Tutorial 4

Quentin Bouet

2024-08-13

Regression Analysis

From this week onwards, we will be exploring different types of supervised learning methods. Generally, a predictive task can be either a classification or regression problem. In a classification problem, the goal is to categorise data into predefined classes or categories. For example, classifying emails as "spam" or "not spam" involves predicting which of the discrete categories an email belongs to based on its features. In contrast, a regression problem involves predicting a continuous numerical value. For instance, forecasting the price of a house based on features like location, size, and number of rooms requires estimating a continuous output. While classification yields discrete outcomes, regression provides a continuous range of values, and the choice between the two depends on the nature of the prediction task at hand.

We will start with the simplest approach: assuming a linear relationship between predictors and response. This week, we focus on regression problems, and next week, we will explore models for classification problems.

There is so much to learn about linear and generalized linear models; attempting to thoroughly understand the art of linear modelling in just one or two lectures is ludicrous. But this is unfortunately the modern approach to 'Data Science': knowing how to program it without understanding the ins and outs of the model. Only by truly understanding linear or generalised linear model can we grasp why they are often not desirable for the problem at hand, yet still frequently chosen as contenders.

Discuss the following topics:

What is the model structure of a linear model?

What are the parameters of a linear model?

How do we estimate the parameters?

Why is there a variance associated with each parameter?

Explain a residual in layman's terms

What are assumptions of a linear model?

Explain variance-bias trade-off

Describe how forward and backward model selection work?

Assume a predictor has four categories; explain why R only produces coefficients for three of those categories.

3.6 Lab: Linear Regression

3.6.1 Libraries

The library() function is used to load libraries, or groups of functions library() and data sets that are not included in the base R distribution. Basic func- tions that perform least squares linear regression and other

simple analyses come standard with the base distribution, but more exotic functions require additional libraries. Here we load the MASS package, which is a very large collection of data sets and functions. We also load the ISLR2 package, which includes the data sets associated with this book.

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 4.3.3
```

```
library(ISLR2)
```

```
## Warning: package 'ISLR2' was built under R version 4.3.3
```

```
##
## Attaching package: 'ISLR2'
```

```
## The following object is masked from 'package:MASS':
##
##     Boston
```

If you receive an error message when loading any of these libraries, it likely indicates that the corresponding library has not yet been installed on your system. Some libraries, such as MASS, come with R and do not need to be separately installed on your computer. However, other packages, such as ISLR2, must be downloaded the first time they are used. This can be done di- rectly from within R. For example, on a Windows system, select the Install package option under the Packages tab. After you select any mirror site, a list of available packages will appear. Simply select the package you wish to install and R will automatically download the package. Alternatively, this can be done at the R command line via install.packages("ISLR2"). This installation only needs to be done the first time you use a package. However, the library() function must be called within each R session.

3.6.2 Simple Linear Regression

The ISLR2 library contains the Boston data set, which records medv (me- dian house value) for 506 census tracts in Boston. We will seek to predict medv using 12 predictors such as rm (average number of rooms per house), age (proportion of owner-occupied units built prior to 1940) and lstat (per- cent of households with low socioeconomic status).

```
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7  5.33 36.2
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7  5.21 28.7
```

To find out more about the data set, we can type ?Boston. We will start by using the lm() function to fit a simple linear regression model, with medv as the response and lstat as the predictor. The basic syntax is lm(y ~ x, data), where y is the response, x is the predictor, and data is the data set in which these two variables are kept.

```
lm.fit <- lm(medv ~ lstat, Boston)

# OR
attach(Boston)
lm.fit <- lm(medv ~ lstat)
```

If we type lm.fit, some basic information about the model is output. For more detailed information, we use summary(lm.fit). This gives us p- values and standard errors for the coefficients, as well as the R2 statistic and F -statistic for the model.

```
lm.fit
```

```
##
## Call:
## lm(formula = medv ~ lstat)
##
## Coefficients:
## (Intercept)        lstat
##       34.55        -0.95
```

```
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.168  -3.990  -1.318   2.034  24.500
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.55384    0.56263   61.41   <2e-16 ***
## lstat       -0.95005    0.03873  -24.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.216 on 504 degrees of freedom
## Multiple R-squared:  0.5441, Adjusted R-squared:  0.5432
## F-statistic: 601.6 on 1 and 504 DF,  p-value: < 2.2e-16
```

We can use the names() function in order to find out what other pieces of information are stored in lm.fit. Although we can extract these quan- tities by name—e.g. lm.fit$coefficients—it is safer to use the extractor functions like coef() to access them.

```
names(lm.fit)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

```
coef(lm.fit)
```

```
## (Intercept)       lstat
##  34.5538409  -0.9500494
```

In order to obtain a confidence interval for the coefficient estimates, we can use the confint() command.

```
confint(lm.fit)
```

```
##                 2.5 %     97.5 %
## (Intercept) 33.448457 35.6592247
## lstat       -1.026148 -0.8739505
```

The predict() function can be used to produce confidence intervals and prediction intervals for the prediction of medv for a given value of lstat.

```
predict(lm.fit , data.frame(lstat = (c(5, 10, 15))),
interval = "confidence")
```

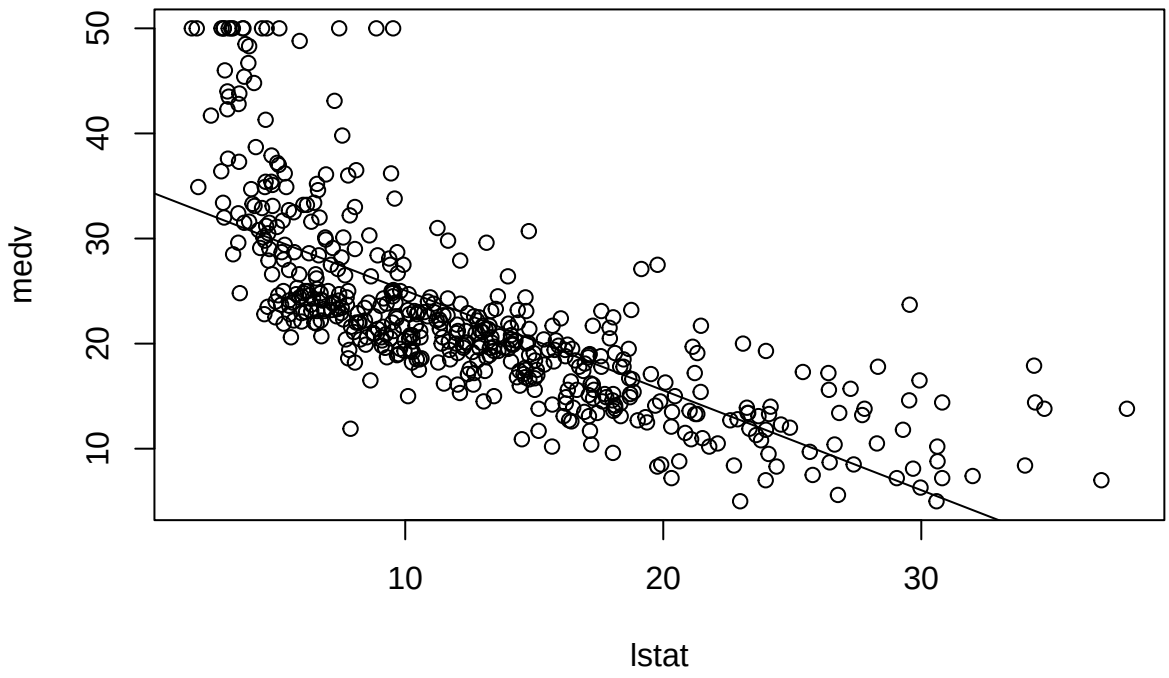
```
##        fit      lwr      upr
## 1 29.80359 29.00741 30.59978
## 2 25.05335 24.47413 25.63256
## 3 20.30310 19.73159 20.87461
```

```
predict(lm.fit , data.frame(lstat = (c(5, 10, 15))),
interval = "prediction")
```

```
##        fit       lwr      upr
## 1 29.80359 17.565675 42.04151
## 2 25.05335 12.827626 37.27907
## 3 20.30310  8.077742 32.52846
```

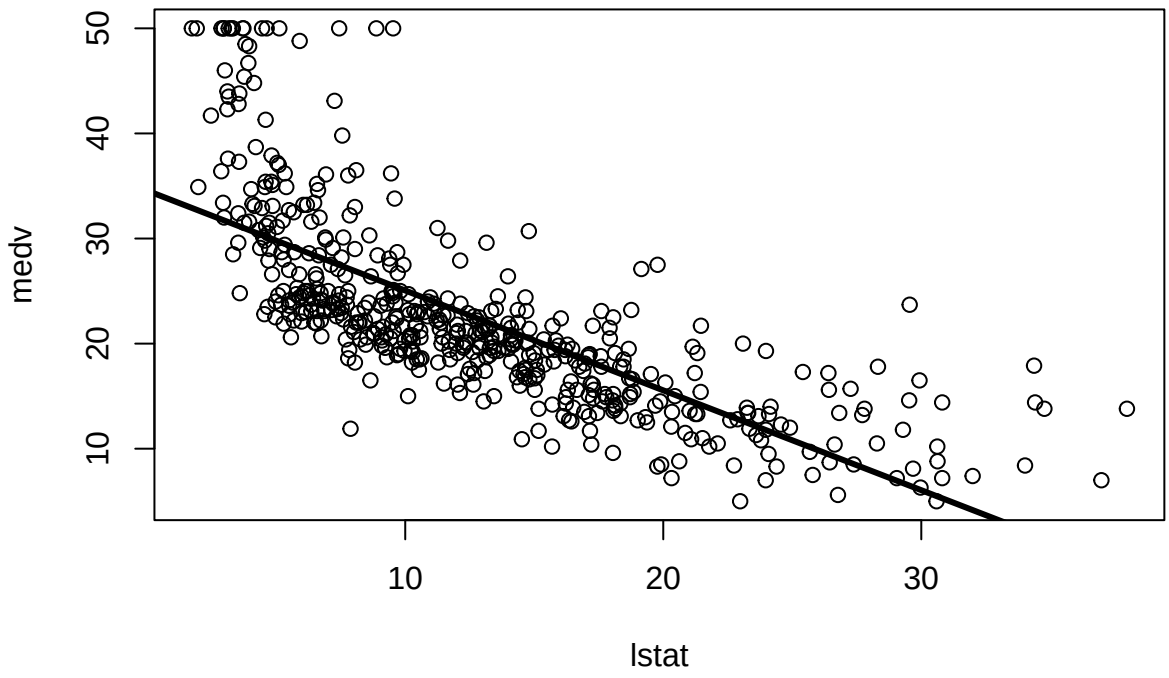
For instance, the 95 % confidence interval associated with a lstat value of 10 is (24.47, 25.63), and the 95 % prediction interval is (12.828, 37.28). As expected, the confidence and prediction intervals are centered around the same point (a predicted value of 25.05 for medv when lstat equals 10), but the latter are substantially wider. We will now plot medv and lstat along with the least squares regression line using the plot() and abline() functions.

```
plot(lstat , medv)
abline(lm.fit)
```

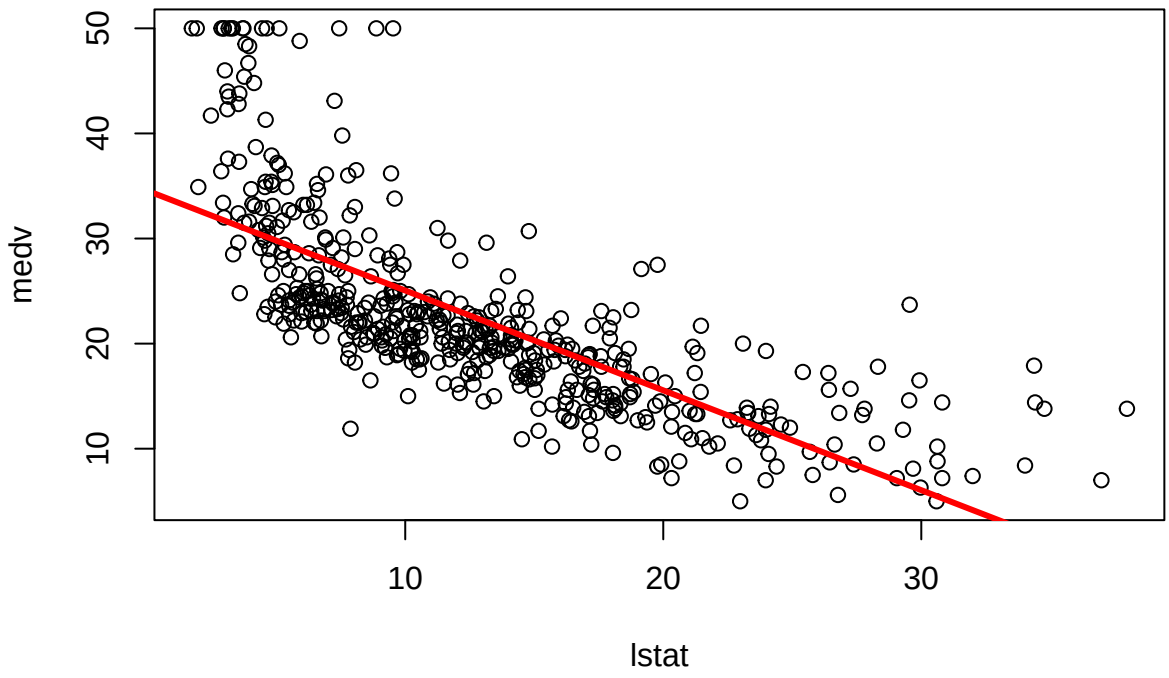


There is some evidence for non-linearity in the relationship between lstat and medv. We will explore this issue later in this lab. The abline() function can be used to draw any line, not just the least squares regression line. To draw a line with intercept a and slope b, we type abline(a, b). Below we experiment with some additional settings for plotting lines and points. The lwd = 3 command causes the width of the regression line to be increased by a factor of 3; this works for the plot() and lines() functions also. We can also use the pch option to create different plotting symbols.

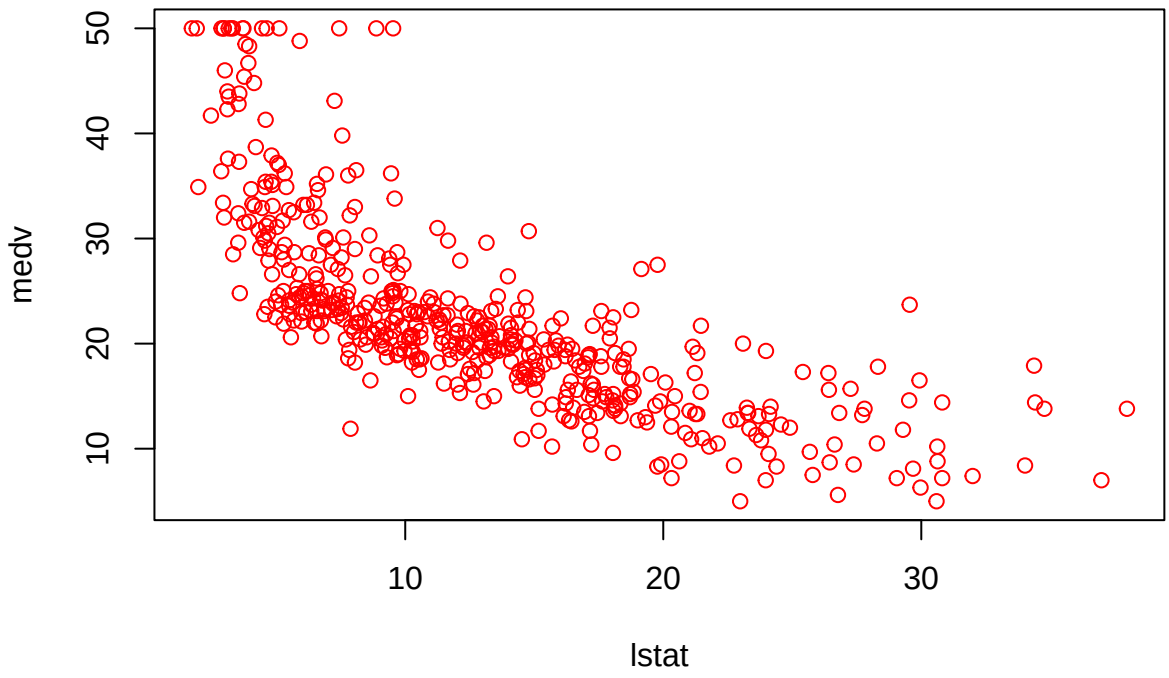
```
plot(lstat , medv)
abline(lm.fit, lwd = 3)
```



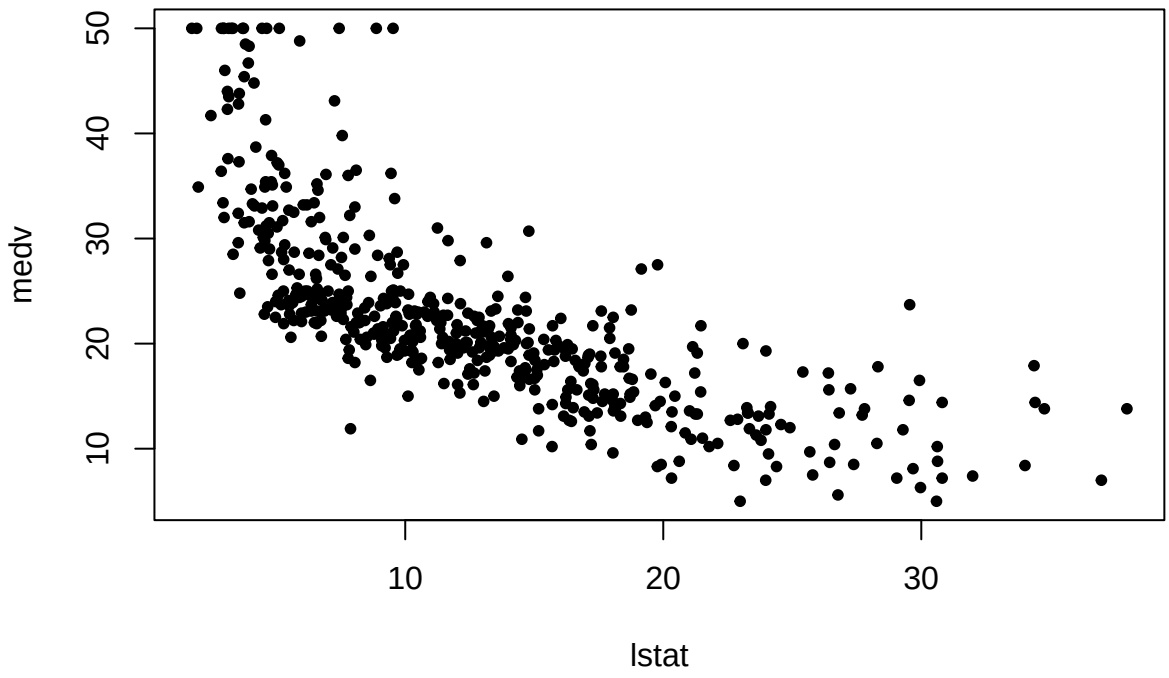

```
plot(lstat , medv)
abline(lm.fit , lwd = 3, col = "red")
```



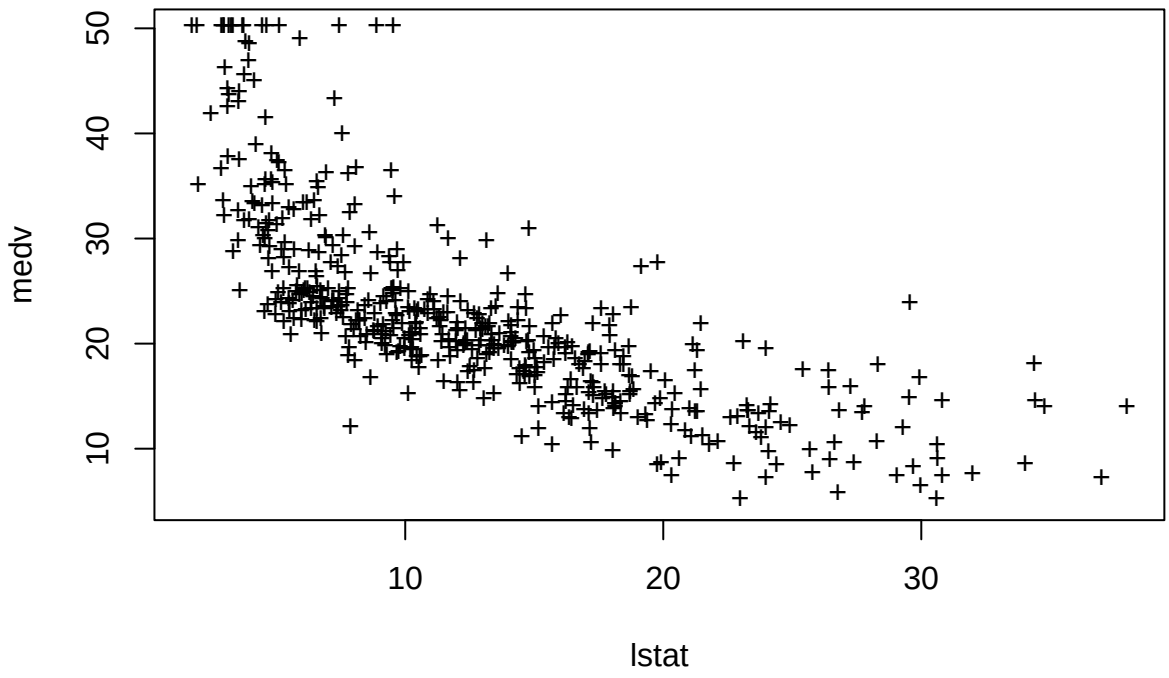

```
plot(lstat , medv , col = "red")
```



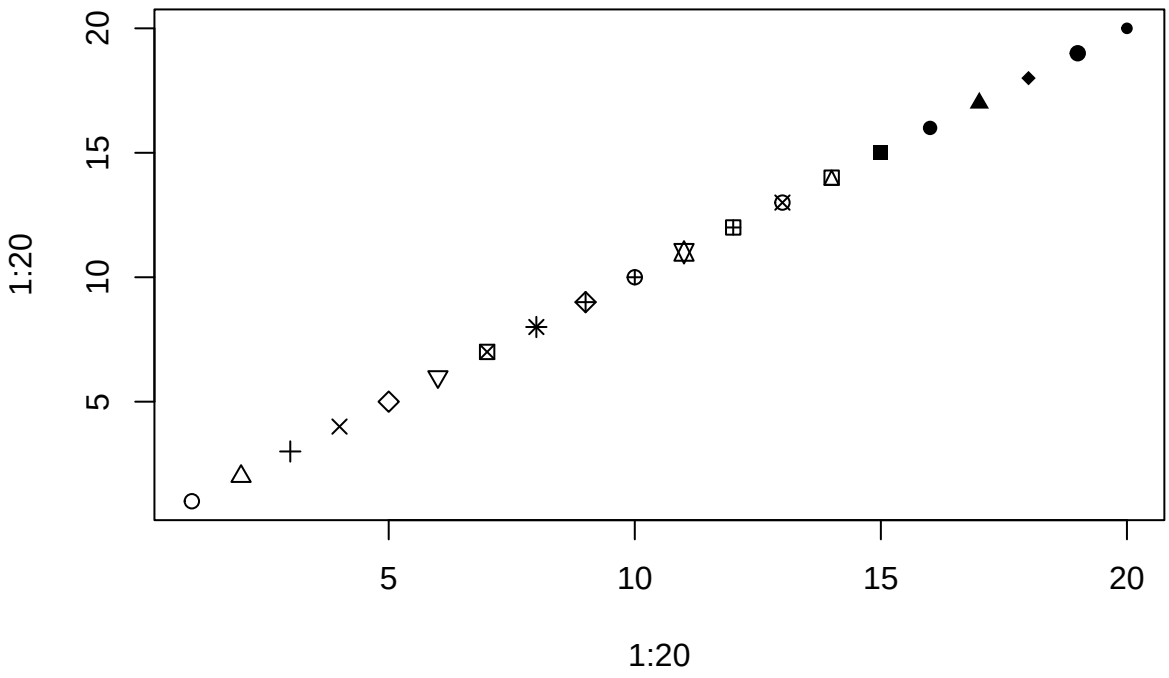

```
plot(lstat , medv , pch = 20)
```




```
plot(lstat , medv , pch = "+")
```

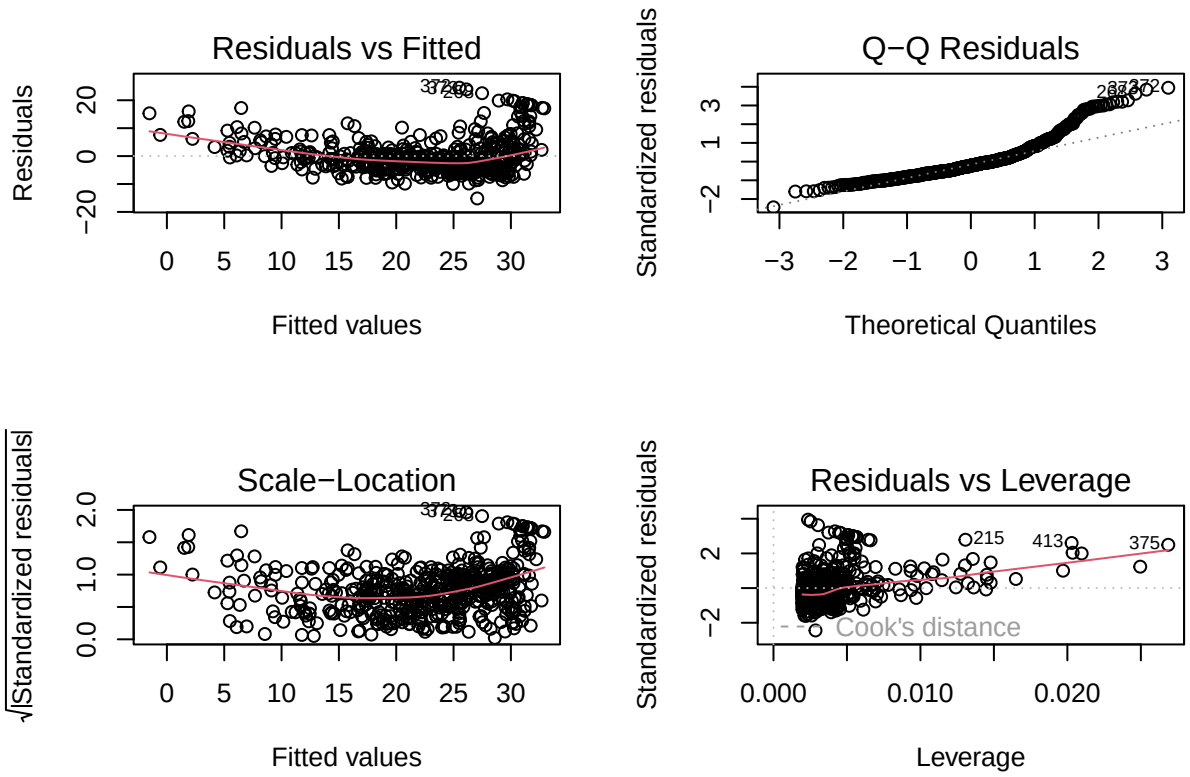



```r
plot(1:20 , 1:20, pch = 1:20)
```



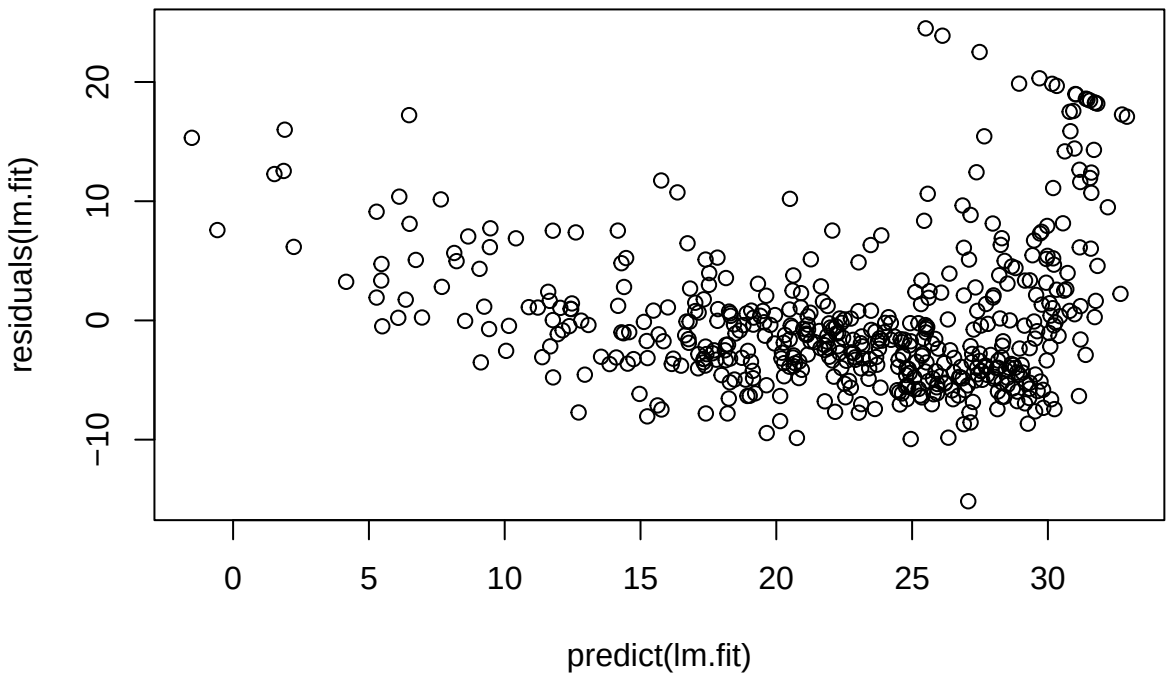
Next we examine some diagnostic plots, several of which were discussed in Section 3.3.3. Four diagnostic plots are automatically produced by ap- plying the plot() function directly to the output from lm(). In general, this command will produce one plot at a time, and hitting Enter will generate the next plot. However, it is often convenient to view all four plots together. We can achieve this by using the par() and mfrow() functions, which tell R to split the display screen into separate panels so that multiple plots can be viewed simultaneously. For example, par(mfrow = c(2, 2)) divides the plotting region into a 2 × 2 grid of panels.

```
par(mfrow = c(2, 2))
plot(lm.fit)
```

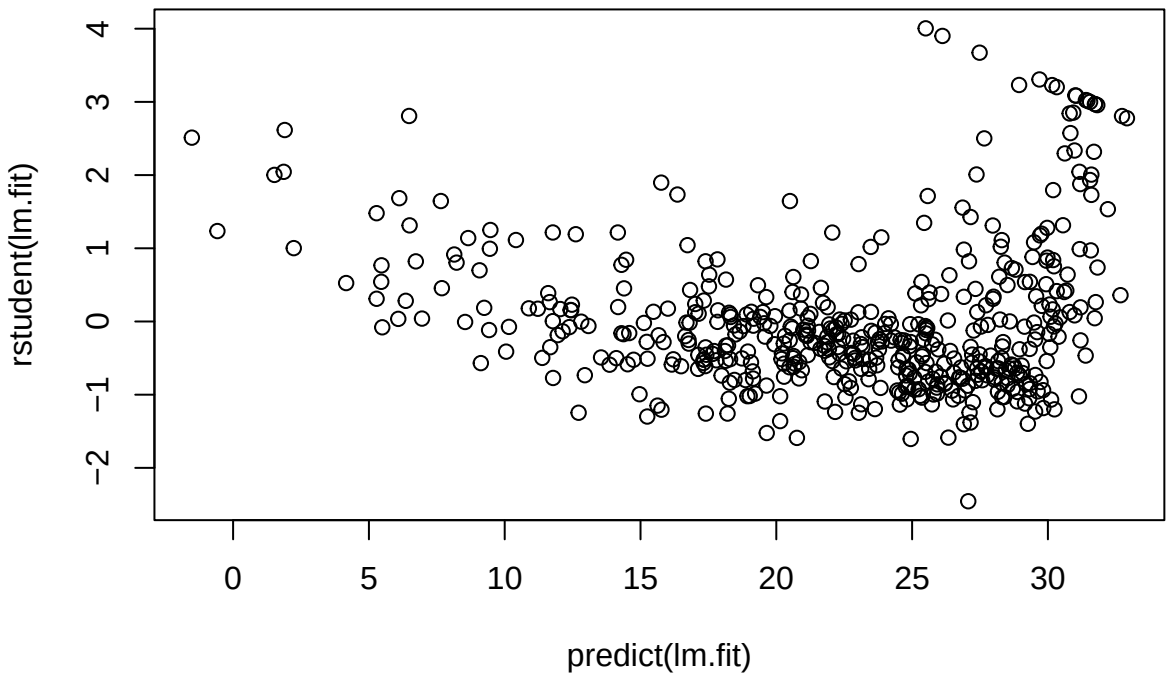


Alternatively, we can compute the residuals from a linear regression fit using the residuals() function. The function rstudent() will return the studentized residuals, and we can use this function to plot the residuals against the fitted values.

```
plot(predict(lm.fit), residuals(lm.fit))
```

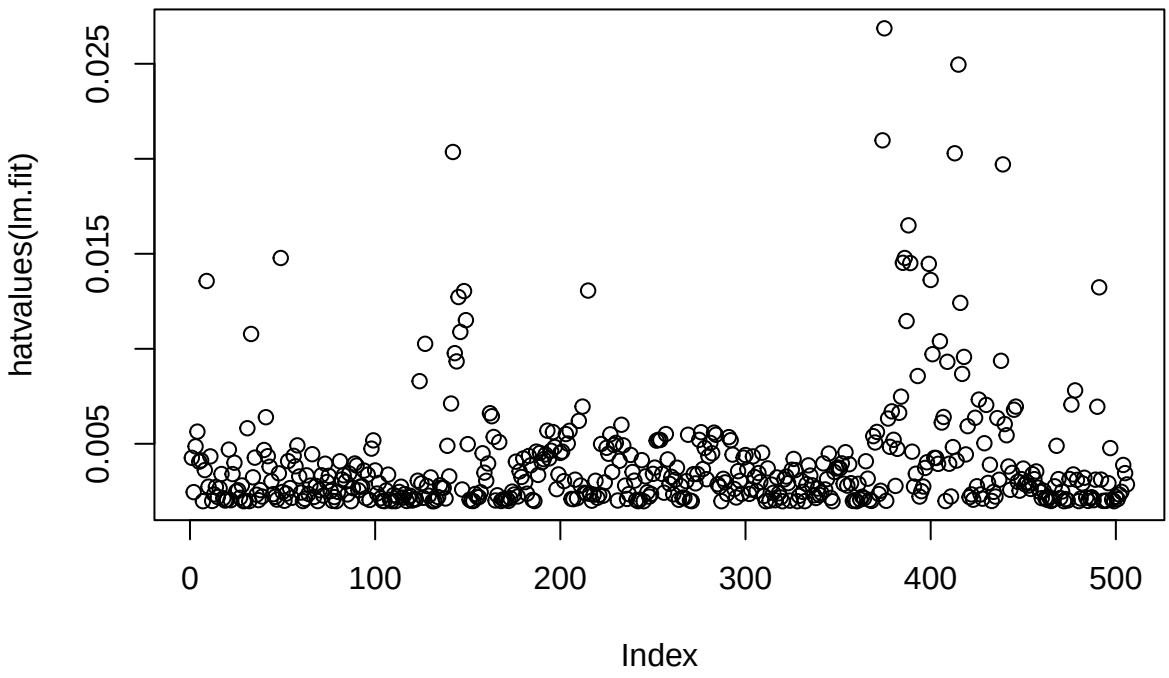



```
plot(predict(lm.fit), rstudent(lm.fit))
```



On the basis of the residual plots, there is some evidence of non-linearity. Leverage statistics can be computed for any number of predictors using the hatvalues() function.

```
plot(hatvalues(lm.fit))
```




```
which.max(hatvalues(lm.fit))
```

```
## 375
## 375
```

The which.max() function identifies the index of the largest element of a vector. In this case, it tells us which observation has the largest leverage statistic.

3.6.3 Multiple Linear Regression

In order to fit a multiple linear regression model using least squares, we again use the lm() function. The syntax lm(y ~ x1 + x2 + x3) is used to fit a model with three predictors, x1, x2, and x3. The summary() function now outputs the regression coefficients for all the predictors.

```
lm.fit <- lm(medv ~ lstat + age , data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

The Boston data set contains 12 variables, and so it would be cumbersome to have to type all of these in order to perform a regression using all of the predictors. Instead, we can use the following short-hand:

```
m.fit <- lm(medv ~ ., data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = medv ~ lstat + age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.981  -3.978  -1.283   1.968  23.158
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.22276    0.73085  45.458  < 2e-16 ***
## lstat       -1.03207    0.04819 -21.416  < 2e-16 ***
## age          0.03454    0.01223   2.826  0.00491 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.173 on 503 degrees of freedom
## Multiple R-squared:  0.5513, Adjusted R-squared:  0.5495
## F-statistic:   309 on 2 and 503 DF,  p-value: < 2.2e-16
```

We can access the individual components of a summary object by name (type ?summary.lm to see what is available). Hence summary(lm.fit)$r.sqgivesustheR2, andsummary(lm.fit)$sigma gives us the RSE. The vif() function, part of the car package, can be used to compute variance inflation factors. Most VIF's are low to moderate for this data. The car package is not part of the base R installation so it must be downloaded the first time you use it via the install.packages() function in R.

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.3.3
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.3.3
```

```
vif(lm.fit)
```

```
##    lstat      age
## 1.569395 1.569395
```

What if we would like to perform a regression using all of the variables but one? For example, in the above regression output, age has a high p-value. So we may wish to run a regression excluding this predictor. The following syntax results in a regression using all predictors except age.

```
lm.fit1 <- lm(medv ~ . - age , data = Boston)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1851  -2.7330  -0.6116   1.8555  26.3838
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  41.525128   4.919684   8.441 3.52e-16 ***
## crim         -0.121426   0.032969  -3.683 0.000256 ***
## zn            0.046512   0.013766   3.379 0.000785 ***
## indus         0.013451   0.062086   0.217 0.828577
## chas          2.852773   0.867912   3.287 0.001085 **
## nox         -18.485070   3.713714  -4.978 8.91e-07 ***
## rm            3.681070   0.411230   8.951  < 2e-16 ***
## dis          -1.506777   0.192570  -7.825 3.12e-14 ***
## rad           0.287940   0.066627   4.322 1.87e-05 ***
## tax          -0.012653   0.003796  -3.333 0.000923 ***
## ptratio      -0.934649   0.131653  -7.099 4.39e-12 ***
## lstat        -0.547409   0.047669 -11.483  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.794 on 494 degrees of freedom
## Multiple R-squared:  0.7343, Adjusted R-squared:  0.7284
## F-statistic: 124.1 on 11 and 494 DF,  p-value: < 2.2e-16
```

Alternatively, the update() function can be used.

```
lm.fit1 <- update(lm.fit , ~ . - age)
```

3.6.4 Interaction Terms

It is easy to include interaction terms in a linear model using the lm() func- tion. The syntax lstat:age tells R to include an interaction term between lstat and age. The syntax lstat * age simultaneously includes lstat, age, and the interaction term lstat×age as predictors; it is a shorthand for lstat + age + lstat:age.

```r
summary(lm(medv ~ lstat * age , data = Boston))
```

```
##
## Call:
## lm(formula = medv ~ lstat * age, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.806  -4.045  -1.333   2.085  27.552
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 36.0885359  1.4698355  24.553  < 2e-16 ***
## lstat       -1.3921168  0.1674555  -8.313 8.78e-16 ***
## age         -0.0007209  0.0198792  -0.036   0.9711
## lstat:age    0.0041560  0.0018518   2.244   0.0252 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.149 on 502 degrees of freedom
## Multiple R-squared:  0.5557, Adjusted R-squared:  0.5531
## F-statistic: 209.3 on 3 and 502 DF,  p-value: < 2.2e-16
```

Exercises

1. Describe the null hypotheses to which the p-values given in Table 3.4 correspond. Explain what conclusions you can draw based on these p-values. Your explanation should be phrased in terms of sales, TV, radio, and newspaper, rather than in terms of the coefficients of the linear model.

The null hypotheses associated with table 3.4 are that advertising budgets of "TV", "radio" or "newspaper" do not have an effect on sales. More precisely $H_0^{(1)}$:β_1=0, $H_0^{(2)}$:β_2=0 and $H_0^{(3)}$:β_3=0. The corresponding p-values are highly significant for "TV" and "radio" and not significant for "newspaper"; so we reject $H_0^{(1)}$ and $H_0^{(2)}$ and we do not reject $H_0^{(3)}$. We may conclude that newspaper advertising budget do not affect sales.

3. Suppose we have a data set with five predictors, X1 = GPA, X2 = IQ, X3 = Level (1 for College and 0 for High School), X4 = Interac- tion between GPA and IQ, and X5 = Interaction between GPA and Level. The response is starting salary after graduation (in thousands of dollars). Suppose we use least squares to fit the model, and get $\hat{\beta}_0 = 50$, $\hat{\beta}_1 = 20$, $\hat{\beta}_2 = 0.07$, $\hat{\beta}_3 = 35$, $\hat{\beta}_4 = 0.01$, $\hat{\beta}_5 = -10$.

(a) Which answer is correct, and why?

i. For a fixed value of IQ and GPA, high school graduates earn more, on average, than college graduates.

ii. For a fixed value of IQ and GPA, college graduates earn more, on average, than high school graduates.

iii. For a fixed value of IQ and GPA, high school graduates earn more, on average, than college graduates provided that the GPA is high enough.

iv. For a fixed value of IQ and GPA, college graduates earn more, on average, than high school graduates provided that the GPA is high enough.

(b) Predict the salary of a college graduate with IQ of 110 and a GPA of 4.0.

(c) True or false: Since the coefficient for the GPA/IQ interaction term is very small, there is very little evidence of an interaction effect. Justify your answer

9. This question involves the use of multiple linear regression on the Auto data set.

(a) Produce a scatterplot matrix which includes all of the variables in the data set.

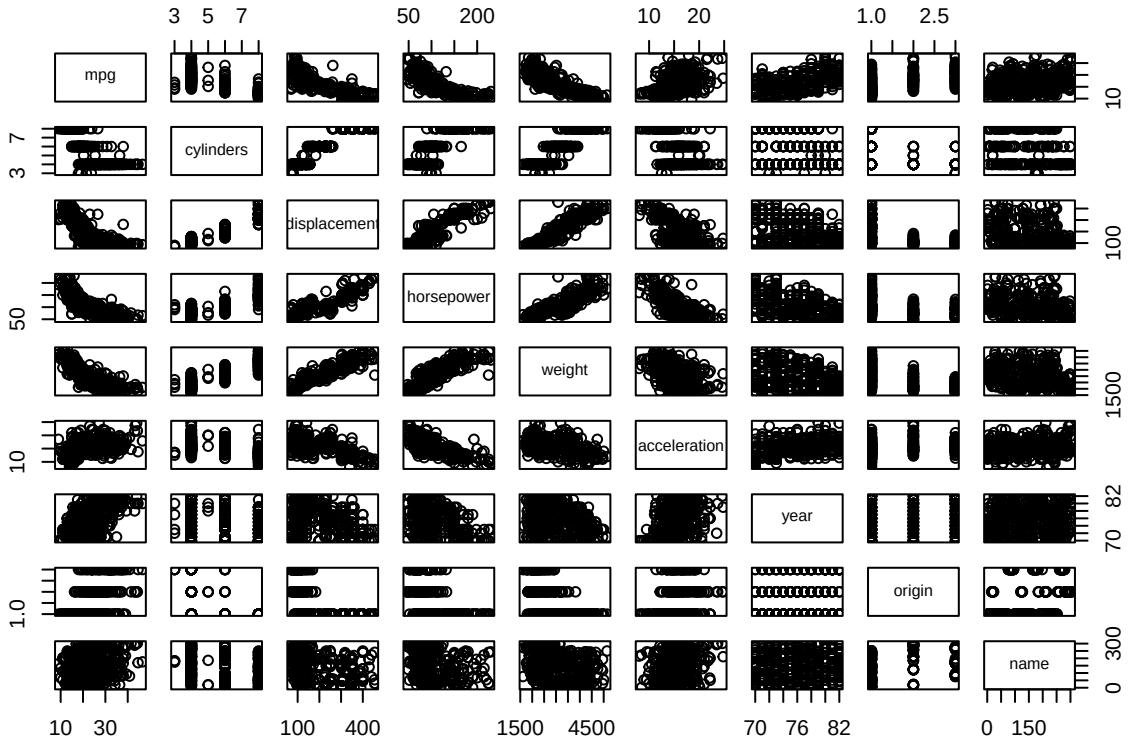
```
library(ISLR) # Auto dataset is in ISLR library
```

```
## Warning: package 'ISLR' was built under R version 4.3.3
```

```
##
## Attaching package: 'ISLR'
```

```
## The following objects are masked from 'package:ISLR2':
##
##     Auto, Credit
```

```
pairs(Auto)
```



(b) Compute the matrix of correlations between the variables using the function cor(). You will need to exclude the name variable, cor() which is qualitative.

```r
names(Auto)
```

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

```r
cor(Auto[1:8]) # excluding name variable
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
##              acceleration       year     origin
## mpg             0.4233285  0.5805410  0.5652088
## cylinders      -0.5046834 -0.3456474 -0.5689316
## displacement   -0.5438005 -0.3698552 -0.6145351
## horsepower     -0.6891955 -0.4163615 -0.4551715
## weight         -0.4168392 -0.3091199 -0.5850054
## acceleration    1.0000000  0.2903161  0.2127458
## year            0.2903161  1.0000000  0.1815277
## origin          0.2127458  0.1815277  1.0000000
```

(c) Use the lm() function to perform a multiple linear regression with mpg as the response and all other variables except name as the predictors. Use the summary() function to print the results. Comment on the output. For instance:

i. Is there a relationship between the predictors and the response?

```r
fit2 <- lm(mpg ~ . - name, data = Auto)
summary(fit2)
```

```
##
## Call:
## lm(formula = mpg ~ . - name, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
## cylinders     -0.493376   0.323282  -1.526  0.12780
## displacement   0.019896   0.007515   2.647  0.00844 **
## horsepower    -0.016951   0.013787  -1.230  0.21963
## weight        -0.006474   0.000652  -9.929  < 2e-16 ***
## acceleration   0.080576   0.098845   0.815  0.41548
```

```
## year           0.750773   0.050973  14.729  < 2e-16 ***
## origin         1.426141   0.278136   5.127 4.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

We can answer this question by again testing the hypothesis H_0:β_i=0 $\forall_i$. The p-value corresponding to the F-statistic is 2.2×10^{-16}, this indicates a clear evidence of a relationship between "mpg" and the other predictors.

ii. Which predictors appear to have a statistically significant relationship to the response?

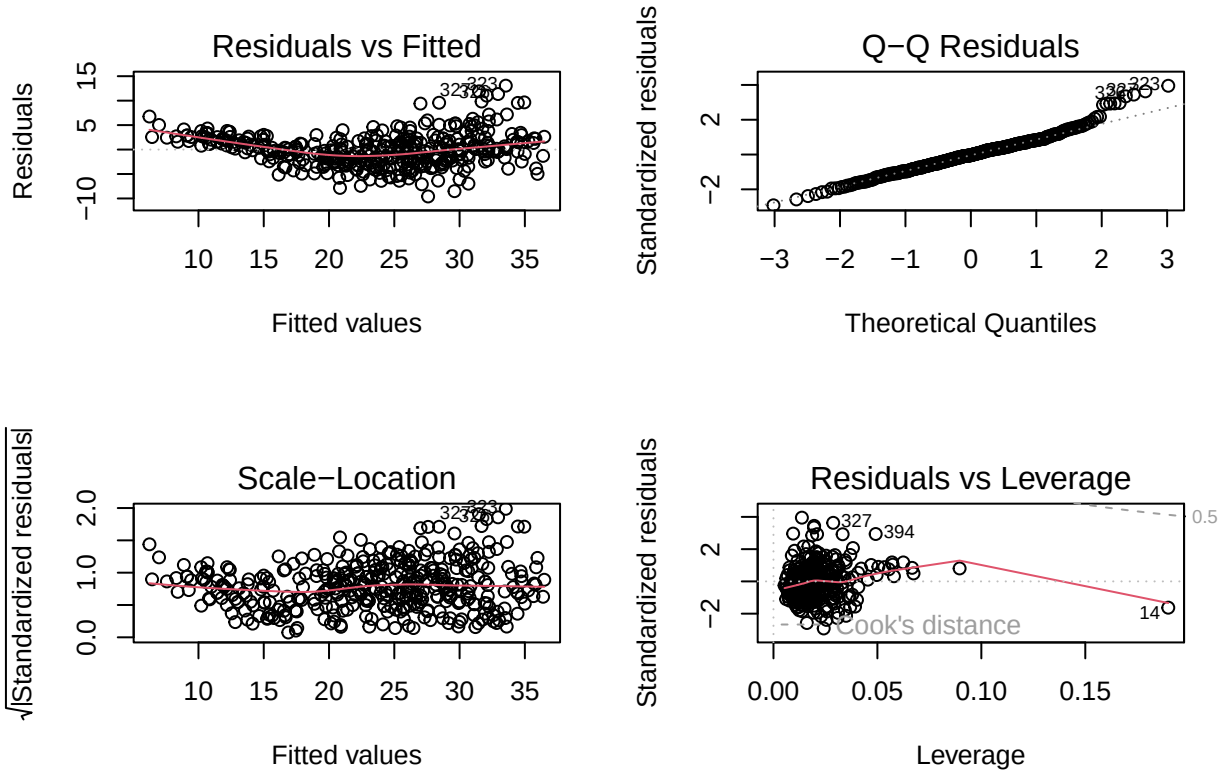
We can answer this question by checking the p-values associated with each predictor's t-statistic. We may conclude that all predictors are statistically significant except "cylinders", "horsepower" and "acceleration". (Note: the significance code for these are 0.1)

iii. What does the coefficient for the year variable suggest?

The coefficient ot the "year" variable suggests that the average effect of an increase of 1 year is an increase of 0.7507727 in "mpg" (all other predictors remaining constant). In other words, cars become more fuel efficient every year by almost 1 mpg / year.

(d) Use the plot() function to produce diagnostic plots of the linear regression fit. Comment on any problems you see with the fit. Do the residual plots suggest any unusually large outliers? Does the leverage plot identify any observations with unusually high leverage?

```
par(mfrow = c(2, 2))
plot(fit2)
```



As before, the plot of residuals versus fitted values indicates the presence of mild non linearity in the data. The plot of standardized residuals versus leverage indicates the presence of a few outliers (higher than 2 or lower than -2) and one high leverage point (point 14).

(e) Use the * and : symbols to fit linear regression models with interaction effects. Do any interactions appear to be statistically significant?

From the correlation matrix, we obtained the two highest correlated pairs and used them in picking interaction effects.

```
fit3 <- lm(mpg ~ cylinders * displacement+displacement * weight, data = Auto[, 1:8])
summary(fit3)
```

```
##
## Call:
## lm(formula = mpg ~ cylinders * displacement + displacement *
##     weight, data = Auto[, 1:8])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.2934  -2.5184  -0.3476   1.8399  17.7723
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 5.262e+01  2.237e+00  23.519  < 2e-16 ***
```

```
## cylinders               7.606e-01  7.669e-01   0.992    0.322
## displacement           -7.351e-02  1.669e-02  -4.403 1.38e-05 ***
## weight                 -9.888e-03  1.329e-03  -7.438 6.69e-13 ***
## cylinders:displacement -2.986e-03  3.426e-03  -0.872    0.384
## displacement:weight     2.128e-05  5.002e-06   4.254 2.64e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.103 on 386 degrees of freedom
## Multiple R-squared:  0.7272, Adjusted R-squared:  0.7237
## F-statistic: 205.8 on 5 and 386 DF,  p-value: < 2.2e-16
```

From the p-values, we can see that the interaction between displacement and weight is statistically signifcant, while the interactiion between cylinders and displacement is not