Lab 02 - Linear Regression

Ken Ye

09/06/2023

Instructions

- Please work through all sections at the beginning of the lab before moving onto the problems at the end
- Feel free to work in groups
- Work in your groups for about 1 hour, then we will present our results at the end of lab
- Submit the Rmd and pdf files under the assignment section of Sakai

1. Libraries

The `library()` function is used to access functionality that is provided by R packages, but is not included in base R. `install.packages()` can be used to install new packages. Run this command from the console.

```
# install.packages("ISLR")
```

First, load the packages `MASS` and `ISLR` that will be used throughout the lab.

```
library(MASS)
library(ISLR)
```

2. Simple Linear Regression

This lab will be using the `Boston` data from the `MASS` package. Load this data using the `attach()` function:

```
attach(Boston)
```

The functions `head()` and `names()` can be used to explore the data.

```
head(Boston)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
##   medv
```

```
## 1 24.0
## 2 21.6
## 3 34.7
## 4 33.4
## 5 36.2
## 6 28.7
```

```r
names(Boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

The `MASS` library contains the Boston data set, which records medv (median house value) for 506 neighborhoods around Boston. We will seek to predict medv using 13 predictors such as rm (average number of rooms per house), age (average age of houses), and lstat (percent of households with low socioeconomic status). To find out more about the data set, we can type `?Boston`.

```r
?Boston
```

We'll start with a fitting a simple linear model using the `lm()` function. Instead of attaching the `Boston` dataset, we also can specify the data from the `lm()` function. In the `lm()` function, the first variable is the response variable and the variables to the right of the ~ symbol are the predictor variable(s).

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

There are several ways that we can examine the model results. First, we can just call the name of the `lm()` model for a brief summary.

```r
lm.fit
```

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

We can also use the `names()` function to list all of the names of variables in the `lm.fit` model:

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

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

The `summary()` function gives a more extensive overview of the model fit:

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

```
##
## Call:
## lm(formula = medv ~ lstat, data = Boston)
##
## 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
```

The coefficients of the linear regression model can be extracted using the `coef()` function and the confidence interval(s) with the `confint()` function.

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

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

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

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

We can use the `predict()` function to obtain prediction intervals or confidence intervals for a given value of the predictor variable, `lstat`. Note that when using the predict function, the column names and format of the new points at which to predict needs to be the same as the original data frame used to fit the `lm()` model. If you encounter errors using the `predict()` function, this is a good first thing to check.

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

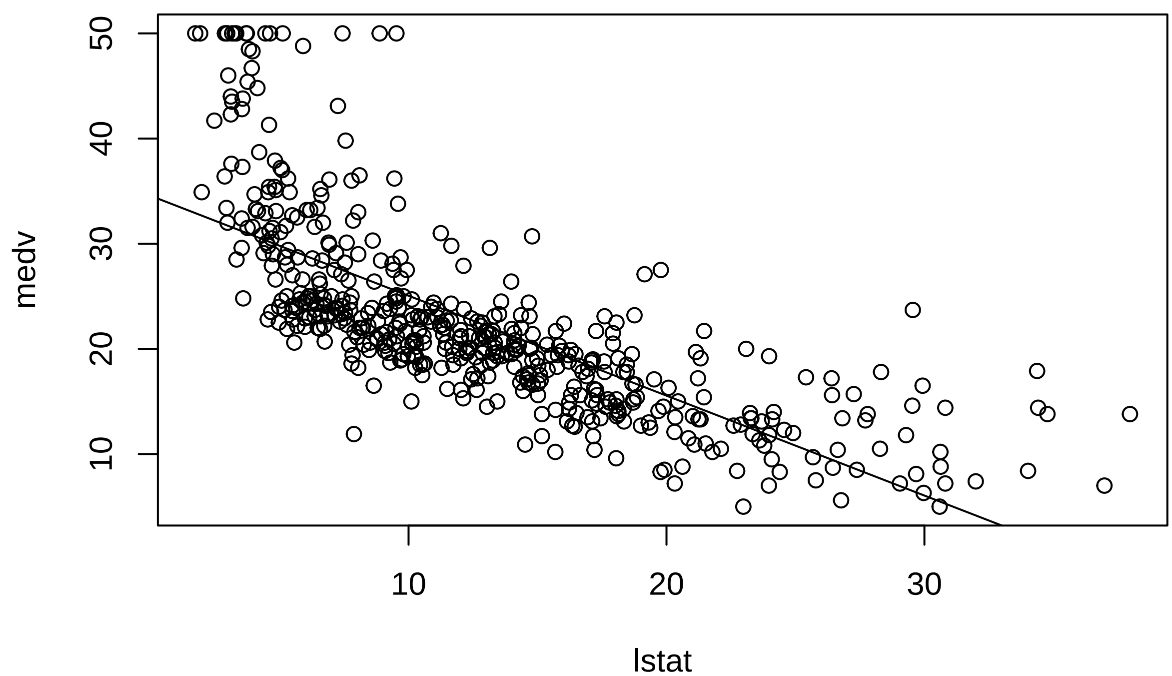
```
##        fit      lwr      upr
## 1 29.80359 29.00741 30.59978
## 2 25.05335 24.47413 25.63256
## 3 20.30310 19.73159 20.87461
```

```r
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
```

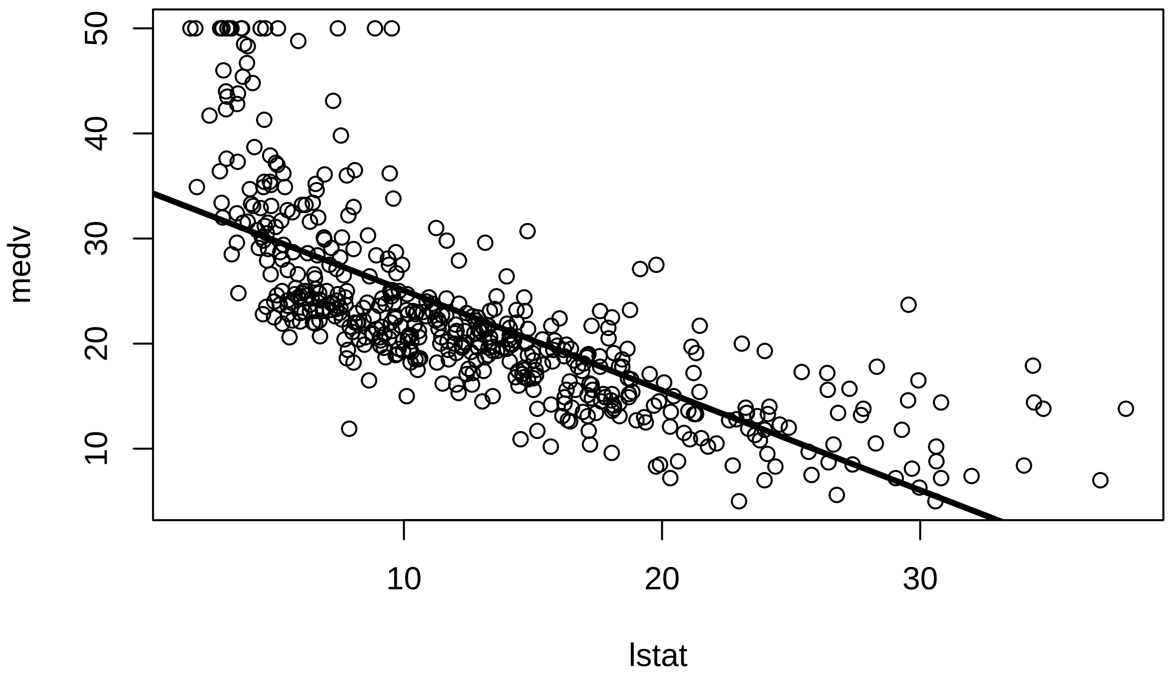
We can plot the variables `lstat` and `medv` using the `plot()` function and overlay the regression line found using `lm()` with the `abline()` function.

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

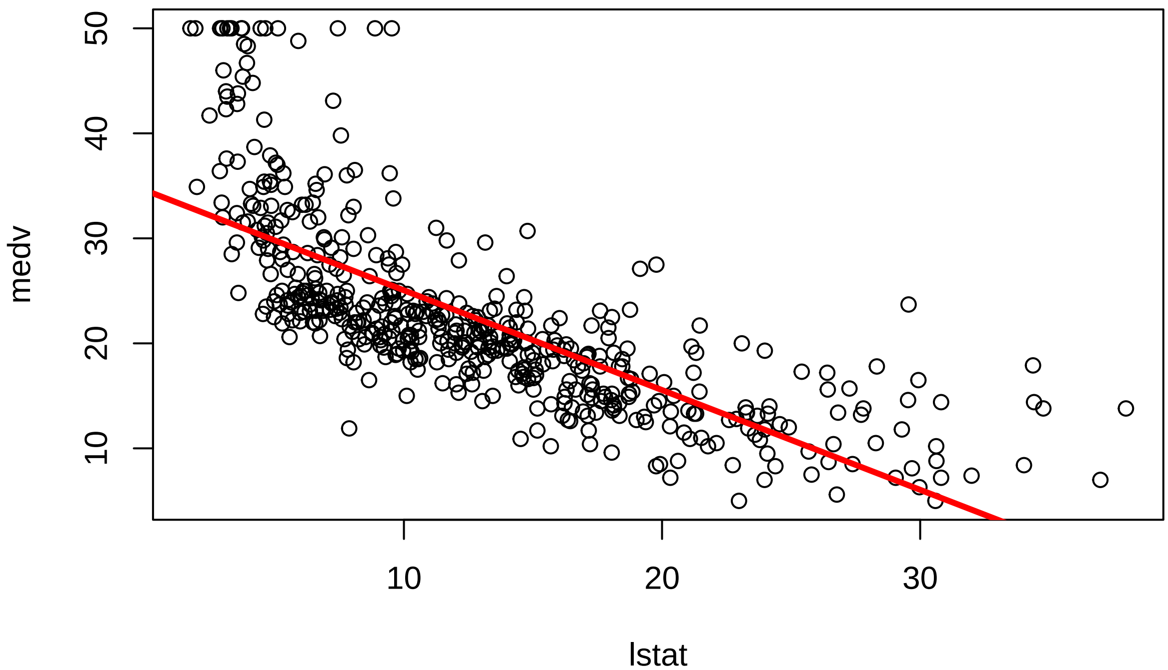


We can experiment with different options for `abline()` by changing the line width and color in `abline()`.

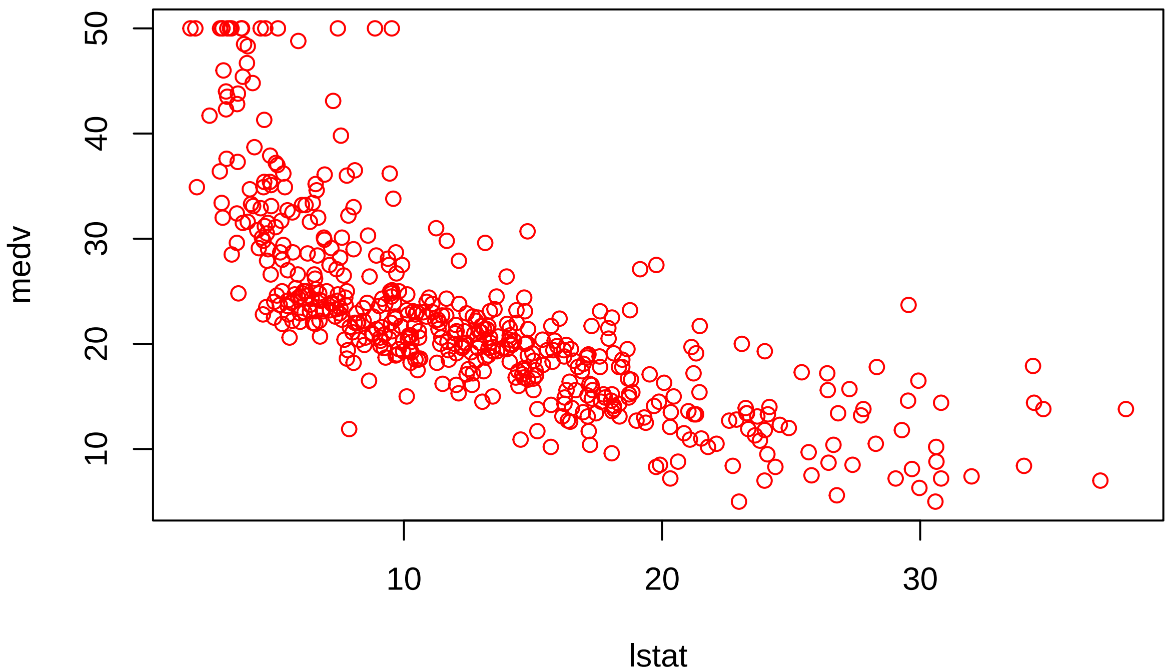
```r
plot(lstat, medv)
abline(lm.fit, lwd = 3)
```




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

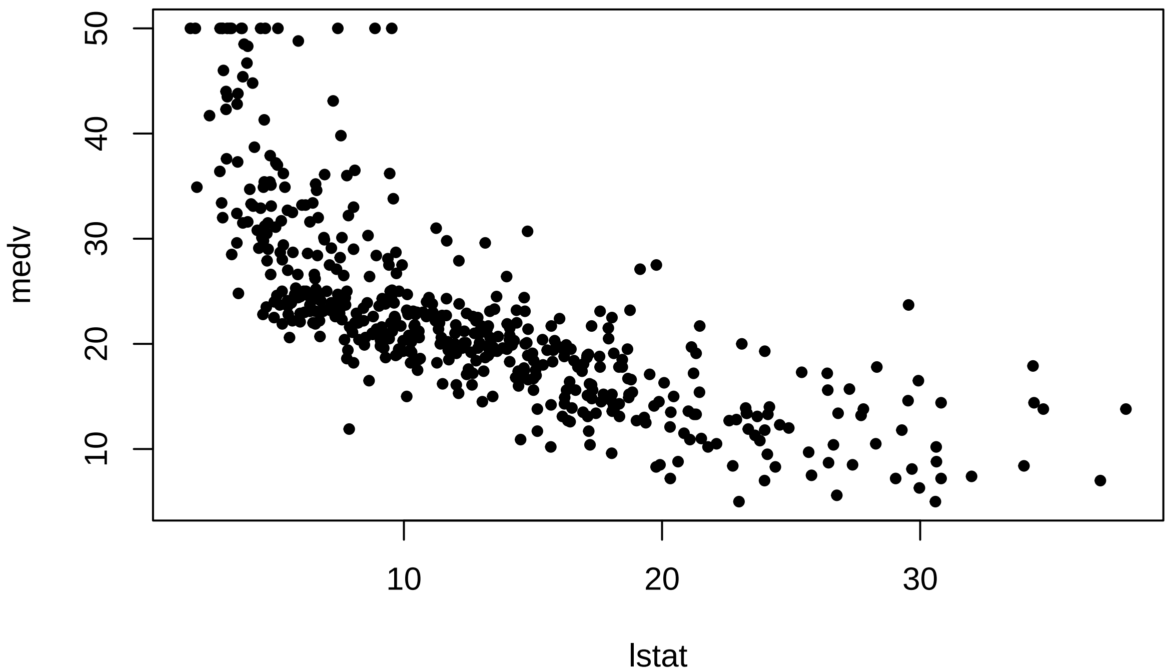



```
plot(lstat, medv, col = "red")
```

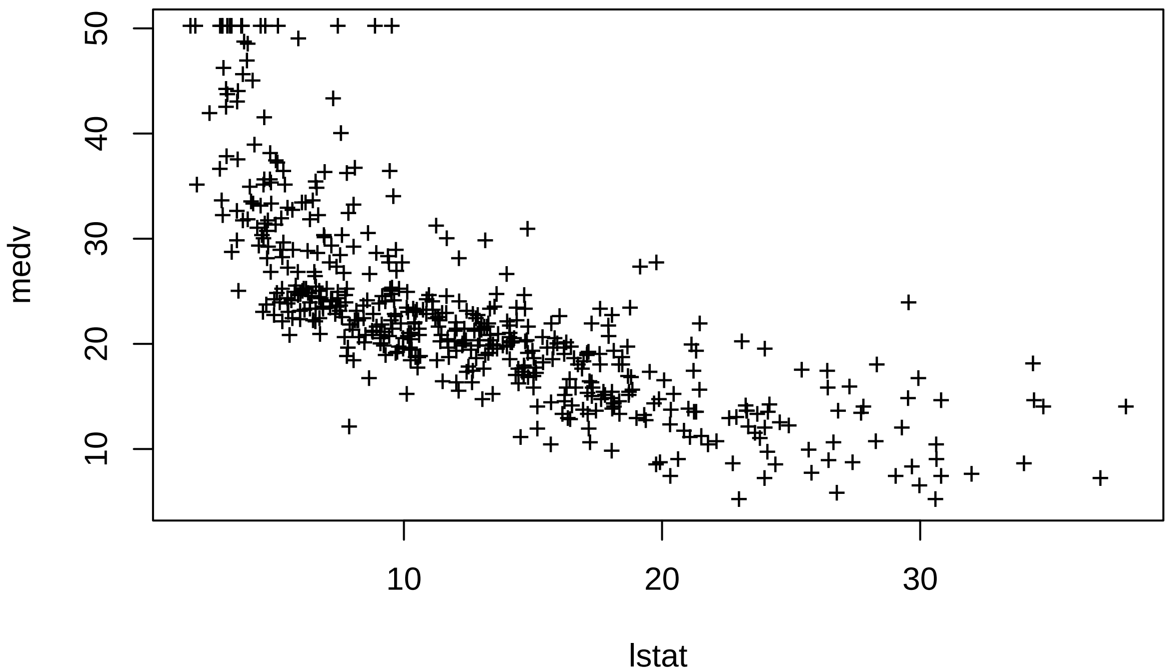


The `pch` argument in `plot()` changes the shape/type of the points that are plotted.

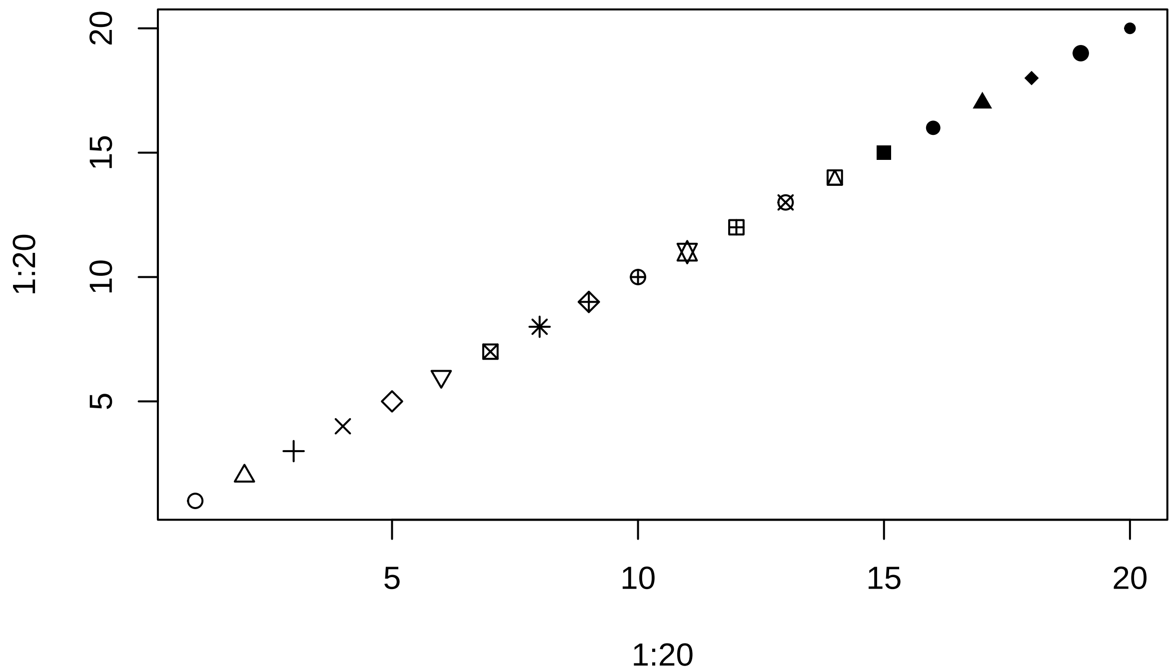
```
plot(lstat, medv, pch = 20)
```




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

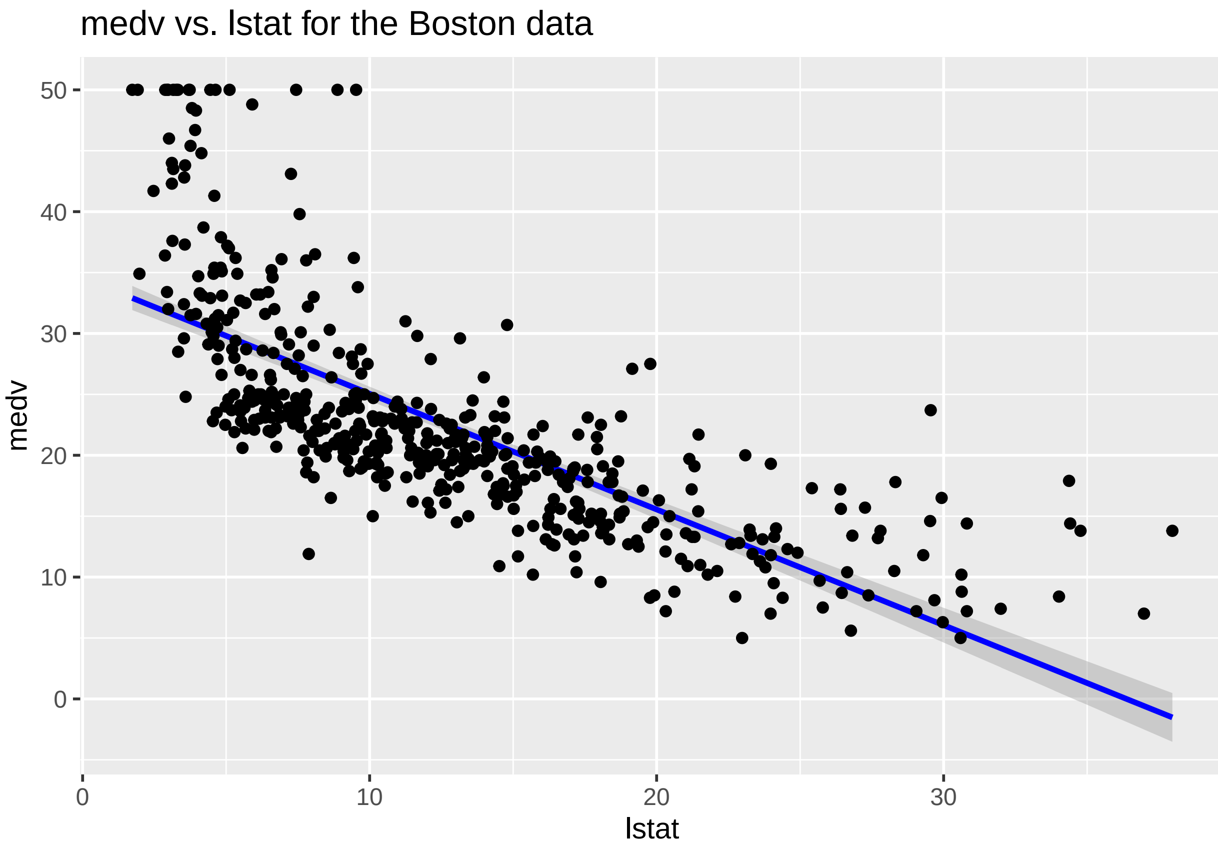



```
plot(1:20, 1:20, pch = 1:20)
```



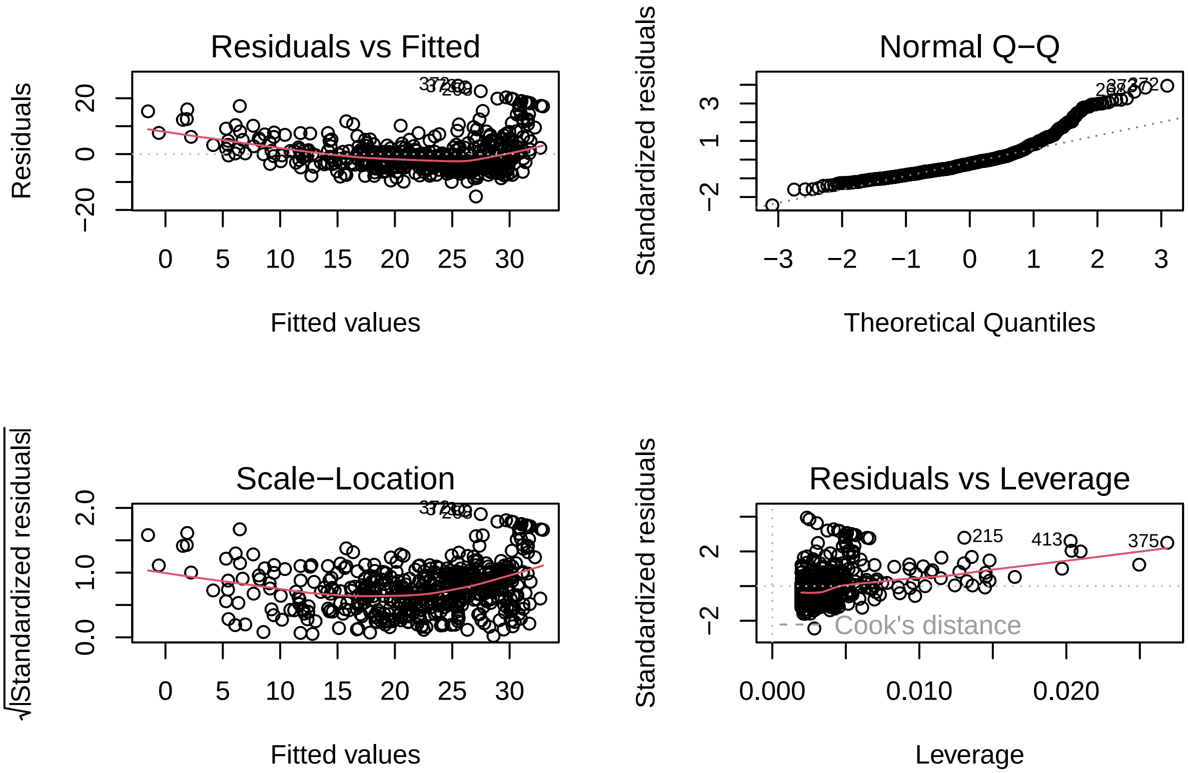
Optional: We can make a similar plot using `ggplot`, where we fit the linear regression model using `ggplot()`.

```
library(ggplot2)
ggplot(Boston, aes(y = medv, x = lstat)) +
    geom_smooth(method = "lm", formula = y ~ x, colour = "blue") +
    geom_point() +
  ggtitle("medv vs. lstat for the Boston data")
```



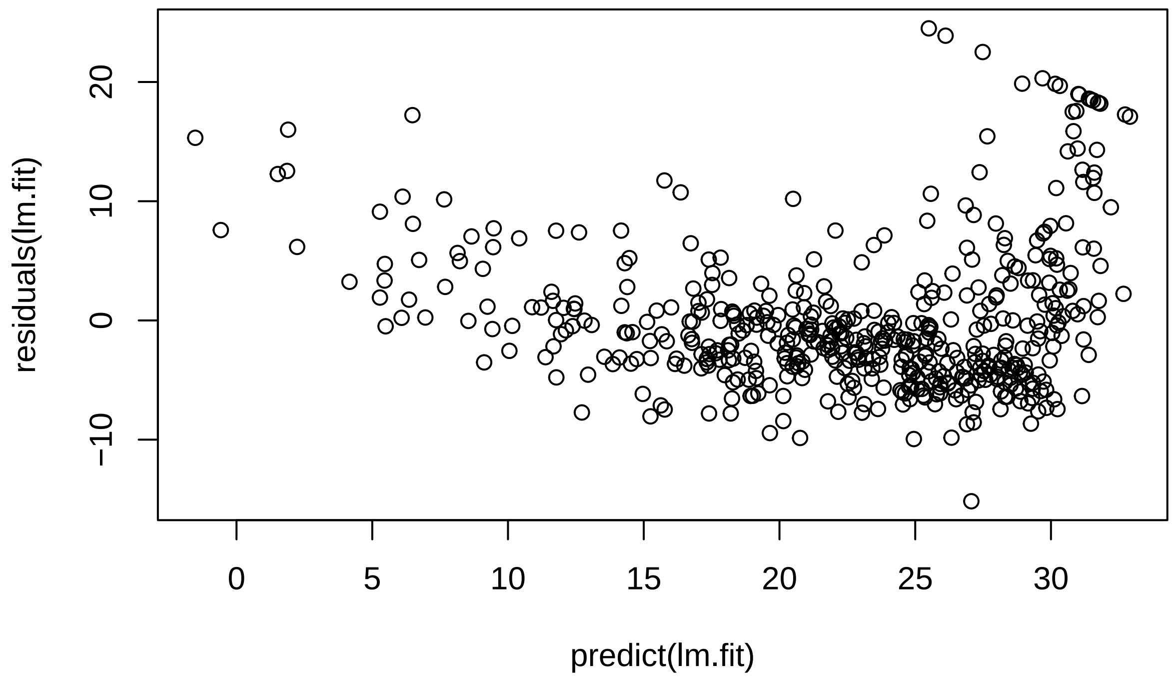
The `par()` function can be used to create a grid of multiple subplots.

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

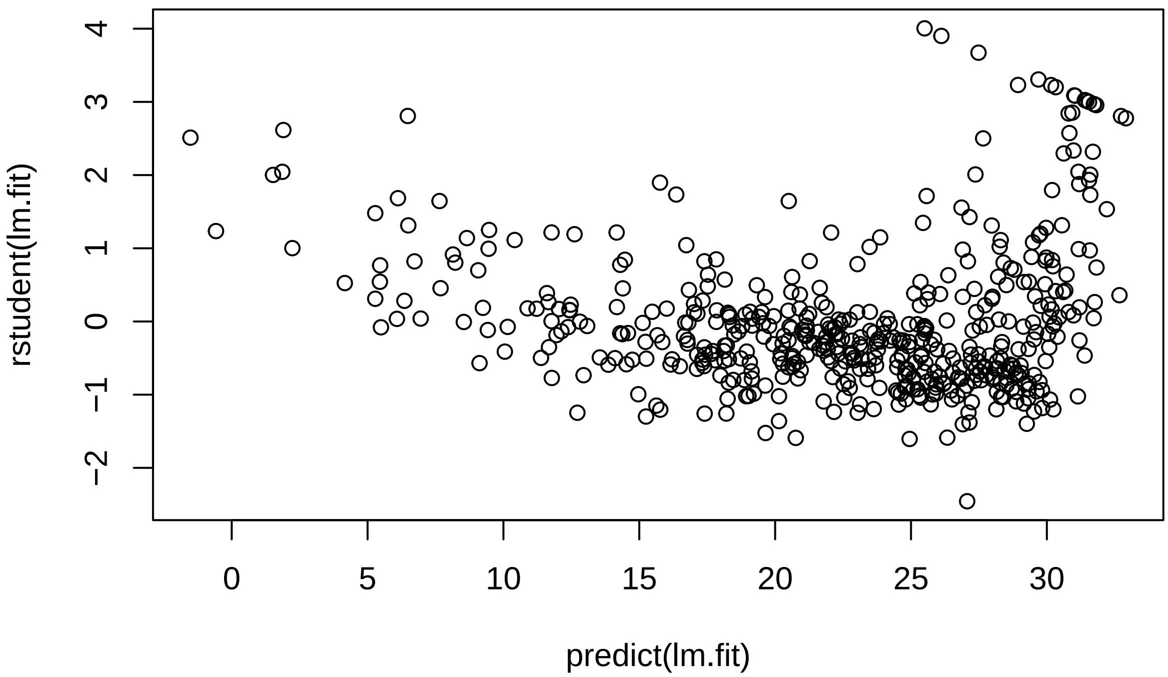


We can use the `residuals()` and `rstudent()` functions to extract the residuals and studentized residuals, respectively, from the linear model and plot them along with the predicted values.

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

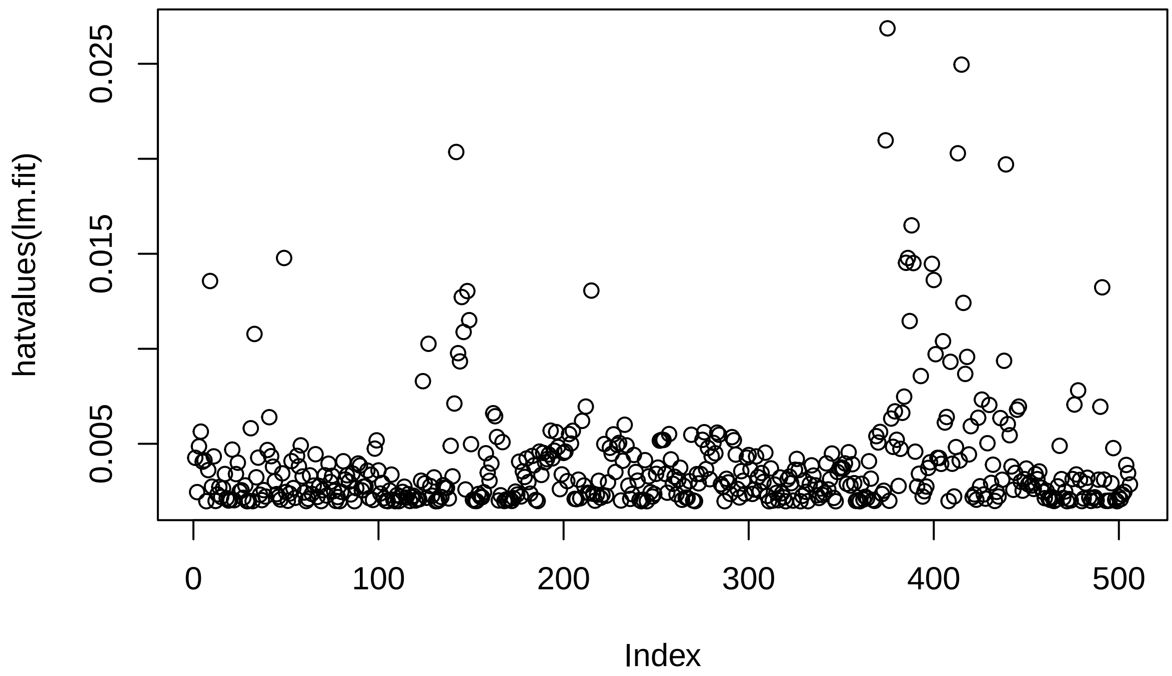



```
plot(predict(lm.fit), rstudent(lm.fit))
```



Additionally, we can compute the influence matrix for the predictors using the `hatvalues()` function.

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




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

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

3. Multiple Linear Regression

The `lm()` function can also fit multiple regression models. In this section, we will use `age` and `lstata` as predictors of the response variable `medv`.

```
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
```

In the lm() formula, a dot . can be used to include all variables in the Boston data as predictors.

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

```
##
## Call:
## lm(formula = medv ~ ., data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.595  -2.730  -0.518   1.777  26.199
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
## zn           4.642e-02  1.373e-02   3.382 0.000778 ***
## indus        2.056e-02  6.150e-02   0.334 0.738288
## chas         2.687e+00  8.616e-01   3.118 0.001925 **
## nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
## rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
## age          6.922e-04  1.321e-02   0.052 0.958229
## dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
## rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
## tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
## ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
## black        9.312e-03  2.686e-03   3.467 0.000573 ***
## lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

The Variance Inflation Factors (VIF) can be calculated using the vif() function from the car package (Companion to Applied Regression). The car package is included in the ISLR package and should already be loaded.

```
library(car)
```

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

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

```
##     crim       zn    indus     chas      nox       rm      age      dis
## 1.792192 2.298758 3.991596 1.073995 4.393720 1.933744 3.100826 3.955945
##      rad      tax  ptratio    black    lstat
## 7.484496 9.008554 1.799084 1.348521 2.941491
```

If we want to exclude specific variables from the list of predictors, we can use the `-` notation. In the following example, all predictor variables but `age` are included in the model.

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

```
##
## Call:
## lm(formula = medv ~ . - age, data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.6054  -2.7313  -0.5188   1.7601  26.2243
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  36.436927   5.080119   7.172 2.72e-12 ***
## crim         -0.108006   0.032832  -3.290 0.001075 **
## zn            0.046334   0.013613   3.404 0.000719 ***
## indus         0.020562   0.061433   0.335 0.737989
## chas          2.689026   0.859598   3.128 0.001863 **
## nox         -17.713540   3.679308  -4.814 1.97e-06 ***
## rm            3.814394   0.408480   9.338  < 2e-16 ***
## dis          -1.478612   0.190611  -7.757 5.03e-14 ***
## rad           0.305786   0.066089   4.627 4.75e-06 ***
## tax          -0.012329   0.003755  -3.283 0.001099 **
## ptratio      -0.952211   0.130294  -7.308 1.10e-12 ***
## black         0.009321   0.002678   3.481 0.000544 ***
## lstat        -0.523852   0.047625 -10.999  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.74 on 493 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7343
## F-statistic: 117.3 on 12 and 493 DF,  p-value: < 2.2e-16
```

Including `-1` excludes the intercept from the model.

```r
lm.fit1 <- lm(medv ~ . - 1, data = Boston)
summary(lm.fit1)
```

```
##
## Call:
## lm(formula = medv ~ . - 1, data = Boston)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21.1100  -2.5630  -0.5529   1.6546  30.7254
##
## Coefficients:
##          Estimate Std. Error t value Pr(>|t|)
## crim    -0.092897   0.034421  -2.699 0.007197 **
## zn       0.048715   0.014403   3.382 0.000776 ***
## indus   -0.004060   0.064440  -0.063 0.949789
## chas     2.853999   0.903913   3.157 0.001689 **
## nox     -2.868436   3.358732  -0.854 0.393507
## rm       5.928148   0.309109  19.178  < 2e-16 ***
## age     -0.007269   0.013815  -0.526 0.598979
## dis     -0.968514   0.195630  -4.951 1.02e-06 ***
## rad      0.171151   0.066752   2.564 0.010644 *
## tax     -0.009396   0.003923  -2.395 0.016988 *
## ptratio -0.392191   0.109869  -3.570 0.000393 ***
## black    0.014906   0.002697   5.528 5.27e-08 ***
## lstat   -0.416304   0.050786  -8.197 2.14e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.98 on 493 degrees of freedom
## Multiple R-squared:  0.9592, Adjusted R-squared:  0.9581
## F-statistic: 891.3 on 13 and 493 DF,  p-value: < 2.2e-16
```

The `update()` function can be used to specify a new formula for an existing model.

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

4. Interaction Terms

There are two ways to include interaction terms in the model, `:` and `*`. The `:` symbol only includes the interaction term between the two variables, while the `*` symbol includes the variables themselves, as well as the interaction terms. This means that `lstat*age` is equivalent to `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
```

A simple way to include all interaction terms is the syntax `.^2`.

```r
summary(lm(medv ~.^2, data = Boston))
```

```
##
## Call:
## lm(formula = medv ~ .^2, data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9374 -1.5344 -0.1068  1.2973 17.8500
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -1.579e+02  6.800e+01  -2.323 0.020683 *
## crim          -1.707e+01  6.554e+00  -2.605 0.009526 **
## zn            -7.529e-02  4.580e-01  -0.164 0.869508
## indus         -2.819e+00  1.696e+00  -1.663 0.097111 .
## chas           4.451e+01  1.952e+01   2.280 0.023123 *
## nox            2.006e+01  7.516e+01   0.267 0.789717
## rm             2.527e+01  5.699e+00   4.435 1.18e-05 ***
## age            1.263e+00  2.728e-01   4.630 4.90e-06 ***
## dis           -1.698e+00  4.604e+00  -0.369 0.712395
## rad            1.861e+00  2.464e+00   0.755 0.450532
## tax            3.670e-02  1.440e-01   0.255 0.798978
## ptratio        2.725e+00  2.850e+00   0.956 0.339567
## black          9.942e-02  7.468e-02   1.331 0.183833
## lstat          1.656e+00  8.533e-01   1.940 0.053032 .
## crim:zn        4.144e-01  1.804e-01   2.297 0.022128 *
## crim:indus    -4.693e-02  4.480e-01  -0.105 0.916621
## crim:chas      2.428e+00  5.710e-01   4.251 2.63e-05 ***
## crim:nox      -1.108e+00  9.285e-01  -1.193 0.233425
## crim:rm        2.163e-01  4.907e-02   4.409 1.33e-05 ***
## crim:age      -3.083e-03  3.781e-03  -0.815 0.415315
## crim:dis      -1.903e-01  1.060e-01  -1.795 0.073307 .
## crim:rad      -6.584e-01  5.815e-01  -1.132 0.258198
## crim:tax       3.479e-02  4.287e-02   0.812 0.417453
## crim:ptratio   4.915e-01  3.328e-01   1.477 0.140476
## crim:black    -4.612e-04  1.793e-04  -2.572 0.010451 *
## crim:lstat     2.964e-02  6.544e-03   4.530 7.72e-06 ***
## zn:indus      -6.731e-04  4.651e-03  -0.145 0.885000
## zn:chas       -5.230e-02  6.450e-02  -0.811 0.417900
## zn:nox         1.998e-03  4.721e-01   0.004 0.996625
## zn:rm         -7.286e-04  2.602e-02  -0.028 0.977672
## zn:age        -1.249e-06  8.514e-04  -0.001 0.998830
```

```
## zn:dis         1.097e-02  7.550e-03   1.452 0.147121
## zn:rad        -3.200e-03  6.975e-03  -0.459 0.646591
## zn:tax         3.937e-04  1.783e-04   2.209 0.027744 *
## zn:ptratio    -4.578e-03  7.015e-03  -0.653 0.514325
## zn:black       1.159e-04  7.599e-04   0.153 0.878841
## zn:lstat      -1.064e-02  4.662e-03  -2.281 0.023040 *
## indus:chas    -3.672e-01  3.780e-01  -0.971 0.331881
## indus:nox      3.138e+00  1.449e+00   2.166 0.030855 *
## indus:rm       3.301e-01  1.327e-01   2.488 0.013257 *
## indus:age     -4.865e-04  3.659e-03  -0.133 0.894284
## indus:dis     -4.486e-02  6.312e-02  -0.711 0.477645
## indus:rad     -2.089e-02  5.020e-02  -0.416 0.677560
## indus:tax      3.129e-04  6.034e-04   0.519 0.604322
## indus:ptratio -6.011e-02  3.783e-02  -1.589 0.112820
## indus:black    1.122e-03  2.034e-03   0.552 0.581464
## indus:lstat    5.063e-03  1.523e-02   0.332 0.739789
## chas:nox      -3.272e+01  1.243e+01  -2.631 0.008820 **
## chas:rm       -5.384e+00  1.150e+00  -4.681 3.87e-06 ***
## chas:age       3.040e-02  5.840e-02   0.521 0.602982
## chas:dis       9.022e-01  1.334e+00   0.676 0.499143
## chas:rad      -7.773e-01  5.707e-01  -1.362 0.173907
## chas:tax       4.627e-02  3.645e-02   1.270 0.204930
## chas:ptratio  -6.145e-01  6.914e-01  -0.889 0.374604
## chas:black     2.500e-02  1.567e-02   1.595 0.111423
## chas:lstat    -2.980e-01  1.845e-01  -1.615 0.107008
## nox:rm         5.990e+00  5.468e+00   1.095 0.273952
## nox:age       -7.273e-01  2.340e-01  -3.108 0.002012 **
## nox:dis        5.694e+00  3.723e+00   1.529 0.126969
## nox:rad       -1.994e-01  1.897e+00  -0.105 0.916360
## nox:tax       -2.793e-02  1.312e-01  -0.213 0.831559
## nox:ptratio   -3.669e+00  3.096e+00  -1.185 0.236648
## nox:black     -1.854e-02  3.615e-02  -0.513 0.608298
## nox:lstat      1.119e+00  6.511e-01   1.719 0.086304 .
## rm:age        -6.277e-02  2.203e-02  -2.849 0.004606 **
## rm:dis         3.190e-01  3.295e-01   0.968 0.333516
## rm:rad        -8.422e-02  1.527e-01  -0.552 0.581565
## rm:tax        -2.242e-02  9.910e-03  -2.262 0.024216 *
## rm:ptratio    -4.880e-01  2.172e-01  -2.247 0.025189 *
## rm:black      -4.528e-03  3.351e-03  -1.351 0.177386
## rm:lstat      -2.968e-01  4.316e-02  -6.878 2.24e-11 ***
## age:dis       -1.678e-02  8.882e-03  -1.889 0.059589 .
## age:rad        1.442e-02  4.212e-03   3.423 0.000682 ***
## age:tax       -3.403e-04  2.187e-04  -1.556 0.120437
## age:ptratio   -7.520e-03  6.793e-03  -1.107 0.268946
## age:black     -7.029e-04  2.136e-04  -3.291 0.001083 **
## age:lstat     -6.023e-03  1.936e-03  -3.111 0.001991 **
## dis:rad       -5.580e-02  7.075e-02  -0.789 0.430678
## dis:tax       -3.882e-03  2.496e-03  -1.555 0.120623
## dis:ptratio   -4.786e-02  9.983e-02  -0.479 0.631920
## dis:black     -5.194e-03  5.541e-03  -0.937 0.349116
## dis:lstat      1.350e-01  4.866e-02   2.775 0.005774 **
## rad:tax        3.131e-05  1.446e-03   0.022 0.982729
## rad:ptratio   -4.379e-02  8.392e-02  -0.522 0.602121
## rad:black     -4.362e-04  2.518e-03  -0.173 0.862561
```

```
## rad:lstat     -2.529e-02  1.816e-02  -1.392 0.164530
## tax:ptratio    7.854e-03  2.504e-03   3.137 0.001830 **
## tax:black     -4.785e-07  1.999e-04  -0.002 0.998091
## tax:lstat     -1.403e-03  1.208e-03  -1.162 0.245940
## ptratio:black  1.203e-03  3.361e-03   0.358 0.720508
## ptratio:lstat  3.901e-03  2.985e-02   0.131 0.896068
## black:lstat   -6.118e-04  4.157e-04  -1.472 0.141837
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.852 on 414 degrees of freedom
## Multiple R-squared:  0.9212, Adjusted R-squared:  0.9039
## F-statistic: 53.18 on 91 and 414 DF,  p-value: < 2.2e-16
```

5. Non-Linear Transformations of the Predictors

Non-linear transformations of variables can be included in the `lm()` function, too. Powers of terms must be included inside the `I()` function to be treated `as is`.

```
lm.fit2 <- lm(medv ~ lstat + I(lstat^2))
summary(lm.fit2)
```

```
##
## Call:
## lm(formula = medv ~ lstat + I(lstat^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.2834  -3.8313  -0.5295   2.3095  25.4148
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 42.862007   0.872084   49.15   <2e-16 ***
## lstat       -2.332821   0.123803  -18.84   <2e-16 ***
## I(lstat^2)   0.043547   0.003745   11.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.524 on 503 degrees of freedom
## Multiple R-squared:  0.6407, Adjusted R-squared:  0.6393
## F-statistic: 448.5 on 2 and 503 DF,  p-value: < 2.2e-16
```

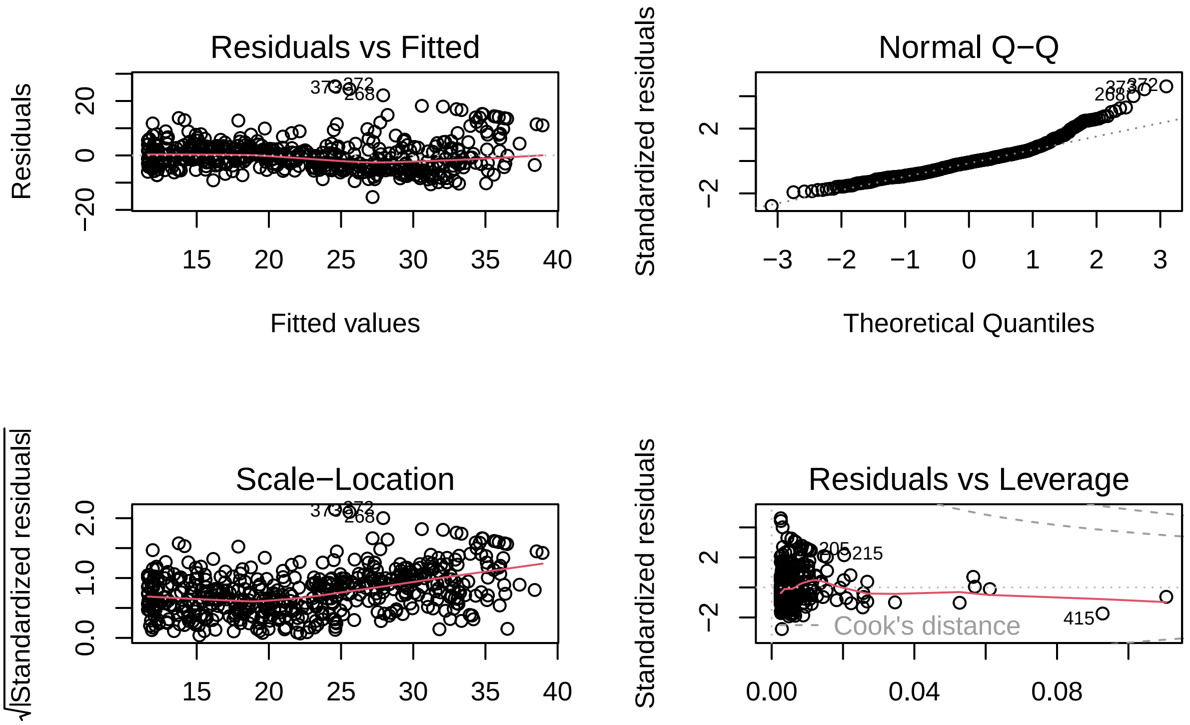
We can also examine the analysis of variance (ANOVA) for one or more models with the `anova()` function.

```
lm.fit <- lm(medv ~ lstat)
anova(lm.fit, lm.fit2)
```

```
## Analysis of Variance Table
##
## Model 1: medv ~ lstat
## Model 2: medv ~ lstat + I(lstat^2)
##   Res.Df   RSS Df Sum of Sq     F    Pr(>F)
```

```
## 1    504 19472
## 2    503 15347  1    4125.1 135.2 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
par(mfrow = c(2, 2))
plot(lm.fit2)
```



The poly() function can be used to include all polynomial terms up to the specified degree.

```r
lm.fit5 <- lm(medv ~ poly(lstat, 5))
summary(lm.fit5)
```

```
##
## Call:
## lm(formula = medv ~ poly(lstat, 5))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5433  -3.1039  -0.7052   2.0844  27.1153
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        22.5328     0.2318  97.197  < 2e-16 ***
## poly(lstat, 5)1 -152.4595     5.2148 -29.236  < 2e-16 ***
```

```
## poly(lstat, 5)2   64.2272     5.2148  12.316  < 2e-16 ***
## poly(lstat, 5)3  -27.0511     5.2148  -5.187 3.10e-07 ***
## poly(lstat, 5)4   25.4517     5.2148   4.881 1.42e-06 ***
## poly(lstat, 5)5  -19.2524     5.2148  -3.692 0.000247 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.215 on 500 degrees of freedom
## Multiple R-squared:  0.6817, Adjusted R-squared:  0.6785
## F-statistic: 214.2 on 5 and 500 DF,  p-value: < 2.2e-16
```

`lm()` can handle other transformations, in addition to polynomial transformations.

```r
summary(lm(medv ~ log(rm), data = Boston))
```

```
##
## Call:
## lm(formula = medv ~ log(rm), data = Boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -19.487  -2.875  -0.104   2.837  39.816
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -76.488      5.028  -15.21   <2e-16 ***
## log(rm)       54.055      2.739   19.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.915 on 504 degrees of freedom
## Multiple R-squared:  0.4358, Adjusted R-squared:  0.4347
## F-statistic: 389.3 on 1 and 504 DF,  p-value: < 2.2e-16
```

6. Qualitative Predictors

For this section, we will use the `Carseats` dataset from the `ISLR` package. We can use the `attach()` function again to load this dataset.

```r
attach(Carseats, warn.conflicts = FALSE)
head(Carseats)
```

```
##   Sales CompPrice Income Advertising Population Price ShelveLoc Age Education
## 1  9.50       138     73          11        276   120       Bad  42        17
## 2 11.22       111     48          16        260    83      Good  65        10
## 3 10.06       113     35          10        269    80    Medium  59        12
## 4  7.40       117    100           4        466    97    Medium  55        14
## 5  4.15       141     64           3        340   128       Bad  38        13
## 6 10.81       124    113          13        501    72       Bad  78        16
##   Urban  US
## 1   Yes Yes
## 2   Yes Yes
```

```
## 3   Yes Yes
## 4   Yes Yes
## 5   Yes  No
## 6    No Yes
```

```r
names(Carseats)
```

```
##  [1] "Sales"       "CompPrice"   "Income"      "Advertising" "Population"
##  [6] "Price"       "ShelveLoc"   "Age"         "Education"   "Urban"
## [11] "US"
```

When we have qualitative/categorical variables, R automatically generates dummy variables.

```r
lm.fit <- lm(Sales ~ . + Income:Advertising + Price:Age, data = Carseats)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Sales ~ . + Income:Advertising + Price:Age, data = Carseats)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9208 -0.7503  0.0177  0.6754  3.3413
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         6.5755654  1.0087470   6.519 2.22e-10 ***
## CompPrice           0.0929371  0.0041183  22.567  < 2e-16 ***
## Income              0.0108940  0.0026044   4.183 3.57e-05 ***
## Advertising         0.0702462  0.0226091   3.107 0.002030 **
## Population          0.0001592  0.0003679   0.433 0.665330
## Price              -0.1008064  0.0074399 -13.549  < 2e-16 ***
## ShelveLocGood       4.8486762  0.1528378  31.724  < 2e-16 ***
## ShelveLocMedium     1.9532620  0.1257682  15.531  < 2e-16 ***
## Age                -0.0579466  0.0159506  -3.633 0.000318 ***
## Education          -0.0208525  0.0196131  -1.063 0.288361
## UrbanYes            0.1401597  0.1124019   1.247 0.213171
## USYes              -0.1575571  0.1489234  -1.058 0.290729
## Income:Advertising  0.0007510  0.0002784   2.698 0.007290 **
## Price:Age           0.0001068  0.0001333   0.801 0.423812
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.011 on 386 degrees of freedom
## Multiple R-squared:  0.8761, Adjusted R-squared:  0.8719
## F-statistic:   210 on 13 and 386 DF,  p-value: < 2.2e-16
```

To examine the coding for the qualitative variables, the `constasts()` function can be used.

```r
contrasts(ShelveLoc)
```

```
##        Good Medium
## Bad       0      0
## Good      1      0
## Medium    0      1
```

7. Writing Functions

We can write our own functions to expand the functionality of R.

```
LoadLibraries <- function() {
    library(ISLR)
    library(MASS)
}
```

```
LoadLibraries()
```

Exercises:

1. For this exercise, we will work with the `Cars93` data from the `MASS` package, which contains information about 93 cars on sale in the US in 1993.

(a) Load the data and evaluate the columns. Choose 3 of the columns and describe each of those variables.

```
library(MASS)
data("Cars93")
names(Cars93)
```

```
##  [1] "Manufacturer"       "Model"              "Type"
##  [4] "Min.Price"          "Price"              "Max.Price"
##  [7] "MPG.city"           "MPG.highway"        "AirBags"
## [10] "DriveTrain"         "Cylinders"          "EngineSize"
## [13] "Horsepower"         "RPM"                "Rev.per.mile"
## [16] "Man.trans.avail"    "Fuel.tank.capacity" "Passengers"
## [19] "Length"             "Wheelbase"          "Width"
## [22] "Turn.circle"        "Rear.seat.room"     "Luggage.room"
## [25] "Weight"             "Origin"             "Make"
```

```
# Price
summary(Cars93$Price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    7.40   12.20   17.70   19.51   23.30   61.90
```

Price: Midrange Price (in $1,000): average of `Min.Price` and `Max.Price`.

```
# Horsepower
summary(Cars93$Horsepower)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    55.0   103.0   140.0   143.8   170.0   300.0
```

Horsepower: Horsepower (maximum).

```
# Cylinders
summary(Cars93$Cylinders)
```

```
##       3       4       5       6       8 rotary
##       3      49       2      31       7       1
```

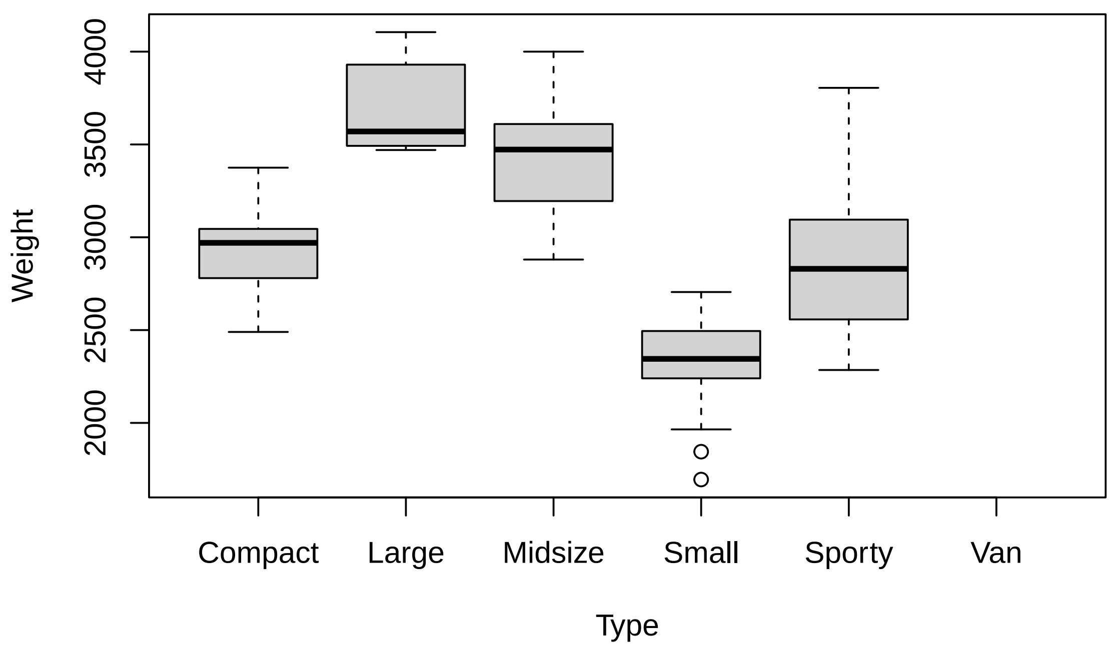
Cylinders: Number of cylinders (missing for Mazda RX-7, which has a rotary engine).

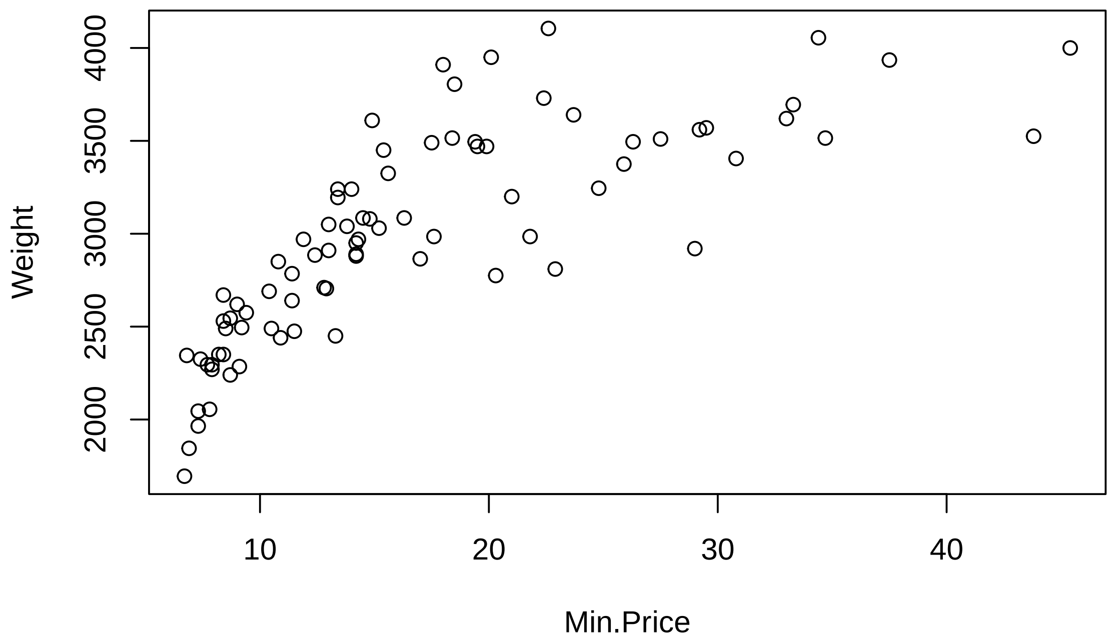
(b) using data subsetting techniques from last week's lab, remove the columns `Manufacturer`, `Model` and `Make`. Additionally, remove any rows with `NA` values.

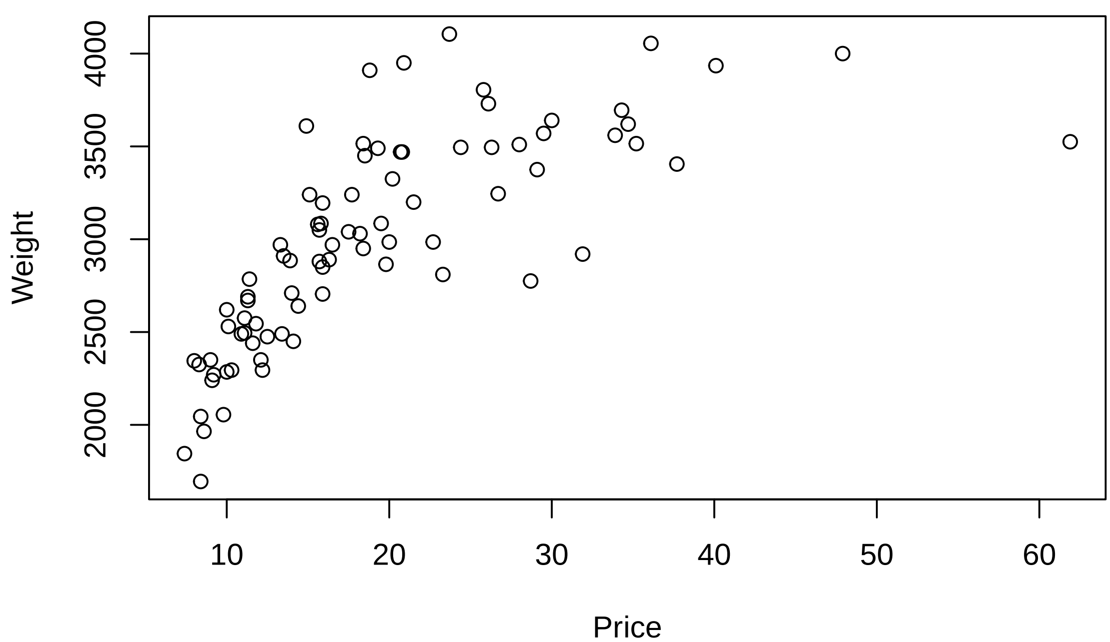
```
Cars93$Manufacturer <- NULL
Cars93$Model <- NULL
Cars93$Make <- NULL
Cars93 <- na.omit(Cars93)
```

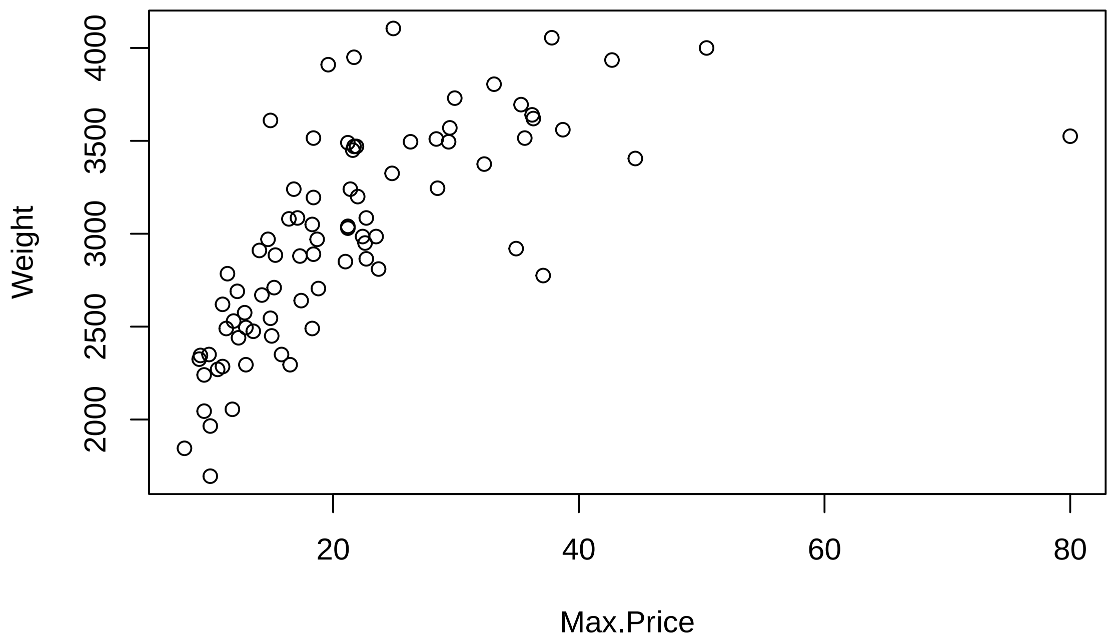
(c) Suppose that we wish to predict the `Weight` of each car, given the other variables as predictor variables. Visually examine the relationship between `Weight` and the other predictors. What predictors show a strong positive association with `Weight`? Which variables show a strong negative association? Which variables have a non-linear relationship with `Weight`?

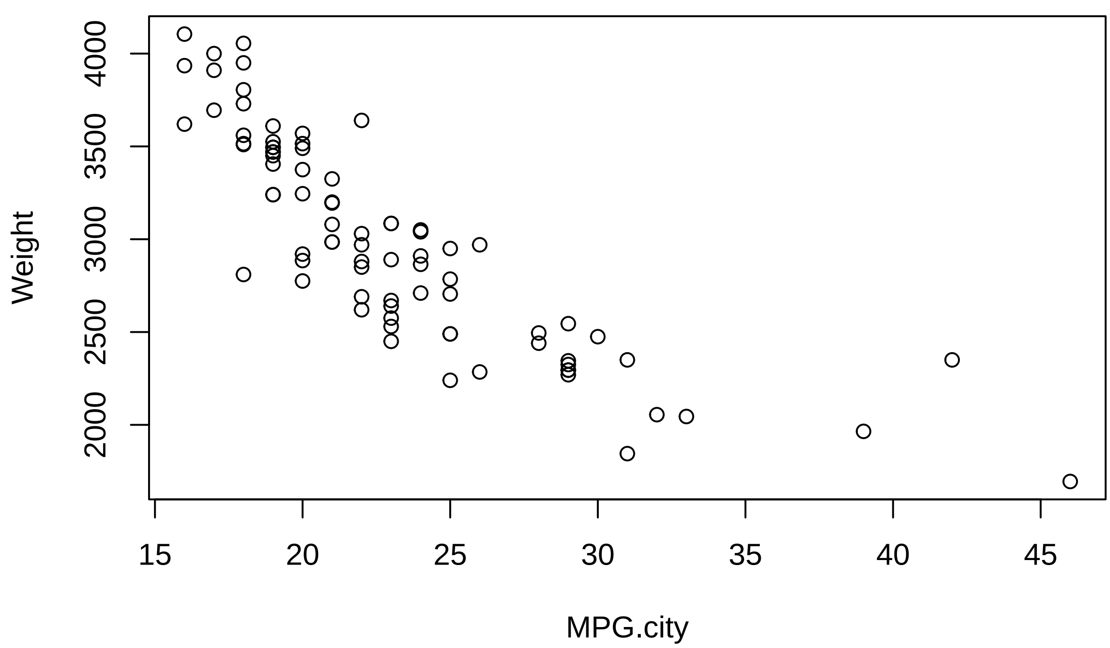
```
for(i in 1:22) {
  plot(Cars93[[i]], Cars93$Weight, xlab=names(Cars93)[i], ylab="Weight")
}
```

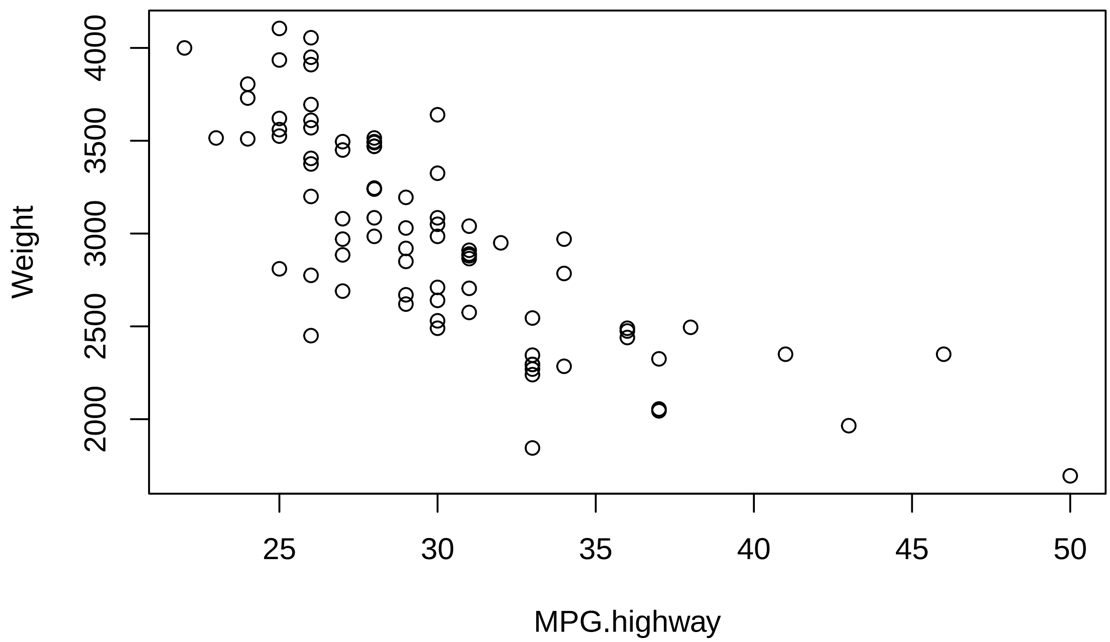



Weight
2000
2500
3000
3500
4000
10
20
30
40
Min.Price

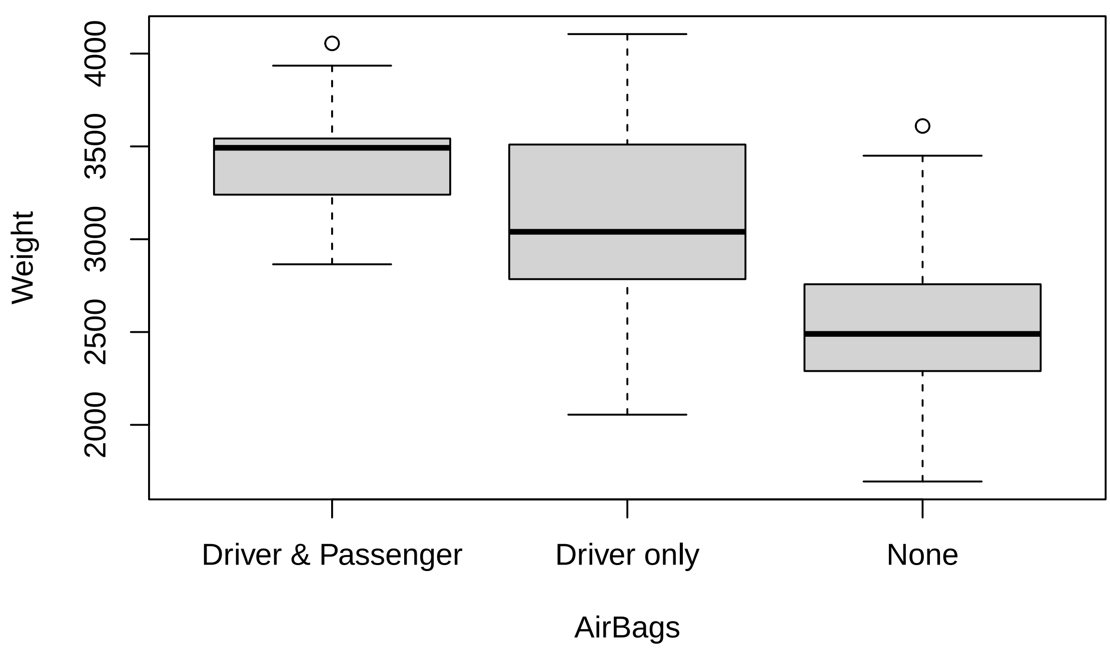

Weight
2000
2500
3000
3500
4000
10
20
30
40
50
60
Price


Weight
2000
2500
3000
3500
4000
20
40
60
80
Max.Price

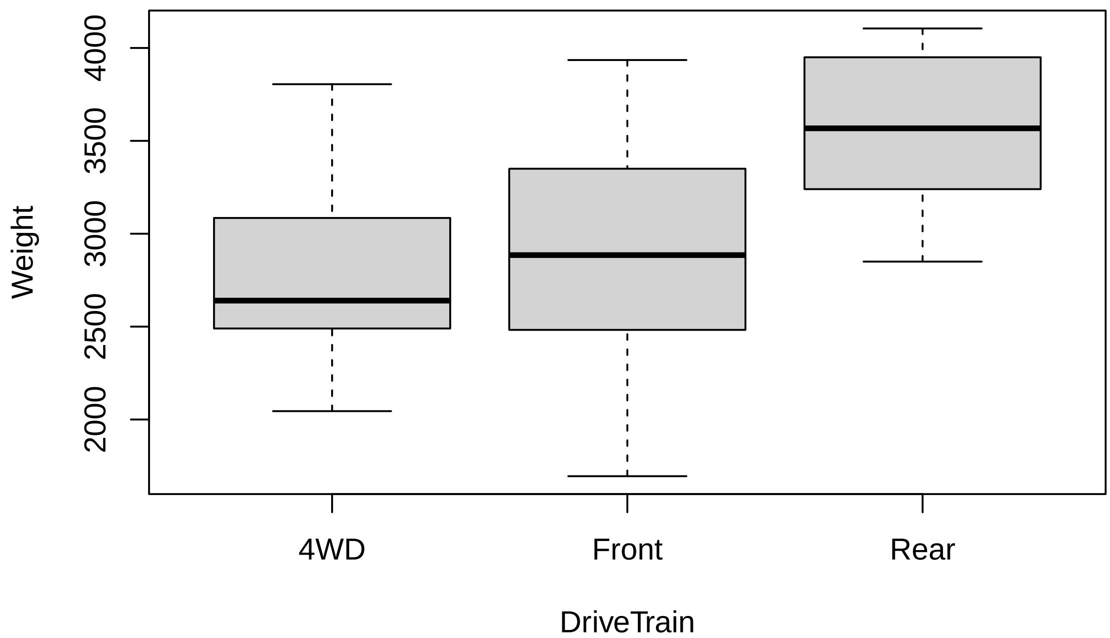

Weight
2000
2500
3000
3500
4000
15
20
25
30
35
40
45
MPG.city

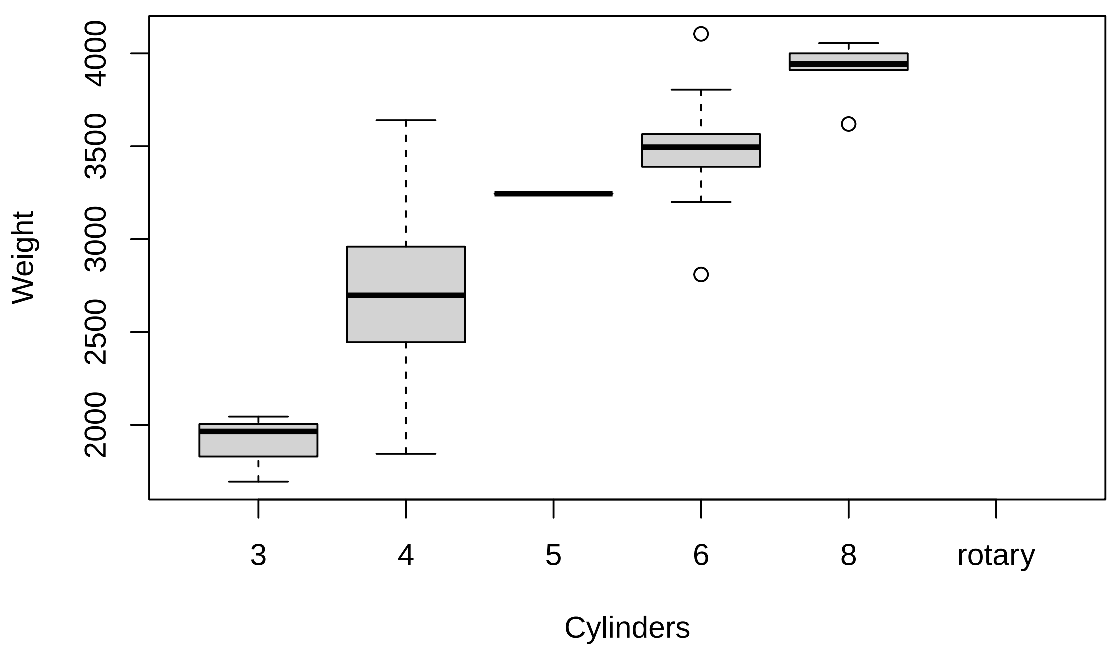


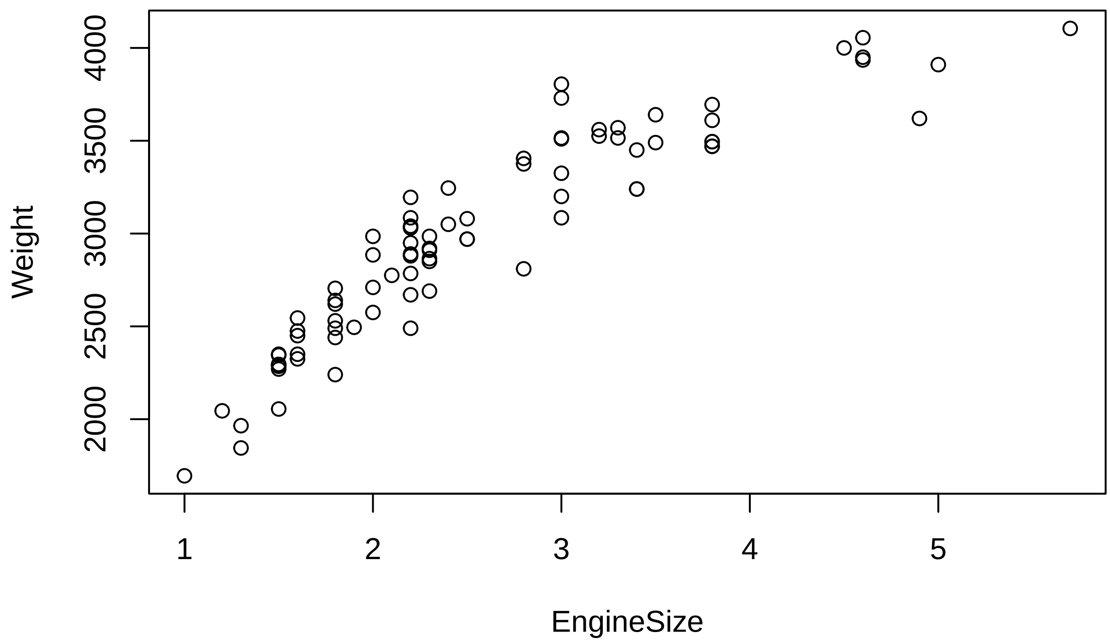
Weight
2000
2500
3000
3500
4000
25
30
35
40
45
50
MPG.highway

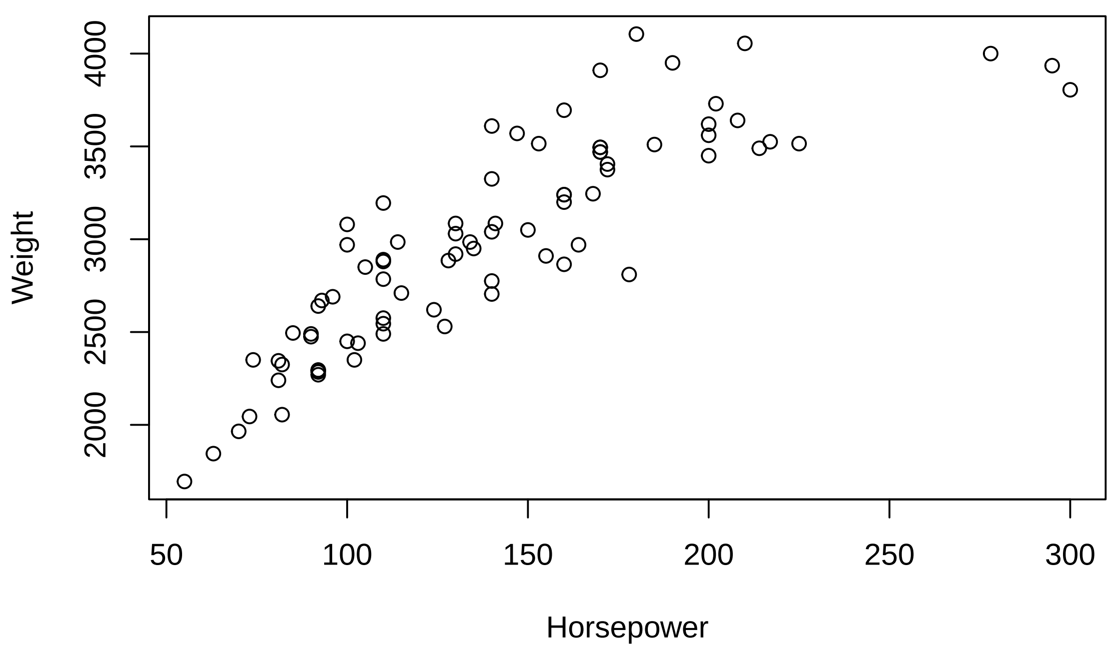


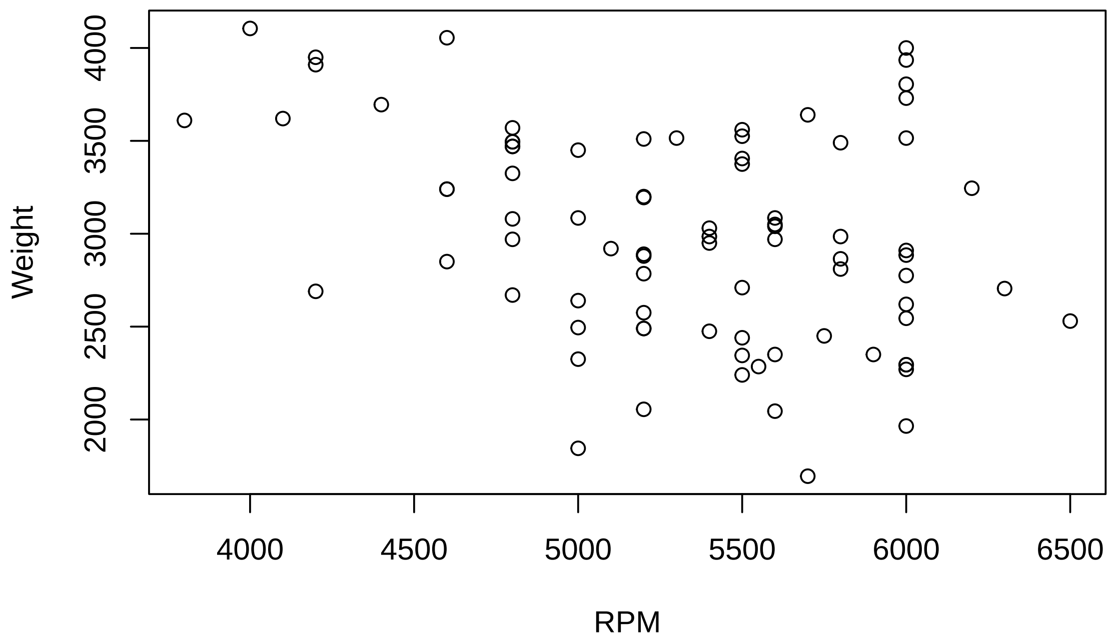
Weight
2000
2500
3000
3500
4000
Driver & Passenger
Driver only
None
AirBags

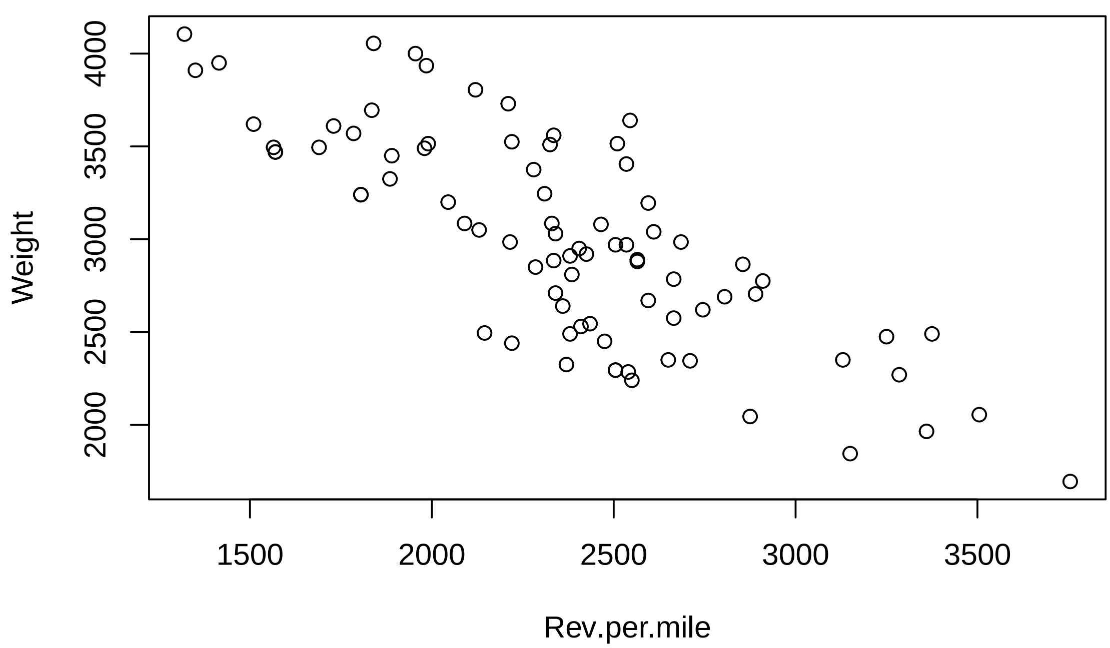

Weight
2000
2500
3000
3500
4000
4WD
Front
Rear
DriveTrain

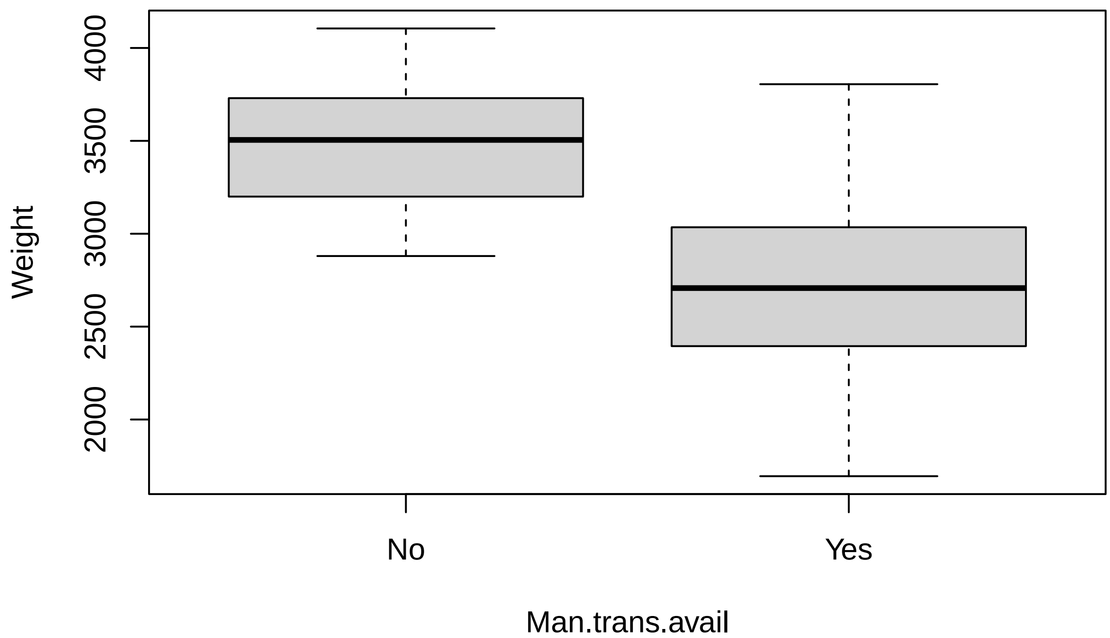

Weight
2000
2500
3000
3500
4000
3
4
5
6
8
rotary
Cylinders


Weight
EngineSize
1
2
3
4
5
2000
2500
3000
3500
4000

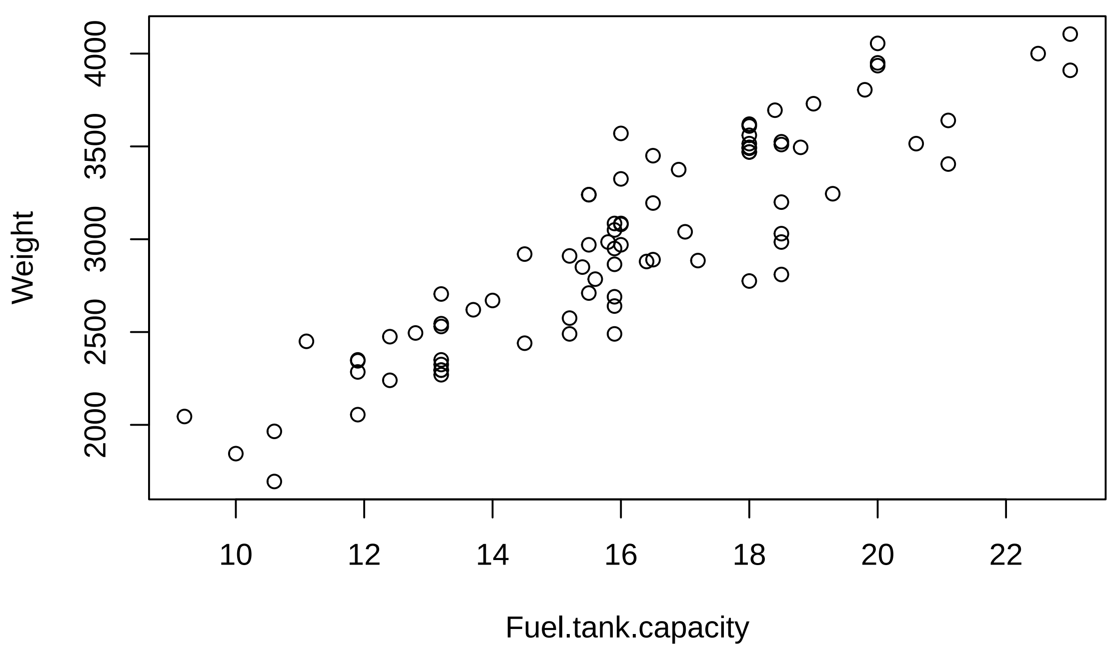

Weight
Horsepower
50
100
150
200
250
300
2000
2500
3000
3500
4000


Weight
2000
2500
3000
3500
4000
4000
4500
5000
5500
6000
6500
RPM

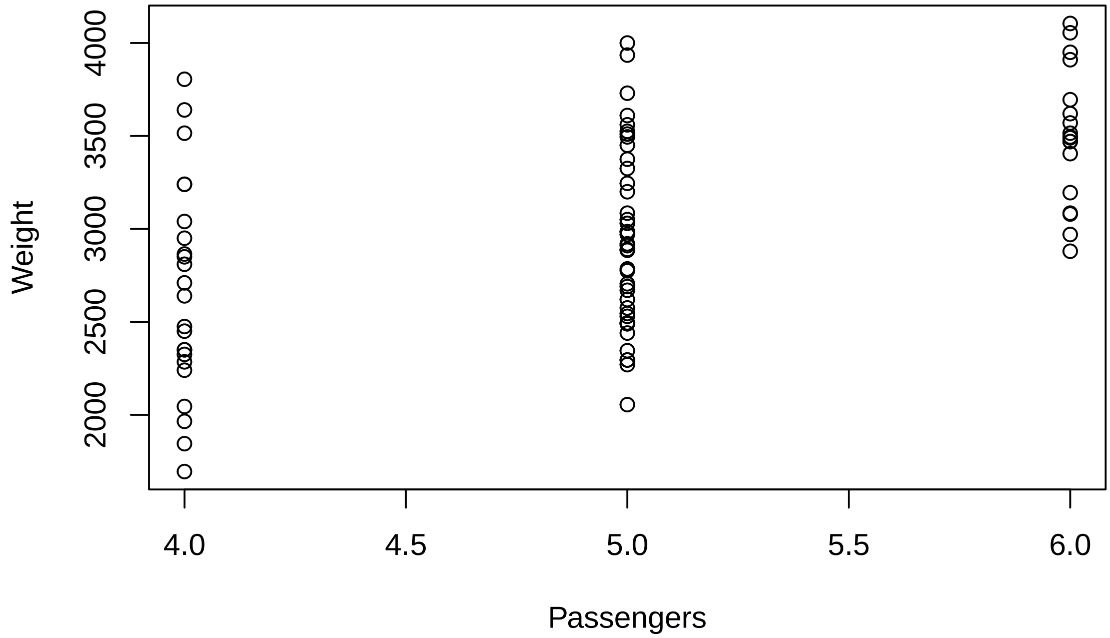

Weight
2000
2500
3000
3500
4000
1500
2000
2500
3000
3500
Rev.per.mile

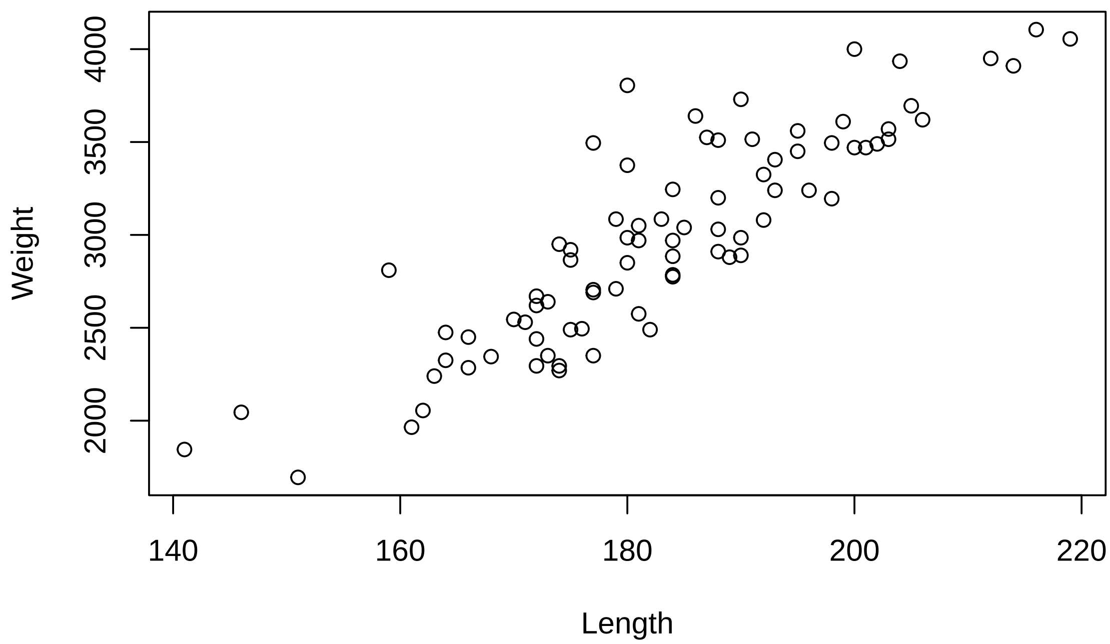


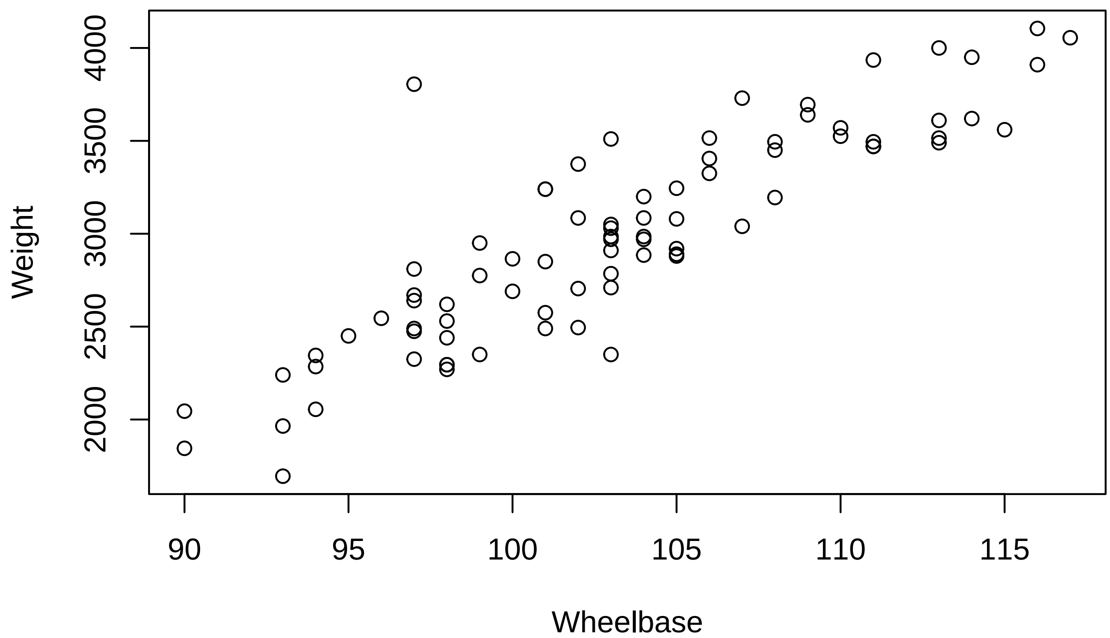
4000
3500
3000
2500
2000
Weight
No
Yes
Man.trans.avail

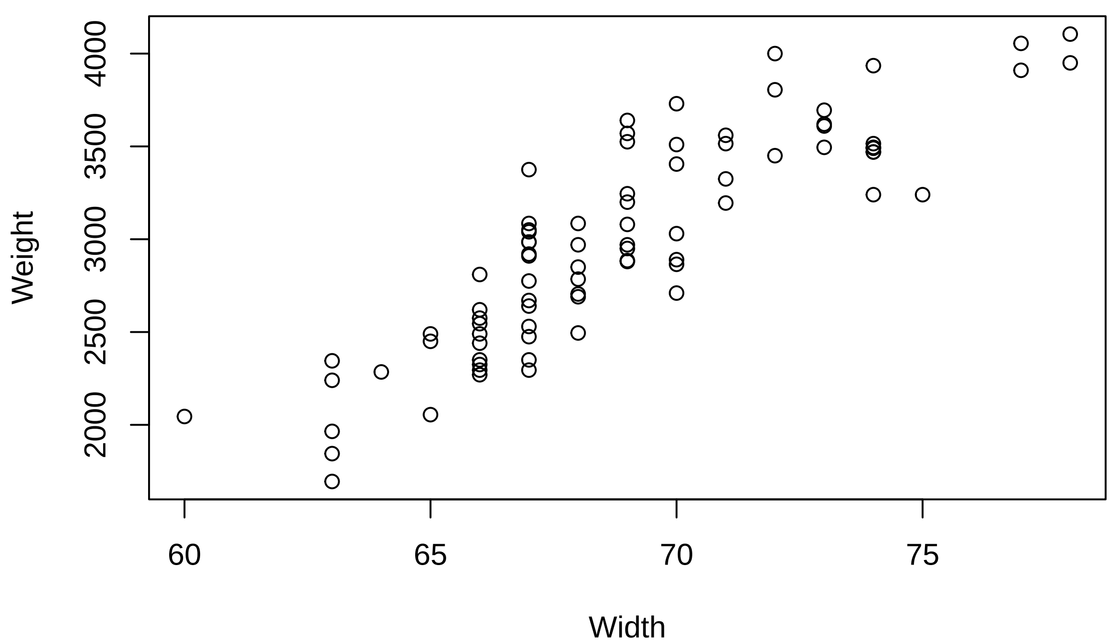


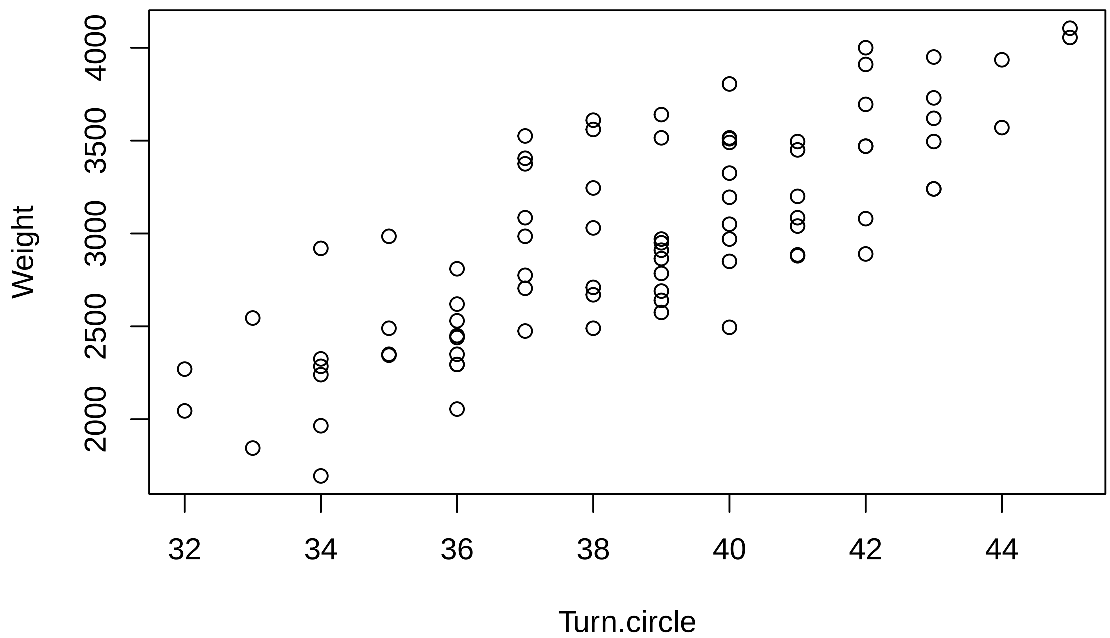
4000
3500
3000
2500
2000
Weight
10
12
14
16
18
20
22
Fuel.tank.capacity

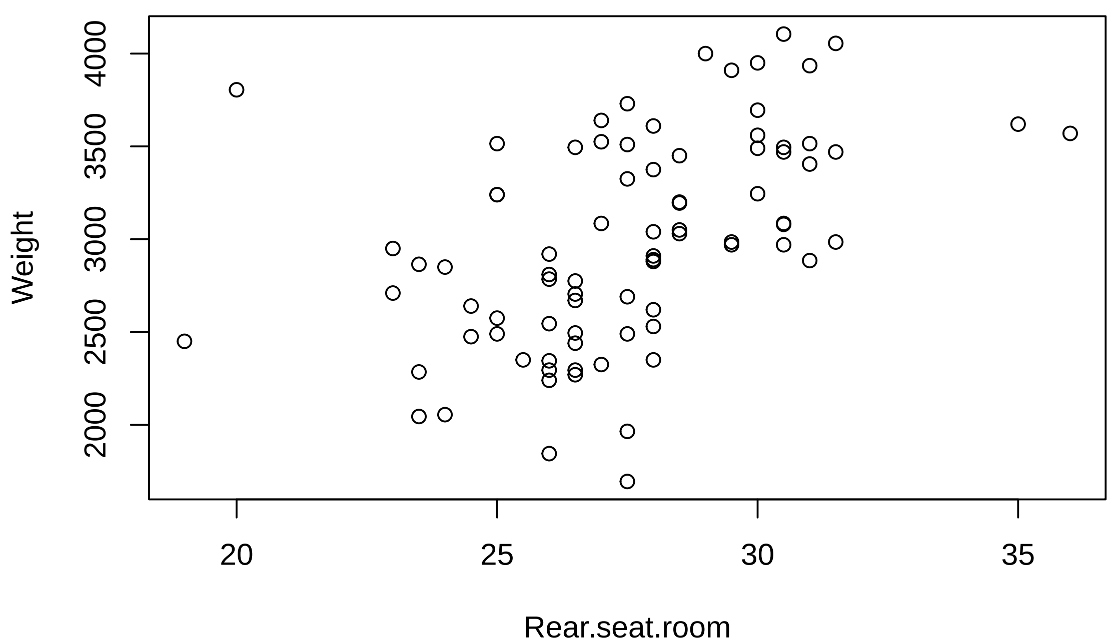

Weight
4000
3500
3000
2500
2000
4.0
4.5
5.0
5.5
6.0
Passengers

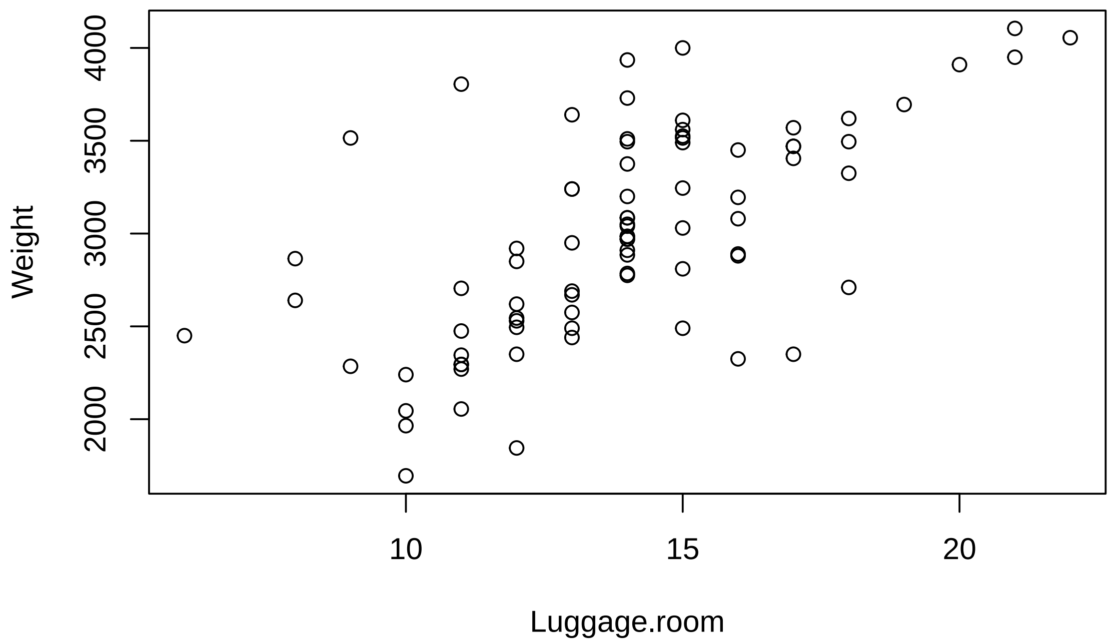

Weight
4000
3500
3000
2500
2000
140
160
180
200
220
Length


Weight
4000
3500
3000
2500
2000
90
95
100
105
110
115
Wheelbase

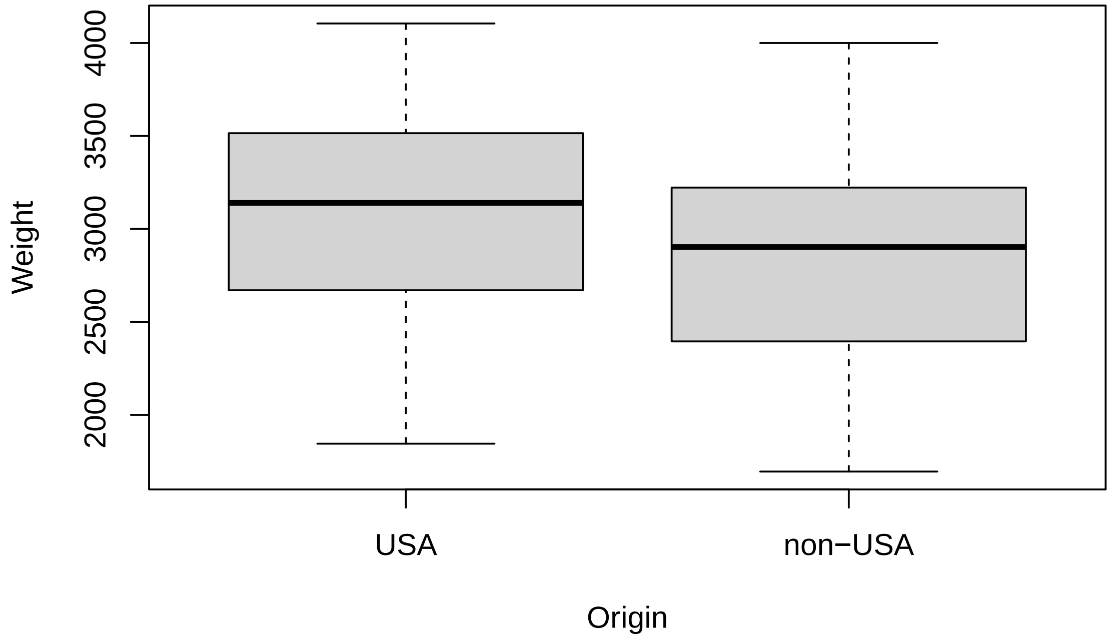

Weight
4000
3500
3000
2500
2000
60
65
70
75
Width


Weight
4000
3500
3000
2500
2000
32
34
36
38
40
42
44
Turn.circle


Weight
4000
3500
3000
2500
2000
20
25
30
35
Rear.seat.room




```
for(i in 24:24) {
  plot(Cars93[[i]], Cars93$Weight, xlab=names(Cars93)[i], ylab="Weight")
}
```



What predictors show a strong positive association with `Weight`?

Min.Price, Price, Max.Price, EngineSize, Horsepower, Fuel.tank.capacity, length, Wheelbase, Width, Turn.circle

Which variables show a strong negative association?

MPG.city, MPG.highway, Rev.per.mile

Which variables have a non-linear relationship with `Weight`?

Min.Price, Price, Max.Price

(d) Suppose you want to perform a simple linear regression on `Weight`. You want to select the 1 predictor variable that best predicts `Weight`. Which predictor variable do you select? Why did you select this variable?

I would select EngineSize because it has one of the strongest positive linear relationship with weight.

(e) For the variable selected in part (d), perform a simple linear regression. Report the estimated value of $\hat{\beta}_1$, the standard error and the R^2 value of your model.

```
lm.fit <- lm(Weight ~ EngineSize, data = Cars93)
summary(lm.fit)
```

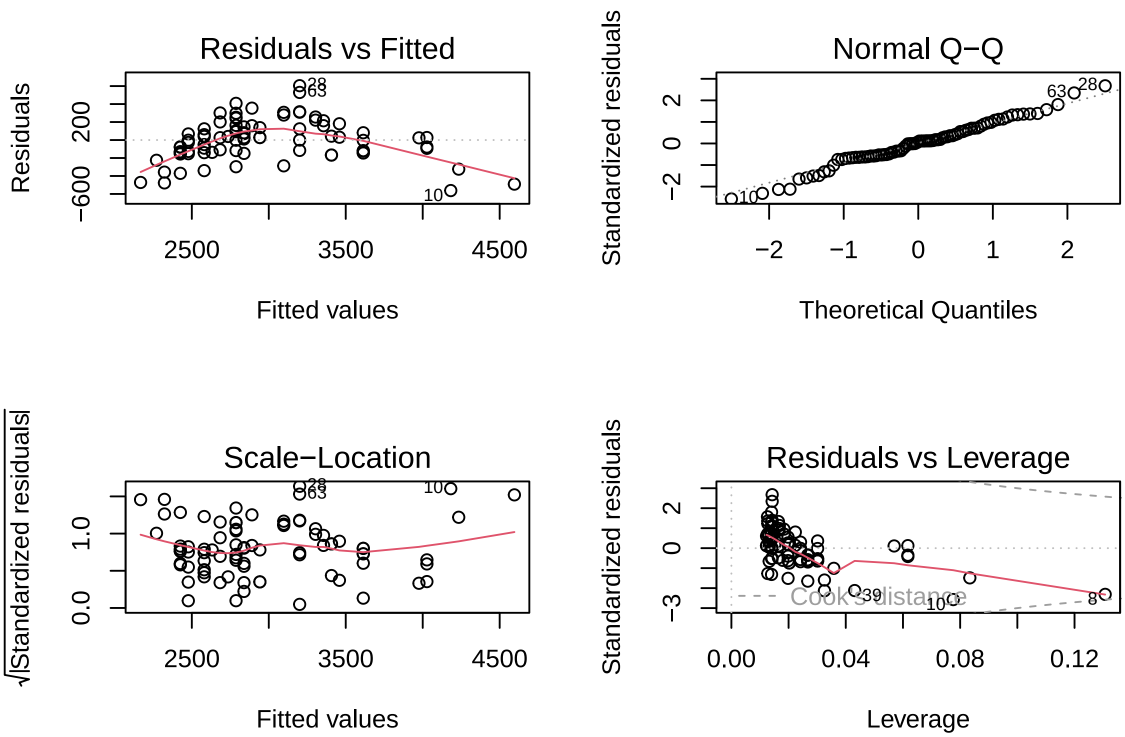
```
##
## Call:
## lm(formula = Weight ~ EngineSize, data = Cars93)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -562.28 -135.47   17.79  144.09  604.47
```

```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1650.39      69.85   23.63   <2e-16 ***
## EngineSize    516.71      25.17   20.52   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 227.5 on 80 degrees of freedom
## Multiple R-squared:  0.8404, Adjusted R-squared:  0.8384
## F-statistic: 421.3 on 1 and 80 DF,  p-value: < 2.2e-16
```

$\hat{\beta}_1$ is 516.71, SE is 25.17, and R^2 is 0.8404.

(f) Perform model diagnostics on the fit from part (e). Include appropriate plots. What can you conclude about the model fit?

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



The model fit is ok but not perfect, as indicated by the curved pattern of residuals etc.

2. Multiple Linear Regression: Now we want to include multiple predictors in our model for `Weight`. The goal is to predict `Weight` well with the fewest number of predictors possible. Explore possible multiple linear regression models, including transformations of predictor variables and interaction terms. Choose 3 of the coefficients in your model and write an interpretation for them. Perform some model diagnostics and comment on the quality of your model fit. Do the fitted coefficients make sense for this dataset?

```r
lm.fit2 <- lm(Weight ~ (EngineSize + Horsepower + Fuel.tank.capacity)^2, data = Cars93)
summary(lm.fit2)
```

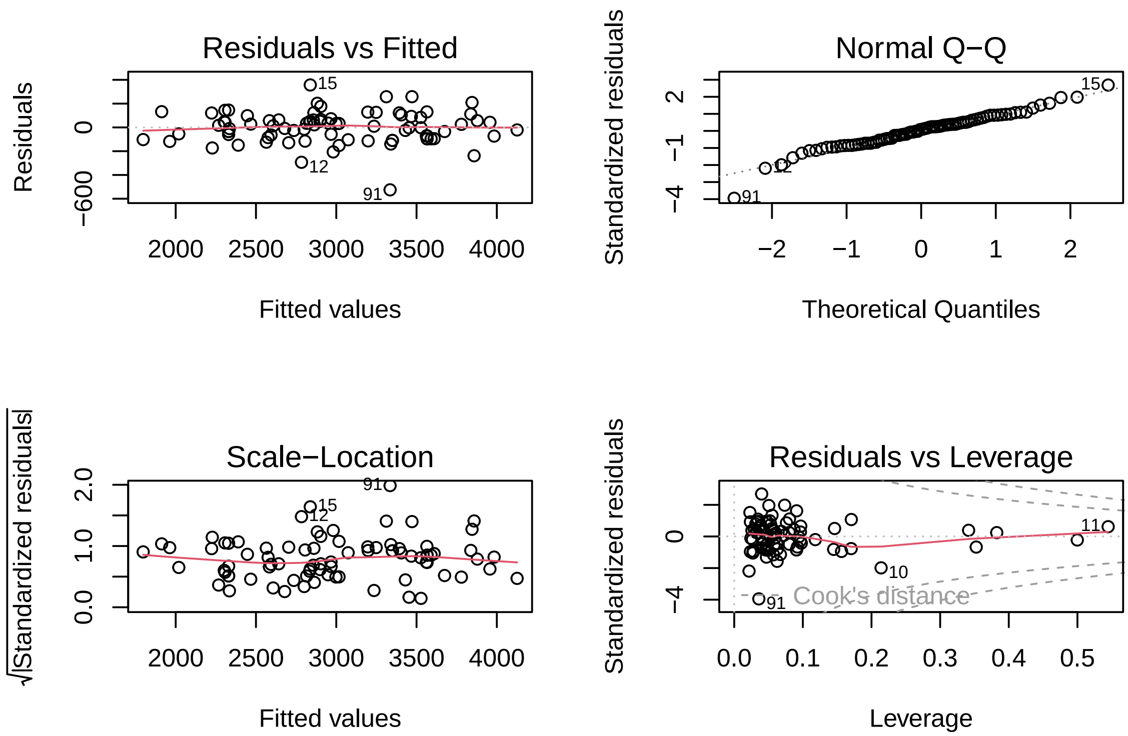
```
##
## Call:
## lm(formula = Weight ~ (EngineSize + Horsepower + Fuel.tank.capacity)^2,
##     data = Cars93)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -525.25  -92.48   11.42   71.16  356.88
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   367.9402   264.2290   1.393 0.167884
## EngineSize                    741.0168   185.8707   3.987 0.000154 ***
## Horsepower                      4.6507     3.5210   1.321 0.190568
## Fuel.tank.capacity             52.9357    28.7537   1.841 0.069576 .
## EngineSize:Horsepower          -1.5073     0.6166  -2.445 0.016851 *
## EngineSize:Fuel.tank.capacity -11.6422     8.1115  -1.435 0.155367
## Horsepower:Fuel.tank.capacity   0.1335     0.2304   0.580 0.563981
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 135.5 on 75 degrees of freedom
## Multiple R-squared:  0.9469, Adjusted R-squared:  0.9427
## F-statistic:   223 on 6 and 75 DF,  p-value: < 2.2e-16
```

Holding other variables constant, a one-unit increase in the engine size is associated with an increase of approximately 741.0168 units in the weight of the car.

Holding other variables constant, a one-unit increase in the horsepower is associated with an increase of approximately 4.6507 units in the weight of the car.

Holding other variables constant, a one-unit increase in the fuel tank capacity is associated with an increase of approximately 52.9357 units in the weight of the car.

```r
par(mfrow=c(2,2))
plot(lm.fit2)
```



Accordingt to the model diagnostics, the model seems to be a great fit. And the fitted coefficients indeed make sense for this dataset — for example, for a car with larger engine size, we do expect its weight to be heavier.

3. Write a function that accepts two arguments, `x` and `y` and returns the following calculations: `x+y`, `x*y` and `x/y`. Make sure to check if the division can be performed, and return an error message if not. The function should return a named list. Test your function on some sample values and print the results.

```
calc <- function(x,y) {
    if(y == 0) {
    division_result <- "Error: Division by zero is not allowed"
  } else {
    division_result <- x / y
  }

  results <- list(
    sum = x + y,
    product = x * y,
    division = division_result
  )

  return(results)
}
```

```
# Testing the function with sample values
test1 <- calc(4, 2)
test2 <- calc(5, 0)
```

```r
# Printing the results
print(test1)
```

```
## $sum
## [1] 6
##
## $product
## [1] 8
##
## $division
## [1] 2
```

```r
print(test2)
```

```
## $sum
## [1] 5
##
## $product
## [1] 0
##
## $division
## [1] "Error: Division by zero is not allowed"
```