Laboratorio con `R` - 1

Metodi e Modelli per l'Inferenza Statistica - Ing. Matematica - a.a. 2018-19

17/05/2018

---

0. Required packages

```
library( car )
## Warning: package 'car' was built under R version 3.4.4
## Loading required package: carData
## Warning: package 'carData' was built under R version 3.4.4
library( ellipse )
## Warning: package 'ellipse' was built under R version 3.4.4
##
## Attaching package: 'ellipse'
## The following object is masked from 'package:car':
##
##     ellipse
## The following object is masked from 'package:graphics':
##
##     pairs
library( faraway )
## Warning: package 'faraway' was built under R version 3.4.4
##
## Attaching package: 'faraway'
## The following objects are masked from 'package:car':
##
##     logit, vif
library( leaps )
## Warning: package 'leaps' was built under R version 3.4.4
library( qpcR )
## Warning: package 'qpcR' was built under R version 3.4.4
## Loading required package: MASS
## Loading required package: minpack.lm
## Warning: package 'minpack.lm' was built under R version 3.4.4
## Loading required package: rgl
## Warning: package 'rgl' was built under R version 3.4.4
## Loading required package: robustbase
## Warning: package 'robustbase' was built under R version 3.4.4
##
## Attaching package: 'robustbase'
## The following object is masked from 'package:faraway':
##
##     epilepsy
## Loading required package: Matrix
```

---

1. Linear regression and tests for coefficients significance.

1.a Upload `faraway` library and the dataset `savings` , an economic dataset on 50 different countries. These data are averages over 1960-1970 (to remove business cycle or other short-term fluctuations). The recorded covariates are:

- **sr** is aggregate personal saving divided by disposable income (risparmio personale diviso per il reddito disponibile).
- **pop15** is the percentage population under 15.
- **pop75** is the percentage population under 75.
- **dpi** is per-capita disposable income in U.S. dollars (reddito pro-capite in dollari, al netto delle tasse).
- **ddpi** is the rate [percentage] of change in per capita disposable income (potere d'acquisto - indice economico aggregato, espresso in %).

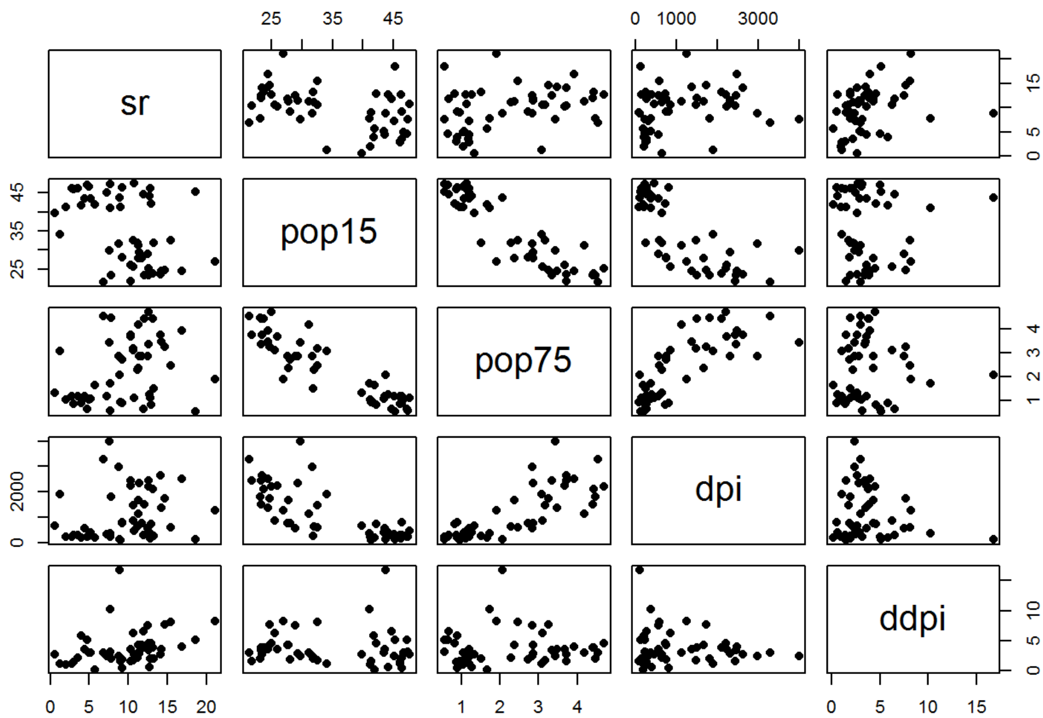
solution

```
data( savings )

# Dimensioni
dim( savings )
## [1] 50  5

# Overview delle prime righe
head( savings )
##              sr pop15 pop75     dpi ddpi
## Australia 11.43 29.35  2.87 2329.68 2.87
## Austria   12.07 23.32  4.41 1507.99 3.93
## Belgium   13.17 23.80  4.43 2108.47 3.82
## Bolivia    5.75 41.89  1.67  189.13 0.22
## Brazil    12.88 42.19  0.83  728.47 4.56
## Canada     8.79 31.72  2.85 2982.88 2.43
```

1.b Visualize the data and try to fit a complete linear model, in which `sr` is the outcome of interest. Explore the output of the model.

solution For visualizing the data, we can plot the pairs.

```
pairs( savings[ , c( 'sr', 'pop15', 'pop75', 'dpi', 'ddpi' ) ], pch = 16 )
```



Secondly, we can fit the complete linear model.

```
g = lm( sr ~ pop15 + pop75 + dpi + ddpi, data = savings )
#g = lm( sr ~ ., savings )
summary( g )
##
## Call:
## lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2422 -2.6857 -0.2488  2.4280  9.7509
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 28.5660865  7.3545161   3.884 0.000334 ***
## pop15       -0.4611931  0.1446422  -3.189 0.002603 **
## pop75       -1.6914977  1.0835989  -1.561 0.125530
## dpi         -0.0003369  0.0009311  -0.362 0.719173
## ddpi         0.4096949  0.1961971   2.088 0.042471 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.803 on 45 degrees of freedom
## Multiple R-squared:  0.3385, Adjusted R-squared:  0.2797
## F-statistic: 5.756 on 4 and 45 DF,  p-value: 0.0007904

gs = summary( g )

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

g$call
## lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings)
g$coefficients #beta_hat
##   (Intercept)         pop15         pop75           dpi          ddpi
## 28.5660865407 -0.4611931471 -1.6914976767 -0.0003369019  0.4096949279
g$fitted.values
##      Australia        Austria        Belgium        Bolivia         Brazil
##      10.566420      11.453614      10.951042       6.448319       9.327191
##         Canada          Chile          China       Colombia     Costa Rica
##       9.106892       8.842231       9.363964       6.431707       5.654922
##        Denmark        Ecuador        Finland         France        Germany
##      11.449761       5.995631      12.921086      10.164528      12.730699
##         Greece      Guatamala       Honduras        Iceland          India
##      13.786168       6.365284       6.989976       7.480582       8.491326
##        Ireland          Italy          Japan          Korea     Luxembourg
##       7.948869      12.353245      15.818514      10.086981      12.020807
##          Malta         Norway    Netherlands    New Zealand      Nicaragua
##      12.505090      11.121785      14.224454       8.384445       6.653603
##         Panama       Paraguay           Peru    Philippines       Portugal
##       7.734166       8.145759       6.160559       6.104992      13.258445
##   South Africa South Rhodesia          Spain         Sweden    Switzerland
##      10.656834      12.008566      12.441156      11.120283      11.643174
##         Turkey        Tunisia United Kingdom  United States      Venezuela
##       7.795682       5.627920      10.502413       8.671590       5.587482
##         Zambia        Jamaica        Uruguay          Libya       Malaysia
##       8.809086      10.738531      11.503827      11.719526       7.680869
X = model.matrix(g)
y_hat_man = X %*% g$coefficients #beta_hat

g$residuals
##      Australia        Austria        Belgium        Bolivia         Brazil
##      0.8635798      0.6163860      2.2189579     -0.6983191      3.5528094
##         Canada          Chile          China       Colombia     Costa Rica
##     -0.3168924     -8.2422307      2.5360361     -1.4517071      5.1250782
##        Denmark        Ecuador        Finland         France        Germany
```

```
##       5.4002388      -2.4056313      -1.6810857       2.4754718      -0.1806993
##          Greece       Guatamala        Honduras         Iceland           India
##      -3.1161685      -3.3552838       0.7100245      -6.2105820       0.5086740
##         Ireland           Italy           Japan           Korea      Luxembourg
##       3.3911306       1.9267549       5.2814855      -6.1069814      -1.6708066
##           Malta          Norway     Netherlands     New Zealand       Nicaragua
##       2.9749098      -0.8717854       0.4255455       2.2855548       0.6463966
##          Panama        Paraguay            Peru     Philippines        Portugal
##      -3.2941656      -6.1257589       6.5394410       6.6750084      -0.7684447
##    South Africa South Rhodesia           Spain          Sweden     Switzerland
##       0.4831656       1.2914342      -0.6711565      -4.2602834       2.4868259
##          Turkey         Tunisia United Kingdom   United States       Venezuela
##      -2.6656824      -2.8179200      -2.6924128      -1.1115901       3.6325177
##          Zambia         Jamaica         Uruguay           Libya        Malaysia
##       9.7509138      -3.0185314      -2.2638273      -2.8295257      -2.9708690

g$rank #p
## [1] 5

#help( vcov )
vcov( g )
##               (Intercept)         pop15         pop75           dpi
## (Intercept) 54.088907156 -1.046928e+00 -6.4480864740 -1.135929e-03
## pop15       -1.046927609  2.092137e-02  0.1199574165  2.422953e-05
## pop75       -6.448086474  1.199574e-01  1.1741866426 -3.703298e-04
## dpi         -0.001135929  2.422953e-05 -0.0003703298  8.669606e-07
## ddpi        -0.271654582  2.907814e-03 -0.0116339234  4.667202e-05
##                      ddpi
## (Intercept) -2.716546e-01
## pop15        2.907814e-03
## pop75       -1.163392e-02
## dpi          4.667202e-05
## ddpi         3.849331e-02
```

In order to measure the goodness of fit of the model, we have to look at R^2 and R^2_{adj}. Both of them are low in this case.

Through the F-statistic, we are investigating whether there is at least one covariate's parameter among β_1, β_2, β_3 and β_4 is different from 0. Since the p-value of F-statistic is so small (0.0007904), the null hypothesis is rejected and there is at least one covariate's parameter that is different from 0.

1.c Try to compute F-test, manually.

solution

```
# SStot = Sum ( yi-ybar )^2
SS_tot = sum( ( savings$sr-mean( savings$sr ) )^2 )

# SSres = Sum ( residuals^2 )
SS_res = sum( g$res^2 )

p = g$rank # p = 5
n = dim(savings)[1] # n = 50

f_test = ( ( SS_tot - SS_res )/(p-1) )/( SS_res/(n-p) )

1 - pf( f_test, p - 1, n - p )
## [1] 0.0007903779
```

1.d Test the significance of the parameter β_1 (the parameter related to pop_15), manually.

solution

We want to test:

$$H_0 : \beta_1 = 0 \qquad vs \qquad H_! : \beta_1 \neq 0$$

There are several ways to execute this test:

- **t-test**

We compute the test, whose output is shown in the R summary.

```
X = model.matrix( g )

sigma2 = (summary( g )$sigma)^2
#manually
sigma2 = sum( ( savings$sr - g$fitted.values )^2 ) / ( n -p )

se_beta_1 = summary( g )$coef[ 2, 2 ]
#manually
se_beta_1 = sqrt( sigma2 * diag( solve( t( X ) %*% X ) )[2] )

T.0 = abs( ( g$coefficients[ 2 ] - 0 )/ se_beta_1 )

2*( 1-pt( T.0, n-p ) )
##       pop15
## 0.002603019
```

- **F-test on nested model**

You fit a nested model (the complete model without the covariate in which you are interested) then you compute the residuals of the 2 models and execute the F-test.

REMARK it is NOT the F-test that you find in the summary!

$$F_0 = \frac{\frac{SS_{res}(\text{complete\_model}) - SS_{res}(\text{nested\_model})}{df(\text{complete\_model}) - df(\text{nested\_model})}}{\frac{SS_{res}(\text{complete\_model})}{df(\text{complete\_model})}} \sim F(df(\text{complete\_model}) - df(\text{nested\_model}), df(\text{complete\_model}))$$

```
g2 = lm( sr ~ pop75 + dpi + ddpi, data = savings )
summary( g2 )
##
## Call:
## lm(formula = sr ~ pop75 + dpi + ddpi, data = savings)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.0577 -3.2144  0.1687  2.4260 10.0763
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.4874944  1.4276619   3.844  0.00037 ***
## pop75       0.9528574  0.7637455   1.248  0.21849
## dpi         0.0001972  0.0010030   0.197  0.84499
## ddpi        0.4737951  0.2137272   2.217  0.03162 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.164 on 46 degrees of freedom
## Multiple R-squared:  0.189,  Adjusted R-squared:  0.1361
## F-statistic: 3.573 on 3 and 46 DF,  p-value: 0.02093
SS_res_2 = sum( g2$residuals^2 )

f_test_2 = ( ( SS_res_2 - SS_res ) / 1 )/( SS_res / (n-p) )

1 - pf( f_test_2, 1, n-p )
## [1] 0.002603019
```

- **t-test on the nested model**

You fit a nested model (the complete model without the covariate in which you are interested) then you compute the residuals of the 2 models and execute the t-test.

```
2 * ( 1-pt( sqrt( f_test_2 ), n-p ) )
## [1] 0.002603019
```

- **ANOVA between the two nested models**

```
anova( g2, g )
## Analysis of Variance Table
##
## Model 1: sr ~ pop75 + dpi + ddpi
## Model 2: sr ~ pop15 + pop75 + dpi + ddpi
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1     46 797.72
## 2     45 650.71  1    147.01 10.167 0.002603 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We notice that the outcome is the same in all the three methods. β_1 is signficant.

Homework

1.e Test the significance of all the regression parameters, separately.

1.f Test the regression parameter β_4 (the one related to 'ddpi') for this test:

$$H_0 : \beta_4 = 0.35 \qquad vs \qquad H_1 : \beta_4 > 0.35$$

---

2. Confidence Intervals and Regions

Confidence Intervals

2.a Compute the 95% confidence intervals for the regression parameter related to 'pop75'.

solution

The formula for the required confidence interval is:

$$IC_{(1-\alpha)}(\beta_2) = [\hat{\beta}_2 \pm t_{1-\alpha/2}(n-p) \cdot se(\hat{\beta}_2)],$$

where $\alpha = 5\%$ and $df = n - p = 45$.

```
alpha = 0.05
t_alpha2 = qt( 1-alpha/2, n-p )
beta_hat_pop75 = g$coefficients[3]
se_beta_hat_pop75 = summary( g )[[4]][3,2]

IC_pop75 = c( beta_hat_pop75 - t_alpha2 * se_beta_hat_pop75, beta_hat_pop75 + t_alpha2 * se_beta_hat_pop75 )
IC_pop75
##      pop75      pop75
## -3.8739780  0.4909826
```

We observe that $IC_{(1-\alpha)}(\beta_2)$ includes 0, so there is no evidence for rejectig $H_0 : \beta_2 = 0$, at the 5% level. Indeed, this parameter was not significant even in the previous section (p-value 12.5%).

2.b Compute the 95% confidence intervals for the regression parameter related to 'ddpi'.

solution

```
alpha = 0.05
t_alpha2 = qt( 1-alpha/2, n-p )
beta_hat_ddpi = g$coefficients[5]
se_beta_hat_ddpi = summary( g )[[4]][5,2]

IC_ddpi = c( beta_hat_ddpi - t_alpha2 * se_beta_hat_ddpi, beta_hat_ddpi + t_alpha2 * se_beta_hat_ddpi )
IC_ddpi
##       ddpi       ddpi
## 0.01453363 0.80485623
```

In this case, we observe that $IC_{(1-\alpha)}(\beta_4)$ does NOT include 0, so there is evidence for rejecting $H_0 : \beta_4 = 0$, at the 5% level. However, the lower bound of the $IC_{(1-\alpha)}(\beta_4)$ is really close to 0. We can see from the output above that the p-value is 4.2% - lower than 5% - confirming this point.

Notice that this confidence interval is pretty wide in the sense that the upper limit is about 80 times larger than the lower limit. This means that we are not really that confident about what the exact effect of growth on savings really is.

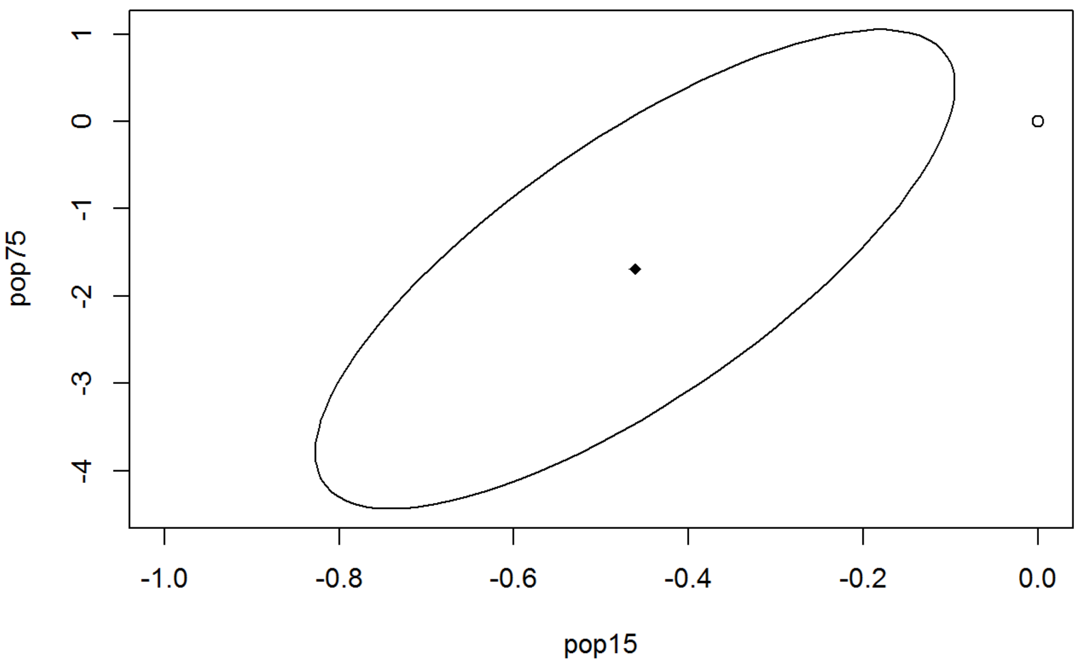
REMARK Confidence intervals often have a duality with two-sided hypothesis tests. A 95% confidence interval contains all the null hypotheses that would not be rejected at the 5% level.

Confidence Regions

2.c Build the joint 95% confidence region for parameters 'pop15' e 'pop75'. And add the value of (β_1, β_2) according to the null hypothesis.

solution

```
#help( ellipse )

plot( ellipse( g, c( 2, 3 ) ), type = "l", xlim = c( -1, 0 ) )

#add the origin and the point of the estimates:
#vettore che stiamo testando nell'hp nulla

points( 0, 0 )
points( g$coef[ 2 ] , g$coef[ 3 ] , pch = 18 )
```



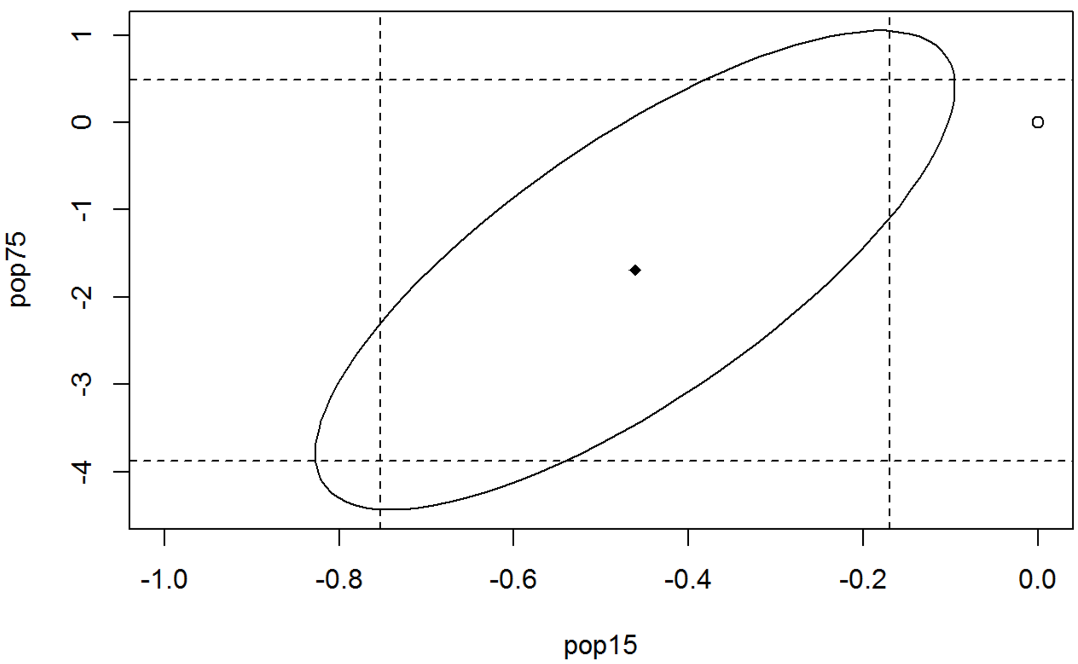
The filled dot is the center of the ellipse and represents the estimates of the 2 parameters $(\hat{\beta}_1, \hat{\beta}_2)$. Now, we are interested in this test:

$$H_0 : (\beta_1, \beta_2) = (0, 0) \qquad vs \qquad H_1 : (\beta_1, \beta_2) \neq (0, 0)$$

We observe that the empty dot (0,0) is not included in the Confidence Region (which is now an ellipse), so we reject H_0 at 5% level. In other words, we are saying that there is at least one parameter between β_1 and β_2 which is not equal to 0.

REMARK It is important to stress that this Confidence Region is different from the one obtained by the cartesian product of the two Confidence Intervals, $IC_{(1-\alpha)}(\beta_1)$ X $IC_{(1-\alpha)}(\beta_2)$. The cartesian product of the two Confidence Intervals is represented by the four dashed lines.

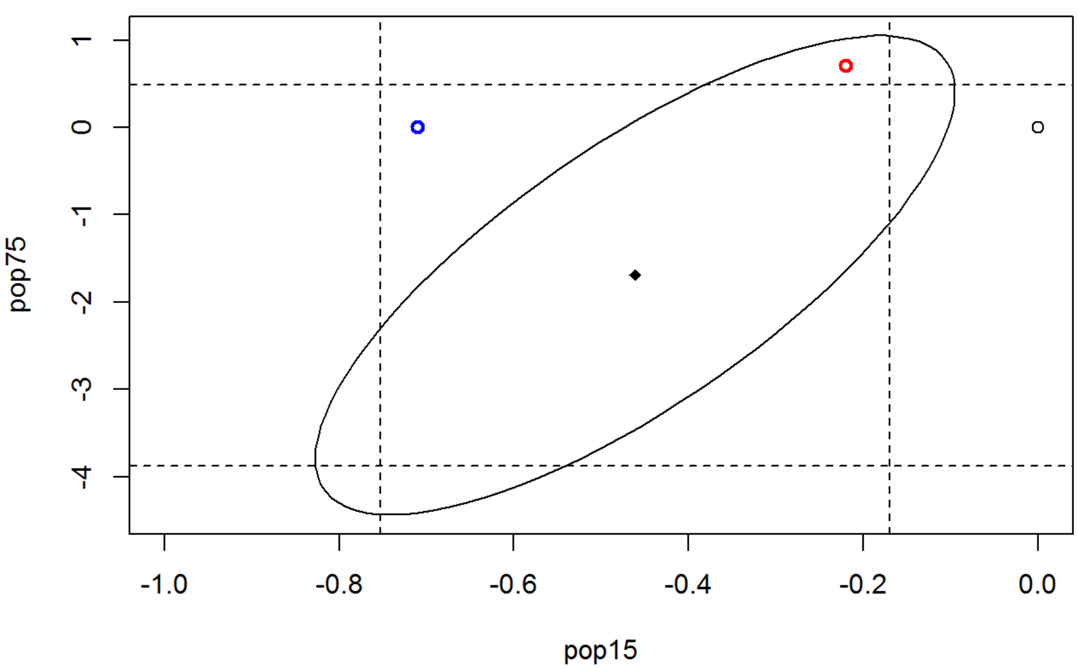
```
beta_hat_pop15 = g$coefficients[2]
se_beta_hat_pop15 = summary( g )[[4]][2,2]

IC_pop15 = c( beta_hat_pop15 - t_alpha2 * se_beta_hat_pop15, beta_hat_pop15 + t_alpha2 * se_beta_hat_pop15 )
IC_pop15
##      pop15      pop15
## -0.7525175 -0.1698688

plot( ellipse( g, c( 2, 3 ) ), type = "l", xlim = c( -1, 0 ) )

points( 0, 0 )
points( g$coef[ 2 ] , g$coef[ 3 ] , pch = 18 )

#new part
abline( v = c( IC_pop15[1], IC_pop15[2] ), lty = 2 )
abline( h = c( IC_pop75[1], IC_pop75[2] ), lty = 2 )
```



REMARK The origin $(0, 0)$ is included in the $IC_{(1-\alpha)}(\beta_2)$ and is NOT included in the $IC_{(1-\alpha)}(\beta_1)$, as expected from the previous point.

REMARK It can happen that you could reject according to one Confidence Region and accept according to the other Confidence Region. So which region should we choose?
We should alaways choose the joint Confidence Region (the elliptic one), because it is taking into account the correlation between the parameters.

```
plot( ellipse( g, c( 2, 3 ) ), type = "l", xlim = c( -1, 0 ) )

points( 0, 0 )
points( g$coef[ 2 ] , g$coef[ 3 ] , pch = 18 )

abline( v = c( IC_pop15[1], IC_pop15[2] ), lty = 2 )
abline( h = c( IC_pop75[1], IC_pop75[2] ), lty = 2 )

#new part
points( -0.22, 0.7, col = "red", lwd = 2 )
points( -0.71, 0, col = "blue", lwd = 2 )
```




```
cor( savings$pop15, savings$pop75 )
## [1] -0.9084787
```

---

3. Diagnostics: detecting influential points

The goal of diagnostics consists in detecting possible influential points in a sample. In general, an influential point is one whose removal from the dataset would cause a large change in the fit. influential points are outliers and leverages. The definitions of outliers and leverages can overlap. A possible definition of outlier is ‘a point that does not fit the chosen model’. On the other hand, a leverage is ‘a point that significantly affects the estimates of the model’. It is immediate to see that often an outlier is also a leverage point.

Some measures of influential:

1. Change in the coefficients: $\hat{\beta} - \hat{\beta}_i$
2. Change in the fit: $x^T(\hat{\beta} - \hat{\beta}_i) = \hat{y} - \hat{y}_i$ These are hard to judge in the sense that the scale varies between datasets.

There are several approaches for identifying influential points in a sample, such as:

- **Leverages**
- **Standardized Residuals**

- **Studentized Residuals**
- **Jacknife Residuals**
- **Cook's Distance**

Leverages

3.a Investigate possible leverages among data. Leverages are defined as the diagonal elements of H matrix:

$$H = X(X^T X)^{-1} X^T$$

solution

```
X = model.matrix( g )
X
##                (Intercept) pop15 pop75     dpi  ddpi
## Australia                1 29.35  2.87 2329.68  2.87
## Austria                  1 23.32  4.41 1507.99  3.93
## Belgium                  1 23.80  4.43 2108.47  3.82
## Bolivia                  1 41.89  1.67  189.13  0.22
## Brazil                   1 42.19  0.83  728.47  4.56
## Canada                   1 31.72  2.85 2982.88  2.43
## Chile                    1 39.74  1.34  662.86  2.67
## China                    1 44.75  0.67  289.52  6.51
## Colombia                 1 46.64  1.06  276.65  3.08
## Costa Rica               1 47.64  1.14  471.24  2.80
## Denmark                  1 24.42  3.93 2496.53  3.99
## Ecuador                  1 46.31  1.19  287.77  2.19
## Finland                  1 27.84  2.37 1681.25  4.32
## France                   1 25.06  4.70 2213.82  4.52
## Germany                  1 23.31  3.35 2457.12  3.44
## Greece                   1 25.62  3.10  870.85  6.28
## Guatamala                1 46.05  0.87  289.71  1.48
## Honduras                 1 47.32  0.58  232.44  3.19
## Iceland                  1 34.03  3.08 1900.10  1.12
## India                    1 41.31  0.96   88.94  1.54
## Ireland                  1 31.16  4.19 1139.95  2.99
## Italy                    1 24.52  3.48 1390.00  3.54
## Japan                    1 27.01  1.91 1257.28  8.21
## Korea                    1 41.74  0.91  207.68  5.81
## Luxembourg               1 21.80  3.73 2449.39  1.57
## Malta                    1 32.54  2.47  601.05  8.12
## Norway                   1 25.95  3.67 2231.03  3.62
## Netherlands              1 24.71  3.25 1740.70  7.66
## New Zealand              1 32.61  3.17 1487.52  1.76
## Nicaragua                1 45.04  1.21  325.54  2.48
## Panama                   1 43.56  1.20  568.56  3.61
## Paraguay                 1 41.18  1.05  220.56  1.03
## Peru                     1 44.19  1.28  400.06  0.67
## Philippines              1 46.26  1.12  152.01  2.00
## Portugal                 1 28.96  2.85  579.51  7.48
## South Africa             1 31.94  2.28  651.11  2.19
## South Rhodesia           1 31.92  1.52  250.96  2.00
## Spain                    1 27.74  2.87  768.79  4.35
## Sweden                   1 21.44  4.54 3299.49  3.01
## Switzerland              1 23.49  3.73 2630.96  2.70
## Turkey                   1 43.42  1.08  389.66  2.96
## Tunisia                  1 46.12  1.21  249.87  1.13
## United Kingdom           1 23.27  4.46 1813.93  2.01
## United States            1 29.81  3.43 4001.89  2.45
## Venezuela                1 46.40  0.90  813.39  0.53
## Zambia                   1 45.25  0.56  138.33  5.14
## Jamaica                  1 41.12  1.73  380.47 10.23
## Uruguay                  1 28.13  2.72  766.54  1.88
## Libya                    1 43.69  2.07  123.58 16.71
## Malaysia                 1 47.20  0.66  242.69  5.08
## attr(,"assign")
## [1] 0 1 2 3 4
```

```
lev = hat( X )
lev
##  [1] 0.06771343 0.12038393 0.08748248 0.08947114 0.06955944 0.15840239
##  [7] 0.03729796 0.07795899 0.05730171 0.07546780 0.06271782 0.06372651
## [13] 0.09204246 0.13620478 0.08735739 0.09662073 0.06049212 0.06008079
## [19] 0.07049590 0.07145213 0.21223634 0.06651170 0.22330989 0.06079915
## [25] 0.08634787 0.07940290 0.04793213 0.09061400 0.05421789 0.05035056
## [31] 0.03897459 0.06937188 0.06504891 0.06425415 0.09714946 0.06510405
## [37] 0.16080923 0.07732854 0.12398898 0.07359423 0.03964224 0.07456729
```

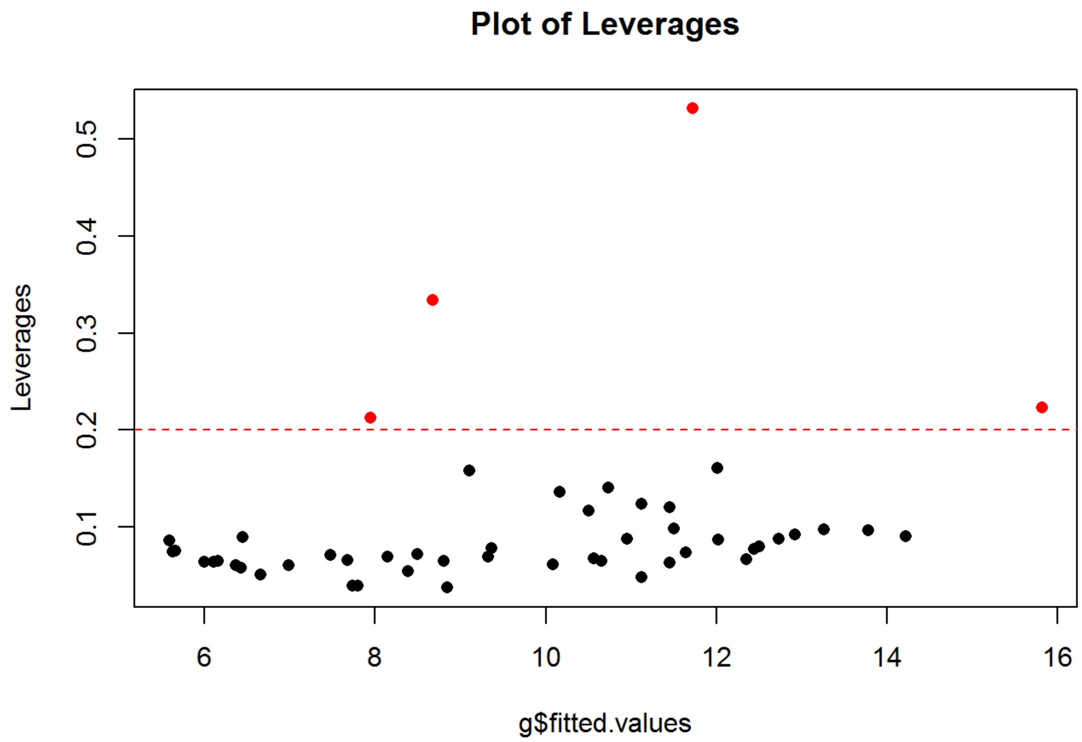
```
## [43] 0.11651375 0.33368800 0.08628365 0.06433163 0.14076016 0.09794717
## [49] 0.53145676 0.06523300
# oppure
lev = hatvalues( g )
lev
##      Australia      Austria      Belgium      Bolivia       Brazil
##     0.06771343   0.12038393   0.08748248   0.08947114   0.06955944
##         Canada        Chile        China     Colombia   Costa Rica
##     0.15840239   0.03729796   0.07795899   0.05730171   0.07546780
##        Denmark      Ecuador      Finland       France      Germany
##     0.06271782   0.06372651   0.09204246   0.13620478   0.08735739
##         Greece    Guatamala     Honduras      Iceland        India
##     0.09662073   0.06049212   0.06008079   0.07049590   0.07145213
##        Ireland        Italy        Japan        Korea   Luxembourg
##     0.21223634   0.06651170   0.22330989   0.06079915   0.08634787
##          Malta       Norway  Netherlands  New Zealand    Nicaragua
##     0.07940290   0.04793213   0.09061400   0.05421789   0.05035056
##         Panama     Paraguay         Peru  Philippines     Portugal
##     0.03897459   0.06937188   0.06504891   0.06425415   0.09714946
##   South Africa South Rhodesia        Spain       Sweden  Switzerland
##     0.06510405   0.16080923   0.07732854   0.12398898   0.07359423
##         Turkey      Tunisia United Kingdom United States    Venezuela
##     0.03964224   0.07456729   0.11651375   0.33368800   0.08628365
##         Zambia      Jamaica      Uruguay        Libya     Malaysia
##     0.06433163   0.14076016   0.09794717   0.53145676   0.06523300

#manually
H = X %*% solve( t( X ) %*% X ) %*% t( X )
lev = diag( H )

sum(lev)
## [1] 5
```

REMARK The trace of the H matrix (sum of the diagonal elements of a matrix) is equal to the rank of X matrix, which is $p+1$, assuming that covariates are all linearly independent and $p < n$. This is the size of the vectorial subspace generated by the linear combinations of the columns of X. The geometric interpretation of the linear regression (OLS) states that H acts on $\mathbf{y}$ (vector of outcomes) by projecting it on the former subspace. The final output is $\hat{\mathbf{y}}$.

Rule of thumb: a point is a leverage if:

$$\hat{h}_{ii} > 2 \cdot \frac{p}{n}$$

```
plot( g$fitted.values, lev, ylab = "Leverages", main = "Plot of Leverages", pch = 16, col = 'black' )

abline( h = 2 * p/n, lty = 2, col = 'red' )

watchout_points_lev = lev[ which( lev > 2 * p/n  ) ]
watchout_ids_lev = seq_along( lev )[ which( lev > 2 * p/n ) ]

points( g$fitted.values[ watchout_ids_lev ], watchout_points_lev, col = 'red', pch = 16 )
```




```
sum( lev )      # verifica: sum_i hat( x )_i = r + 1
## [1] 5

lev [ lev >  2 * 5 / 50 ]
##       Ireland         Japan United States         Libya
##     0.2122363     0.2233099     0.3336880     0.5314568
sum( lev [ lev >  2 * 5 / 50 ] )
## [1] 1.300691
```

3.b Fit the model without leverages.

solution

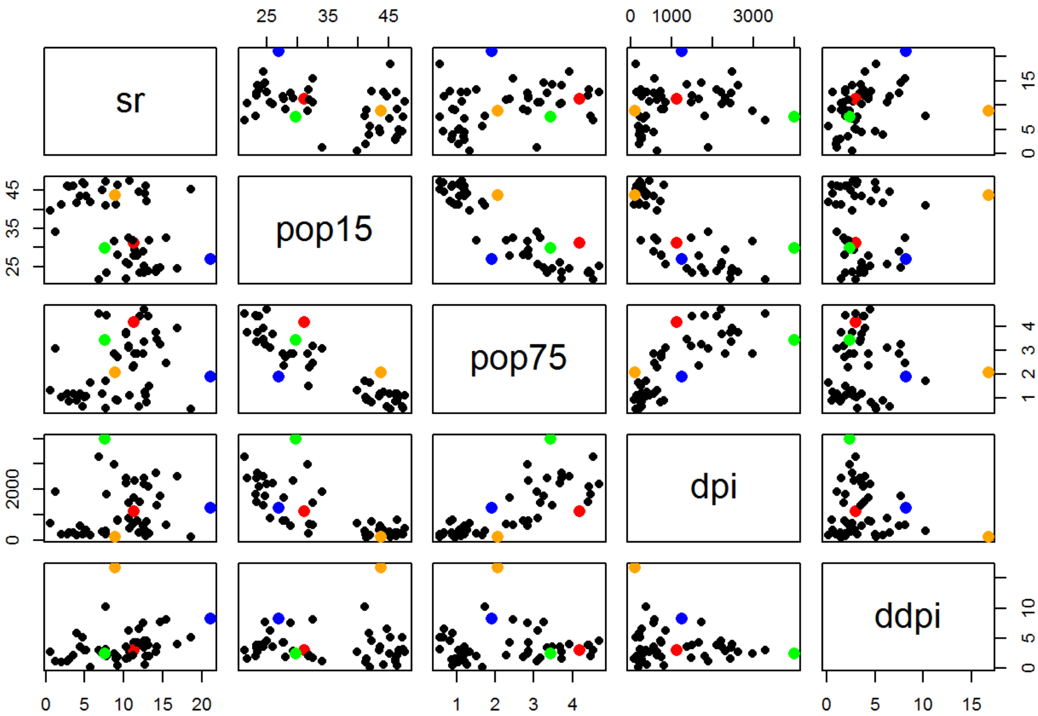
```
gl = lm( sr ~ pop15 + pop75 + dpi + ddpi, savings, subset = ( lev < 0.2 ) )
summary( gl )
##
## Call:
## lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings,
##     subset = (lev < 0.2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9632 -2.6323  0.1466  2.2529  9.6687
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.221e+01  9.319e+00   2.384   0.0218 *
## pop15       -3.403e-01  1.798e-01  -1.893   0.0655 .
## pop75       -1.124e+00  1.398e+00  -0.804   0.4258
## dpi         -4.499e-05  1.160e-03  -0.039   0.9692
## ddpi         5.273e-01  2.775e-01   1.900   0.0644 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.805 on 41 degrees of freedom
## Multiple R-squared:  0.2959, Adjusted R-squared:  0.2272
## F-statistic: 4.308 on 4 and 41 DF,  p-value: 0.005315
#summary( g )
```

Moreover, investigate the relative variation of $\hat{\beta}$ due to these influential points.

```
abs( ( g$coefficients - gl$coefficients ) / g$coefficients )
## (Intercept)       pop15       pop75         dpi        ddpi
##   0.2223914   0.2622274   0.3353998   0.8664714   0.2871002
```

The leverages affect the estimate heavily (there is a variation of 22% at least).

```
colors = rep( 'black', nrow( savings ) )
colors[ watchout_ids_lev ] = c('red', 'blue', 'green', 'orange')

pairs( savings[ , c( 'sr', 'pop15', 'pop75', 'dpi', 'ddpi' ) ], pch = 16, col = colors, cex = 1 + 0.5 * as.num
eric( colors != 'black' )    )
```

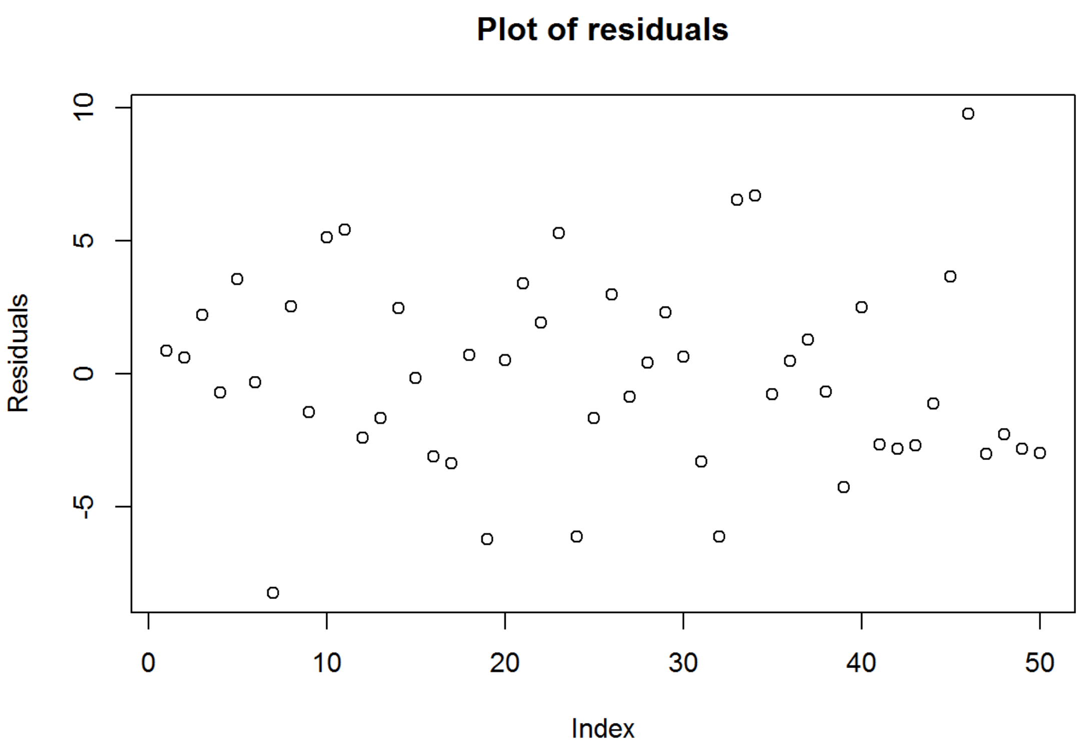


Standardized Residuals

3.c Plot the residuals of the complete model.

solution

```
# Residui non standardizzati (nÃ¨ studentizzati)

plot( g$res, ylab = "Residuals", main = "Plot of residuals" )

sort( g$res )
##          Chile        Iceland       Paraguay          Korea         Sweden
##     -8.2422307     -6.2105820     -6.1257589     -6.1069814     -4.2602834
##      Guatamala         Panama         Greece        Jamaica       Malaysia
##     -3.3552838     -3.2941656     -3.1161685     -3.0185314     -2.9708690
##          Libya        Tunisia United Kingdom         Turkey        Ecuador
##     -2.8295257     -2.8179200     -2.6924128     -2.6656824     -2.4056313
##        Uruguay        Finland     Luxembourg       Colombia  United States
##     -2.2638273     -1.6810857     -1.6708066     -1.4517071     -1.1115901
##         Norway       Portugal        Bolivia          Spain         Canada
##     -0.8717854     -0.7684447     -0.6983191     -0.6711565     -0.3168924
##        Germany    Netherlands   South Africa          India        Austria
##     -0.1806993      0.4255455      0.4831656      0.5086740      0.6163860
##      Nicaragua       Honduras      Australia South Rhodesia          Italy
##      0.6463966      0.7100245      0.8635798      1.2914342      1.9267549
##        Belgium    New Zealand         France    Switzerland          China
##      2.2189579      2.2855548      2.4754718      2.4868259      2.5360361
##          Malta        Ireland         Brazil      Venezuela     Costa Rica
##      2.9749098      3.3911306      3.5528094      3.6325177      5.1250782
##          Japan        Denmark           Peru    Philippines         Zambia
##      5.2814855      5.4002388      6.5394410      6.6750084      9.7509138
sort( g$res ) [ c( 1, 50 ) ]
##     Chile    Zambia
## -8.242231  9.750914

countries = row.names( savings )
identify( 1:50, g$res, countries ) # cliccare 2 volte sui punti a cui si vuole apporre il label
```




```
## integer(0)
```

`identify` is a useful function for detecting influent points. In input, you should call the x and y axes of the plot and the labels of data.

Usually, the residuals are represented wrt y-values or the single predictors.

This is useful also for testing the model hypotheses: omoschedasticity and normality of residuals (flash forward).

The representation with the index of the observation as x-axis is not that useful (except if we are interested in investigating the distribution of the residuals wrt the procedure used for data collection).

3.d Plot the **Standardized Residuals** of the complete model.

Rule of thumb Given that standardized residuals are defined as:

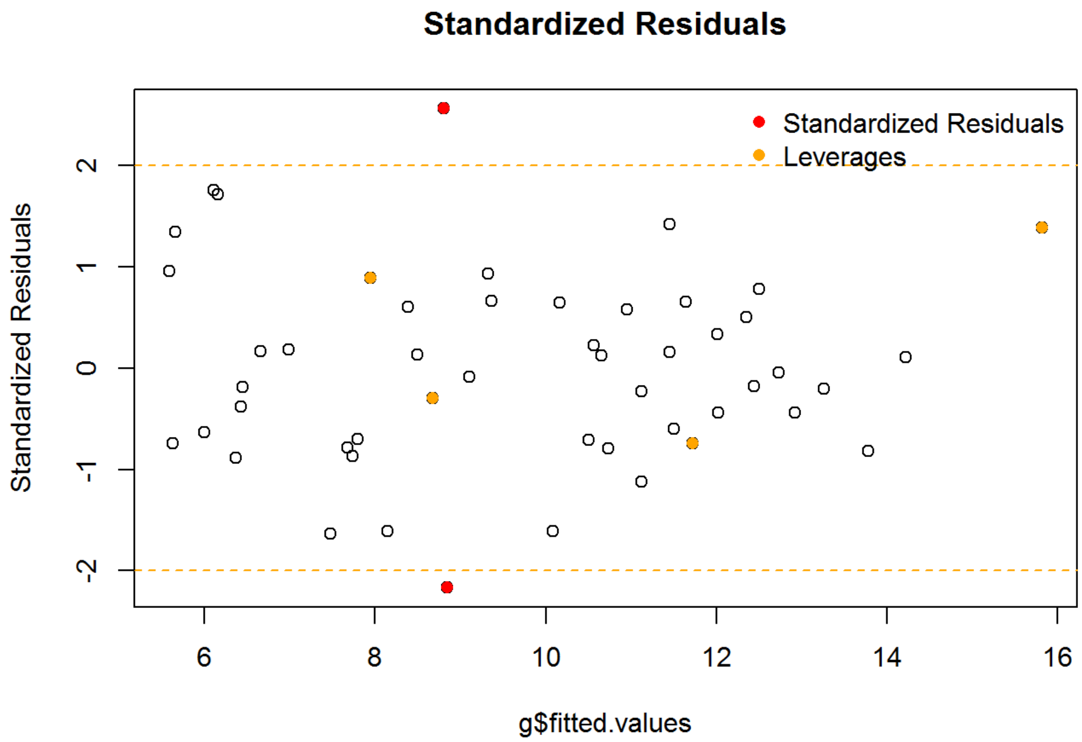
$$r_i^{std} = \frac{y_i - \hat{y}_i}{\hat{S}}$$

influential points satisfy the following inequality:

$$|r_i^{std}| > 2$$

solution

```
gs = summary(g)
res_std = g$res/gs$sigma
watchout_ids_rstd = which( abs( res_std ) > 2 )
watchout_rstd = res_std[ watchout_ids_rstd ]
watchout_rstd
##     Chile    Zambia
## -2.167486  2.564229

# Residui standardizzati (non studentizzati)

plot( g$fitted.values, res_std, ylab = "Standardized Residuals", main = "Standardized Residuals" )
abline( h = c(-2,2), lty = 2, col = 'orange' )
points( g$fitted.values[watchout_ids_rstd], res_std[watchout_ids_rstd], col = 'red', pch = 16 )
points( g$fitted.values[watchout_ids_lev], res_std[watchout_ids_lev], col = 'orange', pch = 16 )
legend('topright', col = c('red','orange'), c('Standardized Residuals', 'Leverages'), pch = rep( 16, 2 ), bty
 = 'n' )

sort( g$res/gs$sigma )
##          Chile        Iceland       Paraguay          Korea         Sweden
##    -2.16748590    -1.63321671    -1.61091051    -1.60597254    -1.12034042
##      Guatamala         Panama         Greece        Jamaica       Malaysia
##    -0.88234977    -0.86627732    -0.81946884    -0.79379291    -0.78125899
##          Libya        Tunisia United Kingdom         Turkey        Ecuador
##    -0.74408946    -0.74103748    -0.70803244    -0.70100307    -0.63261660
##        Uruguay        Finland     Luxembourg       Colombia  United States
##    -0.59532595    -0.44208050    -0.43937737    -0.38176008    -0.29231843
##         Norway       Portugal        Bolivia          Spain         Canada
##    -0.22925620    -0.20208037    -0.18363922    -0.17649617    -0.08333420
##        Germany    Netherlands   South Africa          India        Austria
##    -0.04751907     0.11190707     0.12705960     0.13376764     0.16209300
##      Nicaragua       Honduras      Australia South Rhodesia          Italy
##     0.16998499     0.18671742     0.22709835     0.33961261     0.50668493
##        Belgium    New Zealand         France    Switzerland          China
##     0.58352650     0.60103970     0.65098279     0.65396859     0.66690956
##          Malta        Ireland         Brazil      Venezuela     Costa Rica
##     0.78232159     0.89177653     0.93429372     0.95525486     1.34775829
##          Japan        Denmark           Peru    Philippines         Zambia
##     1.38888924     1.42011817     1.71969783     1.75534843     2.56422914
sort( g$res/gs$sigma ) [ c( 1, 50 ) ]
##     Chile    Zambia
## -2.167486  2.564229

countries = row.names( savings )
identify( 1:50, g$res/gs$sigma, countries ) # cliccare 2 volte sui punti a cui si vuole apporre il label
```




```
## integer(0)
```

Studentized Residuals

3.e Compute the Studentized Residuals, highlighting the influential points.

solution

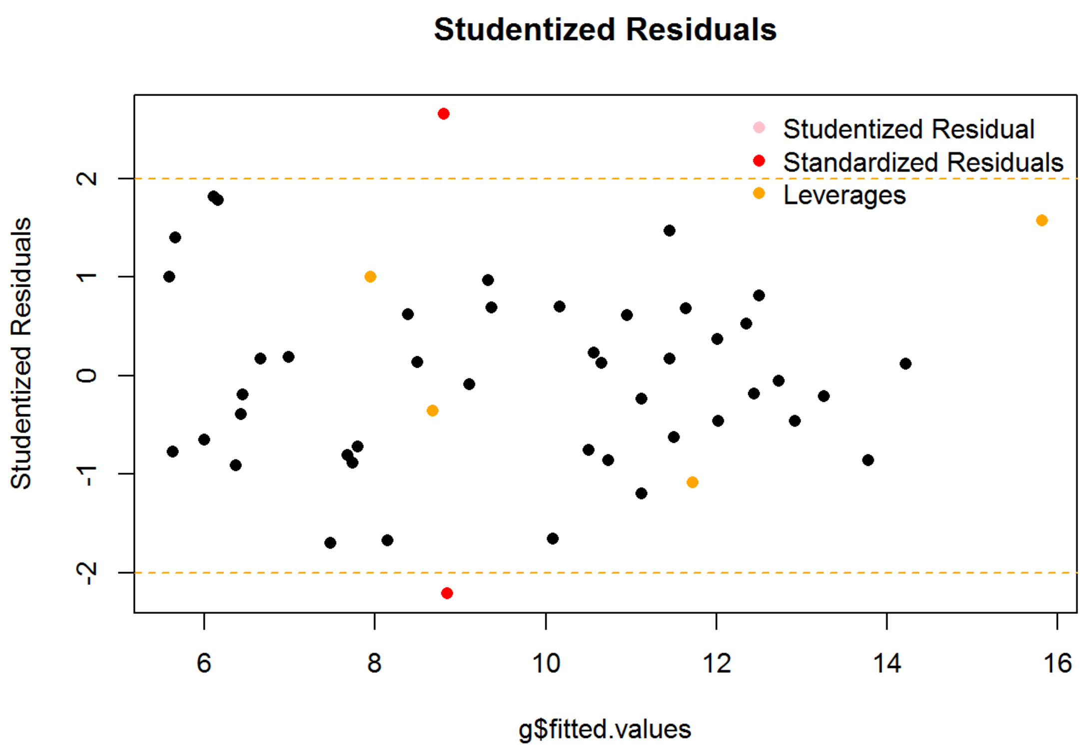
Studentized residuals, r_i, are computed as:

$$r_i = \frac{\hat{\varepsilon}_i}{\hat{S} \cdot \sqrt{(1 - h_{ii})}} \sim t(n-p)$$

Since r_i is distributed according to a Student-t with $(n-p)$ df, we can calculate a p-value to test whether point $i-th$ is an outlier. However, we would probably want to test all cases so we must adjust the level of the test accordingly (for example, using Bonferroni correction).

`rstandard` gives jackknife residuals automaically.

```
gs = summary( g )

gs$sigma
## [1] 3.802669

#manually
stud = g$residuals / ( gs$sigma * sqrt( 1 - lev ) )

#automatically
stud = rstandard( g )

watchout_ids_stud = which( abs( stud ) > 2 )
watchout_stud = stud[ watchout_ids_stud ]
watchout_stud
##     Chile    Zambia
## -2.209074  2.650915


plot( g$fitted.values, stud, ylab = "Studentized Residuals", main = "Studentized Residuals", pch = 16 )
points( g$fitted.values[watchout_ids_stud], stud[watchout_ids_stud], col = 'pink', pch = 16 )
points( g$fitted.values[watchout_ids_rstd], stud[watchout_ids_rstd], col = 'red', pch = 16 )
points( g$fitted.values[watchout_ids_lev], stud[watchout_ids_lev], col = 'orange', pch = 16 )
abline( h = c(-2,2), lty = 2, col = 'orange' )
legend('topright', col = c('pink','red','orange'), c('Studentized Residual', 'Standardized Residuals', 'Levera
ges'), pch = rep( 16, 3 ), bty = 'n' )
```



We do not see pink dots, because Studentized residuals and Standardized residuals identify the same influential points.

Jackknife residuals

The idea behind Jackknife residuals consists in seeing the impact of a single realization on the model. In order to do so, we exclude one observation at a time and we estimate model parameters without that specific observation.

For example, if we exclude the $i-th$ observation, then we compute:

$$\hat{y}_{(-i)} = X^T_{(-i)} \cdot \hat{\beta}_{(-i)}$$

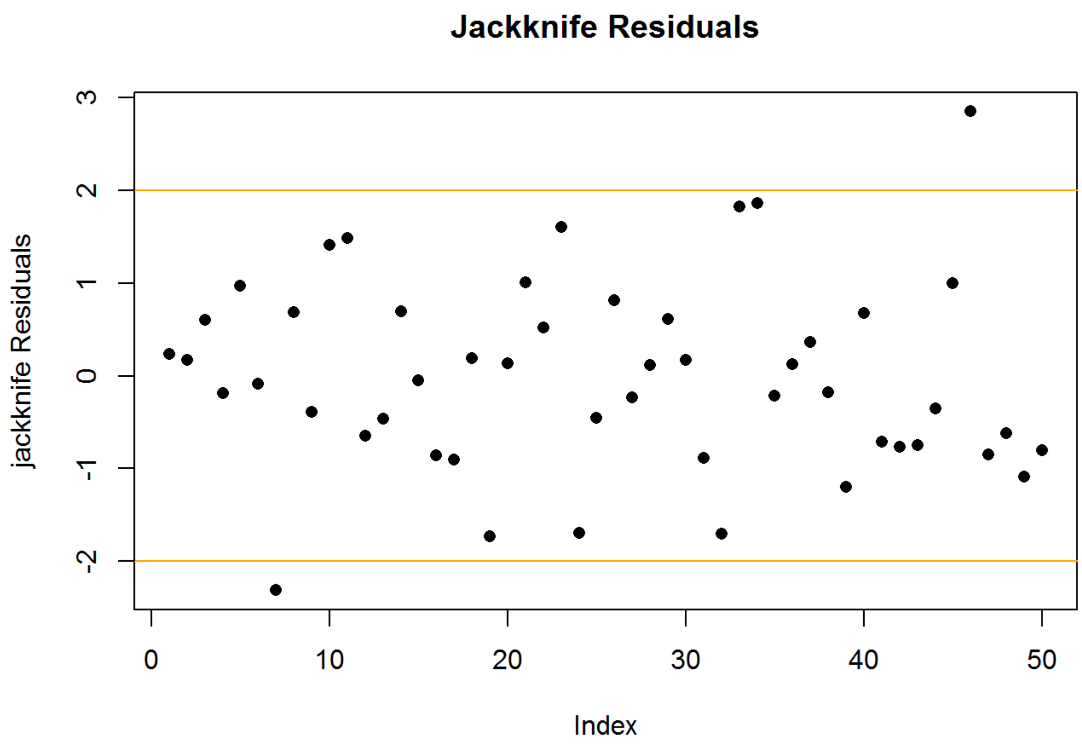
The former procedure is called *Jackknife*, or *Leave-one-out*, and has a general validity in Statistics as a method for defining the sensitibility of an estimate with respect to specific data that determined it. The formula for computing the Jackknife residuals is:

$$r_{(-i)} = r_i \cdot \sqrt{\frac{(n-p-1)}{(n-p-r_i^2)}}$$

in which r_i is the studentized residuals.

`rstudent` gives jackknife residuals automatically.

```
jack = rstudent( g )

jack_man = stud * sqrt( ( n - p - 1 )/( n - p - stud^2 ) )

plot( jack, ylab = "jackknife Residuals", main = "Jackknife Residuals", pch = 16 )
abline( h = c( -2, 2 ), col = 'orange' )
```

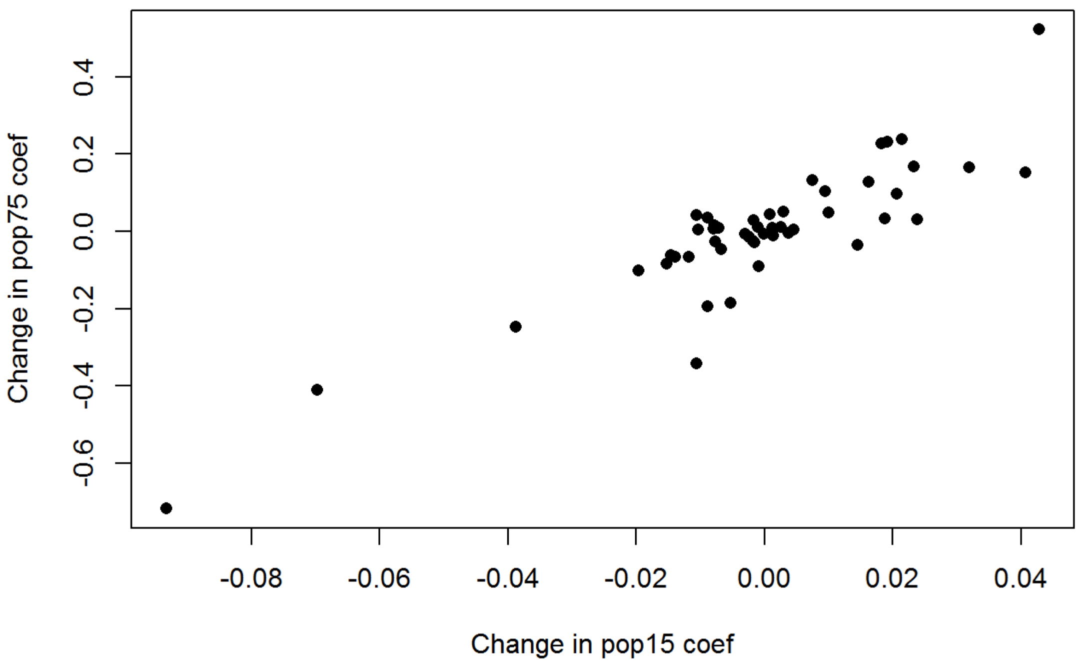



```
jack [ abs( jack ) > 2 ]
##     Chile    Zambia
## -2.313429  2.853558
```

The highest jackknife residual is 2.853558, related to Zambia. Two can be considered as outlier: Chile and Zambia.

Graphical explanation of leave-one-out idea. We fit n-model through `lm.influence` function. We use all the observations except the i-th observation (*leave-one-out*) for fitting the i-th model. The output of the function is the diffence between the estimates in the model fitted with $n-1$ obesrvations and the model fitted with n observations.

```
ginf = lm.influence( g )
names( ginf )
## [1] "hat"          "coefficients" "sigma"        "wt.res"

#we leave Australia out
g_aus = lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings, subset = ( countries != "Australia" ) )
g$coef - g_aus$coef
##   (Intercept)         pop15         pop75           dpi          ddpi
##  9.157623e-02 -1.525554e-03 -2.905432e-02  4.266614e-05 -3.157482e-05
ginf$coefficients[ 1, ]
##   (Intercept)         pop15         pop75           dpi          ddpi
##  9.157623e-02 -1.525554e-03 -2.905432e-02  4.266614e-05 -3.157482e-05

countries = rownames( savings )

plot( ginf$coef [ , 2 ] , ginf$coef [ , 3 ] , xlab = "Change in pop15 coef",
      ylab = "Change in pop75 coef", pch = 16 )
```




```
#identify( ginf$coef [ , 2 ] , ginf$coef [ , 3 ] , countries )
```

Japan is an influential point.

Cook’s distance

Cook’s distance is a commonly used influential measure that combines the two characteristics of an influential point. It can be expressed as:

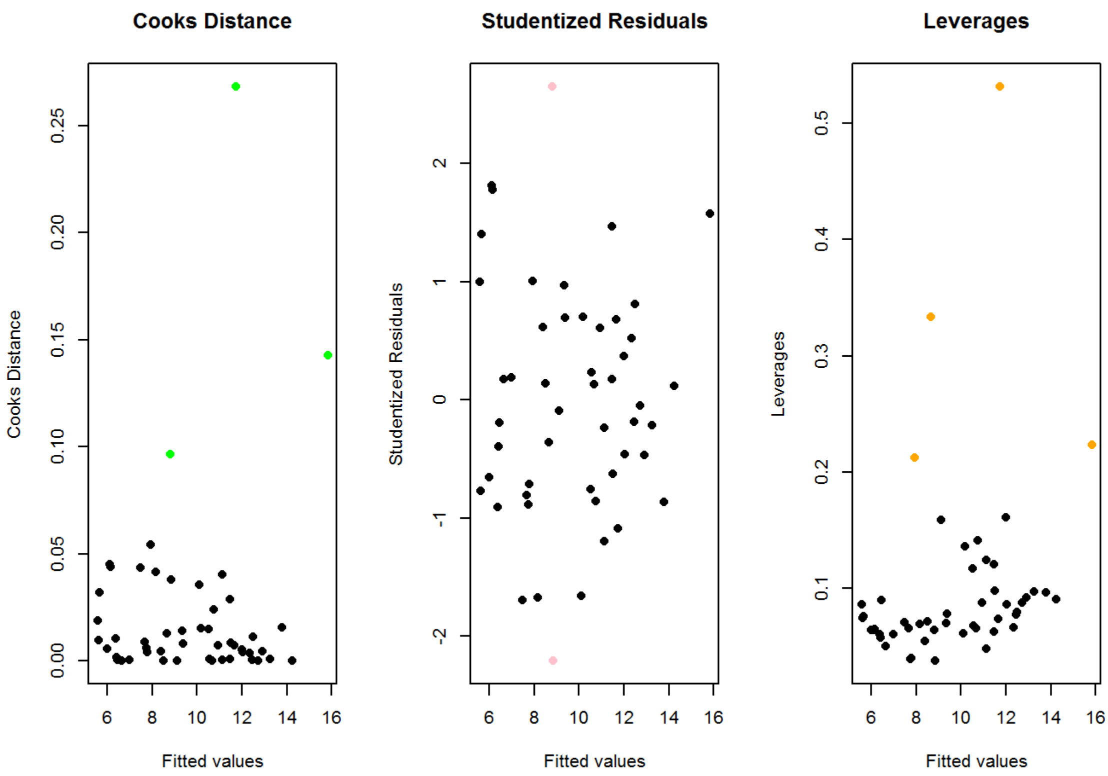
$$C_i = r_i^2/p \cdot \left[\frac{h_{ii}}{1-h_{ii}}\right]$$

in which r_i are the studentized residuals.

Rule of thumb A point is defined influential if:

$$C_i > \frac{4}{n-p}$$

```
Cdist = cooks.distance( g )

watchout_ids_Cdist = which( Cdist > 4/(n-p) )
watchout_Cdist = Cdist[ watchout_ids_Cdist ]
watchout_Cdist
##      Japan     Zambia      Libya
## 0.14281625 0.09663275 0.26807042


par( mfrow = c( 1, 3 ) )
plot( g$fitted.values, Cdist, pch = 16, xlab = 'Fitted values', ylab = 'Cooks Distance', main = 'Cooks Distanc
e' )
points( g$fitted.values[ watchout_ids_Cdist ], Cdist[ watchout_ids_Cdist ], col = 'green', pch = 16 )
plot( g$fitted.values, stud, pch = 16, xlab = 'Fitted values', ylab = 'Studentized Residuals', main = 'Student
ized Residuals' )
points( g$fitted.values[ watchout_ids_stud ], stud[ watchout_ids_stud ], col = 'pink', pch = 16 )
plot( g$fitted.values, lev, pch = 16, xlab = 'Fitted values', ylab = 'Leverages', main = 'Leverages' )
points( g$fitted.values[ watchout_ids_lev ], lev[ watchout_ids_lev ], col = 'orange', pch = 16 )
```



3.f Fit the model without influential points wrt Cook’s distance and compare the outcome to the former model (on the complete dataset).

solution

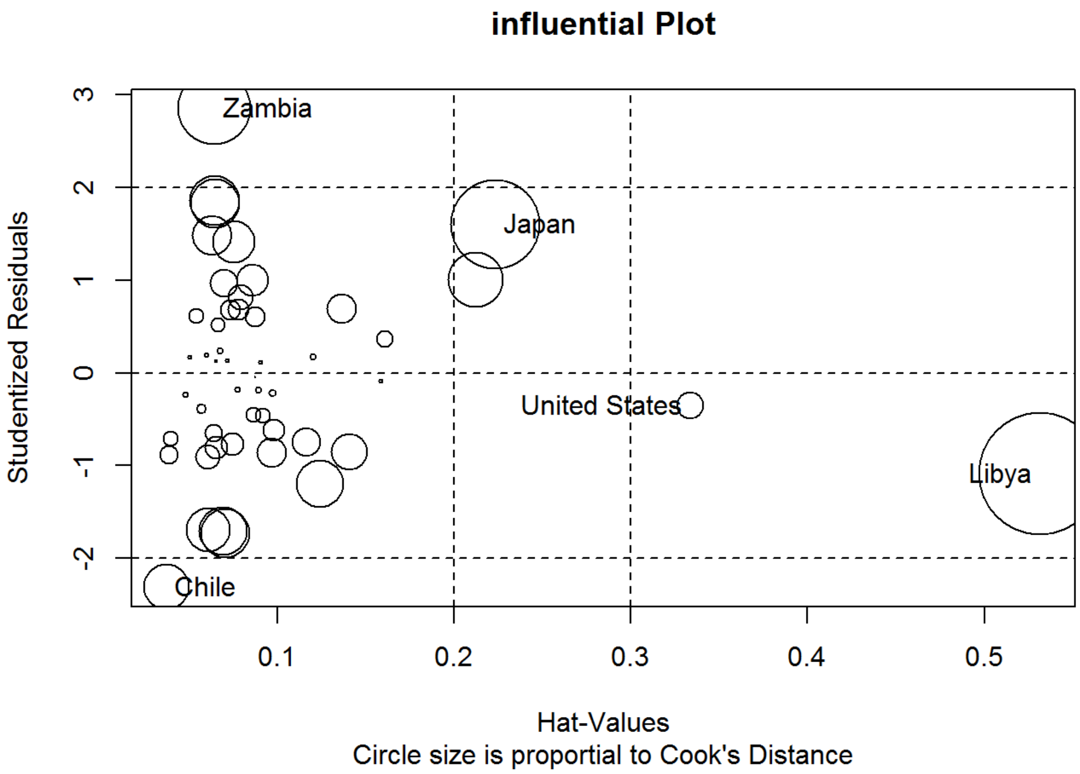
```
#id_to_keep = (1:n)[ - watchout_ids_Cdist ]
id_to_keep = !( 1:n %in% watchout_ids_Cdist )

gl = lm( sr ~ pop15 + pop75 + dpi + ddpi, savings[ id_to_keep, ]  )

abs( ( gl$coef - g$coef )/g$coef )
## (Intercept)       pop15       pop75         dpi        ddpi 
## 0.305743704 0.339320881 0.820854095 0.642906116 0.009976742
summary( g )
## 
## Call:
## lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -8.2422 -2.6857 -0.2488  2.4280  9.7509 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 28.5660865  7.3545161   3.884 0.000334 ***
## pop15       -0.4611931  0.1446422  -3.189 0.002603 ** 
## pop75       -1.6914977  1.0835989  -1.561 0.125530    
## dpi         -0.0003369  0.0009311  -0.362 0.719173    
## ddpi         0.4096949  0.1961971   2.088 0.042471 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 3.803 on 45 degrees of freedom
## Multiple R-squared:  0.3385, Adjusted R-squared:  0.2797 
## F-statistic: 5.756 on 4 and 45 DF,  p-value: 0.0007904
```

The coefficient for dpi changed by about 64%.

Influential Plot

The influential plot represents the studentized residuals vs leverages, and highlights them with a circle which is proportional to Cook's distance.

```
influencePlot( g, id.method = "identify", main = "influential Plot",
               sub = "Circle size is proportial to Cook's Distance" )
## Warning in plot.window(...): "id.method" is not a graphical parameter
## Warning in plot.xy(xy, type, ...): "id.method" is not a graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" is
## not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "id.method" is
## not a graphical parameter
## Warning in box(...): "id.method" is not a graphical parameter
## Warning in title(...): "id.method" is not a graphical parameter
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "id.method" is not a
## graphical parameter
```




```
##                  StudRes        Hat      CookD
## Chile         -2.3134295 0.03729796 0.03781324
## Japan          1.6032158 0.22330989 0.14281625
## United States -0.3546151 0.33368800 0.01284481
## Zambia         2.8535583 0.06433163 0.09663275
## Libya         -1.0893033 0.53145676 0.26807042
```

Influential measures

influential.measures produces a class “infl” object tabular display showing several diagnostics measures (such as h_{ii} and Cook’s distance). Those cases which are influential with respect to any of these measures are marked with an asterisk.

```
influence.measures( g )
## Influence measures of
##   lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings) :
##
##                   dfb.1_ dfb.pp15 dfb.pp75  dfb.dpi  dfb.ddpi   dffit cov.r
## Australia       0.01232 -0.01044 -0.02653  0.04534 -0.000159  0.0627 1.193
## Austria        -0.01005  0.00594  0.04084 -0.03672 -0.008182  0.0632 1.268
## Belgium        -0.06416  0.05150  0.12070 -0.03472 -0.007265  0.1878 1.176
## Bolivia         0.00578 -0.01270 -0.02253  0.03185  0.040642 -0.0597 1.224
## Brazil          0.08973 -0.06163 -0.17907  0.11997  0.068457  0.2646 1.082
## Canada          0.00541 -0.00675  0.01021 -0.03531 -0.002649 -0.0390 1.328
## Chile          -0.19941  0.13265  0.21979 -0.01998  0.120007 -0.4554 0.655
## China           0.02112 -0.00573 -0.08311  0.05180  0.110627  0.2008 1.150
## Colombia        0.03910 -0.05226 -0.02464  0.00168  0.009084 -0.0960 1.167
## Costa Rica     -0.23367  0.28428  0.14243  0.05638 -0.032824  0.4049 0.968
## Denmark        -0.04051  0.02093  0.04653  0.15220  0.048854  0.3845 0.934
## Ecuador         0.07176 -0.09524 -0.06067  0.01950  0.047786 -0.1695 1.139
## Finland        -0.11350  0.11133  0.11695 -0.04364 -0.017132 -0.1464 1.203
## France         -0.16600  0.14705  0.21900 -0.02942  0.023952  0.2765 1.226
## Germany        -0.00802  0.00822  0.00835 -0.00697 -0.000293 -0.0152 1.226
## Greece         -0.14820  0.16394  0.02861  0.15713 -0.059599 -0.2811 1.140
## Guatamala       0.01552 -0.05485  0.00614  0.00585  0.097217 -0.2305 1.085
## Honduras       -0.00226  0.00984 -0.01020  0.00812 -0.001887  0.0482 1.186
## Iceland         0.24789 -0.27355 -0.23265 -0.12555  0.184698 -0.4768 0.866
## India           0.02105 -0.01577 -0.01439 -0.01374 -0.018958  0.0381 1.202
## Ireland        -0.31001  0.29624  0.48156 -0.25733 -0.093317  0.5216 1.268
## Italy           0.06619 -0.07097  0.00307 -0.06999 -0.028648  0.1388 1.162
## Japan           0.63987 -0.65614 -0.67390  0.14610  0.388603  0.8597 1.085
## Korea          -0.16897  0.13509  0.21895  0.00511 -0.169492 -0.4303 0.870
## Luxembourg     -0.06827  0.06888  0.04380 -0.02797  0.049134 -0.1401 1.196
## Malta           0.03652 -0.04876  0.00791 -0.08659  0.153014  0.2386 1.128
## Norway          0.00222 -0.00035 -0.00611 -0.01594 -0.001462 -0.0522 1.168
## Netherlands     0.01395 -0.01674 -0.01186  0.00433  0.022591  0.0366 1.229
## New Zealand    -0.06002  0.06510  0.09412 -0.02638 -0.064740  0.1469 1.134
## Nicaragua      -0.01209  0.01790  0.00972 -0.00474 -0.010467  0.0397 1.174
## Panama          0.02828 -0.05334  0.01446 -0.03467 -0.007889 -0.1775 1.067
## Paraguay       -0.23227  0.16416  0.15826  0.14361  0.270478 -0.4655 0.873
## Peru           -0.07182  0.14669  0.09148 -0.08585 -0.287184  0.4811 0.831
## Philippines    -0.15707  0.22681  0.15743 -0.11140 -0.170674  0.4884 0.818
## Portugal       -0.02140  0.02551 -0.00380  0.03991 -0.028011 -0.0690 1.233
## South Africa    0.02218 -0.02030 -0.00672 -0.02049 -0.016326  0.0343 1.195
## South Rhodesia  0.14390 -0.13472 -0.09245 -0.06956 -0.057920  0.1607 1.313
## Spain          -0.03035  0.03131  0.00394  0.03512  0.005340 -0.0526 1.208
## Sweden          0.10098 -0.08162 -0.06166 -0.25528 -0.013316 -0.4526 1.086
## Switzerland     0.04323 -0.04649 -0.04364  0.09093 -0.018828  0.1903 1.147
## Turkey         -0.01092 -0.01198  0.02645  0.00161  0.025138 -0.1445 1.100
## Tunisia         0.07377 -0.10500 -0.07727  0.04439  0.103058 -0.2177 1.131
## United Kingdom  0.04671 -0.03584 -0.17129  0.12554  0.100314 -0.2722 1.189
## United States   0.06910 -0.07289  0.03745 -0.23312 -0.032729 -0.2510 1.655
## Venezuela      -0.05083  0.10080 -0.03366  0.11366 -0.124486  0.3071 1.095
## Zambia          0.16361 -0.07917 -0.33899  0.09406  0.228232  0.7482 0.512
## Jamaica         0.10958 -0.10022 -0.05722 -0.00703 -0.295461 -0.3456 1.200
## Uruguay        -0.13403  0.12880  0.02953  0.13132  0.099591 -0.2051 1.187
## Libya           0.55074 -0.48324 -0.37974 -0.01937 -1.024477 -1.1601 2.091
## Malaysia        0.03684 -0.06113  0.03235 -0.04956 -0.072294 -0.2126 1.113
##                  cook.d    hat inf
## Australia      8.04e-04 0.0677
## Austria        8.18e-04 0.1204
## Belgium        7.15e-03 0.0875
## Bolivia        7.28e-04 0.0895
## Brazil         1.40e-02 0.0696
## Canada         3.11e-04 0.1584
## Chile          3.78e-02 0.0373   *
## China          8.16e-03 0.0780
## Colombia       1.88e-03 0.0573
## Costa Rica     3.21e-02 0.0755
## Denmark        2.88e-02 0.0627
```

```
## Ecuador        5.82e-03 0.0637
## Finland        4.36e-03 0.0920
## France         1.55e-02 0.1362
## Germany        4.74e-05 0.0874
## Greece         1.59e-02 0.0966
## Guatamala      1.07e-02 0.0605
## Honduras       4.74e-04 0.0601
## Iceland        4.35e-02 0.0705
## India          2.97e-04 0.0715
## Ireland        5.44e-02 0.2122
## Italy          3.92e-03 0.0665
## Japan          1.43e-01 0.2233
## Korea          3.56e-02 0.0608
## Luxembourg     3.99e-03 0.0863
## Malta          1.15e-02 0.0794
## Norway         5.56e-04 0.0479
## Netherlands    2.74e-04 0.0906
## New Zealand    4.38e-03 0.0542
## Nicaragua      3.23e-04 0.0504
## Panama         6.33e-03 0.0390
## Paraguay       4.16e-02 0.0694
## Peru           4.40e-02 0.0650
## Philippines    4.52e-02 0.0643
## Portugal       9.73e-04 0.0971
## South Africa   2.41e-04 0.0651
## South Rhodesia 5.27e-03 0.1608
## Spain          5.66e-04 0.0773
## Sweden         4.06e-02 0.1240
## Switzerland    7.33e-03 0.0736
## Turkey         4.22e-03 0.0396
## Tunisia        9.56e-03 0.0746
## United Kingdom 1.50e-02 0.1165
## United States  1.28e-02 0.3337   *
## Venezuela      1.89e-02 0.0863
## Zambia         9.66e-02 0.0643   *
## Jamaica        2.40e-02 0.1408
## Uruguay        8.53e-03 0.0979
## Libya          2.68e-01 0.5315   *
## Malaysia       9.11e-03 0.0652
```

There are other indices for detecting influential points, such as DFBETAs and DFFITs.

`https://cran.r-project.org/web/packages/olsrr/vignettes/influence_measures.html`

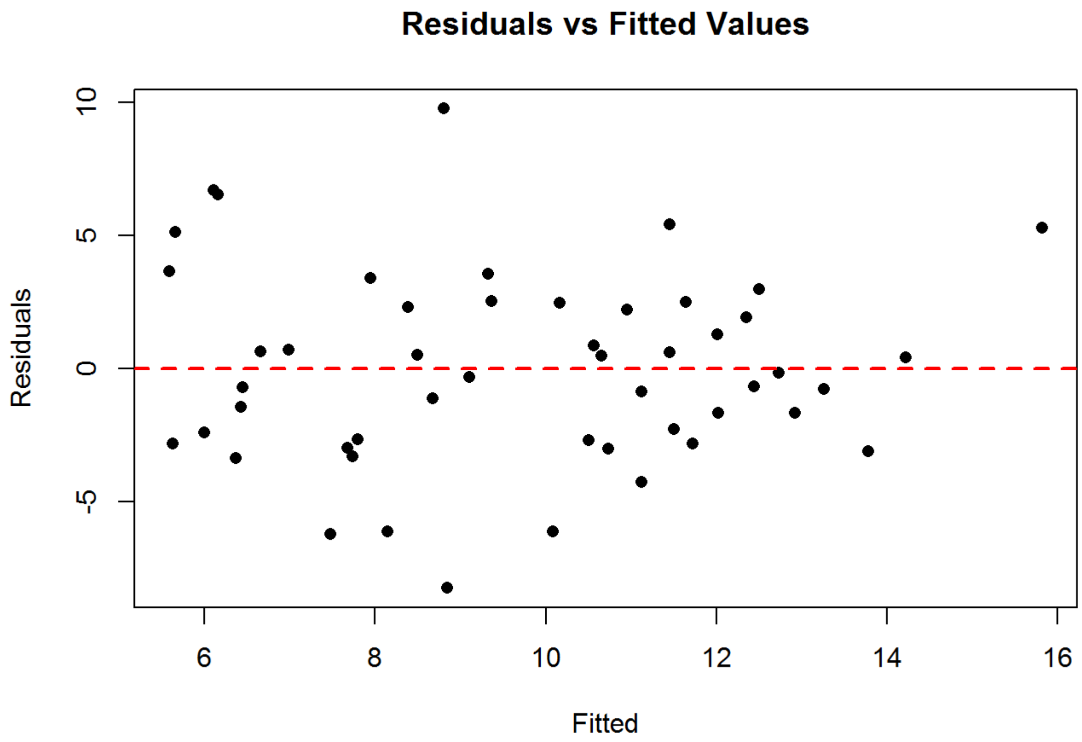
---

4. HYPOTHESES TESTING

Omoschedasticity

4.a Plot residuals ($\hat{\varepsilon}$) vs fitted values ($\hat{y}$).

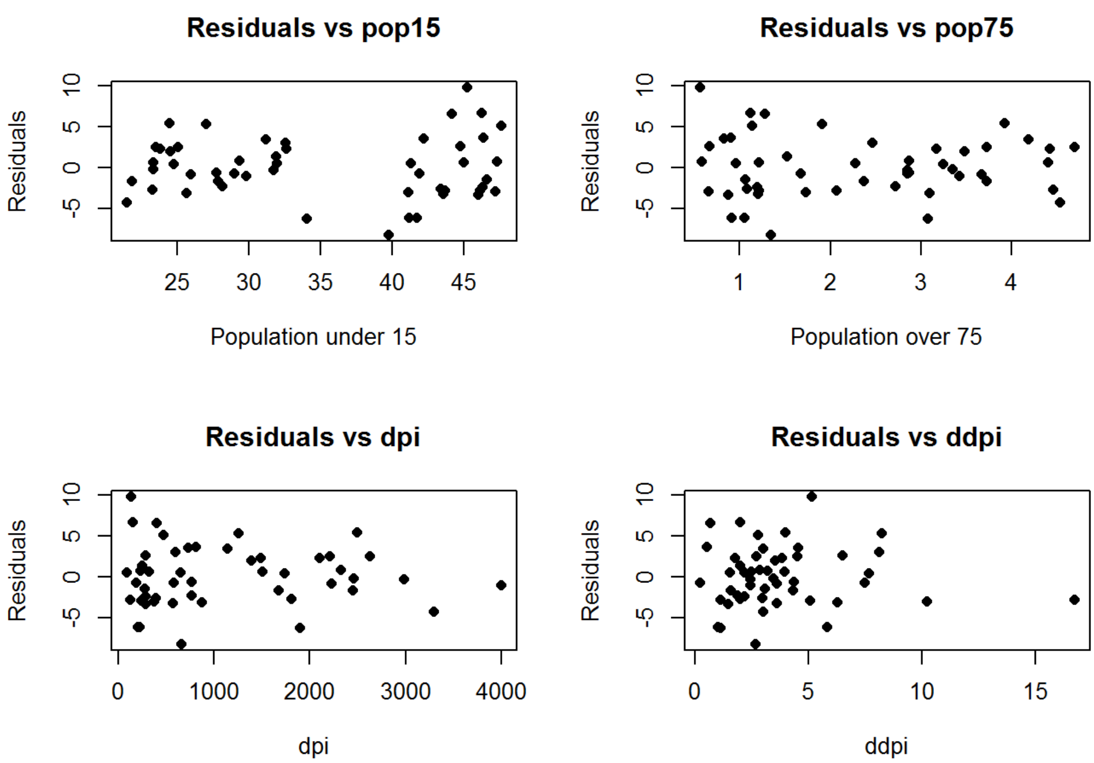
```
plot( g$fit, g$res, xlab = "Fitted", ylab = "Residuals", main = "Residuals vs Fitted Values", pch = 16 )
abline( h = 0, lwd = 2, lty = 2, col = 'red' )  # variabilitÃ  sembra sufficientemente uniforme
```



Nonlinearity

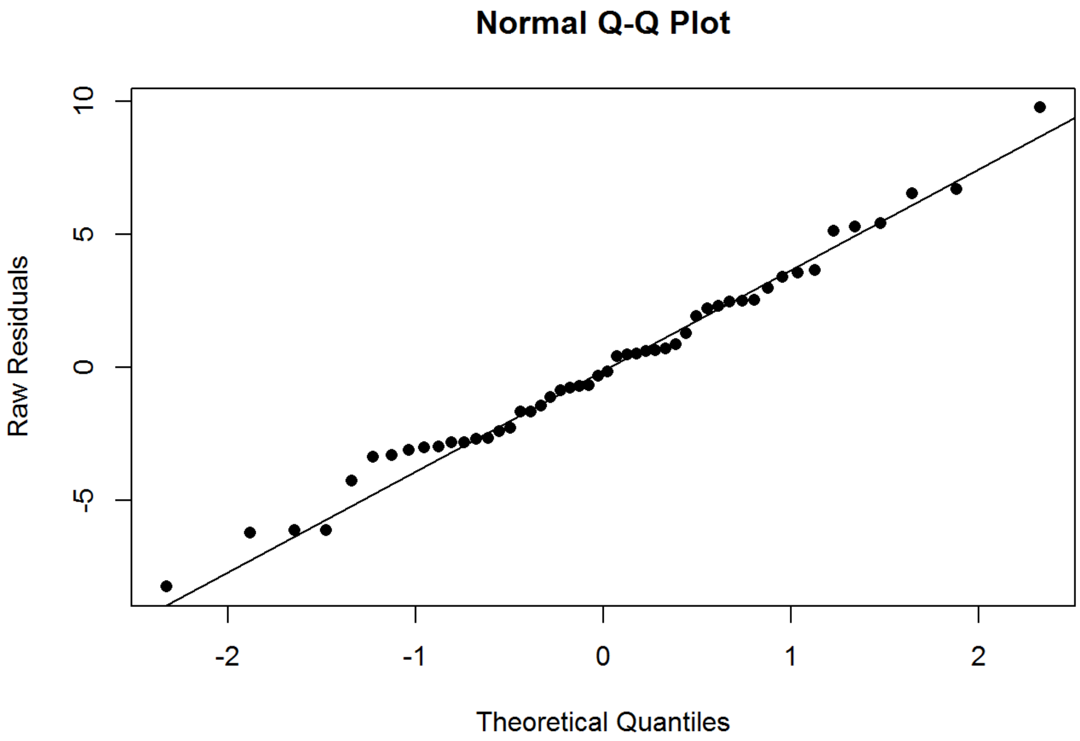
4.b Plot residuals ($\hat{\varepsilon}$) vs predictors (x_i).

```
par( mfrow = c( 2, 2 ) )
plot( savings$pop15, g$res, xlab = "Population under 15", ylab = "Residuals",
      main = "Residuals vs pop15", pch = 16 )
plot( savings$pop75, g$res, xlab = "Population over 75", ylab = "Residuals",
      main = "Residuals vs pop75", pch = 16 )
plot( savings$dpi, g$res, xlab = "dpi", ylab = "Residuals", main = "Residuals vs dpi", pch = 16 )
plot( savings$ddpi, g$res, xlab = "ddpi", ylab = "Residuals", main = "Residuals vs ddpi", pch = 16 )
```



Normality

```
# QQ plot
qqnorm( g$res, ylab = "Raw Residuals", pch = 16 )
qqline( g$res )
```



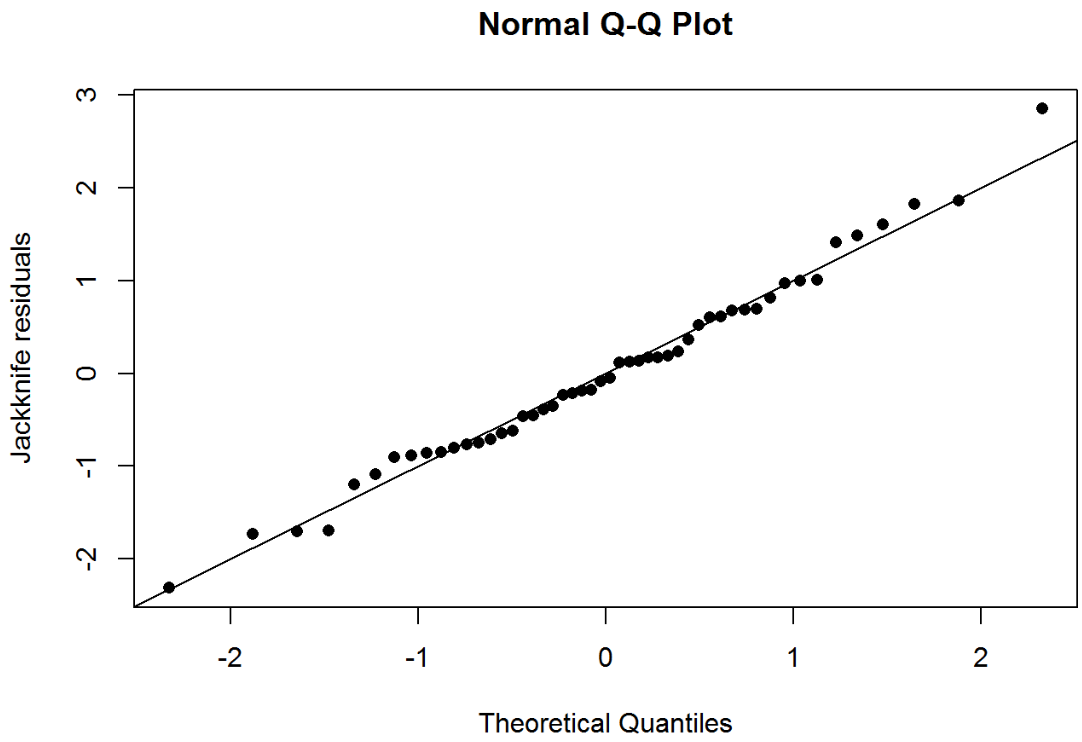

```
# Andamento adeguatamente rettilineo, l'ipotesi Ã¨ verificata

# Shapiro-Wilk normality test
# p-val molto alto = > NON rifiuto H0: dati Gaussiani
shapiro.test( g$res )
##
##  Shapiro-Wilk normality test
##
## data:  g$res
## W = 0.98698, p-value = 0.8524
```

REMARK

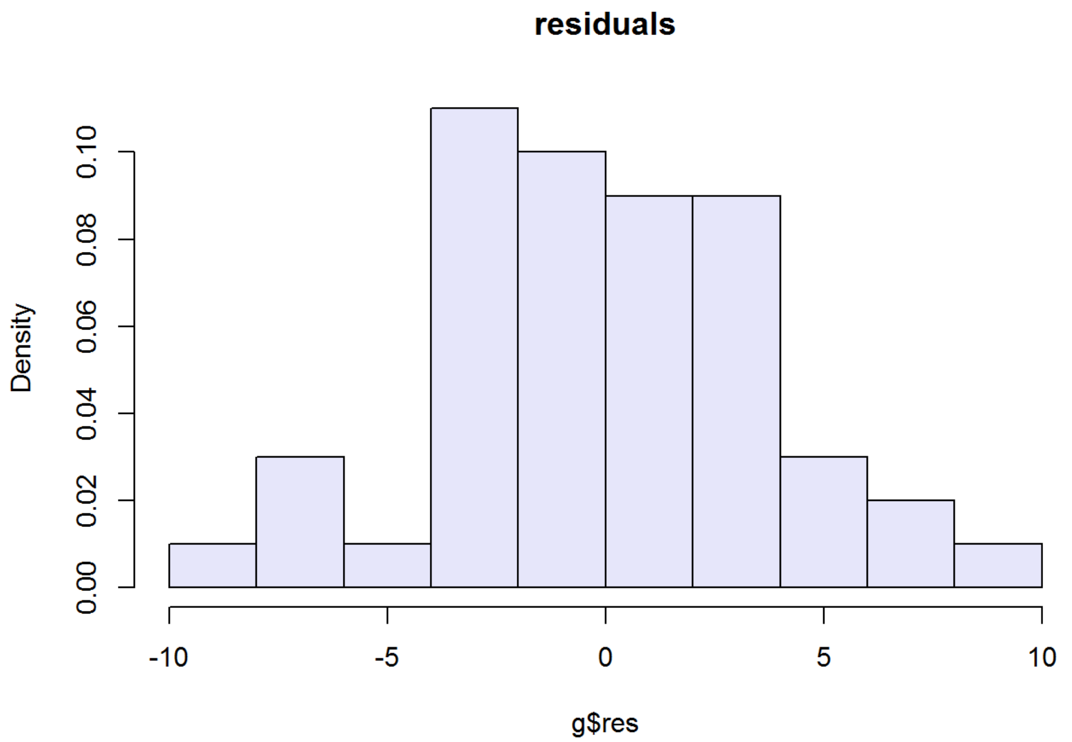
rstandard function automatically computes the studentized residuals.

rstudent function automatically computes the jackknife residuals.

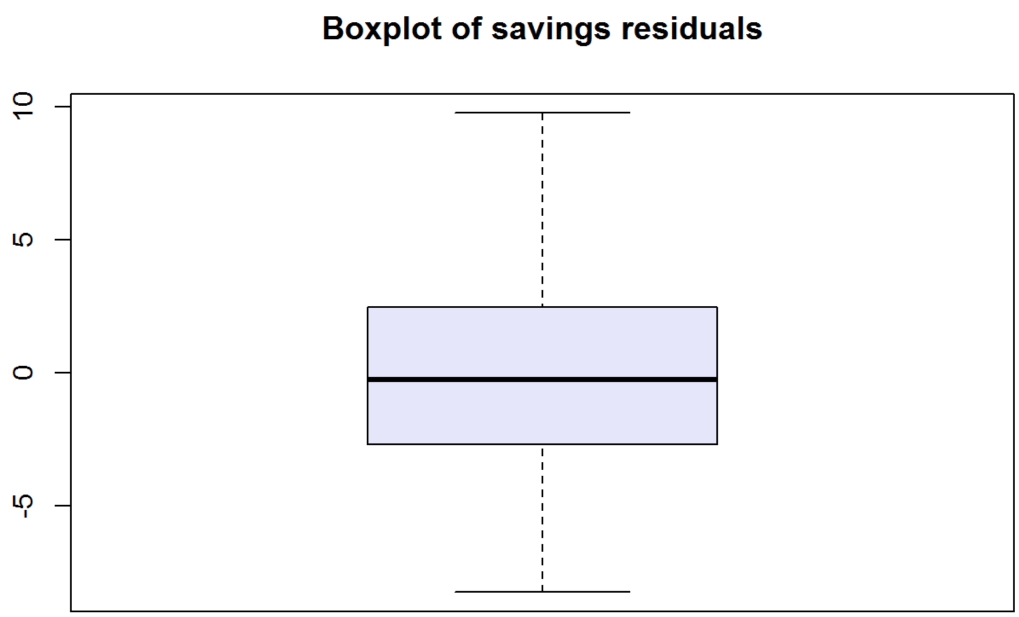
```
qqnorm( rstudent( g ), ylab = "Jackknife residuals", pch = 16 )
abline( 0, 1 )
```




```
# altri strumenti utili...
hist( g$res, 10, probability = TRUE, col = 'lavender', main = 'residuals'  )
```




```
boxplot( g$res, main = "Boxplot of savings residuals", pch = 16, col = 'lavender' )
```



Nonlinearity/Collinearity

Added Variable Plots (partial regression plots)

These functions build added-variable (also called partial-regression) plots for linear and generalized linear models.

Given a set of predictors and an outcome, the function *avPlots* allows to visualize the effect of a specific predictor that is not explained by the others.

Let's suppouse to focus on X_1, the function plots the residuals of:

$$Y \sim X_2 + X_3 + \ldots + X_p,$$

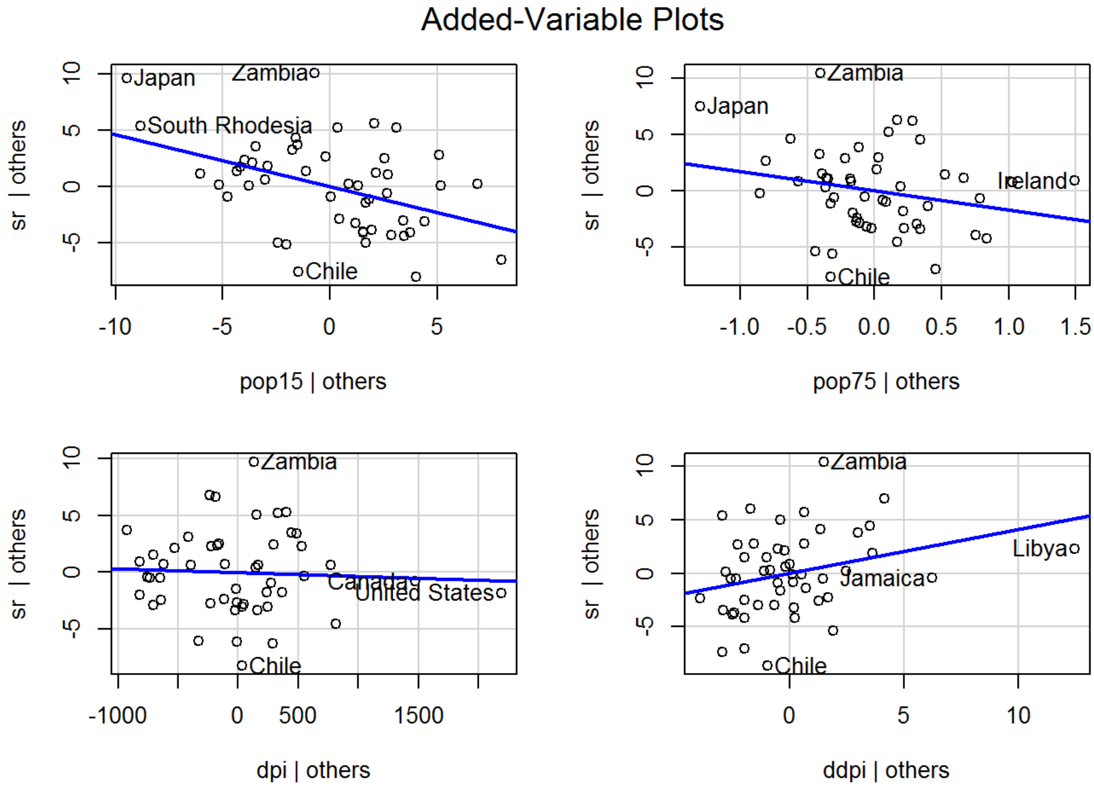
vs the residuals of:

$$X_1 \sim X_2 + X_3 + \ldots + X_p.$$

The regression of residuals gives the regression coefficient X_1, by considering all the other variables, this means its net effect on the outcome Y.

This plot can be also used for the diagnosis influential points and nonlinearity.

```
avPlots( g )
```




```
# Si confrontino tali grafici con
summary( g )
##
## Call:
## lm(formula = sr ~ pop15 + pop75 + dpi + ddpi, data = savings)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2422 -2.6857 -0.2488  2.4280  9.7509
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 28.5660865  7.3545161   3.884 0.000334 ***
## pop15       -0.4611931  0.1446422  -3.189 0.002603 **
## pop75       -1.6914977  1.0835989  -1.561 0.125530
## dpi         -0.0003369  0.0009311  -0.362 0.719173
## ddpi         0.4096949  0.1961971   2.088 0.042471 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.803 on 45 degrees of freedom
## Multiple R-squared:  0.3385, Adjusted R-squared:  0.2797
## F-statistic: 5.756 on 4 and 45 DF,  p-value: 0.0007904
```

VIF Variance Inflation Factor is another index of collinearity.

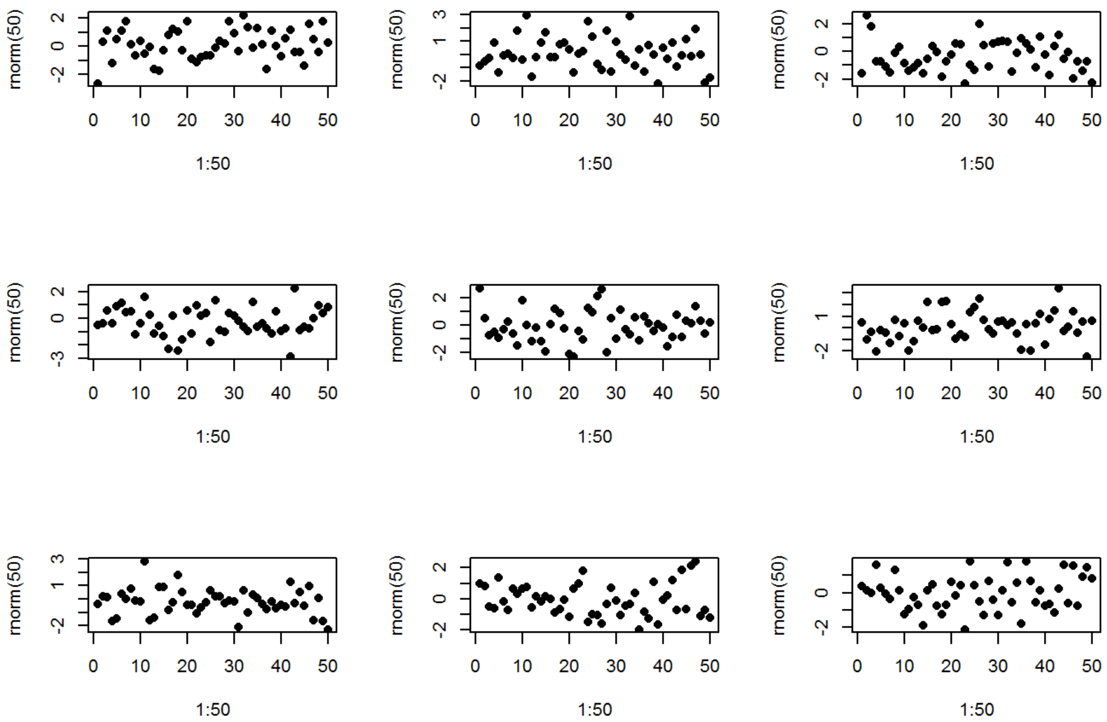
$$Var(\beta_j) = \frac{S^2}{(n-1)\cdot S_j^2}\cdot\frac{1}{1-R_j^2}$$

```
vif( g )
##    pop15    pop75      dpi     ddpi
## 5.937661 6.629105 2.884369 1.074309
```

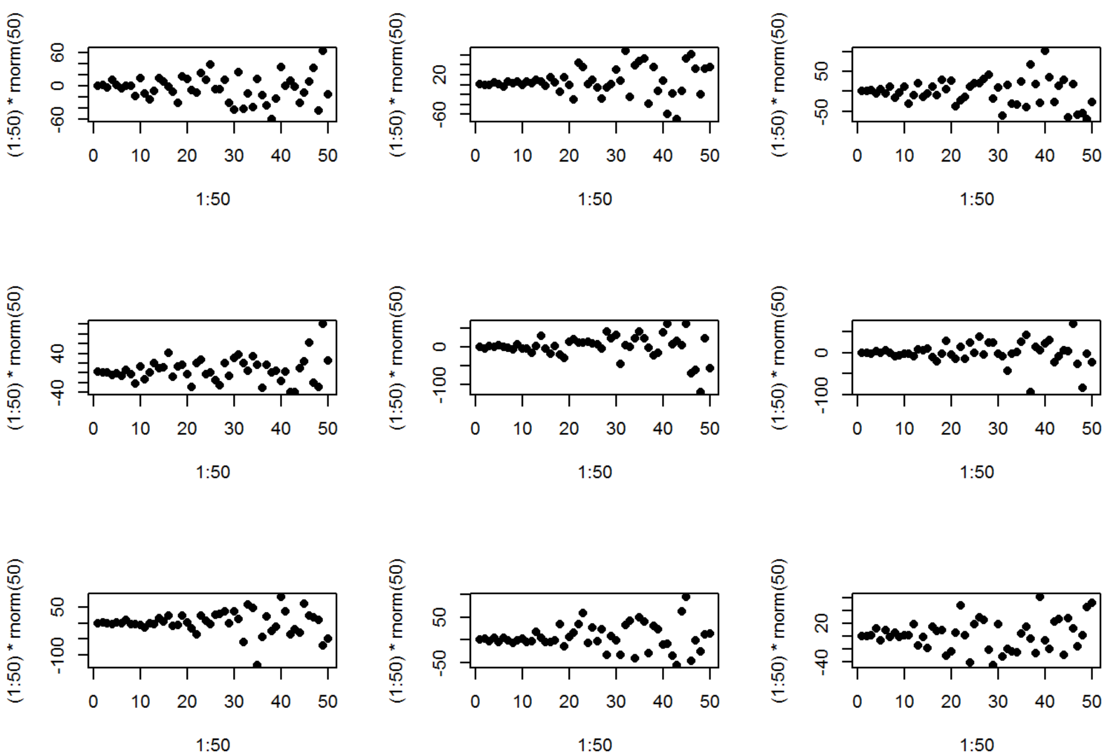
In this case, **pop75** and **pop15** show the highest collinearity.

Violation of omoschedasticity

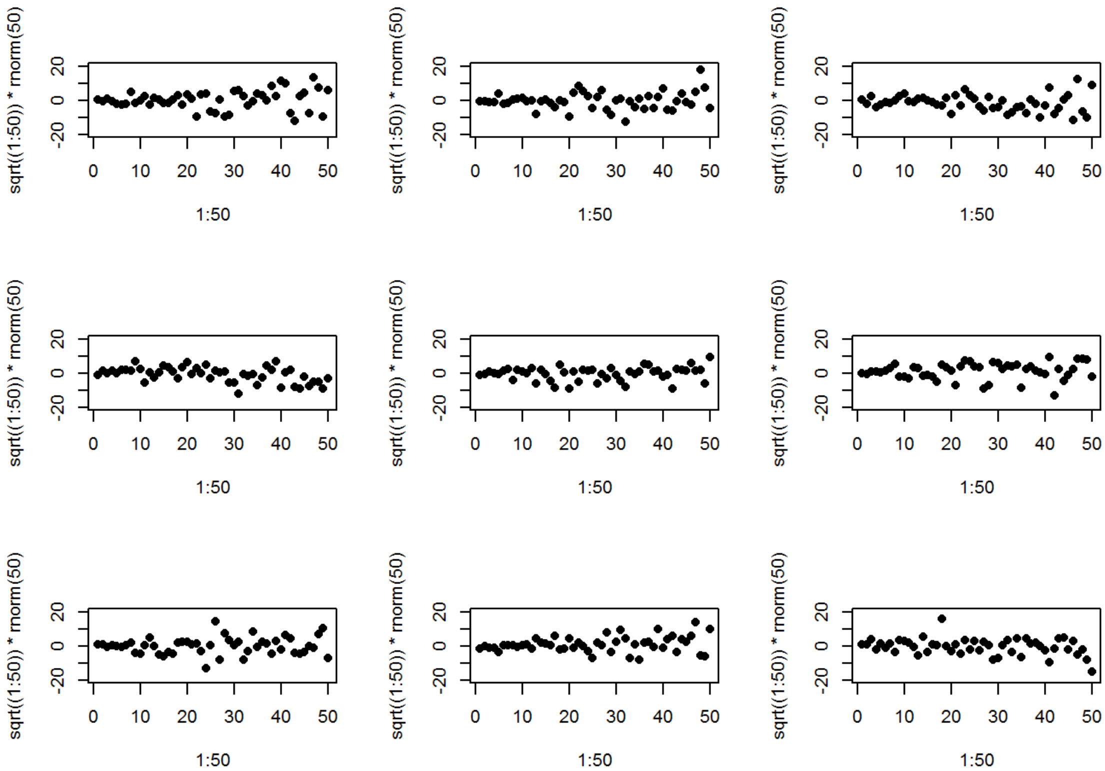
```
par( mfrow = c( 3, 3 ) )

# OmoschedasticitÃ
for( i in 1:9 )
  plot( 1:50, rnorm( 50 ), pch = 16 )
```



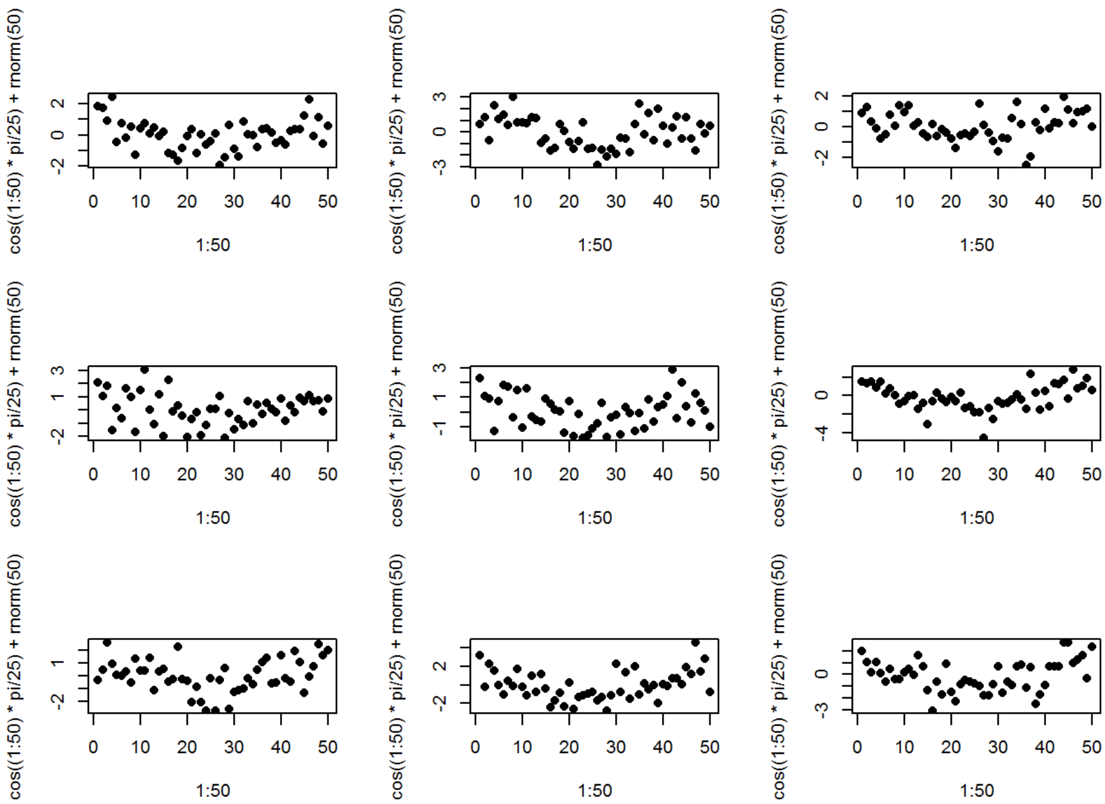

```
# EteroschedasticitÃ  marcata (varianza funzione lineare di x)
for( i in 1:9 )
  plot( 1:50, ( 1:50 ) * rnorm( 50 ), pch = 16 )
```



```
# EteroschedasticitÃ  blanda (varianza funzione sublineare di x)
for( i in 1:9 )
  plot( 1:50, sqrt( ( 1:50 ) ) * rnorm( 50 ), ylim = c( -20, 20 ), pch = 16 )
```

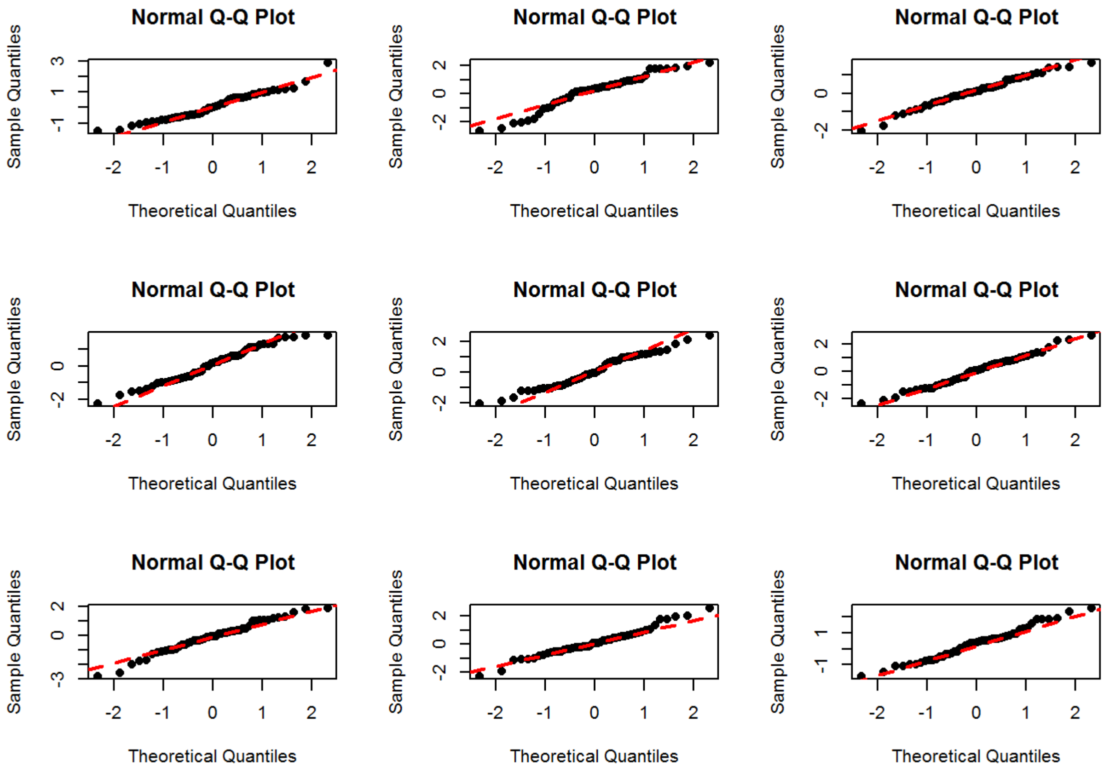



```
# Non linearitÃ  (varianza funzione nonlineare di x)
for( i in 1:9 )
  plot( 1:50, cos( ( 1:50 ) * pi/25 ) + rnorm( 50 ), pch = 16 )
```

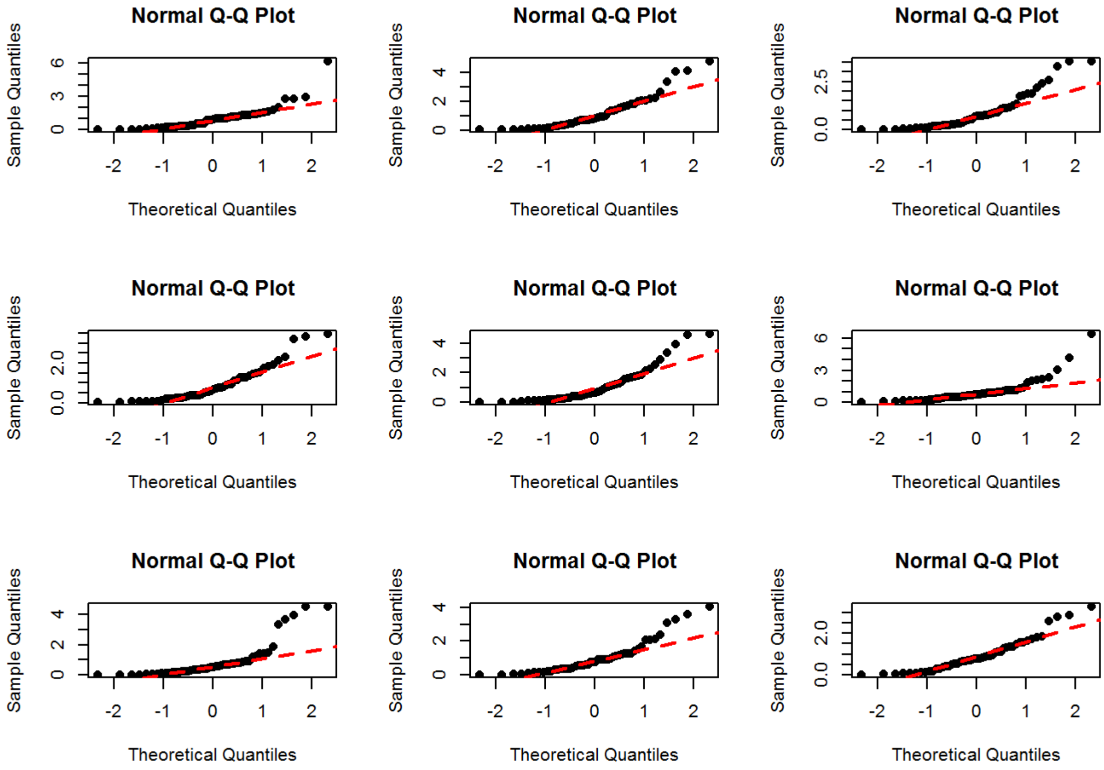


Violation of normality

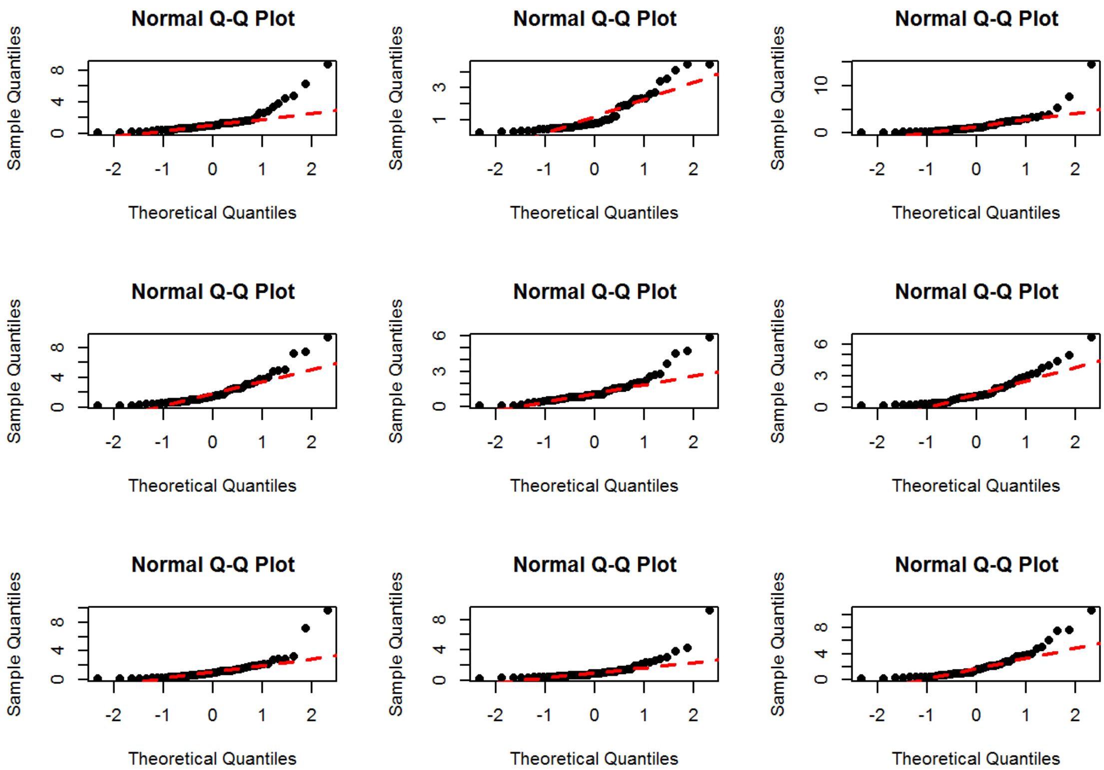
```
par( mfrow = c( 3, 3 ) )

# Normali
for( i in 1:9 )
{
  D = rnorm( 50 )

  qqnorm( D, pch = 16 )

  qqline( D, lty = 2, lwd = 2, col = 'red' )
}
```



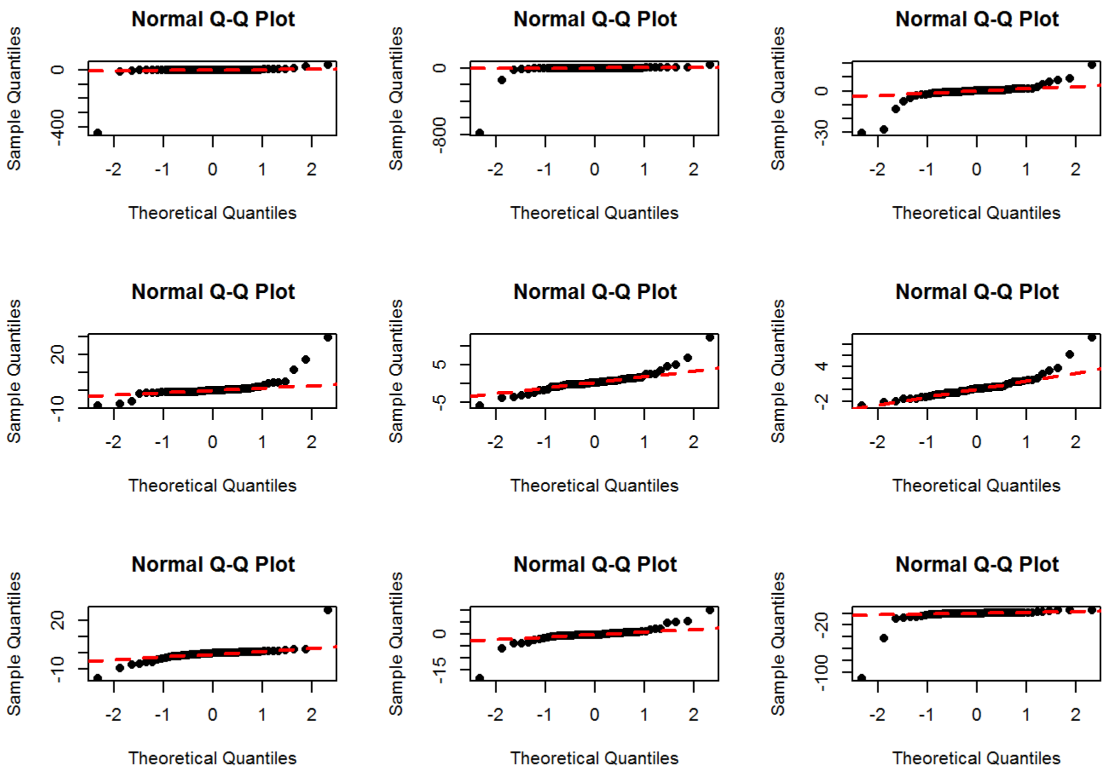

```


# Esponenziali
for( i in 1:9 )
{
  D = rexp( 50 )

  qqnorm( D, pch = 16 )

  qqline( D, lty = 2, lwd = 2, col = 'red' )
}
```



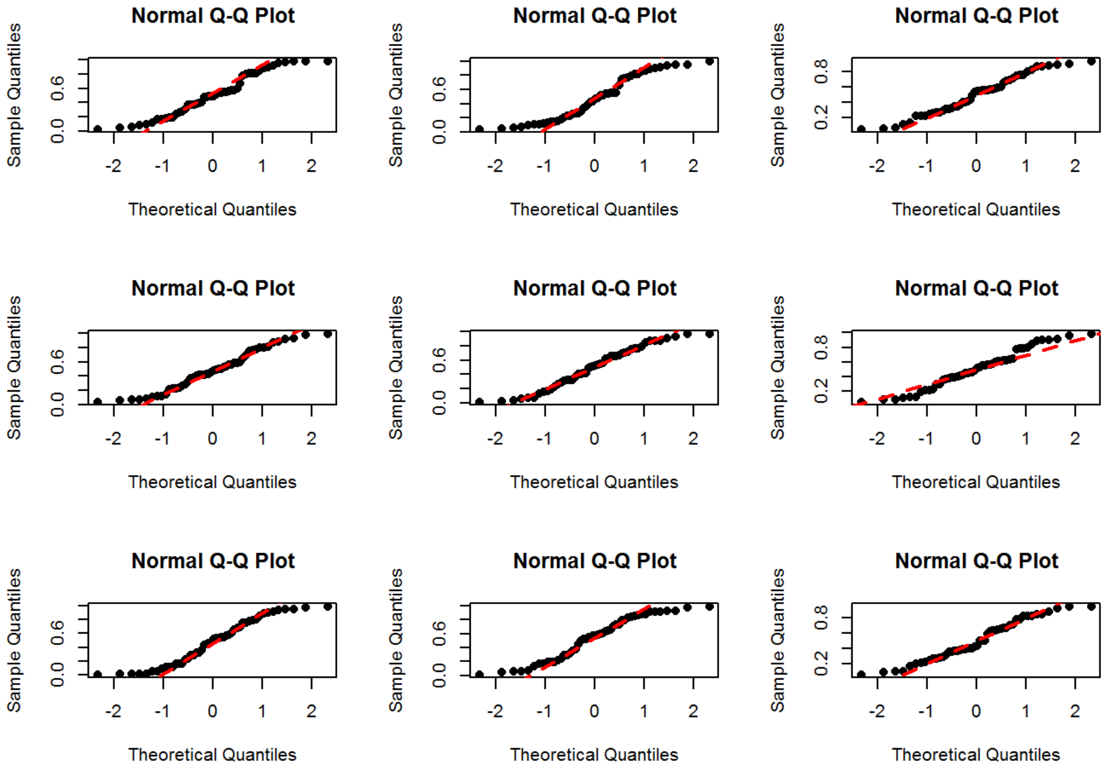

```r
par( mfrow = c(3,3) )
# Log-normali
for( i in 1:9 )
{
  D = exp( rnorm( 50 ) )

  qqnorm( D, pch = 16 )

  qqline( D, lty = 2, lwd = 2, col = 'red' )
}
```



```
# Cauchy
for( i in 1:9 )
{
  D = rcauchy( 50 )

  qqnorm( D, pch = 16 )

  qqline( D, lty = 2, lwd = 2, col = 'red' )
}
```




```
par( mfrow = c(3,3) )
# Uniforme
for( i in 1:9 )
{
  D = runif( 50 )

  qqnorm( D, pch = 16 )

  qqline( D, lty = 2, lwd = 2, col = 'red' )
}
```



5. Transformation: Box-Cox

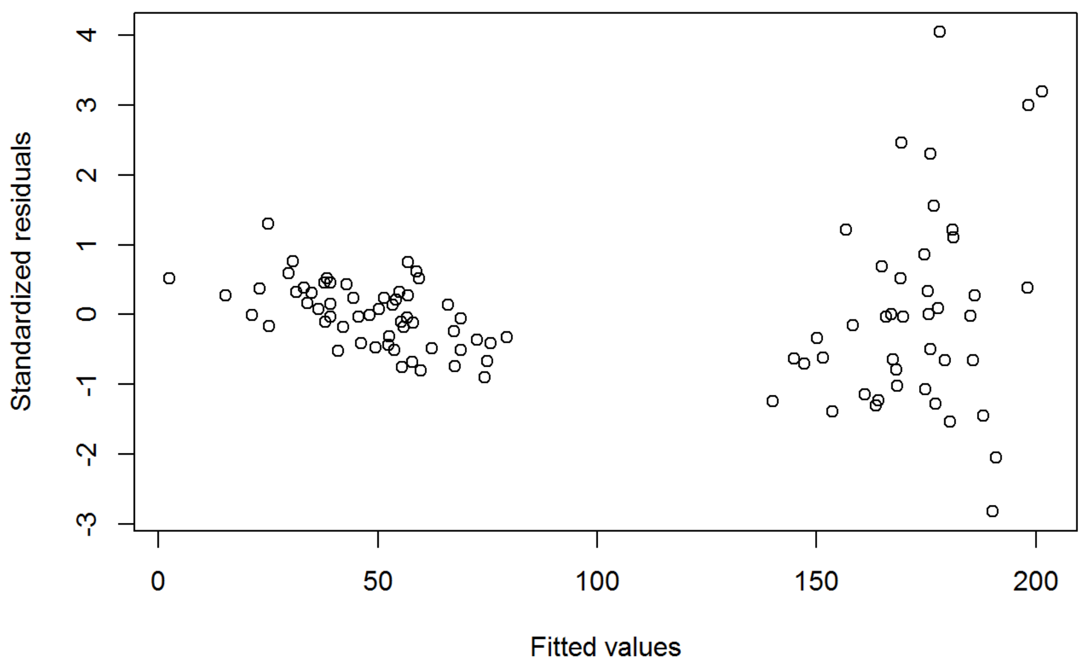
In this section we would like to answer the following question: what should we do when there is a clear violation of hypotheses? The answer consists in investigating variable transformations (transformation of the outcome).

Warning Transforming a variable can lead to a more difficult interpretation of the model.

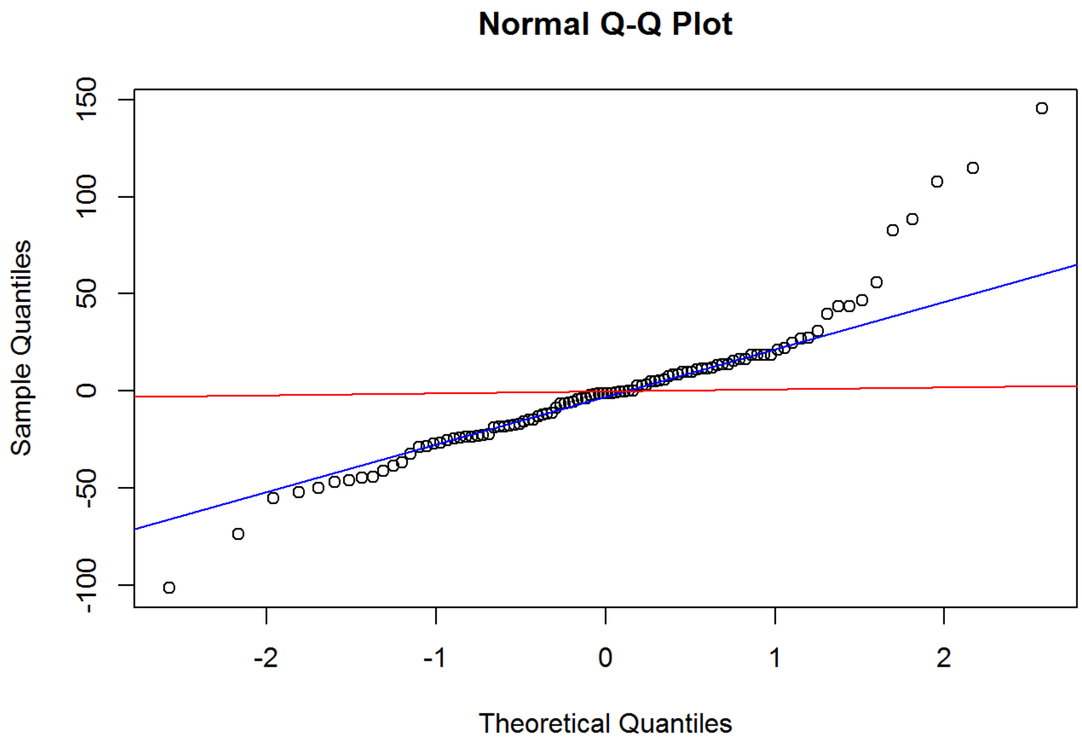
An algorithm that helps us in variable transformation is the *Box-Cox* algorithm. It detects the best λ among a family of transformations ($\frac{y^\lambda - 1}{\lambda}$, if $\lambda \neq 0$, otherwise $log(y)$) in oder to gain the Normality/Omoschedasticity for **positive data**.

Here we report an example (multiple linear regression). We consider a dataset with 4 columns: freq, length, sex, age. Then we apply a linear regression model in which 'freq' is the outcome and the others are the predictors. Finally, we test the Normality hypothesis of the model.

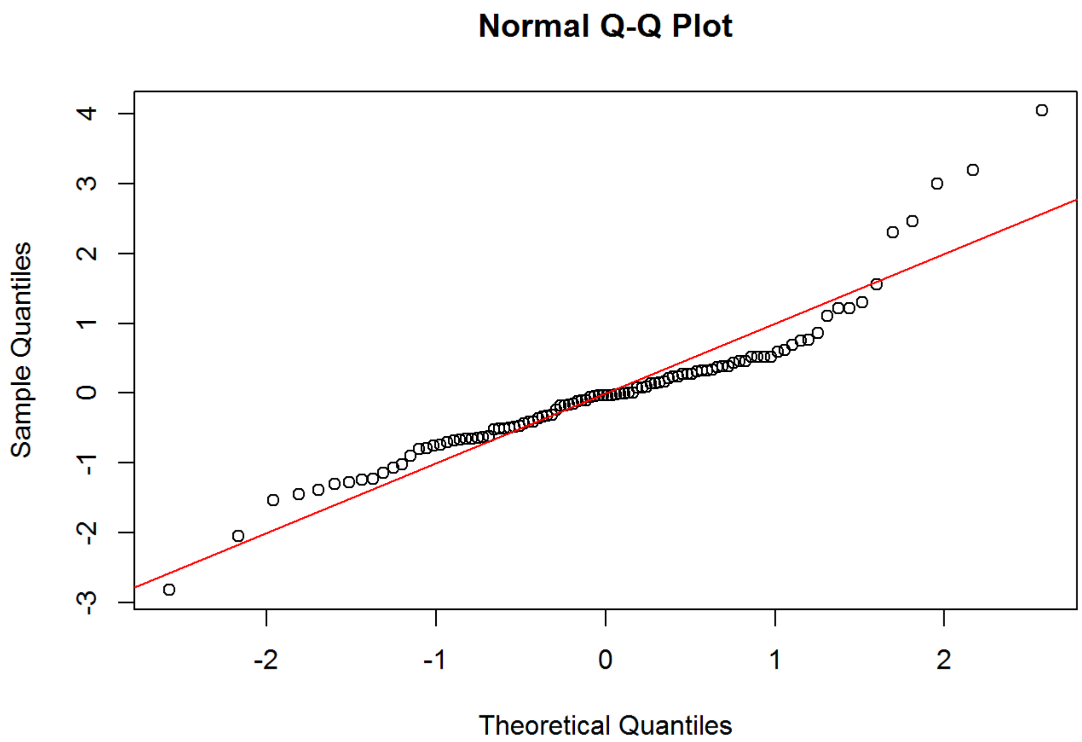
```
#import data
dati = read.table( 'delfini.txt', header = T )

mod = lm( freq ~ length + sex + age, data = dati )
summary( mod )
##
## Call:
## lm(formula = freq ~ length + sex + age, data = dati)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -101.319  -19.543   -1.146   13.470  145.273
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -187.696     79.161  -2.371 0.019734 *
## length        77.860     19.329   4.028 0.000112 ***
## sex          123.549      7.388  16.723  < 2e-16 ***
## age            6.930      4.647   1.491 0.139225
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.49 on 96 degrees of freedom
## Multiple R-squared:  0.7575, Adjusted R-squared:  0.7499
## F-statistic: 99.96 on 3 and 96 DF,  p-value: < 2.2e-16

#summary(mod)
mod_res <- ( mod$residuals - mean( mod$residuals ) )/sd( mod$residuals )
plot( mod$fitted, mod_res, xlab = 'Fitted values',  ylab = 'Standardized residuals'  )
```




```
qqnorm( mod$residuals )
qqline( mod$residuals, col = 'blue' )
abline( 0, 1, col = 'red' )
```



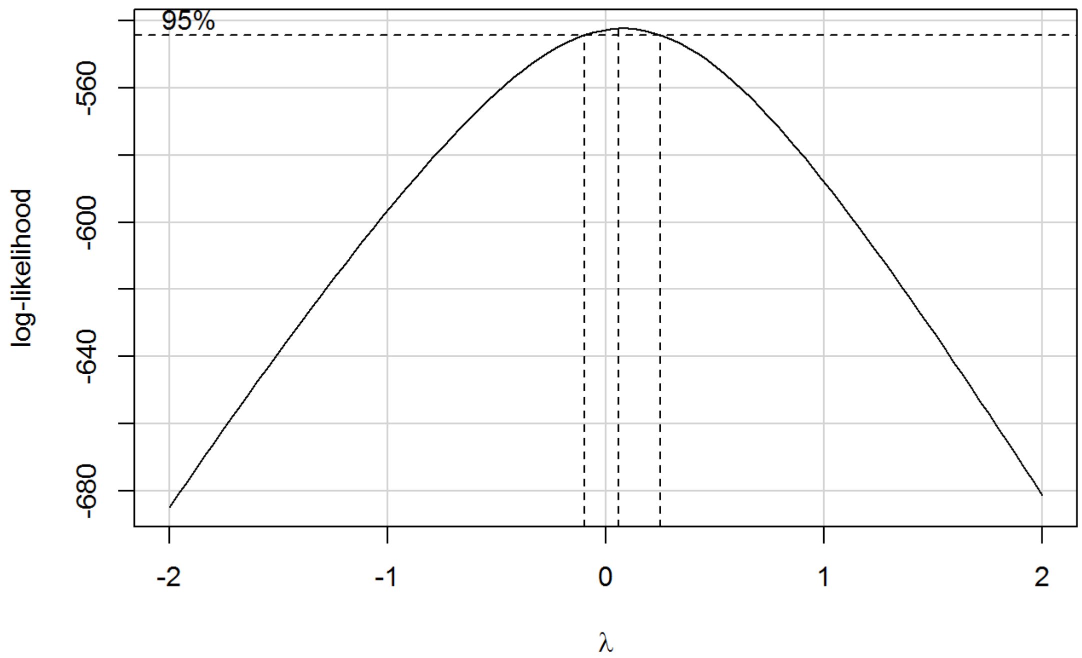

```
qqnorm( mod_res )
abline( 0, 1, col = 'red' )
```




```
shapiro.test( mod_res )
##
##  Shapiro-Wilk normality test
##
## data:  mod_res
## W = 0.90916, p-value = 3.914e-06
```

Good fit of the model: $R^2 = 75.75\%$ and the predictor is significant $p - value < 2.2e - 16$. However, there is a clear evidence of nonnormality (we can see it both with the qqplot and the Shapiro-Wilk test). So, we apply the Box-Cox transformation. *Remark* we can apply the Box-Cox transformation, because *freq* is positive.

```
b = boxCox( mod )
```

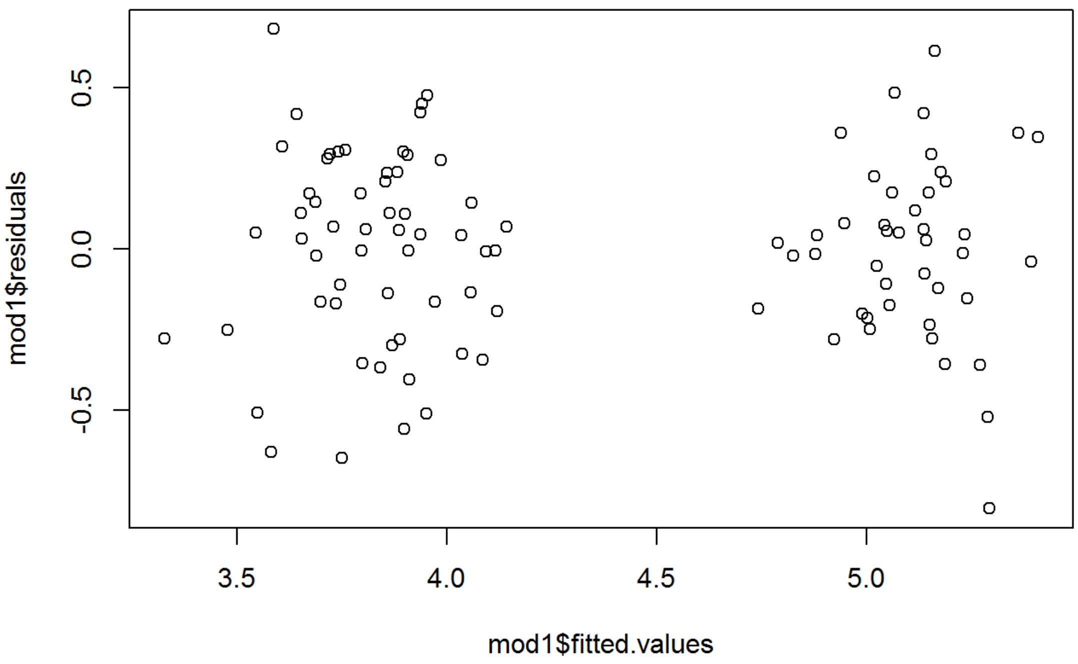



```
names(b)
## [1] "x" "y"
#y likelihood evaluation
#x lambda evaluated
best_lambda_ind = which.max( b$y )
best_lambda = b$x[ best_lambda_ind ]
best_lambda
## [1] 0.06060606
```

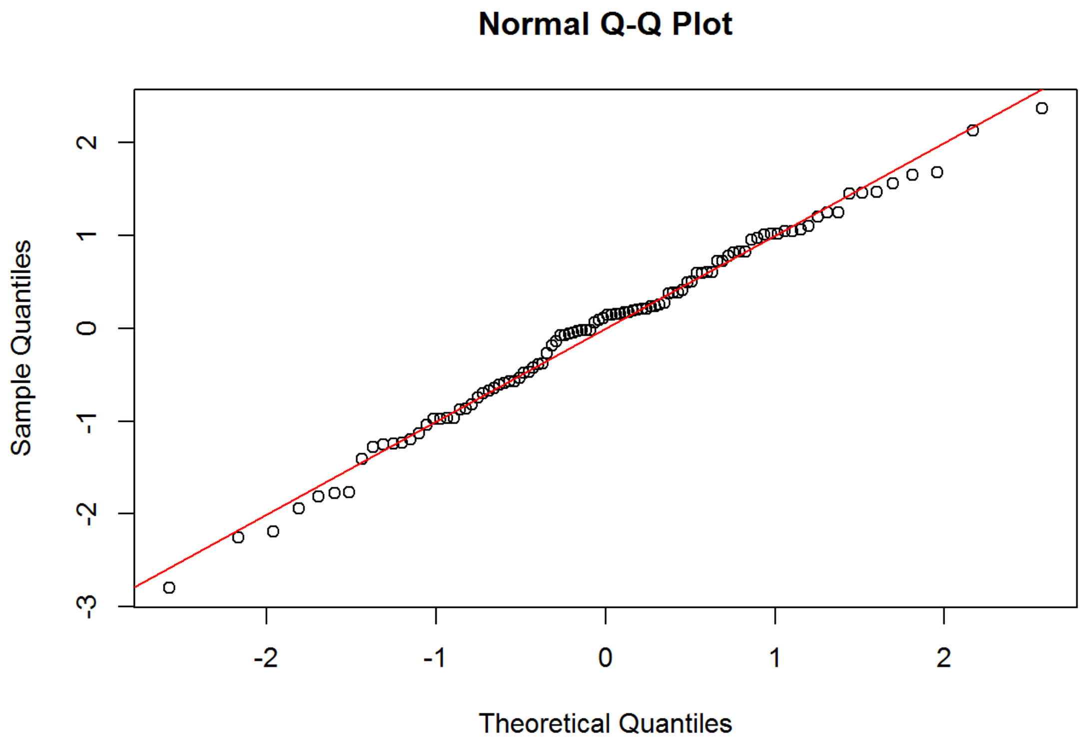
We can see that the best transformation is the one related to the *maximum* of the curve. The estimates are obtained through *Maximum Likelihood* method. According to this method, the best λ is very close to 0. Despite of that, we would prefer the most interpretable transformation among the allowed ones: $\lambda = 0$. So the chosen transformation is the logarithm of the outcome variable.

Finally, we test the new model and we investigate the standardized residuals.

```
dati$freq2 = log( dati$freq )

mod1 = lm( freq2 ~ length + sex + age, data = dati )
summary( mod1 )
##
## Call:
## lm(formula = freq2 ~ length + sex + age, data = dati)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.80618 -0.18850  0.03721  0.20845  0.68222
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.12179    0.63494   1.767   0.0804 .
## length       0.81003    0.15503   5.225 1.01e-06 ***
## sex          1.27232    0.05926  21.470  < 2e-16 ***
## age          0.08857    0.03728   2.376   0.0195 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2927 on 96 degrees of freedom
## Multiple R-squared:  0.8377, Adjusted R-squared:  0.8326
## F-statistic: 165.1 on 3 and 96 DF,  p-value: < 2.2e-16

plot( mod1$fitted.values, mod1$residuals )
```




```
qqnorm( ( mod1$residuals - mean( mod1$residuals ) )/sd( mod1$residuals ) )
abline( 0, 1, col = 'red' )
```




```
shapiro.test( mod1$residuals )
##
##  Shapiro-Wilk normality test
##
## data:  mod1$residuals
## W = 0.99101, p-value = 0.7458
```

We can conclude that the Box-Cox transformation helped us in improving the model.

---

6. Variable Selection: stepwise procedure

6.a Load `state` dataset, in which data about the 50 states of USA are collected. The variables are population estimate as of July 1, 1975, per capita:

- **income** (1974);
- **illiteracy** (1970, percent of population);
- **life expectancy** in years (1969-71);
- **murder** and non-negligent manslaughter rate per 100, 000 population (1976);
- **percent high-school graduates** (1970);
- **mean number of days with min temperature 32 degrees**(1931-1960);
- **in capital or large city**;
- **land area** (in square miles).

We will take life expectancy as the response and the remaining variables as predictors.

```
data( state )
statedata = data.frame( state.x77, row.names = state.abb, check.names = T )

head( statedata )
##    Population Income Illiteracy Life.Exp Murder HS.Grad Frost   Area
## AL       3615   3624        2.1    69.05   15.1    41.3    20  50708
## AK        365   6315        1.5    69.31   11.3    66.7   152 566432
## AZ       2212   4530        1.8    70.55    7.8    58.1    15 113417
## AR       2110   3378        1.9    70.66   10.1    39.9    65  51945
## CA      21198   5114        1.1    71.71   10.3    62.6    20 156361
## CO       2541   4884        0.7    72.06    6.8    63.9   166 103766
```

6.b Fit and investigate the complete linear model on this data.

Which predictors should be included - can you tell from the p-values? Looking at the coefficients, can you see what operation would be helpful?

Does the murder rate decrease life expectancy - that's obvious a priori, but how should these results be interpreted?

```
g = lm( Life.Exp ~ ., data = statedata )
summary( g )
##
## Call:
## lm(formula = Life.Exp ~ ., data = statedata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.48895 -0.51232 -0.02747  0.57002  1.49447
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.094e+01  1.748e+00  40.586  < 2e-16 ***
## Population   5.180e-05  2.919e-05   1.775   0.0832 .
## Income      -2.180e-05  2.444e-04  -0.089   0.9293
## Illiteracy   3.382e-02  3.663e-01   0.092   0.9269
## Murder      -3.011e-01  4.662e-02  -6.459 8.68e-08 ***
## HS.Grad      4.893e-02  2.332e-02   2.098   0.0420 *
## Frost       -5.735e-03  3.143e-03  -1.825   0.0752 .
## Area        -7.383e-08  1.668e-06  -0.044   0.9649
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7448 on 42 degrees of freedom
## Multiple R-squared:  0.7362, Adjusted R-squared:  0.6922
## F-statistic: 16.74 on 7 and 42 DF,  p-value: 2.534e-10
```

Manual backward selection method

At each step we remove the predictor with the largest p-value and (ideally) stop when we have only predictors with p-values below 0.05

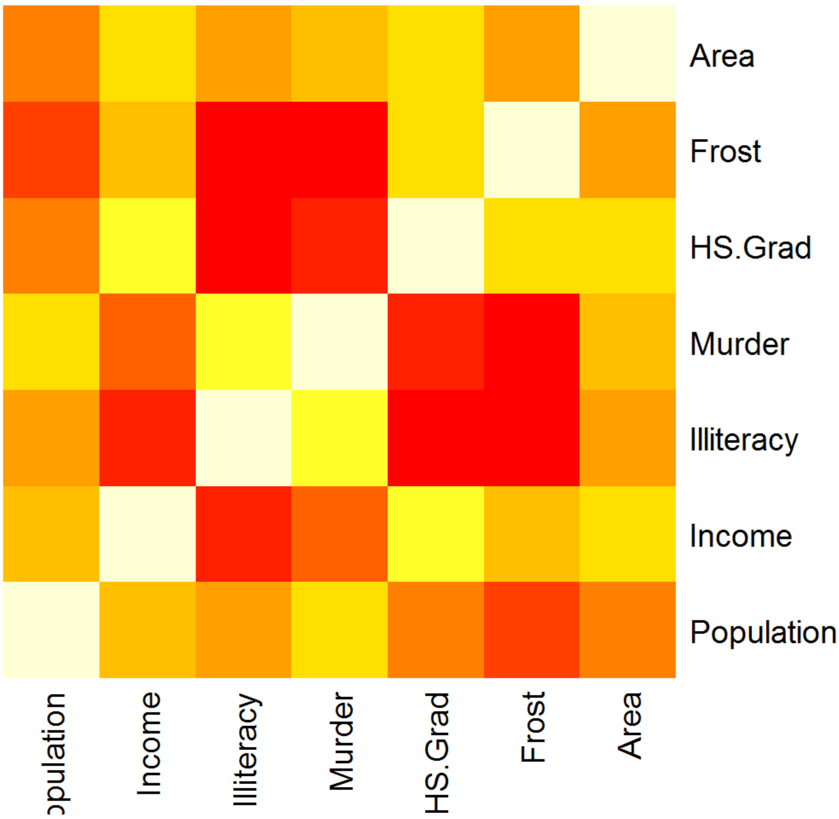
```
help('update')
## starting httpd help server ... done
help('update.formula')

# remove Area
g1 = update( g, . ~ . - Area )
summary( g1 )
##
## Call:
## lm(formula = Life.Exp ~ Population + Income + Illiteracy + Murder +
##     HS.Grad + Frost, data = statedata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.49047 -0.52533 -0.02546  0.57160  1.50374
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.099e+01  1.387e+00  51.165  < 2e-16 ***
## Population   5.188e-05  2.879e-05   1.802   0.0785 .
## Income      -2.444e-05  2.343e-04  -0.104   0.9174
## Illiteracy   2.846e-02  3.416e-01   0.083   0.9340
## Murder      -3.018e-01  4.334e-02  -6.963 1.45e-08 ***
## HS.Grad      4.847e-02  2.067e-02   2.345   0.0237 *
## Frost       -5.776e-03  2.970e-03  -1.945   0.0584 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7361 on 43 degrees of freedom
## Multiple R-squared:  0.7361, Adjusted R-squared:  0.6993
## F-statistic: 19.99 on 6 and 43 DF,  p-value: 5.362e-11

# remove Illiteracy
g2 = update( g1, . ~ . - Illiteracy )
summary( g2 )
##
## Call:
## lm(formula = Life.Exp ~ Population + Income + Murder + HS.Grad +
##     Frost, data = statedata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.4892 -0.5122 -0.0329  0.5645  1.5166
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.107e+01  1.029e+00  69.067  < 2e-16 ***
## Population   5.115e-05  2.709e-05   1.888   0.0657 .
## Income      -2.477e-05  2.316e-04  -0.107   0.9153
## Murder      -3.000e-01  3.704e-02  -8.099 2.91e-10 ***
## HS.Grad      4.776e-02  1.859e-02   2.569   0.0137 *
## Frost       -5.910e-03  2.468e-03  -2.395   0.0210 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7277 on 44 degrees of freedom
## Multiple R-squared:  0.7361, Adjusted R-squared:  0.7061
## F-statistic: 24.55 on 5 and 44 DF,  p-value: 1.019e-11

# Remove Income
g3 = update( g2, . ~ . - Income )
summary( g3 )
##
## Call:
## lm(formula = Life.Exp ~ Population + Murder + HS.Grad + Frost,
##     data = statedata)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1.47095 -0.53464 -0.03701  0.57621  1.50683 
## 
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  7.103e+01  9.529e-01  74.542  < 2e-16 ***
## Population   5.014e-05  2.512e-05   1.996  0.05201 .  
## Murder      -3.001e-01  3.661e-02  -8.199 1.77e-10 ***
## HS.Grad      4.658e-02  1.483e-02   3.142  0.00297 ** 
## Frost       -5.943e-03  2.421e-03  -2.455  0.01802 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.7197 on 45 degrees of freedom
## Multiple R-squared:  0.736,  Adjusted R-squared:  0.7126 
## F-statistic: 31.37 on 4 and 45 DF,  p-value: 1.696e-12

# remove Population
g4 = update( g3, . ~ . - Population )
summary( g4 )
## 
## Call:
## lm(formula = Life.Exp ~ Murder + HS.Grad + Frost, data = statedata)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -1.5015 -0.5391  0.1014  0.5921  1.2268 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 71.036379   0.983262  72.246  < 2e-16 ***
## Murder      -0.283065   0.036731  -7.706 8.04e-10 ***
## HS.Grad      0.049949   0.015201   3.286  0.00195 ** 
## Frost       -0.006912   0.002447  -2.824  0.00699 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.7427 on 46 degrees of freedom
## Multiple R-squared:  0.7127, Adjusted R-squared:  0.6939 
## F-statistic: 38.03 on 3 and 46 DF,  p-value: 1.634e-12
```

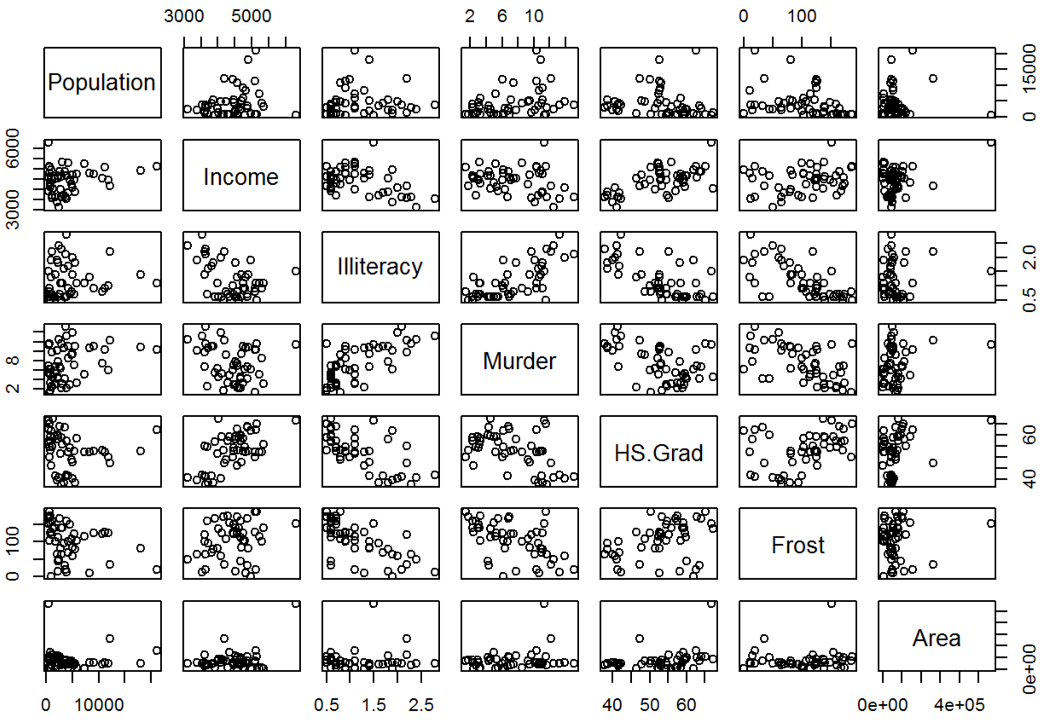
The final removal of variable *Population* is a close call. We may want to consider including this variable if interpretation is aided. Notice that the R^2 for the full model of 0.736 is reduced only slightly to 0.713 in the final model.

Thus the removal of four predictors causes only a minor reduction in fit.

```
X = statedata [ , -4 ] #not considering the response variable
cor( X )
##             Population     Income  Illiteracy     Murder      HS.Grad
## Population  1.00000000  0.2082276  0.10762237  0.3436428 -0.09848975
## Income      0.20822756  1.0000000 -0.43707519 -0.2300776  0.61993232
## Illiteracy  0.10762237 -0.4370752  1.00000000  0.7029752 -0.65718861
## Murder      0.34364275 -0.2300776  0.70297520  1.0000000 -0.48797102
## HS.Grad    -0.09848975  0.6199323 -0.65718861 -0.4879710  1.00000000
## Frost      -0.33215245  0.2262822 -0.67194697 -0.5388834  0.36677970
## Area        0.02254384  0.3633154  0.07726113  0.2283902  0.33354187
##                 Frost       Area
## Population -0.3321525 0.02254384
## Income      0.2262822 0.36331544
## Illiteracy -0.6719470 0.07726113
## Murder     -0.5388834 0.22839021
## HS.Grad     0.3667797 0.33354187
## Frost       1.0000000 0.05922910
## Area        0.0592291 1.00000000
# Note the high positive correlation between Murder and Illiteracy!

heatmap( cor( X ), Rowv = NA, Colv = NA, symm = TRUE, keep.dendro = F)
```




```
#image( as.matrix( cor( X ) ), main = 'Correlation of X' )


pairs( X )
```



Mind spurious correlations! `http://www.tylervigen.com/spurious-correlations` `http://guessthecorrelation.com`

Automatic backward selection method

At each step we remove the predictor with the largest p-value over 0.05:

```
g = lm( Life.Exp ~ ., data = statedata )

help( step )

# We can specify either backward or forward (or even both of them)
step( g, direction = "both" ) #it goes backward
## Start:  AIC=-22.18
## Life.Exp ~ Population + Income + Illiteracy + Murder + HS.Grad +
##     Frost + Area
##
##              Df Sum of Sq    RSS     AIC
## - Area        1    0.0011 23.298 -24.182
## - Income      1    0.0044 23.302 -24.175
## - Illiteracy  1    0.0047 23.302 -24.174
## <none>                    23.297 -22.185
## - Population  1    1.7472 25.044 -20.569
## - Frost       1    1.8466 25.144 -20.371
## - HS.Grad     1    2.4413 25.738 -19.202
## - Murder      1   23.1411 46.438  10.305
##
## Step:  AIC=-24.18
## Life.Exp ~ Population + Income + Illiteracy + Murder + HS.Grad +
##     Frost
##
##              Df Sum of Sq    RSS     AIC
## - Illiteracy  1    0.0038 23.302 -26.174
## - Income      1    0.0059 23.304 -26.170
## <none>                    23.298 -24.182
## - Population  1    1.7599 25.058 -22.541
## + Area        1    0.0011 23.297 -22.185
## - Frost       1    2.0488 25.347 -21.968
## - HS.Grad     1    2.9804 26.279 -20.163
## - Murder      1   26.2721 49.570  11.569
##
## Step:  AIC=-26.17
## Life.Exp ~ Population + Income + Murder + HS.Grad + Frost
##
##              Df Sum of Sq    RSS     AIC
## - Income      1     0.006 23.308 -28.161
## <none>                    23.302 -26.174
## - Population  1     1.887 25.189 -24.280
## + Illiteracy  1     0.004 23.298 -24.182
## + Area        1     0.000 23.302 -24.174
## - Frost       1     3.037 26.339 -22.048
## - HS.Grad     1     3.495 26.797 -21.187
## - Murder      1    34.739 58.041  17.456
##
## Step:  AIC=-28.16
## Life.Exp ~ Population + Murder + HS.Grad + Frost
##
##              Df Sum of Sq    RSS     AIC
## <none>                    23.308 -28.161
## + Income      1     0.006 23.302 -26.174
## + Illiteracy  1     0.004 23.304 -26.170
## + Area        1     0.001 23.307 -26.163
## - Population  1     2.064 25.372 -25.920
## - Frost       1     3.122 26.430 -23.877
## - HS.Grad     1     5.112 28.420 -20.246
## - Murder      1    34.816 58.124  15.528
##
## Call:
## lm(formula = Life.Exp ~ Population + Murder + HS.Grad + Frost,
##     data = statedata)
##
## Coefficients:
## (Intercept)   Population       Murder      HS.Grad        Frost
##   7.103e+01    5.014e-05   -3.001e-01    4.658e-02   -5.943e-03
```

```
#AIC criterion is the default

AIC( g1 )
## [1] 119.7116
AIC( g2 )
## [1] 117.7196
AIC( g3 )
## [1] 115.7326
AIC( g4 )
## [1] 117.9743
```

According to an automatic selection on AIC, the best model is $g3$: Population + Murder + HS.Grad + Frost.

Criterion based procedures

1. AIC & BIC;
2. R^2 adjusted;
3. Mallow's Cp;
4. PRESS.[shown in class, not in lab]

1. AIC & BIC

$$AIC = -2 \cdot \log(\text{ likelihood }) + 2k$$

where k is the number of parameter in the model.

$$BIC = -2 \cdot \log(\text{ likelihood }) + k \cdot \log(\text{n})$$

```
g = lm( Life.Exp ~ ., data = statedata )

AIC( g )
## [1] 121.7092
BIC( g )
## [1] 138.9174

g_AIC_back = step( g, direction = "backward" ) #k=2 is the default for pure AIC
## Start:  AIC=-22.18
## Life.Exp ~ Population + Income + Illiteracy + Murder + HS.Grad +
##     Frost + Area
##
##              Df Sum of Sq    RSS     AIC
## - Area        1    0.0011 23.298 -24.182
## - Income      1    0.0044 23.302 -24.175
## - Illiteracy  1    0.0047 23.302 -24.174
## <none>                    23.297 -22.185
## - Population  1    1.7472 25.044 -20.569
## - Frost       1    1.8466 25.144 -20.371
## - HS.Grad     1    2.4413 25.738 -19.202
## - Murder      1   23.1411 46.438  10.305
##
## Step:  AIC=-24.18
## Life.Exp ~ Population + Income + Illiteracy + Murder + HS.Grad +
##     Frost
##
##              Df Sum of Sq    RSS     AIC
## - Illiteracy  1    0.0038 23.302 -26.174
## - Income      1    0.0059 23.304 -26.170
## <none>                    23.298 -24.182
## - Population  1    1.7599 25.058 -22.541
## - Frost       1    2.0488 25.347 -21.968
## - HS.Grad     1    2.9804 26.279 -20.163
## - Murder      1   26.2721 49.570  11.569
##
## Step:  AIC=-26.17
## Life.Exp ~ Population + Income + Murder + HS.Grad + Frost
##
##              Df Sum of Sq    RSS     AIC
## - Income      1     0.006 23.308 -28.161
## <none>                    23.302 -26.174
## - Population  1     1.887 25.189 -24.280
## - Frost       1     3.037 26.339 -22.048
## - HS.Grad     1     3.495 26.797 -21.187
## - Murder      1    34.739 58.041  17.456
##
## Step:  AIC=-28.16
## Life.Exp ~ Population + Murder + HS.Grad + Frost
##
##              Df Sum of Sq    RSS     AIC
## <none>                    23.308 -28.161
## - Population  1     2.064 25.372 -25.920
## - Frost       1     3.122 26.430 -23.877
## - HS.Grad     1     5.112 28.420 -20.246
## - Murder      1    34.816 58.124  15.528
g_BIC_back = step( g, direction = "backward", k = log(n) )
## Start:  AIC=-6.89
## Life.Exp ~ Population + Income + Illiteracy + Murder + HS.Grad +
##     Frost + Area
##
##              Df Sum of Sq    RSS      AIC
## - Area        1    0.0011 23.298 -10.7981
## - Income      1    0.0044 23.302 -10.7910
## - Illiteracy  1    0.0047 23.302 -10.7903
## - Population  1    1.7472 25.044  -7.1846
## - Frost       1    1.8466 25.144  -6.9866
## <none>                    23.297  -6.8884
```

```
## - HS.Grad     1    2.4413 25.738  -5.8178
## - Murder      1   23.1411 46.438  23.6891
##
## Step:  AIC=-10.8
## Life.Exp ~ Population + Income + Illiteracy + Murder + HS.Grad +
##     Frost
##
##              Df Sum of Sq    RSS      AIC
## - Illiteracy  1    0.0038 23.302 -14.7021
## - Income      1    0.0059 23.304 -14.6975
## - Population  1    1.7599 25.058 -11.0691
## <none>                    23.298 -10.7981
## - Frost       1    2.0488 25.347 -10.4960
## - HS.Grad     1    2.9804 26.279  -8.6912
## - Murder      1   26.2721 49.570  23.0406
##
## Step:  AIC=-14.7
## Life.Exp ~ Population + Income + Murder + HS.Grad + Frost
##
##              Df Sum of Sq    RSS     AIC
## - Income      1     0.006 23.308 -18.601
## - Population  1     1.887 25.189 -14.720
## <none>                    23.302 -14.702
## - Frost       1     3.037 26.339 -12.488
## - HS.Grad     1     3.495 26.797 -11.627
## - Murder      1    34.739 58.041  27.017
##
## Step:  AIC=-18.6
## Life.Exp ~ Population + Murder + HS.Grad + Frost
##
##              Df Sum of Sq    RSS     AIC
## <none>                    23.308 -18.601
## - Population  1     2.064 25.372 -18.271
## - Frost       1     3.122 26.430 -16.228
## - HS.Grad     1     5.112 28.420 -12.598
## - Murder      1    34.816 58.124  23.176

BIC(g1)
## [1] 135.0077
BIC(g2)
## [1] 131.1038
BIC(g3)
## [1] 127.2048
BIC(g4)
## [1] 127.5344
```

The best model according to BIC is still $g3$.

2. R^2 adjusted

```
help( leaps )

# solo matrice dei predittori senza colonna di 1
x = model.matrix( g ) [ , -1 ]
y = statedata$Life

adjr = leaps( x, y, method = "adjr2" )
adjr
## $which
##       1     2     3     4     5     6     7
## 1 FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## 1 FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## 1 FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## 1 FALSE  TRUE FALSE FALSE FALSE FALSE FALSE
## 1 FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
## 1 FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
## 1  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
## 2 FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE
## 2  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE
## 2 FALSE FALSE FALSE  TRUE FALSE  TRUE FALSE
## 2 FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE
## 2 FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE
## 2 FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
## 2 FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
## 2 FALSE FALSE  TRUE FALSE  TRUE FALSE FALSE
## 2 FALSE FALSE  TRUE FALSE FALSE  TRUE FALSE
## 2 FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
## 3 FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE
## 3  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE
## 3 FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
## 3  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE
## 3 FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE
## 3 FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE
## 3 FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE
## 3  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE
## 3 FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
## 3  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE
## 4  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE
## 4 FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
## 4 FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
## 4 FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
## 4  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE
## 4  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE
## 4  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE
## 4  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE
## 4  TRUE FALSE FALSE  TRUE FALSE  TRUE  TRUE
## 4 FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
## 5  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE
## 5  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
## 5  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
## 5 FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
## 5 FALSE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## 5  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
## 5 FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
## 5  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
## 5  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE
## 5  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
## 6  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
## 6  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
## 6  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
## 6 FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## 6  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
## 6  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
## 6  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## 7  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##
## $label
```

```
## [1] "(Intercept)" "1"           "2"           "3"           "4"
## [6] "5"           "6"           "7"
##
## $size
##  [1] 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5
## [36] 5 5 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 8
##
## $adjr2
##  [1]  0.601589257  0.332687649  0.325204369  0.097352314  0.049277135
##  [6] -0.009073186 -0.016105785  0.648499092  0.640531108  0.630123171
## [11]  0.621505415  0.598658042  0.596338472  0.417498412  0.388611537
## [16]  0.352293622  0.327377601  0.693922972  0.681157120  0.660546563
## [21]  0.657142928  0.652680094  0.646188613  0.644205869  0.644065363
## [26]  0.642090700  0.640621243  0.712569018  0.689369653  0.689240309
## [31]  0.687438534  0.686844403  0.675198232  0.675188701  0.669545043
## [36]  0.665612898  0.664714272  0.706112903  0.706085962  0.706046460
## [41]  0.683964374  0.683039206  0.682778094  0.682185571  0.680321045
## [46]  0.671976112  0.668571103  0.699326842  0.699283899  0.699279833
## [51]  0.676792721  0.675509995  0.667835310  0.400695811  0.692182314

bestmodel_adjr2_ind = which.max( adjr$adjr2 )
g$coef[ which( adjr$which[ bestmodel_adjr2_ind, ] ) + 1 ]
##    Population        Murder       HS.Grad         Frost
##  5.180036e-05 -3.011232e-01  4.892948e-02 -5.735001e-03

help( maxadjr )
maxadjr( adjr, 5 )
##     1,4,5,6   1,2,4,5,6   1,3,4,5,6   1,4,5,6,7 1,2,3,4,5,6
##       0.713       0.706       0.706       0.706       0.699
```

We see that also in this case $g3$, the model with Population + Murder + HS graduation + Frost, is the best one, since it has the largest R^2_{adj} (71.26%).

```
# Other possibilities
R2 = leaps( x, y, method = "r2" )

bestmodel_R2_ind = which.max( R2$r2 )
R2$which[ bestmodel_R2_ind, ]
##    1    2    3    4    5    6    7
## TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

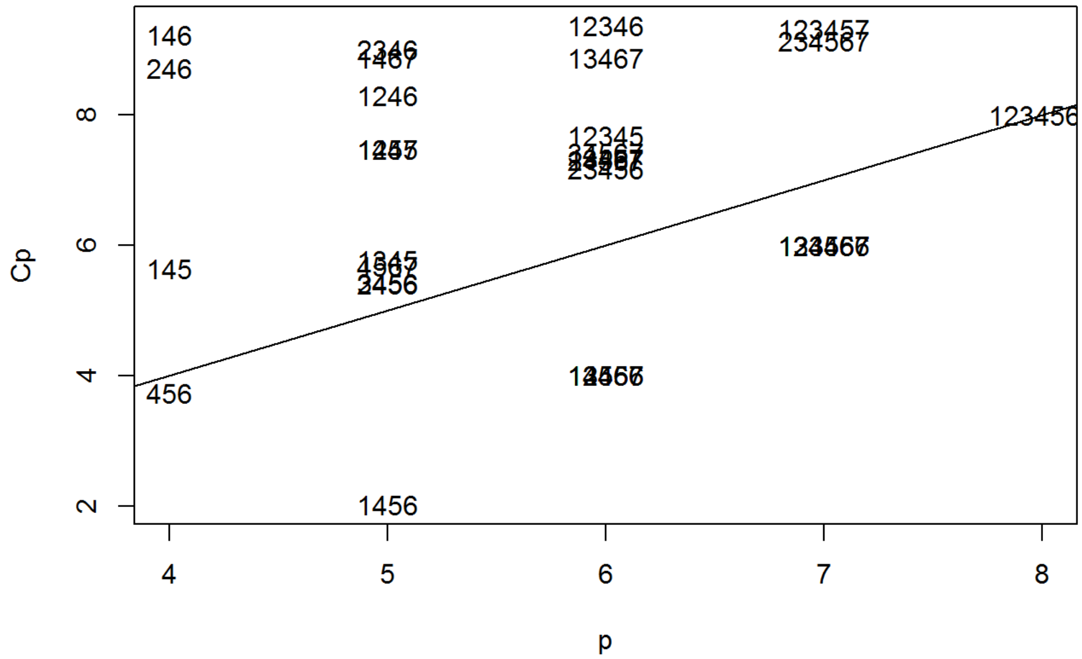
According to R^2, the best model is the complete one.

REMARK Remember that Variable selection methods are sensitive to influential points.

3. Mallow's Cp

Rule of thumb The best model according to Cp, is the one that leads to Cp close to p. if we are uncertain, we should choose the simplest model.

```
g_Cp_model = leaps(  y = statedata[ , 4 ], x = statedata[ , -  4 ], method = 'Cp' )
# g = leaps( x, y, method = "Cp" ) # Mallow's Statistics
Cpplot( g_Cp_model )
```




```
g_Cp_coef = which( g_Cp_model$which[ which.min( g_Cp_model$Cp ) , ] )
g$coefficients[ g_Cp_coef + 1 ]
##    Population        Murder       HS.Grad         Frost
##  5.180036e-05 -3.011232e-01  4.892948e-02 -5.735001e-03

#il modello migliore Ã¨ 456, cioÃ¨ g4 (Cp ottimo ~ p).
```

The models are denoted by indices for the predictors. The best model according to Cp is the “456” model i.e. the Murder, HS graduation and Frost model.

---

7. Prediction

We want to establish the relation between the height of tomatoes plants and the average weight of the picked tomatoes [g].

The data are as follows:

```
peso    = c( 60, 65, 72, 74, 77, 81, 85, 90 )
altezza = c( 160, 162, 180, 175, 186, 172, 177, 184 )
```

7.a Fit the complete model and analyze it.

```
mod = lm( peso ~ altezza )
summary( mod )
##
## Call:
## lm(formula = peso ~ altezza)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.860 -4.908 -1.244  7.097  7.518
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -62.8299    49.2149  -1.277   0.2489
## altezza       0.7927     0.2817   2.814   0.0306 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.081 on 6 degrees of freedom
## Multiple R-squared:  0.569,  Adjusted R-squared:  0.4972
## F-statistic: 7.921 on 1 and 6 DF,  p-value: 0.03058
```

7.b Compute the **Confidence Interval** for the prediction of the *average outcome*.

solution

We define a grid of values (in the range of the data, in order to have reliable prediction).

We compute the predicted values:

$$\hat{y}_{new} = x_{new}\hat{\beta}$$

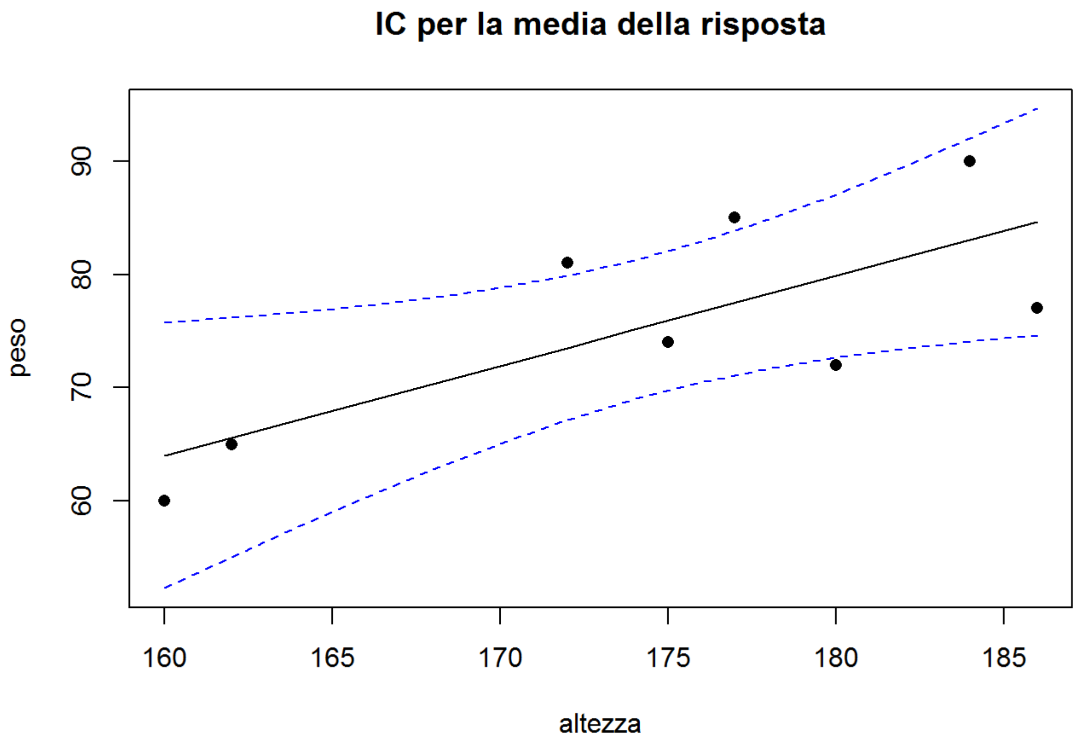
and the related standard errors:

$$se(\mathbb{E}[y_{new}]) = \hat{S} \cdot \sqrt{x_{new}^T (X^T X)^{-1} x_{new}}$$

REMARK The second call of `predict` , x_{new}, must be a data.frame with the columns with the same names of the original predictors in the model.

```
grid = seq( min( altezza ), max( altezza ), 2 )

#automatically
y.pred = predict( mod, data.frame( altezza = grid ), interval = "confidence", se = T )

names( y.pred )
## [1] "fit"            "se.fit"         "df"             "residual.scale"

y.pred$fit[ ,1 ] # predicted values $\hat{y}_{new}$.
##        1        2        3        4        5        6        7        8 
## 64.00554 65.59098 67.17642 68.76187 70.34731 71.93275 73.51820 75.10364 
##        9       10       11       12       13       14 
## 76.68908 78.27453 79.85997 81.44541 83.03085 84.61630
y.pred$fit[ ,2 ] # LB confidence interval for $y_{new}$.
##        1        2        3        4        5        6        7        8 
## 52.28376 55.01985 57.69498 60.28561 62.75808 65.06647 67.15458 68.96805 
##        9       10       11       12       13       14 
## 70.47655 71.69078 72.65607 73.43001 74.06439 74.59890
y.pred$fit[ ,3 ] # UB confidence interval for $y_{new}$.
##        1        2        3        4        5        6        7        8 
## 75.72731 76.16211 76.65787 77.23813 77.93654 78.79904 79.88181 81.23923 
##        9       10       11       12       13       14 
## 82.90161 84.85827 87.06386 89.46081 91.99731 94.63370


# manually
ndata = cbind( rep( 1, length( grid ) ), grid )
y.pred_fit = ndata %*% mod$coefficients
y.pred_fit
##           [,1]
##  [1,] 64.00554
##  [2,] 65.59098
##  [3,] 67.17642
##  [4,] 68.76187
##  [5,] 70.34731
##  [6,] 71.93275
##  [7,] 73.51820
##  [8,] 75.10364
##  [9,] 76.68908
## [10,] 78.27453
## [11,] 79.85997
## [12,] 81.44541
## [13,] 83.03085
## [14,] 84.61630

#standard error
y.pred$se
##        1        2        3        4        5        6        7        8 
## 4.790436 4.320193 3.874862 3.464065 3.101553 2.806103 2.600672 2.507483 
##        9       10       11       12       13       14 
## 2.538926 2.690635 2.944076 3.275721 3.664398 4.093895
#manually
y.pred_se = rep( 0, 14 )
X = model.matrix( mod )
for( i in 1:14 )
{
 y.pred_se[ i ] = summary( mod )$sigma * sqrt( t( ndata[i,] ) %*% solve( t(X) %*% X ) %*% ndata[i,] )
}
y.pred_se
##  [1] 4.790436 4.320193 3.874862 3.464065 3.101553 2.806103 2.600672
##  [8] 2.507483 2.538926 2.690635 2.944076 3.275721 3.664398 4.093895

# n - p = 8 - 2 = 6
y.pred$df
## [1] 6

tc    = qt( 0.975, length( altezza ) - 2 )
```

```
y     = y.pred$fit[ ,1 ]
y.sup = y.pred$fit[ ,1 ] + tc * y.pred$se
y.inf = y.pred$fit[ ,1 ] - tc * y.pred$se

IC = cbind( y, y.inf, y.sup )

IC
##           y    y.inf    y.sup
## 1  64.00554 52.28376 75.72731
## 2  65.59098 55.01985 76.16211
## 3  67.17642 57.69498 76.65787
## 4  68.76187 60.28561 77.23813
## 5  70.34731 62.75808 77.93654
## 6  71.93275 65.06647 78.79904
## 7  73.51820 67.15458 79.88181
## 8  75.10364 68.96805 81.23923
## 9  76.68908 70.47655 82.90161
## 10 78.27453 71.69078 84.85827
## 11 79.85997 72.65607 87.06386
## 12 81.44541 73.43001 89.46081
## 13 83.03085 74.06439 91.99731
## 14 84.61630 74.59890 94.63370
y.pred$fit
##         fit      lwr      upr
## 1  64.00554 52.28376 75.72731
## 2  65.59098 55.01985 76.16211
## 3  67.17642 57.69498 76.65787
## 4  68.76187 60.28561 77.23813
## 5  70.34731 62.75808 77.93654
## 6  71.93275 65.06647 78.79904
## 7  73.51820 67.15458 79.88181
## 8  75.10364 68.96805 81.23923
## 9  76.68908 70.47655 82.90161
## 10 78.27453 71.69078 84.85827
## 11 79.85997 72.65607 87.06386
## 12 81.44541 73.43001 89.46081
## 13 83.03085 74.06439 91.99731
## 14 84.61630 74.59890 94.63370

matplot( grid, cbind( y, y.inf, y.sup ), lty = c( 1, 2, 2 ), col = c( 1, 'blue', 'blue' ), type = "l", xlab =
"altezza", ylab = "peso", main = 'IC per la media della risposta' )
points( altezza, peso, col = "black", pch = 16 )
```



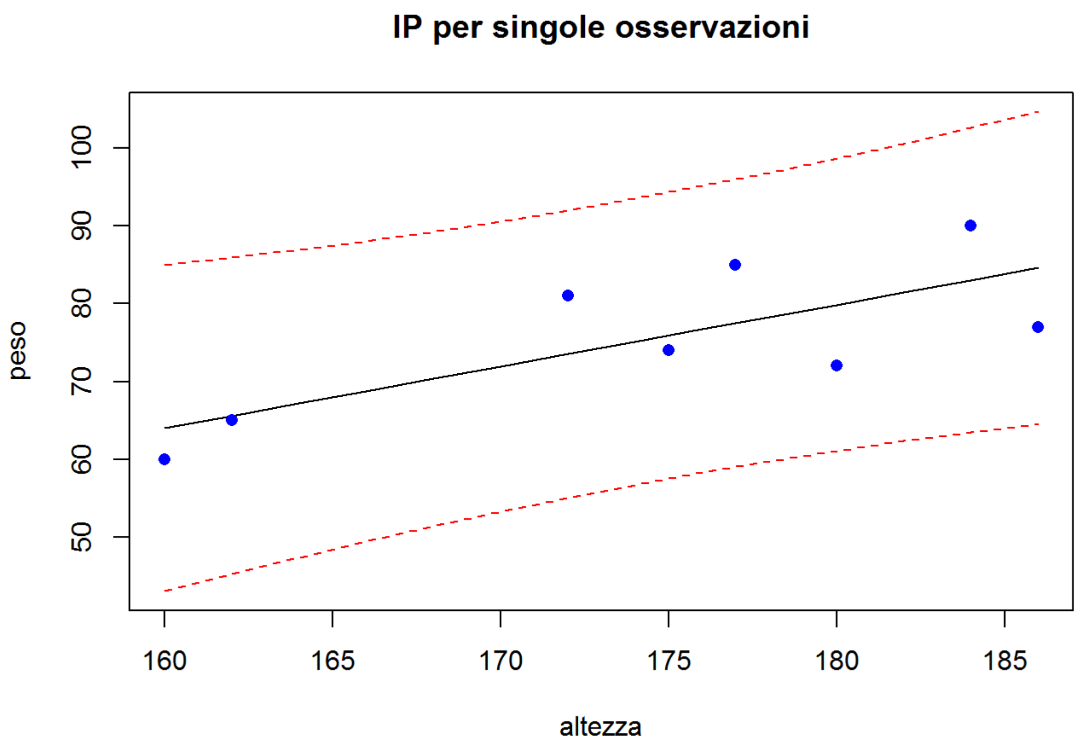
7.c

Compute the *Prediction Interval* for the one new observation. In this case the standard errors are:

$$se(y_{new}) = \hat{S} \cdot \sqrt{1 + x_{new}^T (X^T X)^{-1} x_{new}}$$

```
y.pred2 = predict( mod, data.frame( altezza = grid ), interval = "prediction", se = T )
# fornisce direttamente gli estremi inf e sup, che prima abbiamo costruito a mano (in un altro caso)

y.pred2$fit[ ,1 ] # predicted values $\hat{y}_{new}$.
##        1        2        3        4        5        6        7        8
## 64.00554 65.59098 67.17642 68.76187 70.34731 71.93275 73.51820 75.10364
##        9       10       11       12       13       14
## 76.68908 78.27453 79.85997 81.44541 83.03085 84.61630
y.pred2$fit[ ,2 ] # LB prediction interval for $y_{new}$.
##        1        2        3        4        5        6        7        8
## 43.08632 45.29412 47.42518 49.47299 51.43144 53.29517 55.05989 56.72270
##        9       10       11       12       13       14
## 58.28231 59.73917 61.09538 62.35457 63.52160 64.60225
y.pred2$fit[ ,3 ] # UB prediction interval for $y_{new}$.
##         1         2         3         4         5         6         7
##  84.92475  85.88784  86.92766  88.05074  89.26318  90.57034  91.97651
##         8         9        10        11        12        13        14
##  93.48458  95.09585  96.80988  98.62456 100.53625 102.54011 104.63034


#manually
ndata = cbind( rep( 1, length( grid ) ), grid )
y.pred_fit = ndata %*% mod$coefficients
y.pred_fit
##           [,1]
##  [1,] 64.00554
##  [2,] 65.59098
##  [3,] 67.17642
##  [4,] 68.76187
##  [5,] 70.34731
##  [6,] 71.93275
##  [7,] 73.51820
##  [8,] 75.10364
##  [9,] 76.68908
## [10,] 78.27453
## [11,] 79.85997
## [12,] 81.44541
## [13,] 83.03085
## [14,] 84.61630

# standard error
y.pred2$se.fit
##        1        2        3        4        5        6        7        8
## 4.790436 4.320193 3.874862 3.464065 3.101553 2.806103 2.600672 2.507483
##        9       10       11       12       13       14
## 2.538926 2.690635 2.944076 3.275721 3.664398 4.093895
#manually
y.pred2_se = rep( 0, 14 )

for( i in 1:14 )
{
 y.pred2_se[ i ] = summary( mod )$sigma * sqrt(  1 + t( ndata[i,] ) %*% solve( t(X) %*% X ) %*% ndata[i,] )
}
y.pred2_se
##  [1] 8.549232 8.294887 8.071905 7.882946 7.730506 7.616778 7.543512
##  [8] 7.511894 7.522448 7.574999 7.668681 7.802015 7.973011 8.179307

#In this case y.pred2_se != y.pred2$se.fit

tc    = qt( 0.975, length( altezza ) - 2 )
y     = y.pred2$fit[,1]
y.sup = y.pred2$fit[,1] + tc * y.pred2_se
y.inf = y.pred2$fit[,1] - tc * y.pred2_se

IP = cbind( y, y.inf, y.sup )
y.pred2$fit
##         fit      lwr       upr
```

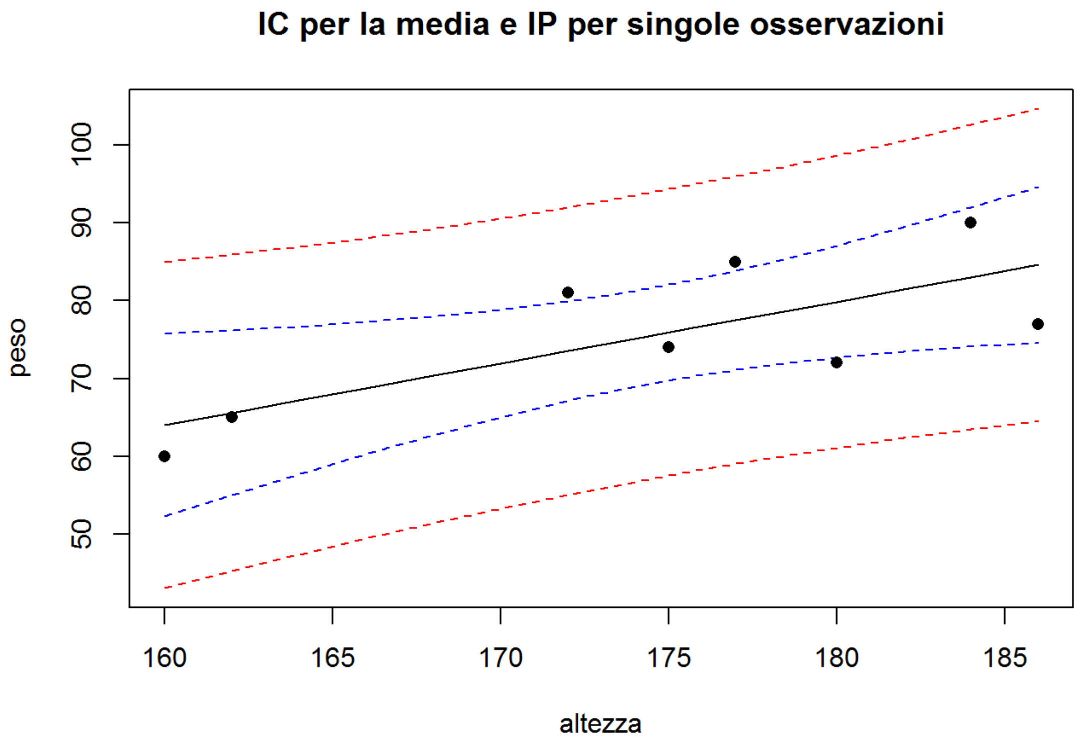
```
## 1  64.00554 43.08632  84.92475
## 2  65.59098 45.29412  85.88784
## 3  67.17642 47.42518  86.92766
## 4  68.76187 49.47299  88.05074
## 5  70.34731 51.43144  89.26318
## 6  71.93275 53.29517  90.57034
## 7  73.51820 55.05989  91.97651
## 8  75.10364 56.72270  93.48458
## 9  76.68908 58.28231  95.09585
## 10 78.27453 59.73917  96.80988
## 11 79.85997 61.09538  98.62456
## 12 81.44541 62.35457 100.53625
## 13 83.03085 63.52160 102.54011
## 14 84.61630 64.60225 104.63034

matplot( grid, y.pred2$fit, lty = c( 1, 2, 2 ), col = c( 1, 2, 2 ), type = "l",
         xlab = "altezza", ylab = "peso", main = 'IP per singole osservazioni' )
points( altezza, peso, col = "blue", pch = 16 )
```



7.d

Compare the Intervals obtained at **7.b** and **7.c**.

```
matplot( grid, y.pred2$fit, lty = c( 1, 2, 2 ), col = c( 1, 2, 2 ), type = "l", xlab = "altezza", ylab = "pes
o", main = "IC per la media e IP per singole osservazioni" )
lines( grid, y.pred$fit[ , 2 ] , col = "blue", lty = 2, xlab = "altezza", ylab = "peso" )
lines( grid, y.pred$fit[ , 3 ] , col = "blue", lty = 2, xlab = "altezza", ylab = "peso" )
points( altezza, peso, col = "black", pch = 16 )
```



According to theory, the prediction interval is broader that the confidence interval (see the standard errors) and all the points are inside the prediction interval, while only few of them are inside the confidence interval.