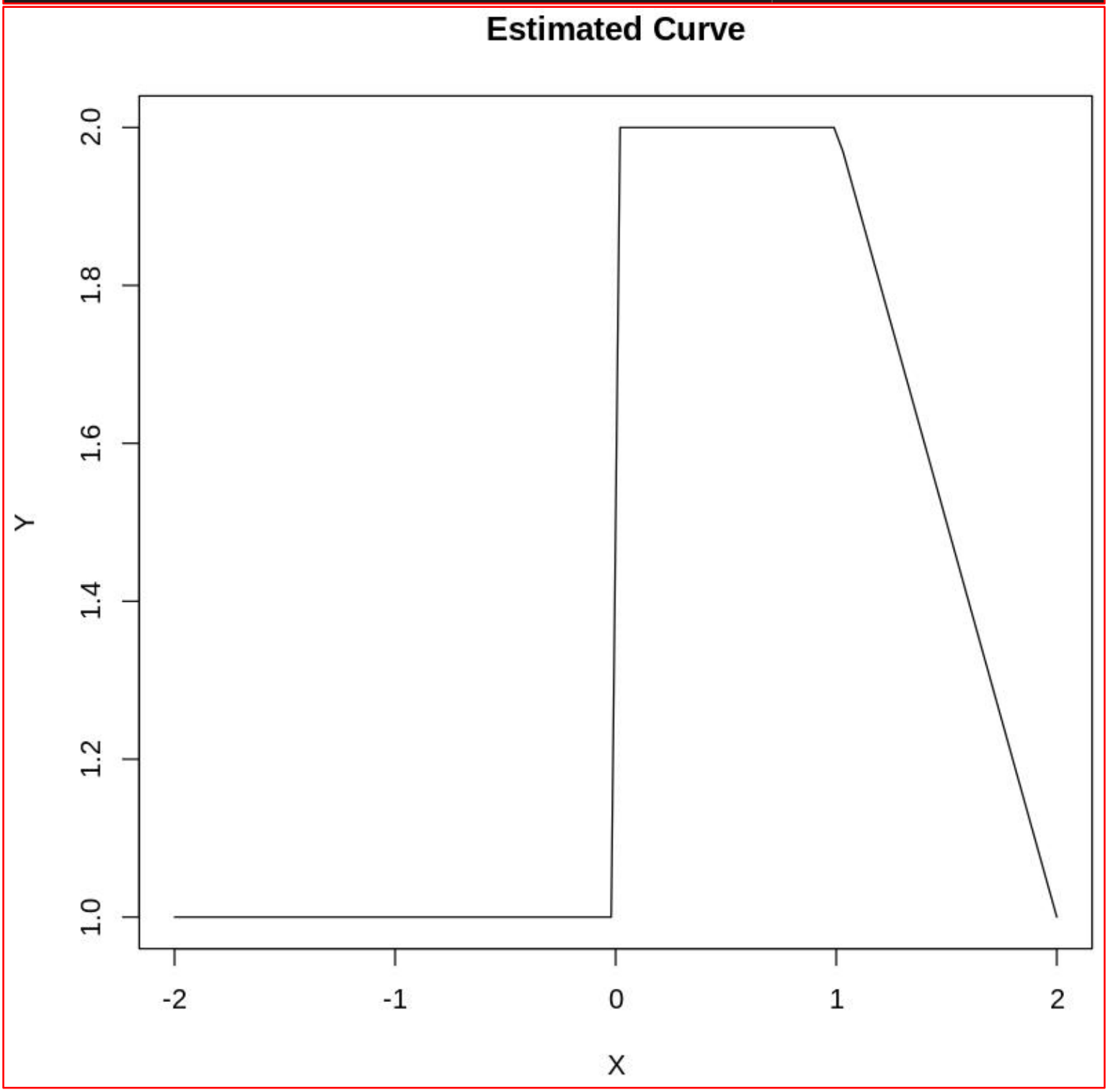
3. Suppose we fit a curve with basis functions $b_1(X) = X$, $b_2(X) = (X - 1)^2 I(X \geq 1)$. (Note that $I(X \geq 1)$ equals 1 for $X \geq 1$ and 0 otherwise.) We fit the linear regression model $Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \varepsilon$, and obtain coefficient estimates $\hat{\beta}_0 = 1$, $\hat{\beta}_1 = 1$, $\hat{\beta}_2 = -2$. Sketch the estimated curve between X = -2 and X = 2. Note the intercepts, slopes, and other relevant information.

```
x_feature=seq(from=-2,to=2,length.out=100)
beta0=1
beta1=(1 * x_feature)
beta2=(-2 * ((x_feature-1)^2 * I(x_feature>1)))
y_target=beta0+beta1+beta2
plot(x_feature,y_target,type="l",xlab="X",ylab="Y",main="Estimated Curve")
```



4. Suppose we fit a curve with basis functions $b_1(X) = I(0 \leq X \leq 2) - (X - 1)\, I(1 \leq X \leq 2)$, $b_2(X) = (X - 3)\, I(3 \leq X \leq 4) + I(4 < X \leq 5)$. We fit the linear regression model $Y = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \varepsilon$, and obtain coefficient estimates $\hat{\beta}_0 = 1$, $\hat{\beta}_1 = 1$, $\hat{\beta}_2 = 3$. Sketch the estimated curve between X = -2 and X = 2. Note the intercepts, slopes, and other relevant information.

```
x_feature=seq(from=-2,to=2,length.out=100)
beta0=1
beta1=(1 * ((I(0 <= x_feature) & I(x_feature <= 2)) - ((x_feature - 1) * (I(1 <= x_feature) & I(x_feature <= 2)))))
beta2=(3 * (((x_feature - 3) * (I(3 <= x_feature) & I(x_feature <= 4))) + (I(4 < x_feature) & I(x_feature <= 5))))
y_target=beta0+beta1+beta2
plot(x_feature,y_target,type="l",xlab="X",ylab="Y",main="Estimated Curve")
```



5. In this exercise, you will further analyze the Wage data set considered throughout this chapter.

(a) Perform polynomial regression to predict wage using age. Use cross-validation to select the optimal degree d for the polynomial. What degree was chosen, and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting polynomial fit to the data.

```
1  library(ISLR)
2  library(boot)
3  set.seed(123)
```

```
1  sum(is.na(Wage))
```

```
0
```

```
1  names(Wage)
```

```
'year' · 'age' · 'maritl' · 'race' · 'education' · 'region' · 'jobclass' · 'health' · 'health_ins' · 'logwage' · 'wage'
```

```
1  cv_error_10_fold=rep(0,10)
2
3  for(i in 1:10){
4    poly_fit=glm(wage~poly(age,i),data=Wage)
5    cv_error_10_fold[i]=cv.glm(Wage,poly_fit,K=10)$delta[1]
6  }
```

```r
# Optimal Degree
print(which.min(cv_error_10_fold))
```

```
[1] 10
```

```r
cv_error_10_fold
```

```
1676.98840217094 · 1602.47265509628 · 1597.03646644026 · 1594.58213961531 · 1594.42423224147 · 1594.15741237273 · 1596.53155300822 · 1593.3514562093 · 1593.28531739186 · 1593.15364551316
```

```r
# Checking the differnce between the nearest values
cv_error_10_fold[4] - cv_error_10_fold[10]
cv_error_10_fold[5] - cv_error_10_fold[10]
cv_error_10_fold[6] - cv_error_10_fold[10]
```

```
1.42849410215217
1.27058672831186
1.00376685957303
```

```r
# Compare with ANOVA
anova_results=anova(lm(wage ~ poly(age,10),data=Wage))
cat("Degree chosen by ANOVA:",which.min(anova_results$"Pr(>F)"),"\n")
```

```
Degree chosen by ANOVA: 1
```

```r
fit_1=lm(wage~poly(age,1),data=Wage)
fit_2=lm(wage~poly(age,2),data=Wage)
fit_3=lm(wage~poly(age,3),data=Wage)
fit_4=lm(wage~poly(age,4),data=Wage)
fit_5=lm(wage~poly(age,5),data=Wage)
fit_6=lm(wage~poly(age,6),data=Wage)
fit_7=lm(wage~poly(age,7),data=Wage)
fit_8=lm(wage~poly(age,8),data=Wage)
fit_9=lm(wage~poly(age,9),data=Wage)
fit_10=lm(wage~poly(age,10),data=Wage)
anova_results=anova(fit_1,fit_2,fit_3,fit_4,fit_5,fit_6,fit_7,fit_8,fit_9,fit_10)
```

```
1  anova_results
```

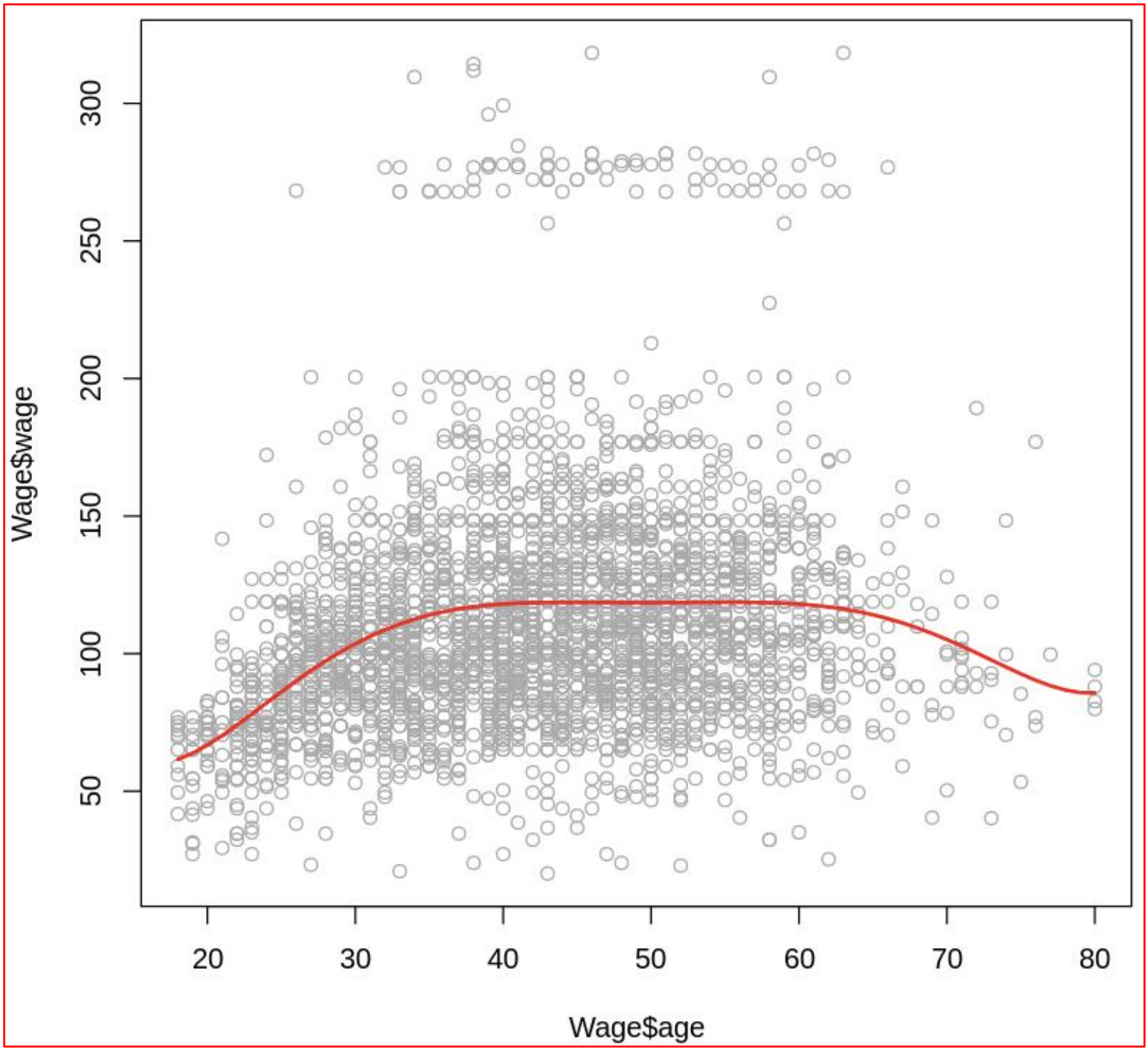
A anova: 10 × 6

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	2998	5022216	NA	NA	NA	NA
2	2997	4793430	1	2.287860e+05	1.437638e+02	2.187330e-32
3	2996	4777674	1	1.575569e+04	9.900512e+00	1.668583e-03
4	2995	4771604	1	6.070152e+03	3.814342e+00	5.090874e-02
5	2994	4770322	1	1.282563e+03	8.059328e-01	3.693978e-01
6	2993	4766389	1	3.932258e+03	2.470939e+00	1.160744e-01
7	2992	4763834	1	2.555281e+03	1.605679e+00	2.051989e-01
8	2991	4763707	1	1.266690e+02	7.959584e-02	7.778654e-01
9	2990	4756703	1	7.004317e+03	4.401350e+00	3.599425e-02
10	2989	4756701	1	2.637537e+00	1.657367e-03	9.675292e-01

```
1  anova_results$"Pr(>F)"
```

<NA> · 2.18733010887078e-32 · 0.00166858259572064 · 0.0509087395644624 · 0.36939776998176 · 0.11607444505432 · 0.205198937138797 · 0.777865390323261 · 0.0359942486411385 · 0.967529190782386

```
1  age_lim=range(Wage$age)
2  age_grid=seq(from=age_lim[1],to=age_lim[2])
3  age_preds=predict(fit_6,newdata=list(age=age_grid),se=TRUE)
4  plot(Wage$age,Wage$wage,col="darkgrey")
5  lines(age_grid,age_preds$fit,lwd=2,col='red',xlim=age_lim)
```


Wage$wage
300
250
200
150
100
50
20
30
40
50
60
70
80
Wage$age

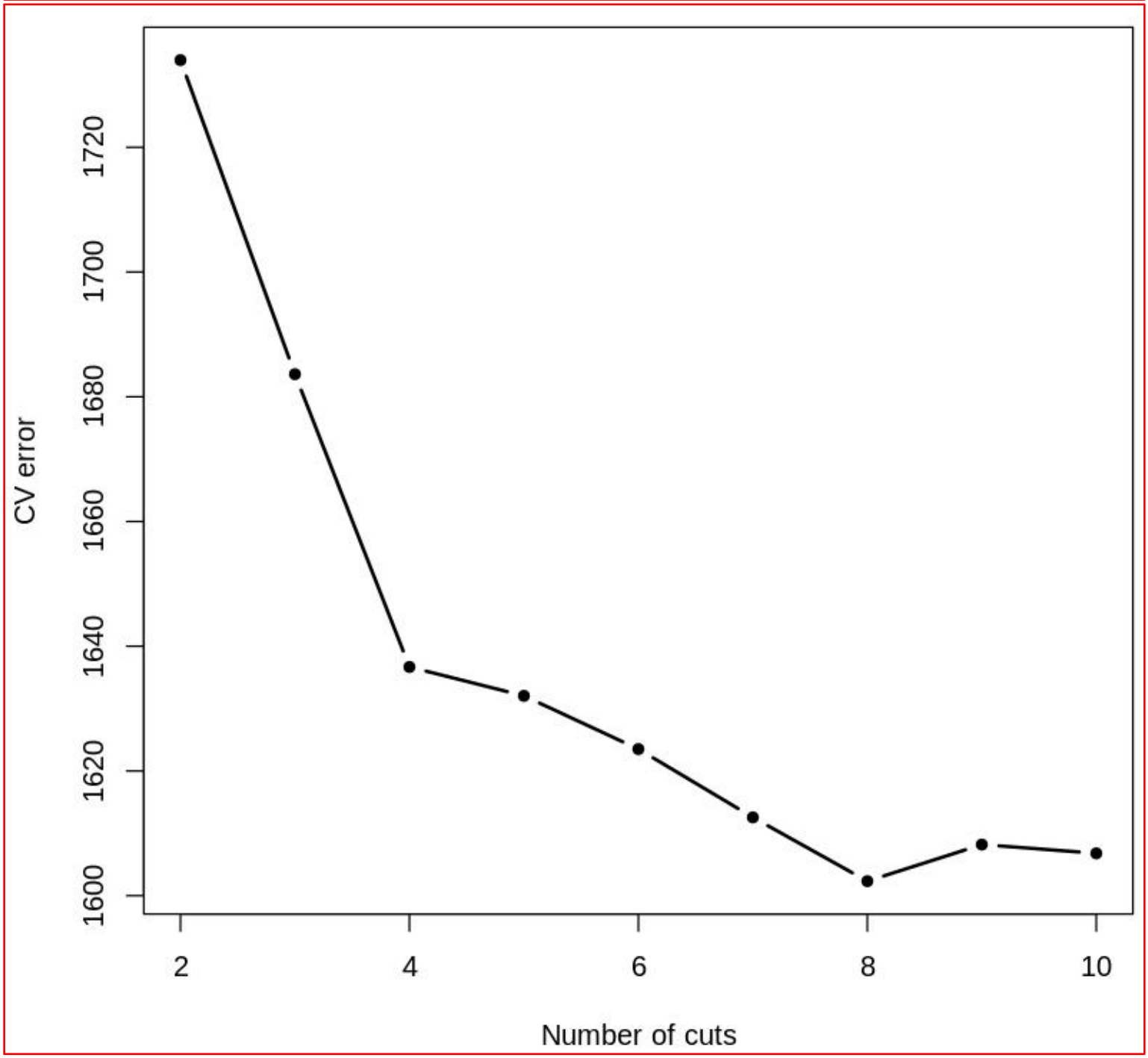
(b) Fit a step function to predict wage using age, and perform cross-validation to choose the optimal number of cuts. Make a plot of the fit obtained.

```
cv_error_10_fold_step=rep(NA,10)
for(i in 2:10){
  Wage$new_age_cut=cut(Wage$age,i)
 step_fit=glm(wage~new_age_cut,data=Wage)
 cv_error_10_fold_step[i]=cv.glm(Wage,step_fit,K=10)$delta[1]
}
```

```
which.min(cv_error_10_fold_step)
```

```
8
```

```
plot(2:10,cv_error_10_fold_step[-1],type='b',xlab="Number of cuts",ylab="CV error",pch=20,lwd=2)
```

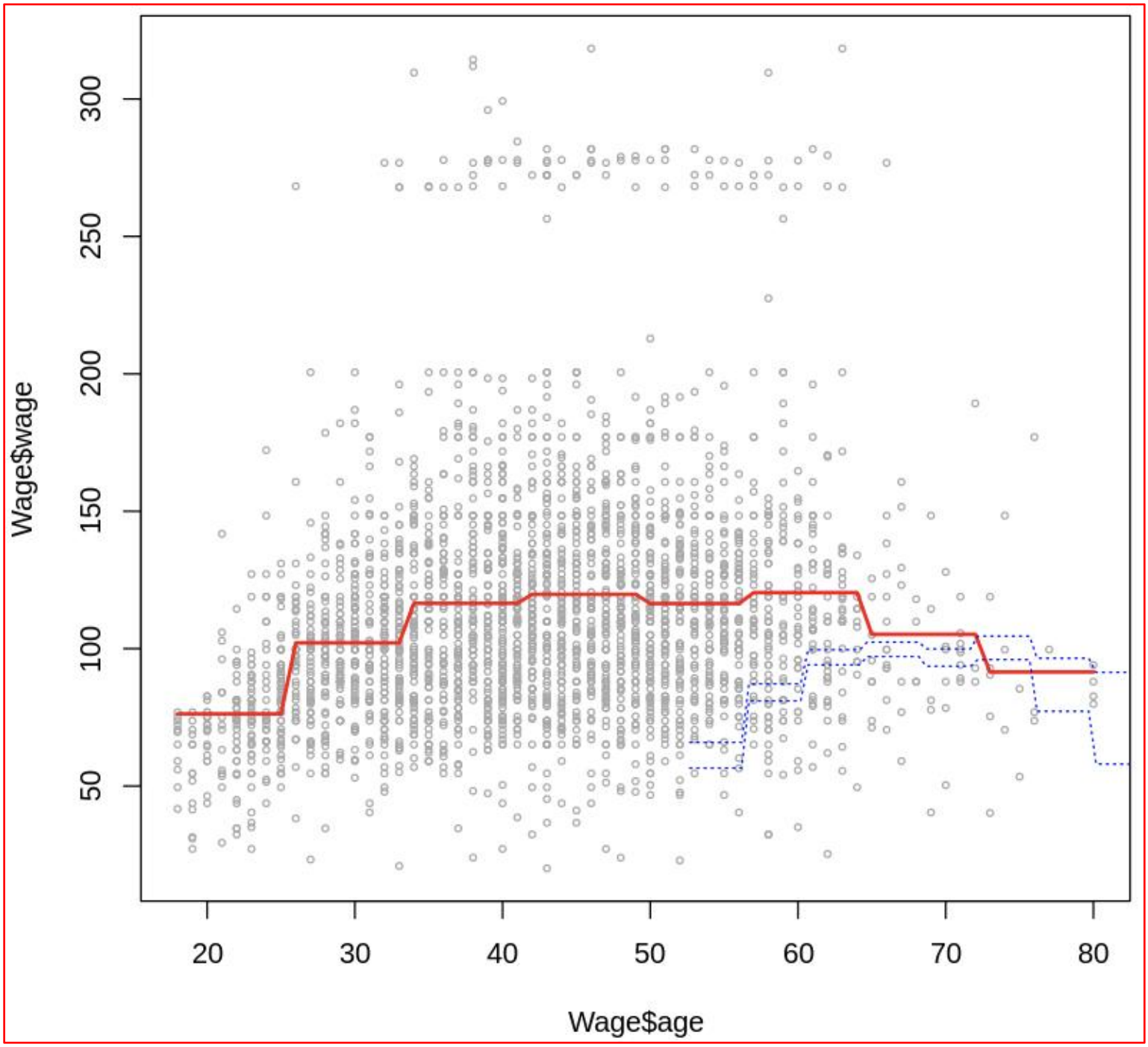


```r
age_lim=range(Wage$age)
age_grid=seq(from=age_lim[1],to=age_lim[2])
step_fit=glm(wage~cut(age,8),data=Wage)
age_preds=predict(step_fit,newdata=list(age=age_grid),se=TRUE)
```

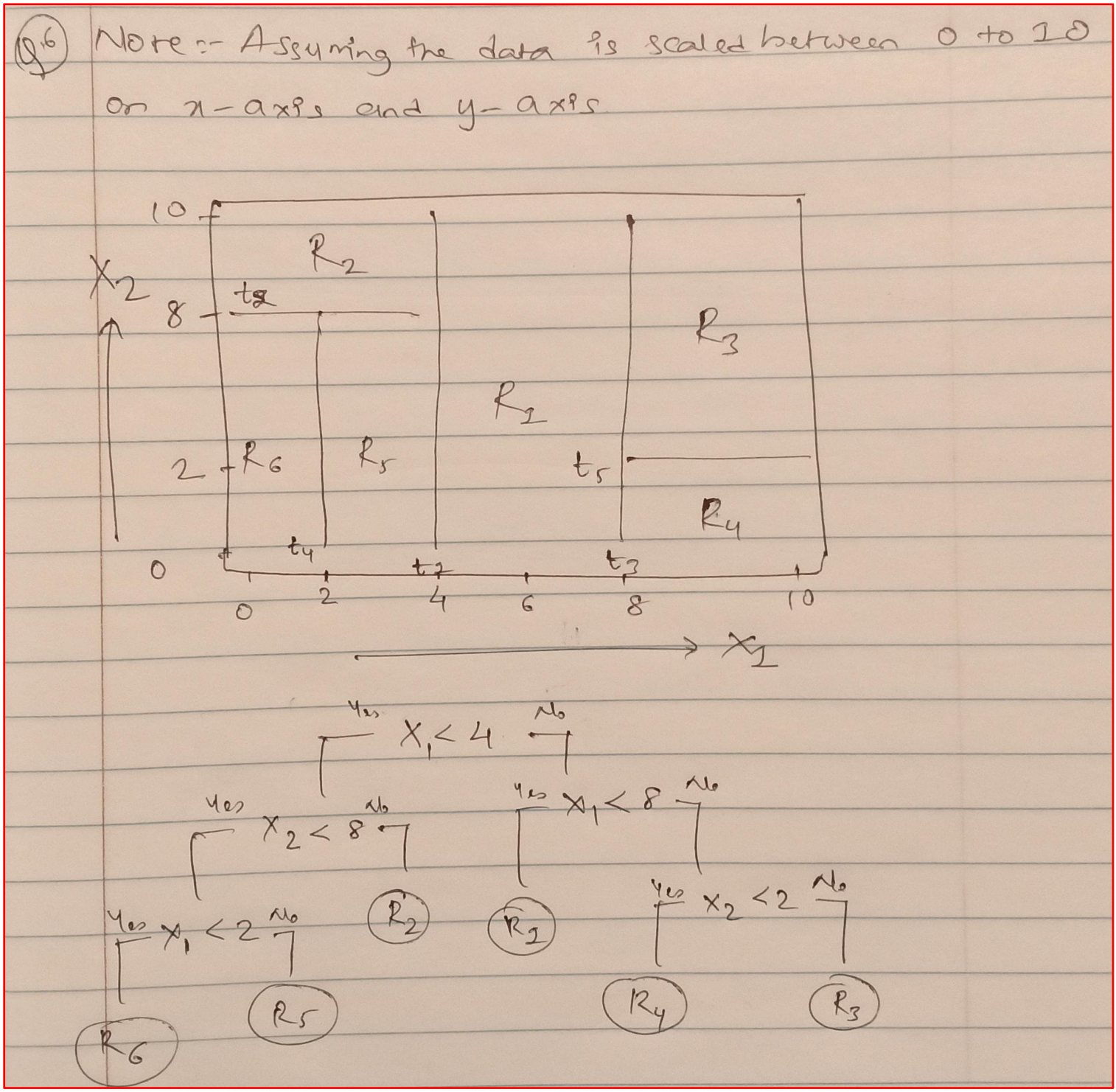
```r
summary(step_fit)
```

```
Call:
glm(formula = wage ~ cut(age, 8), data = Wage)

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)              76.282      2.630  29.007  < 2e-16 ***
cut(age, 8)(25.8,33.5]   25.833      3.161   8.172 4.44e-16 ***
cut(age, 8)(33.5,41.2]   40.226      3.049  13.193  < 2e-16 ***
cut(age, 8)(41.2,49]     43.501      3.018  14.412  < 2e-16 ***
cut(age, 8)(49,56.8]     40.136      3.177  12.634  < 2e-16 ***
cut(age, 8)(56.8,64.5]   44.102      3.564  12.373  < 2e-16 ***
cut(age, 8)(64.5,72.2]   28.948      6.042   4.792 1.74e-06 ***
cut(age, 8)(72.2,80.1]   15.224      9.781   1.556     0.12
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 1597.576)

    Null deviance: 5222086  on 2999  degrees of freedom
Residual deviance: 4779946  on 2992  degrees of freedom
AIC: 30652

Number of Fisher Scoring iterations: 2
```

```r
plot(Wage$age,Wage$wage,xlim=age_lim,cex=.5,col="darkgrey")
lines(age_grid,age_preds$fit,lwd=2,col='red',xlim=age_lim)
se_bands=age_preds$fit + cbind(2*age_preds$se.fit,-2*age_preds$se.fit)
par(mfrow=c(1,2),mar=c(4.5,4.5,1,1) ,oma=c(0,0,4,0))
matlines(age_grid,se_bands,lwd=1,col="blue",lty=3)
```


Wage$wage
50
100
150
200
250
300
20
30
40
50
60
70
80
Wage$age

6. Draw an example (of your own invention) of a partition of two-dimensional feature space that could result from recursive binary splitting. Your example should contain at least six regions. Draw a decision tree corresponding to this partition. Be sure to label all aspects of your figures, including the regions R1, R2, . . ., the cutpoints t1,t2,..., and so forth. Hint: Your result should look something like Figures 8.1 and 8.2.



7. In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable.

(a) Split the data set into a training set and a test set.

```
sum(is.na(Carseats))
```

```
0
```

```
round((nrow(Carseats)*0.2),0)
```

```
80
```

```
test_index=sample(nrow(Carseats),round((nrow(Carseats)*0.2),0),replace=FALSE)
train_set=Carseats[-test_index,]
test_set=Carseats[test_index,]
```

```
c(nrow(train_set),nrow(test_set),nrow(train_set)+nrow(test_set),nrow(Carseats))
```

```
320 · 80 · 400 · 400
```

```
names(Carseats)
```

```
'Sales' · 'CompPrice' · 'Income' · 'Advertising' · 'Population' · 'Price' · 'ShelveLoc' · 'Age' · 'Education' · 'Urban' · 'US'
```

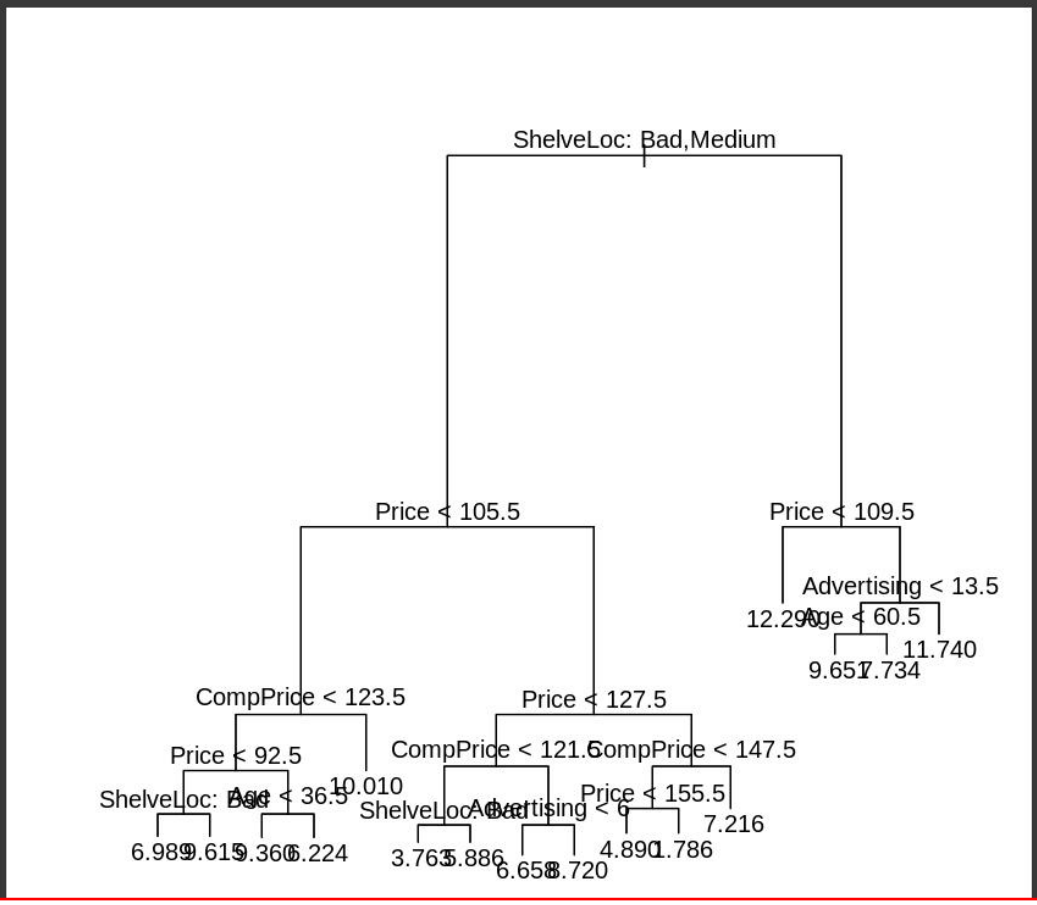
(b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test error rate do you obtain?

```
tree_fit_carseats=tree(Sales~.,data=train_set)
```

```
summary(tree_fit_carseats)
```

```
Regression tree:
tree(formula = Sales ~ ., data = train_set)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "CompPrice"   "Age"         "Advertising"
Number of terminal nodes:  16
Residual mean deviance:  2.368 = 719.8 / 304
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-4.73000 -1.06900 -0.03295  0.00000  1.12500  3.98300
```

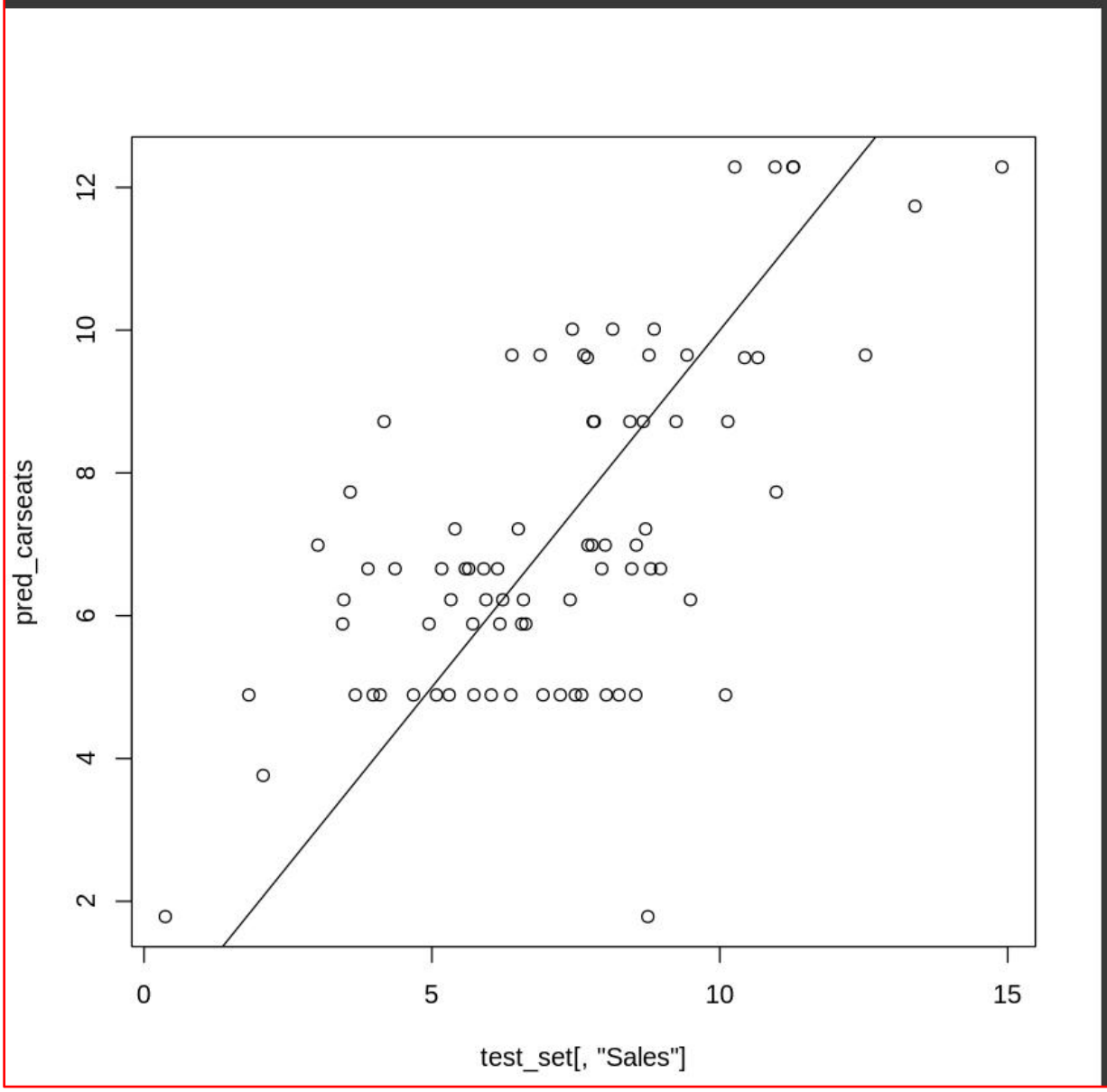
```
plot(tree_fit_carseats)
text(tree_fit_carseats,pretty=0)
```



```r
pred_carseats=predict(tree_fit_carseats,newdata=test_set)
# RSS
print(mean((test_set[,"Sales"]-pred_carseats)^2))
```

```
[1] 4.524503
```

```r
plot(test_set[,"Sales"],pred_carseats)
abline(0,1)
```



We fit a regression tree to the training set of the Carseats dataset, treating Sales as a quantitative variable which involves constructing a decision tree that predicts the numerical value of Sales based on the predictor variables.

The interpretation of the results is as follows:

- ✓ Residual Analysis: The residual, which measures the difference between the observed Sales values and the values predicted by the regression tree, is 2.37. This indicates how much variability in Sales remains unexplained by the tree.

- ✓ Residual Distribution: The distribution of residuals has a median value of -1.07 at the 25th quantile and 1.13 at the 75th quantile. This information helps understand the spread and central tendency of the residuals.

- ✓ Terminal Nodes: The regression tree consists of a total of 16 terminal nodes, which are the endpoints of the tree branches where predictions are made.

- ✓ Test Residual Sum of Squares (RSS): The test RSS, a measure of the total error between predicted and observed Sales values on the test set, is found to be 4.53. This provides an overall assessment of the model's predictive performance.

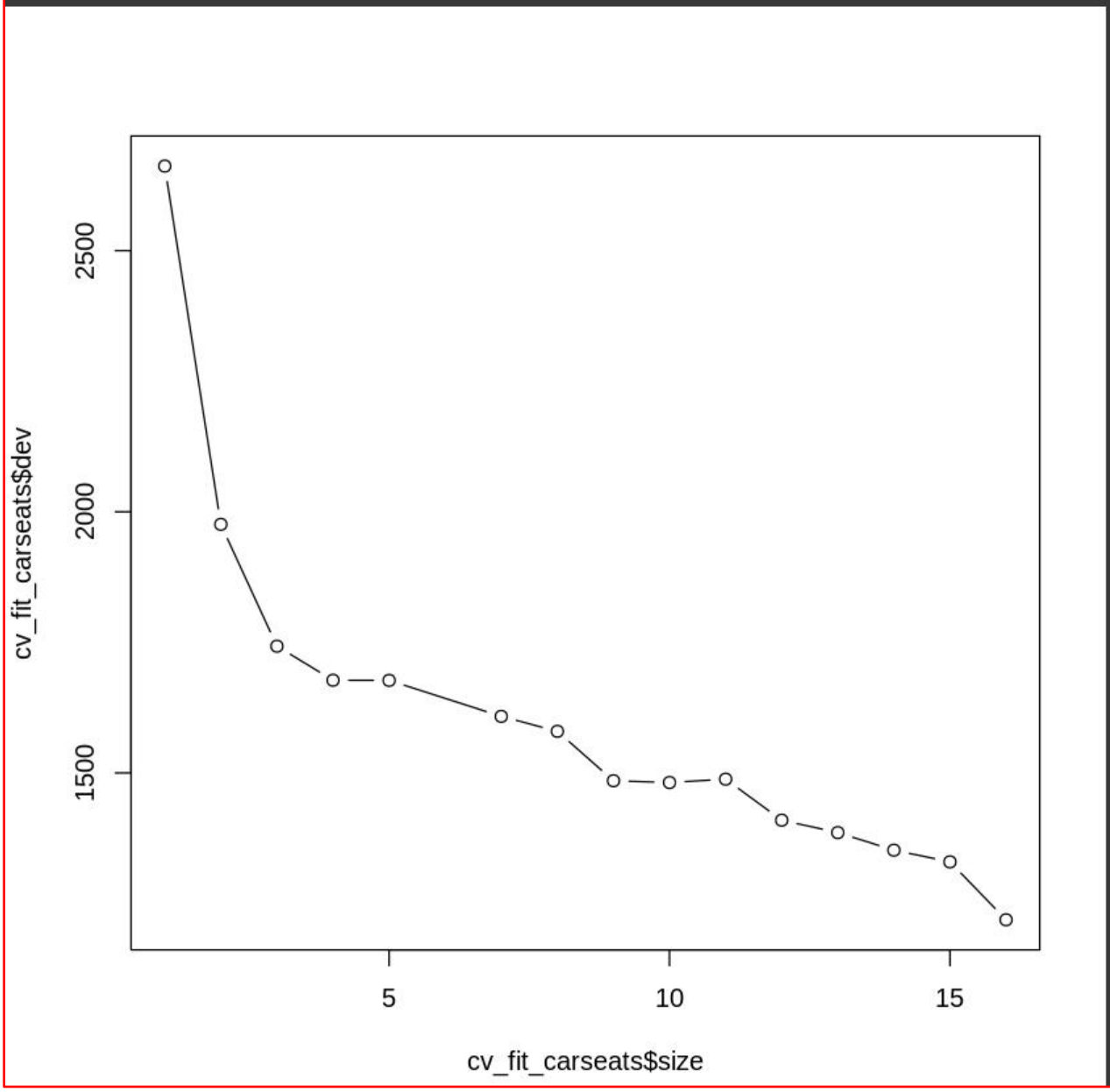
- ✓ Major Splits: The major splits in the tree occur when the Price variable is less than 105.5 and when the Price variable is less than 109.5. These splits indicate the points at which the predictor variables best separate the data into different groups.

- ✓ Predictor Variables: The regression tree utilizes only 5 variables in its construction, namely ShelveLoc, Price, CompPrice, Age, and Advertising. These variables are the most influential in predicting Sales according to the tree's decision-making process.

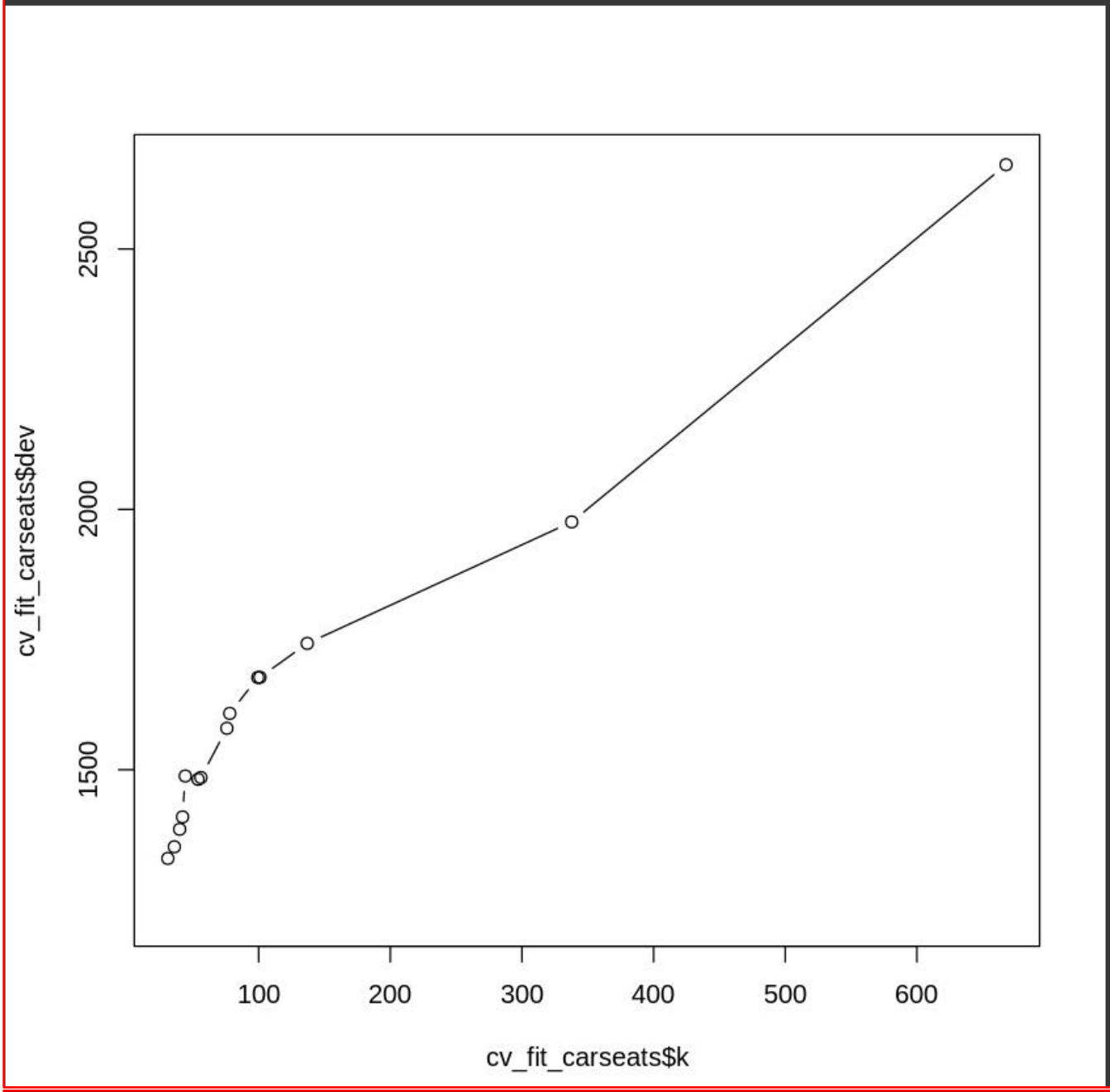
(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test error rate?

```
cv_fit_carseats=cv.tree(tree_fit_carseats,FUN=prune.tree)
```

```
1  plot(cv_fit_carseats$size,cv_fit_carseats$dev,type="b")
```



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




```
cv_fit_carseats$dev
```

1218.82883839918 · 1329.70937648351 · 1352.05355455167 · 1385.75493167961 · 1409.45198923083 · 1488.02747926886 · 1481.6451533784 · 1485.03733853642 · 1579.6427362698 · 1608.2248128534 · 1677.16262975728 · 1677.46664264813 1742.62135775337 · 1975.86783035494 · 2661.80695093087

```
order(cv_fit_carseats$dev,decreasing=TRUE)
```

15 · 14 · 13 · 12 · 11 · 10 · 9 · 6 · 8 · 7 · 5 · 4 · 3 · 2 · 1

```
cv_fit_carseats$dev[order(cv_fit_carseats$dev,decreasing=TRUE)]
```

2661.80695093087 · 1975.86783035494 · 1742.62135775337 · 1677.46664264813 · 1677.16262975728 · 1608.2248128534 · 1579.6427362698 · 1488.02747926886 · 1485.03733853642 · 1481.6451533784 · 1409.45198923083 · 1385.75493167961 1352.05355455167 · 1329.70937648351 · 1218.82883839918

```
which.min(cv_fit_carseats$dev)
```

1

```
cv_fit_carseats$size
```

16 · 15 · 14 · 13 · 12 · 11 · 10 · 9 · 8 · 7 · 5 · 4 · 3 · 2 · 1

```
cv_fit_carseats$size[which.min(cv_fit_carseats$dev)]
```

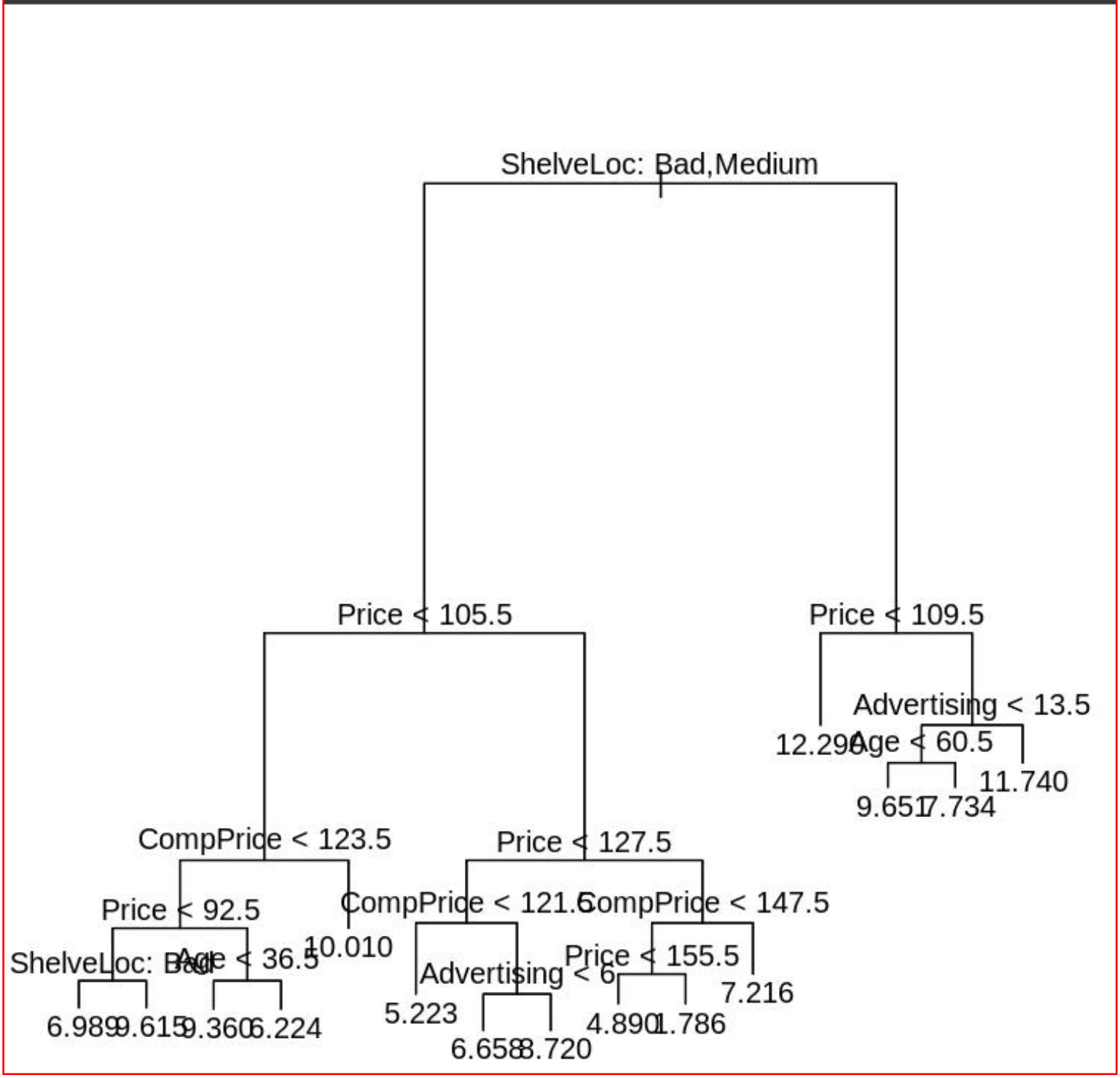
16

```r
cv_fit_carseats$size[2]
```

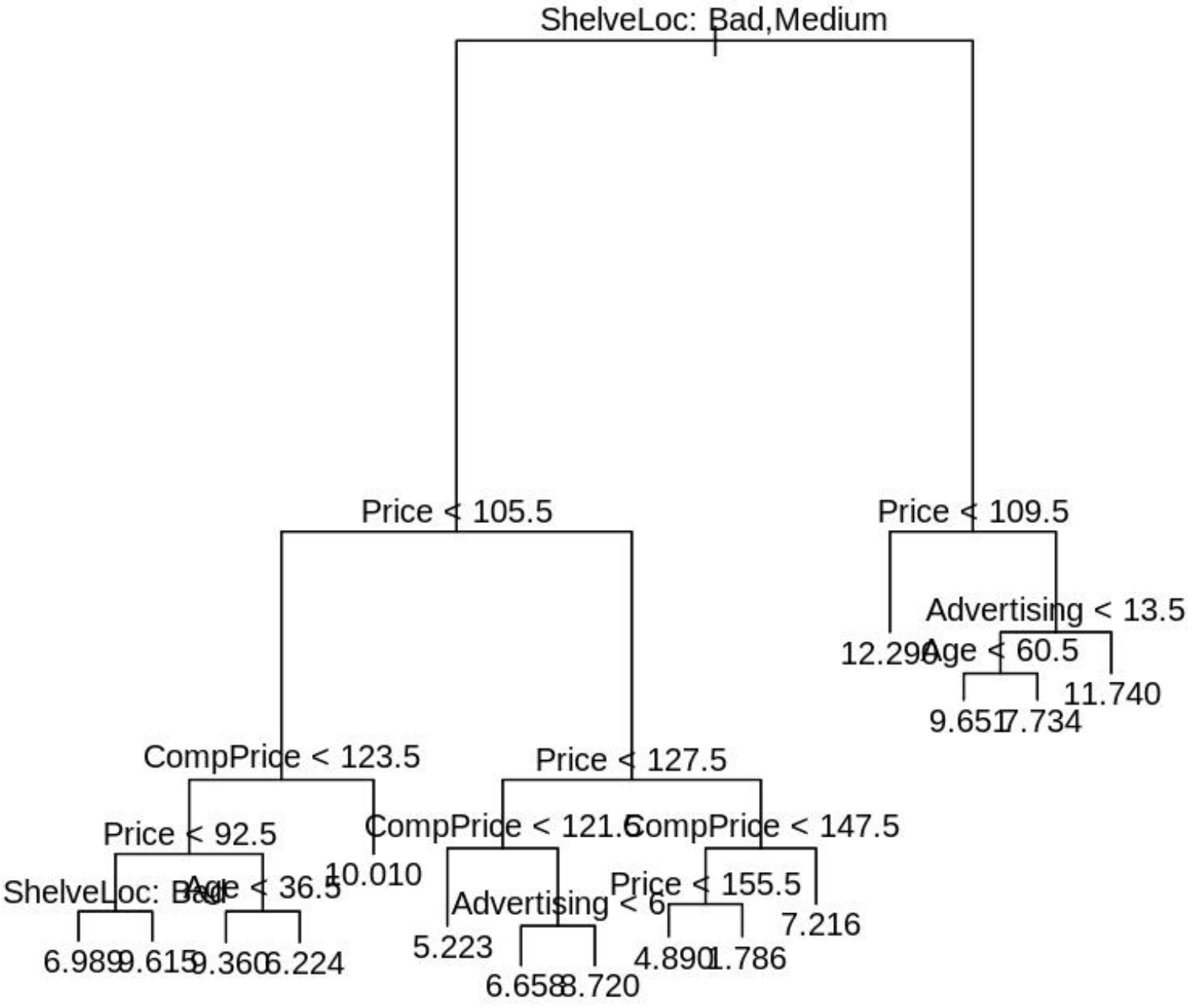
```
15
```

```r
# Best size=15
pruned_tree_fit_carseats=prune.tree(tree_fit_carseats,best=15)
```

```r
par(mfrow=c(1,1))
plot(pruned_tree_fit_carseats)
text(pruned_tree_fit_carseats,pretty=0)
```



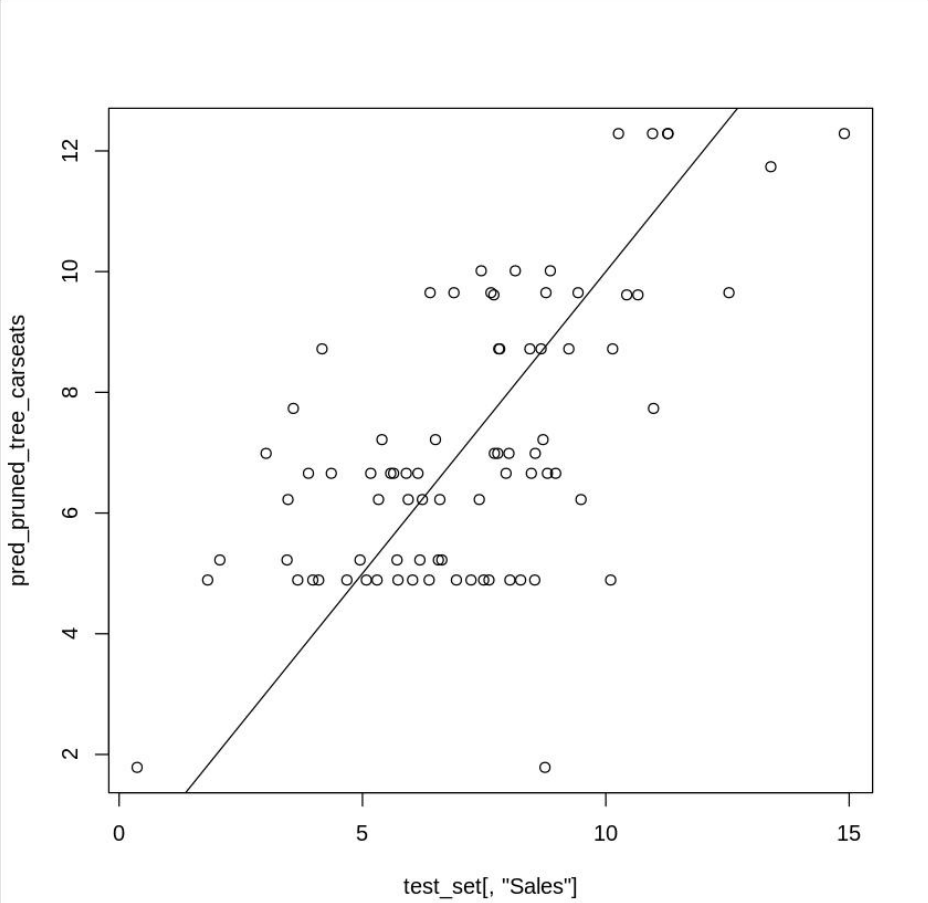
```
plot(pruned_tree_fit_carseats)
text(pruned_tree_fit_carseats,pretty=0)
```



```r
pred_pruned_tree_carseats=predict(pruned_tree_fit_carseats,newdata=test_set)
# RSS
print(mean((test_set[,"Sales"]-pred_pruned_tree_carseats)^2))
```

```
[1] 4.61547
```

```r
plot(test_set[,"Sales"],pred_pruned_tree_carseats)
abline(0,1)
```



After applying cross-validation, we have determined that a regression tree with a total of 15 terminal nodes is optimal for describing the data. This indicates that the tree should have 15 endpoints where predictions are made, which strikes a balance between model complexity and predictive performance. We have used the process of pruning the trees where this process is all about simplifying the tree by removing unnecessary nodes, helping achieve this optimal level of complexity by reducing the number of terminal nodes. This pruning process ensures that the tree is not overly complex and does not capture noise in the data, thereby improving its ability to generalize to new, unseen data. Despite the benefits of pruning in reducing the complexity of the tree, the test Residual Sum of Squares (RSS) is found to be 4.6, which is higher than that of the unpruned tree. This suggests that although pruning helps simplify the tree structure, it may not always lead to a reduction in test error rate.

(d) Use the bagging approach in order to analyze this data. What test error rate do you obtain? Use the importance() function to determine which variables are most important.

```
install.packages("randomForest")
library(randomForest)
set.seed(123)
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

randomForest 4.7-1.1

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

```
names(Carseats)
```

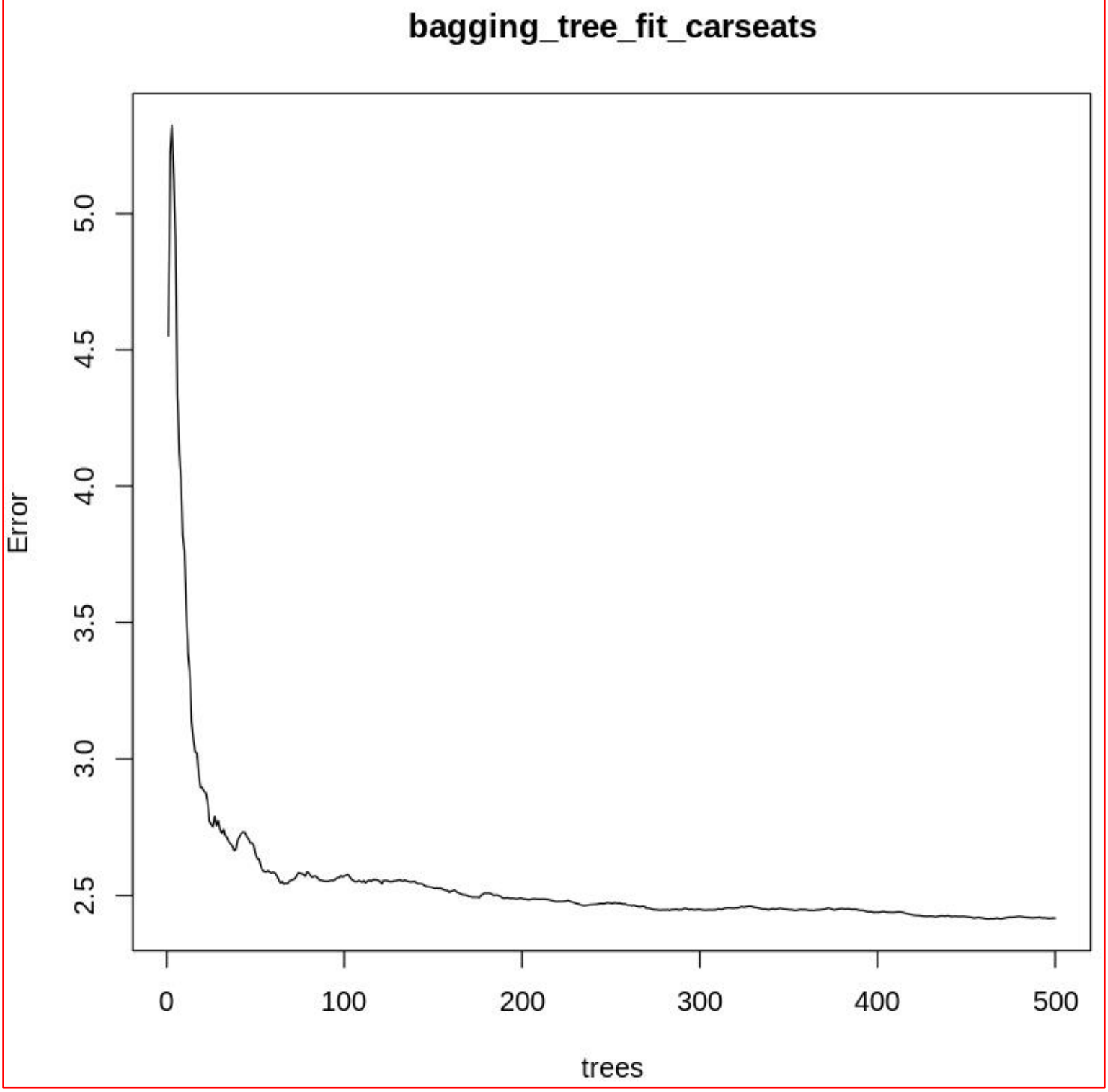
```
'Sales' · 'CompPrice' · 'Income' · 'Advertising' · 'Population' · 'Price' · 'ShelveLoc' · 'Age' · 'Education' · 'Urban' · 'US'
```

```
length(names(Carseats))
```

```
11
```

```
# mtry=10 since there are 10 features in our dataset
bagging_tree_fit_carseats=randomForest(Sales~.,data=train_set,mtry=10,ntree=500,importance=TRUE)
```

```
1  plot(bagging_tree_fit_carseats)
```




```
1  bagging_tree_fit_carseats
```

```
Call:
 randomForest(formula = Sales ~ ., data = train_set, mtry = 10,      ntree = 500, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 10

          Mean of squared residuals: 2.416323
                    % Var explained: 70.48
```

```r
importance(bagging_tree_fit_carseats)
```

A matrix: 10 × 2 of type dbl

	%IncMSE	IncNodePurity
CompPrice	39.2991436	295.86022
Income	8.5434106	127.38073
Advertising	25.9778905	187.24770
Population	-0.8111062	77.44502
Price	75.2007060	793.15835
ShelveLoc	77.4531097	779.61842
Age	24.4879951	223.69338
Education	0.1221569	58.22658
Urban	-1.5513958	12.65388
US	3.4161960	12.28613

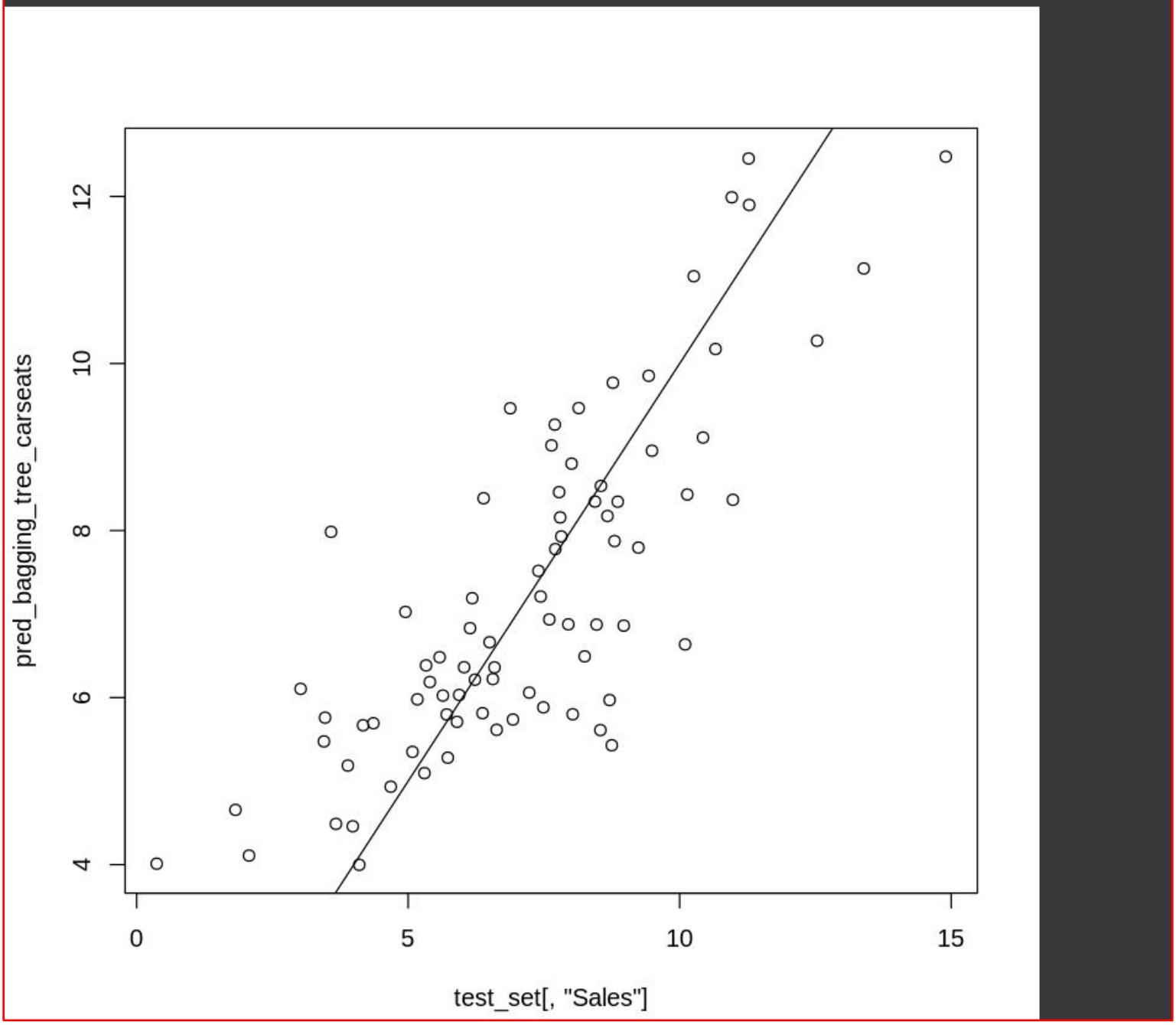
```r
importance_df=as.data.frame(importance(bagging_tree_fit_carseats))
importance_df=importance_df[order(importance_df$IncNodePurity,decreasing=TRUE),]
print(importance_df)
```

```
               %IncMSE IncNodePurity
Price       75.2007060     793.15835
ShelveLoc   77.4531097     779.61842
CompPrice   39.2991436     295.86022
Age         24.4879951     223.69338
Advertising 25.9778905     187.24770
Income       8.5434106     127.38073
Population  -0.8111062      77.44502
Education    0.1221569      58.22658
Urban       -1.5513958      12.65388
US           3.4161960      12.28613
```

```r
pred_bagging_tree_carseats=predict(bagging_tree_fit_carseats,newdata=test_set)
# RSS
print(mean((test_set[,"Sales"]-pred_bagging_tree_carseats)^2))
```

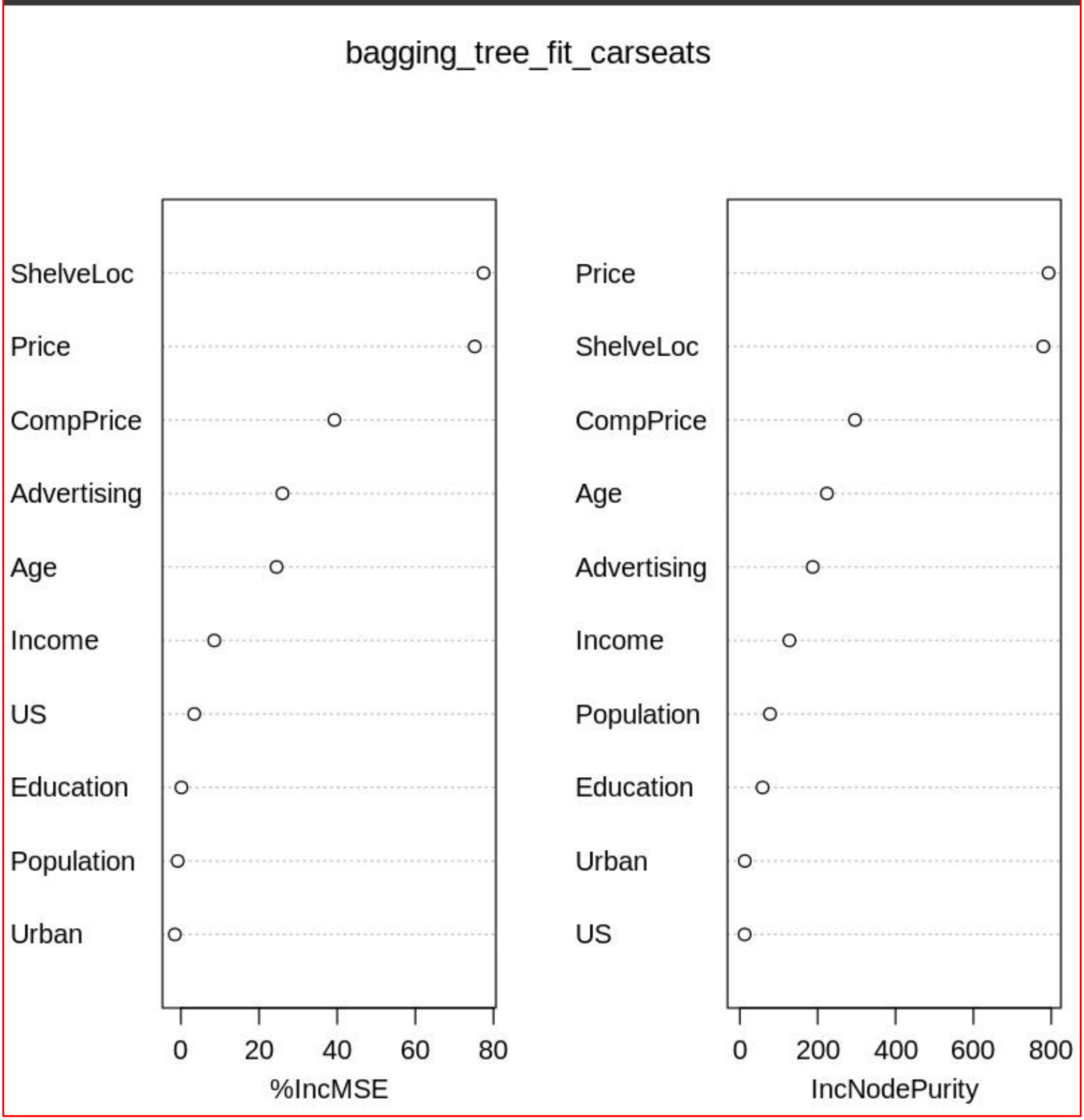
```
[1] 2.4614
```

```
plot(test_set[,"Sales"],pred_bagging_tree_carseats)
abline(0,1)
```



Here we applied the bagging approach to analyze the Carseats dataset, using regression trees to predict Sales. Bagging involves creating multiple decision trees using subsets of the data and then averaging their predictions to improve accuracy and reduce overfitting. Here we have specified the number of trees (ntree) as 500, indicating a large ensemble of trees to capture the variability in the data. This resulted in a mean squared residual of 2.42, indicating the average squared difference between predicted and observed Sales values. Additionally, approximately 70.48% of the variance in Sales is explained by the bagging model. We have the parameter mtry=10 which suggests that all 10 predictors in the dataset were considered for each split of the tree during the bagging process. This allows for a comprehensive exploration of potential predictor variables in the model. To determine variable importance, we have utilized the importance() function, which provides two measures: one based on the mean decrease in accuracy when excluding a variable from the model, and the other based on the total decrease in node impurity resulting from splits over that variable, averaged across all trees. After fitting the model we found that across all trees in the bagging model, ***"Price"*** and ***"ShelveLoc"*** emerge as the two most important variables for predicting Sales. This suggests that variations in ***"Price"*** and ***"ShelveLoc"*** have the most significant impact on Sales according to the bagging model. The test Residual Sum of Squares (RSS) is reported as 2.46, which serves as a measure of the model's predictive accuracy on unseen data.

```
varImpPlot(bagging_tree_fit_carseats)
```



(e) Use random forests to analyze this data. What test error rate do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

```
1   rf_test_tree_fit_carseats=randomForest(Sales~.,data=train_set,mtry=4,ntree=500,importance=TRUE)
```

```
1   rf_test_tree_fit_carseats
```

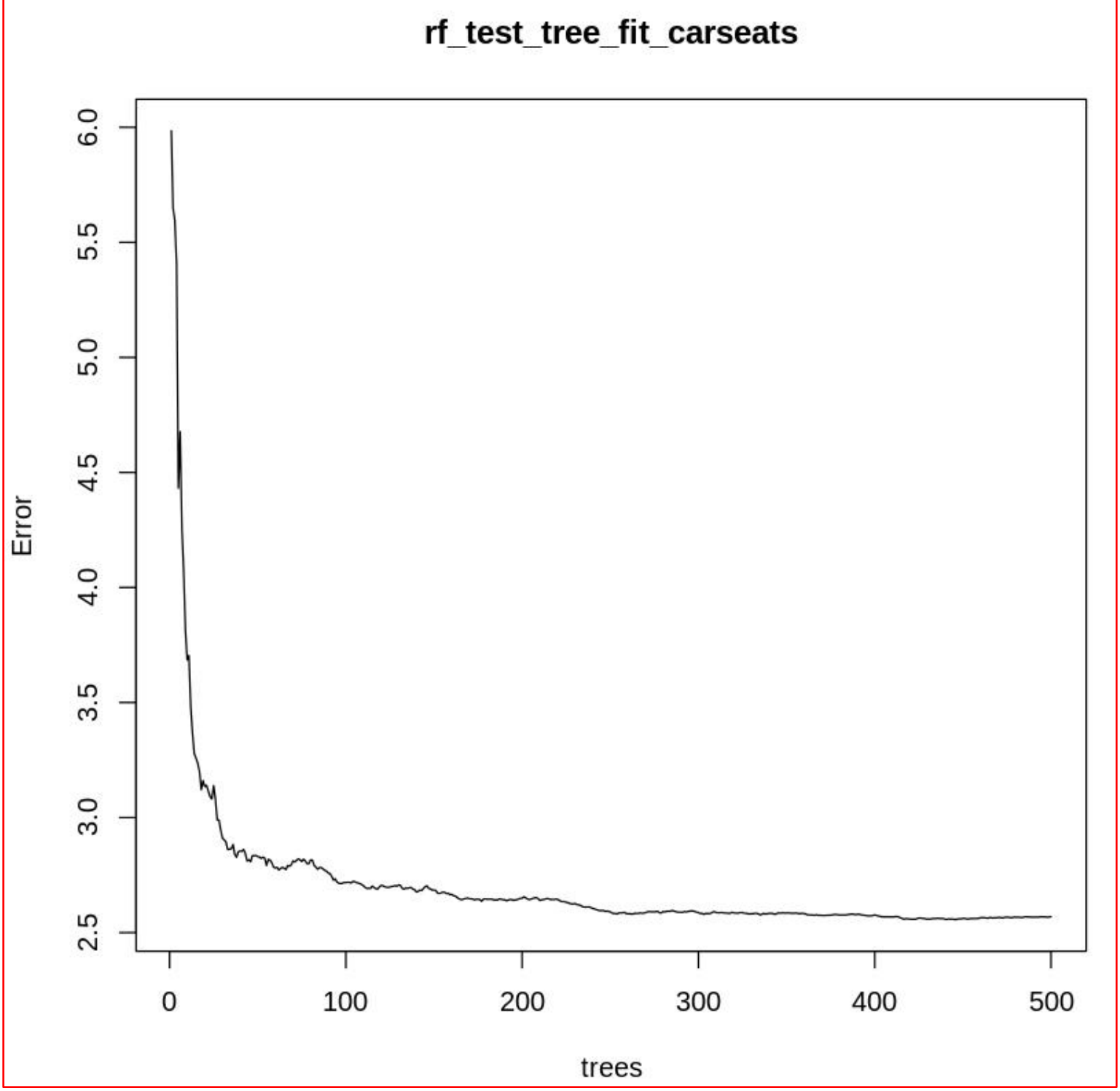
```
Call:
 randomForest(formula = Sales ~ ., data = train_set, mtry = 4,      ntree = 500, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 2.569219
                    % Var explained: 68.61
```

```
1  importance(rf_test_tree_fit_carseats)
```

A matrix: 10 × 2 of type dbl

	%IncMSE	IncNodePurity
CompPrice	22.28020940	248.11592
Income	6.20923721	161.83926
Advertising	19.23940938	220.50459
Population	-1.21460898	136.45040
Price	58.34148199	685.32685
ShelveLoc	62.63892103	657.96488
Age	17.47530115	265.96282
Education	1.80244823	86.70930
Urban	-0.05418617	17.47154
US	4.99441804	34.69823

```r
plot(rf_test_tree_fit_carseats)
```



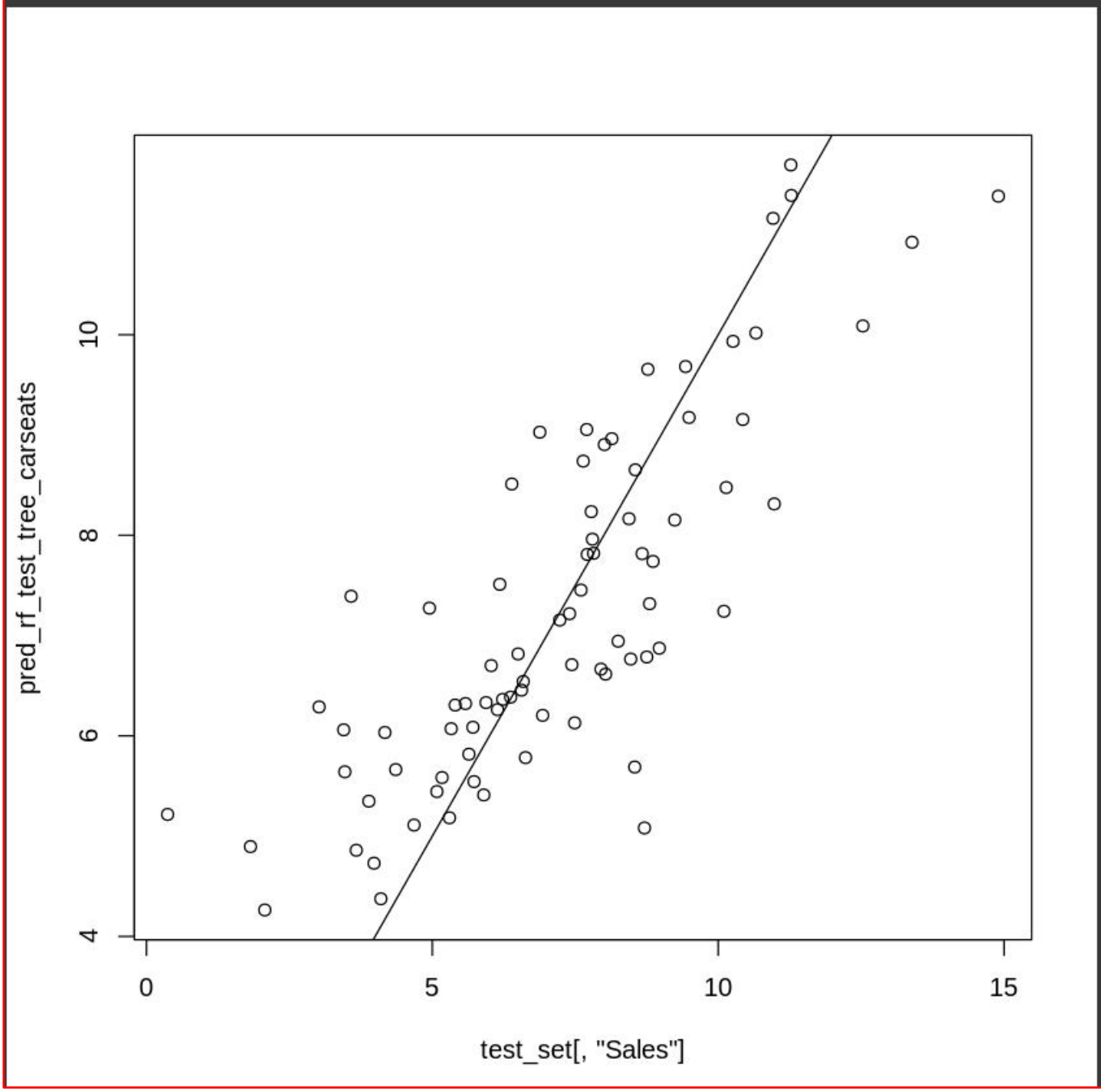
```r
importance_df=as.data.frame(importance(rf_test_tree_fit_carseats))
importance_df=importance_df[order(importance_df$IncNodePurity,decreasing=TRUE),]
print(importance_df)
```

```
               %IncMSE IncNodePurity
Price       58.34148199     685.32685
ShelveLoc   62.63892103     657.96488
Age         17.47530115     265.96282
CompPrice   22.28020940     248.11592
Advertising 19.23940938     220.50459
Income       6.20923721     161.83926
Population  -1.21460898     136.45040
Education    1.80244823      86.70930
US           4.99441804      34.69823
Urban       -0.05418617      17.47154
```

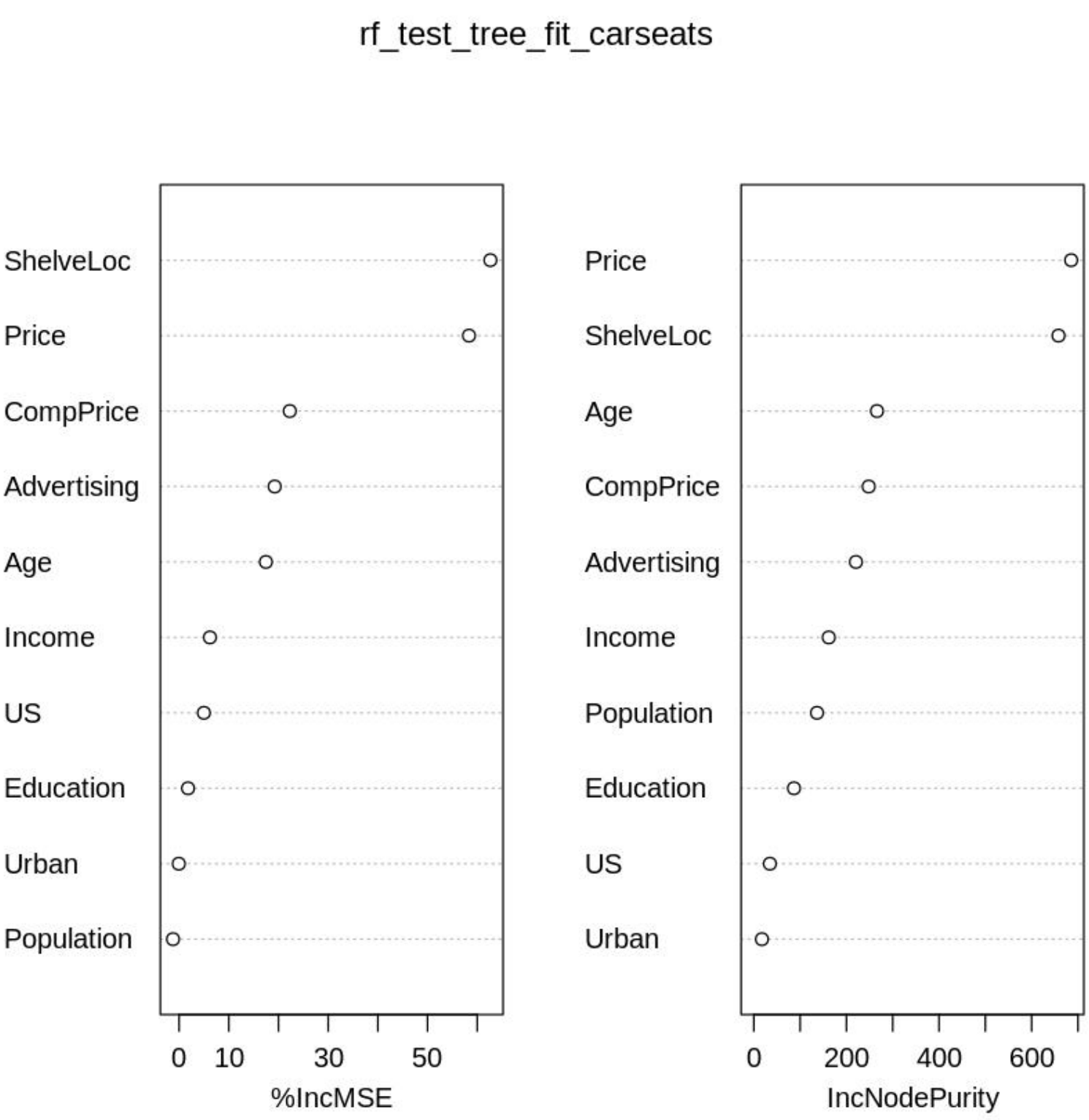
```r
pred_rf_test_tree_carseats=predict(rf_test_tree_fit_carseats,newdata=test_set)
# RSS
print(mean((test_set[,"Sales"]-pred_rf_test_tree_carseats)^2))
```

```
[1] 2.515899
```

```r
plot(test_set[,"Sales"],pred_rf_test_tree_carseats)
abline(0,1)
```



```r
varImpPlot(rf_test_tree_fit_carseats)
```

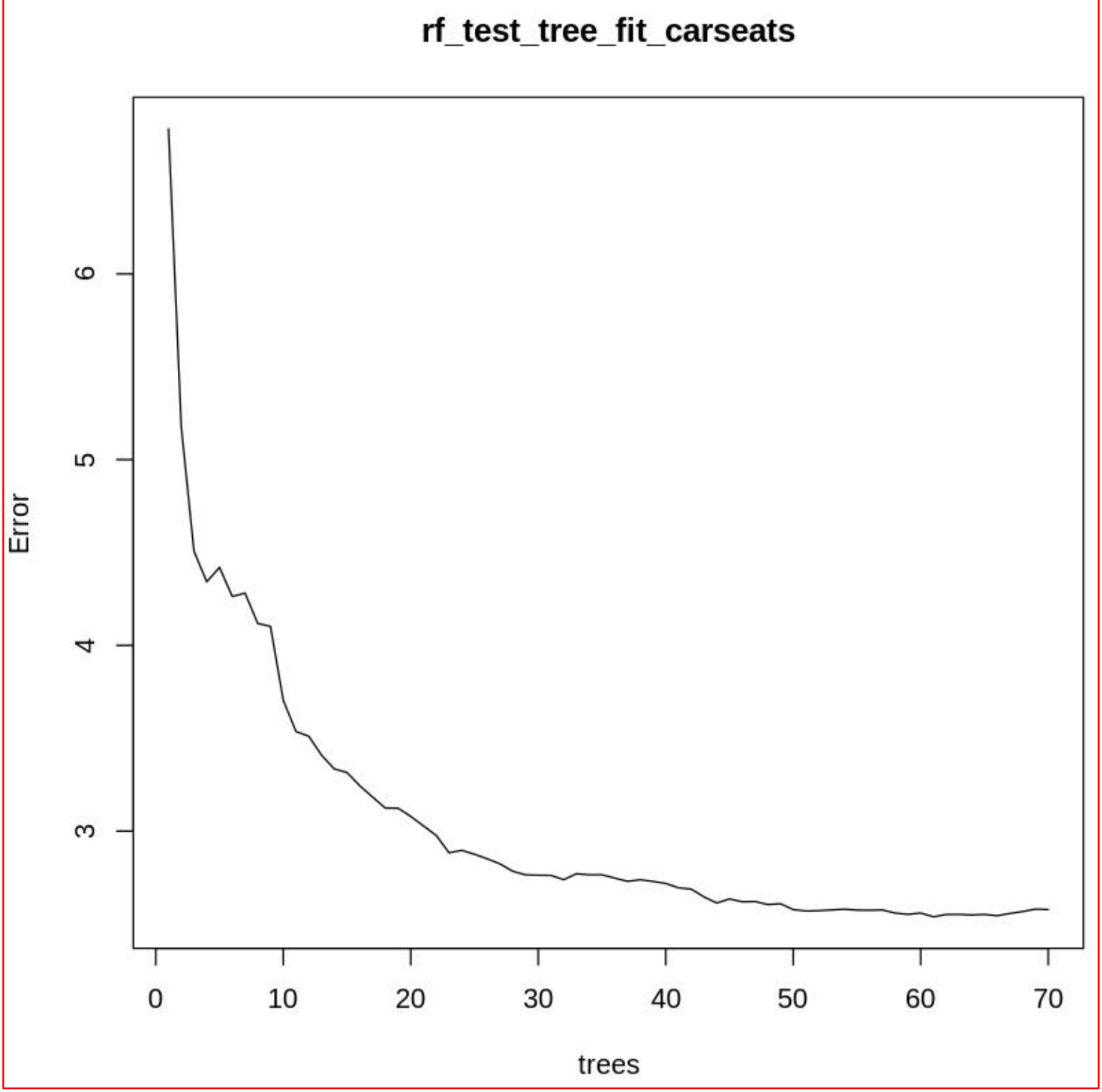


```
1  rf_test_tree_fit_carseats=randomForest(Sales~.,data=train_set,mtry=4,ntree=70,importance=TRUE)
```

```
1  rf_test_tree_fit_carseats
```

```
Call:
 randomForest(formula = Sales ~ ., data = train_set, mtry = 4,      ntree = 70, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 70
No. of variables tried at each split: 4

          Mean of squared residuals: 2.577535
                    % Var explained: 68.51
```

```
1   plot(rf_test_tree_fit_carseats)
```



```
1  importance(rf_test_tree_fit_carseats)
```

A matrix: 10 × 2 of type dbl

	%IncMSE	IncNodePurity
CompPrice	10.8157026	257.62494
Income	2.7129265	166.43085
Advertising	6.7914465	207.61943
Population	0.3878201	143.74432
Price	20.9572455	705.61601
ShelveLoc	23.4340204	677.21917
Age	6.3841250	248.29709
Education	0.7013594	74.82883
Urban	-1.1596397	18.59521
US	3.3066630	28.12392

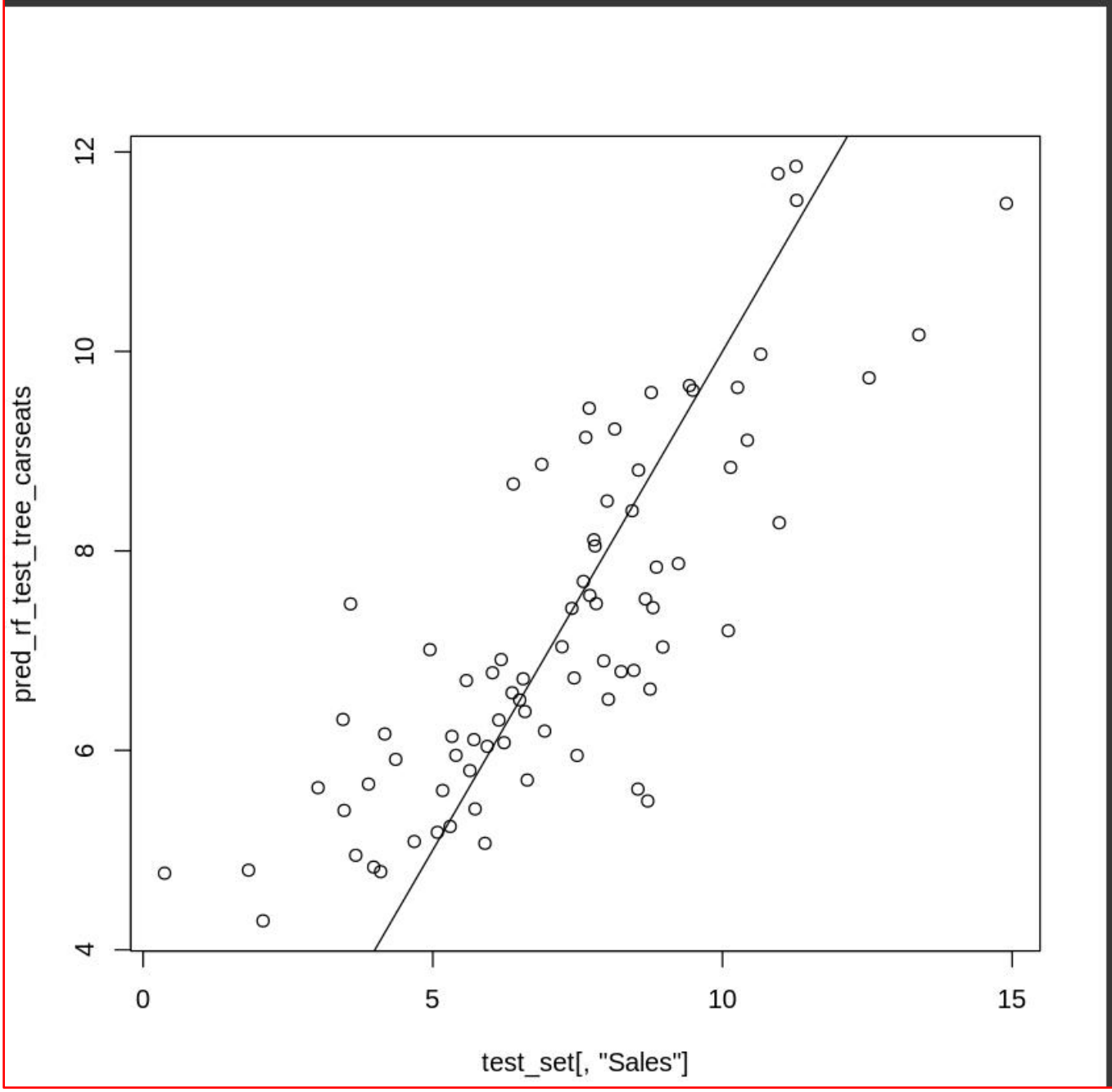
```r
importance_df=as.data.frame(importance(rf_test_tree_fit_carseats))
importance_df=importance_df[order(importance_df$IncNodePurity,decreasing=TRUE),]
print(importance_df)
```

```
              %IncMSE IncNodePurity
Price       20.9572455     705.61601
ShelveLoc   23.4340204     677.21917
CompPrice   10.8157026     257.62494
Age          6.3841250     248.29709
Advertising  6.7914465     207.61943
Income       2.7129265     166.43085
Population   0.3878201     143.74432
Education    0.7013594      74.82883
US           3.3066630      28.12392
Urban       -1.1596397      18.59521
```

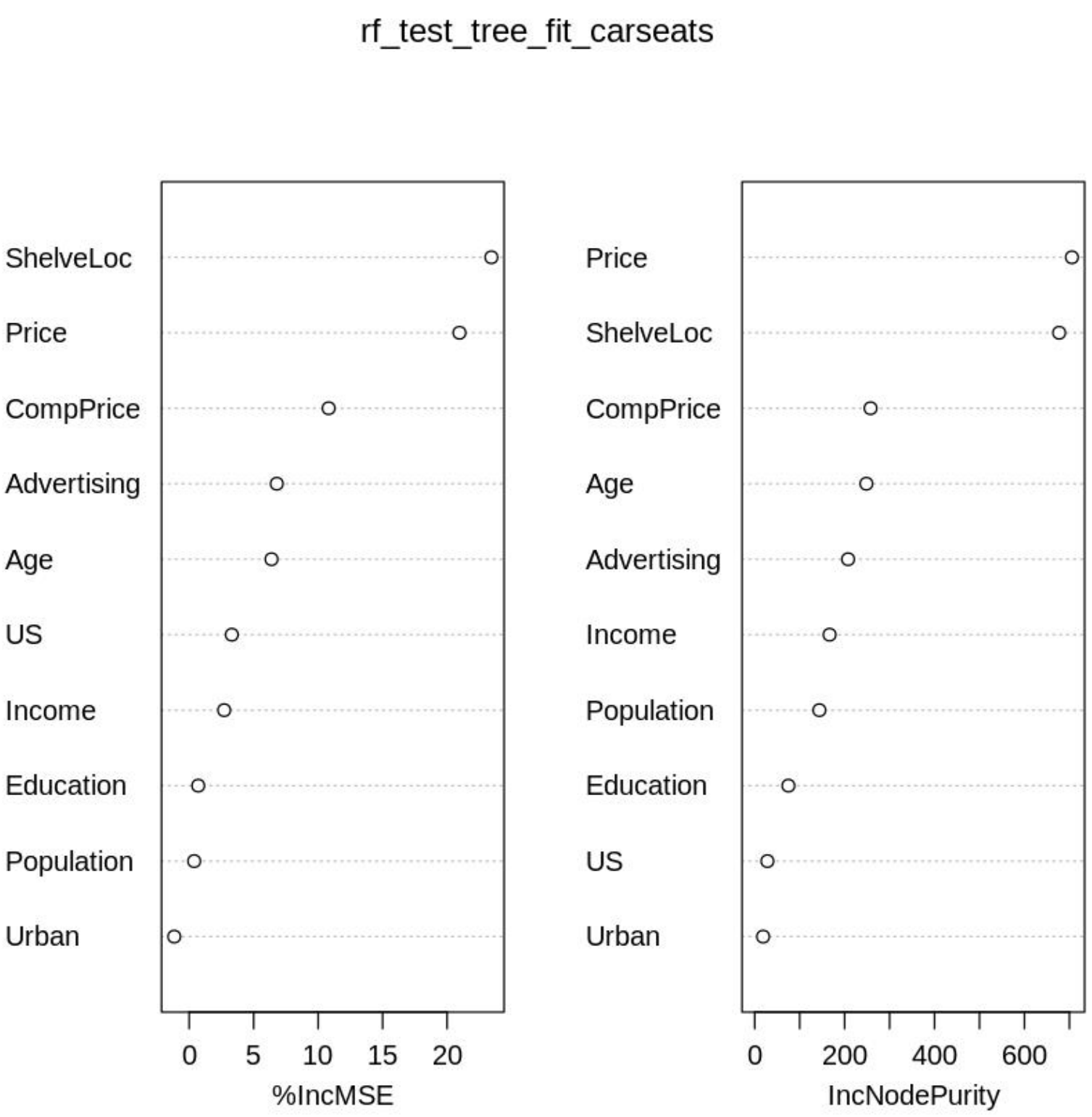
```r
pred_rf_test_tree_carseats=predict(rf_test_tree_fit_carseats,newdata=test_set)
# RSS
print(mean((test_set[,"Sales"]-pred_rf_test_tree_carseats)^2))
```

```
[1] 2.527695
```

```r
plot(test_set[,"Sales"],pred_rf_test_tree_carseats)
abline(0,1)
```

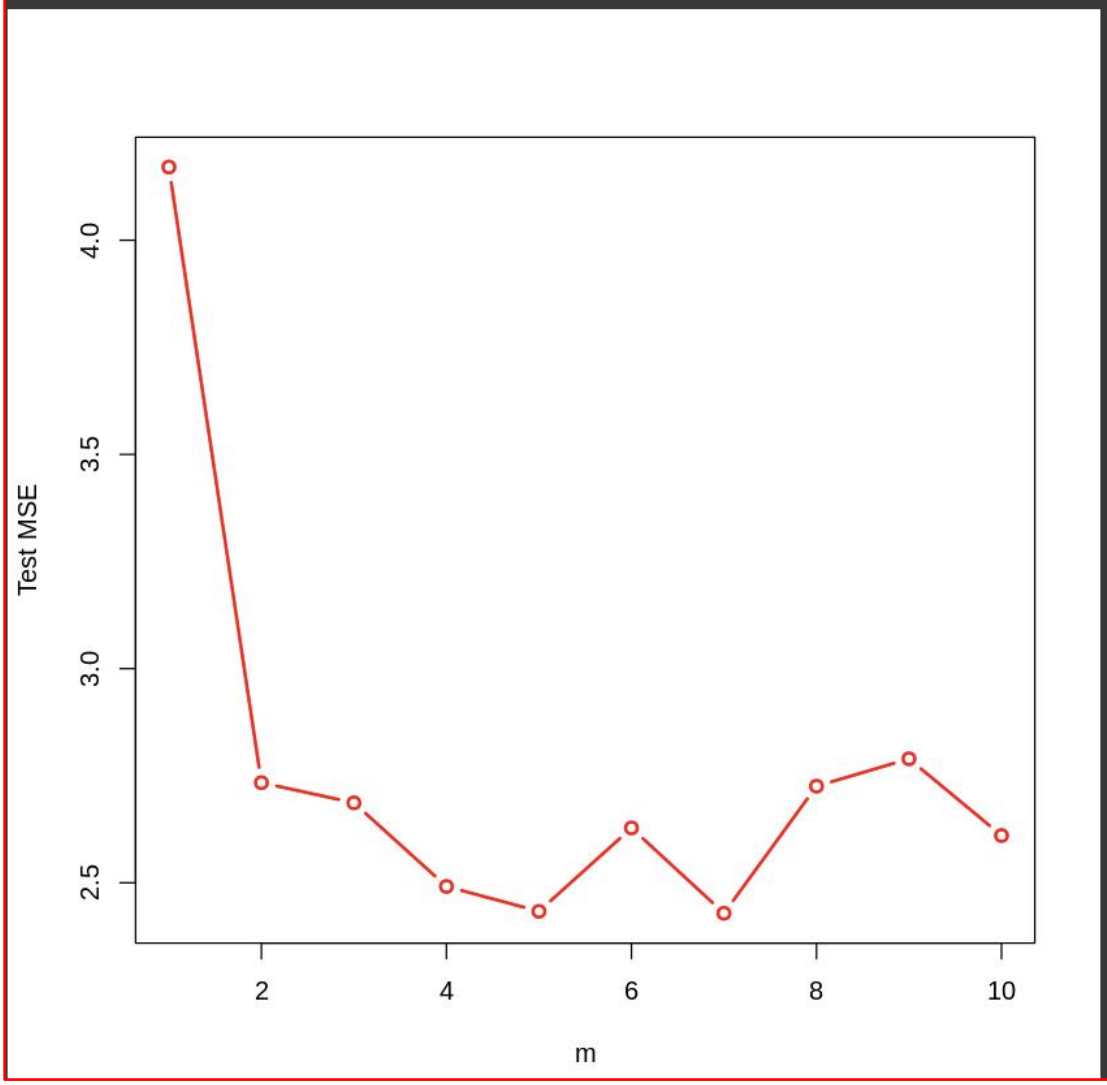


```
1   varImpPlot(rf_test_tree_fit_carseats)
```



```r
d_errors=numeric(10)
for ( d in seq(1,10) ) {
  rf_test_tree_fit_carseats=randomForest(Sales~.,data=train_set,mtry=d,ntree=50,importance=TRUE)
  pred_rf_test_tree_carseats=predict(rf_test_tree_fit_carseats,newdata=test_set)
  # RSS
  d_errors[d]=mean((test_set[,"Sales"]-pred_rf_test_tree_carseats)^2)
}
```

```r
plot(d_errors,col="red",type="b",lwd=2,xlab="m",ylab="Test MSE")
```



Here we have utilized the random forests algorithm to analyze the Carseats dataset for predicting Sales using regression trees. Random forests involve creating multiple decision trees and combining their predictions to improve accuracy and reduce overfitting. Here we have specified different values for the parameters ntree (number of trees) and mtry (number of variables tried at each split) to explore their effects on the model's performance. Firstly, you ran a random forest model with ***ntree=500*** and ***mtry=4***, which means four variables are considered at each split. This model yielded a mean squared residual of ***2.6*** and explained approximately ***68.61%*** of the variance in Sales. The resulting test Residual Sum of Squares (RSS) was ***2.52***, providing an assessment of the model's predictive accuracy on unseen data. Secondly, we ran another random forest model with ***ntree=70*** and ***mtry=4***, which produced similar results with a mean squared residual of ***2.6*** and explained a variance of approximately ***68.51%***. The test RSS for this model was ***2.53***, indicating comparable predictive performance to the previous model. And since the variance was about the same with a very small change in test Residual Sum of Squares (RSS) (a change of only ***0.01***) we finalized ntree to be 50 so that the complexity of the tree remains less. Lastly, we conducted a series of experiments by fitting trees with ***ntree=50*** and varying mtry from 1 to 10. By observing the RSS for each model, we found that the lowest RSS values were obtained for mtry values of ***5*** and ***7***, followed by ***mtry=4***. This suggests that considering a moderate number of variables at each split, such as 5 or 7, tends to lead to better predictive performance compared to considering fewer or more variables.