Workshop 8

- Univariate regression
    - Using the inbuilt function
    - From first principles
    - Test linear hypothesis
- Multivariate regression

Univariate regression

In this section, we are going to look at how to do a regression in `R`. First we download the advertising data set that is an example from the book "Introduction to statistical learning".

```
ads <- read.csv('http://www-bcf.usc.edu/~gareth/ISL/Advertising.csv', row.names
= 1)
```

The data is stored in a `data.frame`. We can inspect a bit of the data contained in the `data.frame`.

```
head(ads)
```

```
##      TV Radio Newspaper Sales
## 1 230.1  37.8      69.2  22.1
## 2  44.5  39.3      45.1  10.4
## 3  17.2  45.9      69.3   9.3
## 4 151.5  41.3      58.5  18.5
## 5 180.8  10.8      58.4  12.9
## 6   8.7  48.9      75.0   7.2
```

We can get the descriptive statistics of the data frame.

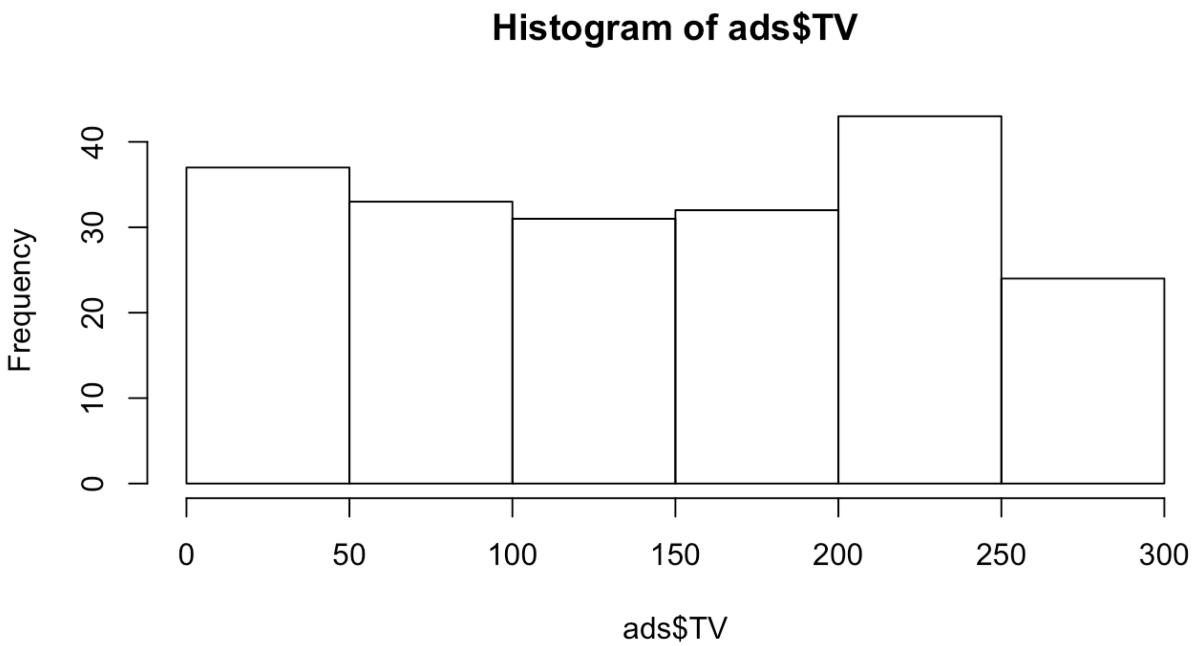
```
summary(ads)
```

```
##        TV             Radio          Newspaper          Sales
##  Min.   :  0.70   Min.   : 0.000   Min.   :  0.30   Min.   : 1.60
##  1st Qu.: 74.38   1st Qu.: 9.975   1st Qu.: 12.75   1st Qu.:10.38
##  Median :149.75   Median :22.900   Median : 25.75   Median :12.90
##  Mean   :147.04   Mean   :23.264   Mean   : 30.55   Mean   :14.02
##  3rd Qu.:218.82   3rd Qu.:36.525   3rd Qu.: 45.10   3rd Qu.:17.40
##  Max.   :296.40   Max.   :49.600   Max.   :114.00   Max.   :27.00
```

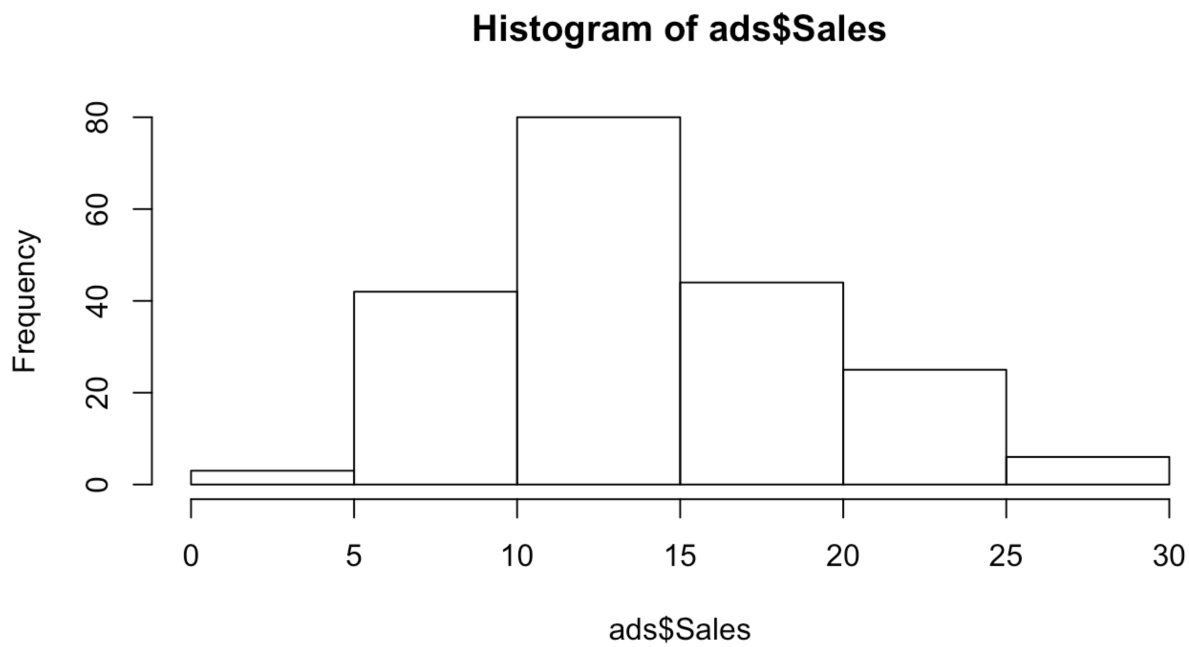
Using the inbuilt function

As we want to perform a simple regression model involves ‘TV’ and ‘Sales’ we do a bit of exploration of these variables.

```
hist(ads$TV)
```

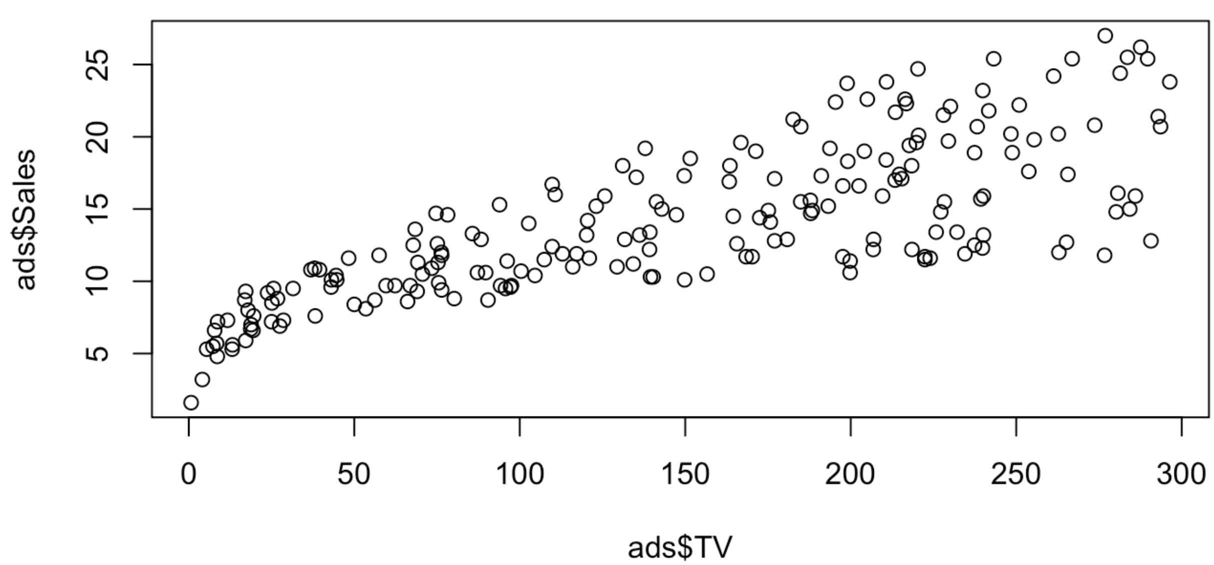



```
hist(ads$Sales)
```



Scatterplot and correlation between ‘TV’ and ‘Sales’ to make sure that fitting a regression line “makes sense”.

```
plot(ads$TV, ads$Sales)
```



```
cor(ads$TV, ads$Sales)
```

```
## [1] 0.7822244
```

Simple linear regression with function `lm()` (i.e. a linear model).

```
model <- lm(Sales ~ TV, data = ads)
model
```

```
##
## Call:
## lm(formula = Sales ~ TV, data = ads)
##
## Coefficients:
## (Intercept)           TV
##     7.03259      0.04754
```

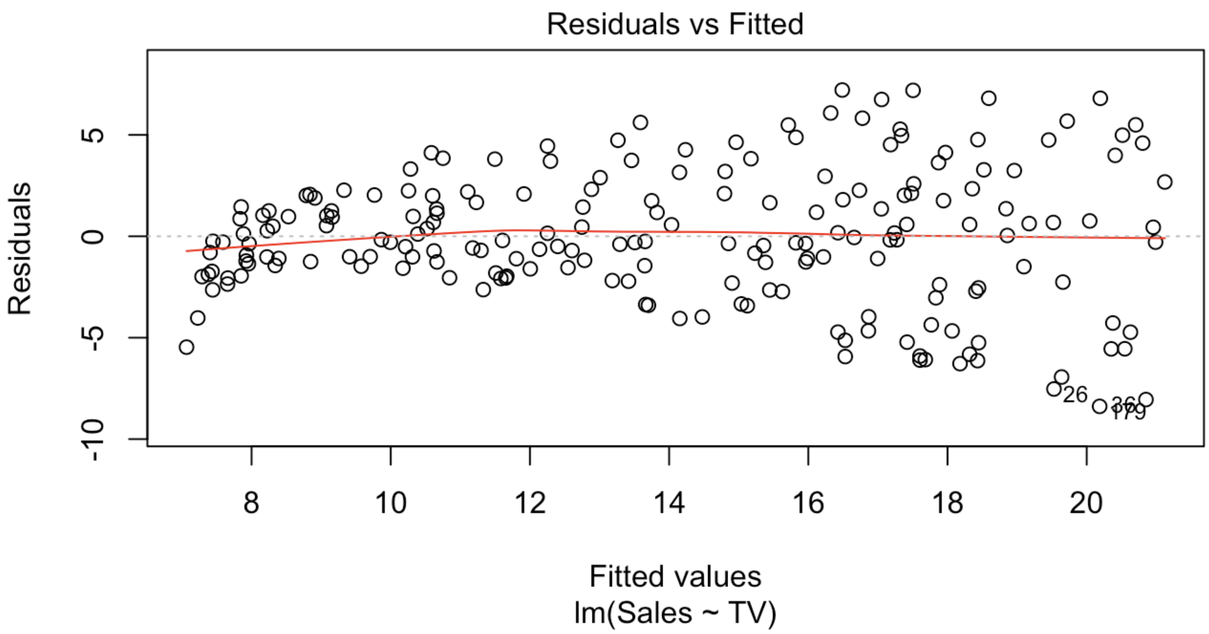
`lm()` returns an object of class “lm”.

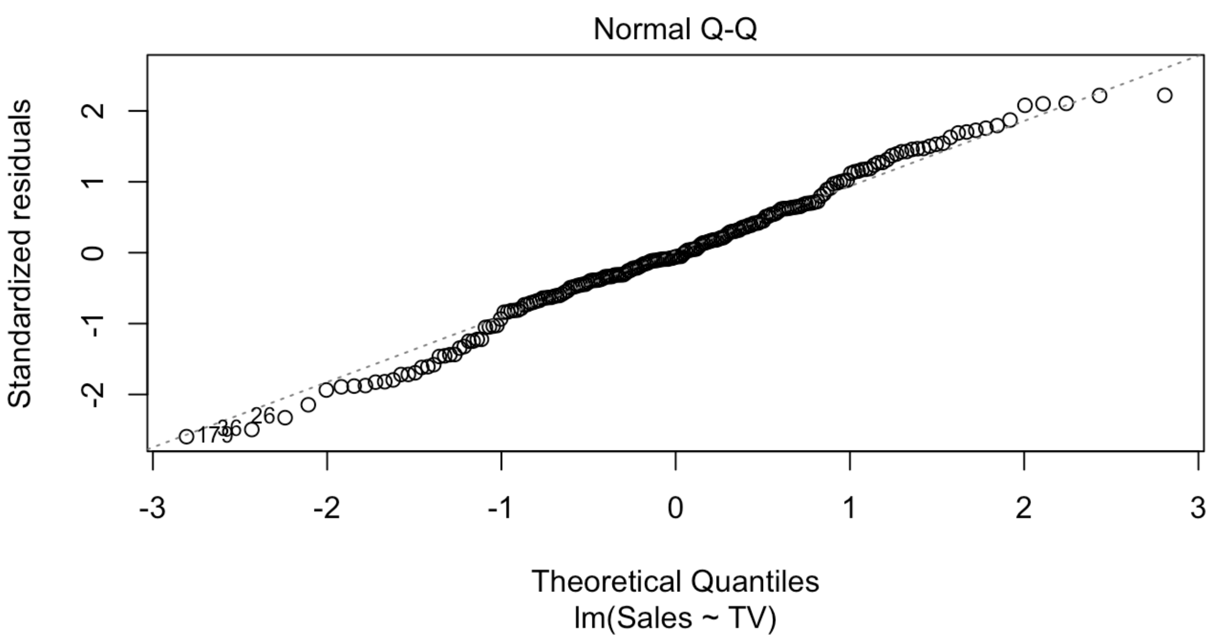
```
class(model)
```

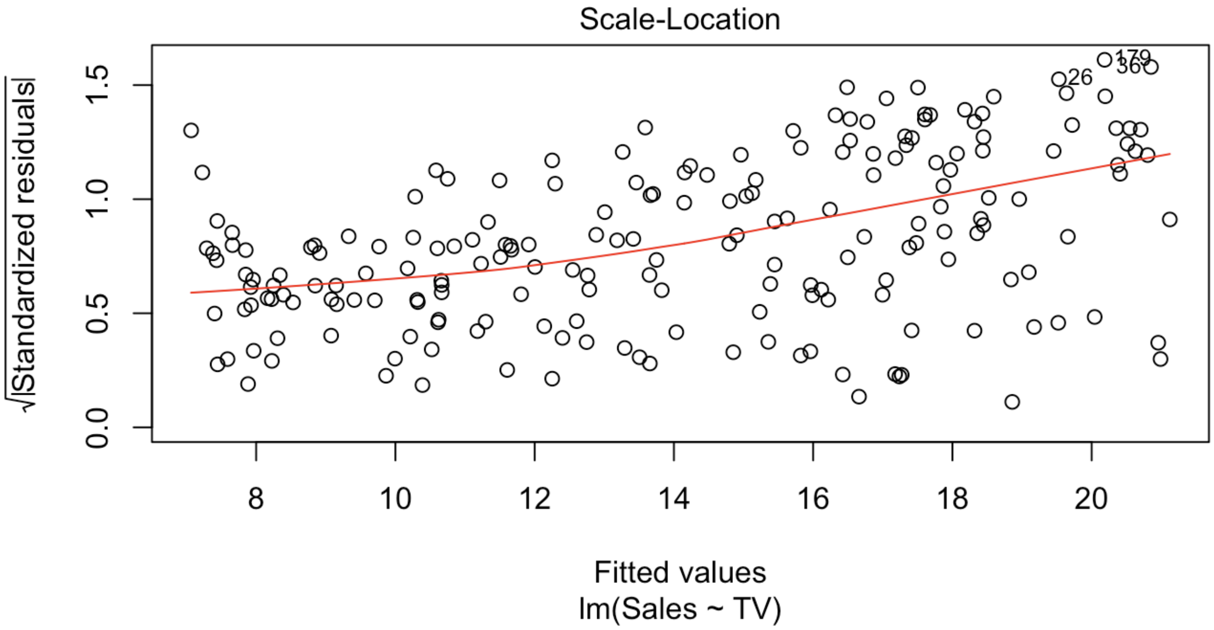
```
## [1] "lm"
```

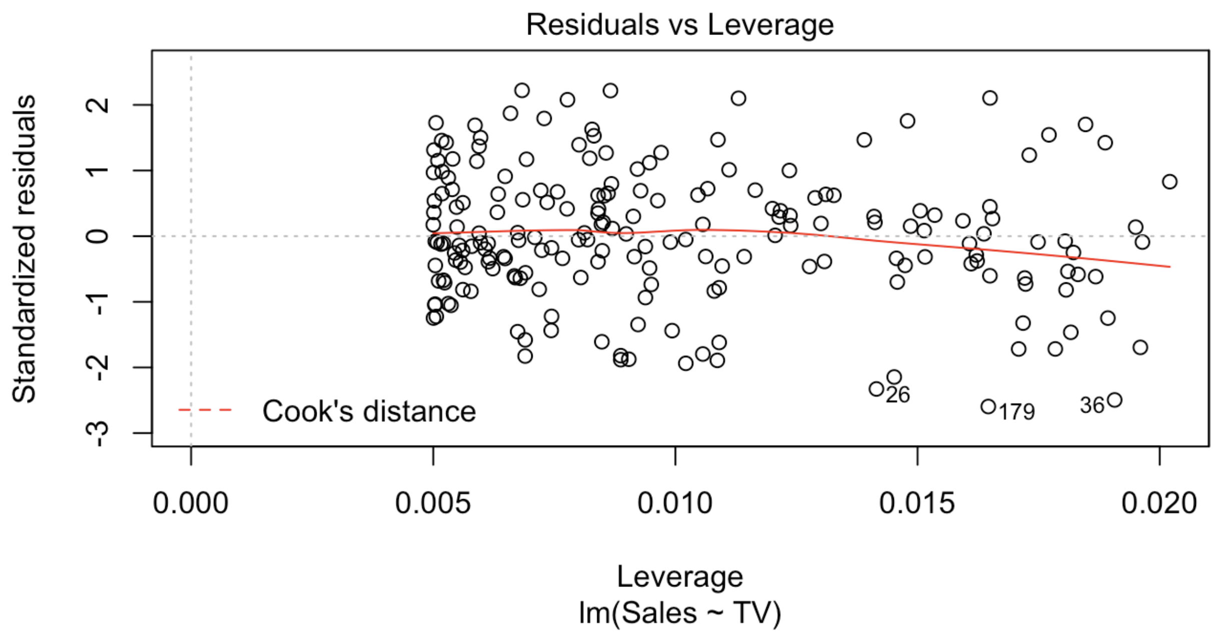
An object of class “lm” has some default plots.

```
plot(model)
```


Residuals vs Fitted
Residuals
5
0
-5
-10
8
10
12
14
16
18
20
Fitted values
lm(Sales ~ TV)
26
36
179


Normal Q-Q
Standardized residuals
2
1
0
-1
-2
-3
-2
-1
0
1
2
3
Theoretical Quantiles
lm(Sales ~ TV)
179
36
26





An object of class “lm” has a ‘summary()’ method.

```
summary(model)
```

```
##
## Call:
## lm(formula = Sales ~ TV, data = ads)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.3860 -1.9545 -0.1913  2.0671  7.2124
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.032594   0.457843   15.36   <2e-16 ***
## TV          0.047537   0.002691   17.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.259 on 198 degrees of freedom
## Multiple R-squared:  0.6119, Adjusted R-squared:  0.6099
## F-statistic: 312.1 on 1 and 198 DF,  p-value: < 2.2e-16
```

The object ‘model’ also contains more results.

```
names(model)
```

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

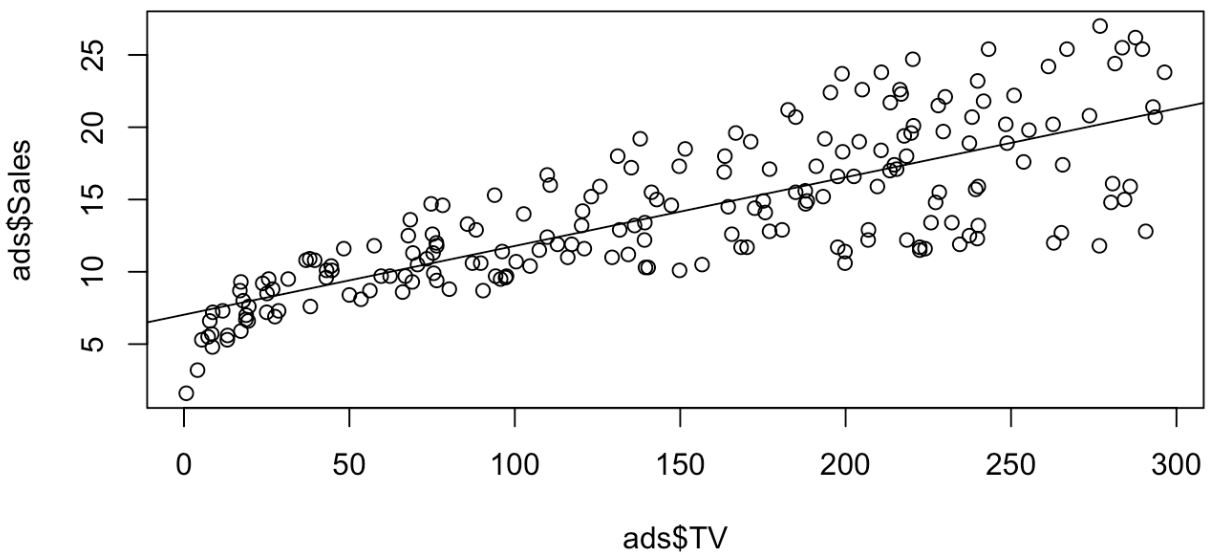
95% confidence intervals of regression coefficients.

```
confint(model, level = 0.95)
```

```
##                  2.5 %     97.5 %
## (Intercept) 6.12971927 7.93546783
## TV          0.04223072 0.05284256
```

Scatter plot with regression line.

```
plot(ads$TV, ads$Sales)
abline(model)
```



From first principles

We can also do the regression from first principles (without using the inbuilt function). We start by setting up our data.

```
y <- ads$Sales
X <- cbind(rep(1, length(y)), ads$TV)
```

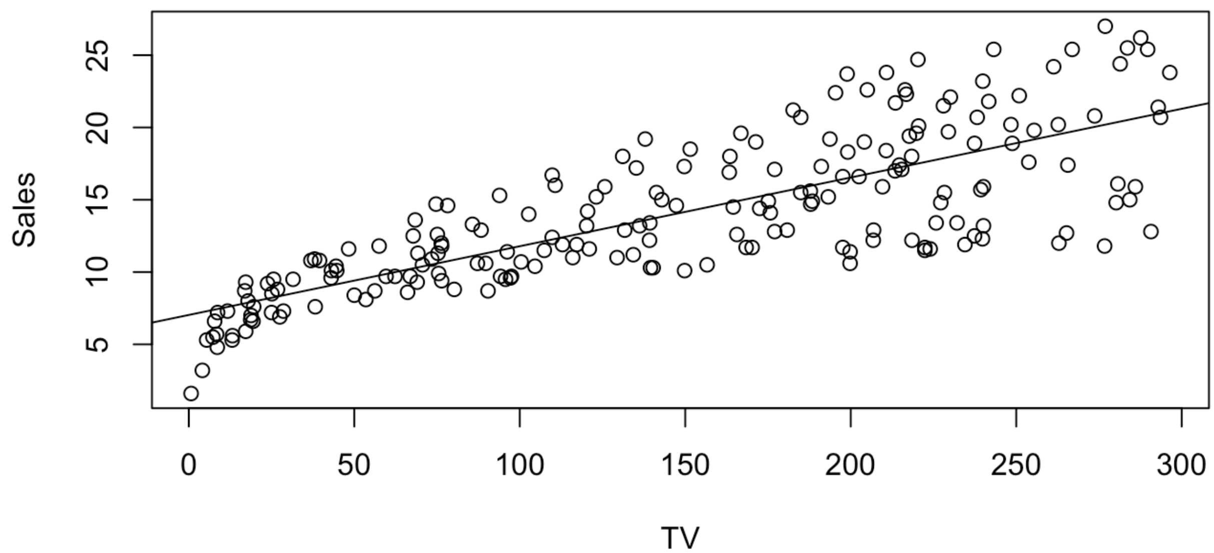
Solve the least squares problem, directly from the normal equations, to get our $\hat{\beta}$.

```
betahat <- solve(t(X) %*% X) %*% t(X) %*% y
betahat
```

```
##             [,1]
## [1,] 7.03259355
## [2,] 0.04753664
```

Plot the solution against the data values.

```
plot(ads$TV, ads$Sales, xlab="TV", ylab="Sales")
abline(betahat[1], betahat[2])
```



Also notice that $\hat{\beta}$ is the same as the coefficients found using the inbuilt `lm()` function. We can retrieve them using the `coef` function.

```
coef(model)
```

```
## (Intercept)          TV
##  7.03259355  0.04753664
```

```
betahat
```

```
##            [,1]
## [1,] 7.03259355
## [2,] 0.04753664
```

Test linear hypothesis

We can install the package `car` (`install.packages('car')`)that has the asymptotic Chi-squared approach built-in and some other data sets.

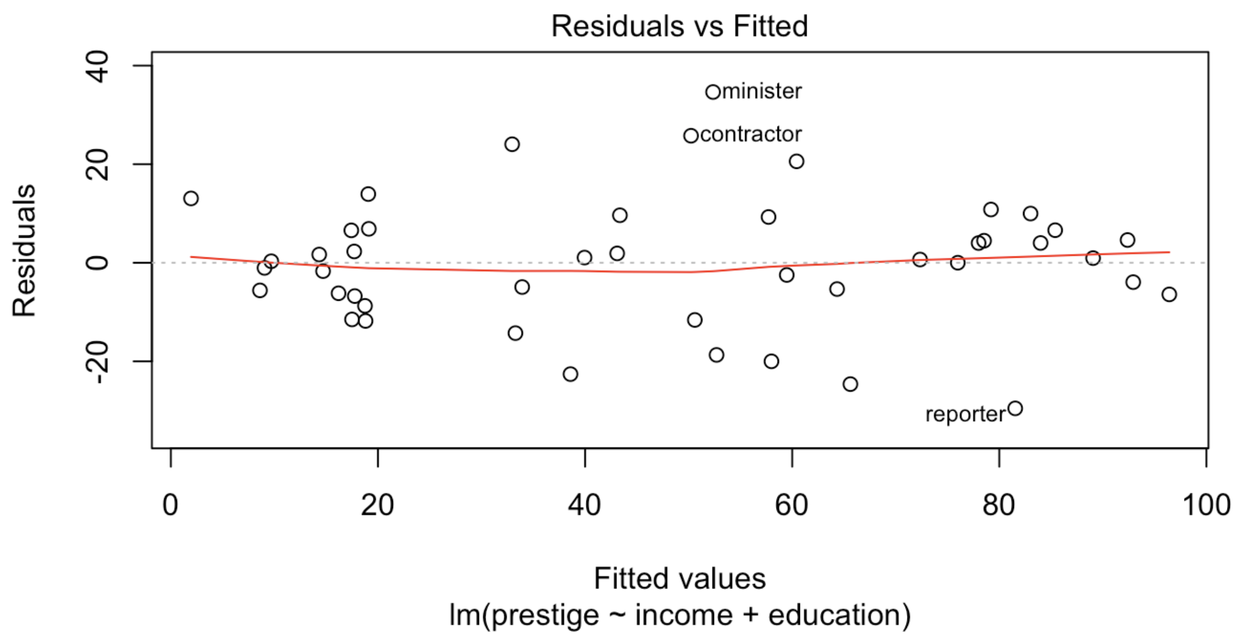
```
library(car)
```

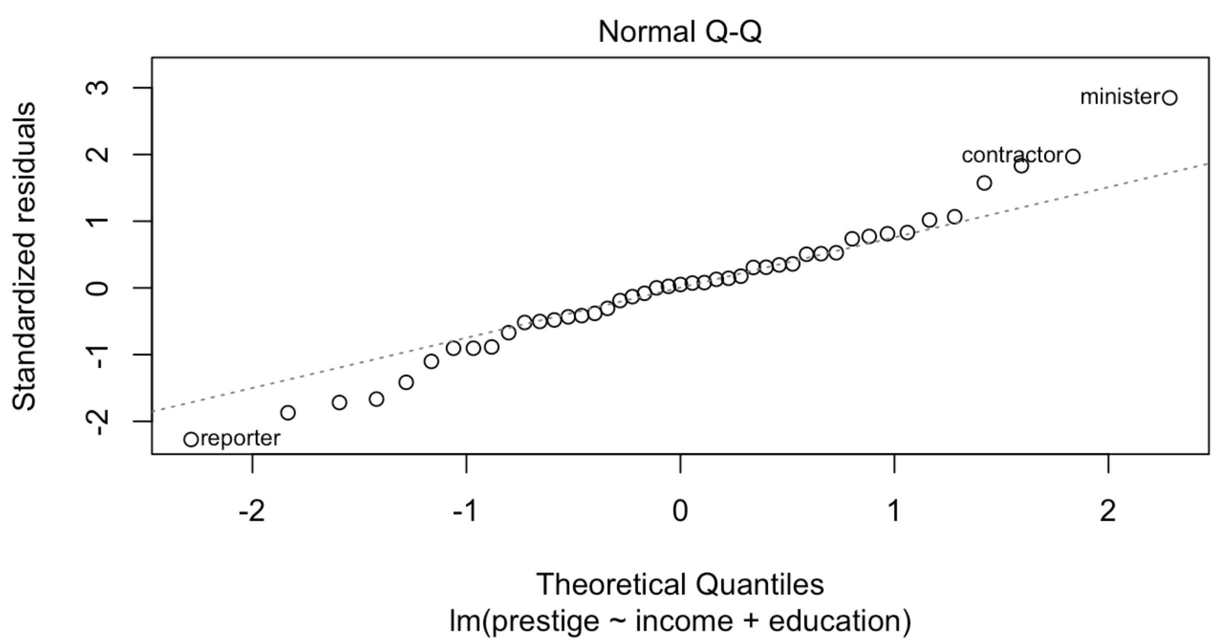
We load the data set and perform a regression.

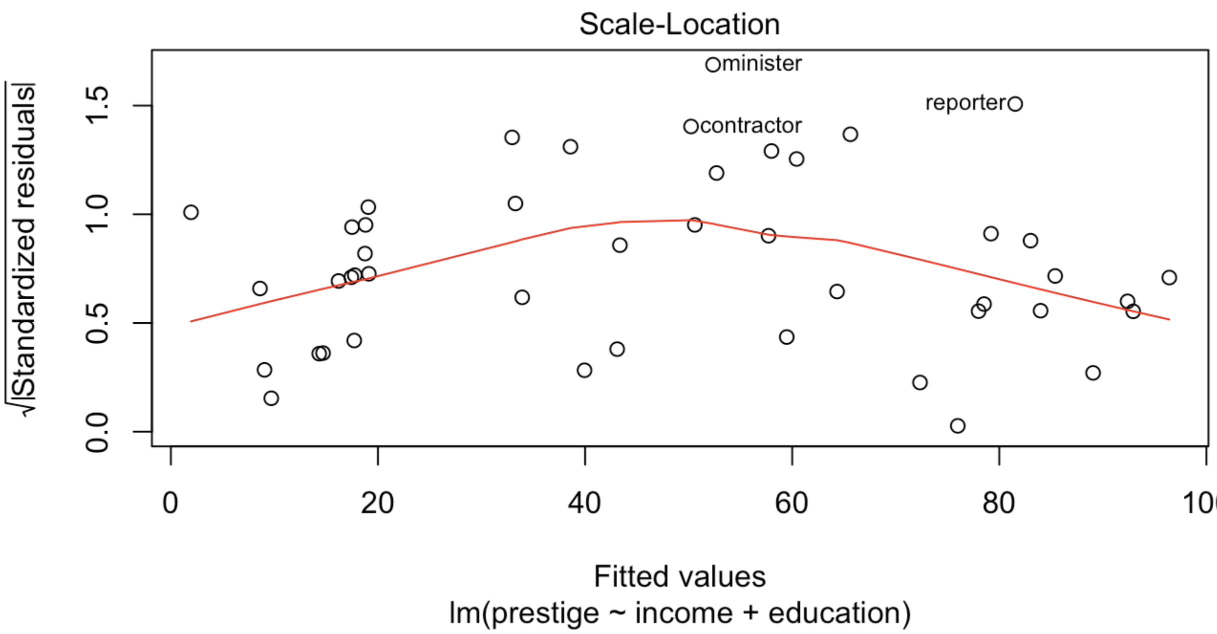
```
model <- lm(prestige ~ income + education, data=Duncan)
```

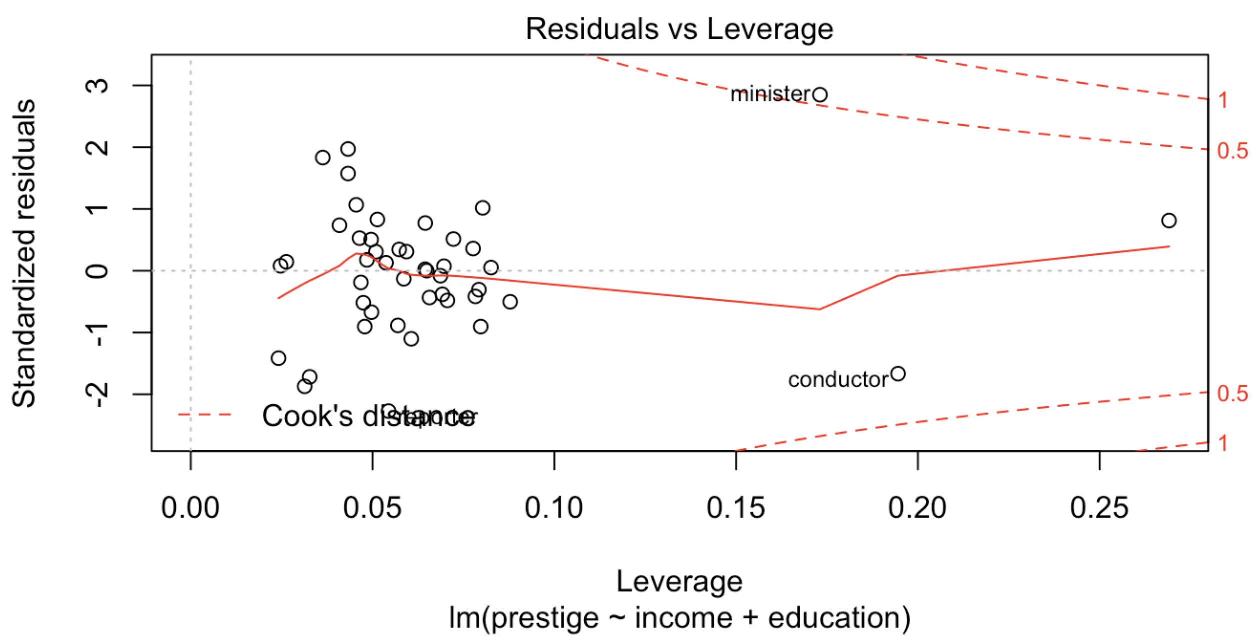
We can generate the standard plots.

```
plot(model)
```










```
linearHypothesis(model, "income = 0", test="Chisq")
```

```
## Linear hypothesis test
##
## Hypothesis:
## income = 0
##
## Model 1: restricted model
## Model 2: prestige ~ income + education
##
##   Res.Df     RSS Df Sum of Sq  Chisq Pr(>Chisq)
## 1     43 11980.9
## 2     42  7506.7  1    4474.2 25.033  5.635e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model2 <- lm(prestige ~ type*(income + education), data=Duncan)
coefs <- names(coef(model2))
```

Test against the null model (i.e., only the intercept is not set to 0).

```
linearHypothesis(model2, coefs[-1])
```

```
## Linear hypothesis test
##
## Hypothesis:
## typeprof = 0
## typewc = 0
## income = 0
## education = 0
## typeprof:income = 0
## typewc:income = 0
## typeprof:education = 0
## typewc:education = 0
##
## Model 1: restricted model
## Model 2: prestige ~ type * (income + education)
##
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1     44 43688
## 2     36  3351  8     40337 54.174 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Multivariate regression

We will look at a multivariate regression in the form

$$\mathbf{x}_i = \mathbf{B}\mathbf{z}_i + \epsilon_i, \quad i = 1, \ldots, n.$$

Generate some data.

```
n <- 100
c <- rbinom(n, 1, 0.2)
H <- rnorm(n, -10, 2)
A <- -1.4*c + 0.6*H + rnorm(n, 0, 3)
B <-  1.4*c - 0.6*H + rnorm(n, 0, 3)
X <- cbind(A, B)
```

We define the multivariate regression model. The inbuilt function `lm()` actually handles multivariate models.

```
model <- lm(X ~ c + H)
summary(model)
```

```
## Response A :
##
## Call:
## lm(formula = A ~ c + H)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.1544 -1.6405 -0.0511  2.2568  4.9527
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.6527     1.3587   0.480    0.632
## c             0.7997     0.6571   1.217    0.227
## H             0.7070     0.1337   5.289  7.6e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.761 on 97 degrees of freedom
## Multiple R-squared:  0.2293, Adjusted R-squared:  0.2134
## F-statistic: 14.43 on 2 and 97 DF,  p-value: 3.27e-06
##
##
## Response B :
##
## Call:
## lm(formula = B ~ c + H)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5396 -1.4619 -0.0807  1.5040  6.0065
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.5425     1.1747   0.462   0.6452
## c             1.4457     0.5681   2.545   0.0125 *
## H            -0.5831     0.1156  -5.046 2.11e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.387 on 97 degrees of freedom
## Multiple R-squared:  0.2561, Adjusted R-squared:  0.2407
## F-statistic:  16.7 on 2 and 97 DF,  p-value: 5.872e-07
```

We can also construct the design matrix X and compare to R’s design matrix.

```
Z  <- cbind(1, c, H)
ZR <- model.matrix(~ c + H)
all.equal(Z, ZR, check.attributes=FALSE)
```

```
## [1] TRUE
```

We can solve from first principles to find our coefficient matrix $\hat{\mathbf{B}}$ and compare it to R's.

```
Bhat <- solve(t(Z) %*% Z) %*% t(Z) %*% X
BhatR <- coef(model)
all.equal(Bhat, BhatR, check.attributes=FALSE)
```

```
## [1] TRUE
```

We can get our MLE estimate $\hat{\Sigma}$.

```
Sigmahat <- t(X - Z %*% Bhat) %*% (X - Z %*% Bhat) / n
Sigmahat
```

```
##             A           B
## A  7.39645812 -0.03706196
## B -0.03706196  5.52859643
```