Introduction to Regression - Chapter 4

MAP 535

Contents

Chapter 4: Model estimation/validation through R example

In this Chapter, we will see through an example how to estimate and validate a model. Moreover, we see how to detect atypical points.

4.1. The data

To illustrate the method, we consider the example~**??**.

> We try to explain and predict gasoline consumption (in liters per 100 km) of different automobile models based on several variables. For this, we have the following characteristics for 31 different cars:
>
> - `Type` = Type of the vehicle.
> - `Consommation` = Fuel consumption in liters per 100 km.
> - `Prix` = Vehicle price in Swiss francs.
> - `Cylindree` = Cylinder capacity in cm3.
> - `Puissance` = Power in kW.
> - `Poids` = Weight in kg.
>
> In this example, Y is the variable `Consommation`. The variables X_j correspond to the other 4 variables.

We first download the dataset `conso.txt` with the command `read.table`.

```
conso_voit = read.table("conso.txt", header=TRUE, sep="\t", dec=",",row.names=1)
conso_voit_complet = read.table("conso.txt", header=TRUE, sep="\t", dec=",")
```

To print the names of the variables is possible with the command `names`.

```
names(conso_voit_complet)
```

```
## [1] "Type"         "Prix"         "Cylindree"    "Puissance"
## [5] "Poids"        "Consommation"
```

To display the data in a table, we use the `knitr` library and the function `kable`.

```
library(knitr)
```

```
kable(conso_voit_complet)
```

Type	Prix	Cylindree	Puissance	Poids	Consommation
Daihatsu Cuore	11600	846	32	650	5.7
Suzuki Swift 1.0 GLS	12490	993	39	790	5.8
Fiat Panda Mambo L	10450	899	29	730	6.1
VW Polo 1.4 60	17140	1390	44	955	6.5
Opel Corsa 1.2i Eco	14825	1195	33	895	6.8
Subaru Vivio 4WD	13730	658	32	740	6.8
Toyota Corolla	19490	1331	55	1010	7.1
Ferrari 456 GT	285000	5474	325	1690	21.3
Mercedes S 600	183900	5987	300	2250	18.7
Maserati Ghibli GT	92500	2789	209	1485	14.5
Opel Astra 1.6i 16V	25000	1597	74	1080	7.4
Peugeot 306 XS 108	22350	1761	74	1100	9.0
Renault Safrane 2.2. V	36600	2165	101	1500	11.7
Seat Ibiza 2.0 GTI	22500	1983	85	1075	9.5
VW Golt 2.0 GTI	31580	1984	85	1155	9.5
Citroen ZX Volcane	28750	1998	89	1140	8.8
Fiat Tempra 1.6 Liberty	22600	1580	65	1080	9.3
Fort Escort 1.4i PT	20300	1390	54	1110	8.6
Honda Civic Joker 1.4	19900	1396	66	1140	7.7
Volvo 850 2.5	39800	2435	106	1370	10.8
Ford Fiesta 1.2 Zetec	19740	1242	55	940	6.6
Hyundai Sonata 3000	38990	2972	107	1400	11.7
Lancia K 3.0 LS	50800	2958	150	1550	11.9
Mazda Hachtback V	36200	2497	122	1330	10.8
Mitsubishi Galant	31990	1998	66	1300	7.6
Opel Omega 2.5i V6	47700	2496	125	1670	11.3
Peugeot 806 2.0	36950	1998	89	1560	10.8
Nissan Primera 2.0	26950	1997	92	1240	9.2
Seat Alhambra 2.0	36400	1984	85	1635	11.6
Toyota Previa salon	50900	2438	97	1800	12.8
Volvo 960 Kombi aut	49300	2473	125	1570	12.7

The `dim` command displays the size of the data (number of lines, number of columns). Here $n = 31$ and we have 5 variables.

```
dim(conso_voit_complet)
```

```
## [1] 31  6
```

The command `head` allows you to view the first 6 lines of the data. The command `tail` allows you to view the last 6 lines of the data.

```
head(conso_voit_complet)
```

```
##                   Type  Prix Cylindree Puissance Poids Consommation
## 1       Daihatsu Cuore 11600       846        32   650           5.7
## 2 Suzuki Swift 1.0 GLS 12490       993        39   790           5.8
## 3   Fiat Panda Mambo L 10450       899        29   730           6.1
## 4       VW Polo 1.4 60 17140      1390        44   955           6.5
## 5  Opel Corsa 1.2i Eco 14825      1195        33   895           6.8
## 6     Subaru Vivio 4WD 13730       658        32   740           6.8
```

The command `str` allows us to chek if the nature of each variable is well determined by **R**. Here, there is no mistake.

```
str(conso_voit_complet)
```

```
## 'data.frame':    31 obs. of  6 variables:
##  $ Type        : Factor w/ 31 levels "Citroen ZX Volcane",..: 2 25 4 31 17 24 26 3
##  $ Prix        : int  11600 12490 10450 17140 14825 13730 19490 285000 183900 92500
##  $ Cylindree   : int  846 993 899 1390 1195 658 1331 5474 5987 2789 ...
##  $ Puissance   : int  32 39 29 44 33 32 55 325 300 209 ...
##  $ Poids       : int  650 790 730 955 895 740 1010 1690 2250 1485 ...
##  $ Consommation: num  5.7 5.8 6.1 6.5 6.8 6.8 7.1 21.3 18.7 14.5 ...
```

4.2. Descriptive dataset analysis

A very useful command is the function `summary`. Here this command gives us a summary of the datas (average, quantiles, . . .)

```
summary(conso_voit)
```

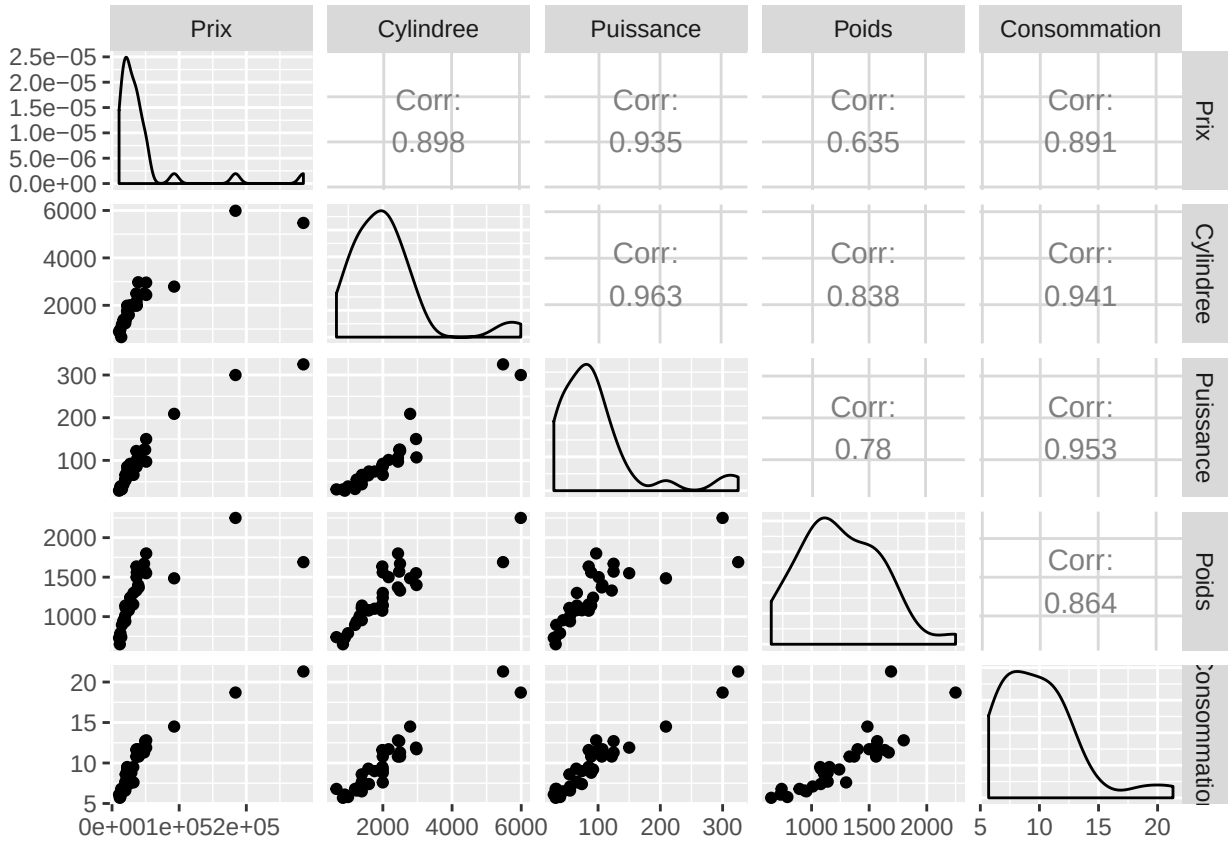
```
##       Prix          Cylindree      Puissance         Poids
##  Min.   : 10450   Min.   : 658   Min.   : 29.0   Min.   : 650
##  1st Qu.: 19820   1st Qu.:1390   1st Qu.: 55.0   1st Qu.:1042
##  Median : 28750   Median :1984   Median : 85.0   Median :1155
##  Mean   : 43756   Mean   :2094   Mean   : 97.1   Mean   :1256
##  3rd Qu.: 39395   3rd Qu.:2456   3rd Qu.:106.5   3rd Qu.:1525
##  Max.   :285000   Max.   :5987   Max.   :325.0   Max.   :2250
##   Consommation
##  Min.   : 5.700
##  1st Qu.: 7.250
##  Median : 9.300
##  Mean   : 9.955
##  3rd Qu.:11.650
##  Max.   :21.300
```

Some useful packages.

```
library(ggplot2)
library(dplyr)
library(GGally)
```

To visualize the relation between each pairs of variables, we use the commande `ggpairs`.

```
ggpairs(conso_voit)
```



It seems to be a linear link between variable `Consommation` and the others variables. On the diagonal, there are plot of the estimated density of each variable. Above the diagonal, it is given the value of the correlations between variables. The command `cor` also gives the correlations.

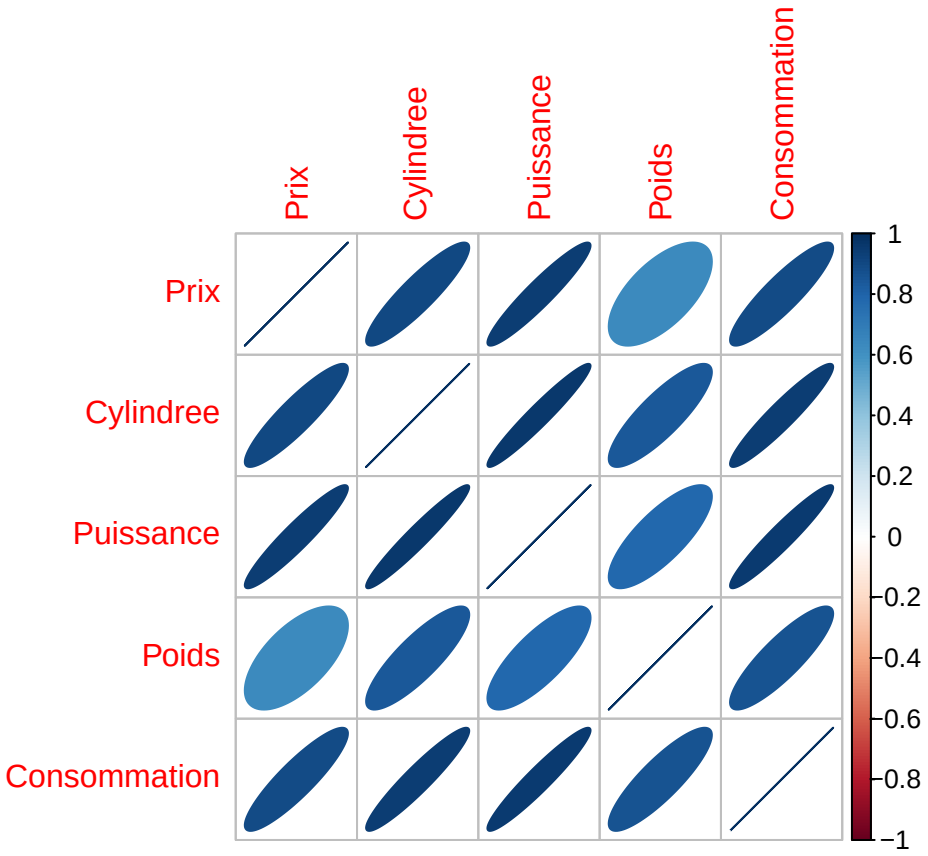
```
library(corrplot)
```

```
cor(conso_voit)
```

```
##                    Prix Cylindree Puissance     Poids Consommation
## Prix          1.0000000 0.8977790 0.9351708 0.6349611    0.8911104
## Cylindree     0.8977790 1.0000000 0.9625134 0.8378676    0.9409920
## Puissance     0.9351708 0.9625134 1.0000000 0.7798228    0.9526249
## Poids         0.6349611 0.8378676 0.7798228 1.0000000    0.8638623
## Consommation  0.8911104 0.9409920 0.9526249 0.8638623    1.0000000
```

A graphical visualization of the correlations is easier to interpret graphically by using the function `corrplot` of the library `corrplot`).

```
r=round(cor(conso_voit),2)
corrplot(r,method="ellipse")
```



The `Consommation` is very correlated with the 4 others variables. Note that the variable `Cylindree` and `Puissance` are highly correlated.

4.3. The linear regression model

The linear regression of the `Consommation` variable on the other variables is done using the `lm` function.

```
reg = lm(Consommation~Prix+Cylindree+Puissance+Poids,data=conso_voit)
```

Comments:

- ☛ By default **R** adds an intercept (a column of 1). Here, the design matrix X is a matrix of size $n \times p$ with $p = 5$.
- ☛ The order in which variables are entered gives the indice j of the regressor X_j
  - Y =`Consommation` = Fuel consumption in liters per 100 km.

- X_1 =`Prix` = Vehicle price in Swiss francs.
- X_2 =`Cylindree` = Cylinder capacity in cm3.
- X_3 =`Puissance` = Power in kW.
- X_4 =`Poids` = Weight in kg.

☛ Then, the linear model defined here is

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \beta_3 X_{i3} + \beta_4 X_{i4} + \varepsilon_i, \ \forall i = 1, \cdots, n. \tag{1}$$

☛ We assume that the postulates **[P1]**–**[P4]** are statisfied.

We then visualize the results using the function `summary`.

```
summary(reg)
```

```
##
## Call:
## lm(formula = Consommation ~ Prix + Cylindree + Puissance + Poids,
##     data = conso_voit)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5677 -0.6704  0.1183  0.5283  1.4361
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.456e+00  6.268e-01   3.919 0.000578 ***
## Prix         2.042e-05  8.731e-06   2.339 0.027297 *
## Cylindree   -5.006e-04  5.748e-04  -0.871 0.391797
## Puissance    2.499e-02  9.992e-03   2.501 0.018993 *
## Poids        4.161e-03  8.788e-04   4.734 6.77e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8172 on 26 degrees of freedom
## Multiple R-squared:  0.9546, Adjusted R-squared:  0.9476
## F-statistic: 136.5 on 4 and 26 DF,  p-value: < 2.2e-16
```

Interpretation of R outputs

➤ **`Call`** : A reminder of the formula used.

➤ **`Residuals`** : A summary descriptive analysis of residues $\widehat{\varepsilon_i}$.

➤ **`Coefficients`** : This table includes in columns:

- **`Estimate`** : The value of $\widehat{\beta}_j$ the least square estimator $\widehat{\beta}$ (which is the maximum likelihood estimator under **[P1]**–**[P4]**). Here

$$\widehat{\beta}_0 = 2.456e{+}00, \quad \widehat{\beta}_1 = 2.042e{-}05, \quad \widehat{\beta}_2 = -5.006e{-}04, \quad \widehat{\beta}_3 = 2.499e{-}02, \quad \widehat{\beta}_4 = 4.161e{-}03$$

- **`Std. Error`** : The value of $\widehat{\sigma}_j = \widehat{\mathbb{V}\mathrm{ar}_\beta(\widehat{\beta}_j)}$, estimator of the standard deviation of $\widehat{\beta}_j$.
- **`t value`** : Here we test

$$H_0 : \beta_j = 0 \quad \text{vs} \quad H_1 : \beta_j \neq 0.$$

  The `t value` is the value of the Student test statistic T, such that under H_0

$$T = \frac{\widehat{\beta}_j}{\widehat{\sigma}\sqrt{(X^TX)^{-1}_{jj}}} \sim t_{n-p}$$

  with $(X^TX)^{-1}_{jj}$ the $j-th$ diagonal element of the matrix $(X^TX)-1$.
- **`Pr(>|t|)`** : The $p-value$ of the previous Student tests.

➤ **`Signif. codes`** : Significance level symbols.

➤ **`Residual standard error`** : The value of $\widehat{\sigma}$ and the number of degrees of freedom : $(n-p)$ (here $31-5=26$). Here

$$\widehat{\sigma}^2 = 0.8172^2$$

➤ **`Multiple R-squared`** : The value of $R^2 = 0.9546$.

➤ **`Adjusted R-squared`** : The value of the adjusted R^2 : $R^2_a = 0.9476$.

➤ **`F-statistic`** : Here we test

$$H_0 : Y_i = \beta_0 + \varepsilon_i \quad \text{vs} \quad H_1 : Y_i = \beta_0 + \sum_{j=1}^{4} \beta_j X_{ij} + \varepsilon_i.$$

The `F-statistic` is the value of the Fisher's global test statistic F such that under H_O

$$F = \frac{\|P_X Y - \overline{Y}\mathbf{1}\|^2/(p-1)}{\widehat{\sigma}^2} \sim F_{(p-1,n-p)}.$$

In this example, $F = 136.5$ and the associated degrees of freedom $(p-1, n-p) = (4, 26)$. The $p-value < 2.2e-16$ is very small so we reject H_0, the test is meaningful.

4.4. Model validation

We recall that we assume the model~(1), under the Rank assumption and under **[P1]**–**[P4]** where

- **[P1]** : Errors are centered/(the model is linear) : $\forall i = 1, \cdots, n \quad \mathbb{E}_\beta[\varepsilon_i] = 0$.
- **[P2]** : Errors have homoscedastic variance : $\forall i = 1, \cdots, n \quad \mathbb{V}\text{ar}_\beta[\varepsilon_i] = \sigma^2 > 0$.
- **[P3]** : Errors are uncorrelated: $\forall i \neq j \quad \mathbb{C}\text{ov}(\varepsilon_i, \varepsilon_j)$.
- **[P4]** : Errors are gaussian : $\forall i = 1, \cdots, n \quad \varepsilon_i \sim \mathcal{N}(0, \sigma^2)$.

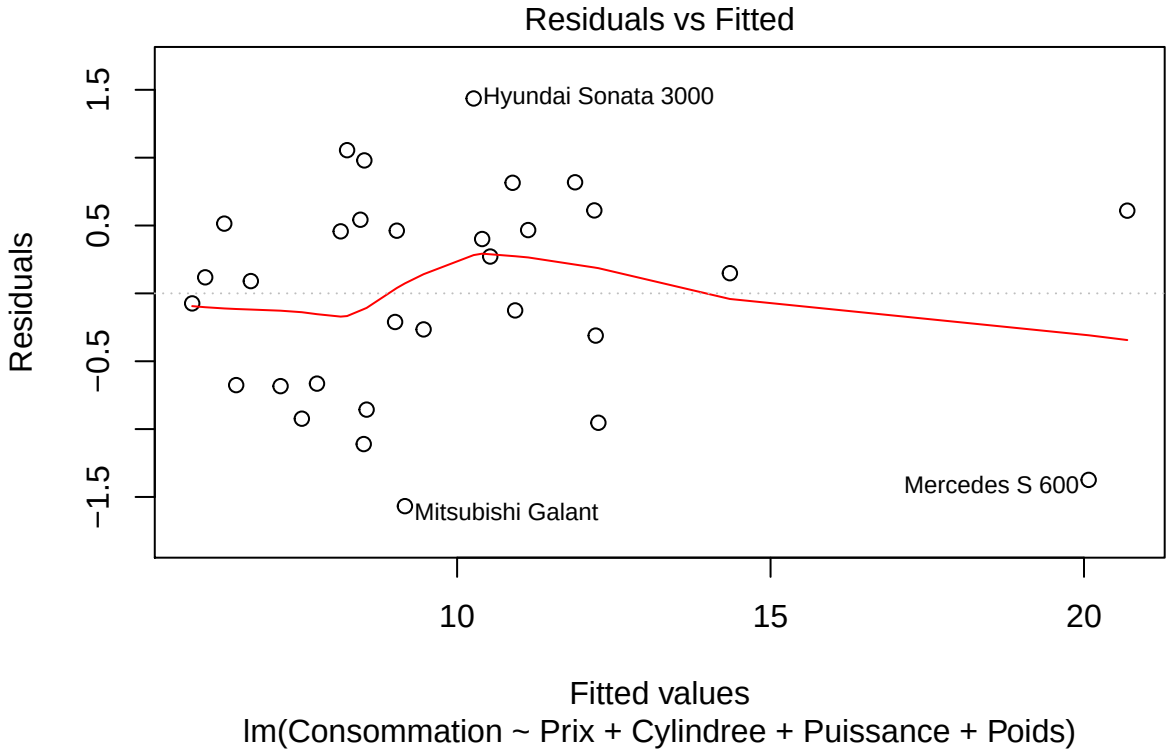
The rank hypothesis is easily verifiable with a simple calculation. For the other assumptions **[Pi]**, this requires an analysis of the residus. First upload the needed **R** library.

```
library(MASS)
library(carData)
library(car)
```

Validation of the postulates [P1]: Errors are centered

The centered postulat (the linearity assumption in practice) can be assesed by inspecting the *Residuals vs Fitted*-plot (or the *Studentized residuals*-plots). The command for the *Residuals vsFitted*-plot is `plot( ,which=1)`.

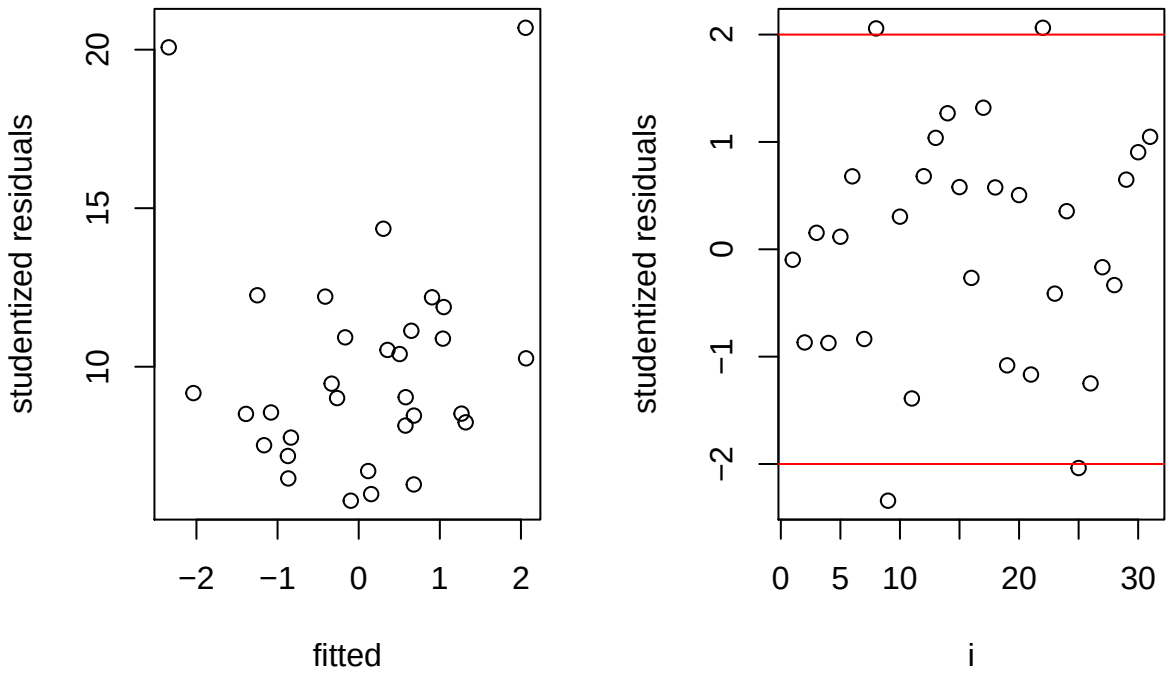
```
plot(reg,which=1)
```



The plot shows that when the responses predicted by the model (fitted values) increase, the residuals remain globally uniformly distributed on both sides of 0. The red line is approximately horizontal at zero.

The command `stdres()` displays the *Studentized residuals* $(t_i^*)_i$, needed to draw the *Studentized residuals*-plot.

```
par(mfrow=c(1,2))
SR=stdres(reg)
plot(SR,fitted(reg),xlab="fitted",ylab="studentized residuals")
plot(SR,xlab="i",ylab="studentized residuals")
abline(h=2,col="red")
abline(h=-2,col="red")
```

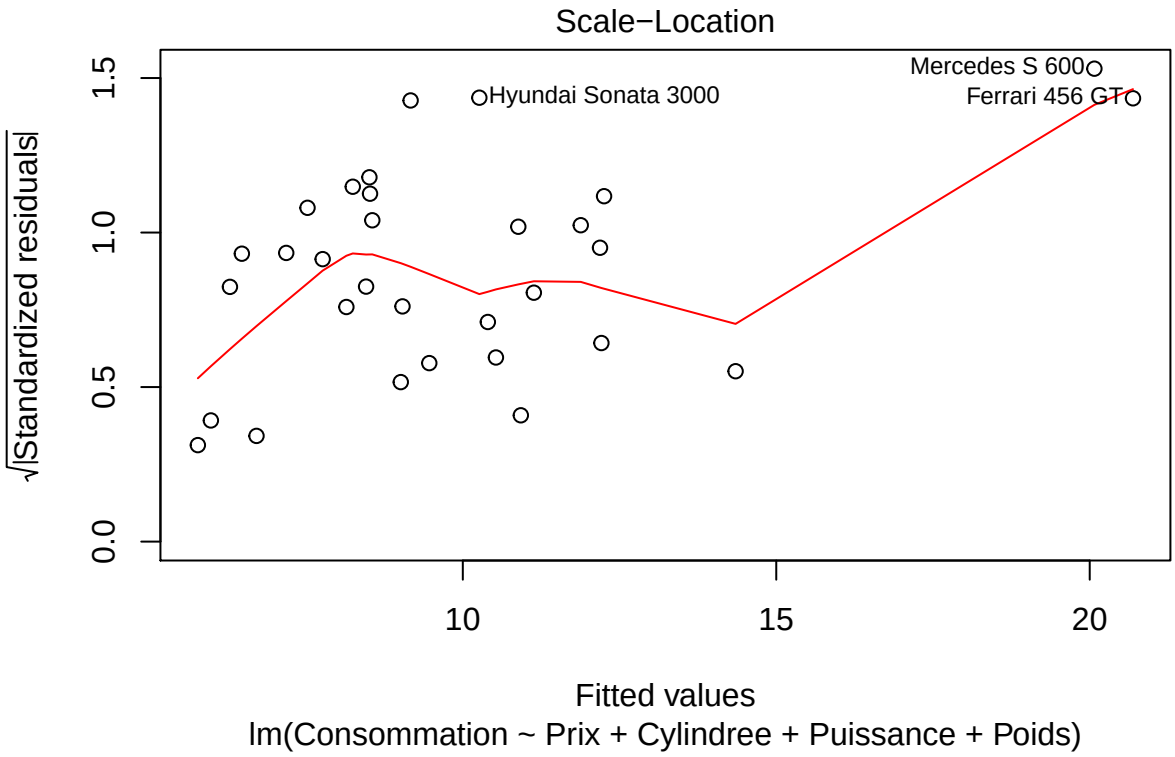


The plots show no fitted pattern, the residuals remain globally (reasonably) uniformly distributed. therefore that the assumption of linearity is acceptable. Thus, we validate the postulate.

Validation of the postulates [P2]: Errors have homoscedastic variance

The homoscedastic assumption can be checked by examining the *Scale-location*-plot (the command `plot( ,which=3)`). The postulate is validated if we see a horizontal line with equally spread points. In our example, it seems difficult to validate the postulat. So, let us make a Breush-Pagan test (H_O : homoscedasticity) to assess it.

```
plot(reg,which=3)
```



The command for the Breush-Pagan test is `ncvTest`. The homoscedasticity is rejected if the *p-value* is less than 0.05. Here, *p-value*= 0.38455 > 0.05, the posutlate is validated.

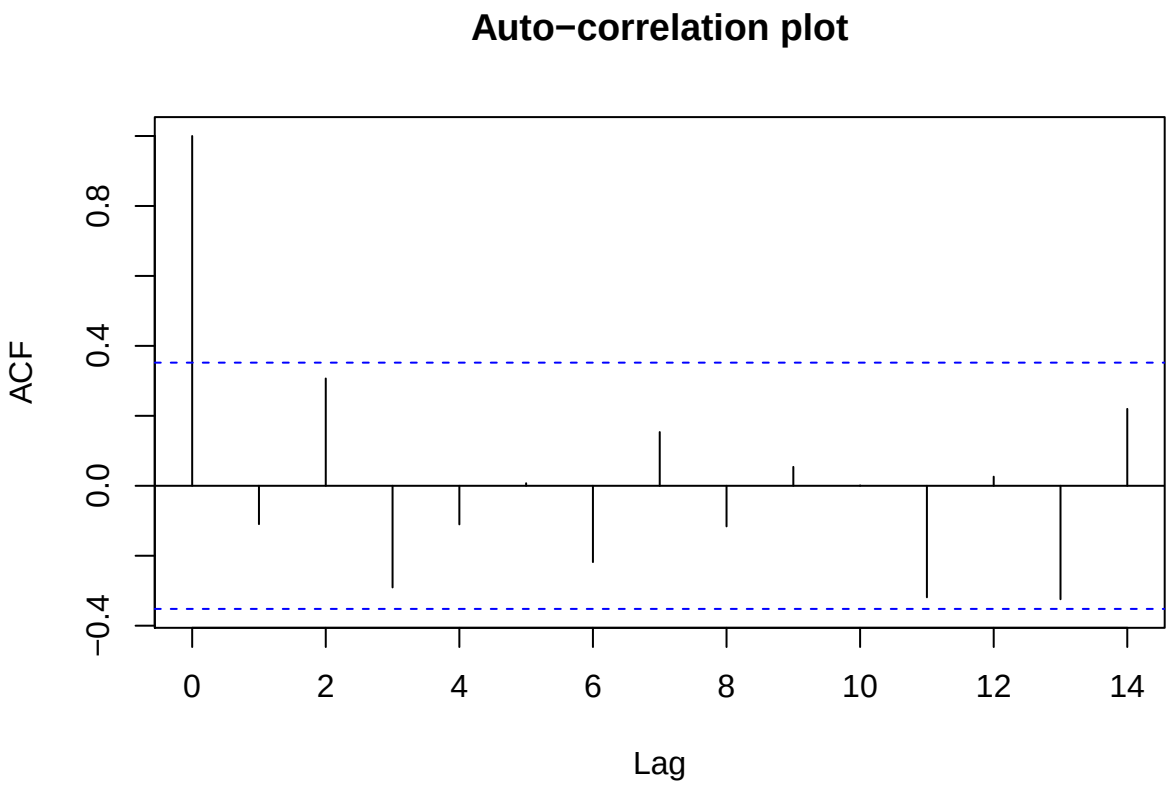
```
ncvTest(reg)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 0.7560996, Df = 1, p = 0.38455
```

Validation of the postulate [P3]: Errors are uncorrelated

Under **R**, we can represent the auto-correlation of the residuals using the command `acf()`. In our example, exept the first one, none exceeds dashed thresholds thus uncorrelation is satisfied.

```
acf(residuals(reg),main="Auto-correlation plot")
```



The Durbin-Watson test can be also used to validate this assumption. The command is `durbinWatsonTest`. Under the null hypothesis the residual are considered auto-uncorrelated.

```
durbinWatsonTest(reg)
```

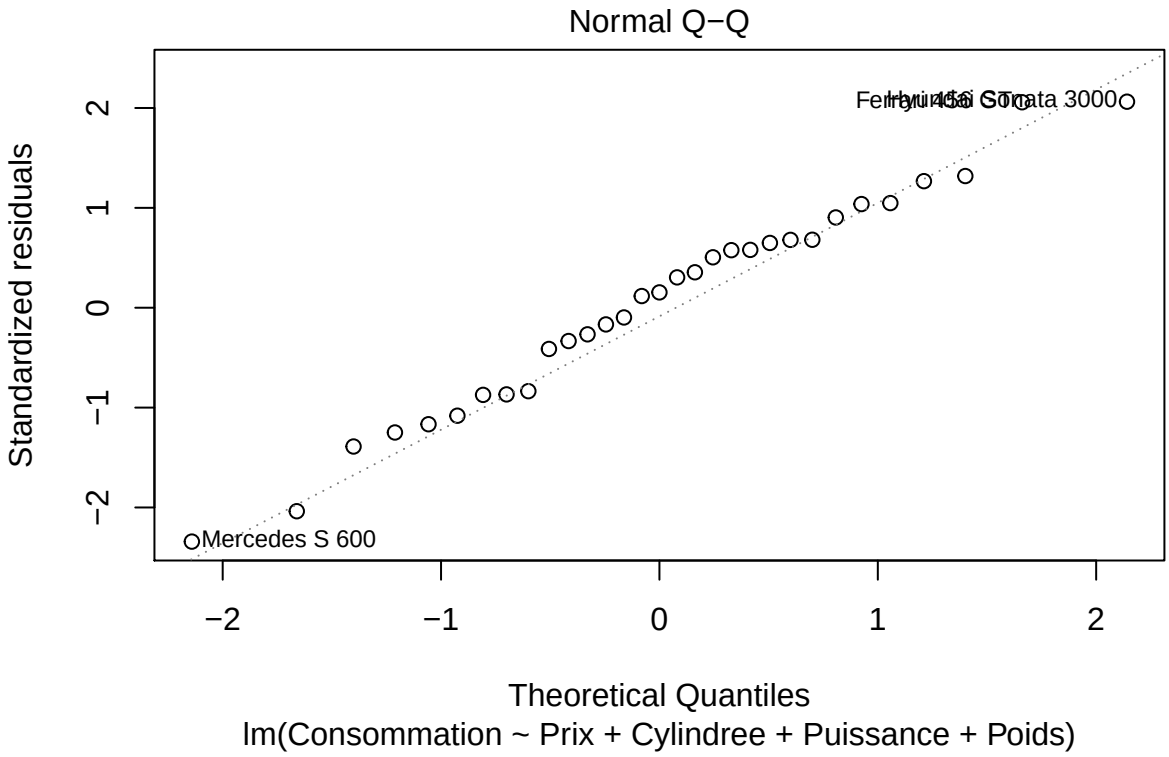
```
##  lag Autocorrelation D-W Statistic p-value
##    1      -0.1096954      2.180495   0.764
##  Alternative hypothesis: rho != 0
```

Here, the *p-value*= 0.782 > 0.05 thus we can't reject H_0, the posutlate is validated.

Validation of the postulate [P4]: Errors are gaussian

To analyze the normality, we use the Q-Q plot with the command `plot( ,which=2)`. The points appear reasonably aligned along the reference line even the sample size $n = 31$ is small, then the postulate is validated.

```
plot(reg,which=2)
```



The Shapiro-Wilk test can also be used to assess the normality of residuals. The normality assumption is rejected if the *p-value* is less than 0.05. Here, *p-value*> 0.05, the posutlate is unvalidated. But as the sample size is small, it was expected. However, we assume that the postulate is verified.

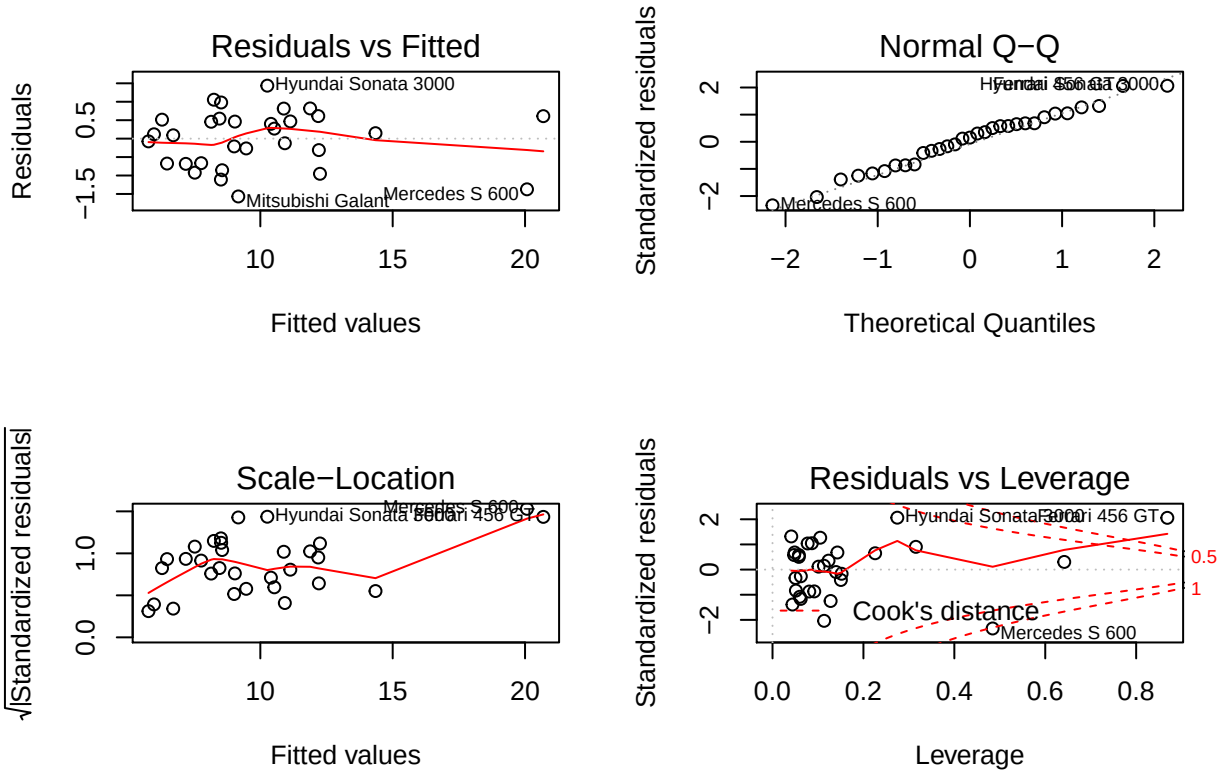
```
shapiro.test(residuals((reg)))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals((reg))
## W = 0.9709, p-value = 0.5442
```

Command `plot`: to resume

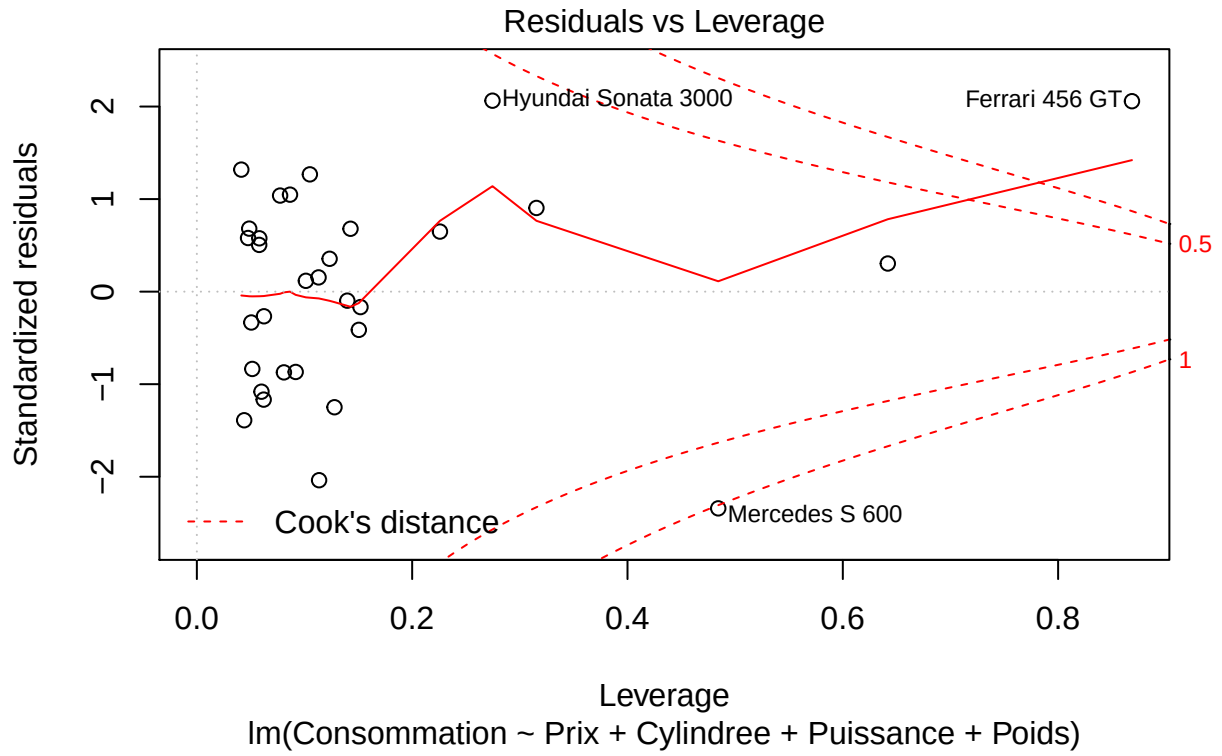
Note that by default, the command `plot)` gives 4 different plots in the linear regression setting. Three of them have been seen and used to validate the postulat. The last one is the `Residuals vs Leverage`-plot.

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



The `Residuals vs Leverage`-plot can be called alone as follows :

```
plot(reg,which=5)
```

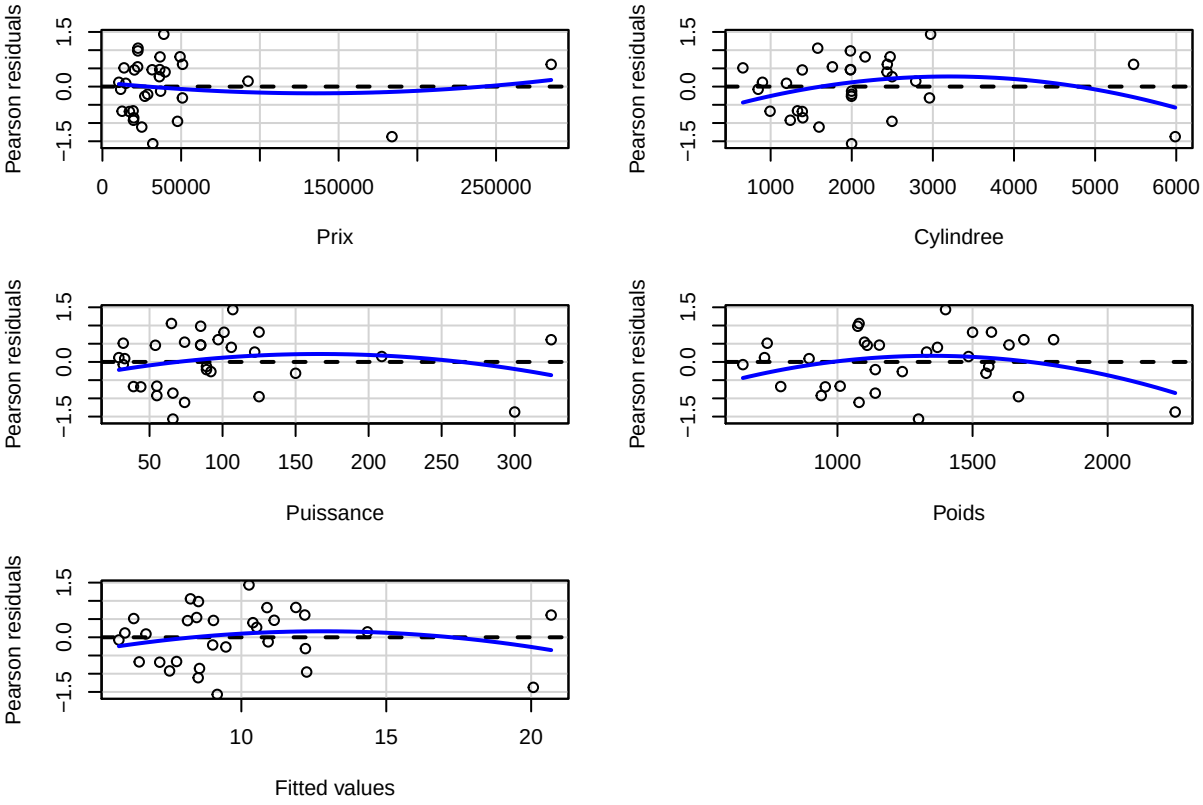


It appears in this plot, that 2 observations have a Cook's distance larger than 1. They are *ouliers* : *regression ouliers* or *leverage points* or both. A sudy of ouliers is done in a further section.

For going further...

For going further, to see the contribution of each variable, we can use the `residualPlots` function from the `cars` library. (it will be discussed in classroom)?

```
residualPlots(reg)
```




```
##             Test stat Pr(>|Test stat|)
## Prix           1.1523          0.26009
## Cylindree     -2.2748          0.03176 *
## Puissance     -2.4246          0.02289 *
## Poids         -1.6631          0.10878
## Tukey test    -2.1976          0.02798 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4.5 Confidence Interval

The `confint` command easily displays confidence intervals for the parameters β_j of the model. They are based on a Student's law, if the postulate **[P4]** is satisfied. If not, these intervals are biased.

```
cbind(confint(reg),coef(reg))
```

```
##                      2.5 %       97.5 %
## (Intercept)  1.167851e+00 3.744737e+00  2.456294e+00
## Prix         2.474392e-06 3.836669e-05  2.042054e-05
## Cylindree   -1.682157e-03 6.809703e-04 -5.005933e-04
## Puissance    4.455929e-03 4.553302e-02  2.499448e-02
## Poids        2.354210e-03 5.966955e-03  4.160583e-03
```

The `predict( ,interval = "confidence")` command displays confidence interval for $x_i^T\beta$ (say, for *estimation*); while the `predict( ,interval = "prediction")` command displays confidence interval for Y_i (say, for *prediction*). Note that $x_i^T\beta$ and Y_i are both estimate by $x_i^T\widehat{\beta}$, but the bound of the confidence interval are differents.

```
ICconf = predict(reg, interval = "confidence", level = 0.95)
head(ICconf)
```

```
##                           fit      lwr      upr
## Daihatsu Cuore       5.773872 5.145857 6.401888
## Suzuki Swift 1.0 GLS 6.475902 5.966890 6.984914
## Fiat Panda Mambo L   5.981720 5.416875 6.546566
## VW Polo 1.4 60       7.183591 6.705853 7.661329
## Opel Corsa 1.2i Eco  6.709359 6.174759 7.243959
## Subaru Vivio 4WD     6.285932 5.651261 6.920603
```

```
ICpred= predict(reg, interval = "prediction", level = 0.95)
```

```
## Warning in predict.lm(reg, interval = "prediction", level = 0.95): predictions on c
```

```
head(ICpred)
```

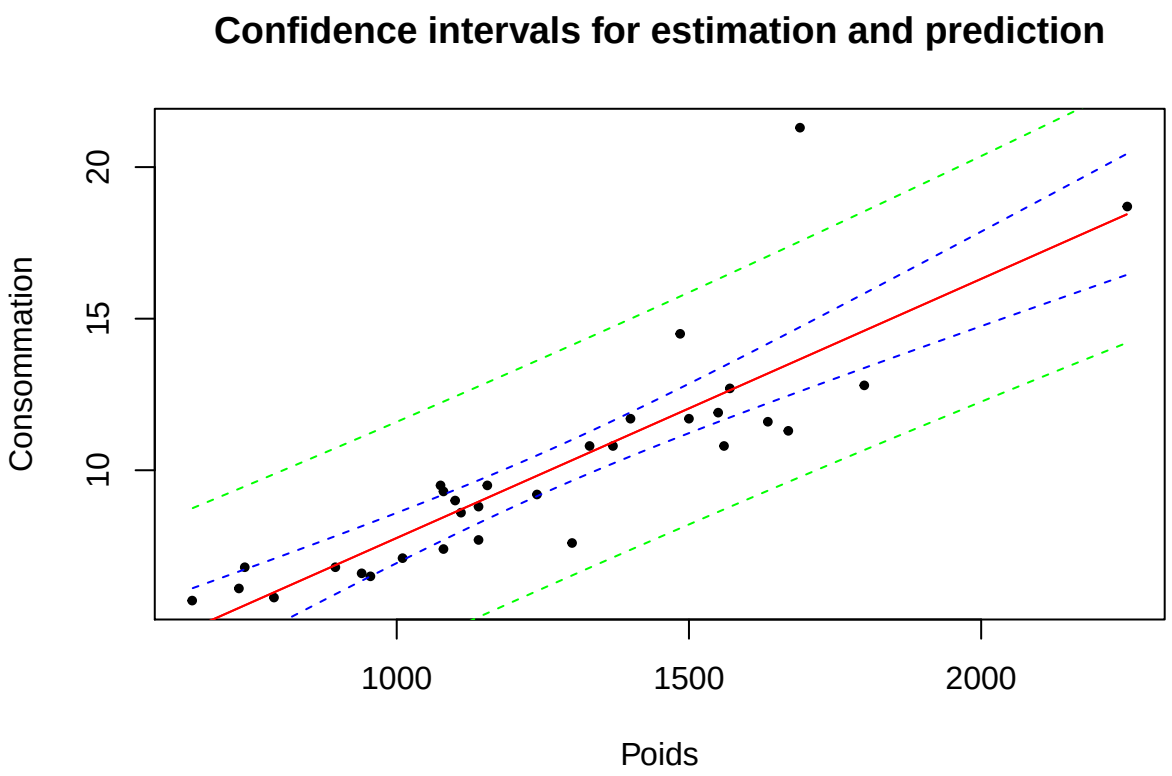
```
##                           fit      lwr      upr
## Daihatsu Cuore       5.773872 3.980461 7.567284
## Suzuki Swift 1.0 GLS 6.475902 4.720620 8.231184
## Fiat Panda Mambo L   5.981720 4.209442 7.753999
## VW Polo 1.4 60       7.183591 5.437121 8.930060
## Opel Corsa 1.2i Eco  6.709359 4.946486 8.472231
## Subaru Vivio 4WD     6.285932 4.490179 8.081685
```

Let us consider the simple regression of the variable `Consommation` on the predictor `Poids`, and consider new data `Poids` (called `newgrille` in our program). The command `predict.lm( ,interval = 'confidence')` displays prediction of the new $x_{i_{new}}^T\beta$ and the command `predict.lm( ,interval = 'prediction')` displays prediction of the new $Y_{i_{new}}$.

```
regP=lm(Consommation~Poids,data=conso_voit)
newgrille=data.frame(Poids=seq(min(conso_voit$Poids)+1,max(conso_voit$Poids)-1),2)
predicgrille=predict.lm(regP,newgrille, interval='confidence',level=0.95)
head(predicgrille)
```

```
##        fit      lwr      upr
## 1 4.783165 3.455416 6.110914
## 2 4.791711 3.465595 6.117828
## 3 4.800257 3.475773 6.124742
## 4 4.808804 3.485950 6.131658
## 5 4.817350 3.496126 6.138574
## 6 4.825897 3.506302 6.145491
```

```
plot(conso_voit$Poids,conso_voit$Consommation,ylab="Consommation",xlab="Poids",pch=20,
#abline(regP,col='red')
matlines(newgrille$Poids,predicgrille,lty=c(1,2,2),col=c("red","blue","blue"))
predicgrille=predict.lm(regP,newgrille, interval='prediction',level=0.95)
matlines(newgrille$Poids,predicgrille,lty=c(1,2,2),col=c("red","green","green"))
```

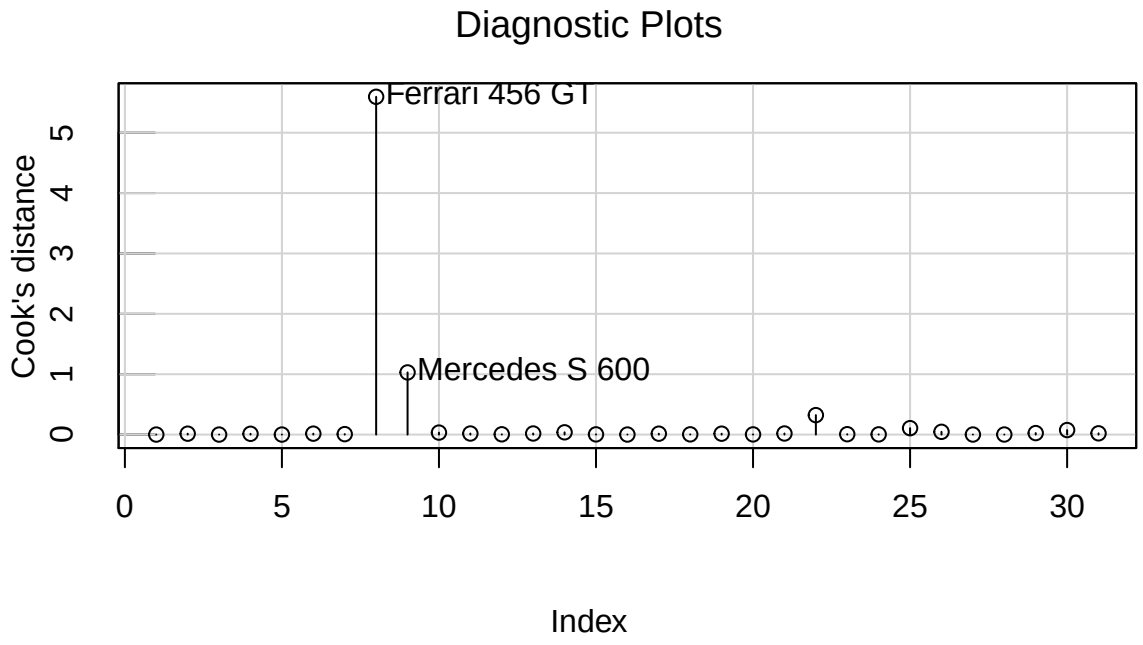


4.6. Outliers and leverage points

The library `car` offers an easy way to detect graphically atypical observations and to assess about their nature by tests. The command `influenceIndexPlot` is an important one.

Cook's distance plot

```
influenceIndexPlot(reg,vars="Cook")
```

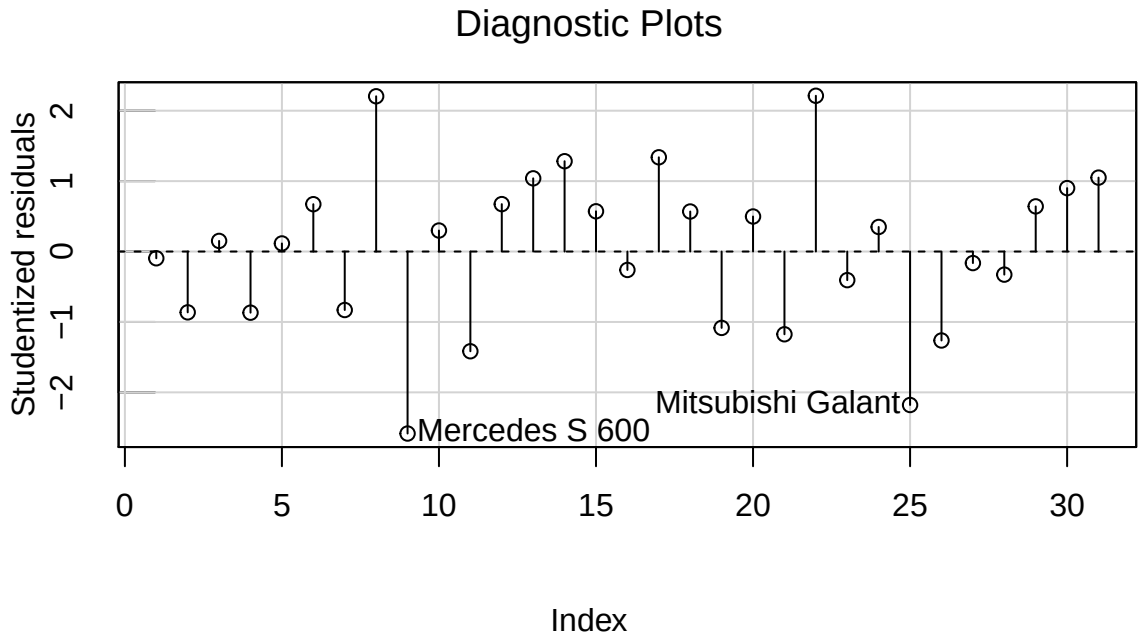


The Cook's distance plot highlight the influence of each observation on the estimation of the model (on β). As seen on the previous chapter, we compare the Cook's distance with 1. Here, two observations have a Cook's distance larger than 1 :

`Ferrari 456 GT` and `Mercedes S 600`

Studentized plot

```
influenceIndexPlot(reg,vars="Studentized")
```

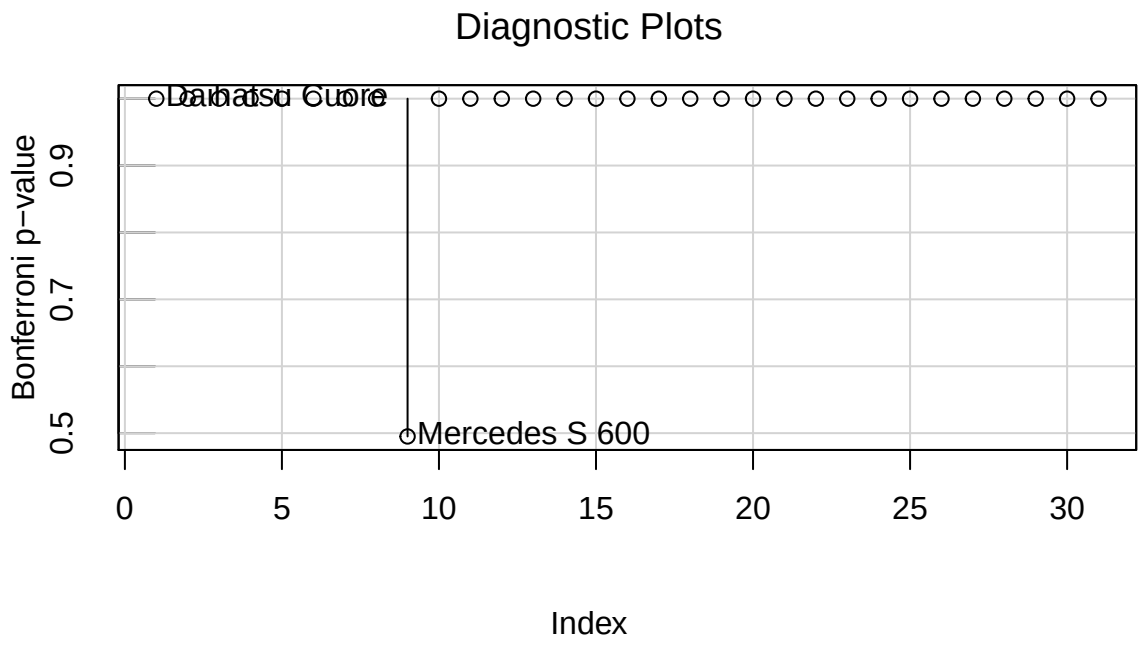


This plot, that of studentized residuals also makes it possible to highlight `outliers`. Here, two observations seems doubtful :

`Mitsubishi Galant` and `Mercedes S 600`

Bonferroni plot

```
influenceIndexPlot(reg,vars="Bonf")
```

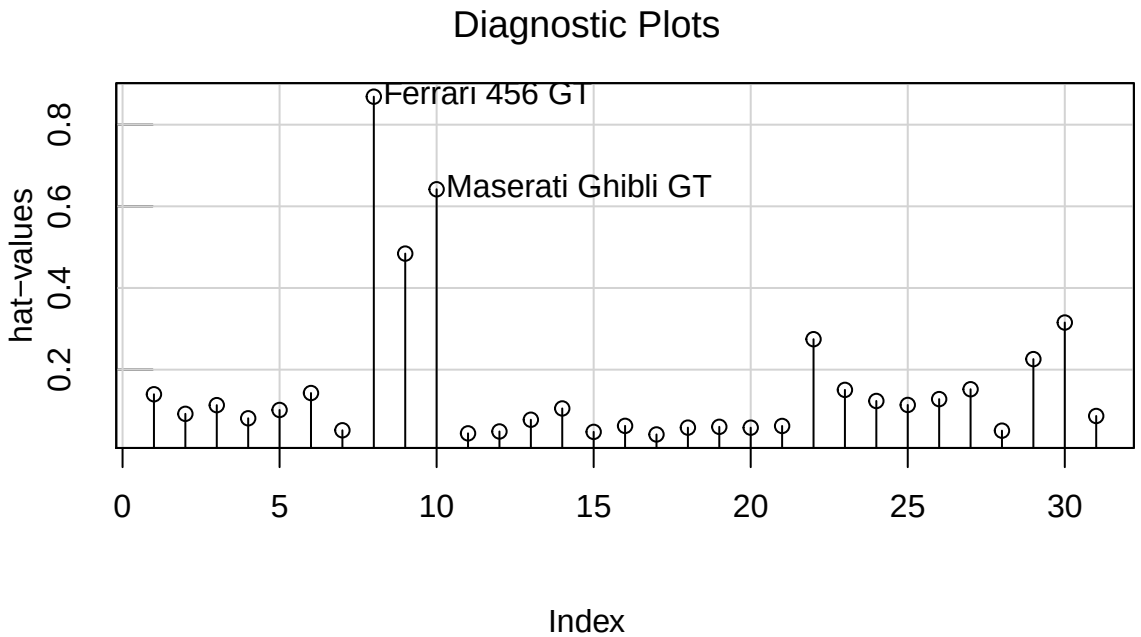


This is the plot of the *p-value* Bonferroni. Is considered as an outlier is a observation with a p-value less than 0.05. Here, the plot detetects one observation :

```
Mercedes S 600
```

Hat plot

```
influenceIndexPlot(reg,vars="hat")
```



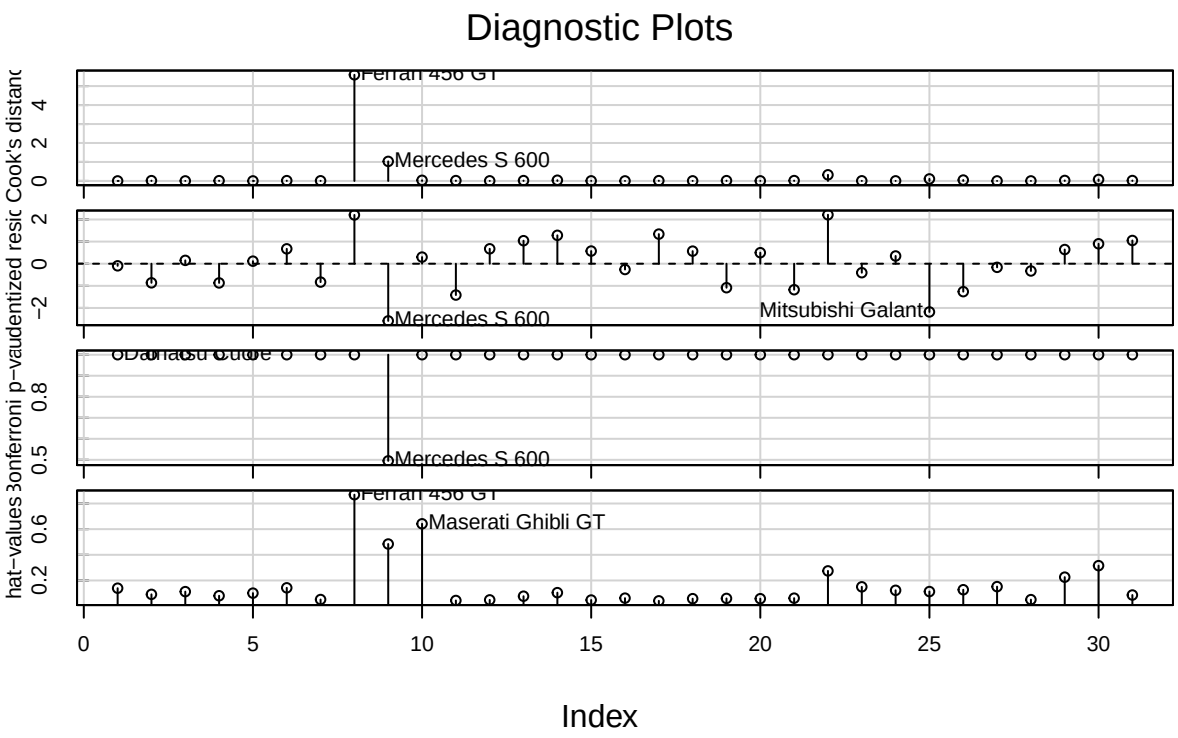
This plot called the *hat value*-plot, reflects the leverage (h_{ii}) of each observation on its own estimate. An observation is considered to be a *leverage point* when this value is less than 0.05. Here, the plot detetects two observations :

`Ferrari 456 GT` and `Maserati Ghibli GT`

The all plots

It's better to display these four graphs in parallel to have a better vision of the atypical points found in the various plot. It can be done as follows :

```
influenceIndexPlot(reg)
```



Here, the doubtful observations are :

`Mercedes S 600` and `Ferrari 456 GT`

We can access to Bonferroni's with the `outlierTest` command.

```
outlierTest(reg)
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##                 rstudent unadjusted p-value Bonferroni p
## Mercedes S 600 -2.584781           0.01597      0.49506
```

The adjusted *p-value* by the Bonferonni method is equal to 0.49506 and is very far from the threshold of 0.05. The `Mercedes S 600` observation can not be considered as `outlier`.

To assess if the observations `Ferrari 456 GT` and `Mercedes S 600` really affect the estimation ouf our model (say β), we can compare the results of the estimation of β with and without these observation. We can done this with the command `comparCoefs`.

```
regbis = lm(Consommation~Prix + Cylindree + Puissance + Poids, data=
               conso_voit[-c(which(conso_voit_complet$Type=="Ferrari 456 GT"),
                             which(conso_voit_complet$Type=="Mercedes S 600")),])
compareCoefs(reg ,regbis)
```

```
## Calls:
## 1: lm(formula = Consommation ~ Prix + Cylindree + Puissance + Poids,
##   data = conso_voit)
## 2: lm(formula = Consommation ~ Prix + Cylindree + Puissance + Poids,
##   data = conso_voit[-c(which(conso_voit_complet$Type ==
##   "Ferrari 456 GT"), which(conso_voit_complet$Type == "Mercedes S 600")),
##    ])
##
##                Model 1   Model 2
## (Intercept)      2.456     2.071
## SE               0.627     0.664
##
## Prix          2.04e-05  2.56e-05
## SE            8.73e-06  3.03e-05
##
## Cylindree    -0.000501  0.000327
## SE            0.000575  0.000700
##
## Puissance      0.02499   0.01777
## SE             0.00999   0.01554
##
## Poids         0.004161  0.003598
## SE            0.000879  0.001054
##
```

It comes out that, that the observations `Ferrari 456 GT` and `Mercedes S 600` have little influence on the coefficients of the model parameters, as well as on their standard error, since the values do not vary much, even if they have a Cook's distance larger than 1.

```
#summary(regbis)
#plot(regbis)
```